Stat 764 Assignment 4

Rigele Te

10/14/2020

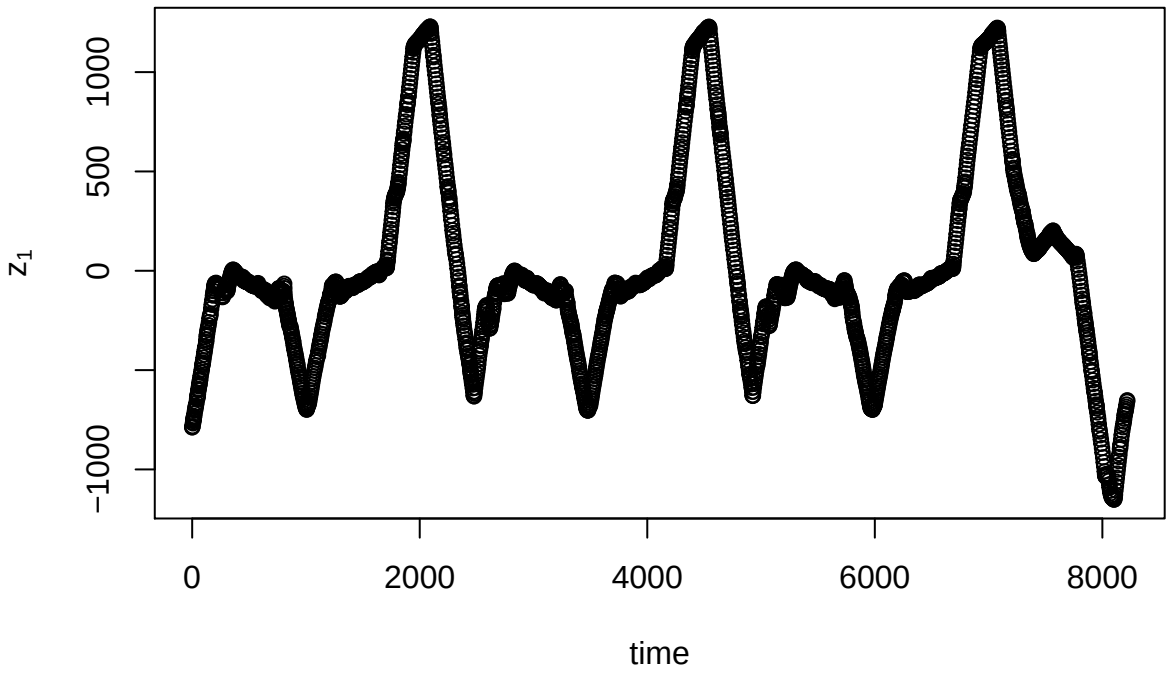
```
setwd("/Users/rigele/Documents/2020fall/stat764/assignment4")
library(maps)
library(maptools)
library(plotKML)
library(lubridate)
library(rgdal)
library(mgcv)
library(raster)
library(latex2exp)
```

```
url <- "https://www.dropbox.com/s/ak8gwm9sl1xikas/Olympic_trials.gpx?dl=1"
# ogrListLayers(url) # See name of layers are included in gpx file
pt.marathon <- readOGR(url, "track_points")
```

