TP : station de comptage de trafic (jour 2)

(Correction)

1 Introduction

On réutilise le jeu de données `S21.csv` provenant d'une station de comptage de trafic routier (boucle magnétique), fournie par la société SISTeMA ITS. On cherche aujourd'hui à avancer dans deux directions :

— calcul de l'ACP lissée

— gestion des valeurs manquantes

Comme précédemment, 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.

On partira si besoin des variables suivantes, utilisées dans la correction du TP1 (ou alors, tout autre code équivalent)

```
> # dat0 contient les donnees brutes
> dat0=as.matrix(read.table("S21.txt", head=T))
> heures=c(0:239)/10
> # Vecteur des donnees manquantes par ligne
> na_count = rowSums(is.na(dat0))
> # Quelques courbes choisies arbitrairement
> courbes=c(1, 100, 200, 300, 400)
> # Journees sans donnee manquante
> dat = dat0[na_count == 0,]
> # Si on veut reduire les grandes valeurs
> #dat=sqrt(dat)
> require(fda)
> # Representation fonctionnelle des donnees de trafic comme au TP1
> # On lisse toutes les courbes
> nderiv = 2
> norder = nderiv + 2
> fdnames=list("heure", "jour", "debit")
> basisobj = create.bspline.basis(breaks=heures, norder=norder)
> lambda=0.1
> fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=lambda)
> dat.fd = smooth.basis(argvals=heures, y=t(dat), fdParobj, fdnames=fdnames)$fd
```

2 ACP lissée

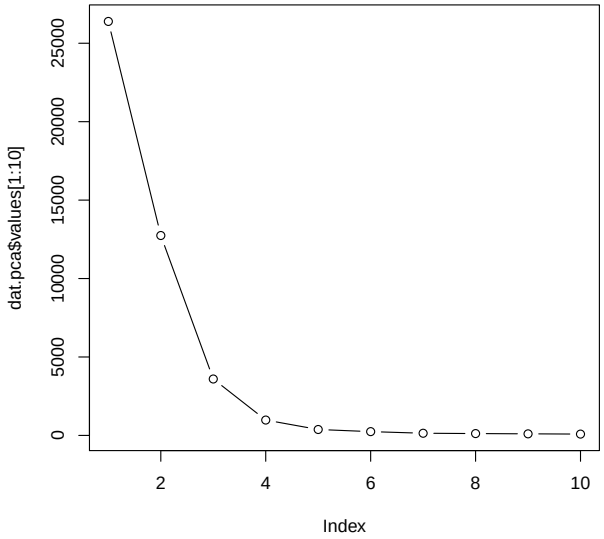
On cherche à faire une ACP lissée des données de trafic (uniquement les journées où toutes les données sont disponibles).

Question 1: *Faire une ACP fonctionnelle des débits. Représenter la courbe de décroissance des valeurs propres : combien d'axes semblent pouvoir être intéressants ?*

Faire une ACP fonctionnelle est assez direct à partir de nos données. Pour le choix de λ, on constate que 0 marche très bien : la courbe de départ est déjà assez lisse.

```
> pca.fdParobj = fdPar(fdobj=basisobj, lambda=0)
> dat.pca = pca.fd(dat.fd, nharm = 8, harmfdPar=pca.fdParobj)
```

```
> plot(dat.pca$values[1:10],type='b')
```

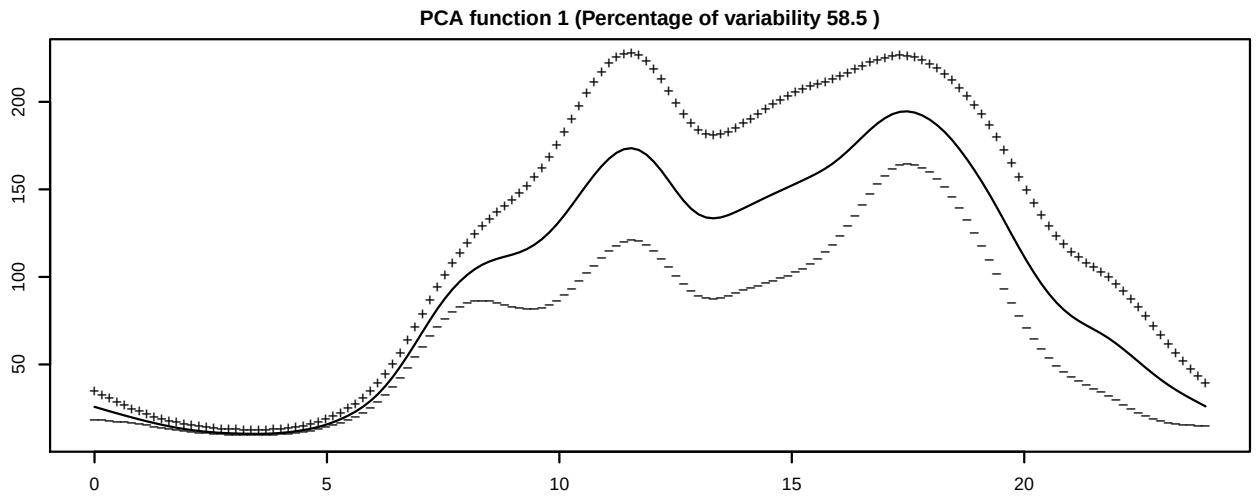


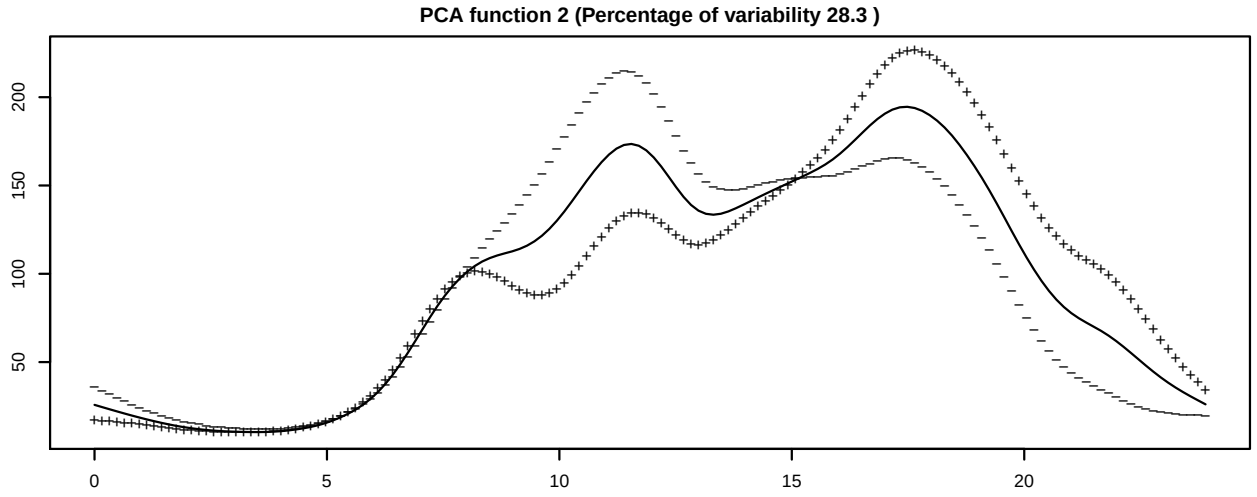
Au vu de la courbe de décroissance des dix premières valeurs propres, on peut supposer les deux premières sont intéressantes, et peut-être les 3 et 4.

