TP : station de comptage de trafic (jour 1)

(Correction)

1 Introduction

On dispose d'un jeu de données `S21.csv` provenant d'une station de comptage de trafic routier (boucle magnétique), fournie par la société SISTeMA ITS. On cherche à étudier la structure des données fournies. Les mesures ont été faites du 01/01/2010 au 31/01/2012.

Le script `sistema.r` a permis de transformer cette liste de mesures en une table `S21.txt` dont chaque ligne donne pour une journée donnée le nombre de véhicules comptés par tranches de 6 minutes. Certaines données de comptage sont manquantes (`NA`).

On demande pour ce TP de fournir un petit rapport d'analyse préliminaire des données. Pour cela on s'appuiera sur

— le logiciel R installé sur les machines, ainsi que le paquet « fda » ;

— le fichier de données `S21.txt`, dont les lignes représentent des journées entières et les colonnes les mesures toutes les 6 minutes. Au total la table comporte 761 lignes et 240 colonnes.

On donne ci-dessous une liste de questions pour guider le travail. On demandera de répondre de manière brève mais aussi précise que possible en donnant :

— le code utilisé,

— le résultat (données, graphiques),

— éventuellement un commentaire et/ou des explications sur ce que vous avez fait.

2 Examen rapide des données

En fait, contrairement à ce que mentionne la documentation du paquet `fda`, les valeurs manquantes ne sont pas gérées correctement. On va donc se limiter aux lignes dont les données sont complètes dans un premier temps. On charge les données et on retire les individus ayant des données manquantes de la manière suivante :

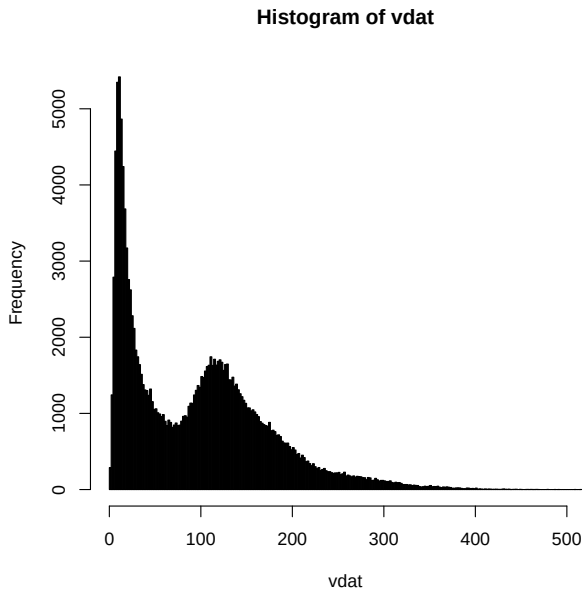
```
> # Il est preferable de conserver les donnees dans une matrice
> dat0 = as.matrix(read.table("S21.txt", head=T))
> na_count = rowSums(is.na(dat0))
> dat = dat0[na_count == 0,]
> # La variable heures contient le temps dans la journee en heures
> # toutes les 6 minutes (240 points).
> heures = c(0:239)/10
```

Dans cette section et la suivante on utilisera le tableau `dat`.

Question 1: *vérifier la répartition des valeurs pour les débits. Expliquer pourquoi on peut appliquer la méthode de stabilisation de la variance pour processus de Poisson. Est-ce que cela améliore la queue de distribution ?*

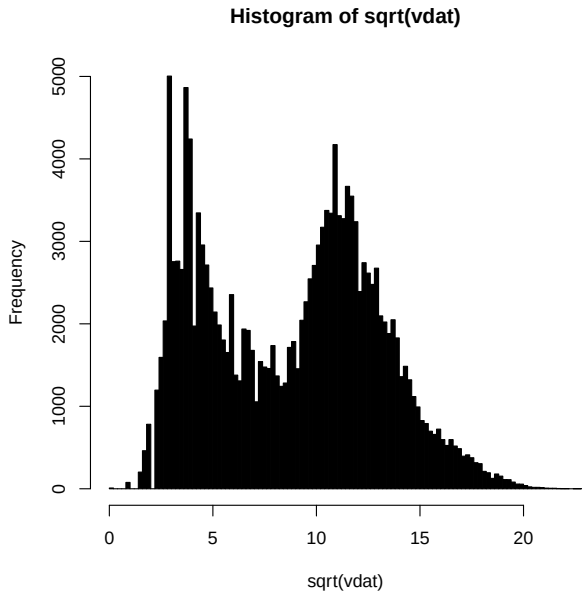
La répartition des débits a une queue assez lourde. Elle est très asymétrique.

```
> vdat = as.vector(dat)
> hist(vdat, breaks=200, col=1)
```



