Homework 3

Author: Dong Bin

email: bindong2@illinois.edu

Question 1

Training data generation

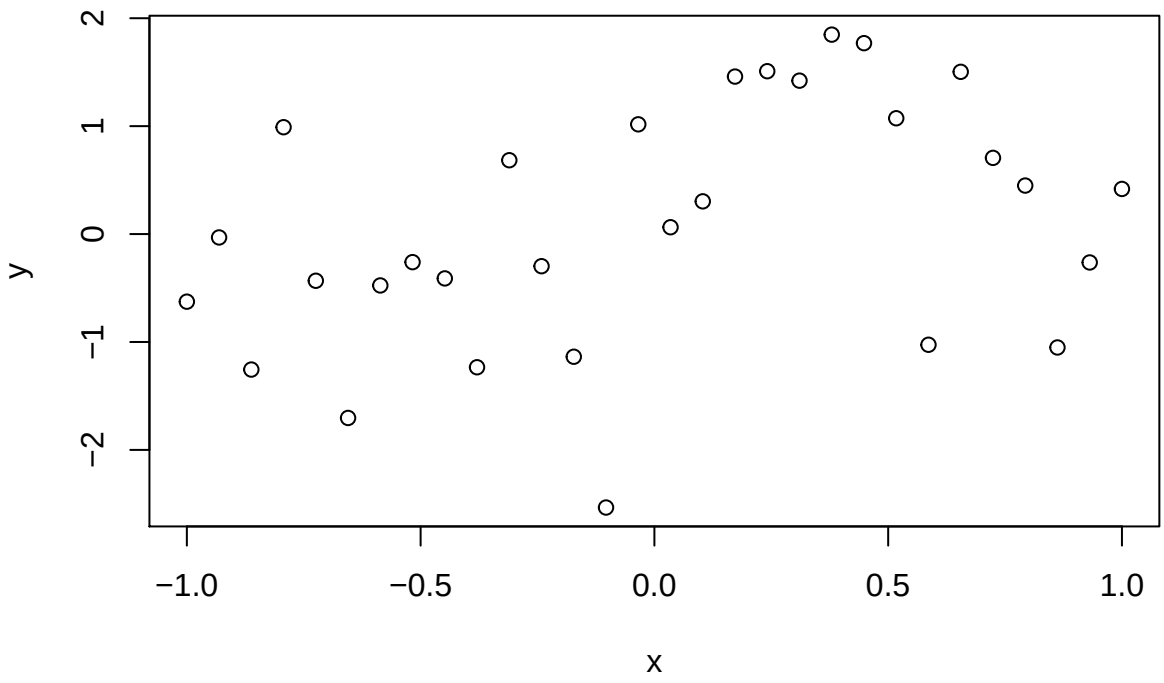
```
rm(list=ls())

set.seed(1)

x <- seq(from=-1, to=1, length.out = 30)

y <- sin(pi*x) + rnorm(30)

plot(x, y)
```




```
data <- data.frame("y"=y, "x"=x);

#https://teazrq.github.io/stat542/rlab/spline.html
```