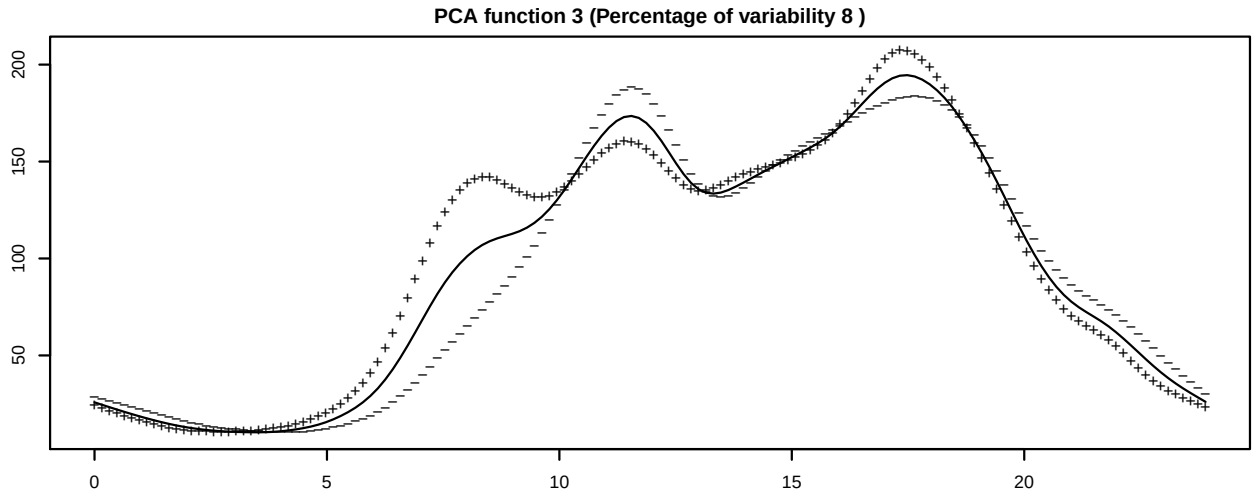
Question 2: *Représenter les 4 premiers axes principaux. Interpréter les axes. Combien faut-il en conserver ? Quelle proportion de l'inertie représentent-ils ?*

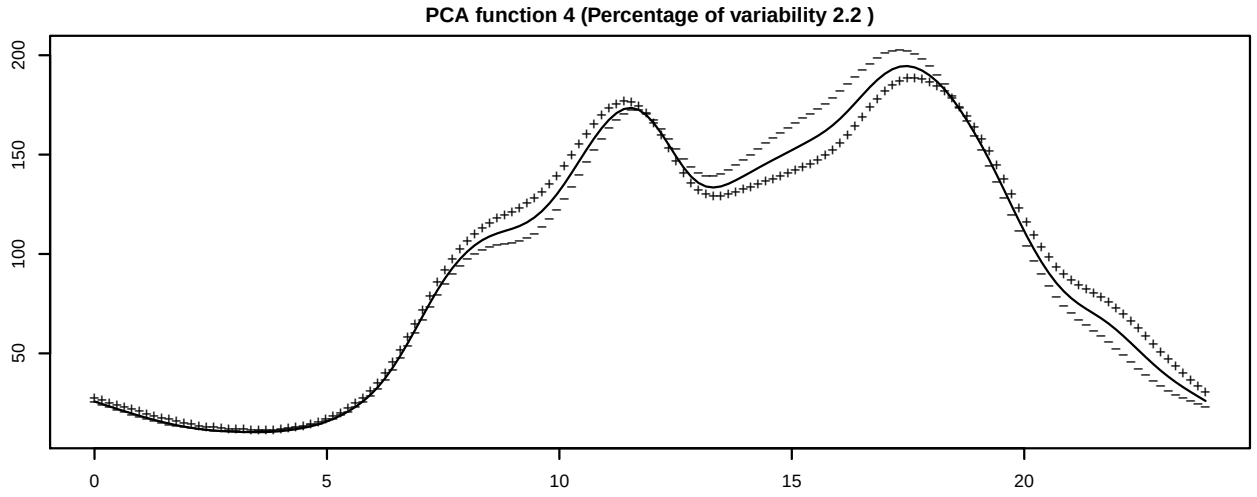
La représentation des axes se fait directement avec la fonction `plot.pca.fd`. Cette fonction va produire 4 graphiques, il faut donc les mettre en page avec `par()`.

```
> save=par(mfrow=c(4,1), cex=0.5, mar=c(2,2,2,2), oma=c(0,0,2,0))
> plot(dat.pca, nx=150, harm=1:4)
> par(save)
```


PCA function 1 (Percentage of variability 58.5)
50
100
150
200
0
5
10
15
20


PCA function 2 (Percentage of variability 28.3)
50
100
150
200
0
5
10
15
20


PCA function 3 (Percentage of variability 8)
50
100
150
200
0
5
10
15
20


PCA function 4 (Percentage of variability 2.2)
50
100
150
200
0
5
10
15
20

L'interprétation est la suivante :

axe 1 volume général du trafic : en positif, les journées sont plus chargées ;

axe 2 décalage matin/après midi : dans le sens positif, le trafic est plus important après 15h et plus limité le matin ;

axe 3 trafic matinal : en positif, le premier pic de circulation est avancé à 8h du matin et le pic normal est très amoindri ;

axe 4 pas vraiment interprétable, on propose de ne pas le garder.

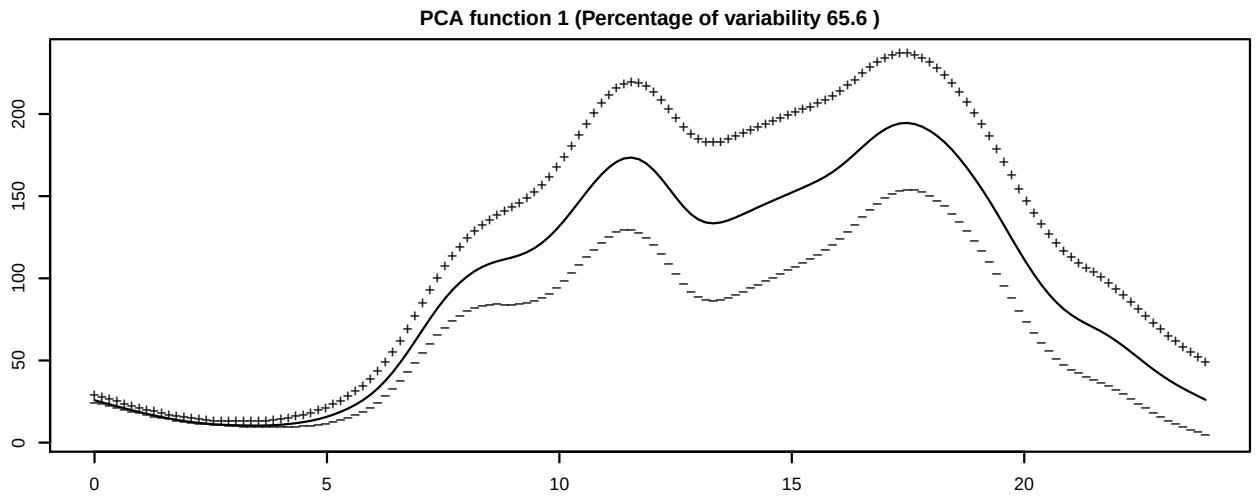
On garde donc au total 3 axes. Calculons la répartition de l'inertie :

