V. Les analyses statistiques avec R

V.1 Cas pratique *Prix d'un appartement* La régression `lm()`

Nous reprenons ici l'exemple du livre de Michel Tenenhaus, « Méthodes statistiques en gestion » sur le prix des appartements.

Les données

Nous avons relevé quelques annonces d'appartements à vendre dans la presse parisienne journal. Les données sont reproduites dans le tableau 8.

1. CENSIER, bas de R. Mouffetard, pied-à-terre, 28m², tt confort. Visite vendredi, samedi, dim. 650.000 F à discuter. Facilités	2. CONTRESCARPE, imm. Ancien, pierre de taille, beau duplex caractère, 50m², poutres, refait neuf, 1.400.000 F
3. R. St-Simon, en pleine verdure, calme, plein soleil, Superbe appt 4p., 106m², cuis. aménagée, s. de bains moderne, chff. cent. Parfait état. Px 3.250.000 à discuter. Agence s'abstenir. Direct propriétaire.	4. RAPP 7P., 196m² standing, 9 fenêtres plein soleil, 4.000.000 F.
5. R. St André-des-Arts, beau liv + chbre, imm. XVIIIe siècle, 55m², 1.350.000 F.	6. 5e PRES QUAIS, 7 pces, 190m² caractère, standing, 3.950.000 F
7. GOBELINS, Beau 5p., 110m², gd cft, soleil, 250.000 F	8. GOBELINS, et. elevé, calme, asc., 2 pièces,60m², 1.600.000 F
9. CENSIER, très grand studio + entrée 48m², tt cft, ensoleillé, calme, bel imm., 1.250.000 F	10. PANTHEON, 7e étage, ascenseur, grand studio 35m² + terrasse. Vue. 1.250.000F.
11. RUE MADAME, 3P. + Serv., 86m², 1.750.000 F.	12. RUE DE SEINE, 3P., tt cft, 65m², calme, soleil, 1.500.000 F.
13. PANTHEON, bel imm., verdure, magnifique studio 32m², caractère, 775.000 F.	14. SEVRES BAB, 1er ét., 2P., gde cuis., bns 52m², état neuf, 1.225.000 F.
15. MONTPARNASSE, Part. vend atelier d'artiste 40m², duplex, vue imprenable, tout confort, Prix 1.000.000 F.	16. RUE D'ASSAS, imm. gd standing, bel appart 260m², triple récept. + 5 ch., tt cft (travaux) 2 park., 2 ch. Serv., Prix 7.500.000 F à déb.
17. BD St-GERMAIN, 4P., 70m², à amén., 4e ét., 1.625.000 F.	18. ILE St-LOUIS, Lux. appt., 117m², en duplex, gde récept., gde chambre, 2 sdb, Terras., parf. et., décor tr. bon goût, 4.750.000 F.
19. JUSSIEU, Charme, gd 3pces, 90m², 1.890.000 F.	20. QUARTIER LATIN, 30m² à aménager, prix 390.000 F.
21. MONTPARNASSE, Imm. p.d.t., 4-5 P., 105m², bon état, 1.875.000 F.	22. RUE MAZARINE, 4e ét., sans ascens., 52m² à rénover. Prix total 1.000.000 F.
23. CENSIER, Bel imm., 4P. 80m², tt cft, petits travaux, 1.350.000 F.	24. ASSAS LUXEMBOURG, 3P. 60m² s/arbres, imm. caractère, 1.475.000 F.
25. SUR JARDINS OBSERVATOIRE, 140m², grand charme, 4.950.000 F.	26. RUE DE SAVOIE, 4e ét., Studio 20m², dche, 425.000 F. crédit possible.
27. PRES LUXEMBOURG, Bel imm., pierre de taille, Appartement 100m², salon, sal. à manger, 2 chbres, office, cuis., bains, chf. cent., asc., prix 2.475.000 F.	28. Mo GOBELINS, studio, cuis., s. de bains, 28m², calme. Prix 425.000 F.

Tab 8. Description des 28 appartements à vendre

Pour avoir une vision d'ensemble des données du tabeau 8, nous avons construit le tableau 9 en associant à chaque appartement son prix, sa surface et son prix au m².

Rue	Surf	Prix	Prix au m²	Rue	Surf	Prix	Prix au m²
1 censier	28	650	23.21	15 montparnasse	40	1000	25.00
2 contrescarpe	50	1400	28.00	16 rue d'assas	260	7500	28.85
3saint-simon	106	3250	30.66	17 St-germain	70	1625	23.21
4 Rapp	196	4000	20.41	18 île St-louis	117	4750	40.60
5 St-A.des arts	55	1340	24.36	19 jussieu	90	1890	21.00
6 près quais	190	3950	20.79	20 quartier-latin	30	390	13.00
7 gobelins	110	2500	22.73	21 montparnasse	105	1875	17.86
8 gobelins	60	1600	26.67	22 rue mazarine	52	1000	19.23

9.censier	48	1250	26.04	23 censier	80	1350	16.88
10 panthéon	35	1250	35.71	24 assas	60	1475	24.58
11 rue madame	86	1750	20.35	25 observatoire	140	4950	35.36
12 rue de seine	65	1500	23.08	26 rue de savoie	20	425	21.25
13 panthéon	32	775	24.22	27 Luxembourg	100	2475	24.75
14 S. -babylone	52	1225	23.56	28 gobelins	28	425	15.18

Tab 9 Surface, prix et prix au m² des 28 appartements Surf Surface, prix Prix en KF et prix au m² Prix au m² en KF

```
surface = c(28, 50, 106, 196, 55, 190, 110, 60, 48, 35, 86, 65, 32, 52, 40, 260,
70, 117, 90, 30, 105, 52, 80, 60, 140, 20, 100, 28)

prix = c(650, 1400, 3250, 4000, 1340, 3950, 2500, 1600, 1250, 1250, 1750, 1500,
775, 1225, 1000, 7500, 1625, 4750, 1890, 390, 1875, 1000, 1350, 1475, 4950, 425,
2475, 425)

prix_au_m2 = c(23.21, 28, 30.66, 20.41, 24.36, 20.79, 22.73, 26.67, 26.04, 35.71,
20.35, 23.08, 24.22, 23.56, 25, 28.85, 23.21, 40.6, 21, 13, 17.86, 19.23, 16.88,
24.58, 35.36, 21.25, 24.75, 15.18)

appart = data.frame(surface, prix, prix_au_m2)

> appart
obs     surface       prix     prix_au_m2     obs     surface       prix     prix_au_m2
1       28            650           23.21     15      40            1000          25.00
2       50            1400          28.00     16      260           7500          28.85
3       106           3250          30.66     17      70            1625          23.21
4       196           4000          20.41     18      117           4750          40.60
5       55            1340          24.36     19      90            1890          21.00
6       190           3950          20.79     20      30            390           13.00
7       110           2500          22.73     21      105           1875          17.86
8       60            1600          26.67     22      52            1000          19.23
9       48            1250          26.04     23      80            1350          16.88
10      35            1250          35.71     24      60            1475          24.58
11      86            1750          20.35     25      140           4950          35.36
12      65            1500          23.08     26      20            425           21.25
13      32            775           24.22     27      100           2475          24.75
14      52            1225          23.56     28      28            425           15.18
```

On souhaite faire la régression linéaire du prix en fonction de la surface. La régression linéaire recherche l'hyperplan qui minimise la somme des distances des points à leur projection sur l'hyperplan. Autrement dit, la régression linéaire recherche l'hyperplan qui passe le « mieux possible » au milieu du nuage de points.

