TP : station de comptage de trafic (jour 3)

(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 étudier le recalage des données.

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 part là encore des mêmes données. On n'a pas pris les données reconstruites du TP2, mais cela ne change pas grand chose. Par contre, on retire les vendredi et les samedis, dont on a vu au TP précédent qu'ils sont très différents des autres jours.

```
> # 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))
> # Journees sans donnee manquante
> dat = dat0[na_count == 0,]
> # Les jours de la semaine (0=dimanche, ... 6=samedi)
> jours=as.numeric(format(as.Date(rownames(dat), format="%d/%m/%Y"),"%w"))
> # on se limite  dimanche-jeudi
> dat = dat[jours <= 4,]
> # on adapte les jours
> jours = jours[jours <= 4]
> njours = nrow(dat)
> # Si on veut reduire les grandes valeurs
> #dat=sqrt(dat)
> require(fda)
```

2 Modélisation efficace de la dérivée du flux

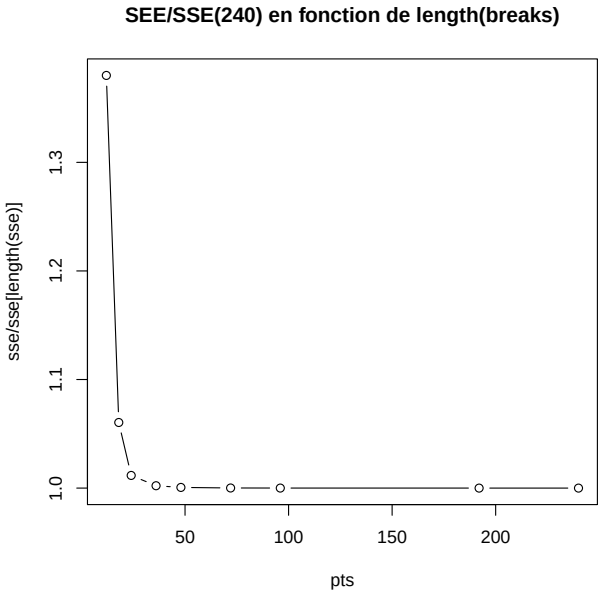
On veut dans la partie suivante explorer les possibilités de recalage de courbes de débit. Comme les extrema locaux correspondent à l'annulation de la dérivée, on va commencer par faire un bonne modélisation de cette dérivée. On sait que les calculs de recalage sont longs et on cherche à réduire le nombre de fonctions de la base.

Question 1: *représenter de manière fonctionnelle (avec une base de B-splines) les débits de manière à pouvoir calculer leur dérivée première. Étudier l'effet de la diminution du nombre de points sur le SSE et choisir une taille de base raisonnable.*

On recrée des courbes lissées comme dans le TP2. Comme on va prendre un nombre de bases inférieur au nombre de points, la lissage n'est pas nécessaire et on peut se contenter de faire une régression linéaire.

```
> nderiv = 3
> norder = nderiv + 2
> fdnames=list("heure", "jour", "debit")
> pts = c(12, 18, 24, 36, 48, 72, 96, 192, 240)
> sse = NULL
> for (npoints in pts) {
+   basisobj = create.bspline.basis(breaks=seq(0,23.9,len=npoints), norder=norder)
+   fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=0.1)
+   dat.fd = smooth.basis(argvals=heures, y=t(dat), fdParobj, fdnames=fdnames)
+   sse = c(sse, dat.fd$SSE/njours)
+ }
```

```
> plot(pts,sse/sse[length(sse)], type='b', main="SEE/SSE(240) en fonction de length(breaks)")
```