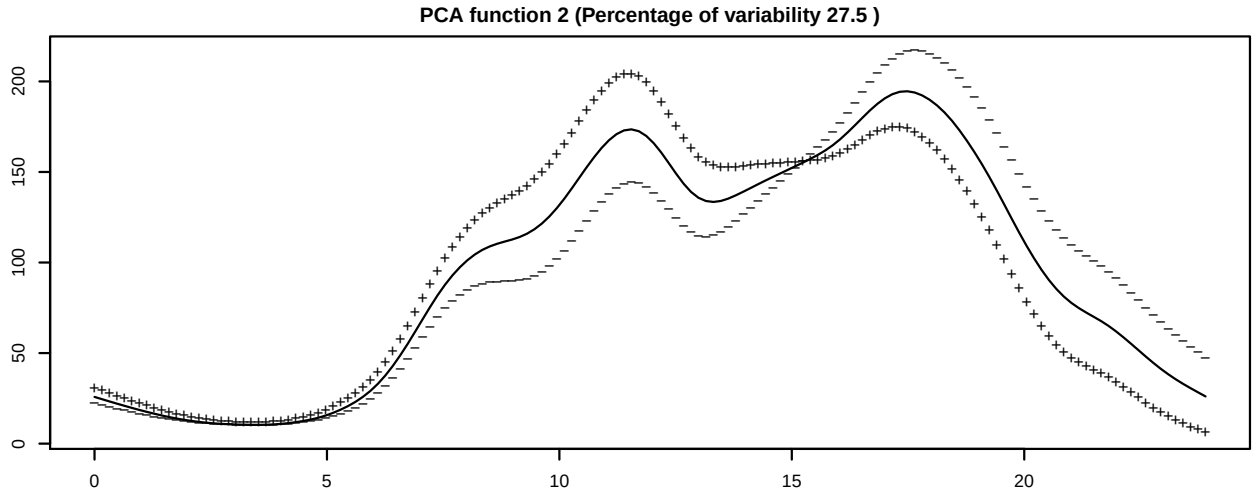
```
> cum=round(cumsum(dat.pca$varprop[1:8])*100)
> cum

[1] 59 87 95 97 98 98 99 99
```

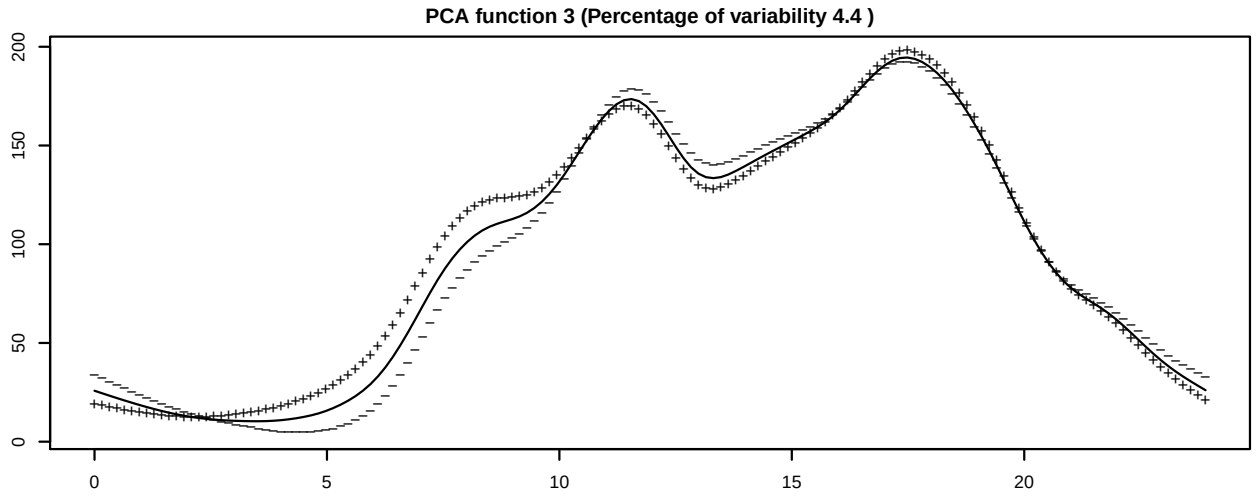
Les trois premiers axes représentent 95% de l'inertie totale. C'est donc une très bonne analyse.

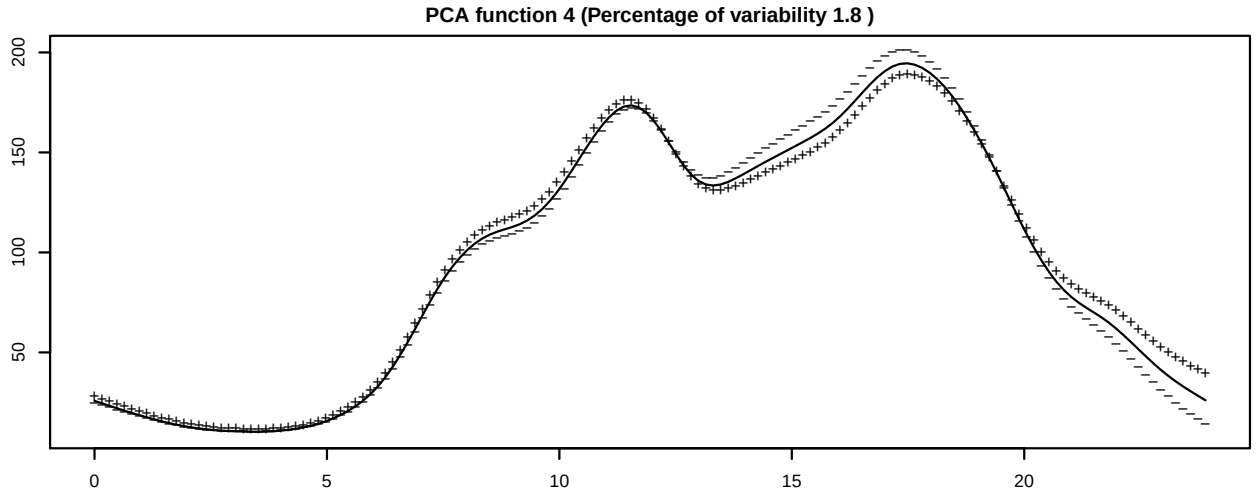
Influence du lissage On essaie de faire la même analyse avec un lissage assez fort $\lambda = 10$ (les valeurs plus petites ne font pas grand-chose).

```
> spca.fdParobj = fdPar(fdobj=basisobj, lambda=10)
> dat.spca = pca.fd(dat.fd, nharm = 8, harmfdPar=spca.fdParobj)
> round(cumsum(dat.spca$varprop[1:8])*100)

[1]  66  93  98  99 100 100 100 100

> save=par(mfrow=c(4,1), cex=0.5, mar=c(2,2,2,2), oma=c(0,0,2,0))
> plot(dat.spca, nx=150, harm=1:4)
> par(save)
```


