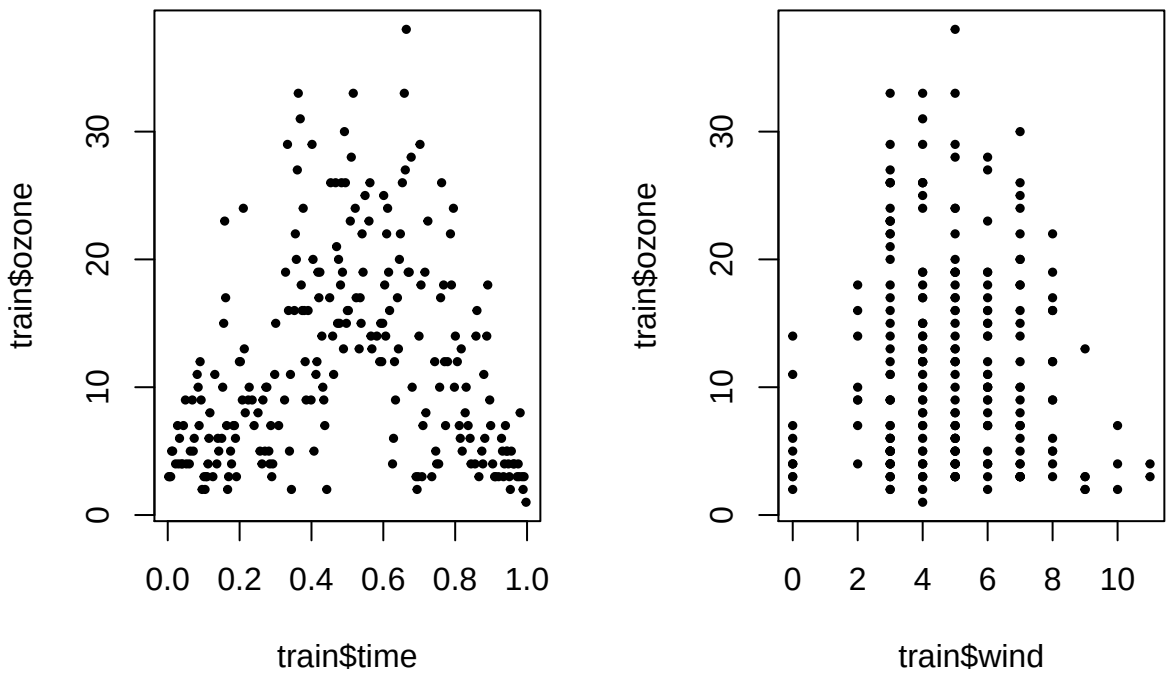
HW6

Question 1

For each method, produce a figure consists of training data, testing data and your fitted curve.

```
library(mlbench)
  data(Ozone)

  # Wind will only be used for Q2
  mydata = data.frame("time" = seq(1:nrow(Ozone))/nrow(Ozone), "ozone" = Ozone$V4, "wind" = Ozone$V6)

  trainid = sample(1:nrow(Ozone), 250)
  train = mydata[trainid, ]
  test = mydata[-trainid, ]
  par(mfrow=c(1,2))

  plot(train$time, train$ozone, pch = 19, cex = 0.5)
  plot(train$wind, train$ozone, pch = 19, cex = 0.5)
```



```
train = train[order(train$time),]
```

a.

