Séance 3 : Régression Non Linéaire avec R

Exercice 0.1 *Modèle exponentiel*

Procédé en batch dans un fermenteur

L'objectif d'un "procédé en batch" de génie fermentaire est de déterminer les caractéristiques cinétiques d'un microorganisme en particulier son taux de croissance μ. Pour cela, on va ensemencer un bioréacteur avec (entre autre) une certaine concentration de microorganismes et de substrat dont on va mesurer l'évolution au cours du temps. Des mesures de densité optique seront régulièrement effectuées à l'aide d'un spectrophotomètre. Les échantillons prélevés seront dilués afin de rester dans la zone de linéarité du spectrophotomètre, zone pour laquelle la densité optique est proportionnelle à la concentration. Après différentes calibrations, on a obtenu les concentrations suivantes

Temps t_i (h)	*Biomasse x_i (g/L)*
0	*0,11*
1	*0,15*
2	*0,19*
3	*0,26*
4	*0,33*
5	*0,47*
6	*0,63*
7	*0,95*
8	*1,36*

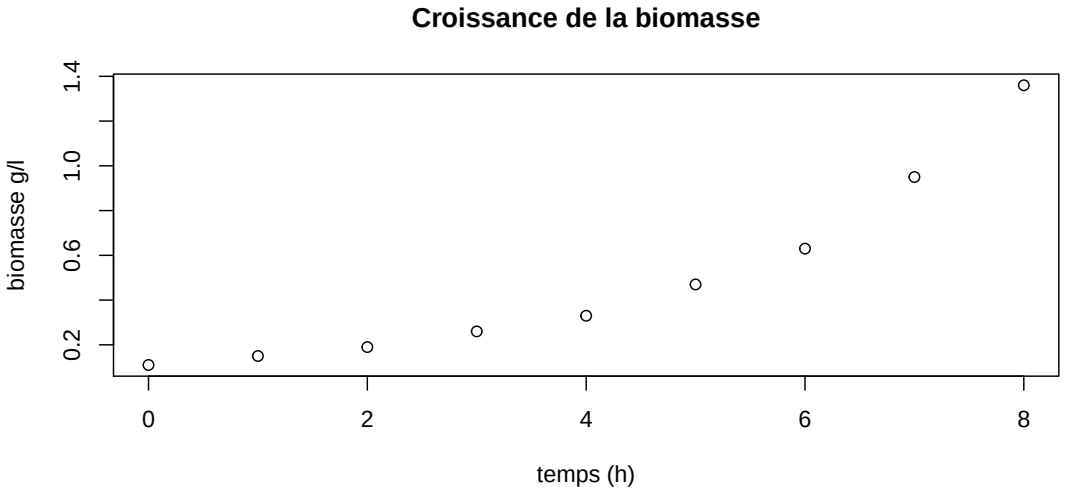
On veut modéliser l'évolution de la concentration x sous les conditions "batch" par l'équation différentielle suivante :

$$\begin{cases} \dfrac{dx}{dt} &= \mu x \\ x(0) &= x_0 \end{cases}$$

où μ le taux de croissance de la biomasse est supposé constant au cours du temps et x_0 est la concentration initiale (ce sont des constantes strictement positives).

1. *Lire le fichier de données chemostat.csv dans la data frame chemo. Tracer le nuage de points expérimentaux (t_i, x_i).*

```
> chemo <- read.csv2("chemostat.csv")
> attach(chemo)# pour simplifier les noms des variables
> plot(ti,xi,main="Croissance de la biomasse", xlab="temps (h)",ylab="biomasse g/l")
```



2. *Résoudre l'équation différentielle (où μ est une constante):* $\dfrac{dx}{dt} = \mu x(t) \quad x(0) = x_0$.

$$x(t) = x_0 exp(\mu t)$$

3. *On veut déterminer le taux de croissance μ à partir des données (t_i, x_i). Afin de ne pas donner une importance exagérée à x_0, on considèrera que x_0 est aussi un paramètre à déterminer (sinon la courbe passera exactement par ce point $(t_0 = 0, x_0 = 0.11)$ alors qu'elle ne passera qu'à proximité des autres).Quelle fonction S veut-on alors minimiser ? Ce problème est-il linéaire par rapport au(x) paramètre(s) ?*
On cherche le minimum de la fonction S définie par

$$S(x_0, \mu) = \sum_{i=1}^{n} (x_0 exp(\mu t_i) - x_i)^2$$

Le problème n'est pas linéaire: la recherche des points critiques ne conduit pas à un système linéaire. On va devoir utiliser la fonction nls pour le résoudre. Mais cette fonction utilise un procédé itératif et nécessite donc d'avoir des valeurs initiales des paramètres pour être mise en oeuvre.