PCA function 1 (Percentage of variability 65.6)
0
50
100
150
200
0
5
10
15
20


PCA function 2 (Percentage of variability 27.5)
0
50
100
150
200
0
5
10
15
20


PCA function 3 (Percentage of variability 4.4)
0
50
100
150
200
0
5
10
15
20


PCA function 4 (Percentage of variability 1.8)
50
100
150
200
0
5
10
15
20

Le résultat n'est pas très différent, sauf peut-être que la décroissance des valeurs propres est plus rapide et que le décalage de pic du matin sur l'axe 3 est moins visible. D'une manière générale, le lissage n'était pas nécessaire ici. En pratique on le fait quand les représentations montrent des zigzags importants.

Question 3: *Est-ce que l'ACP fonctionnelle sur la racine carrée des données est plus intéressante ? Quelle différence y a-t-il ?*

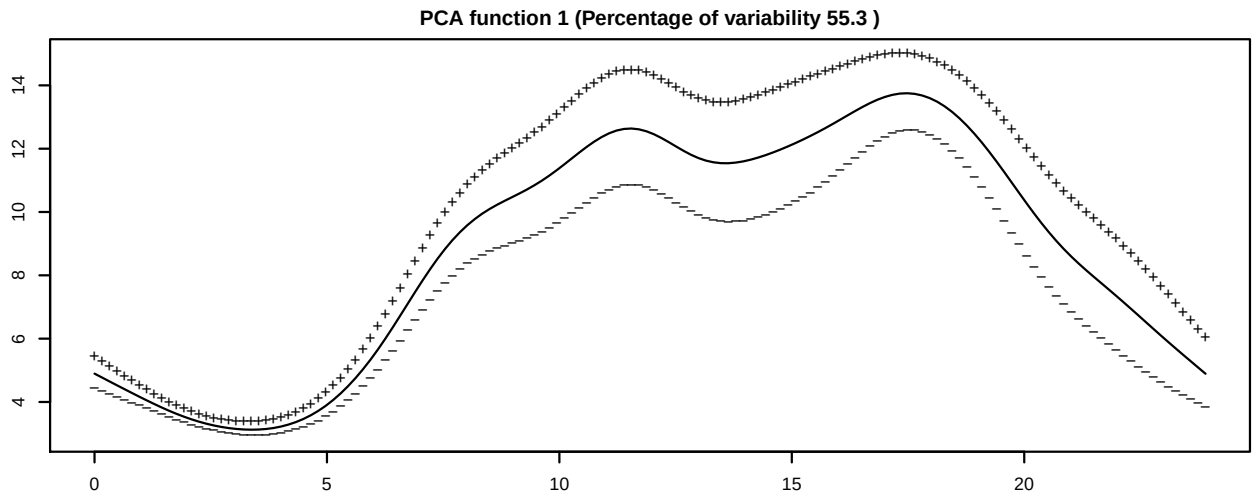
On prend la racine carrée des données et on conserve la valeur $\lambda = 0$: il n'est toujours pas nécessaire de lisser.

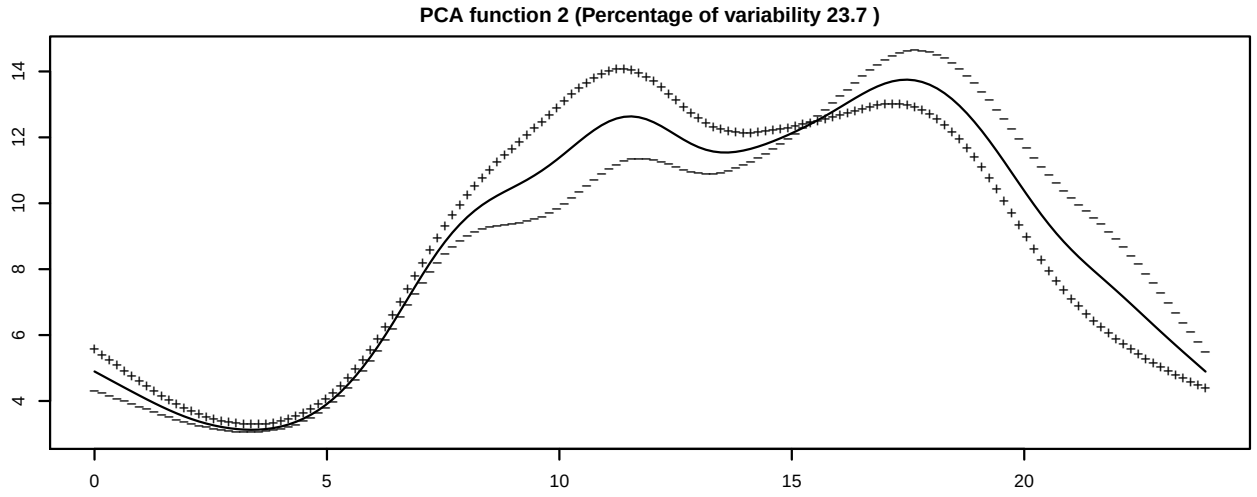
```
> sqdat=sqrt(dat)
> lambda=1
> fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=lambda)
> sqdat.fd = smooth.basis(argvals=heures, y=t(sqdat), fdParobj, fdnames=fdnames)$fd
> pca.fdParobj = fdPar(fdobj=basisobj, lambda=0.0)
> sqdat.pca = pca.fd(sqdat.fd, nharm = 8, harmfdPar=pca.fdParobj)
> round(cumsum(sqdat.pca$varprop[1:8])*100)

[1] 55 79 95 97 98 99 99 99
```