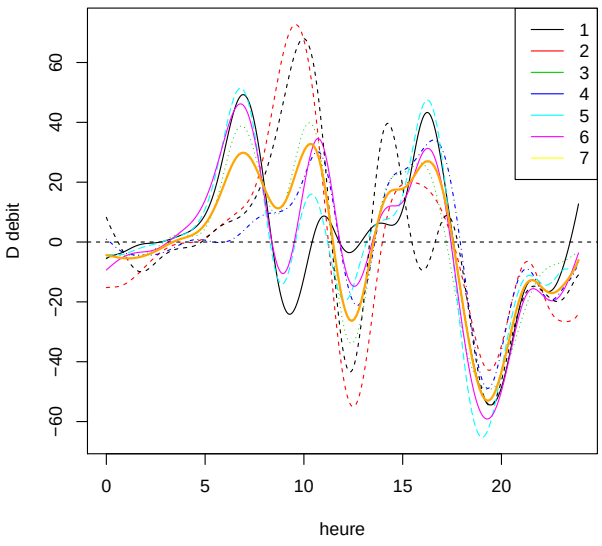


On voit qu'un bon compromis est de se limiter à 24 points dans la base de splines. C'est 10 fois moins que 240 et le SSE augmente d'à peine 2%.

```
> basisobj = create.bspline.basis(breaks=seq(0,23.9,len=24), norder=norder)
> fdParobj = fdPar(fdobj=basisobj, Lfdobj=nderiv, lambda=0.1)
> dat.fd = smooth.basis(argvals=heures, y=t(dat), fdParobj, fdnames=fdnames)
> # Quelques courbes choisies arbitrairement
> courbes=c(12, 88, 142, 207, 273, 339, 400)
> jours[courbes]

[1] 2 1 2 0 3 4 0

> nrow(dat)

[1] 457

> # Nos donnees de travail. On se limite a 5 courbes pour des questions de performance
> ddat.fd = deriv(dat.fd$fd[courbes,])
> # ceci va etre utile plus loin pour specifier les points de repere
> save(ddat.fd, file="ddat.Rdata")
```

On obtient finalement les courbes suivantes pour les dérivées

```
> dummy=plot(ddat.fd)
> lines(mean(ddat.fd), lwd=2,col="orange")
> legend("topright", legend=1:7, col=1:7, lwd=1)
```



3 Recalage des données

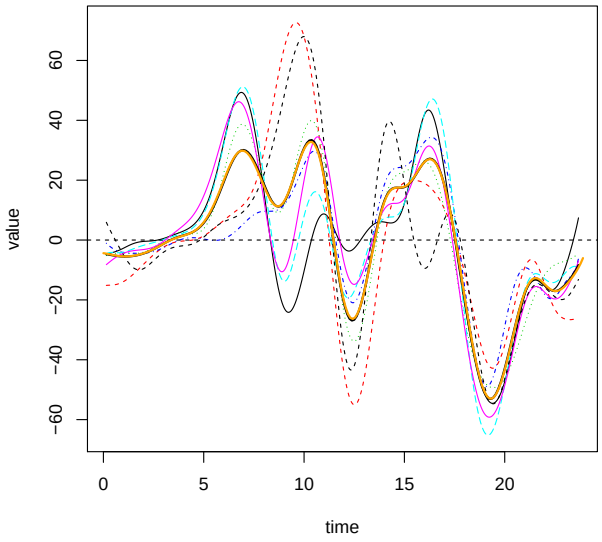
Les calculs de recalage étant longs, on pourra se limiter, notamment pour le recalage fonctionnel, à quelques courbes (au moins 5).

Question 2: *Faire un recalage par décalage uniforme de la dérivée du débit. On notera le temps de calcul (voir fonction* `system.time()`*)*

On recale `ddat.fd` par rapport à sa moyenne

```
> system.time(ddat.regfd0 <- register.fd0(mean(ddat.fd), ddat.fd))

   user  system elapsed
  1.896   0.220   1.851
```

```
> dummy=plot(ddat.regfd0$regfd)
> lines(mean(ddat.regfd0$regfd), lwd=2)
> lines(mean(ddat.fd), lwd=2,col="orange")
```



On ne dispose pas de moyen de calculer la proportion de variation due à la phase avec cette modification. Les courbes n'ont pas l'air très alignées et les moyennes n'ont pas beaucoup évolué, mais le temps de calcul est très court.