4. *Linéarisation du problème: recherche des valeurs initiales pour (x_0, μ).*

   (a) *On définit $z(t) = \ln(x(t))$. Montrer que $z(t) = a_1 + a_2 t$ (ie z suit un modèle linéaire par rapport à t, une droite). Exprimer μ et x_0 en fonction de a_1 et a_2.*

$$z(t) = \ln(x_0 exp(\mu t)) = \ln(x_0) + \mu t$$

   On a donc $a_1 = \ln(x_0)$ et $a_2 = \mu$ et par conséquent $x_0 = exp(a_1)$ et $\mu = a_2$. On va maintenant chercher les paramètres (a_1, a_2) qui minimisent la fonction

$$S_l(a_1, a_2) = \sum_{i=1}^{n} (a_1 + a_2 t_i - ln(x_i))^2$$

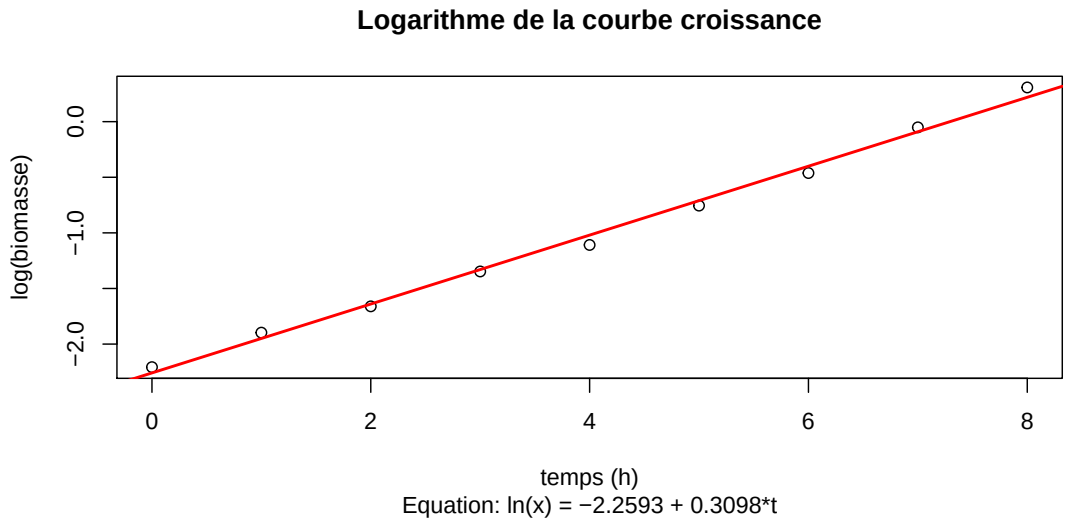
   avec la fonction lm puisque nous nous sommes ramenés à un problème linéaire.

   (b) *Déterminer les paramètres (a_1, a_2) puis en déduire (x_0, μ).*

```
> lin=lm(log(xi)~ti,data=chemo)
> summary(lin)

Call:
lm(formula = log(xi) ~ ti, data = chemo)

Residuals:
     Min       1Q   Median       3Q      Max
-0.08847 -0.04460 -0.01712  0.05198  0.08861

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.259256   0.039685  -56.93 1.35e-10 ***
ti           0.309766   0.008336   37.16 2.66e-09 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06457 on 7 degrees of freedom
Multiple R-squared:  0.995,        Adjusted R-squared:  0.9942
F-statistic:  1381 on 1 and 7 DF,  p-value: 2.656e-09

> a1=coef(lin)[1]
> a2=coef(lin)[2]
> x0=exp(a1)
> mu=a2
> print(c(a1,a2,x0,mu))

(Intercept)          ti (Intercept)          ti
 -2.2592562   0.3097660   0.1044281   0.3097660
```

   Affichage et contrôle des résultats

```
> eq = paste0("Equation: ln(x) = ", round(a1,4)," + ", round(a2,4),"*t" )
> plot(ti,log(xi),main="Logarithme de la courbe croissance",sub=eq,xlab="temps (h)",ylab="log(biomas
> abline(lin,col="red" ,lwd = 2)
```



On dispose donc de valeurs approchées de (x_0, μ) obtenues en linéarisant le problème. On revient à notre problème initial.