Ici, on recherche la droite qui passe le mieux possible au milieu du nuage de points définit par les 28 appartements de coordonnées (surface, prix). La fonction `lm()` permet le calcul de l'équation de cette droite.

```
appart.lm = lm(appart$prix~appart$surface)

appart.lm
Call:
lm(formula = appart$prix ~ appart$surface)

Coefficients:
(Intercept)  appart$surface
-147.33      26.77
```

On peut accéder à un résultat plus détaillé de la régression grâce à la fonction `summary()`.

```
summary(appart.lm)

Call:
lm(formula = appart$prix ~ appart$surface)

Residuals:
      Min        1Q  Median       3Q      Max
-1098.771 -273.462 -2.142 119.772 1765.728
Coefficients:

               Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)    -147.33   206.23      -0.714   0.481
appart$surface 26.77     2.07        12.931   7.86e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 614.7 on 26 degrees of freedom
Multiple R-Squared: 0.8654, Adjusted R-squared: 0.8603
F-statistic: 167.2 on 1 and 26 DF,  p-value: 7.862e-13
```

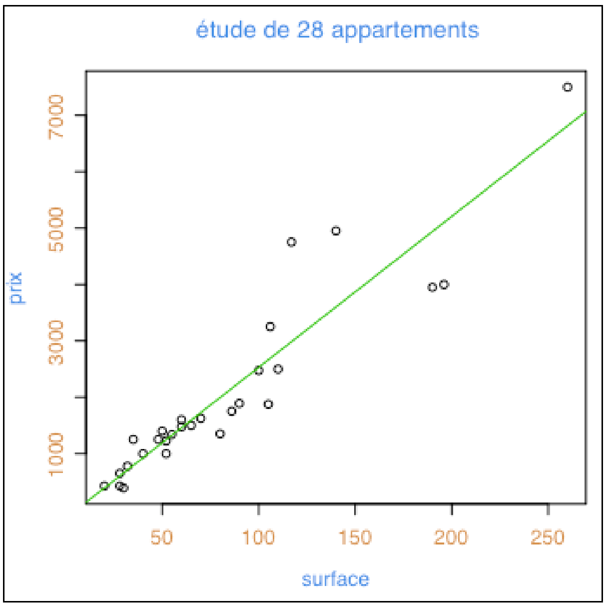
Remarque fondamentale Le résultat de fonction `lm()` i.e. appart.lm est un objet de mode list.

```
mode(appart.lm)
[1] "list"
```

On peut donc accéder à toutes les valeurs du résultat soit par l'indexation soit par le nom de la manière décrite aux paragraphes III.4.4.1.2 et III.4.4.1.3

On souhaite tracer la droite de régression de régression associée

```
plot(appart$surface, appart$prix,
     xlab = "surface", ylab = "prix",
     col.axis = "peru",
     col.lab = "royalblue",
     col.main = "royalblue",
     main = " étude de 28 appartements"
    )

abline(appart.lm, col = "green3")
```



Graphique 13. Régression sur les données d'appartement

On souhaite calculer les prévisions du modèle sur les 28 appartements `predict(lm(appart$prix ~ appart$surface))` ou de manière équivalente `predict(appart.lm)`

```
1           2           3           4           5           6
602.1140    1190.9620   2689.8478   5098.7714   1324.7910   4938.1765
7           8           9           10          11          12
2796.9110   1458.6201   1137.4303   789.4747    2154.5314   1592.4492
13          14          15          16          17          18
709.1772    1244.4936   923.3038    6811.7837   1726.2783   2984.2718
19          20          21          22          23          24
2261.5947   655.6456    2663.0819   1244.4936   1993.9365   1458.6201
25          26          27          28
3599.8856   387.9874    2529.2529   602.1140
```

On peut également construire des intervalles de prédiction grâce au paramètre `interval="prediction"` de la fonction `predict()`:

```
pred.w.plim = predict(lm(appart$prix~appart$surface),
                      as.data.frame(surface),
                      interval="prediction"
                     )
```

```
pred.w.plim
      fit         lwr          upr
1     602.1140    -704.37931   1908.607
2     1190.9620   -102.25810    2484.182
3     2689.8478   1400.01973   3979.676
4     5098.7714   3724.92790   6472.615
5     1324.7910   33.65992     2615.922
6     4938.1765   3573.11135   6303.242
7     2796.9110   1505.64221   4088.180
8     1458.6201   169.23027    2748.010
9     1137.4303   -156.72228   2431.583
10    789.4747    -512.07997   2091.029
11    2154.5314   868.54870    3440.514
12    1592.4492   304.45156    2880.447
13    709.1772    -594.41294   2012.767
14    1244.4936   -47.84928    2536.836
15    923.3038    -375.13012   2221.738
16    6811.7837   5320.13817   8303.429
17    1726.2783   439.32265    3013.234
18    2984.2718   1689.94682   4278.597
19    2261.5947   975.29226    3547.897
20    655.6456    -649.36919   1960.660
21    2663.0819   1373.57927   3952.585
22    1244.4936   -47.84928    2536.836
23    1993.9365   708.01110    3279.862
24    1458.6201   169.23027    2748.010
25    3599.8856   2290.78996   4908.981
26    387.9874    -924.95477   1700.930
27    2529.2529   1241.16735   3817.338
28    602.1140    -704.37931   1908.607
```

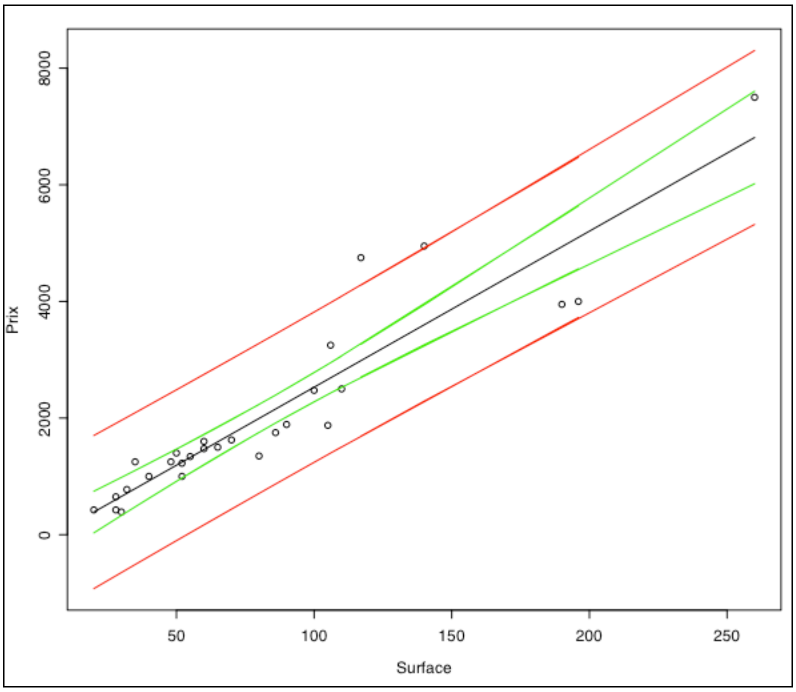
On peut également construire des intervalles de confiance grâce au paramètre `interval="confidence"` de la focntion predict :

```
pred.w.clim = predict(lm(appart$prix~appart$surface),
                      as.data.frame(surface),
```

```
                interval="confidence"
               )
```

