

PROJET MODÈLES LINÉAIRES

UNIVERSITÉ LYON 1

## Classification automatique des puits de pétrol



réalisé par  
Florian Robinet et Hugo Duportet

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Approximation des courbes par régression sur des modèles polynômiales</b>	<b>2</b>
<b>3</b>	<b>Approximation des courbes par régression sur des modèles exponentiels</b>	<b>6</b>
<b>4</b>	<b>Degré d'incertitude sur les régressions</b>	<b>7</b>
<b>5</b>	<b>Utilisation de la regression logistique</b>	<b>9</b>
<b>6</b>	<b>Lissage des courbes</b>	<b>11</b>
<b>7</b>	<b>Conclusion</b>	<b>15</b>
<b>8</b>	<b>Code</b>	<b>16</b>

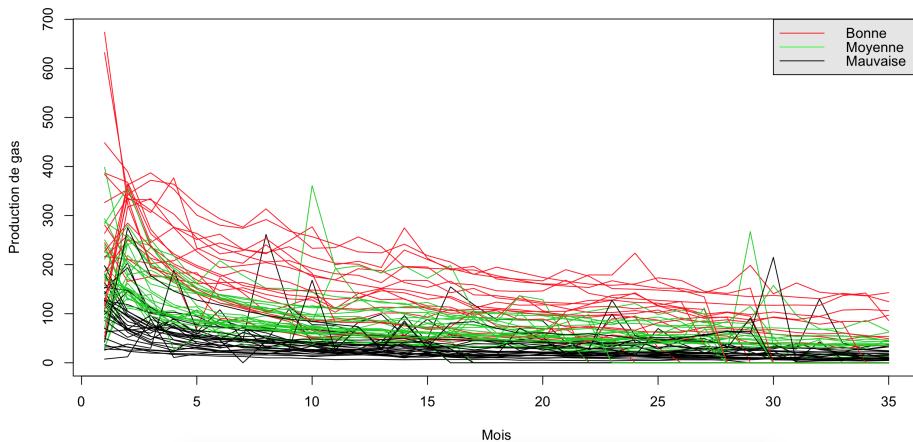
# 1 Introduction

Jeux de données :

- Production de 75 puits dans un champ pétrolier du Canada
- Périodes différentes mais ramenées à une durée de 36 mois
- Les valeurs manquantes sont remplacées par un 0
- Les productions sont ordonnées en trois classes : bonne, moyenne et mauvaise

On commence par importer le jeux de données sur R.

On réalise une première visualisation des données avec la classification expert.



Objectif :

L'objectif de ce projet est de mettre en place une classification automatique des puits.

On va donc réaliser différents types de modélisation des courbes pour pouvoir faire de la classification avec les coefficients obtenus.

## 2 Approximation des courbes par régression sur des modèles polynômiaux

On va chercher ici la solution au problème de minimisation suivant :

$$\operatorname{argmin}_{\beta}(\|y - X\beta\|_2)$$

Cette solution existe sous réserve d'inversibilité de  ${}^t XX$  et correspond à :

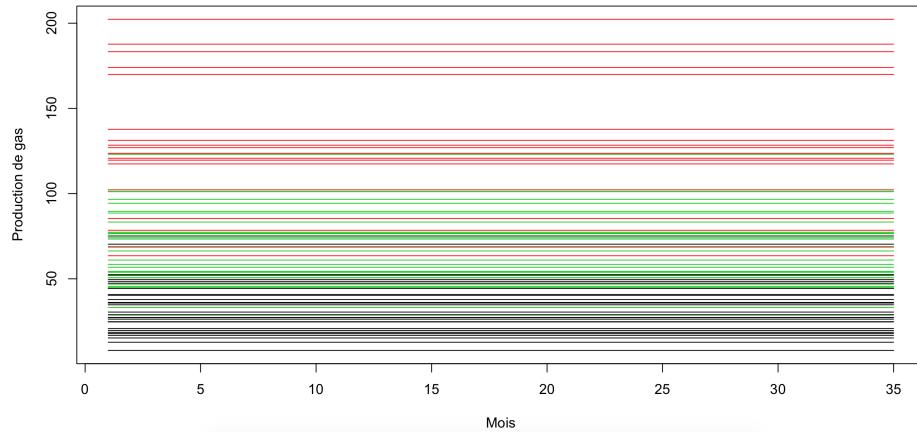
$$\beta = ({}^t XX)^{-1} {}^t X y$$

Il s'agit donc de remplacer chaque courbe par une fonction paramétrique polynomiale. L'ajustement des paramètres se fait par la méthode de régression et on le fait grâce à la fonction lm() sur R. On réalise ces approximations pour des fonctions polynomiale de degré 0,1,2,3 et 4.

#### **Approximation par fonction polynomiale de degré 0 :**

Il s'agit de réaliser des approximations avec des fonctions constantes.

Le modèle s'écrit :  $Y = \beta_0 + \epsilon$

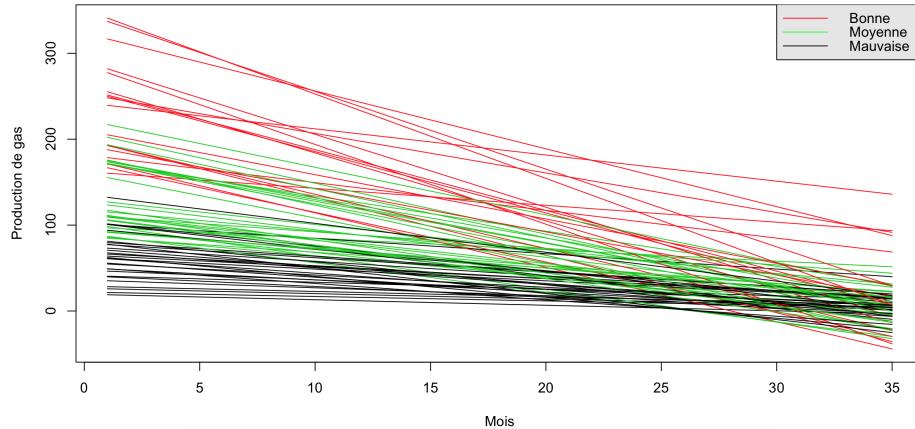


Dans ce type de régression les modélisations se font grâce aux valeurs moyennes des différents puits. Cela permet de conserver une classification relativement bonne mais concernant la modélisation des courbes cette méthode est peu efficace. En effet, on voit ici que les courbes sont des constantes et l'aspect convexe initial n'est pas conservé.

#### **Approximation par fonction polynomiale de degré 1 :**

Il s'agit de réaliser des approximations avec des fonctions affines.