5. *Utilisation de la fonction nls: recherche des paramètres (x_0, μ).*

   (a) *Définir une liste contenant les valeurs x_0 et μ déterminées précedemment.*

   ```
   > initialisation=list(a=x0,b=mu)
   ```

   Cette liste contient les valeurs de départ que nous allons utiliser avec la fonction nls pour déterminer plus précisément (x_0, μ).

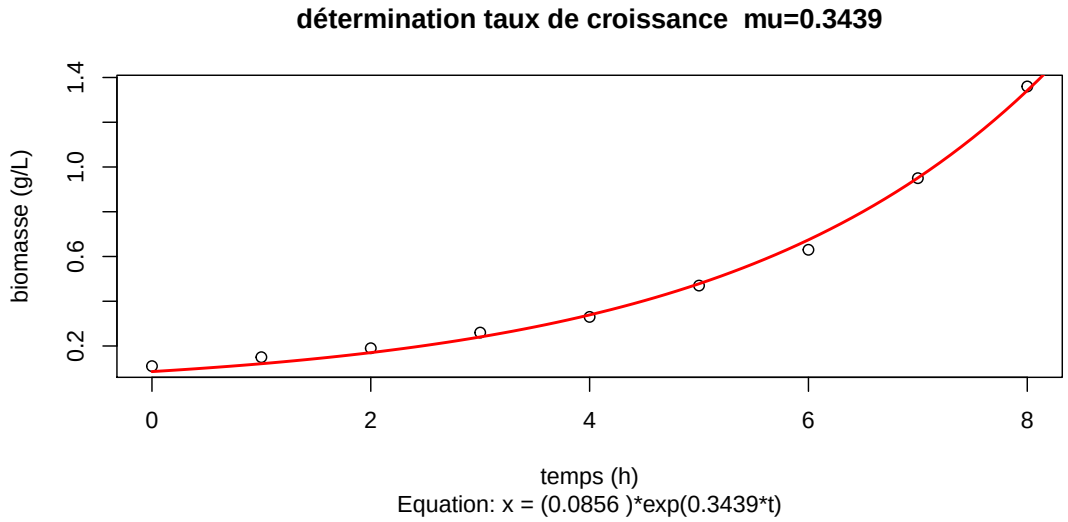
   (b) *Déterminer une estimation de (x_0, μ) par la méthode des moindres carrés à l'aide de la fonction nls (Nonlinear Least Squares) de R studio. Afficher les résultats de nls avec la commande summary. Que signifie "Number of iterations to convergence " ?*

   ```
   > modele <- nls(xi~ a*exp(b*ti), data = chemo, start = initialisation)
   > summary(modele)

   Formula: xi ~ a * exp(b * ti)

   Parameters:
     Estimate Std. Error t value Pr(>|t|)
   a 0.085647   0.006086   14.07 2.17e-06 ***
   b 0.343939   0.009952   34.56 4.40e-09 ***
   ---
   Signif. codes:
   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

   Residual standard error: 0.02583 on 7 degrees of freedom

   Number of iterations to convergence: 4
   Achieved convergence tolerance: 6.075e-06
   ```

   Pour utiliser nls, il faut donner des valeurs de départ (start) pour les paramètres, valeurs contenues dans initialisation. Le procédé est itératif: "Number of iterations to convergence " donne le nombre d'itérations nécessaires pour converger vers une valeur acceptable.

