Homework3 Ji Qi

Problem 1 (Degree of freedom for smoothing splines)

$y = f(x_i) + \varepsilon_i$, where $\varepsilon_i \sim N(0, \sigma^2)$.

$$ASR(\lambda) = E_{\varepsilon_i}\left(\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2\right)$$

$y_i^* = f(x_i) + \varepsilon_i^*$, $\varepsilon_i^* \sim N(0, \sigma^2)$,

$$PE(\lambda) = E_{\varepsilon_i, \varepsilon_i^*}\left(\frac{1}{N}\sum_{i=1}^{N}(y_i^* - \hat{f}_\lambda(x_i))^2\right).$$

Start with computing ASR. We have

$$\begin{aligned} ASR(\lambda) &= \frac{1}{N} E_{\varepsilon_i}\left(\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2\right) \\ &= \frac{1}{N} E\left((y - S_\lambda y)^T (y - S_\lambda y)\right) \\ &= \frac{1}{N} E\left(((I - S_\lambda)y)^T (I - S_\lambda)y\right) \\ &= \frac{1}{N} E\left(y^T (I - S_\lambda)^T (I - S_\lambda) y\right) \end{aligned}$$

While

$$\begin{aligned} ASR(\lambda) &= \frac{1}{N} E\left(tr\left(y^T (I - S_\lambda)^2 y\right)\right) \\ &= \frac{1}{N} tr\left((I - S_\lambda)^2 E(yy^T)\right) \\ &= \frac{1}{N} tr\left((I - S_\lambda)^2 (ff^T + \sigma^2 I)\right) \\ &= \frac{1}{N}\left(f^T (I - S_\lambda)^2 f\right) + \frac{\sigma^2}{N} tr(I - 2S_\lambda + S_\lambda^2) \\ &= \frac{1}{N} f^T (I - S_\lambda)^2 f + \sigma^2 - \frac{2\sigma^2}{N} tr(S_\lambda) \\ &\quad + \frac{\sigma^2}{N} tr(S_\lambda^2) \quad (1) \end{aligned}$$

$$PE(\lambda) = \frac{1}{N} E_{\varepsilon_i, \varepsilon_i^*}\left(\sum_{i=1}^{N} (y_i - \hat{f}_\lambda(x_i))^2\right)$$

$$= \frac{1}{N} E_{\varepsilon_i, \varepsilon_i^*}\left(\sum_{i=1}^{N} (f(x_i) + \varepsilon_i^* - \hat{f}_\lambda(x_i))^2\right)$$

$$= \frac{1}{N} E_{\varepsilon_i, \varepsilon_i^*}\left(\sum_{i=1}^{N} \left((f(x_i) - \hat{f}(x_i))^2 + 2\varepsilon_i^*(f(x_i) - \hat{f}_\lambda(x_i)) + \varepsilon_i^{*2}\right)\right)$$

$$= \frac{1}{N} E_{\varepsilon_i}\left(\sum_{i=1}^{N} (f(x_i) - \hat{f}_\lambda(x_i))^2\right) + \frac{2}{N} \sum_{i=1}^{N} E_{\varepsilon_i^*}(\varepsilon_i^*) E_{\varepsilon_i}((f(x_i) - \hat{f}_\lambda(x_i)))$$

$$+ \frac{1}{N} E_{\varepsilon_i^*}\left(\sum_{i=1}^{N} \varepsilon_i^{*2}\right)$$

$$= \frac{1}{N} E((f - S_\lambda y)^T (f - S_\lambda y)) + \sigma^2$$

$$= \frac{1}{N}(f - S_\lambda f)^T (f - S_\lambda f) + \frac{1}{N} E(\varepsilon^T S_\lambda^2 \varepsilon) + \sigma^2$$

$$= \frac{1}{N} f^T (I - S_\lambda)^2 f + \frac{\sigma^2}{N} tr(S_\lambda^2) + \sigma^2 \quad (2)$$

Hence, from (1) (2) we have

$$PE(\lambda) - ASR(\lambda) = \frac{2\sigma^2}{N} tr(S_\lambda)$$

Problem 2 Boostrap

1. model: $y = f(x) + \varepsilon$ with $x \sim U[0,1]$, $\varepsilon \sim N(0, 0.2^2)$

Let $f(x) = 1 + x^2$. We want to estimate $\hat{f}(1.3)$.