Write your own code (you cannot use bs() or similar functions) to implement a continuous piecewise linear fitting. Pick 3 knots using your own judgment.

```
library(dplyr)
```

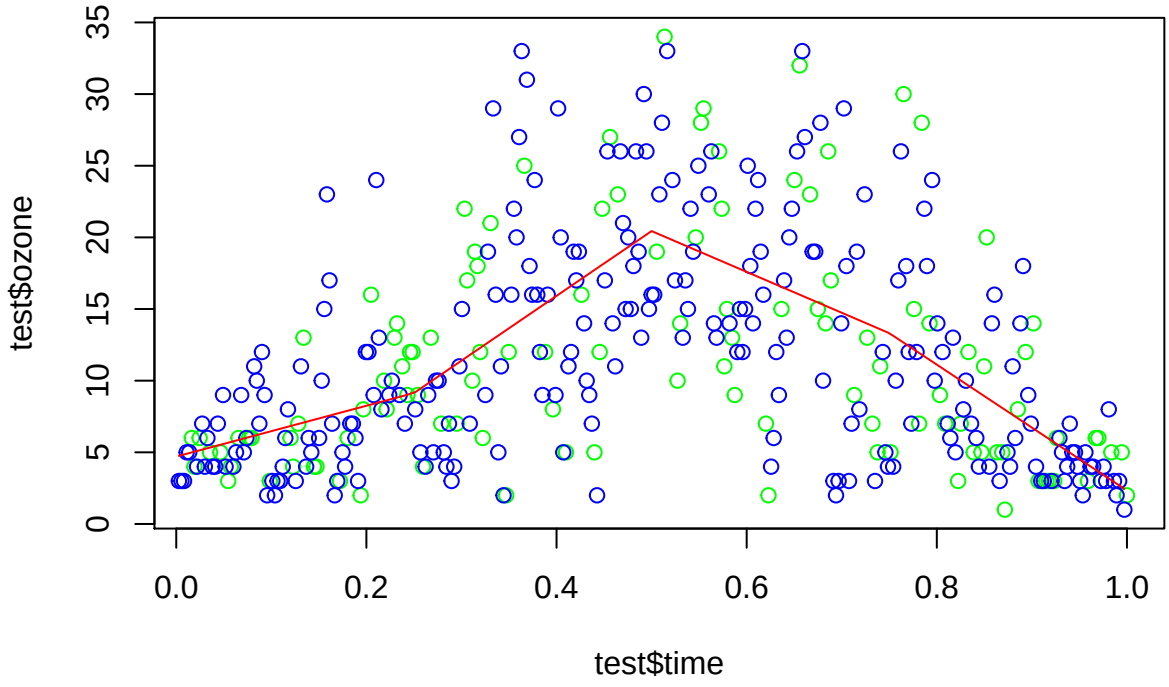
```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
knot = c(1/4, 1/2, 3/4)
train = na.omit(train)

h1 = rep(1, length(train$time))
h2 = train$time
H = cbind(h1,h2)

positive = function(x){
  output = max(0,x)
}


for ( i in 1:length(knot)){
   h = sapply(train$time-knot[i],positive)
   H = cbind(H,h)
}
y = train$ozone
colnames(H)=NULL
H = cbind(y,H)
df = as.data.frame(H)
fit = lm(y~., data = df)

y_hat = predict(fit, newdata =select(df, -y))
```

```
## Warning in predict.lm(fit, newdata = select(df, -y)): prediction from a rank-
## deficient fit may be misleading
```

```
lo <- loess(y_hat~train$time)

H1 = select(df, -y)
```

```
plot(x=test$time, y=test$ozone, col='green')
points(x=train$time, y=y, col='blue')
lines(x=as.vector(train$time),y=y_hat, col='red')
```