Le modèle s'écrit :  $Y = \beta_0 + \beta_1 X + \epsilon$

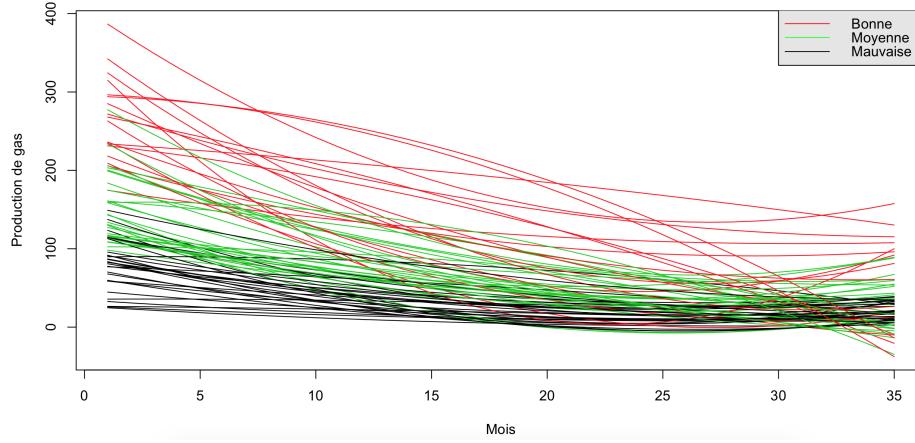


La classification dans ce type de modèle est légèrement dégradée par rapport à la modélisation par rapport à une constante. De plus les valeurs de plusieurs courbes deviennent négatives à partir du 30 ème mois ce qui n'est pas possible dans notre contexte. Cependant la modélisation des courbes est amélioré dans le sens où celles-ci sont décroissantes.

#### Approximation par fonction polynomiale de degré 2 :

Il s'agit de réaliser des approximations avec des fonctions polynomiales de degré 2.

Le modèle s'écrit :  $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$

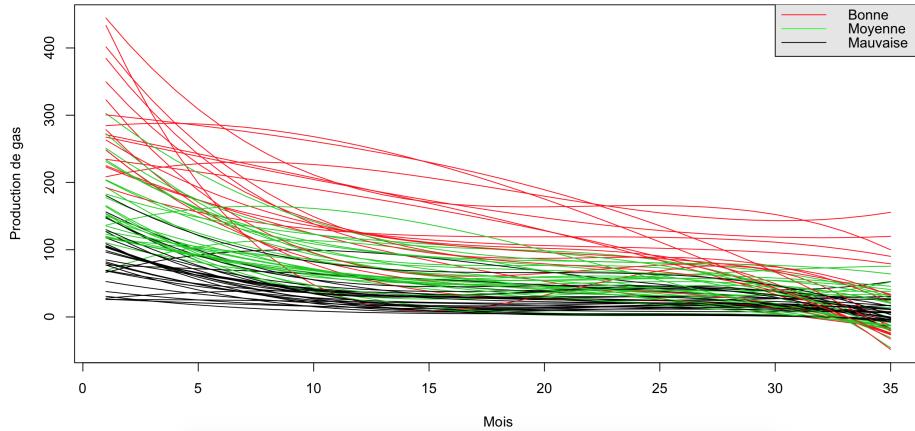


L'insertion du degré 2 permet une bien meilleure modélisation des courbes et l'on retrouve l'aspect convexe des courbes. Cependant certaines courbes deviennent concaves ce qui paraît étrange dans notre exemple. De plus, les courbes ont tendances à remonter en fin de période. La qualité de la classification reste relativement constante.

#### **Approximation par fonction polynomiale de degré 3 :**

Il s'agit de réaliser des approximations avec des fonctions polynomiales de degré 3.

Le modèle s'écrit :  $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$

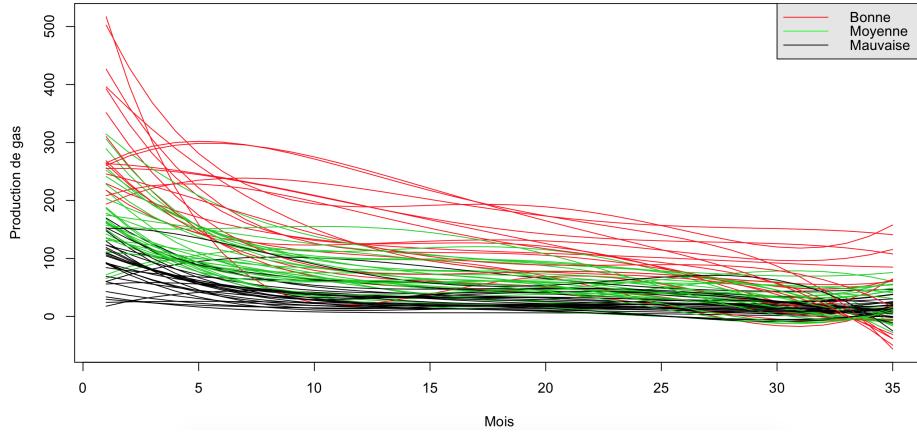


La qualité discriminante du modèle reste relativement stable la aussi. L'introduction du troisième degré augmente encore un peu la qualité de modélisation des courbes. Les courbes ne remonte plus sur la fin de période mais sont décroissantes.

#### **Approximation par fonction polynomiale de degré 4 :**

Il s'agit de réaliser des approximations avec des fonctions polynomiales de degré 4.

Le modèle s'écrit :  $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$



La qualité de classification s'améliore un peu cependant la qualité de modélisation diminue. Il y a un plus grand nombre de courbes concaves et certaines courbes remontent sur encore sur la fin de période.

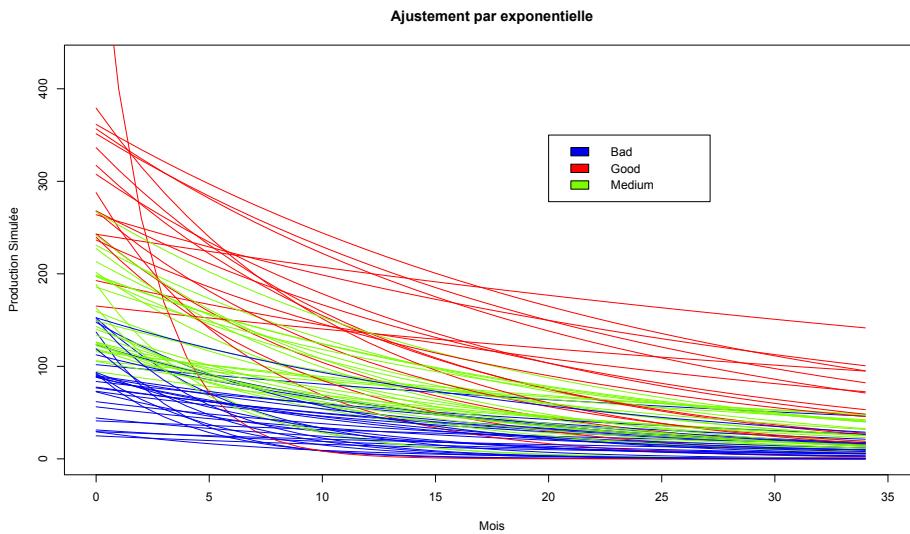
### 3 Approximation des courbes par régression sur des modèles exponentiels