Question 3: *Faire un recalage fonctionnel de la dérivée du débit. On notera le temps de calcul et la proportion de variabilité due à la phase.*

On recale `ddat.fd` par rapport à sa moyenne. Pour cela une base de dimension 5 et un λ faible suffisent : les h_i varient peu. Pour des questions de performance, on met le calcul du recalage dans une script séparé qui sauvegarde son résultat :

```
require(fda)
regnbasis = 7
regbasis = create.bspline.basis(rangeval=c(0,23.9), nbasis=regnbasis)
regfd0 = fd(matrix(0,regnbasis,1),regbasis)
regfdParobj <- fdPar(regfd0, Lfdobj=2, lambda=0.01)
load("ddat.Rdata")
st.regfd = system.time(ddat.regfd <- register.fd(mean(ddat.fd), ddat.fd, regfdParobj))
apd.regfd = AmpPhaseDecomp(ddat.fd, ddat.regfd$regfd, ddat.regfd$warpfd)
save(list=c("ddat.regfd", "st.regfd", "apd.regfd"), file="ddat.regfd.Rdata")
```

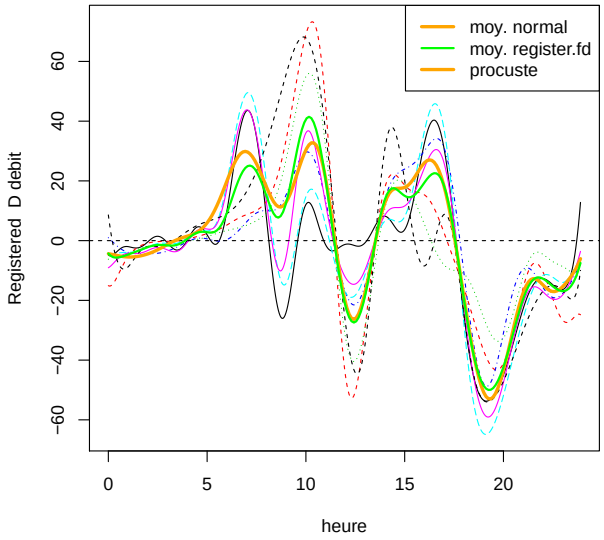
Le résultat obtenu est le suivant pour le temps de calcul et l'erreur due à la phase

```
> load("ddat.regfd.Rdata")
> #temps de calcul
> st.regfd

   user  system elapsed
 94.232 148.012  51.623
```

```
> # proportion d'erreur quadratique due a la phase
> apd.regfd$RSQR

[1] 0.1336317
```

Et voici ci-dessous les courbes recalées avec leur moyenne (vert) et la moyenne des données de départ (orange). On voit que la moyenne est mieux marquée

```
> dummy=plot(ddat.regfd$regfd)
> lines(mean(ddat.fd), lwd=3, col="orange")
> lines(mean(ddat.regfd$regfd), lwd=2, col="green")
> legend("topright", legend=c("moy. normal","moy. register.fd", "procuste"),col=c("orange", "green")
```



Question 4: *Appliquer la méthode de Procuste. Est-ce que les résultats sont meilleurs ?*

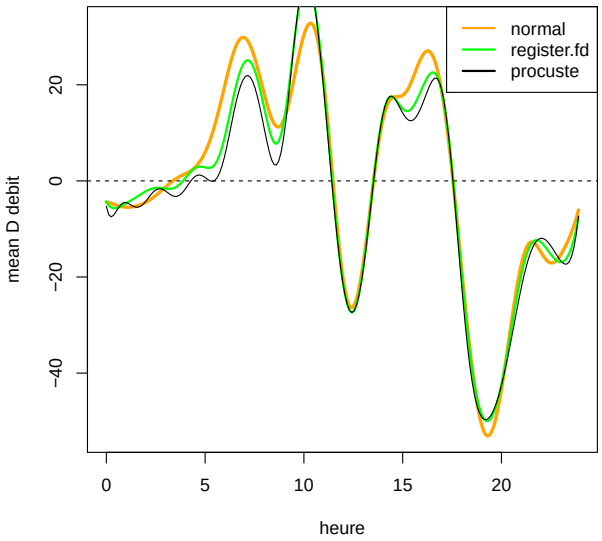
On utilise la méthode de Procuste pour aligner une seconde fois les données.

```
require(fda)
regnbasis = 7
regbasis = create.bspline.basis(rangeval=c(0,23.9), nbasis=regnbasis)
regfd0 = fd(matrix(0,regnbasis,1),regbasis)
regfdParobj <- fdPar(regfd0, Lfdobj=2, lambda=0.01)
load("ddat.Rdata")
load("ddat.regfd.Rdata")
st.regfd.1 = system.time(
  ddat.regfd.1 <- register.fd(mean(ddat.regfd$regfd),
    ddat.fd, regfdParobj)
  )
apd.regfd.1 = AmpPhaseDecomp(ddat.fd, ddat.regfd.1$regfd, ddat.regfd.1$warpfd)
save(list=c("ddat.regfd.1", "st.regfd.1", "apd.regfd.1"), file="ddat.regfd.1.Rdata")
```

```
> load("ddat.regfd.1.Rdata")
> st.regfd.1

   user  system elapsed
 99.608 163.776  51.304

> apd.regfd.1$RSQR

[1] 0.2569096
```