On peut alors vouloir visualiser la bande de confiance et de prédiction:

```
ord<-order(appart$prix)
appart$prix <- appart$prix[ord]
appart$surface <- appart$surface [ord]
reg = lm(appart$prix~appart$surface)

# Intervalle de confiance pour la droite de régression
pred1 = predict(reg,interval="confidence")

# Intervalle de prédiction
pred2 = predict.lm(reg,interval="prediction")

# Trouver les valeurs minimales et maximales pour les intervalles
l.min = min(pred1[,2],pred2[, 2])
l.max = max(pred1[,3],pred2[, 3])

plot(appart$surface,appart$prix,ylim = c(l.min, l.max),
     xlab = "Surface", ylab = "Prix")
# Ligne qui trace le droit de régression
lines(appart$surface,pred1[, 1],lwd = 1)

# Lignes pour l'intervalle du droit de régression
lines(appart$surface, pred1[, 2], lwd = 1, col = "green")
lines(appart$surface, pred1[, 3], lwd = 1, col = "green")

# Lignes pour l'intervalle des observations
lines(appart$surface,pred2[, 2], lwd = 1, col = "red")
lines(appart$surface,pred2[, 3], lwd = 1, col = "red")
```

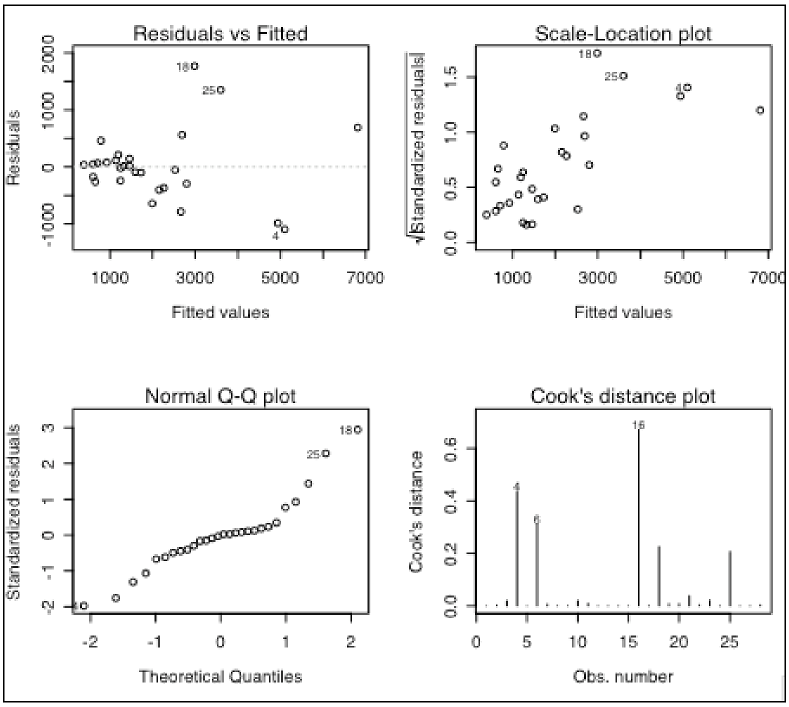


Si on souhaite tester le modèle sur de nouvelles données *Xnew*, on utilise la fonction `predict(appart.lm, Xnew)`

La fonction plot, une fonction générique

Les fonctions de R agissent en fonction des attributs des objets passés en argument. Les objets qui contiennent les résultats d'une analyse ont, quant à eux, un attribut particulier nommé la « classe » qui contient la signature de la fonction qui a fait l'analyse. Les fonctions qui serviront ensuite à extraire des informations de l'objet-résultat agiront spécifiquement en fonction de la classe de l'objet. Ces fonctions sont dites génériques.
Par exemple, une fonction souvent utilisée pour extraire des résultats d'analyse est la fonction `summary()`. Selon que l'objet qui est passé en argument est de la classe « lm » (modèle linéaire) ou « aov » (analyse de la variance), il est clair que les informations à afficher ne seront pas les mêmes. L'avantage des fonctions génériques est d'avoir une syntaxe unique pour toutes les analyses. La fonction `plot()` propose également divers sortie graphique en fonction de la classe de l'objet rentré en argument.

```
layout(matrix(1:4, 2, 2))
     plot(appart.lm)
```



Graphique 15. La fonction plot sur les résultats de l'analyse de lm

Le graphique 15 illustre l'utilisation de la fonction `plot()` sur le résultat de l'analyse de la fonction `lm()`.

V.2 Les iris de Fisher : Un exemple d'analyse discriminante `lda()`

Les Iris de Fisher constituent un jeu de données classique, utilisé dans un contexte de classification pour la première fois en 1936. Ces données comprennent quatre mesures distinctes de caractéristiques d'une population de trois variétés différentes d'iris: iris setosa, iris versicolor et iris virginica.

Setosa

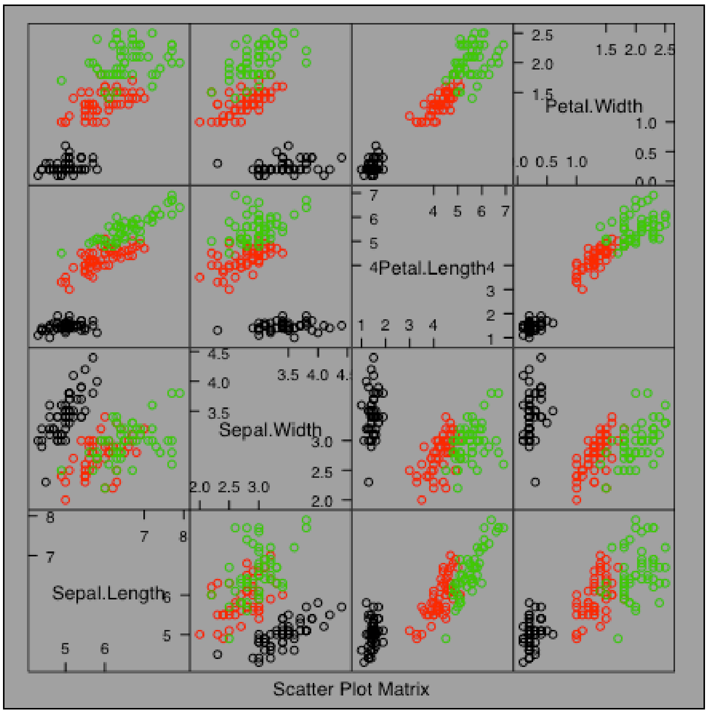
Versicolor

Virginica

Vous trouverez dans le package datasets les données sur les iris de Ficher, où chaque ligne est constituée des variables : Largeur des pétales, longueur des pétales, largeur des sépales, longueur des sépales, variété de l'iris.

```
library(lattice)
splom(~iris[ ,1:4], col = as.numeric(iris[ , 5]))
```