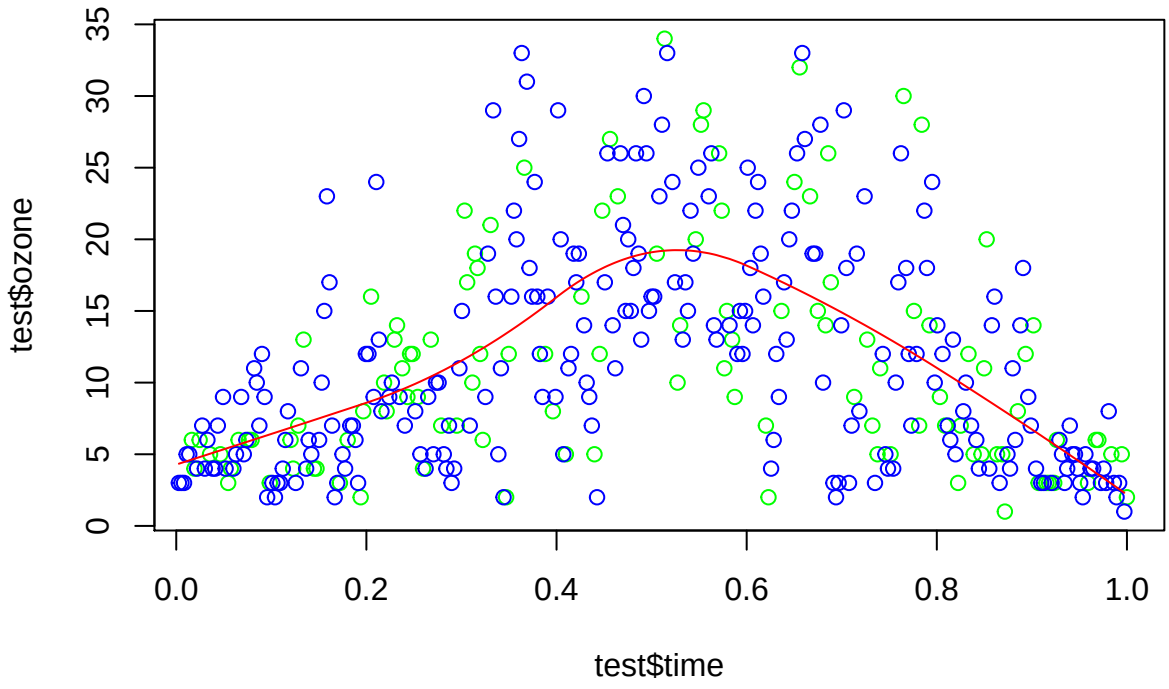
Write your own code to implement a quadratic spline fitting. Your spline should be continuous up to the first derivative. Pick 4 knots using your own judgment.

```
knot = c(0.2,0.4,0.6,0.8)

h1 = rep(1, length(train$time))
h2 = train$time
h3 = h2^2
H = cbind(h1,h2)
H = cbind(H,h3)

positive = function(x){
  output = (max(0,x))^2
}

for ( i in 1:length(knot)){
  h = sapply(train$time-knot[i],positive)
  H = cbind(H,h)
}

y = train$ozone
```

```
colnames(H)=NULL
H = cbind(y,H)
df = as.data.frame(H)


fit = lm(y~., data = df)
y_hat = predict(fit, newdata = select(df,-y))
```

```
## Warning in predict.lm(fit, newdata = select(df, -y)): prediction from a rank-
## deficient fit may be misleading
```

```
names(y_hat)=NULL


#plot(x=as.vector(train$time),y=y_hat, col='red')
plot(x=test$time, y=test$ozone, col='green')
points(x=as.vector(train$time), y=y, col='blue')
lines(x=as.vector(train$time),y=y_hat, col='red')
```