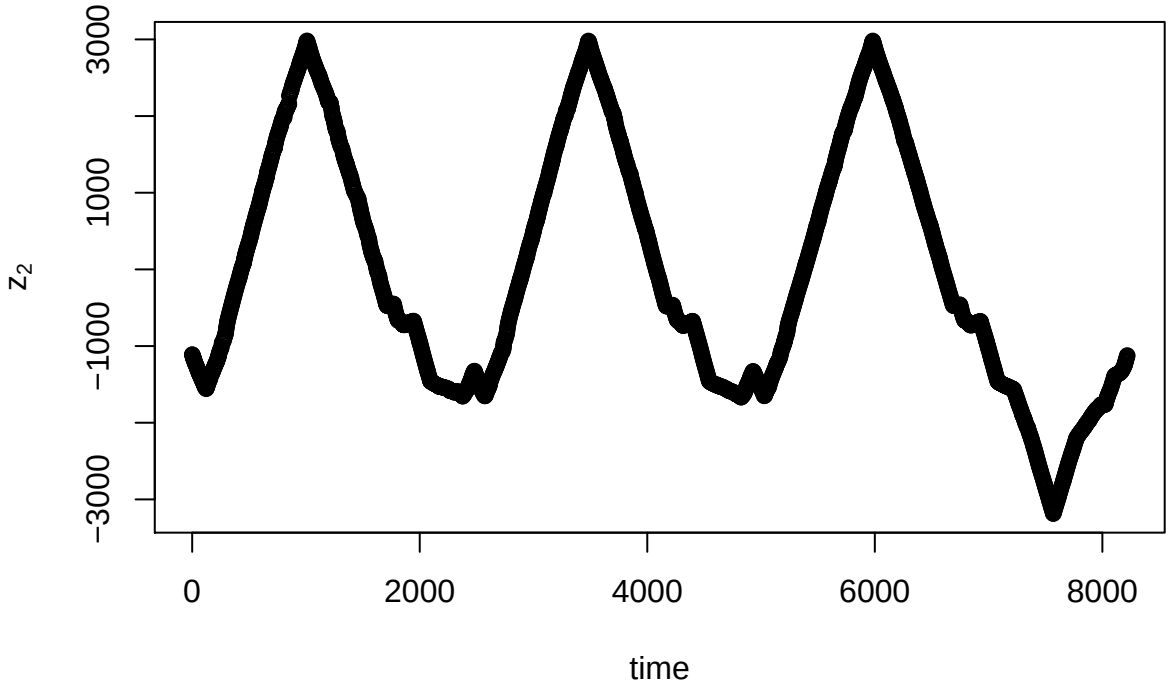
```
## OGR data source with driver: GPX
## Source: "https://www.dropbox.com/s/ak8gwm9sl1xikas/Olympic_trials.gpx?dl=1", layer: "track_points"
## with 2369 features
## It has 27 fields
```

```
# Change date/time to class 'POSIXct'
pt.marathon$time <- ymd_hms(pt.marathon$time, "%Y-%m-%d %H:%M:%S", tz = "America/Chicago")[-2370]

# Drop all extra variables
pt.marathon <- pt.marathon[, 5]
```

```
#Assign coordinate reference system to pt.marathon and then project to UTM zone 14
proj4string(pt.marathon)   <-  "+proj=longlat +zone=14 +datum=WGS84"
pt.traj <- spTransform(pt.marathon, CRS=CRS("+proj=utm +zone=14 +datum=WGS84"))

kml(pt.traj)


# Make data frame
df.traj <- data.frame(pt.traj)[,-4]
colnames(df.traj) <- c("t.obs","s1.obs","s2.obs")
df.traj$t.obs <- as.numeric(df.traj$t.obs - min(df.traj$t.obs))
df.traj$z1.obs <- df.traj$s1.obs - mean(df.traj$s1.obs)
df.traj$z2.obs <- df.traj$s2.obs - mean(df.traj$s2.obs)

#Plot time series of spatial location
#par(mfrow=c(2,1))
plot(df.traj$t.obs,df.traj$z1.obs,typ="p",xlab="time",ylab=expression(z[1]))
```




```
plot(df.traj$t.obs,df.traj$z2.obs,typ="p",xlab="time",ylab=expression(z[2]))
```



Q 1

This trajectory data records the Marathon route Chris Melgares run through for the Olympic trials that were held on January 22, 2020 in Atlanta. Chris Melgares finished 33rd with a time of 2 hours 16 min and 59 seconds. There are total 2369 data points and the starting point is next to the Marietta St @ Park Ave West. Chris then went down to the street and passed Intensive English Program, Georgia State University, FHLBank, Mayor's Park... After runing around the downtown area, he finally stoped around the visitor center in central Olympic park.