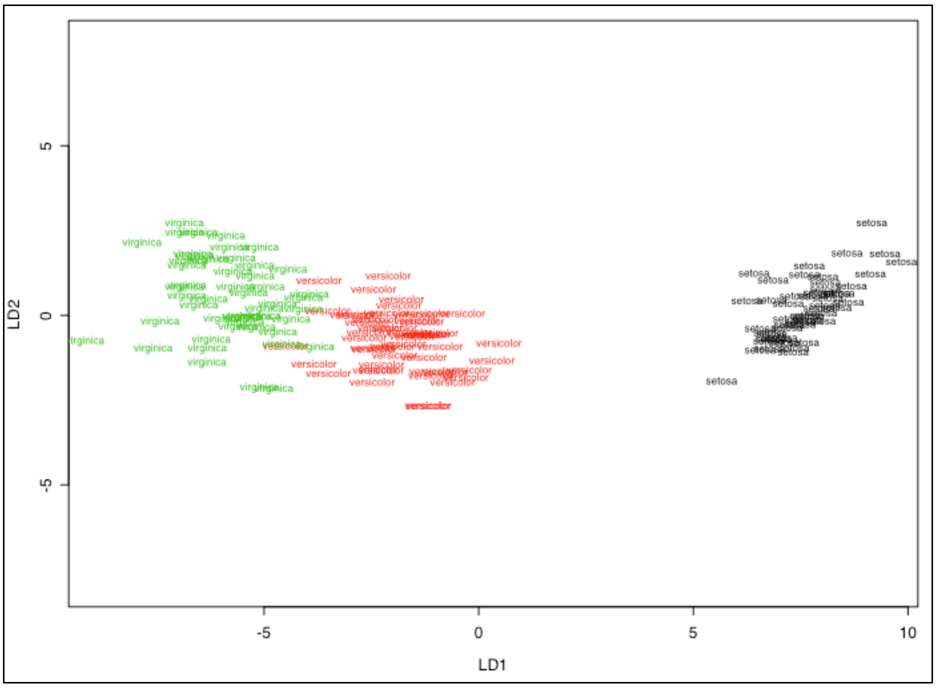
L'argument principal de la fonction `splom()` est une matrice (ici, les quatre premières colonne de Iris). Le résultat retourné par `splom()` est l'ensemble des graphes bivariés possibles entre les variables de la matrice. La fonction `splom()` fournit, dans ce cas, les mêmes résultats que la fonction `pairs()`.



Graphique 16. Résultat de la fonction splom() pour les Iris

On souhaite construire les axes discriminants résultants de l'analyse discriminante.
Soient *G* modalités et *n* individus appartenant à l'une des *G* modalités. L'idée de l'analyse discriminante est de trouver un espace de dimension au plus égal à *G-1* tel qu'en projection, l'inertie interclasse du nuage soit maximisée tandis que l'inertie intraclasse soit minimisée. En d'autres termes, l'analyse discriminante cherche à trouver un espace où les individus d'une même classe soient bien regroupés et éloignés des individus d'autres classes. La fonction `lda()` (*« linear discriminant analysis »*) du package MASS permet cette analyse.

```
library(MASS)
iris.lda = lda(iris[, 1:4], iris[, 5])
iris.lda

Prior probabilities of groups:
setosa     versicolor virginica
0.3333333 0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa     5.006        3.428       1.462        0.246
versicolor 5.936        2.770       4.260        1.326
virginica  6.588        2.974       5.552        2.026

Coefficients of linear discriminants:
             LD1        LD2
Sepal.Length 0.8293776  0.02410215
Sepal.Width  1.5344731  2.16452123
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603 2.83918785

Proportion of trace:
LD1    LD2
0.9912 0.0088

plot(iris.lda, col = as.numeric(iris[ ,5]))
```



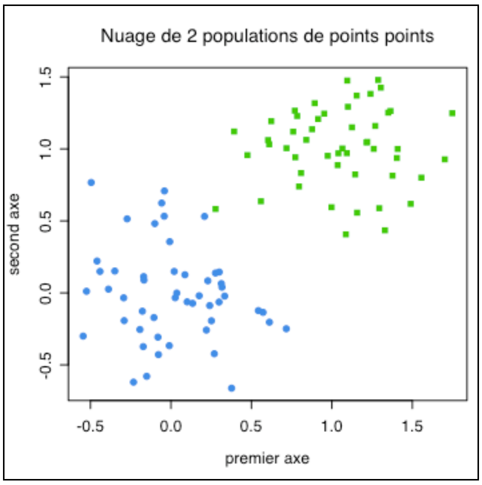
Si on souhaite tester le modèle sur de nouvelles données *Xnew*, on utilise la fonction `predict(iris.lda, Xnew)`

```
result.lda = predict(iris.lda, iris[ ,1:4])
```

V.3 Le clustering : La méthode des k-means `kmeans()`

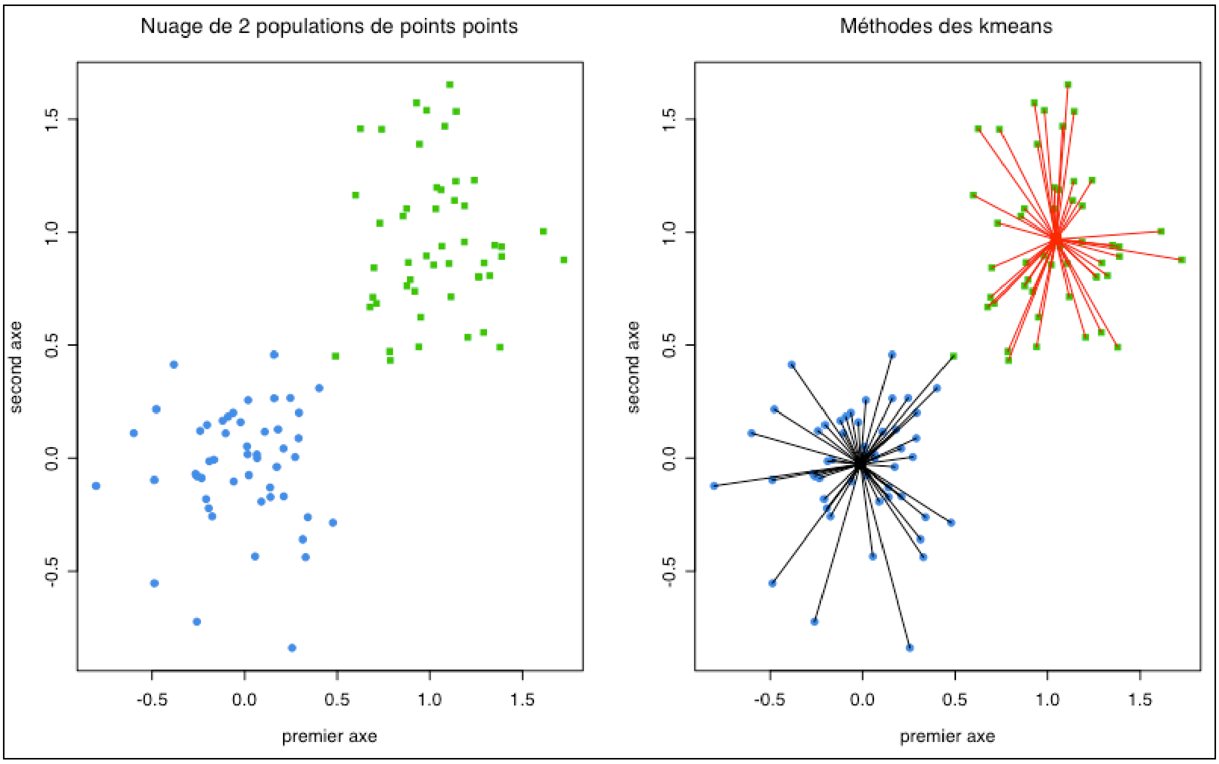
On dispose d'un nuage de n points définis dans un espace à p dimensions. L'objectif des kmeans est de partitionner ce nuage en G groupes (G étant connu a priori et fourni par l'utilisateur). Les kmeans se base sur les proximités (généralement au sens de la métrique euclidienne) entre individus dans l'espace à p dimensions pour partionner le nuage. La fonction `kmeans()` permet la recherche de cette partition.

```
## Création d'une matrice de données artificielles
## contenant deux sous-populations
C1 = matrix(rnorm(100, sd = 0.3), ncol = 2)
C2 = matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2)
mat = rbind(C1, C2)

## Visualisation des données générées
plot(C1, col = "royalblue", pch = 16,
     xlim = range(mat[ ,1]),
     ylim = range(mat[ , 2]),
     xlab = "premier axe",
     ylab = "second axe",
     main = "Nuage de 2 populations de points points"
    )
points(C2, col = "green3", pch = 15)
```