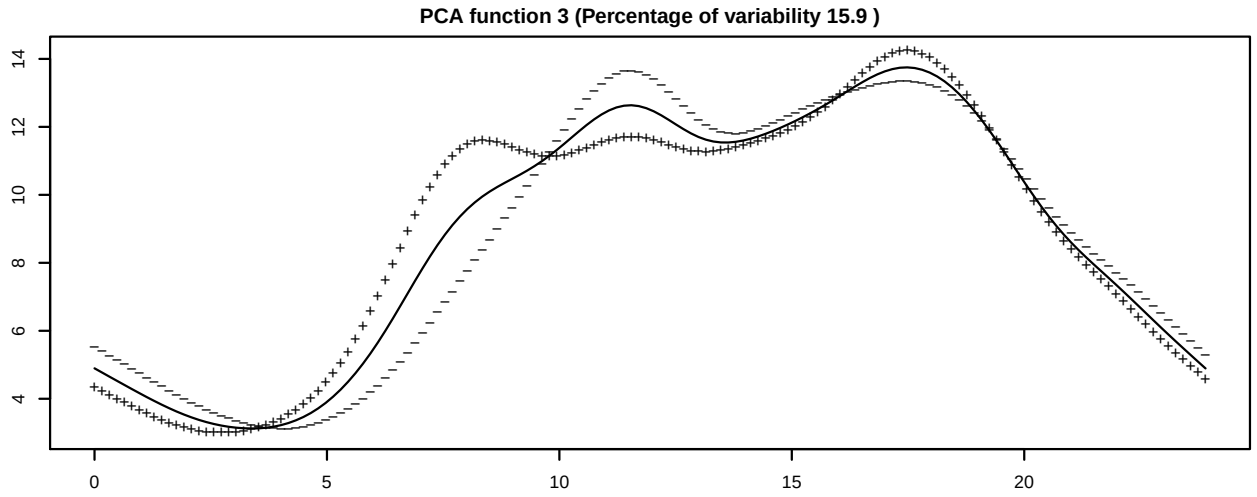
On voit que la répartition des valeurs propres est plus régulière, même si l'inertie expliquée par les 3 premiers axes est la même. Les composantes principales ci-dessous sont toujours interprétables de la même manière, mais sont plus marquées. Globalement, cette analyse n'apporte pas grand-chose de neuf.

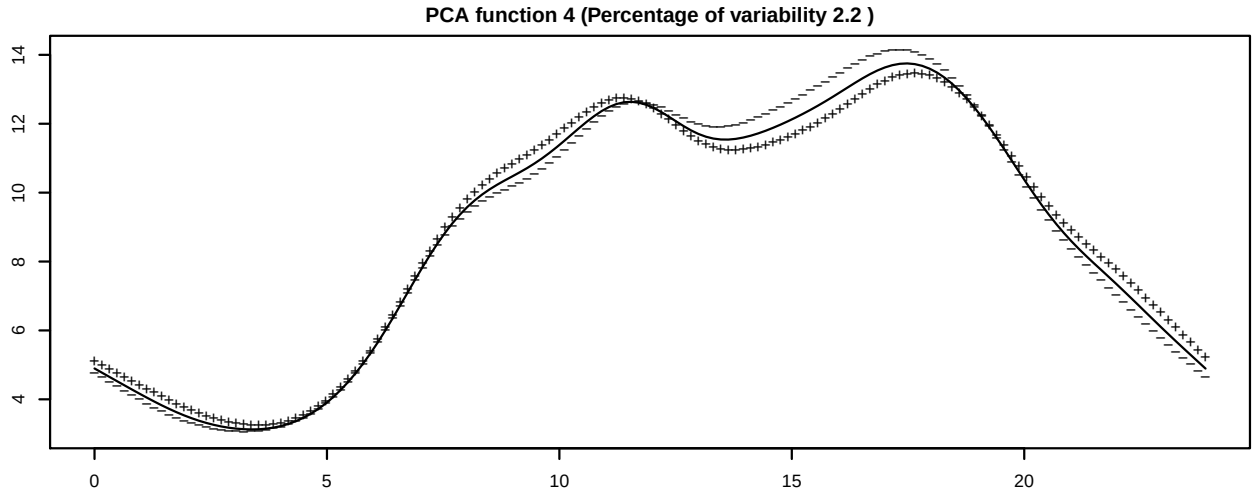
Il y a tout de même une différence remarquable : le signe du second facteur principal est inversé quand on travaille sur les racines carrées. Cela ne change rien car les facteurs sont définis au signe près, mais il faut faire attention.

```
> save=par(mfrow=c(4,1), cex=0.5, mar=c(2,2,2,2), oma=c(0,0,2,0))
> plot(sqdat.pca, nx=150, harm=1:4)
> par(save)
```


PCA function 1 (Percentage of variability 55.3)
4
6
8
10
12
14
0
5
10
15
20


PCA function 2 (Percentage of variability 23.7)
4
6
8
10
12
14
0
5
10
15
20


PCA function 3 (Percentage of variability 15.9)
4
6
8
10
12
14
0
5
10
15
20


PCA function 4 (Percentage of variability 2.2)
4
6
8
10
12
14
0
5
10
15
20

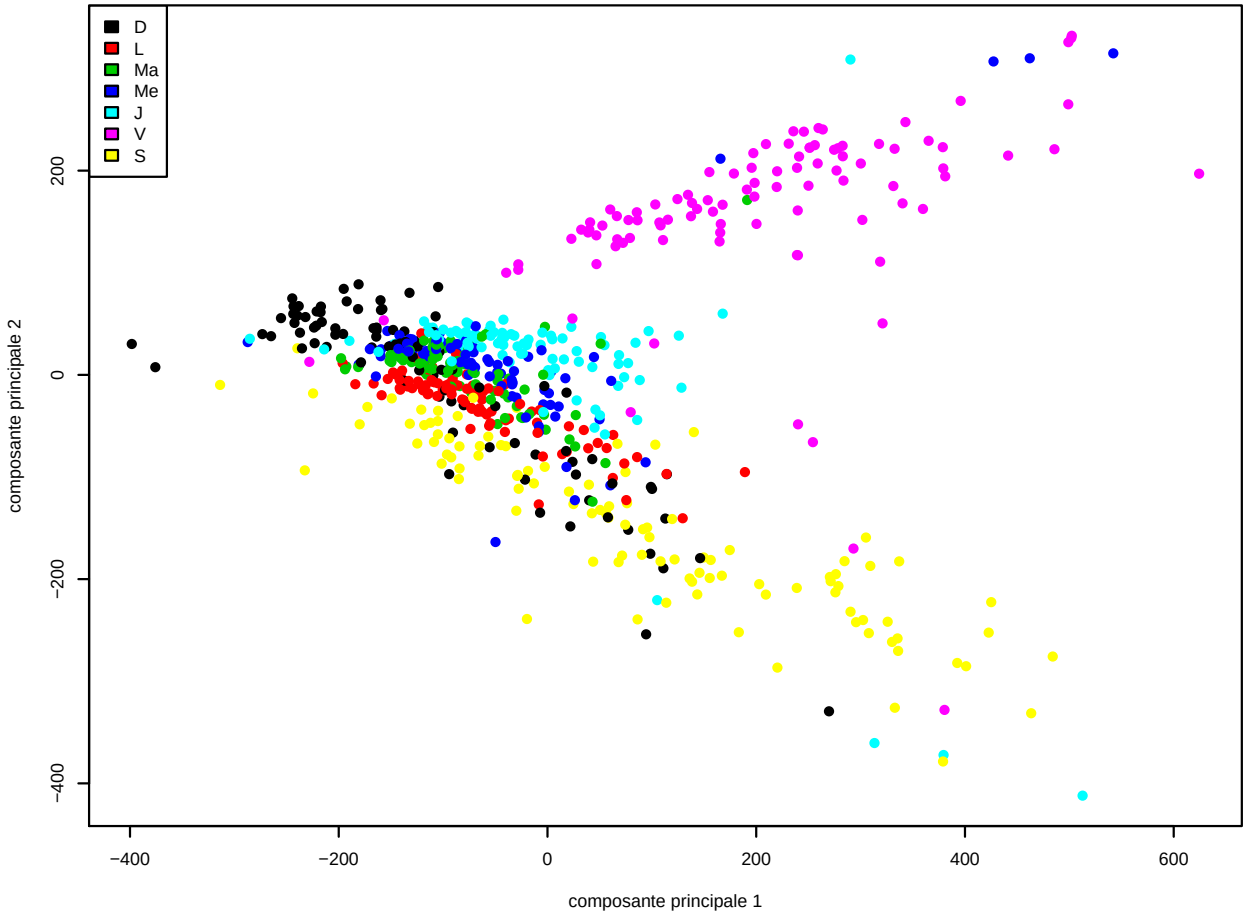
Question 4: *On se donne un tableau contenant les jours de la semaine (0=dimanche, ... 6=samedi) calculé comme suit :*