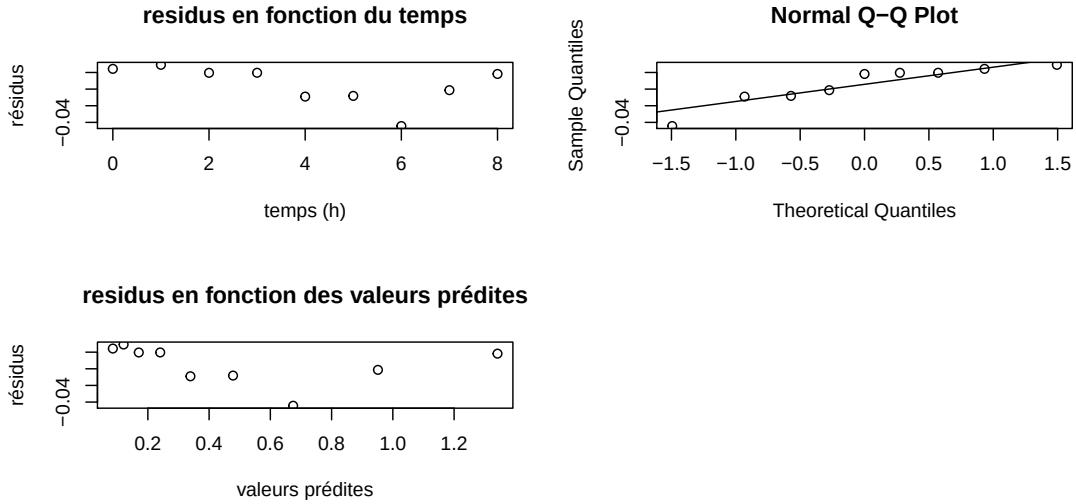
   (c) *Tracer le nuage de points (t_i, x_i) et la courbe théorique obtenue ainsi que son équation. Penser à contrôler les résidus.*

```
> x0=coef(modele)[1];mu=coef(modele)[2]
> eq = paste0("Equation: x = (", round(x0,4)," )*exp(", round(mu,4),"*t)" )# sous titre equation
> titre=paste0("détermination taux de croissance  mu=",round(mu,4))#titre du graphique
> plot(ti,xi,main=titre,sub=eq,xlab="temps (h)",ylab="biomasse (g/L)")#tracé du nuage de points
> curve(x0*exp(mu*x),from=0,to=10,col="red" ,lwd = 2,add=TRUE)# ajout courbe prédite
```



Contrôle des résidus: on ne peut pas utiliser plot(modele) car cette fonction n'est implémentée que suite à l'utilisation de lm (uniquement pour les modèles linéaires). Mais on peut tracer les résidus en fonction des valeurs prédites ou en fonction du temps, vérifier la normalité des résidus,...

```
> layout(matrix(1:4,2,2))# fenetre graphique coupée en 4
> plot(ti,residuals(modele),main="residus en fonction du temps",xlab="temps (h)",ylab="résidus")
> plot(fitted(modele),residuals(modele),main="residus en fonction des valeurs prédites",xlab="valeur
> # manque de points mais les residus augmentent-ils avec t ou x=>hétéroscédasticité ?
> # residus normaux ? si les residus suivent une loi normale, les points doivent être alignés.
> qqnorm(residuals(modele))
> # si la droite passe par l'origine, les residus sont bien de moyenne nulle
> qqline(residuals(modele))
> # residus studentisés: non implémentés avec nls
```



On ne peut pas utiliser plot(modele) lorsque l'on a utilisé nls, ni rstudent. Beaucoup de fonctions ne sont pas implémentées lorsque le problème est non linéaire. On remarque que les résidus ne semblent pas aléatoirement répartis (il

serait souhaitable d'avoir davantage de points).

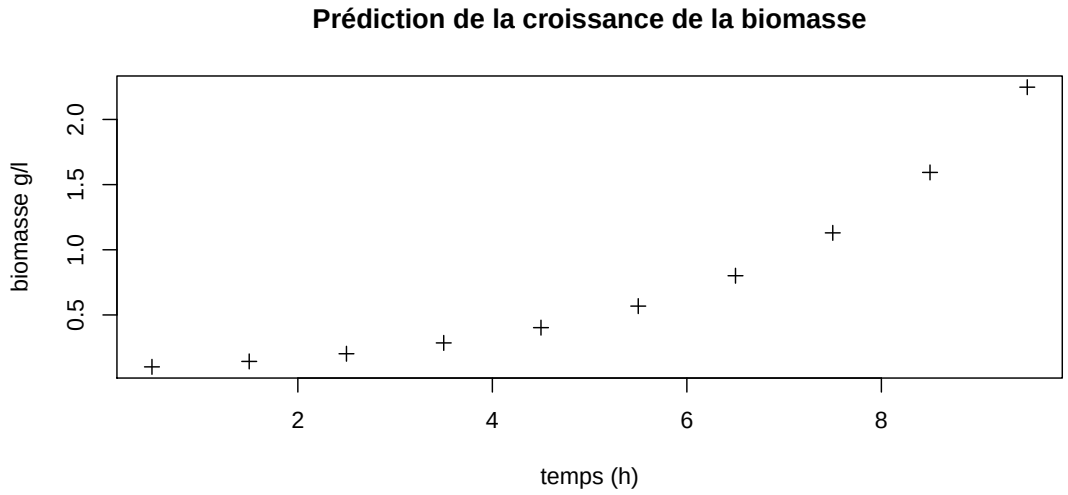
(d) *Prédire les valeurs de x pour $t = 0.5$ à $t = 7,5h$ par pas de 1.*

```
> newt<-seq(0.5,9.5,1)
> pc=predict(modele,data.frame(ti= newt), level = 0.95, interval = "confidence")
> print(pc)
```

```
 [1] 0.1017181 0.1434726 0.2023671 0.2854373 0.4026072
 [6] 0.5678746 0.8009830 1.1297807 1.5935475 2.2476872
```