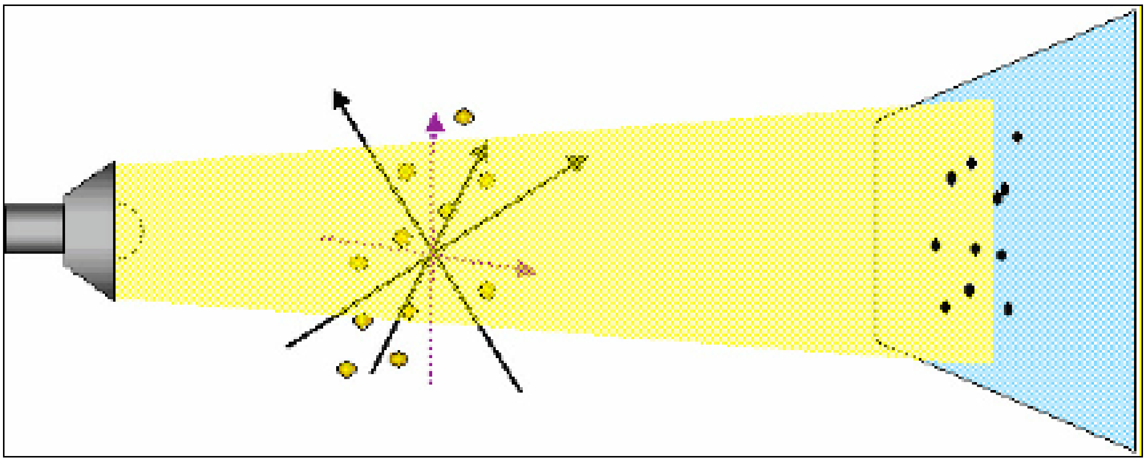

```
## Appliquer l'algorithme des kmeans

result.kmeans = kmeans(mat, 2)

## Affichage du résultat des kmeans

result.kmeans

$cluster
[1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[41] 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[80] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

$centers
          [ , 1]      [ , 2]
1   0.9849960128  0.91393763
2  -0.0006552577  0.04977682

$withinss
[1] 8.812877 7.188790

$size
[1] 50 50

## Visualisation graphique des résultats

layout(t(matrix(1:2)))
plot(C1, col = "royalblue", pch = 16,
     xlim = range(mat[ ,1]),
     ylim = range(mat[ , 2]),
     xlab = "premier axe",
     ylab = "second axe",
     main = "Méthodes des kmeans")
```

```
points(C2, col = "green3", pch = 15)

points(result.kmeans$centers, col = 1□:2, pch = 7, lwd = 3)

segments(mat[result.kmeans$cluster == 1, ][ , 1],
          mat[result.kmeans$cluster == 1, ][ , 2],
         result.kmeans$centers[1, 1],
         result.kmeans$centers[1, 2],
         col = 1
        )

segments(mat[result.kmeans$cluster == 2, ][ , 1],
          mat[result.kmeans$cluster == 2, ][ , 2],
         result.kmeans$centers[2, 1],
         result.kmeans$centers[2, 2],
         col = 2
        )
```



V.4 L'analyse en composantes principales `princomp()`

L'idée de l'analyse en composante principale (ACP) est de trouver le sous-espace qui maximise l'inertie du nuage projeté dans ce sous-espace. En d'autres termes, l'ACP cherche un espace qui déforme le moins possible le nuage de points en projection. Le schéma ci-dessous illustre les principes de l'analyse en composante principale□



Construction du premier plan principal (Source:Umetrics AB, Umea, Suède)

Le problème clé de l'analyse en composante principale réside dans le placement du projecteur: Il faut que les distances entre les points projetés soient aussi proches que possible des distances entre les points dans l'espace d'origine. L'image projetée doit refléter le plus fidèlement possible le nuage de points de l'espace d'origine.
La fonction `princomp()` permet la recherche de ce sous-espace de projection. Nous illustrons l'analyse en composante principale par deux exemples: le jeu de données USArrests et le jeu de données Auto.

Premier exemple: USArrests

Le jeu de données USArrests contient des informations sur les arrestations aux Etats-Unis en fonction des états.

```
Data(USArrests)
USArrests
                    Murder        Assault       UrbanPop      Rape
Alabama             13.2          236           58            21.2
Alaska              10.0          263           48            44.5
Arizona             8.1           294           80            31.0
Arkansas            8.8           190           50            19.5
California          9.0           276           91            40.6
Colorado            7.9           204           78            38.7
Connecticut         3.3           110           77            11.1
Delaware            5.9           238           72            15.8
Florida             15.4          335           80            31.9
Georgia             17.4          211           60            25.8
Hawaii              5.3           46            83            20.2
Idaho               2.6           120           54            14.2
Illinois            10.4          249           83            24.0
Indiana             7.2           113           65            21.0
Iowa                2.2           56            57            11.3
Kansas              6.0           115           66            18.0
Kentucky            9.7           109           52            16.3
Louisiana           15.4          249           66            22.2
Maine               2.1           83            51            7.8
Maryland            11.3          300           67            27.8
Massachusetts       4.4           149           85            16.3
Michigan            12.1          255           74            35.1
Minnesota           2.7           72            66            14.9
Mississippi         16.1          259           44            17.1
```

Missouri	9.0	178	70	28.2
Montana	6.0	109	53	16.4
Nebraska	4.3	102	62	16.5
Nevada	12.2	252	81	46.0
New Hampshire	2.1	57	56	9.5
New Jersey	7.4	159	89	18.8
New Mexico	11.4	285	70	32.1
New York	11.1	254	86	26.1
North Carolina	13.0	337	45	16.1
North Dakota	0.8	45	44	7.3
Ohio	7.3	120	75	21.4
Oklahoma	6.6	151	68	20.0
Oregon	4.9	159	67	29.3
Pennsylvania	6.3	106	72	14.9
Rhode Island	3.4	174	87	8.3
South Carolina	14.4	279	48	22.5
South Dakota	3.8	86	45	12.8
Tennessee	13.2	188	59	26.9
Texas	12.7	201	80	25.5
Utah	3.2	120	80	22.9
Vermont	2.2	48	32	11.2
Virginia	8.5	156	63	20.7
Washington	4.0	145	73	26.2
West Virginia	5.7	81	39	9.3
Wisconsin	2.6	53	66	10.8
Wyoming	6.8	161	60	15.6