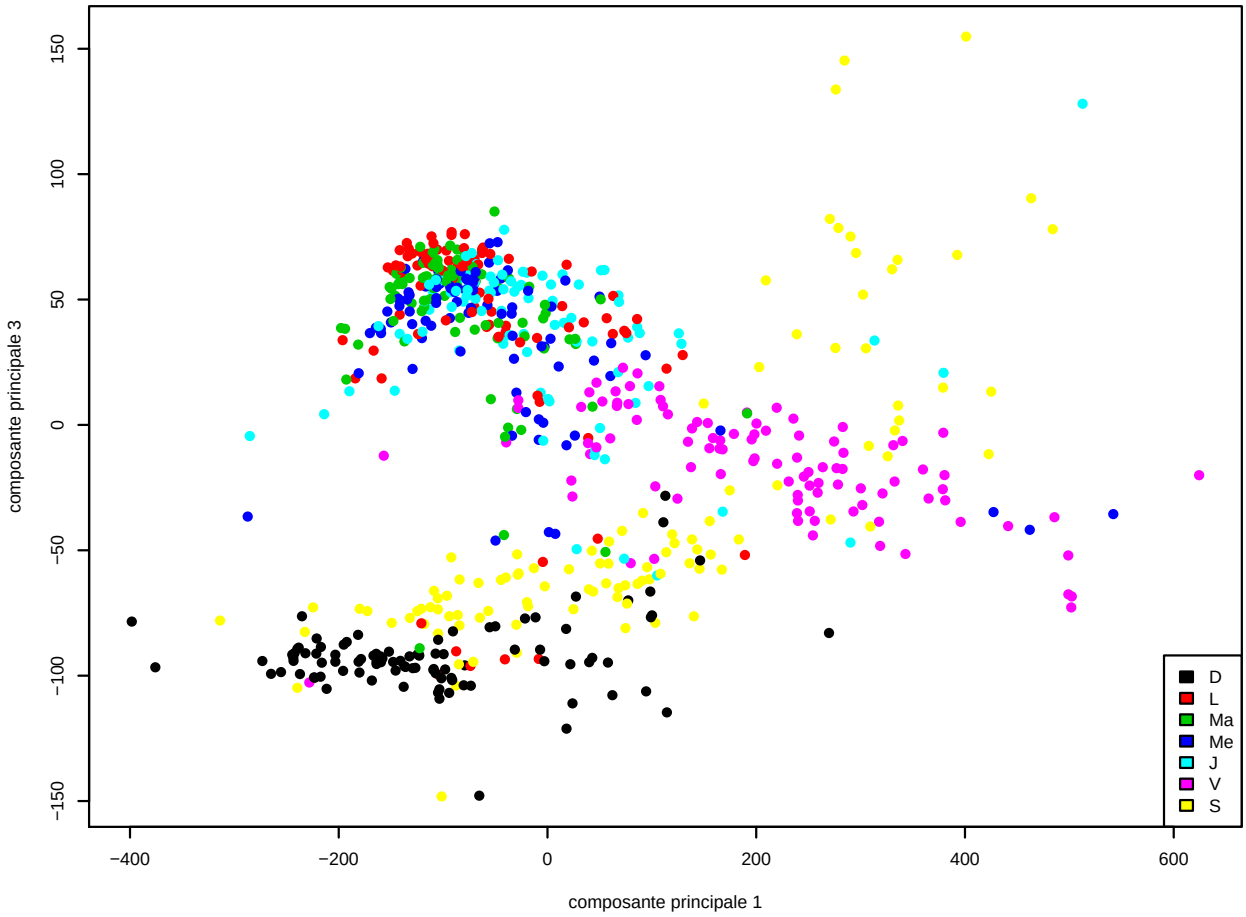
```
> jours=as.numeric(format(as.Date(rownames(dat), format="%d/%m/%Y"),"%w"))
```

Réalisez pour les axes $(1, 2)$ *d'une part, puis* $(1, 3)$ *d'autre part, des graphique où chaque jour est représenté par une couleur qui dépend du jour de la semaine. Que peut-on déduire à propos du lien entre les axes et les jours de la semaine ? Que peut-on déduire en prenant en compte l'interprétation des axes ?*

On réalise les deux diagrammes demandés, et on ajoute une légende permettant de comprendre les jours.

```
> semaine=c("D", "L", "Ma", "Me", "J", "V", "S")
> save=par(mfrow=c(2,1), cex=0.5)#, mar=c(2,2,2,2), oma=c(0,0,2,0))
> plot(dat.pca$scores[,1], dat.pca$scores[,2], col=jours+1, pch=19,
+      xlab="composante principale 1", ylab="composante principale 2")
> legend("topleft", semaine, fill=1:7)
> plot(dat.pca$scores[,1], dat.pca$scores[,3], col=jours+1, pch=19,
+      xlab="composante principale 1", ylab="composante principale 3")
> legend("bottomright", semaine, fill=1:7)
> par(save)
```


D
L
Ma
Me
J
V
S
composante principale 2
composante principale 1
−400
−200
0
200
400
600


D
L
Ma
Me
J
V
S
composante principale 3
composante principale 1
−150
−100
−50
0
50
100
150
−400
−200
0
200
400
600

On peut voir que
— l'axe 1 marque surtout le vendredi en positif et le dimanche en négatif ; le samedi est plus étalé, même s'il semble en positif ;
— l'axe 2 oppose le vendredi en positif et le samedi en négatif ;
— l'axe 3 met en évidence le dimanche en négatif.