Implement my own linear spline fitting

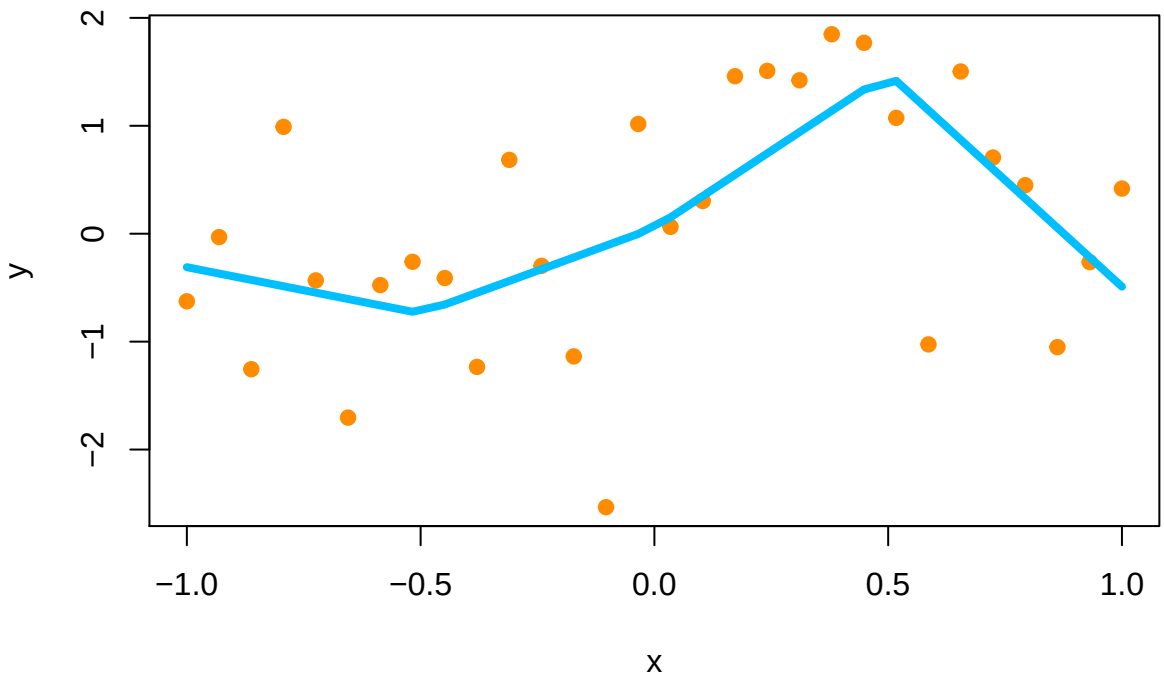
```
pos <- function(x){ x*(x>0); }

mybasis = cbind("int"=1, "x_1"=x, "x_2"=pos(x+0.5), "x_3"=pos(x), "x_4"= pos(x-0.5))


lmfit <- lm(y ~ .-1, data = data.frame(mybasis))

plot(x, y, pch=19, col = "darkorange")

lines(x, lmfit$fitted.values, lty=1, col="deepskyblue", lwd=4)
```




```
mypredict <- function(lmfit, x)
{
  predict(lmfit, data.frame(cbind("int"=1, "x_1"=x, "x_2"=pos(x+0.5), "x_3"=pos(x), "x_4"=pos(x-0.5))))
}

mypredict(lmfit, c(1.1, 2.2))
```

```
##          1          2
## -0.8857813 -5.2311621
```

Quadratic spline

```
library(splines)



fit.quad <- lm(y ~ bs(x, df=5, degree=2), data=data)

predict(fit.quad, newdata = data.frame("x"=c(0.3,0.4)))
```

```
##        1        2
## 1.384134 1.309693
```