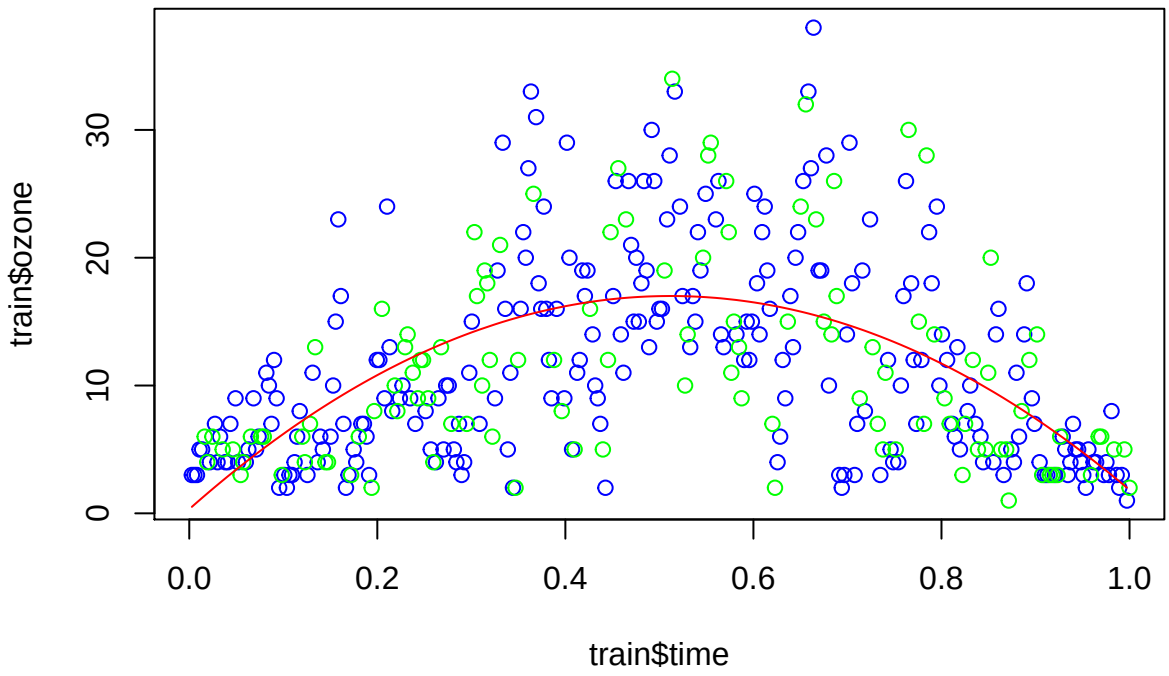
Produce a same set of basis as (ii) using the bs() function. Note that they do not have to be exactly the same as yours. Verify (figure out how) that the column spaces are the same.

```
library(splines)
library(pracma)
knot = c(0.2,0.4,0.6,0.8)
```

```
H = bs(train$time, degree = 2,intercept = TRUE,  knots = knot)

H_old = select(df,-y)
H_old = as.matrix(H_old)

fit = lm( train$ozone~ bs(train$time, degree=2, intercept = TRUE), data=train)
y_hat = predict(fit, newdata = select(df,-y))
```

```
## Warning in predict.lm(fit, newdata = select(df, -y)): prediction from a rank-
## deficient fit may be misleading
```

```
plot(x=train$time, y=train$ozone, col='blue')
points(x=test$time, y=test$ozone, col='green')
lines(x=train$time, y=y_hat, col = "red")
```




```
sum(rref(H)!=rref(H_old))
```

```
## [1] 0
```