Q 2

Here I used Bayesian hierarchical model. The data model is: $[z_1|y_1, \sigma_1^2] = N(y_1, \sigma_1^2)$, $[z_2|y_2, \sigma_2^2] = N(y_2, \sigma_2^2)$. The process model is: $[y_1|\beta_{01}, \Sigma] = N(B_{01}, \Sigma_{y_1})$, $[y_2|\beta_{02}, \Sigma] = N(B_{02}, \Sigma_{y_2})$. $\Sigma_{ij} = \sigma^2 e^{-|t_i - t_j|/\phi}$. Where z_1, z_2 are the perdict value for longtitude and latitude . y_1, y_2 are the true data for longtitude and latitude. B_{01}, B_{02} are the coefficients calculated from the data model. t_i, t_j represents two different time lags.

Q 3

In the R code, I set 2020-02-29 11:08:53 to be the time lag 0, and when the time pass n second, the time lag will increase by n. The last recorede time in the dataset is 2020-02-29 13:25:52, thus the maximum timelag is 8219. For this question, I'm going to perdict the location at time 2020-02-29 11:38:53 to 2020-02-29 11:38:54, which means time passed 1800 and 1801 seconds.

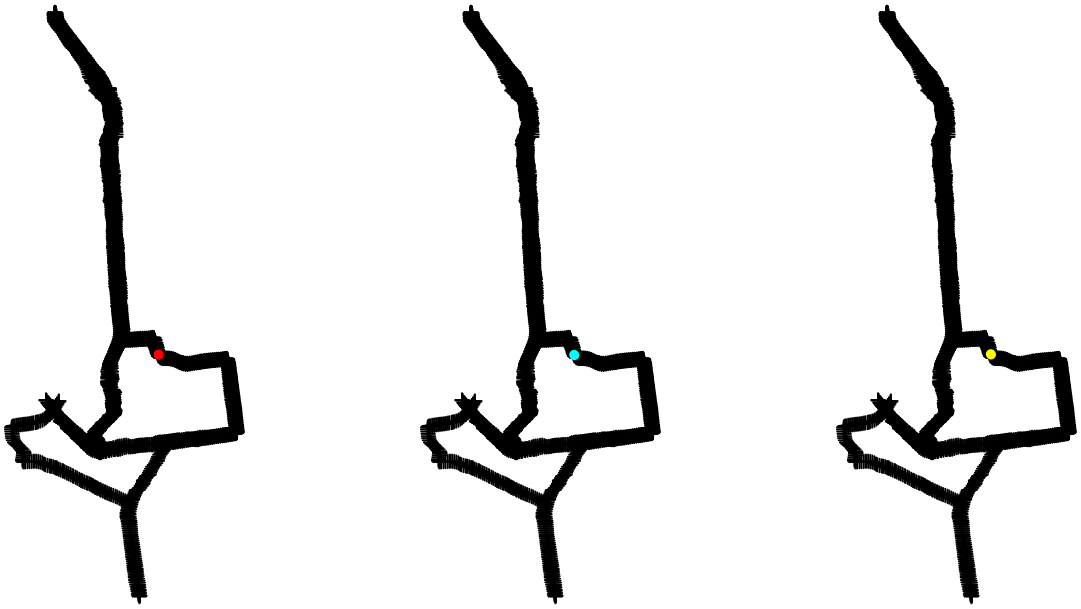
```
# Fit statistical model to data
df.temp <- data.frame(z = c(df.traj$z1.obs,df.traj$z2.obs),t = rep(df.traj$t.obs,2),latlong = rep(as.fa
m1 <- gam(z~-1+s(t,bs = "ad", by=latlong,k=400),family=gaussian(link = "identity"),data=df.temp)
# Make sure to experiment with dimension reduction (i.e., k in s(...,k=?))
summary(m1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## z ~ -1 + s(t, bs = "ad", by = latlong, k = 400)
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F p-value
## s(t):latlongz1 384.7  395.9  58138  <2e-16 ***
## s(t):latlongz2 387.7  396.5 626685  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =      1   Deviance explained =  100%
## GCV = 27.896  Scale est. = 23.349    n = 4738
```