Tableau 10. Murder : Meutre, Assault : Agression, UrbanPop□Population urbaine, Rape□Viol

Dans la fiche détaillée obtenue à l'aide de la commande `?princomp`, on peut lire les informations suivantes□

```
'princomp' returns a list with class "princomp" containing the following components:

sdev: the standard deviations of the principal components.
loadings: the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class '"loadings"': see 'loadings' for its 'print' method.

center: the means that were subtracted.

scale: the scalings applied to each variable.

n.obs: the number of observations.

scores: if 'scores = TRUE', the scores of the supplied data on the principal components.  These are non-null only if 'x' was supplied, and if 'covmat' was also supplied if it was a covariance list.

call: the matched call.

na.action: If relevant.
```

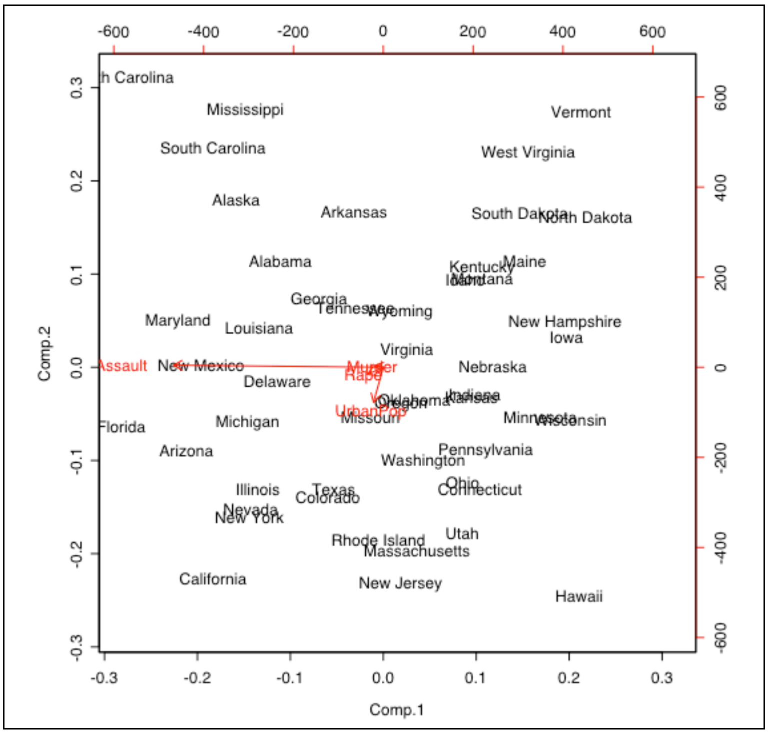
```
pca = princomp(USArrests)
```

Les résultats de l'analyse sont stockés dans une liste. La liste est composé des éléments suivants□sdev, loadings, center, scale, n.obs, scores, call, na.action. On peut donc accéder aux valeurs de l'analyse soit par l'indexation soit par le nom. Par exemple, pour accéder aux « scores » de chaque individu, (c'est à dire les coordonnées des individus dans le nouveau repère généré par l'ACP) on procède de la manière suivante :

- par le nom `pca$scores`
- par l'indexation `pca[[6]]`.

On peut également visualiser les résultats de l'analyse via la représentation biplot (représentation simultanée des individus et des variables) : on utilise la fonction `biplot()`.

`biplot(pca)`



représentation biplot du résultat de l'acp sur USArrests

Deuxième exemple : Auto

Nous reprenons ici l'exemple du livre de Michel Tenenhaus, « Méthodes statistiques en gestion » sur l'étude des voitures. Nous disposons des caratéristques de 24 voitures (cylindrée, puissance, vitesse, poids, largeur, longeur) reporté sur le tableau 11.

```
Auto

                          Cylindree Puissance Vitesse Poids Largeur Longueur
Citroen C2 1.1 Base            1124        61     158   932    1659     3666
Smart Fortwo Coupe              698        52     135   730    1515     2500
Mini 1.6 170                   1598       170     218  1215    1690     3625
Nissan Micra 1.2 65            1240        65     154   965    1660     3715
Renault Clio 3.0 V6            2946       255     245  1400    1810     3812
Audi A3 1.9 TDI                1896       105     187  1295    1765     4203
Peugeot 307 1.4 HDI 70         1398        70     160  1179    1746     4202
Peugeot 407 3.0 V6 BVA         2946       211     229  1640    1811     4676
Mercedes Classe C 270 CDI      2685       170     230  1600    1728     4528
BMW 530d                       2993       218     245  1595    1846     4841
Jaguar S-Type 2.7 V6 Bi-Turbo  2720       207     230  1722    1818     4905
BMW  745i                      4398       333     250  1870    1902     5029
Mercedes Classe S 400 CDI      3966       260     250  1915    2092     5038
```

Citroen C3 Pluriel 1.6i	1587	110	185	1177	1700	3934
BMW Z4 2.5i	2494	192	235	1260	1781	4091
Audi TT 1.8T 180	1781	180	228	1280	1764	4041
Aston Martin Vanquish	5935	460	306	1835	1923	4665
Bentley Continental GT	5998	560	318	2385	1918	4804
Ferrari Enzo	5998	660	350	1365	2650	4700
Renault Scenic 1.9 dCi 120	1870	120	188	1430	1805	4259
Volkswagen Touran 1.9 TDI 105	1896	105	180	1498	1794	4391
Land Rover Defender Td5	2495	122	135	1695	1790	3883
Land Rover Discovery Td5	2495	138	157	2175	2190	4705
Nissan X-Trail 2.2 dCi	2184	136	180	1520	1765	4455

Tableau 11. Caractéristique de 24 voitures

Remarquons que :