Natural Cubic Spline

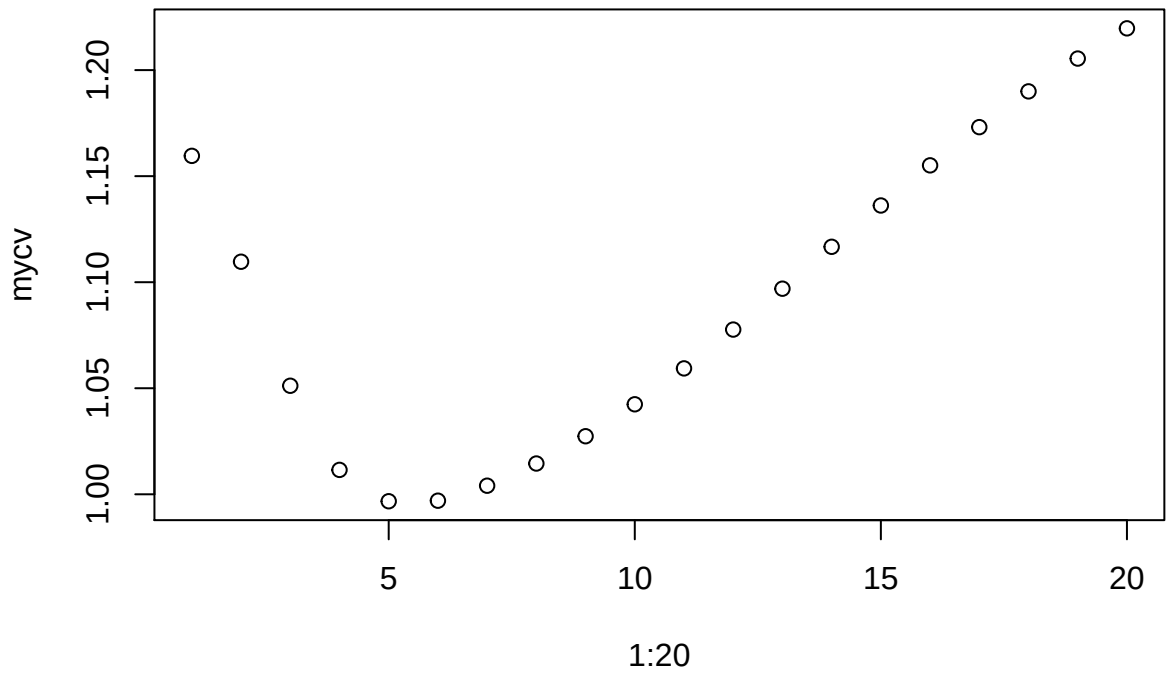
```
fit.cubic <- lm(y ~ ns(x, df=5), data=data)
predict(fit.cubic, newdata = data.frame("x"=c(0.3,0.4)))
```

```
##        1        2
## 1.395978 1.285278
```

Smoothing spline

```
df = 2+(1:20)/2
m = length(df)
mycv = rep(0, m)
for(i in 1:m){
  fit = smooth.spline(x, y, df=df[i], cv=T);
  mycv[i] = fit$cv
}

plot(1:20, mycv)
```




```
# Best df is 5

fit.smooth = smooth.spline(x, y, df=5)

predict(fit.smooth, c(0.3,0.4))$y
```

```
## [1] 1.082777 1.124850
```

Compare

```
x.test <- seq(from=-1, to=1, length.out = 1000)
y.test <- sin(pi*x.test)

predict.myfit <- mypredict(lmfit, x.test)
predict.quad <- predict(fit.quad, newdata=data.frame("x"=x.test))
predict.cubic <- predict(fit.cubic, newdata = data.frame("x"=x.test))
predict.smooth <- predict(fit.smooth, x.test)$y

err.myfit <- sum((predict.myfit-y.test)^2)
err.quad <- sum((predict.quad-y.test)^2)
err.cubic <- sum((predict.cubic-y.test)^2)
err.smooth <- sum((predict.smooth-y.test)^2)
```