```
# Specify time that predictions/forecasts are needed.
t.pred <- 1800:1801
#t.pred <- seq(min(df.temp$t),max(df.temp$t),by=1) #Time is in seconds

# Obtain draws from posterior predictive distribution
K.draws <- 1000
y1.ppd <- matrix(, K.draws, length(t.pred))
y2.ppd <- matrix(, K.draws, length(t.pred))
z1.ppd <- matrix(, K.draws, length(t.pred))
z2.ppd <- matrix(, K.draws, length(t.pred))
H.y1 <- predict(m1,newdata = data.frame(t = t.pred,latlong = "z1"),type="lpmatrix")
H.y2 <- predict(m1,newdata = data.frame(t = t.pred,latlong = "z2"),type="lpmatrix")

for (k in 1:K.draws) {
  y1.ppd[k,] <- H.y1%*%rmvn(1,coef(m1),vcov(m1))
  y2.ppd[k,] <- H.y2%*%rmvn(1,coef(m1),vcov(m1))
}
```

```r
# Summarize posterior predictive density
y1.pred.ev <- apply(y1.ppd,2,mean)
y1.pred.uci <- apply(y1.ppd,2,quantile,prob=0.975)
y1.pred.lci <- apply(y1.ppd,2,quantile,prob=0.025)
y2.pred.ev <- apply(y2.ppd,2,mean)
y2.pred.uci <- apply(y2.ppd,2,quantile,prob=0.975)
y2.pred.lci <- apply(y2.ppd,2,quantile,prob=0.025)
```

```r
# Make shapefiles of predicted trajectory
sf.y.pred.ev <- spLines(cbind(y1.pred.ev + mean(df.traj$s1.obs),y2.pred.ev + mean(df.traj$s2.obs)),crs=
sf.y.pred.lci <- spLines(cbind(y1.pred.lci + mean(df.traj$s1.obs),y2.pred.lci + mean(df.traj$s2.obs)),c
sf.y.pred.uci <- spLines(cbind(y1.pred.uci + mean(df.traj$s1.obs),y2.pred.uci + mean(df.traj$s2.obs)),c

# Plot predicted trajectory
par(mfrow=c(1,3))
plot(pt.traj)
plot(sf.y.pred.ev,lwd=5,col="red",add=TRUE)
plot(pt.traj)
plot(sf.y.pred.lci,lwd=5,col="cyan",add=TRUE)
plot(pt.traj)
plot(sf.y.pred.uci,lwd=5,col="yellow",add=TRUE)
```