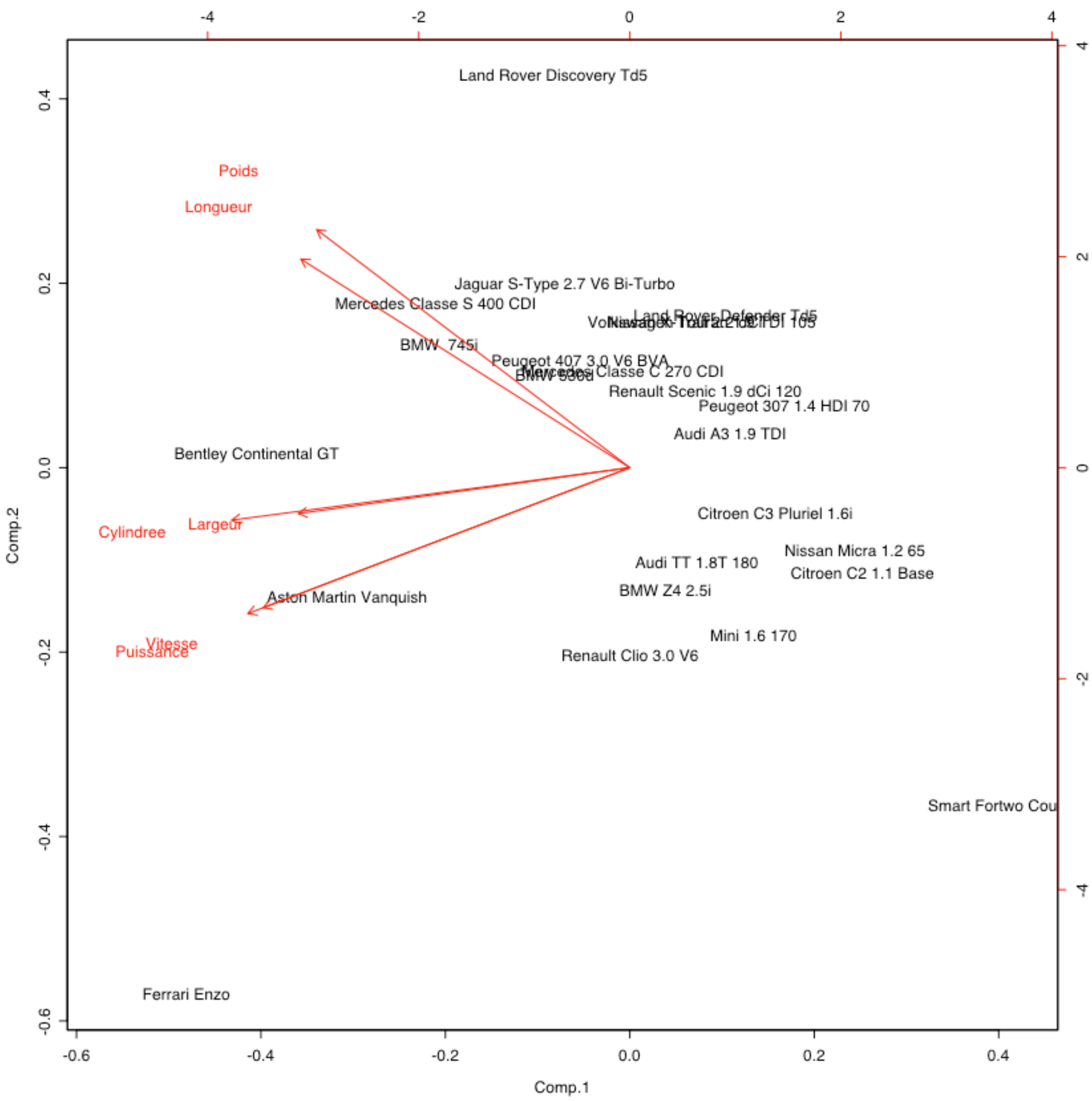
1. La Smart Fortwo Coupé est minimum sur toutes ses caractérisitques.
2. L'Aston Martin, la Bentley et la Ferrari se distinguent par leur motorisation exceptionnelle.
3. La Renault Scenic, la Volkswagen Touran, les deux Land Rover et la Nissan X-Trail ont un habitat important par rapport à leur motorisation.
4. La Renault Clio 3.0 V6 a une motorisation élévée par rapport à son habitat

L'analyse en composante principale permet-elle de mettre en exergue ces remarques ?

```
auto.pca = princomp(auto, cor = TRUE)
```

Notons l'utilisation de l'argument cor = TRUE qui effectue l'analyse en composante principale à partir de la matrice des corrélations plutôt que de la matrice de variance-covariance.

```
biplot(auto.pca)
```

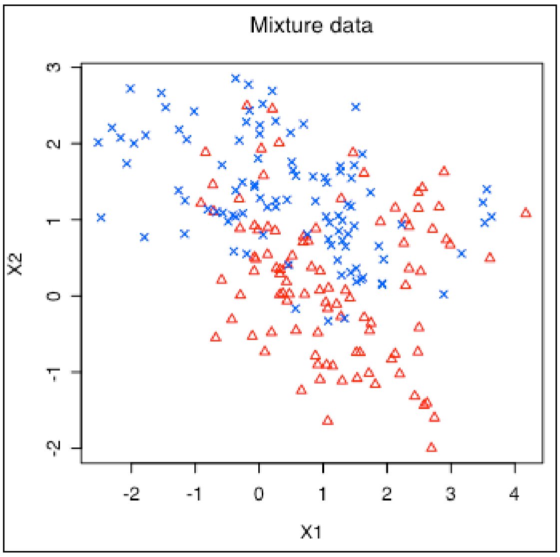


On peut remarquer, par exemple, que les projections des voitures sur l'axe «Vitesse» restituent bien la répartition des données de départ. Les projections des voitures les plus rapides (Ferrari, Bentley et Aston Martin) s'opposent bien à celles les plus lentes (Smart, Land Rover, Defender, Nissan Nicra).

V.5 La régression logistique à travers la fonction `glm()`

La régression logistique modélise la probabilité d'appartenance d'un individu à une classe. Le modèle logistique fait partie de la famille des modèles linéaires généralisés et on peut donc utiliser la fonction `glm()` («generalized linear model») pour construire un modèle logistique. On souhaite construire une régression logistique de mixture data. La figure ci-dessous représente le nuage de points à analyser.

```
plot(X,
     col = Y+3,
     pch = Y + 4,
     main = "Mixture data",
     xlab = "X1",
     ylab = "X2"
    )
```



Construction du modèle logistique

```
result.glm = glm(Y ~ X, family = binomial)

summary(result.glm)

Call:
glm(formula = Y ~ X, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.28489  -0.86579   0.05965   0.90614   1.88232

Coefficients:
              Estimate    Std. Error    z value    Pr(>|z|)
(Intercept)   -0.9780     0.2945        -3.321     0.000897 ***
X1            -0.1344     0.1372        -0.980     0.327272
X2            1.3981      0.2316        6.035      1.59e-09 ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 277.26  on 199 degrees of freedom
Residual deviance: 209.54  on 197 degrees of freedom
AIC: 215.54

Number of Fisher Scoring iterations: 4
```

Si on souhaite tester le modèle sur de nouvelles données *Xnew*, on utilise la fonction `predict(result.glm, Xnew)`

```
Y_predit = predict(result.glm, as.data.frame(Xtest))
```

V.6 Les Support Vector Machines (SVM) `svmpath()`

Proposé par V. Vapnik en 1998, les Support Vector Machines (SVM) consiste à trouver « l'hyperplan séparateur optimal » c'est-à-dire l'hyperplan dont la distance minimale aux exemples d'apprentissage est maximale (Cf. figure 3).