La fonction predict donne uniquement les valeurs prédites. Les intervalles de confiance et de prédiction ne sont pas implémentés en non linéaire (cf help).

```
> layout(1)
> plot( pc ~ newt, type='p',pch=3,main="Prédiction de la croissance de la biomasse", xlab="temps (h)
```



Exercice 0.2 Détermination du Kla d'un fermenteur

La capacité d'oxygénation d'un fermenteur est déterminée en suivant la cinétique de transfert d'oxygène en absence de microorganismes. On rappelle que la concentration en oxygène dissous, O_{2L}, dans un fermenteur en absence de microorganismes est donnée par l'équation différentielle

$$\frac{dO_{2L}}{dt} = Kla(O^*_{2L} - O_{2L}) \quad O_{2L}(0) = O_{2L_0}$$

où Kla est le coefficient de transfert (gaz-liquide) de l'oxygène, supposé constant, O^*_{2L} la concentration saturante d'oxygène liquide à 30°C. On suppose que $O^*_{2L} = 100$. A l'aide de données expérimentales ci-dessous, on veut déterminer Kla et O_{2L_0}.

t_i *temps (s)*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*100*	*130*
O_{2L_i} *(mgl^{-1})*	*27*	*46*	*60*	*67,5*	*79*	*85*	*86*	*94*	*98*

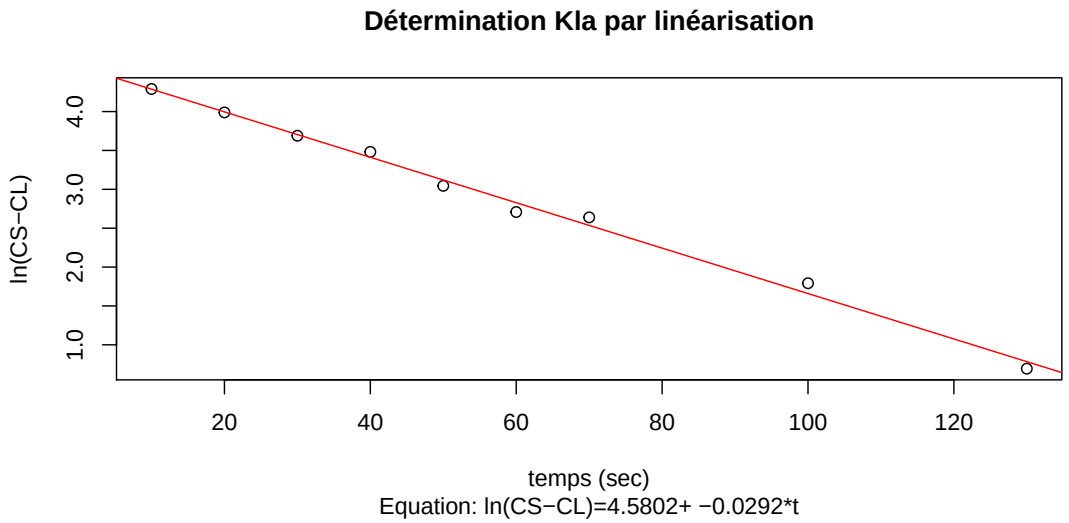
On veut déterminer les paramètres Kla et O_{2L_0}.