Repeat 200 times

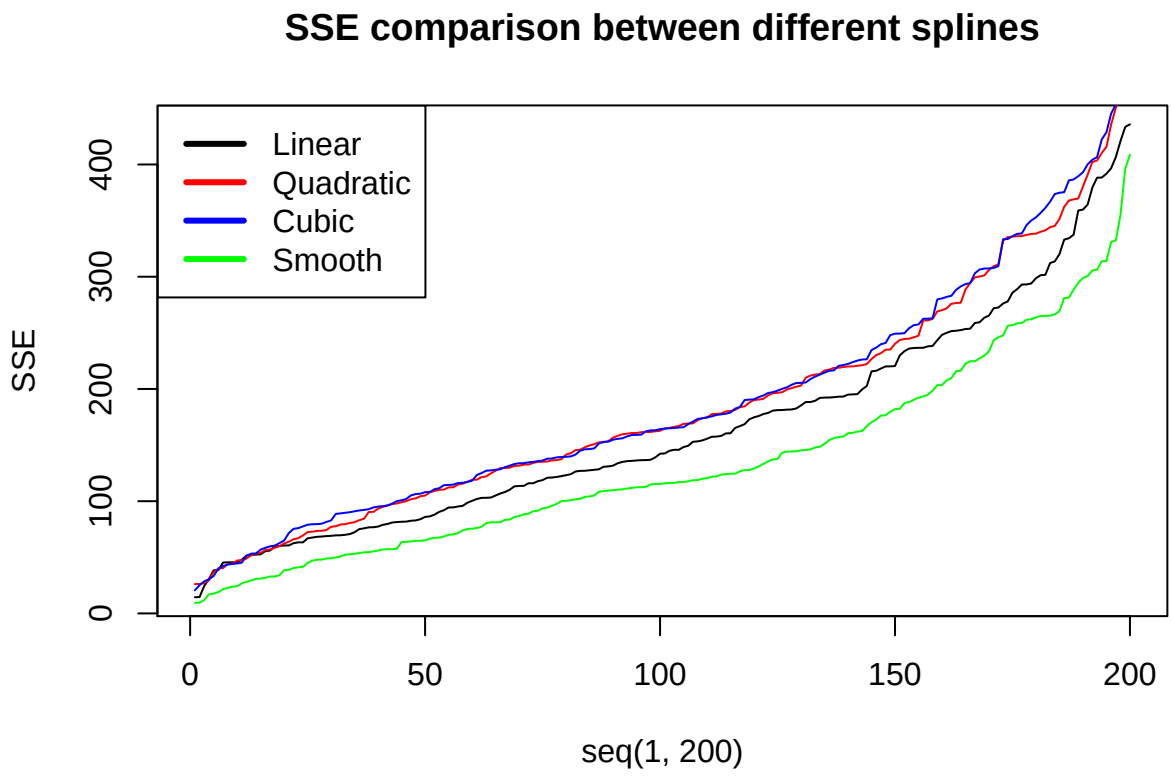
```
errs.myfit <- rep(0, 200)
errs.quad <- rep(0, 200)
errs.cubic <- rep(0, 200)
errs.smooth <- rep(0, 200)
for(i in 1:200)
{
  set.seed(i)
  x <- seq(from=-1, to=1, length.out = 30)
  y <- sin(pi*x) + rnorm(30)
  data <- data.frame("y"=y, "x"=x);

  lmfit <- lm(y ~ .-1, data = data.frame(mybasis))
  fit.quad <- lm(y ~ bs(x, df=5, degree=2), data=data)
  fit.cubic <- lm(y ~ ns(x, df=5), data=data)
  fit.smooth = smooth.spline(x, y, df=5)

  predict.myfit <- mypredict(lmfit, x.test)
  predict.quad <- predict(fit.quad, newdata=data.frame("x"=x.test))
  predict.cubic <- predict(fit.cubic, newdata = data.frame("x"=x.test))
  predict.smooth <- predict(fit.smooth, x.test)$y

  errs.myfit[i] <- sum((predict.myfit-y.test)^2)
  errs.quad[i] <- sum((predict.quad-y.test)^2)
  errs.cubic[i] <- sum((predict.cubic-y.test)^2)
  errs.smooth[i] <- sum((predict.smooth-y.test)^2)
}
```

```
errs.myfit.sort <- sort(errs.myfit)
errs.quad.sort <- sort(errs.quad)
errs.cubic.sort <- sort(errs.cubic)
errs.smooth.sort <- sort(errs.smooth)
plot(seq(1, 200),errs.myfit.sort, type="l", ylab="SSE")
lines(seq(1, 200), errs.quad.sort, col="red")
lines(seq(1, 200), errs.cubic.sort, col="blue")
lines(seq(1, 200), errs.smooth.sort, col="green")
legend("topleft", c("Linear", "Quadratic", "Cubic", "Smooth"), col = c("black", "red", "blue", "green")
title("SSE comparison between different splines")
```




```
cat("Linear: Mean: ", mean(errs.myfit), ", Median: ", median(errs.myfit), ", Std: ", sqrt(var(errs.myfi
```

```
## Linear: Mean:  164.8971 , Median:  142.4148 , Std:  95.52712
```

```
cat("Quadratic: Mean: ", mean(errs.quad), ", Median: ", median(errs.quad), ", Std: ", sqrt(var(errs.quad
```

```
## Quadratic: Mean:  185.4388 , Median:  163.7482 , Std:  107.4016
```

```
cat("Cubic: Mean: ", mean(errs.cubic), ", Median: ", median(errs.cubic), ", Std: ", sqrt(var(errs.cubic
```

```
## Cubic: Mean:  188.9678 , Median:  164.4383 , Std:  109.7267
```

```
cat("Smooth: Mean: ", mean(errs.smooth), ", Median: ", median(errs.smooth), ", Std: ", sqrt(var(errs.sm
```

```
## Smooth: Mean:  133.179 , Median:  115.6563 , Std:  85.28623
```