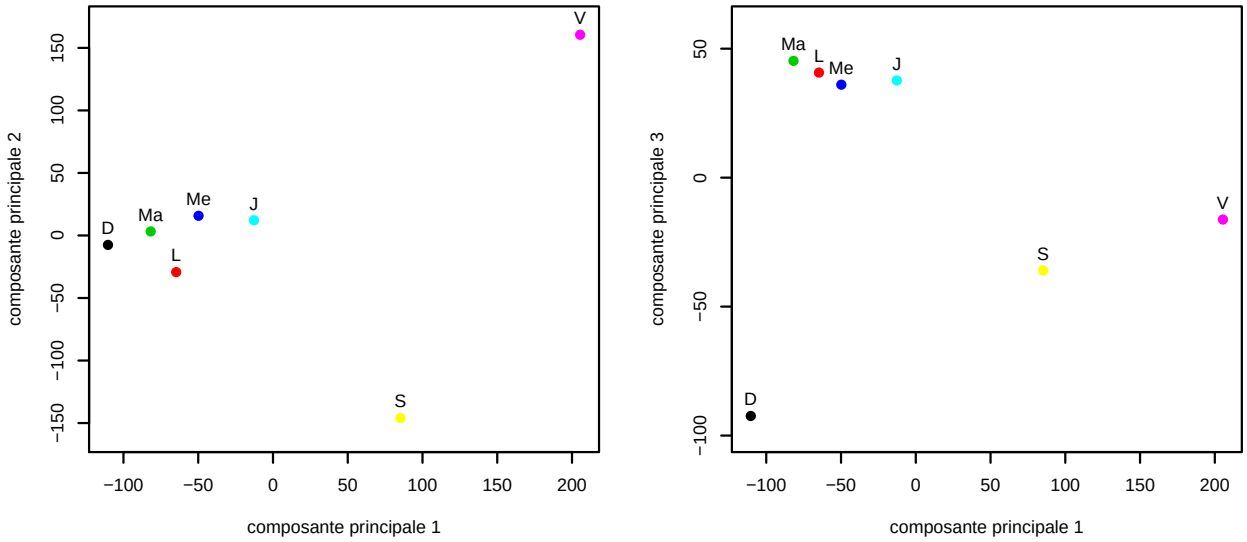
On en déduit (à la louche) que
— le trafic est plus élevé que la moyenne le vendredi (et peut-être le samedi), et moins le dimanche ;
— le trafic est plus élevé que la moyenne le samedi matin et moins que la moyenne le vendredi après-midi ;
— enfin, le pic de circulation se produit plus tôt le dimanche.

On remarquera enfin que les nuages des vendredi et samedi sont diagonaux. Cela signifie qu'un vendredi chargé a un après midi chargé, et qu'un samedi chargé a une matinée chargée.

Ces interprétations sont assez vagues, il faut plus de travail pour être sérieux.

Bonus (non demandé) Pour être plus précis, il faut calculer des valeurs test pour chaque jour (voir mon cours d'ACP à `http://ana-donnees.lasgouttes.net`). On commence par calculer les barycentres des scores de chaque jour et les représenter.

```
> jours.scores=as.matrix(aggregate(dat.pca$scores, list(jours), mean)[,-1])
> jours.num=table(jours)
> jours.tout=sum(jours.num)

> save=par(mfrow=c(1,2), cex=0.5)
> plot(jours.scores[,1], jours.scores[,2], col=1:7, pch=19, ylim=c(-160,170),
+      xlab="composante principale 1", ylab="composante principale 2")
> text(jours.scores[,1], jours.scores[,2], semaine, pos=3)
> plot(jours.scores[,1], jours.scores[,3], col=1:7, pch=19, ylim=c(-100,60),
+      xlab="composante principale 1", ylab="composante principale 3")
> text(jours.scores[,1], jours.scores[,3], semaine, pos=3)
> par(save)
```



Voilà la définition d'une valeur test : si l'effectif d'une journée est n (sur un total de N) et que son point moyen sur l'axe k (de valeur propre θ_k) est z_k, alors, sa valeur test sur l'axe k est

$$z_k \sqrt{\frac{n}{\theta_k} \frac{N-1}{N-n}}.$$

Si la valeur test est plus grande que 3 en valeur absolue, alors on peut considérer que le jour est lié à l'axe k avec le signe de sa coordonnée. On donne ci-dessous les valeurs test obtenues.

```
> j.tmp=sweep(jours.scores, MARGIN=1,  sqrt(dat.pca$values[1:ncol(jours.scores)]), "/")
> valtest = sweep(j.tmp, MARGIN=2, sqrt(jours.num*(jours.tout-1)/(jours.tout-jours.num)), FUN="*")
> rownames(valtest)=semaine
> round(valtest,2)[,1:4]

       V1      V2     V3    V4
D   -7.24   -7.14 -77.90  7.12
L   -5.89   -1.79  38.34 -3.80
```

```
Ma -13.55    0.30   2.90 -1.86
Me -16.22    2.73   3.44 -0.16
J   -6.78    4.19   7.02 -0.29
V  142.52   92.23  -5.53 -0.65
S   80.79 -100.06 -19.01  1.11
```

On en déduit une interprétation plus fine :
— le trafic est plus élevé que la moyenne le vendredi et dans un moindre mesure le samedi ; par contre, il est plutôt plus faible que la moyenne les dimanche, lundi et mardi ;
— le trafic est plus élevé que la moyenne le samedi matin et moins que la moyenne le vendredi après-midi ;
— enfin, le pic de circulation se produit nettement plus tard le dimanche que le samedi, puis le vendredi, puis un paquet plus ou moins uniforme des autres jours qui ont le pic le plus matinal.

3 Gestion des valeurs manquantes

On voudrait analyser aussi les journées dont le nombre de données manquantes est inférieur à 10 (mais non nul)

Question 5: *faire une sélection de ces données. Combien y en a-t-il ?*

On sélectionne les données, puis on affiche le nombre d'individus des différents jeux de données (complet, sans donnée manquante, avec moins de 9 données manquantes).

```
> dat.1.9na=dat0[(na_count >= 1) & (na_count <= 9),]
> nrow(dat0)

[1] 761

> nrow(dat)

[1] 661

> nrow(dat.1.9na)

[1] 77
```

Les données sélectionnées représentent à peu près 10% des données d'origine. On voit bien l'intérêt de les conserver