**Approximation par des fonctions de type exponentielle :**

Il s'agit de réaliser des approximations avec des fonctions du type  

$$Y = -k_0 * \exp(-k_1 t)$$

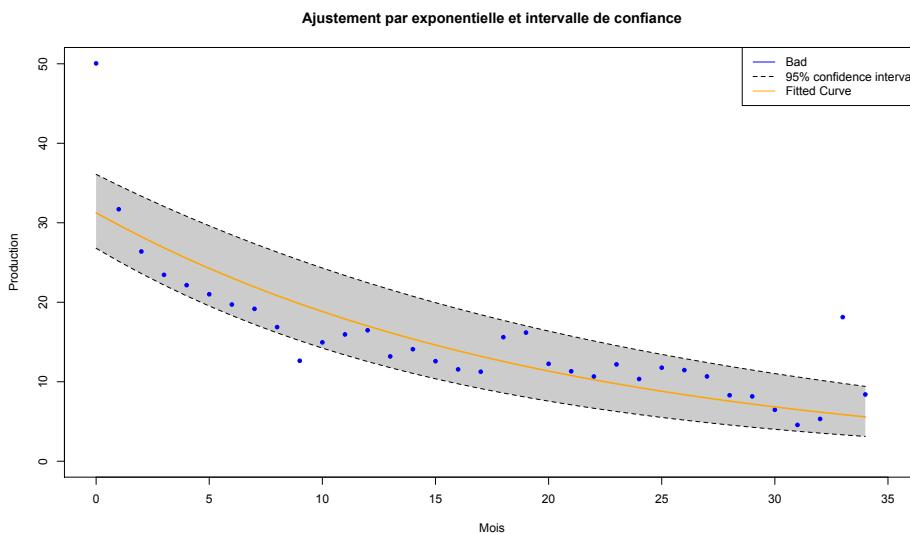
- $Y$  : la production
- $t$  : le mois
- $k_0$  et  $k_1$  deux paramètres à estimer

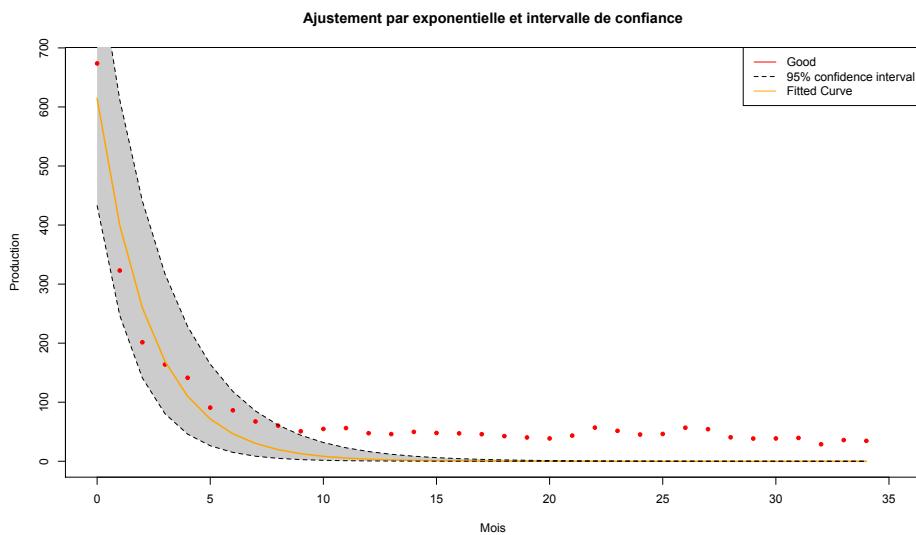
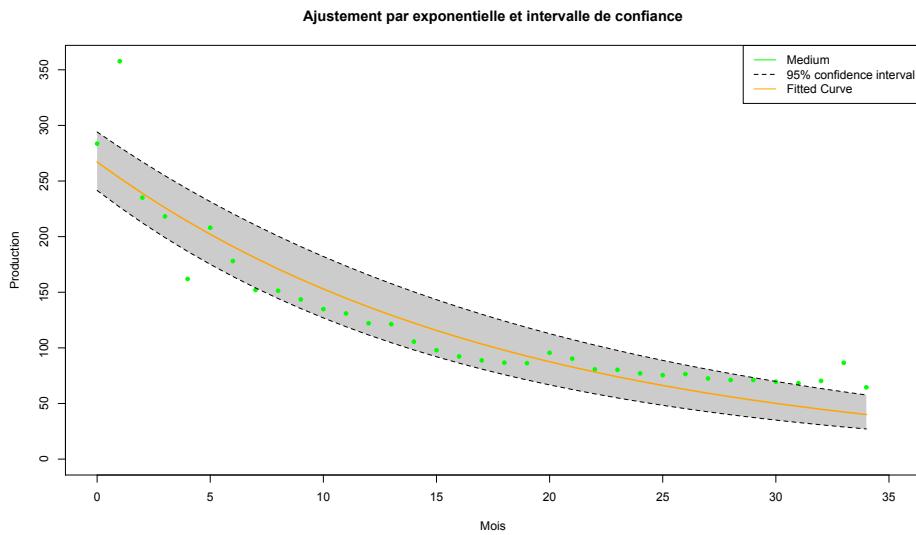


Les courbes sont bien approchées par le modèle. De tous les modèles utilisés, ce sont celles qui semblent le plus se rapprocher de la "réalité".

#### 4 Degré d'incertitude sur les régressions

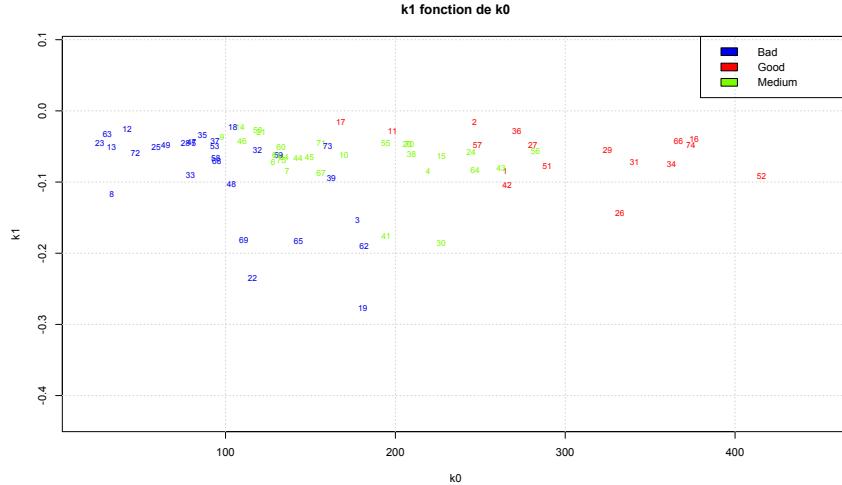
Pour chaque courbe, on trace les courbe basse et haute associé un niveau 0 95%. Si la qualité d'un puit de pétrole dépend effectivement de la courbe exponentielle, cet intervalle fournit un premier indicateur sur la fiabilité des données reçues.





On remarque aussi que le modèle permet de rapidement identifier les caractéristiques de ces courbes afin de mieux les décrire.

## 5 Utilisation de la régression logistique



On trace les différentes courbes en fonction des paramètres de leur courbe exponentielle associée et de leur qualité en couleur.