On peut essayer de corriger cela en prenant la racine carrée des débits. C'est la technique de stabilisation de la variance vue en cours pour des nombres qui pourraient être des tirages de processus de Poisson. Ici en effet les débits en 6 minutes sont des comptages de nombres de véhicules passés en 6 minutes, bien modélisés par des processus de Poisson. On voit que la distribution est beaucoup plus symétrique.

```
> dat=sqrt(dat)
> hist(sqrt(vdat), breaks=100, col=1)
```

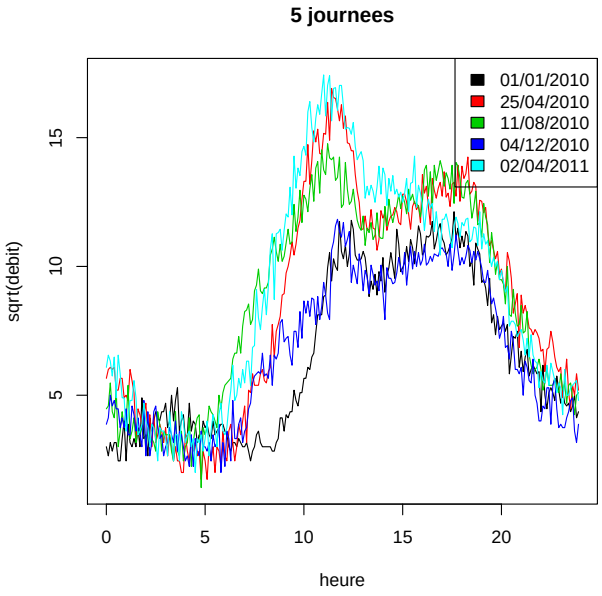


Par la suite, on travaillera sur ces racines carrées (ce qui n'est pas obligatoire ici).

Question 2: *Tracer des courbes pour quelques journées (au moins 5)*

On sélectionne 5 journées arbitrairement : les jours 1, 100, 200, 300 et 400.

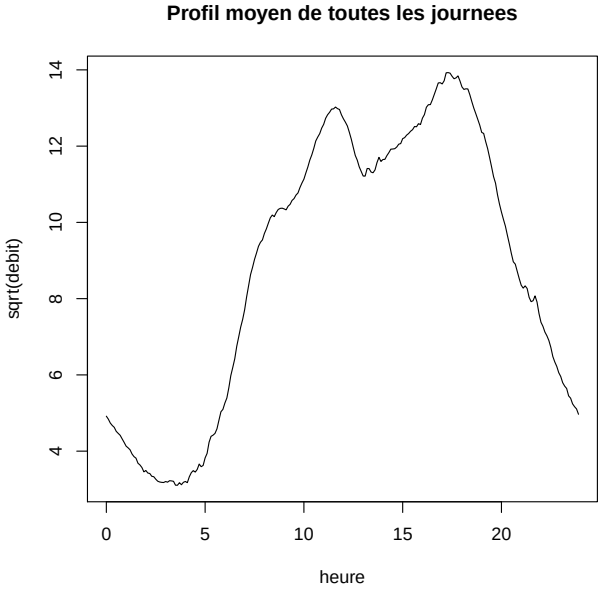
```
> courbes=c(1, 100, 200, 300, 400)
> matplot(heures,t(dat[courbes,]), type='l', lty=1, xlab="heure", ylab="sqrt(debit)", main="5 journe
> legend("topright", rownames(dat[courbes,]), fill=1:6)
```



Question 3: *Tracer la courbe du débit moyen au cours de la journée. Commenter la forme générale de la courbe.*

On utilise la fonction `colMeans` qui calcule exactement la moyenne dont nous avons besoin. On constate que la moyenne empirique a deux modes correspondant à des pics vers 12h et 19h (ce qui est un peu inhabituel).

```
> meandat=colMeans(dat)
> plot(heures, meandat, type='l', xlab="heure", ylab="sqrt(debit)", main="Profil moyen de toutes les
```