The red point in the first plot is the predicted location, the green point in the second plot is the lower bound of that perdicted location and the yellow point in the third plot is the upper bound.

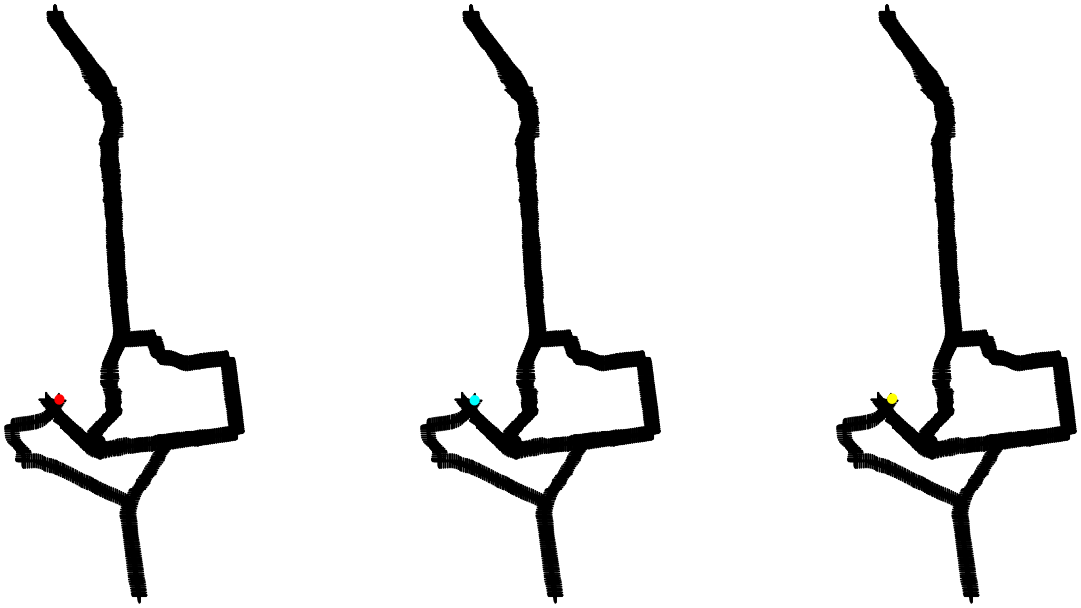
Q 4

Since the last recorede time in the dataset is 2020-02-29 13:25:52, with the timelag is 8219. For this question, I'm going to perdict the location at time 2020-02-29 13:25:53 to 2020-02-29 13:25:54, one to two seconds after the last recorded place. The time lag for this location is 8220 to 8221

```r
# Specify time that predictions/forecasts are needed.
t.pred <- 8220:8221
#t.pred <- seq(min(df.temp$t),max(df.temp$t),by=1) #Time is in seconds
```

```r
# Obtain draws from posterior predictive distribution
K.draws <- 1000
y1.ppd <- matrix(, K.draws, length(t.pred))
y2.ppd <- matrix(, K.draws, length(t.pred))
z1.ppd <- matrix(, K.draws, length(t.pred))
z2.ppd <- matrix(, K.draws, length(t.pred))
H.y1 <- predict(m1,newdata = data.frame(t = t.pred,latlong = "z1"),type="lpmatrix")
H.y2 <- predict(m1,newdata = data.frame(t = t.pred,latlong = "z2"),type="lpmatrix")

for (k in 1:K.draws) {
  y1.ppd[k,] <- H.y1%*%rmvn(1,coef(m1),vcov(m1))
  y2.ppd[k,] <- H.y2%*%rmvn(1,coef(m1),vcov(m1))
}

# Summarize posterior predictive density
y1.pred.ev <- apply(y1.ppd,2,mean)
y1.pred.uci <- apply(y1.ppd,2,quantile,prob=0.975)
y1.pred.lci <- apply(y1.ppd,2,quantile,prob=0.025)
y2.pred.ev <- apply(y2.ppd,2,mean)
y2.pred.uci <- apply(y2.ppd,2,quantile,prob=0.975)
y2.pred.lci <- apply(y2.ppd,2,quantile,prob=0.025)
```

```r
# Make shapefiles of predicted trajectory
sf.y.pred.ev <- spLines(cbind(y1.pred.ev + mean(df.traj$s1.obs),y2.pred.ev + mean(df.traj$s2.obs)),crs=
sf.y.pred.lci <- spLines(cbind(y1.pred.lci + mean(df.traj$s1.obs),y2.pred.lci + mean(df.traj$s2.obs)),c
sf.y.pred.uci <- spLines(cbind(y1.pred.uci + mean(df.traj$s1.obs),y2.pred.uci + mean(df.traj$s2.obs)),c

# Plot predicted trajectory
par(mfrow=c(1,3))
plot(pt.traj)
plot(sf.y.pred.ev,lwd=5,col="red",add=TRUE)
plot(pt.traj)
plot(sf.y.pred.lci,lwd=5,col="cyan",add=TRUE)
plot(pt.traj)
plot(sf.y.pred.uci,lwd=5,col="yellow",add=TRUE)
```



The red point in the first plot is the predicted location, the green point in the second plot is the lower bound of the perdicted location and the yellow point in the third plot is the upper bound.