### Confusion Matrix and Statistics

		Reference		
		bad	Good	medium
Prediction	bad	24	0	5
	Good	0	15	3
	medium	4	3	21

### Overall Statistics

```

Accuracy : 0.8
95% CI : (0.6917, 0.8835)
No Information Rate : 0.3867
P-Value [Acc > NIR] : 3.059e-13

```

```

Kappa : 0.694
McNemar's Test P-Value : NA

```

### Statistics by Class:

	Class: bad	Class: Good	Class: medium
Sensitivity	0.8571	0.8333	0.7241
Specificity	0.8936	0.9474	0.8478
Pos Pred Value	0.8276	0.8333	0.7500
Neg Pred Value	0.9130	0.9474	0.8298
Prevalence	0.3733	0.2400	0.3867
Detection Rate	0.3200	0.2000	0.2800
Detection Prevalence	0.3867	0.2400	0.3733
Balanced Accuracy	0.8754	0.8904	0.7860

Étant donné que nous avons trois classes, nous faisons une régression logistique multiple. Nous obtenons 80% de précision, un AIC de 69.55 et comme attendu,

le classifieur ne parvient pas à différencier certain bad et medium ainsi que certains medium et Good. Nous obtenons 15 courbes mal classées : 6, 7, 9, 11, 17, 18, 24, 32, 39, 42, 43, 46, 56, 59, 73. Nous allons modifier la classe de la courbe 73 et à nouveau appliquer l'algorithme :  
 Nous gagnons plus de 5% de précision, un AIC de 61.18, et il n'y a plus qu'un seul bad qui n'est pas bien classé.

```
Confusion Matrix and Statistics

          Reference
Prediction  bad  Good  medium
    bad      24     0      4
    Good      0     15      3
    medium     1      3     25

Overall Statistics

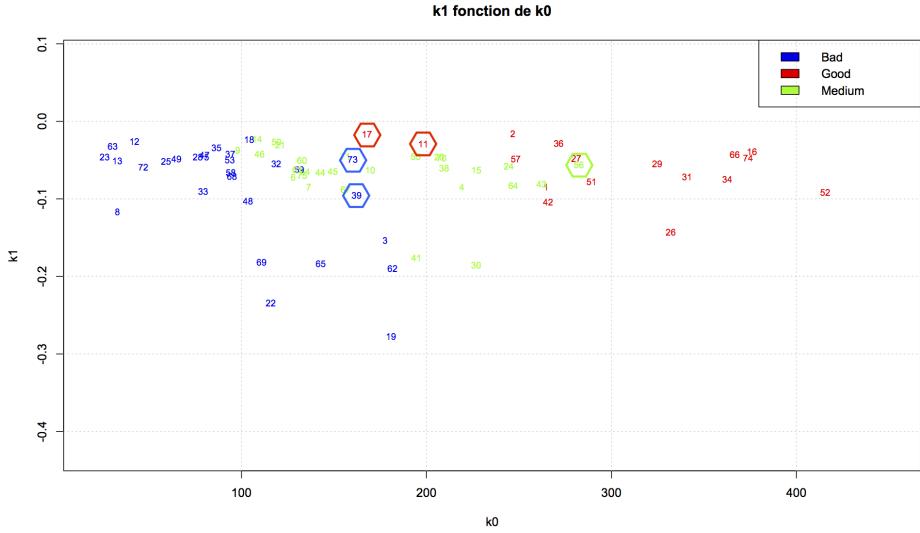
    Accuracy : 0.8533
    95% CI   : (0.7527, 0.9244)
    No Information Rate : 0.4267
    P-Value [Acc > NIR] : 2.605e-14

    Kappa : 0.7754
    Mcnemar's Test P-Value : NA

Statistics by Class:

           Class: bad  Class: Good  Class: medium
Sensitivity       0.9600    0.8333    0.7812
Specificity        0.9200    0.9474    0.9070
Pos Pred Value     0.8571    0.8333    0.8621
Neg Pred Value     0.9787    0.9474    0.8478
Prevalence         0.3333    0.2400    0.4267
Detection Rate     0.3200    0.2000    0.3333
Detection Prevalence  0.3733    0.2400    0.3867
Balanced Accuracy    0.9400    0.8904    0.8441
> |
```

Nous itérons cette étape sur l'ensemble des courbes mal classées et finalement, les courbes qui ont le plus de chances d'être mal classées, c'est à dire dont les facteurs, lorsque modifiés, améliorent au mieux le modèle sont :



Les courbes 56, 17, 11, 73, et 39 ont le plus de chances d'avoir été mal classées. Naturellement, ce sont les points situés le plus à "l'intérieur" d'une classe qui leur est différente qui perturbent la qualité globale du modèle.

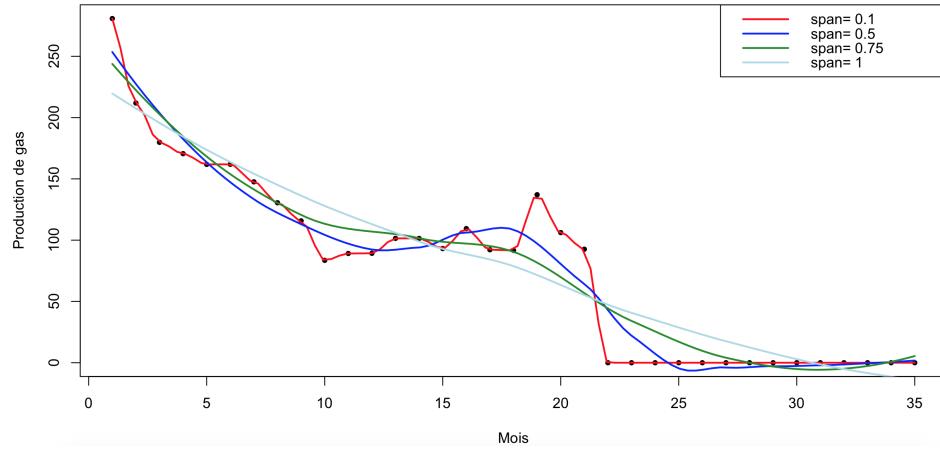
## 6 Lissage des courbes

La régression loess est une méthode de régression non-paramétrique. Une méthode non-paramétrique est une méthode qui n'est pas associée à une équation, comme par exemple une régression linéaire ou polynomiale classique. Elle permet de produire des courbes lissées, ajustées à un nuage de point.

La régression loess est faite par ajustement local de la courbe. Cette méthode fait intervenir trois notions importantes. Tout d'abord on ajuste un polynôme de degré 1 ou 2 pour déterminer la valeur  $y$  que prend la courbe au point d'abscisse  $x_i$ . Ensuite l'ajustement est réalisé à partir des points dans le voisinage du point étudié. Enfin, on modélise la distance dans le voisinage à travers la mise en place d'une pondération, les points les plus proches de  $x_i$  ont davantage de poids dans l'ajustement.