Comparing the performance of different spline regression, the best is smoothed spline, with lower square error and more stable performance, linear is also a good choice.

I think one of the reasons that cubic/quadratic is not performing better than linear splines is that the epsilon is large compared with true value. thus making it less correlated to quadratic and cubic terms.

Question 2

Train Test Split

```
rm(list=ls())
set.seed(1)
data <-read.csv("Folds5x2_pp.csv")
sample <- sample.int(n=nrow(data), size = floor(2*nrow(data)/3), replace=FALSE)

train <- data[sample,]
test <- data[-sample,]

train.x <- train[,!(colnames(train)%in% c("PE"))]
test.x <- test[,!(colnames(test)%in% c("PE"))]

train.y <- train[, "PE"]
test.y <- test[, "PE"]

head(train.x)
```

```
##         AT     V      AP    RH
## 1017 14.55 41.70 1018.58 80.80
## 8004  8.84 42.49 1010.28 89.09
## 4775 28.09 70.02 1010.84 51.29
## 8462 23.39 74.22 1009.83 86.05
## 4050 33.15 68.51 1012.90 51.91
## 8789 28.86 67.90 1006.29 57.19
```

Multi-dimensional Kernel

```
p <- ncol(data)-1;
n <- nrow(data)
# Use AT, V, AP, RH to predict PE

lambda.k <- function(v)
{
  return <- (4/(p+2))^(1/(p+4))*n^(-1/(p+4))*sqrt(v);
}

lambda <- c(lambda.k(var(data$AT)),lambda.k(var(data$V)),lambda.k(var(data$AP)),lambda.k(var(data$RH)))

lambda.matrix <- matrix(rep(lambda, nrow(train.x)), ncol=4, byrow=TRUE)


K.lambda <- function(x.i, x.j, lambda)
{
  result.matrix <-((x.i-x.j)/lambda)^2
  result.col <- apply(result.matrix, 1, sum)
  return <- exp(-0.5 * result.col)
}
```

```
predictions <- rep(0, nrow(test.x))

for(i in 1:nrow(test.x))
{
  test.sample <- test.x[i,];
  test.matrix <- test.sample[rep(seq_len(nrow(test.sample)), nrow(train.x)), ]
  ks <- K.lambda(test.matrix, train.x, lambda.matrix)
  predictions[i] <- sum(ks*train.y)/sum(ks)
}
```

Calculate SSE and compare with linear fittings

```
err<-sum(predictions-test.y)^2/nrow(test.x)

cat("Error of my Multidimensional Kernal model is: " , err, "\n")
```

```
## Error of my Multidimensional Kernal model is:  1.643024
```

```
linear.fit <- lm(PE ~ . , data=train)

linear.predictions <- predict(linear.fit, newdata = test.x)

linear.err <- sum(linear.predictions-test.y)^2/nrow(test.x)

cat("Error of linear model is: ", linear.err, "\n")
```

```
## Error of linear model is:  6.489032
```