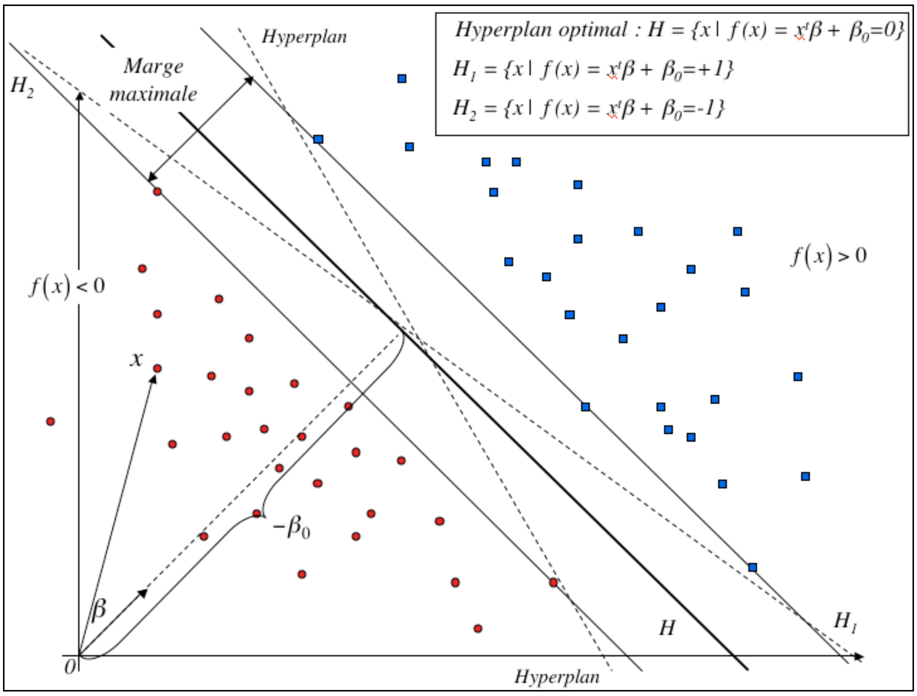
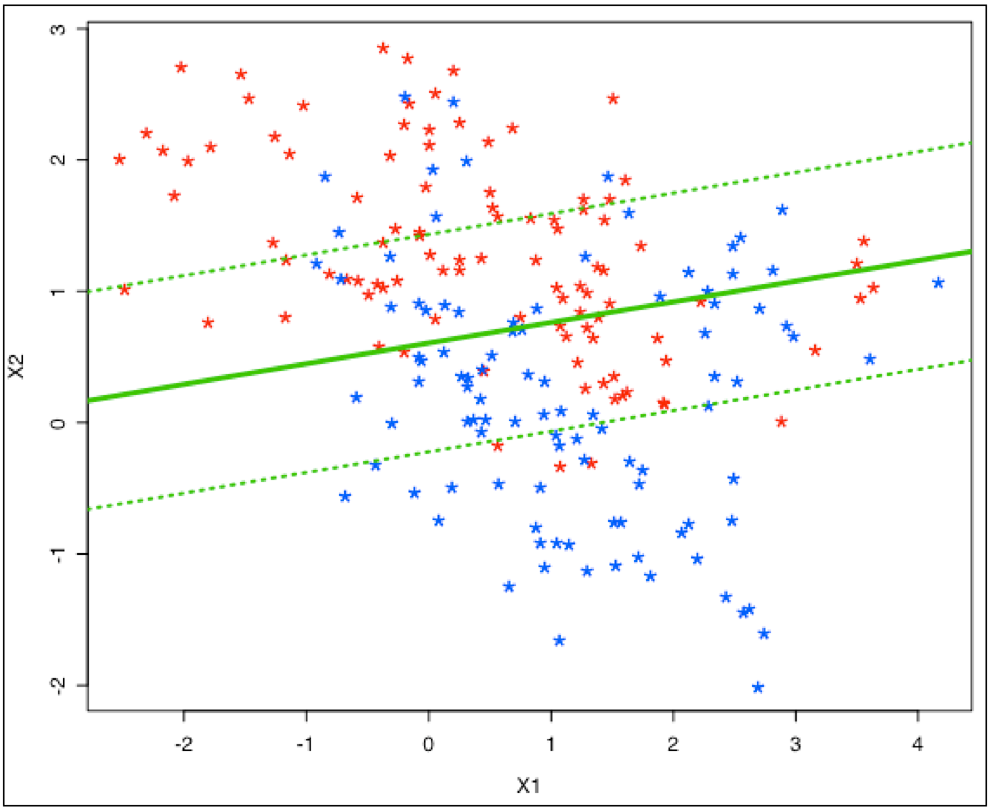


Figure 3. Hyperplan optimal

Les points qui « supportent » les hyperplans H_1 et H_2 sont les « support vectors » ; seuls ces points participent à la construction du modèle.
Le package svmpath permet de constuire des modèles résultant des SVM ; la fonction `svmpath()` permet de générer les SVM. En voici l'illustration les données mixture.data.

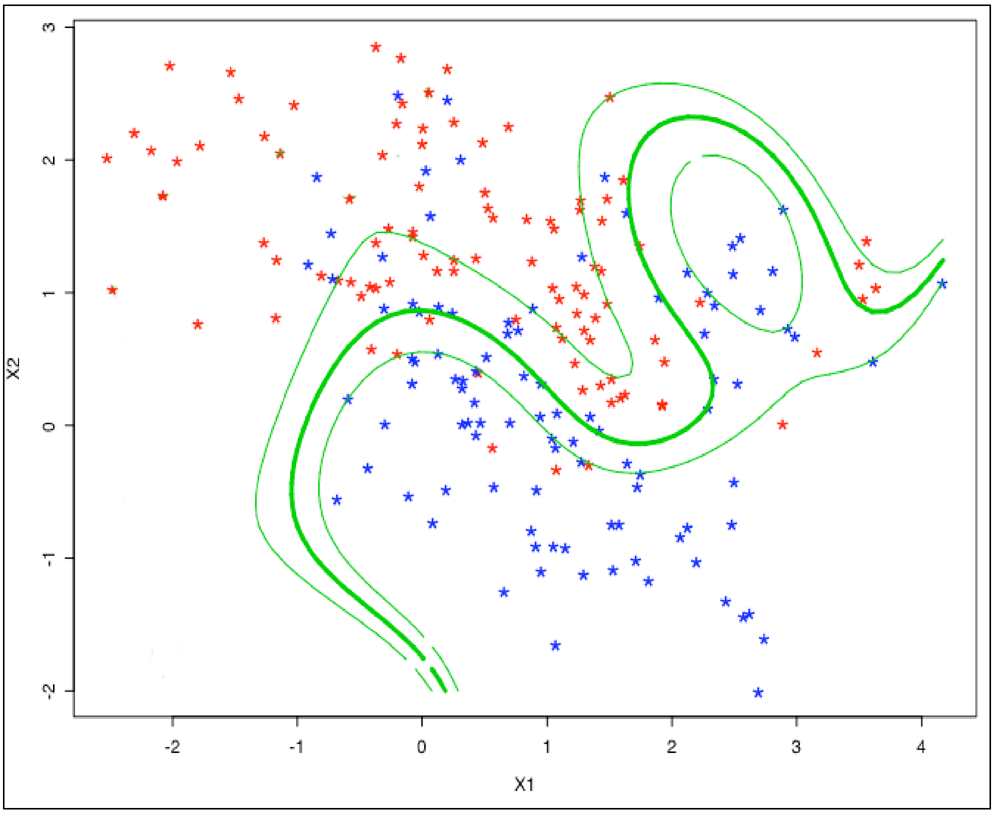
```
library(svmpath)
data(svmpath)
attach(mixture.data)
svmpath(mixture.data$x, mixture.data$y, trace = TRUE, plot = TRUE)
```



Construction d'une frontière de décision linéaire résultant des SVM

Comme nous l'avons dit, les SVM consistent à rechercher un hyperplan séparateur optimal. Cette fonction de décision est une séparatrice linéaire. Elle n'est donc pas adaptée à la majorité des problèmes pratiques. Pour permettre l'application des SVM aux cas les plus généraux, l'idée fondamentale des SVM est de plonger les individus dans un espace de redescription (généralement de plus grande dimension); la construction de l'hyperplan optimal s'effectuant alors dans ce nouvel espace. Cette transformation non linéaire en pratique, fournit une frontière de décision non linéaire dans l'espace d'origine.

```
svmpath(mixture.data$x, mixture.data$y,
        kernel=radial.kernel,
        param.kernel = 3,
        trace = TRUE,
        plot = TRUE
       )
```



Construction d'une frontière de décision non linéaire résultant des SVM

Pour tester des nouveaux individus sur le modèle générer par les SVM on utilise également la foncton générique `predict()`.