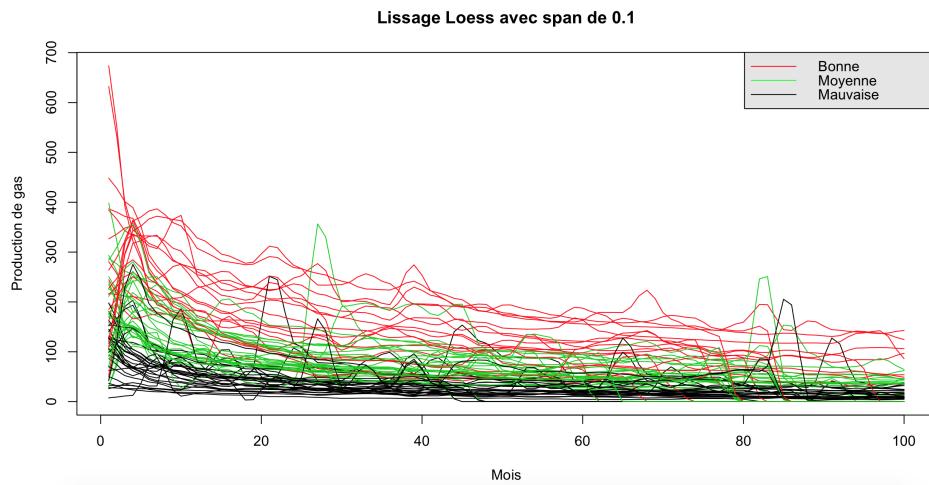
Nous utilisons le modèle loess de lissage :

**Lissage avec Loess en fonction du span**

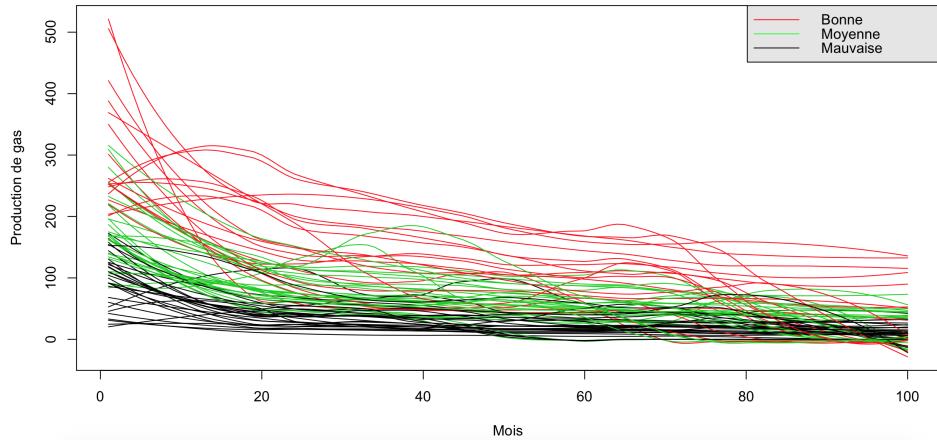


Cette exemple a été réalisé avec les données du premier puits. On a cherché à travers ce graphe à mettre en évidence l'importance de la valeur du span. On voit que plus le span est faible plus l'interpolation va coller aux données. Ce paramètre est important car il va permettre d'enlever les "spikes" et il permet d'ajuster la finesse du lissage.

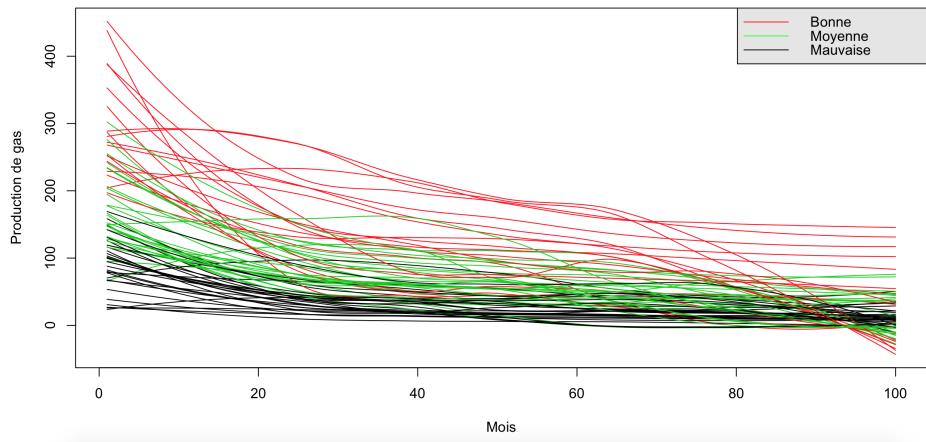
En associant ce modèle à nos données on obtient les courbes suivantes :



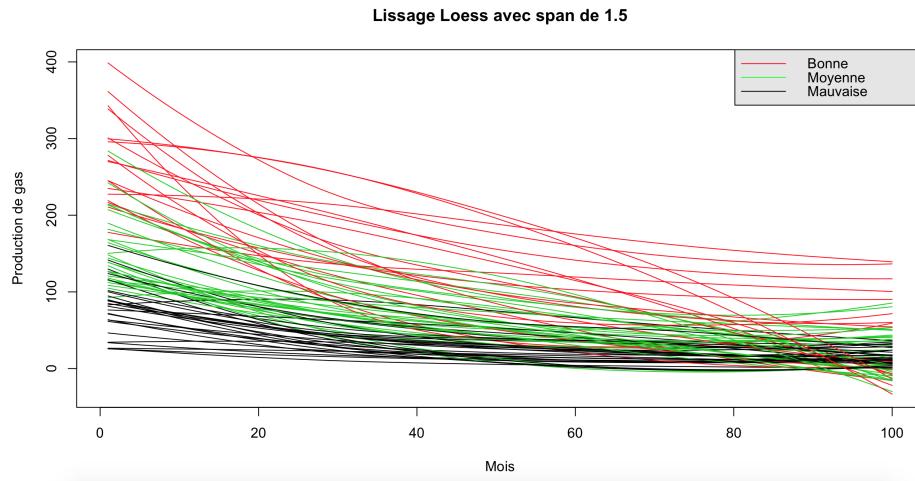
Lissage Loess avec span de 0.5



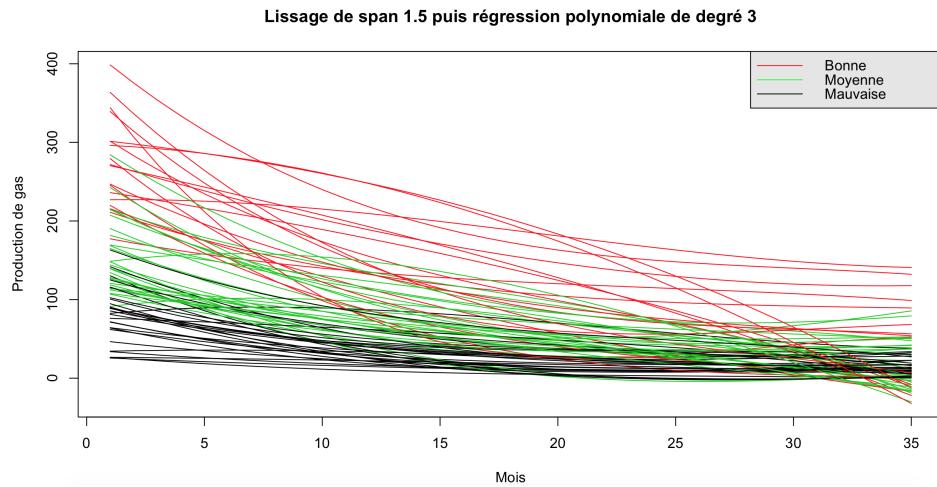
Lissage Loess avec span de 0.75



On décide de choisir un span à 1.5 pour réaliser nos lissages. Ce paramètre de lissage nous paraît pertinent dans le sens où il met en avant une forte différenciation des courbes vis-à-vis de la classification.