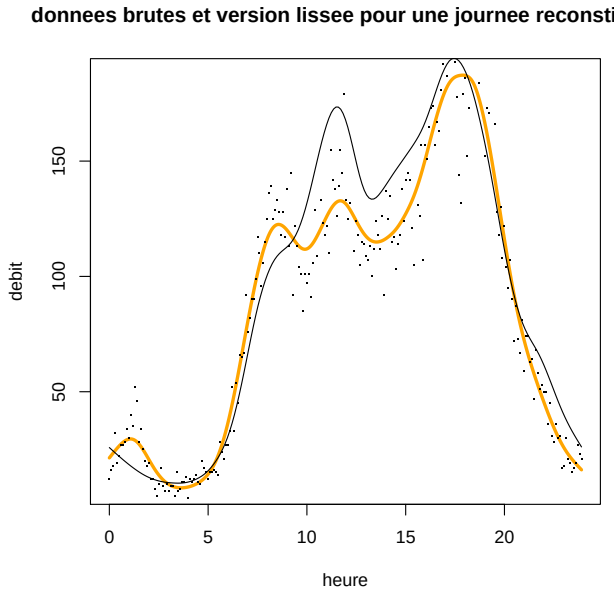
Question 6: *La méthode à utiliser est de traiter les journées une par une en enlevant les données manquantes et les temps correspondants. Écrire le code permettant de lisser une journée avec des données manquantes.*

Le coût va être plus important pour lisser ces courbes, mais cela reste facile. On essaie de modifier la valeur de λ proportionnellement au nombre de valeurs restantes (même si on n'est pas sûr que cela change grand chose).

```
> # on fait arbitrairement la premiere courbe
> i=5
> fdParobj.i = fdParobj
> dat.i=dat.1.9na[i,]
> hr=heures[!is.na(dat.i)]
> dt=dat.i[!is.na(dat.i)]
> fdParobj.i$lambda = fdParobj$lambda * length(hr)/length(heures)
> dat.i.fd = smooth.basis(argvals=hr, y=dt, fdParobj.i, fdnames=fdnames)$fd
```

On représente les données avec la moyenne pour comparer.

```
> dummy=plot(dat.i.fd, lwd=3, col='orange',
+   main='donnees brutes et version lissee pour une journee reconstituee')
> lines(mean(dat.fd))
> points(hr,dt, pch='.')
```



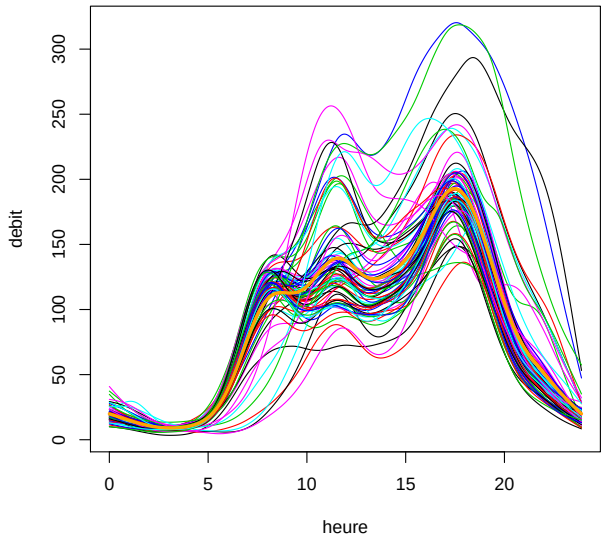
Question 7: *Écrire le code permettant de lisser un ensemble deDernière question journées et de mettre le résultat dans un objet fd unique. Pour cela on utilisera la fonction* `fd()` *pour créer à la main un objet fonctionnel.*

On fait la même chose dans une boucle. On va créer une matrice de coefficients dont les colonnes seront formées par les vecteurs de coefficients des courbes lisses individuelles. On utilise la fonction `cbind()` pour placer les coefficients de chaque courbe dans une matrice commune.

La manière dont les choses sont faites permet de conserver les noms des jours. Sinon, il aurait fallu les remettre à la main.

```
> fdParobj.x = fdParobj
> coeffs=NULL
> # fonction qui lisse la courbe de donnees x et retourne ses coefficients
> unecourbe = function (x) {
+   hr=heures[!is.na(x)]
+   dt=x[!is.na(x)]
+   fdParobj.x$lambda = fdParobj$lambda * length(hr)/length(heures)
+   x.fd = smooth.basis(argvals=hr, y=dt, fdParobj.x)
+   # le <<- modifie la variable globale 'coeff', pas une variable locale
+   coef(x.fd)
+ }
> # on applique la fonction a toutes les lignes une par une
> coeffs=apply(dat.1.9na, 1, unecourbe)
> # on obtient un tableau de la bonne dimension
> dim(coeffs)

[1] 242  77

> dat.1.9na.fd = fd(coef=coeffs, basisobj=basisobj, fdnames=fdnames)
```

```
> dummy=plot(dat.1.9na.fd, lty=1)
> lines(mean(dat.1.9na.fd), lwd=3, col='orange')
```

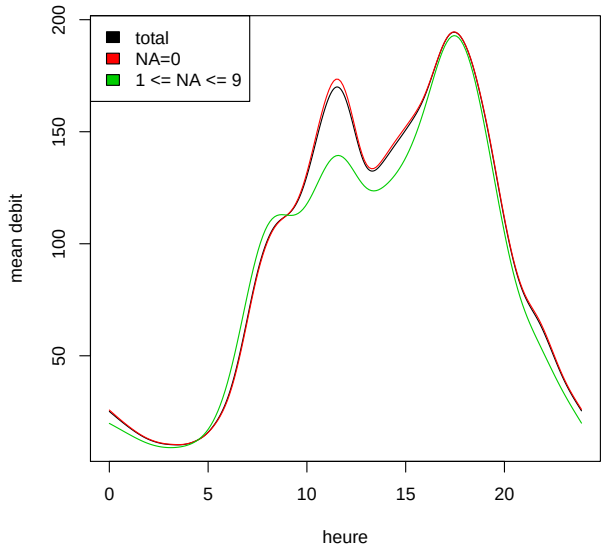


Question 8: *ajouter ce nouvel objet fonctionnel à celui qu'on avait pour les éléments sans valeurs manquantes. On tracera par exemple les courbes des moyennes des différentes données ($NA = 0$, $0 < NA \leq 9$, $NA \geq 9$) pour montrer que les résultats sont cohérents, même s'il y a des différences Expliquer pourquoi on peut maintenant faire toutes les analyses (ACP...) sur cette nouvelle donnée*

Il faut simplement concaténer les tableaux de coefficients. Comme ce sont les colonnes qui doivent être mises côte-à-côte, on utilise `cbind()`.

```
> dat.0.9na.fd = fd(coef=cbind(coef(dat.fd), coef(dat.1.9na.fd)), basisobj=basisobj, fdnames=fdnames)
```

```
> dummy = plot(mean(dat.0.9na.fd))
> lines(mean(dat.fd), col=2)
> lines(mean(dat.1.9na.fd), col=3)
> legend("topleft", c("total", "NA=0", "1 <= NA <= 9"), fill=1:6)
```



On constate que les données pour lesquelles des données sont manquantes ont une moyenne de débit plus basse entre 8 et 16h ; il est difficile d'expliquer pourquoi. On peut imaginer que, quand un capteur de débit ne fonctionne pas pendant une partie de l'intervalle de 6 minutes, le débit retourné est plus petit.