Analysis on Simulated Data

Menghan Hu

3/9/2020

This document demonstrates the analysis (variance decomposition part) using simualted data.

```r
my_packages <- c("MASS", "ggplot2", "dplyr")
suppressMessages(lapply(my_packages, library, character.only = TRUE))
```

First, we generate the simualted data. Let T be an equally-spaced time grid from 0 to 1. Suppose there are 40 subjects, 15 lesions in each subject, and 5 layers within each lesion. We generate data from the following true model: $\begin{cases} Y_{ilj}(t) = \sum_{m=1}^{N_1} \phi_m^X(t)\xi_{im} + \sum_{h=1}^{N_2} \phi_h^U(t)\zeta_{ilh} + \sum_{k=1}^{N_3} \phi_k^W(t)\eta_{iljk} + \epsilon_{iljt} \\ \xi_{im} \sim N(0, \lambda_i^X), \zeta_{ilh} \sim N(0, \lambda_l^U), \eta_{iljk} \sim N(0, \lambda_j^W), \text{and } \epsilon_{iljt} \sim N(0, \sigma^2) \end{cases}$ where $i = 1, \ldots, I = 40$, $l = 1, \ldots, L = 15$, $j = 1, \ldots, J = 5$, $N_1 = N_2 = N_3 = 2$ and $t \in \mathcal{T} = 1/p, 2/p, \ldots, 1$.

```r
T <- seq(0, 1, length.out = 231)
p <- length(T)

L1=40
J=L2=15
L3=5
I_index=rep(1:L1,each=L2*L3)
n=length(I_index)
IJ_index=rep(1:(L1*L2),each=L3)

I=max(I_index)
n_I0=table(I_index)
IJ=max(IJ_index)
n_IJ=table(IJ_index)
```

Since we are only interested in the variation of latent processes, the term of overall mean is dropped here. The goal is to generate data that mimic the observed data structure, so we utilize the information obtained from the estimated eigenvalues and therefore use $\lambda_i^X = (4.5, 0.11)$, and $\lambda_l^U = \lambda_j^W = (90, 2)$. We also set $\phi^X(t) = \phi^W(t) = \left(\frac{5(e^{-2t})^2}{(1+e^{-t})^2}, \frac{8(e^{-2t})^2}{(1+e^{-t})^2} \right)$, $\phi^U(t) = (0.01, 0.05)$, and $\sigma = 0.1$.

```r
lambda1 <- c(4.5, 0.11)
lambda2 <- lambda3 <- c(90, 2)
N1 = length(lambda1)
N2 = length(lambda2)
N3 = length(lambda3)

phi1 = cbind(5*exp(-2*T)^2/(1+ exp(-T))^2,8*exp(-2*T)^2/(1+ exp(-T))^2)
phi2 = cbind(rep(0.01, length(T)),rep(0.05, length(T)))
phi3 = cbind(5*exp(-2*T)^2/(1+ exp(-T))^2,8*exp(-2*T)^2/(1+ exp(-T))^2)

#normalize bases
norm1 = t(phi1)%*%phi1
norm1 = sqrt(diag(norm1))
phi1 = t(t(phi1)/norm1)

norm2 = t(phi2)%*%phi2
```

```r
norm2 = sqrt(diag(norm2))
phi2 = t(t(phi2)/norm2)

norm3 = t(phi3)%*%phi3
norm3 = sqrt(diag(norm3))
phi3 = t(t(phi3)/norm3)

sigma = 0.1
```