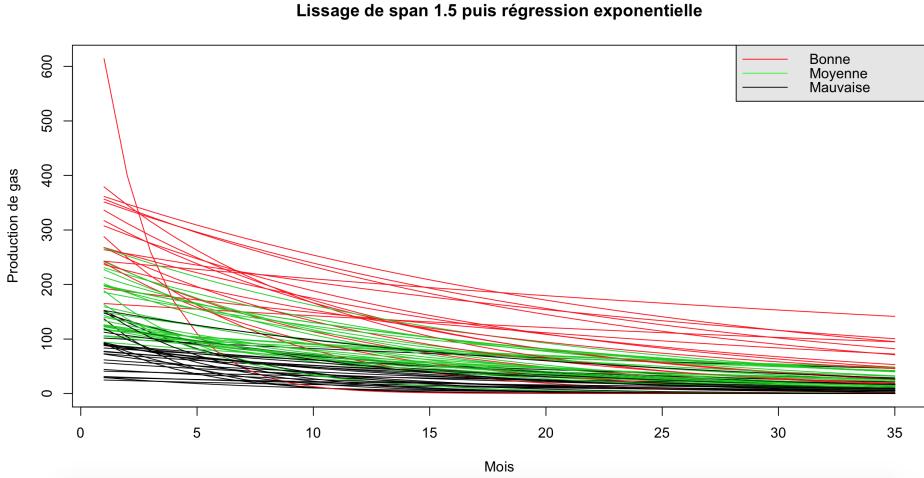


On réalise ensuite une combinaison entre un lissage dans un premier temps puis une régression polynomiale de degré 3.



Cette réalisation est relativement pertinente. Le lissage amène une amélioration des résultats. La modélisation des courbes est plutôt bonne et l'aspect de classification est respecté.

Enfin on réalise une combinaison entre un lissage dans un premier temps puis une régression exponentielle à deux paramètres.



La réalisation est très bonne ici. On a une modélisation des courbes très efficaces et un aspect de classification conservé. Cette méthodologie est pour l'instant la meilleure parmi celles que nous avons réalisées.

Le lissage permet une légère amélioration (1 2%) du modèle de prédiction dans la plupart des cas. Cette précision est sensible au tspan choisi dans le modèle loess.

## 7 Conclusion

Nous avons ainsi vu plusieurs techniques d'approches de courbes de productions de puits pétroliers. Les régressions polynomiales, même si étant mathématiquement viables à un certain degré, ne reflètent pas la réelle évolution de la quantité de pétrole à l'inverse des modèles exponentiels qui s'adaptent très bien au problème. Nous avons notamment vu que certains puits pouvaient présenter des erreurs de jugement de la part des experts et la modification des classes parmi nos clusters de puits préexistants selon ces erreurs améliorent nettement la qualité des modèles. Enfin, l'ajout d'un filtre de lissage permet de gagner encore un peu de précision dans la classification de ces puits mais une phase de tuning est indispensable afin de mettre tout ces paramètres en harmonie.

## 8 Code

```
1 #Importer le jeux de données#
2 table=read.csv('~/Desktop/M2 Data Science/Regression lineaire
  ↪ /Projet/FW_Donnees_Puits.csv',header=TRUE,sep=";")
3 attach(table)
4
5
6 #Données de production avec classification expert#
7 matplot(t(table[c(1:length(table[,1]),-c(1,37,38)])),type="l",
  ↪ col=Classification.Expert,lty=1,xlab="Mois",ylab="Production
  ↪ de gas")
8 legend(30, 700, c("Bonne", "Moyenne", "Mauvaise"), col =
  ↪ c("red","green","black"), text.col = "black", lty = 1, merge
  ↪ = TRUE, bg = "gray90")
9
10
11 #Polynome de degre 0#
12 tab=0
13 for (i in 1:length(table[,1])){
14   vec=as.numeric(table[i,-c(1,37,38)])
15   model=lm(vec ~ 1)
16   tab=rbind(tab,fitted(model))
17 }
18 matplot(t(tab[-1,]),type="l", col=Classification.Expert, lty=1,
  ↪ xlab="Mois", ylab="Production de gas")
19
20
21 #Polynome de degre 1#
22 tab=0
23 x=seq(1:35)
24 for (i in 1:length(table[,1])){
25   vec=as.numeric(table[i,-c(1,37,38)])
26   model=lm(vec ~ x)
27   tab=rbind(tab,fitted(model))
28 }
29 matplot(t(tab[-1,]), type="l", col=Classification.Expert,
  ↪ lty=1,xlab="Mois", ylab="Production de gas")
30 legend(30, 357, c("Bonne", "Moyenne", "Mauvaise"), col =
  ↪ c("red","green","black"), text.col = "black", lty = 1, merge
  ↪ = TRUE, bg = "gray90")
31
32
33 #Polynome de degre 2#
34 tab=0
35 x=seq(1:35)
36 for (i in 1:length(table[,1])){
37   vec=as.numeric(table[i,-c(1,37,38)])
38   model=lm(vec ~ x + I(x^2))
39   tab=rbind(tab,fitted(model))
```

```

40 }
41 matplot(t(tab[-1,]), type="l", col=Classification.Expert, lty=1,
42   ↪ xlab="Mois", ylab="Production de gas")
43 legend(30, 405, c("Bonne", "Moyenne", "Mauvaise"), col =
44   ↪ c("red","green","black"), text.col = "black", lty = 1, merge
45   ↪ = TRUE, bg = "gray90")
46
47 #Polynome de degre 3#
48 tab=0
49 x=seq(1:35)
50 for (i in 1:length(table[,1])){
51   vec=as.numeric(table[i,-c(1,37,38)])
52   model=lm(vec ~ x + I(x^2) + I(x^3))
53   tab=rbind(tab,fitted(model))
54 }
55 matplot(t(tab[-1,]), type="l", col=Classification.Expert, lty=1,
56   ↪ xlab="Mois", ylab="Production de gas")
57 legend(30, 470, c("Bonne", "Moyenne", "Mauvaise"), col =
58   ↪ c("red","green","black"), text.col = "black", lty = 1, merge
59   ↪ = TRUE, bg = "gray90")
60
61 #Polynome de degre 4#
62 tab=0
63 x=seq(1:35)
64 for (i in 1:length(table[,1])){
65   vec=as.numeric(table[i,-c(1,37,38)])
66   model=lm(vec ~ x + I(x^2) + I(x^3) + I(x^4))
67   tab=rbind(tab,fitted(model))
68 }
69 matplot(t(tab[-1,]), type="l", col=Classification.Expert, lty=1,
70   ↪ xlab="Mois", ylab="Production de gas")
71 legend(30, 540, c("Bonne", "Moyenne", "Mauvaise"), col =
72   ↪ c("red","green","black"), text.col = "black", lty = 1, merge
73   ↪ = TRUE, bg = "gray90")
74
75 #Paramètres initiaux#
76 a_start <- 80
77 b_start <- -2*log(2)/a_start
78
79 x <- seq(1,len_exp,1)
80
81 color = c('blue2','red1','lawngreen')
82
83 #Modèle exponentiel
84
85 model <- nls(vec~a*exp(b*x),start=list(a=a_start,b=b_start))
86
87
88 #Exemple de lissage avec Loess sur le premier puit en fonction
89   ↪ du span#
90 y=as.numeric(table[1,-c(1,37,38)])