Q 5

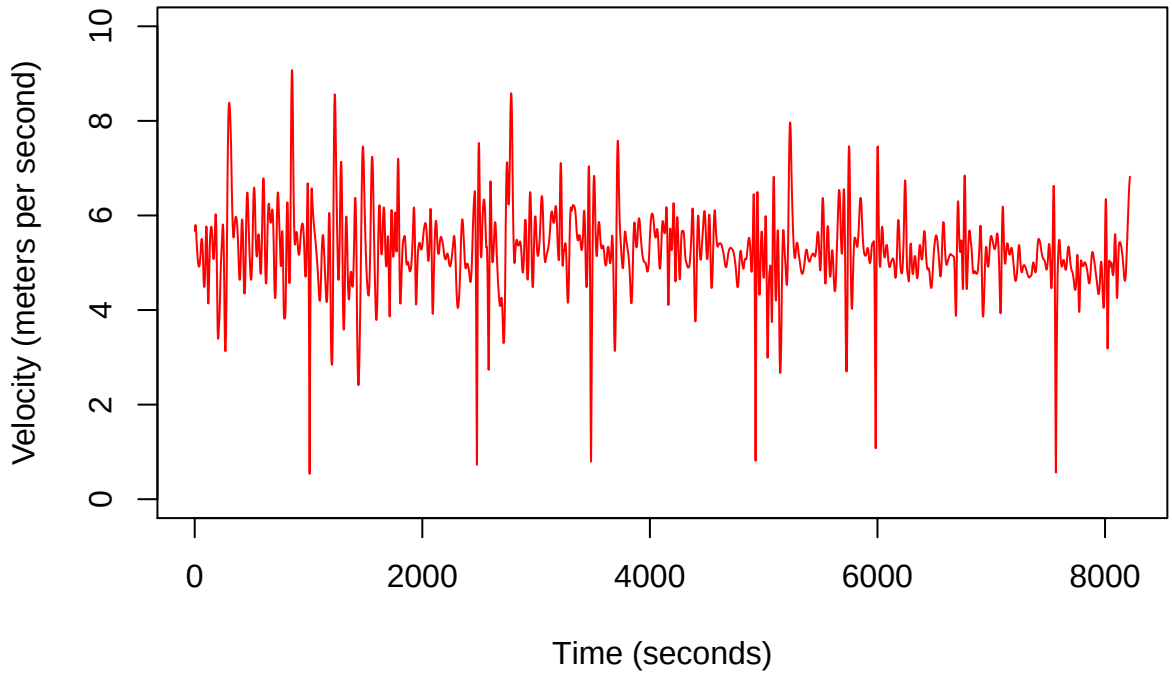
Here I estimate the velocity and acceleration in each locations.

```
# Specify time that predictions/forecasts are needed.
t.pred <- seq(min(df.temp$t),max(df.temp$t),by=1) #Time is in seconds

# Obtain draws from posterior predictive distribution
K.draws <- 1000
y1.ppd <- matrix(, K.draws, length(t.pred))
y2.ppd <- matrix(, K.draws, length(t.pred))
z1.ppd <- matrix(, K.draws, length(t.pred))
z2.ppd <- matrix(, K.draws, length(t.pred))
H.y1 <- predict(m1,newdata = data.frame(t = t.pred,latlong = "z1"),type="lpmatrix")
H.y2 <- predict(m1,newdata = data.frame(t = t.pred,latlong = "z2"),type="lpmatrix")

for (k in 1:K.draws) {
  y1.ppd[k,] <- H.y1%*%rmvn(1,coef(m1),vcov(m1))
  y2.ppd[k,] <- H.y2%*%rmvn(1,coef(m1),vcov(m1))
}
```

```
# Calculate derived quantities from estimated true location
velocity.ppd <- matrix(, K.draws, length(t.pred)-1)
acceleration.ppd <- matrix(, K.draws, length(t.pred)-2)

for(k in 1:K.draws) {
  location <- cbind(y1.ppd[k,],y2.ppd[k,])
  velocity.ppd[k,] <- rowSums((location[1:(dim(location)[1]-1),] - location[2:dim(location)[1],])^2)^0.5
  acceleration.ppd[k,] <- diff(velocity.ppd[k,])
}

velocity.ev <- apply(velocity.ppd,2,mean)
```

```
velocity.uci <- apply(velocity.ppd,2,quantile,prob=0.975)
velocity.lci <- apply(velocity.ppd,2,quantile,prob=0.025)
acceleration.ev <- apply(acceleration.ppd,2,mean)
acceleration.uci <- apply(acceleration.ppd,2,quantile,prob=0.975)
acceleration.lci <- apply(acceleration.ppd,2,quantile,prob=0.025)
```

```
# Plot estimated velocity and acceleration
plot(t.pred[-1],velocity.ev,ylim=c(0,10),typ="l",xlab="Time (seconds)",ylab="Velocity (meters per secon
```




```
#points(t.pred[-1],velocity.uci,typ="l",col="cyan")
#points(t.pred[-1],velocity.lci,typ="l",col="cyan")
```