3 Lissage de données

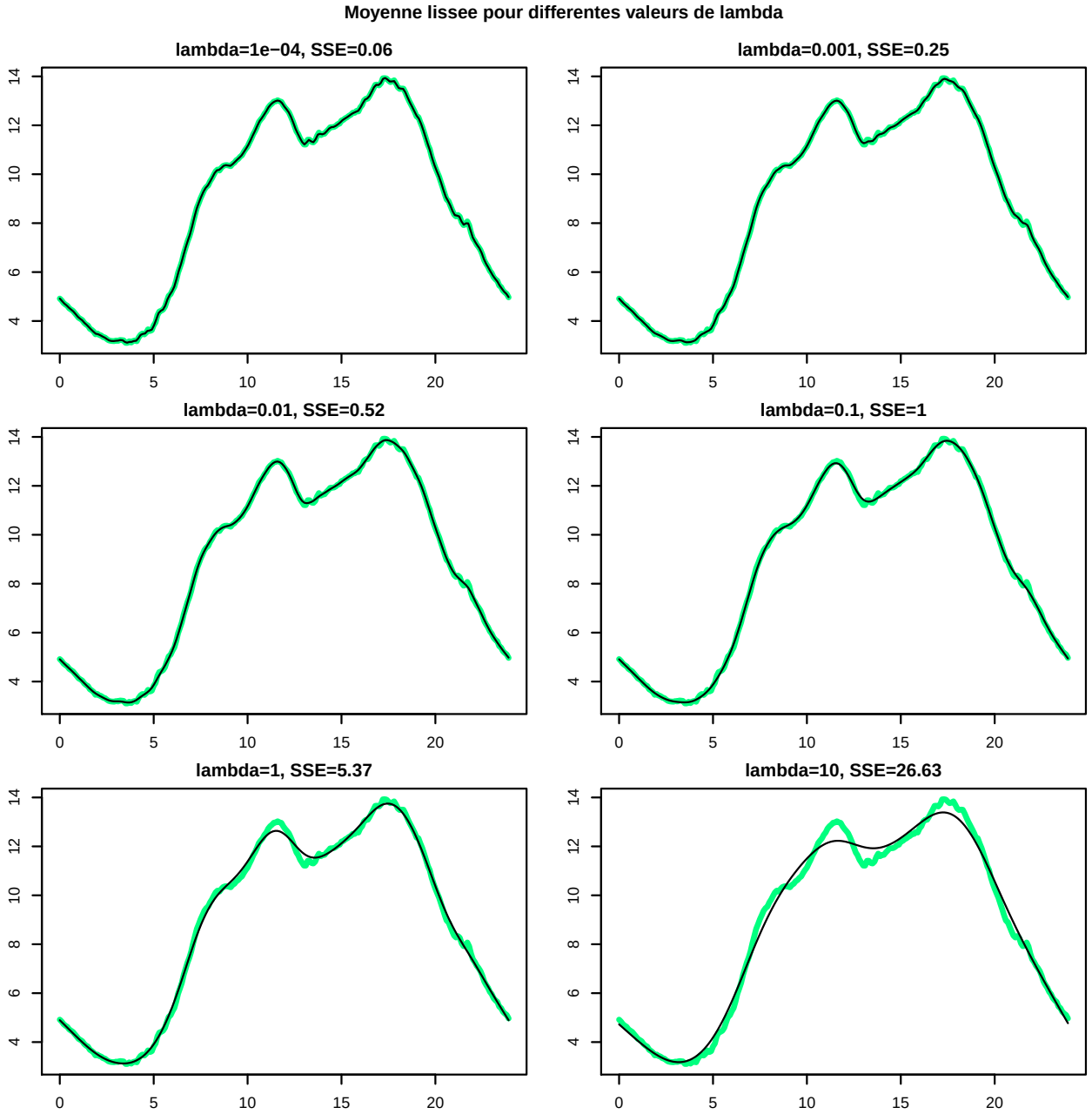
Question 4: *Lissage de la moyenne : essayer avec plusieurs valeurs de λ pour en trouver une qui lisse correctement la moyenne tout en gardant un SSE correct. Tracer la moyenne correspondante*

On crée une base de splines d'ordre 4 avec un spline à chaque tranche de 6 minutes.

```
> require(fda)
> nderiv = 2
> norder = nderiv + 2
> #nbasis = 240 - 2 + norder
> fdnames=list("heure", "jour", "sqrt(debit)")
> basisobj = create.bspline.basis(breaks=heures, norder=norder)
```

On trace les moyennes obtenues pour chacune des valeurs de λ évoluant de 10^{-4} à 10 par pas multiplicatif de 10. On constate que 0.01 est une valeur raisonnable (c'est la première pour laquelle la courbe est lisse).

```
> # Ces commandes permettent de reduire les marges autour des courbes
> save=par(mfrow=c(3,2), cex=0.5, mar=c(2,2,2,2), oma=c(0,0,2,0))
> for (lambda in c(0.0001,0.001, 0.01, 0.1, 1, 10)) {
+   fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=lambda)
+   smoothmeanobj = smooth.basis(argvals=heures, y=meandat, fdParobj, fdnames=fdnames)
+   plot(heures, meandat, col='springgreen', lwd=3, typ='l', main=paste0("lambda=", lambda,", SSE=", round(smoothmeanobj$SSE,2)))
+   dummy=lines(smoothmeanobj)
+ }
> title(main="Moyenne lissee pour differentes valeurs de lambda", outer=TRUE, cex=0.8)
> par(save)
>
```