```

```

80 x=seq(1,35,1)
81 mod1=loess(y~x,span=0.1)
82 mod2=loess(y~x,span=0.5)
83 mod3=loess(y~x,span=0.75)
84 mod4=loess(y~x,span=1)
85 xfit=seq(from=min(x),to=max(x),length.out=100)
86 yfit1=predict(mod1,newdata=xfit)
87 yfit2=predict(mod2,newdata=xfit)
88 yfit3=predict(mod3,newdata=xfit)
89 yfit4=predict(mod4,newdata=xfit)
90 plot(x,y,pch=20,xlab="Mois",ylab="Production de
  ↵ gas",main="Lissage avec Loess en fonction du span")
91 points(xfit,yfit1,type="l",lwd=2,col="red")
92 points(xfit,yfit2,type="l",lwd=2,col="blue")
93 points(xfit,yfit3,type="l",lwd=2,col="forestgreen")
94 points(xfit,yfit4,type="l",lwd=2,col="lightblue")
95 legend("topright",c(paste("span=",c(0.1,0.5,0.75,1))),,
  ↵ lwd=2,lty=1,col=c("red","blue","forestgreen","lightblue"))

96
97
98 #Lissage avec Loess de 0.1#
99 tab=0
100 x=seq(1:35)
101 for (i in 1:length(table[,1])){
102   vec=as.numeric(table[i,-c(1,37,38)])
103   model=loess(vec~x,span=0.1)
104   xfit=seq(from=min(x),to=max(x),length.out=100)
105   tab=rbind(tab,predict(model,newdata=xfit))
106 }
107 matplot(t(tab[-1,]), type="l", col=Classification.Expert, lty=1,
  ↵ xlab="Mois", ylab="Production de gas", main="Lissage Loess
  ↵ avec span de 0.1")
108 legend("topright", c("Bonne", "Moyenne", "Mauvaise"), col =
  ↵ c("red","green","black"), text.col = "black", lty = 1, merge
  ↵ = TRUE, bg = "gray90")

109
110
111 #Lissage avec Loess de 0.5#
112 tab=0
113 x=seq(1:35)
114 for (i in 1:length(table[,1])){
115   vec=as.numeric(table[i,-c(1,37,38)])
116   model=loess(vec~x,span=0.5)
117   xfit=seq(from=min(x),to=max(x),length.out=100)
118   tab=rbind(tab,predict(model,newdata=xfit))
119 }
120 matplot(t(tab[-1,]), type="l",col=Classification.Expert, lty=1,
  ↵ xlab="Mois", ylab="Production de gas", main="Lissage Loess
  ↵ avec span de 0.5")

```

```

121 legend("topright", c("Bonne", "Moyenne", "Mauvaise"), col =
122   ↪  c("red","green","black"), text.col = "black", lty = 1, merge
123   ↪  = TRUE, bg = "gray90")
124
125 #Lissage avec Loess de 0.75#
126 tab=0
127 x=seq(1:35)
128 for (i in 1:length(table[,1])){
129   vec=as.numeric(table[i,-c(1,37,38)])
130   model=loess(vec~x,span=0.75)
131   xfit=seq(from=min(x),to=max(x),length.out=100)
132   tab=rbind(tab,predict(model,newdata=xfit))
133 }
134 matplot(t(tab[-1,]), type="l", col=Classification.Expert, lty=1,
135   ↪  xlab="Mois", ylab="Production de gas", main="Lissage Loess
136   ↪  avec span de 0.75")
137 legend("topright", c("Bonne", "Moyenne", "Mauvaise"), col =
138   ↪  c("red","green","black"), text.col = "black", lty = 1, merge
139   ↪  = TRUE, bg = "gray90")
140
141 #Lissage avec Loess de 1.5#
142 tab=0
143 x=seq(1:35)
144 for (i in 1:length(table[,1])){
145   vec=as.numeric(table[i,-c(1,37,38)])
146   model=loess(vec~x,span=1.5)
147   xfit=seq(from=min(x),to=max(x),length.out=100)
148   tab=rbind(tab,predict(model,newdata=xfit))
149 }
150 matplot(t(tab[-1,]), type="l", col=Classification.Expert, lty=1,
151   ↪  xlab="Mois", ylab="Production de gas", main="Lissage Loess
152   ↪  avec span de 1.5")
153 legend("topright", c("Bonne", "Moyenne", "Mauvaise"), col =
154   ↪  c("red","green","black"), text.col = "black", lty = 1, merge
155   ↪  = TRUE, bg = "gray90")
156
157 #Lissage span 1.5 puis régression polynomiale de degré 3#
158 tab=0
159 tabis=0
160 x=seq(1:35)
161 for (i in 1:length(table[,1])){
162   vec=as.numeric(table[i,-c(1,37,38)])
163   model=loess(vec~x,span=1.5)
164   xfit=seq(from=min(x),to=max(x),length.out=35)
165   tab=rbind(tab,predict(model,newdata=xfit))
166   vecbis=tab[i+1,]
167   modelbis=lm(vecbis~x+I(x^2)+I(x^3))
168   tabis=rbind(tabis,fitted(modelbis))
169 }
170

```

```

161 }
162 matplot(t(tabis[-1,]), type="l", col=Classification.Expert,
163   lty=1, xlab="Mois", ylab="Production de gas", main="Lissage
164   de span 1.5 puis régression polynomiale de degré 3")
165 legend("topright", c("Bonne", "Moyenne", "Mauvaise"), col =
166   c("red","green","black"), text.col = "black", lty = 1, merge
167   = TRUE, bg = "gray90")
168
169
170 #Lissage span 1.5 puis régression exponentielle#
171
172 tab=0
173 tabis=0
174 x=seq(1:35)
175 for (i in 1:length(table[,1])){
176   vec=as.numeric(table[i,-c(1,37,38)])
177   model=loess(vec~x,span=1.5)
178   xfit=seq(from=min(x),to=max(x),length.out=35)
179   tab=rbind(tab,predict(model,newdata=xfit))
180   vecbis=tab[i+1,]
181   a_start <- 80
182   b_start <- -2*log(2)/a_start
183   modelbis=nls(vec~a*exp(b*x),start=list(a=a_start,b=b_start))
184   tabis=rbind(tabis,fitted(modelbis))
185 }
186 matplot(t(tabis[-1,]), type="l", col=Classification.Expert,
187   lty=1, xlab="Mois", ylab="Production de gas", main="Lissage
188   de span 1.5 puis régression exponentielle")
189 legend("topright", c("Bonne", "Moyenne", "Mauvaise"), col =
190   c("red","green","black"), text.col = "black", lty = 1, merge
191   = TRUE, bg = "gray90")