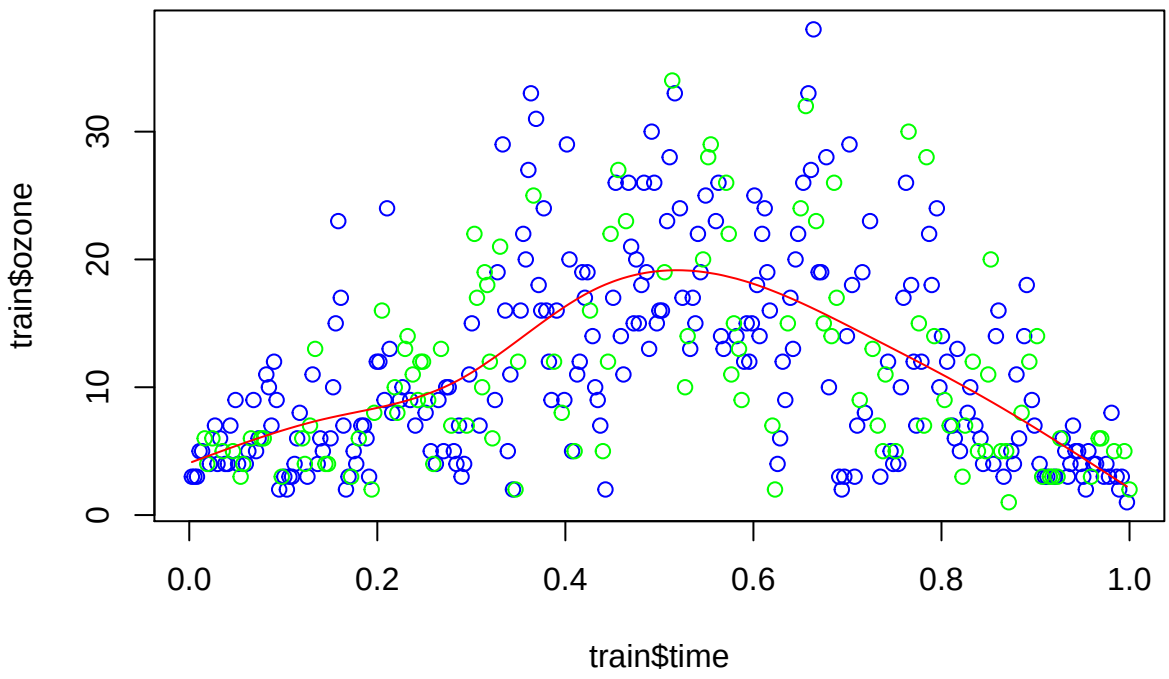
The row echelon form of both design matrix are the same, so their column spaces are the same.

Use existing functions to implement a natural cubic spline with 6 knots. Choose your own knots.

```
library(splines)
library(pracma)
knot = seq(0,1,length.out = 8)[2:7]

H = ns(train$time,intercept = TRUE,  knots = knot)

fit = lm(train$ozone~ns(train$time,intercept = TRUE, knots = knot), data=train)
y_hat = predict(fit, newdata = select(df,-y))
```

```
## Warning in predict.lm(fit, newdata = select(df, -y)): prediction from a rank-
## deficient fit may be misleading
```

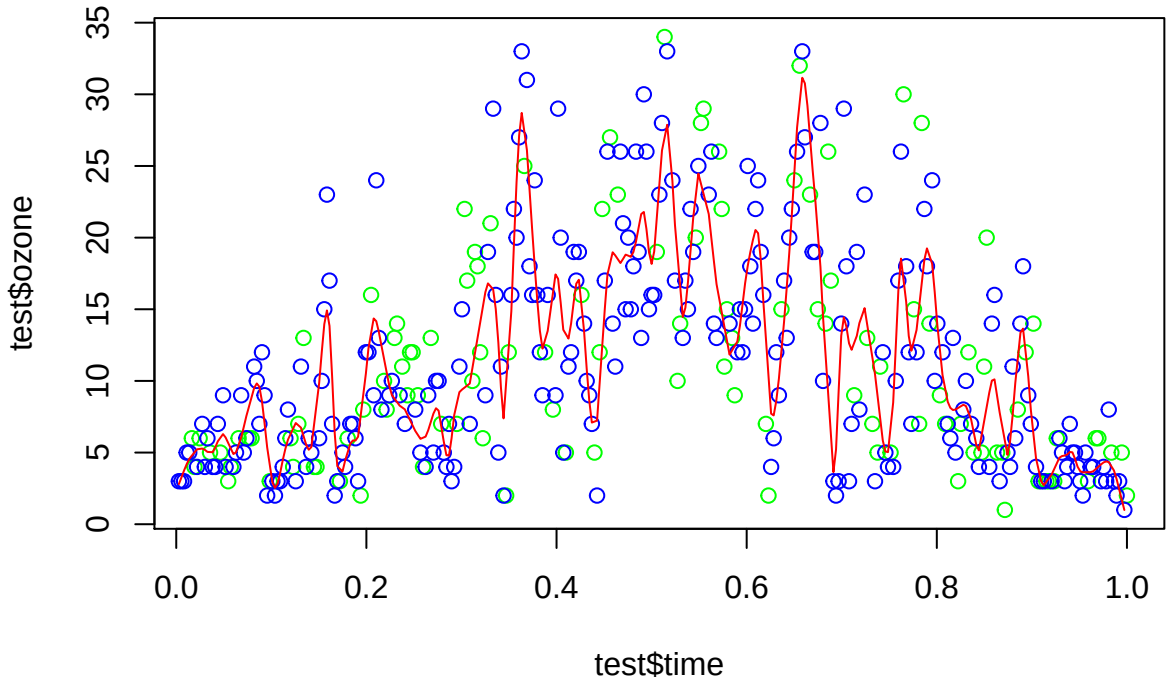
```
plot(x=train$time, y=train$ozone, col='blue')
points(x=test$time, y=test$ozone, col='green')
lines(x=train$time, y=y_hat, col = "red")
```