La proportion de la variation due à la phase s'améliore un petit peu. On trace ci-dessous les moyennes des différents jeux de courbes

```
> dummy = plot(mean(ddat.fd), lwd=3, col="orange")
> lines(mean(ddat.regfd$regfd), lwd=2, col="green")
> lines(mean(ddat.regfd.1$regfd))
> legend("topright", legend=c("normal","register.fd", "procuste"), col=c("orange", "green", "black")
```



Question 5: *Faire un recalage par point de repère de la dérivée du débit. Comment choisit-on un bon point de repère ? On notera le temps de calcul et la proportion de variabilité due à la phase. Commenter la qualité des résultats*

Pour trouver les points de repères, on pourra utiliser la fonction suivante (présente dans le fichier `points.repere.R`*) qui demande interactivement à l'utilisateur de sélectionner le point de repère sur chaque courbe et renvoie un tableau de valeurs.*

```
points.repere = function(dat) {
  ncourbes = ncol(coef(dat))
  pt.rep = rep(0, times=ncourbes)

  for(i in 1:ncourbes) {
    plot(dat[i], main=paste("Courbe", i))
    pt.rep[i] = locator(1)$x
  }
  pt.rep
}
```

On utilise les données sauvées dans le fichier et on exécute dans une session R

```
require(fda)
source("points.repere.R")
load("ddat.Rdata")
reperes = points.repere(ddat.fd)
save(reperes, file="reperes.Rdata")
```

On choisit comme point de repère le maximum de débit de l'après midi, c'est-à-dire le point où la dérivée du trafic croise 0 en descendant. C'est le point le plus simple à repérer. On obtient les valeurs suivantes pour les repères

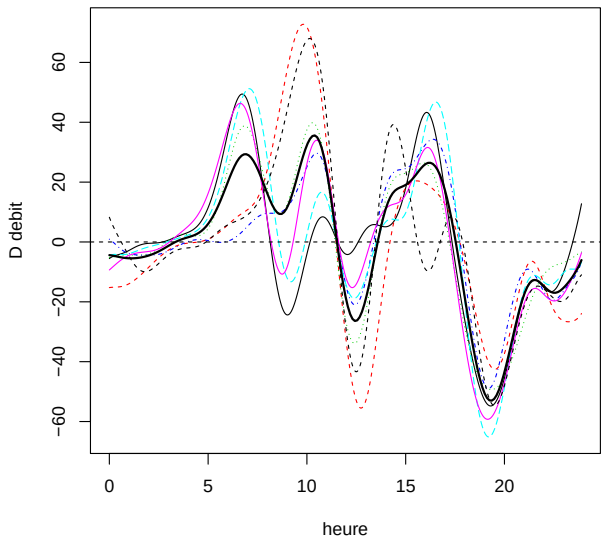
```
> load("reperes.Rdata")
> reperes

[1] 11.77697 11.26257 11.49639 11.73021 11.16904 11.73021 11.35609
```

Maintenant on peut faire une recalage sur les points de repère

```
> lndnbasis = 10
> lndbasis = create.bspline.basis(rangeval=c(0,23.9), nbasis=lndnbasis)
> lndfd0 = fd(matrix(0,lndnbasis,1),lndbasis)
> # on utilise juste un peu de lissage
> lndfdParobj <- fdPar(lndfd0, Lfdobj=2, lambda=0.01)
> st.landmk = system.time(ddat.landmk <- landmarkreg(fdobj=ddat.fd, ximarks=reperes, x0marks=mean(reperes), WfdPar=lndfdParobj))
```

```
Progress:  Each dot is a curve
.......

> apd.landmk = AmpPhaseDecomp(ddat.fd, ddat.landmk$regfd, ddat.landmk$warpfd)
> apd.landmk$RSQR

[1] 0.02382527
```

```
> dummy=plot(ddat.landmk$regfd)
> lines(mean(ddat.landmk$regfd), lwd=2)
```