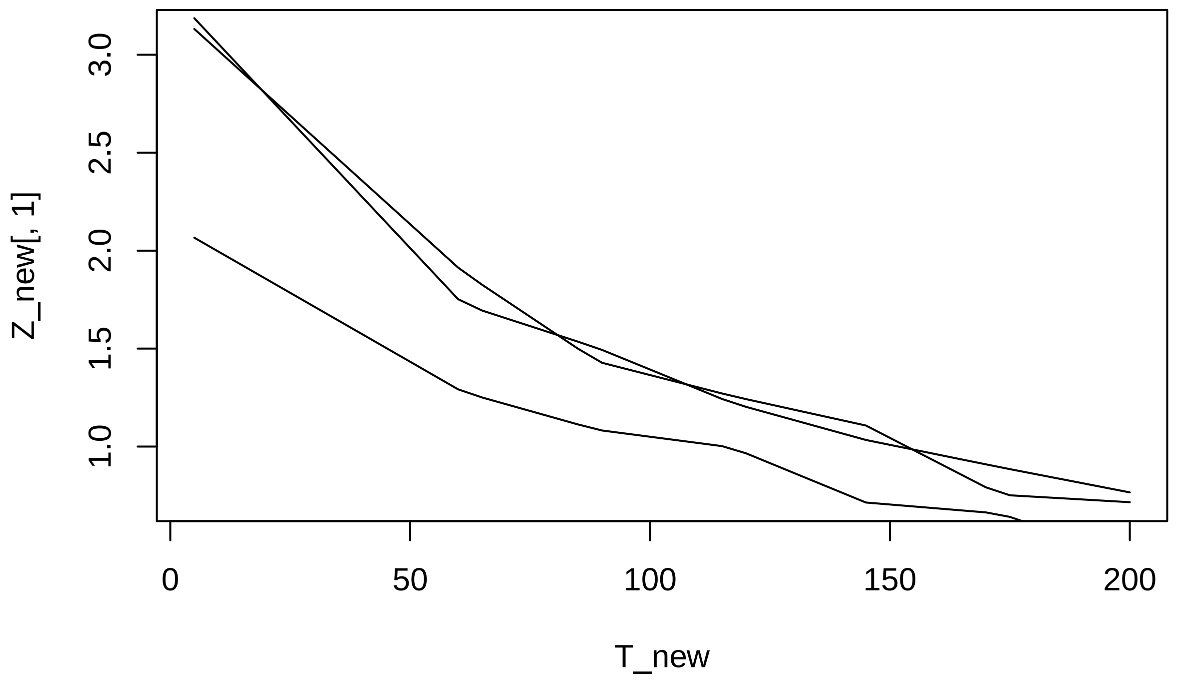
We use the following code to generate the data:

```r
set.seed(100)
xi = matrix(rnorm(N1*I),nrow=N1,ncol=I)
xi_true = xi * sqrt(lambda1)

theta = matrix(rnorm(N2*IJ),nrow=N2,ncol=IJ)
theta_true = theta * sqrt(lambda2)

zeta = matrix(rnorm(N3*n),nrow=N3,ncol=n)
zeta_true  = zeta * sqrt(lambda3)

for (s in 1:N1){
  xi_true[s, xi_true[s,] < 0] <- -1* xi_true[s, xi_true[s,] < 0]
}

for (s in 1:N2){
  theta_true[s, theta_true[s,] < 0] <- -1* theta_true[s, theta_true[s,] < 0]
}

  for (s in 1:N3){
    zeta_true[s, zeta_true[s,] < 0] <- -1* zeta_true[s, zeta_true[s,] < 0]
  }

firstLevel = phi1 %*% xi_true  #of size p x I
secondLevel = phi2 %*% theta_true  #of size p x IJ
thirdLevel = phi3 %*% zeta_true

noise = matrix(rnorm(p*n, mean=0,sd=sigma) ,nrow=p,ncol=n)
Z <- firstLevel[,I_index] + secondLevel[,IJ_index]+thirdLevel+ noise
```

To make the data closer to reality, we add variation for different observation days.

```r
Z_new <- c()
day.list <- list(
    c(1, 57, 85, 113, 141, 169, 200),
    c(1, 63, 91, 126, 154, 200),
    c(1, 63, 85, 113, 141, 176, 200),
    c(1, 56, 77, 98, 133, 154, 175, 200),
    c(1, 77, 98, 133, 154, 175, 200),
    c(1, 70, 98, 133, 154, 175, 200)
)
for (batch in 0:5){
    temp <- Z[(5*batch + 1):(200+5*batch), (500*batch+1):(500*(batch+1))]
    #temp <- Z[1:200, (500*batch+1):(500*(batch+1))]
    temp <- temp[day.list[[batch+1]], ]
    temp_interpolated <- apply(temp, 2, function(t) {
      interpolate = approx(x = day.list[[batch+1]],
                           y = t, xout = seq(1, 200, 5), method = 'linear')
      return(interpolate$y)
      }
      )
  Z_new <- cbind(Z_new, temp_interpolated)
}
```

Z_new has the same format as the data we analyzed in the paper (after preprocessing). The 40 rows are for 40 time points, and the number of columns is the number of layers in total. To visualize, we can randomly pick a few columns to plot:



Now we can decompos the variation by the sources as discussed in the paper. First, we smooth the data

```
smooth.y <- TRUE
  ### smooth the raw data
  if(smooth.y){
    Y_new <-list()
    Y_new<-lapply(1:n, function(i) {m<-smooth.spline(as.numeric(Z_new[,i]));  return(predict(m)$y)} )
    Y <-Reduce(cbind,Y_new)
  }
sd.noise<- sqrt(mean((Y-Z_new)^2))
```

Then, we calculate the covariance matrices K_X, K_U, and K_W.

```
  k1=sum(n_IJ^2)
  Y_IJ <- t(rowsum(t(Y),IJ_index))

  k2=sum(n_I0^2)
  Y_I <- t(rowsum(t(Y),I_index))

  HW <- (Y%*%diag(n_IJ[IJ_index])%*%t(Y))-(Y_IJ%*%t(Y_IJ))*2/(k1-n)
  HU <- (Y%*%diag(n_I0[I_index])%*%t(Y)-(Y_I)%*%t(Y_I)-(k1-n)/2*HW)*2/(k2-k1)
```

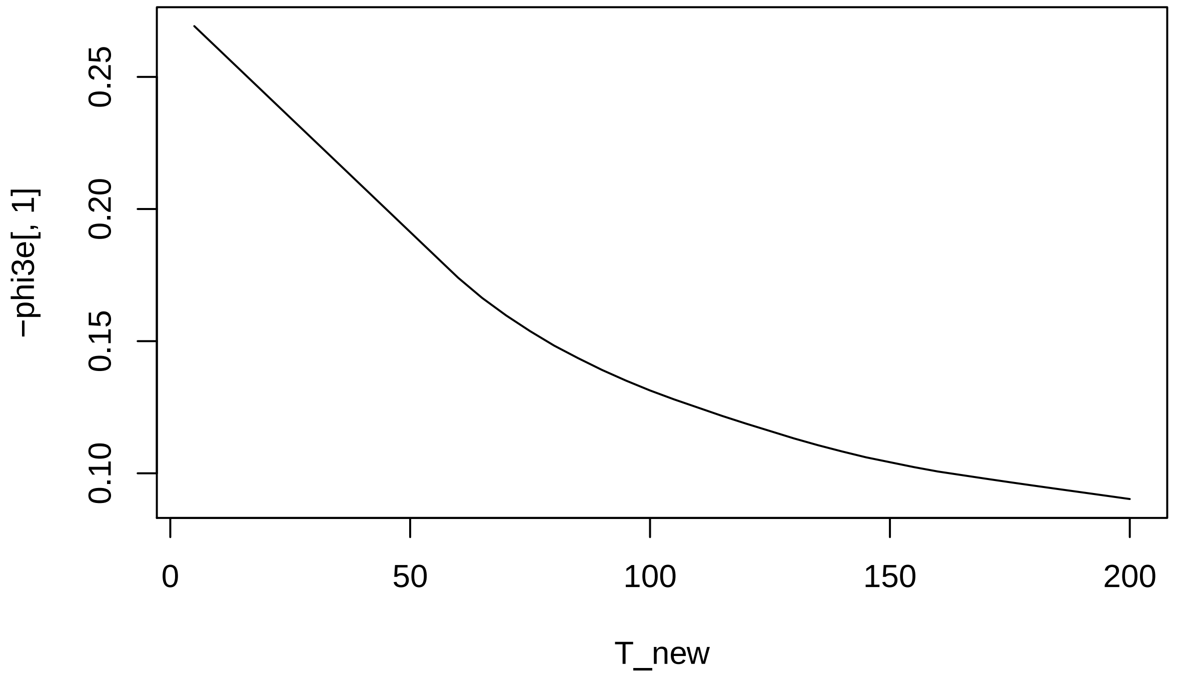
```r
  HX <- (n*Y%*%t(Y)-rowSums(Y)%*%t(rowSums(Y))-(k1-n)/2*HW-(k2-k1)/2*HU)*2/(n^2-k2)

  KW <- HW/2
  KU <- (HU - HW)/2
  KX <- (HX - HU)/2
```

The eigenfunctions and the eigenvalues then can be estimated.

```r
phi3e <- eigen(KW)$vectors[,1:N3]
phi2e<- eigen(KU)$vectors[,1:N2]
phi1e <- eigen(KX)$vectors[,1:N1]


lambda3e <- svd(KW)$d[1:N3]
lambda2e <- svd(KU)$d[1:N2]
lambda1e <- svd(KX)$d[1:N1]
```

The first principal component for layer level over time can be plotted.

```r
plot(T_new, -phi3e[,1], type="l")
```


−phi3e[, 1]
T_new