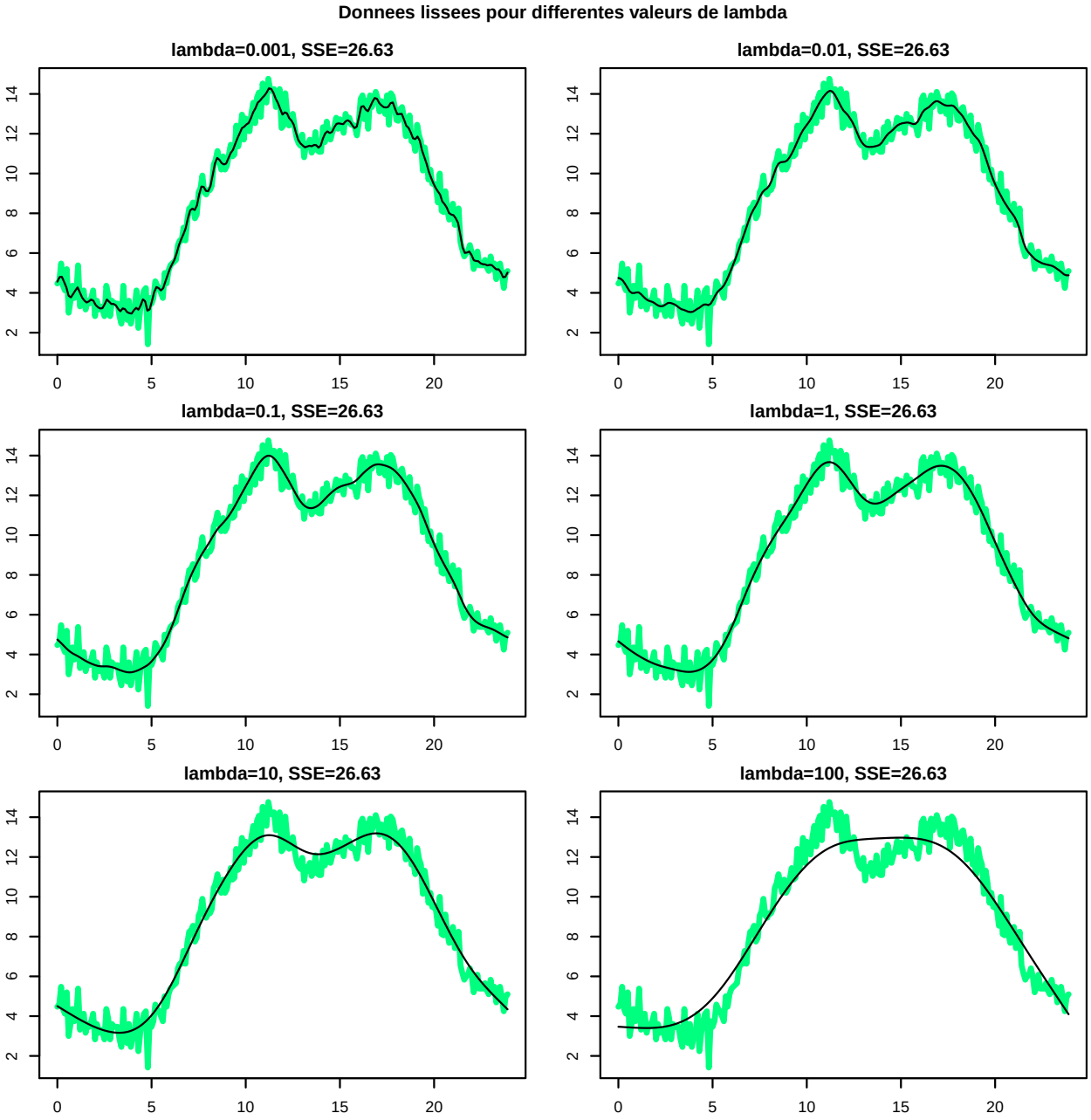


Question 5: *Lissage d'une courbe : choisir un des individus et calculer sa version lissée (on expliquera comment on choisit λ). On fera un graphique joint des données brutes et des données bruitées.*