1. *Résoudre l'équation différentielle vérifiée par O_{2L}. (On montrera que $O_{2L}(t) = (O_{2L}(0) - O^*_{2L})e^{-Klat} + O^*_{2L}$).*
2. *Pour déterminer Kla et O_{2L_0}, quelle fonction S_1 cherche-t-on à minimiser ? On cherche à minimiser la fonction*

$$S(Kla, O_{2L_0}) = \sum_{i=1}^{n} (O_{2L}(t_i) - O_{2L_i})^2$$

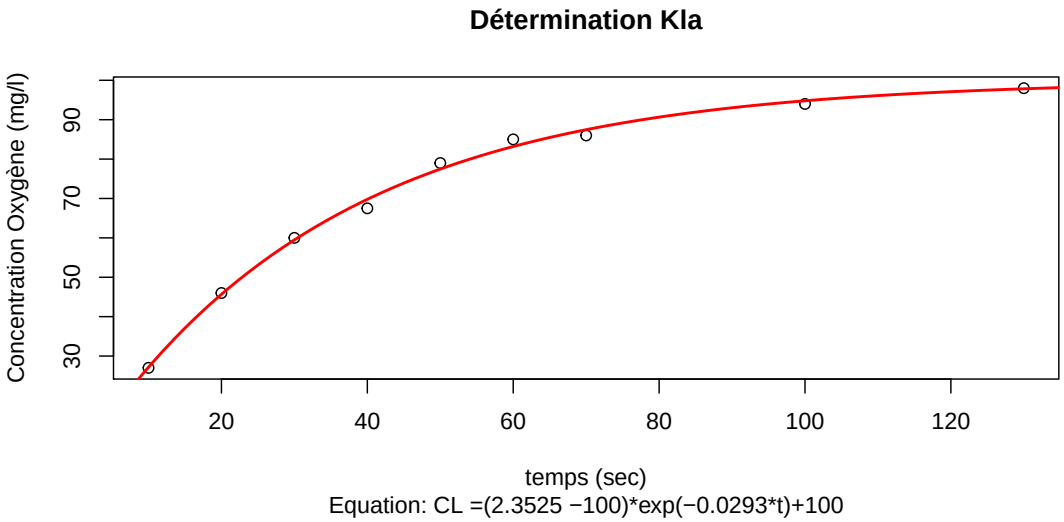
3. *Déterminer des valeurs initiales Kla et O_{2L_0} en linéarisant le problème.*

```
> # Création des données
> t <- c(10,20,30,40,50,60,70,100,130) # abscisse, variable explicative
> CL <- c(27,46,60,67.5,79,85,86,94,98)  # ordonnée, variable expliquée
> # ****************************
> # détermination des valeurs initiales $Kla$ et $O_{2L_0}$ par linéarisation et régression
> donnees <- data.frame(t, CL)
> CS<-100;z=log(CS-CL)
> modlin <- lm(z~t, data = donnees)
> C00 <- -exp(coef(modlin)[1])+CS; kla0 <- -coef(modlin)[2]
> eq = paste0("Equation: ln(CS-CL)=", round(coef(modlin)[1],4),"+ ", -round(kla0,4),"*t" )# sous titre
> titre="Détermination Kla par linéarisation";#titre du graphique
> #tracé du nuage de points
> plot(t, z,main=titre,sub=eq,xlab="temps (sec)",ylab="ln(CS-CL)")# nuage
> abline(modlin,col="red")
>
```



4. *Déterminer Kla et O_{2L_0} à l'aide de la fonction nls (Nonlinear Least Squares) de R studio.*

```
> modele <- nls(CL~(C0-CS)*exp(-kla*t)+CS, data = donnees, start = list(C0 = C00, kla = kla0))
> b0 <- coef(modele)[1]
> kla <- coef(modele)[2]
> eq = paste0("Equation: CL =(", round(b0,4)," -100)*exp(-", round(kla,4),"*t)+100" )# sous titre
> titre="Détermination Kla";#titre du graphique
> #tracé du nuage de points
> plot(t, CL,main=titre,sub=eq,xlab="temps (sec)",ylab="Concentration Oxygène (mg/l)")# nuage
> curve((b0-CS)*exp(-kla*x)+CS,from=0,to=150,col="red" ,lwd = 2,add=TRUE)# ajout courbe prédite
```



5. *Etudier la qualité de la prédiction.*

```
> layout(matrix(1:4,2,2))# fenetre graphique coupée en 4
> plot(donnees$t,residuals(modele))# résidus en fonction du temps
> plot(fitted(modele),residuals(modele))
> qqnorm(residuals(modele))
> qqline(residuals(modele))
```