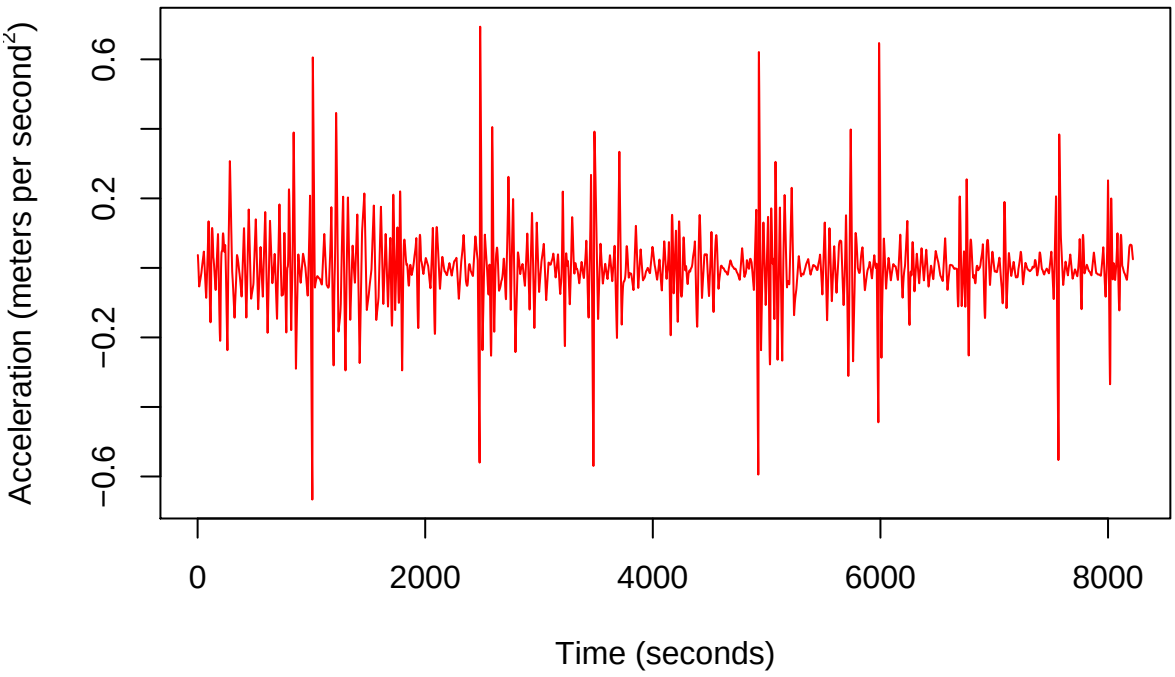
This plot gives the estimated velocity in each seconds. Chris has 6 points velocity almost comes to zero.

```
max(velocity.ev)
```

```
## [1] 9.071311
```

The maximum velocity is around 9.063752.

```
plot(t.pred[-c(1:2)],acceleration.ev,typ="l",xlab="Time (seconds)",ylab=expression("Acceleration (meter
```




```
#points(t.pred[-c(1:2)],acceleration.uci,typ="l",col="cyan")
#points(t.pred[-c(1:2)],acceleration.lci,typ="l",col="cyan")
```

This plot gives the estimated acceleration in each seconds. Chris has 6 points velocity almost comes to zero.

```
max(acceleration.ev)
```

```
## [1] 0.6939103
```

The maximum acceleration is around 0.6937509.