Use existing functions to implement a smoothing spline. Use the built-in generalized cross-validation method to select the best tuning parameter.

```
train = na.omit(train)
fit = smooth.spline(train$time, train$ozone)
y_hat = predict(fit, train$time)


plot(x=test$time, y=test$ozone, col='green')
```

```
points(x=train$time, y=train$ozone, col='blue')
lines(x=train$time, y = y_hat$y, col='red')
```




```
#wind
knot = c(0.2,0.4,0.6,0.8)

h1 = rep(1, length(train$wind))
h2 = train$wind
h3 = h2^2
H = cbind(h1,h2)
H = cbind(H,h3)

positive = function(x){
  output = (max(0,x))^2
}

for ( i in 1:length(knot)){
  h = sapply(train$wind-knot[i],positive)
  H = cbind(H,h)
}

y = train$ozone
colnames(H)=NULL
H = cbind(y,H)
df = as.data.frame(H)
```

```
names(y_hat)=NULL

names(H1)=NULL

df = as.data.frame(cbind(H1,H))

fit = lm(y~. ,data=df)
```

Test error

```
test = na.omit(test)
h1 = rep(1, length(test$time))
h2 = test$time
H = cbind(h1,h2)

positive = function(x){
  output = max(0,x)
}


for ( i in 1:length(knot)){
   h = sapply(test$time-knot[i],positive)
   H = cbind(H,h)
}
y = test$ozone
colnames(H)=NULL
H = cbind(y,H)
df = as.data.frame(H)
fit = lm(y~., data = df)

y_hat = predict(fit, newdata =select(df, -y))
```

```
## Warning in predict.lm(fit, newdata = select(df, -y)): prediction from a rank-
## deficient fit may be misleading
```

```
lo <- loess(y_hat~test$time)

H = select(df, -y)


knot = c(1/4, 1/2, 3/4)
test = na.omit(test)

h1 = rep(1, length(test$time))
h2 = test$time
H = cbind(h1,h2)

positive = function(x){
  output = max(0,x)
}
```

```
for ( i in 1:length(knot)){
   h = sapply(test$time-knot[i],positive)
   H = cbind(H,h)
}
y = test$ozone
colnames(H)=NULL
H = cbind(y,H)
df = as.data.frame(H)
fit = lm(y~., data = df)

y_hat = predict(fit, newdata =select(df, -y))
```

```
## Warning in predict.lm(fit, newdata = select(df, -y)): prediction from a rank-
## deficient fit may be misleading
```

```
lo <- loess(y_hat~test$time)

H1 = select(df, -y)


names(y_hat)=NULL
names(H1)=NULL
df_test = as.data.frame(cbind(H1,H))

y_hat = predict(fit, newdata = df_test)
```

```
## Warning in predict.lm(fit, newdata = df_test): prediction from a rank-deficient
## fit may be misleading
```

```
sum((y_hat-test$ozone)^2)/length(y_hat)
```

```
## [1] 37.09349
```

average prediction error is 28.8845