```

```

1 ## ## Regression exponentielle et modèle de prédiction ## ##
2 ## Chargement des données
3 donnees_puit
4   =read.csv('FW_Donnees_Puits.csv',header=TRUE,sep=";")
5
6 a_start = 80
7 b_start = -2*log(2)/a_start
8 len_exp = length(donnees_puit[1,-c(1,37,38)])
9 x<-seq(1,len_exp,1)
10 color = c('blue2','red1','lawngreen')
11
12 ## Modèle exponentiel
13 for (i in 1:length(donnees_puit[,1])){
14   vec <-as.numeric(donnees_puit[i,-c(1,37,38)])
15   model <- nls(vec~a*exp(b*x),start=list(a=a_start,b=b_start))
16   plot(c(0,len_exp), c(0,max(donnees_puit[i,-c(1,37,38)])),
16       col='white', xlab='Mois', ylab='Production')

```

```

17  #legend(20,500, c('Bad', 'Good', 'Medium', 'Fitted Curve', '95%
18  #confidence interval'),
19  #c('blue2', 'red1', 'lawngreen', 'orange1', 'cyan2'))
20  legend(c('Good', '95% confidence interval', 'Fitted Curve'),
21  col=c('red','black','orange1'),lty=1:2, x='topright')
22  title('Ajustement par exponentielle et intervalle de
23  confiance')
24  if (donnees_puit[i,37]=='Good'){
25    conf_int <- confint(model,level = 0.9)
26    preds_upr <- conf_int[1]*exp(conf_int[2]*x)
27    preds_lwr <- conf_int[3]*exp(conf_int[4]*x)
28    polygon(c(rev(x-1), x-1), c(rev(preds_lwr), preds_upr), col
29    = 'grey80', border = NA)
30    points(x-1,vec,col='red',pch=20)
31    lines(x-1,predict(model),col='orange1',lwd=2)
32    lines(x-1,preds_upr, pch=22, lty=2,col='black', lwd=1.2)
33    lines(x-1,preds_lwr, pch=22, lty=2,col='black', lwd=1.2)
34    print(donnees_puit[i,37])
35    readkey()
36  }
37 }

## Modèle exponentiel - courbes seules

k_list = c()
categorie_list = c()

plot(c(0,len_exp),c(0,430),col='white', xlab="Mois",
      ylab="Production Simulée")
legend(x='topright', c('Bad', 'Good', 'Medium'),
      c('blue2', 'red1', 'lawngreen'))
title('Ajustement par exponentielle')
for (i in 1:length(donnees_puit[,1])){
  vec <-as.numeric(donnees_puit[i,-c(1,37,38)])
  model <- nls(vec~a*exp(b*x),start=list(a=a_start,b=b_start))
  k_list = rbind(k_list, as.numeric(coef(model)))
  categorie_list =
  rbind(categorie_list,as.numeric(donnees_puit[i,37]))
  if (predict(model)[1]>0 && predict(model)[1]<1050){
    lines(x-1, predict(model),
    col=color[as.numeric(donnees_puit[i,37])], lwd=1)
  }
}

####Question 4####

## Modèle exponentiel - k0 vs k1

```

```

57 plot(k_list[1,1],k_list[1,2], col='white',
58   ↪  #color[categorie_list[1]],
59   xlim=c(min(k_list[,1])-5, 450),
60   ylim=c(min(k_list[,2]), max(k_list[,2]+0.1)),
61   xlab="k0", ylab="k1",
62   pch = 20,
63   cex=1.5)
64 grid()
65 legend(x='topright',-0.25, c('Bad','Good','Medium'),
66   ↪  c('blue2','red1','lawngreen'))
67 title('k1 fonction de k0')
68 for (i in 1:length(k_list[,1])){
69 #  points(k_list[i,1],k_list[i,2],
70   ↪  col=color[categorie_list[i]],pch = 20,cex=1.5)
71   text(k_list[i,1],k_list[i,2],
72   ↪  labels=i,col=color[categorie_list[i]],pch = 20,cex=0.75)
73 }
74
75 #names(color) = c('bad','Good','medium')
76 #color = as.data.frame(t(color))
77
78 ## Multiple Logistic regression
79 install.packages('foreign')
80 install.packages('nnet')
81 install.packages('ggplot2')
82 install.packages('reshape2')
83 require('foreign')
84 require('nnet')
85 require('ggplot2')
86 require('reshape2')
87
88 ## First try
89 install.packages('caret')
90 install.packages('e1071', dependencies=TRUE)
91 library('caret')
92
93 ml <- data.frame(a=k_list[,1],
94                     b=k_list[,2],
95                     qual =
96   ↪  donnees_puit[,37])#as.numeric(categorie_list[,1]))
97
98 mod <- multinom(qual ~ a + b, ml)
99
100 predict(mod)
101 predict(mod,ml,"probs")
102
103 confusionMatrix(donnees_puit[,37], predict(mod))
104
105
106 # good not good

```

```

102 not_good = which(predict(mod) != donnees_puit[,37])
103 donnees_puit[not_good,37]
104 good = c('bad', 'bad', 'bad', 'medium', 'medium', 'medium',
105   ↪ 'Good',
106   'medium', 'medium', 'medium',
107   'Good', 'bad', 'Good', 'medium', 'medium')
108
109 ## 73 -> medium try
110 ml2 <- data.frame(a=k_list[,1],
111   b=k_list[,2],
112   qual =
113   ↪ donnees_puit[,37])#as.numeric(categorie_list[,1]))
114
115 ml2$qual[73] = 'medium'
116 mod2 <- multinom(qual ~ a + b, ml2)
117 mod2$AIC
118 predict(mod2)
119 predict(mod2,ml2,"probs")
120
121 ## best AIC
122 AIC_test = c()
123 for (i in 1:length(not_good)){
124
125 ml2 <- data.frame(a=k_list[,1],
126   b=k_list[,2],
127   qual =
128   ↪ donnees_puit[,37])#as.numeric(categorie_list[,1]))
129
130 ml2$qual[not_good[i]] = good[i]
131 mod2 <- multinom(qual ~ a + b, ml2)
132 AIC_test = rbind(AIC_test, mod2$AIC)
133 }
134 sorted_AIC = sort(AIC_test, decreasing = FALSE, index.return =
135   ↪ TRUE)
136
137 ## Final loess exponentiel
138 k_list = c()
139 categorie_list = c()
140 for (i in 1:length(donnees_puit[,1])){
141   vec <-as.numeric(donnees_puit[i,-c(1,37,38)])
142   mod_loess = loess(vec~x,span=2)
143   new_vec = predict(mod_loess, x)
144   new_vec[new_vec<0] = 0
145   model <-
146   ↪ nls(new_vec~a*exp(b*x),start=list(a=a_start,b=b_start))
147   k_list = rbind(k_list, as.numeric(coef(model)))

```

```
146   categorie_list =
147   ↪  rbind(categorie_list,as.numeric(donnees_puit[i,37]))
148 }
149 ml <- data.frame(a=k_list[,1],
150                   b=k_list[,2],
151                   qual =
152                   ↪ donnees_puit[,37])#as.numeric(categorie_list[,1]))
153 mod_final <- multinom(qual ~ a + b, ml)
154 mod_final$AIC
155
156 predict(mod_final)
157 predict(mod_final,ml,"probs")
158
159 confusionMatrix(donnees_puit[,37], predict(mod_final))
```