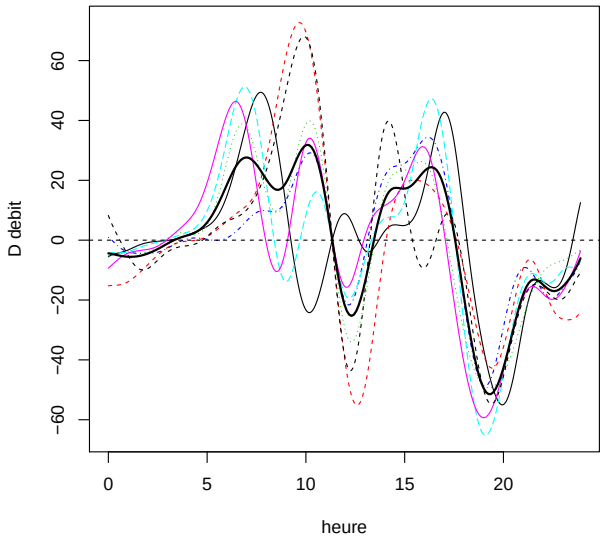
La proportion de la variation expliquée par la phase est faible (2.4%), il faut dire que les courbes de départ ne sont pas très décalées au niveau du point de repère.

Question 6: *Écrire une fonction qui détermine automatiquement le point de repère autour de 12h. Quelle performance obtient-on ?*

On utilise la code suivant

```
> points.repere.auto = function(fdobj, evalargs=NULL, rng=NULL) {
+   ncourbes = ncol(coef(fdobj))
+   pt.rep = rep(0, times=ncourbes)
+   if (is.null(evalargs)) {
+     if (is.null(rng))
+       rng=fdobj$basis$rangeval
+     evalargs=seq(from=rng[1], to=rng[2], len=1000)
+   }
+
+   for(i in 1:ncourbes) {
+     vals = predict(fdobj[i], evalargs)
+     for (j in 1:length(vals)) {
+       if (vals[1] * vals[j] < 0)
+            break
+     }
+     pt.rep[i]=(evalargs[j]*vals[j-1]-evalargs[j-1]*vals[j])/(vals[j-1]-vals[j])
+   }
+   pt.rep
+ }
> reperes=points.repere.auto(ddat.fd, rng=c(10,13))
> system.time(ddat.landmk <- landmarkreg(fdobj=ddat.fd, ximarks=reperes, x0marks=mean(reperes), WfdPar=lndfdParobj))

Progress:  Each dot is a curve
.......
   user  system elapsed
  1.284   2.652   0.506

> apd.landmk = AmpPhaseDecomp(ddat.fd, ddat.landmk$regfd, ddat.landmk$warpfd)
> apd.landmk$RSQR

[1] -0.1100384
```

```
> dummy=plot(ddat.landmk$regfd)
> lines(mean(ddat.landmk$regfd), lwd=2)
```



Question 7: *Quelle est la méthode la plus adaptée à l'analyse des données complètes ? À quel temps de calcul faut-il s'attendre ?*

Les avantages/inconvénients des différentes méthodes sont

- *recalage uniforme* : rapide, mais pas très efficace à priori
- *recalage fonctionnel* : beaucoup plus lourd en calcul mais beaucoup plus efficace en alignement. On a un temps total d'à peu près 52 secondes pour le premier passage de `register.fd` pour ces 5 courbes. Pour 457 courbes, on peut donc estimer un temps total de 79 minutes. Il faudrait vérifier sur plus de courbes si la méthode de Procruste peut se montrer utile.
- *recalage par points de repère* : rapide en calcul, mais nécessite une annotation manuelle des courbes. Vu que le passage par 0 est facile à trouver, on peut imaginer que les 660 courbes peuvent être traitées en une demi heure en étant un peu motivé. Par contre, la proportion de la variation expliquée par la phase est bien plus petite que avec le recalage fonctionnel.
- *recalage automatique par point de repère* : moins efficace que le recalage par point de repère manuel, mais finalement pas si mal comme point de départ. Le temps de calcul est de 0.51 secondes pour 5 courbes et une estimation de 47 secondes pour le total. C'est très rapide.

On peut déduire que la meilleure solution à long terme est le recalage fonctionnel, probablement après un recalage automatique par point de repère.