On choisit arbitrairement la courbe 200. On fait des lissages pour des valeurs de λ évoluant en puissance de 10 : $0.001, 0.01, 0.1, 1, 10, 100$ (on commence plus haut que pour la moyenne car la courbe de départ est moins lisse). On s'arrête là encore dès que la courbe devient assez lisse.

```
> sampledat=dat[200,]
> save=par(mfrow=c(3,2), cex=0.5, mar=c(2,2,2,2), oma=c(0,0,2,0))
> for (lambda in c(0.001, 0.01, 0.1, 1, 10, 100)) {
+   fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=lambda)
```

```
+   smoothobj = smooth.basis(argvals=heures, y=sampledat, fdParobj, fdnames=fdnames)
+   plot(heures, sampledat, col='springgreen', lwd=3, typ='l', main=paste0("lambda=", lambda,", SSE=", round(smoothmeanobj$SSE,2)
+   dummy=lines(smoothobj)
+ }
> title(main="Donnees lissees pour differentes valeurs de lambda", outer=TRUE, cex=0.8)
> par(save)
```



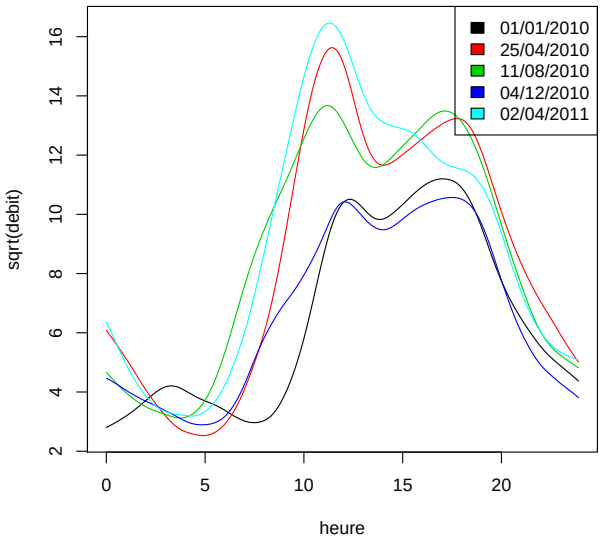
On choisit donc $\lambda = 1$ pour les courbes individuelles. On aurait aussi pu choisir 0.1 qui est un peu moins lisse mais plus précis, on va voir à la question suivante.

```
> lambda=1
> fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=lambda)
```

Question 6: *Lissage de plusieurs courbes : lisser les mêmes courbes qui on été tracées dans la section* **??** *et faire un tracé joint des courbes lissées (pour une valeur raisonnable de λ)*

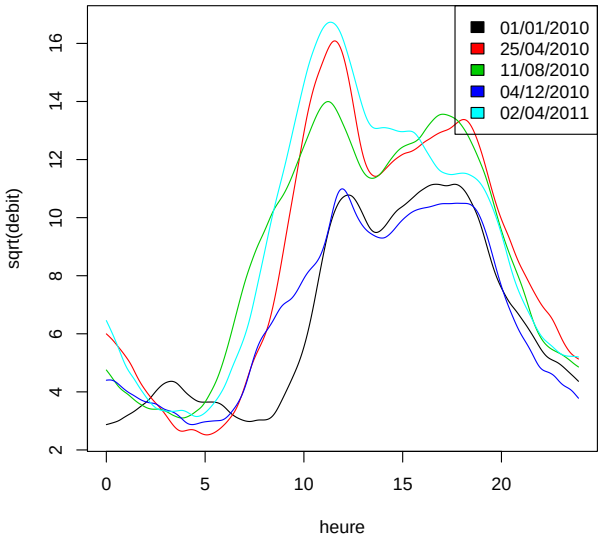
On reprend ici la même valeur $\lambda = 1$ que précédemment. La seule chose à voir ici est que `smooth.basis` peut être utilisé avec un tableau d'individus, à condition de transposer la matrice. On ne met que les courbes lissées pour éviter d'obtenir un graphique illisible.

```
> smoothobj = smooth.basis(argvals=heures, y=t(dat[courbes,]), fdParobj, fdnames=fdnames)
> dummy=plot(smoothobj, lty=1)
> legend("topright", rownames(dat[courbes,]), fill=1:6)
```



On peut aussi essayer la valeur $\lambda = 0.1$ pour comparer sur plusieurs courbes.

```
> lambda=0.1
> fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=lambda)
> smoothobj = smooth.basis(argvals=heures, y=t(dat[courbes,]), fdParobj, fdnames=fdnames)
> dummy=plot(smoothobj, lty=1)
> legend("topright", rownames(dat[courbes,]), fill=1:6)
```



Le résultat est nettement moins lisse, il vaut mieux garder $\lambda = 1$.