Write a function for taking a sample and delivering an estimate for $\hat{x}_0$.

```r
set.seed(1234)
n = 50
x = runif(n, 0, 1)
noise = rnorm(n, 0, 0.2)
y = rep(1,n) + x^2 +noise
```

```r
data = as.data.frame(cbind(x, y))
data$x2 = data$x^2
model = lm(y ~ x2 + x, data =data)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x2 + x, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42676 -0.15765 -0.04785  0.11788  0.57410
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.8487     0.1054   8.051 2.14e-10 ***
## x2            0.5004     0.4840   1.034    0.306
## x             0.5330     0.4907   1.086    0.283
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2198 on 47 degrees of freedom
## Multiple R-squared:  0.6156, Adjusted R-squared:  0.5993
## F-statistic: 37.64 on 2 and 47 DF,  p-value: 1.747e-10
```

Write a function for taking a sample and delivering an estimate for ^x0.

```r
bootstrap = function(data){
  choices = sample(1:n, size=n, replace=T)
  bsdata = as.data.frame(cbind(data$x[choices], data$x2[choices], data$y[choices]))
  colnames(bsdata) = c("x", "x2", "y")
  bs_model = lm(y ~ x2 + x, data=bsdata)
  w0 = bs_model$coefficients[1]
  w2 = bs_model$coefficients[2]
  w1 = bs_model$coefficients[3]
  x0_hat = (-w1 + sqrt(w1^2 - 4*w2*(w0-1.3))) / (2*w2)
  return(x0_hat)
}
```

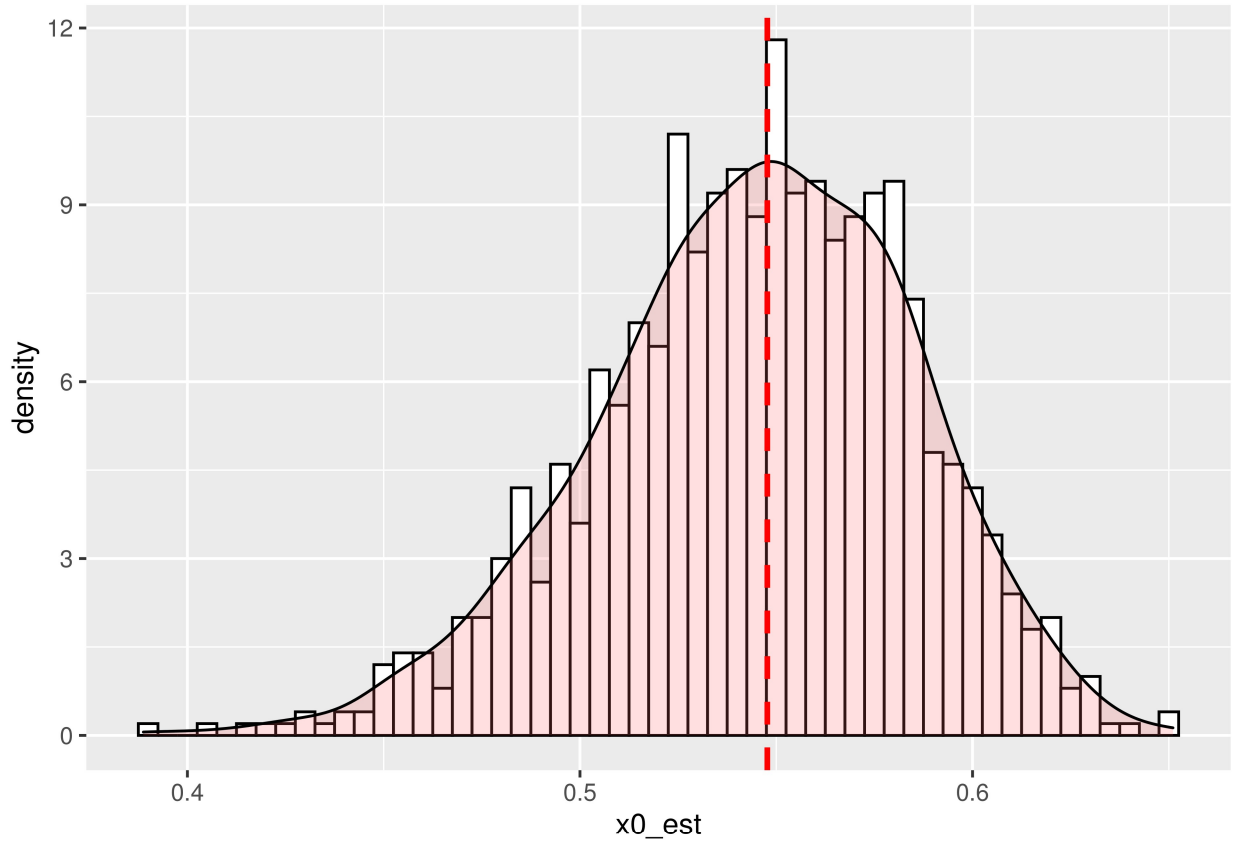
Generate a sample from this model, and compute ^x0 for this sample.

Simulate a 1000 realizations from this model, to approximate the sampling distribution of ^x0.

```r
x0_est = vector()
for (i in 1:1000){
  n = 50
  x = runif(n, 0, 1)
  noise = rnorm(n, 0, 0.2)
  y = rep(1,n) + x^2 +noise
  data = as.data.frame(cbind(x, y))
```

```
  data$x2 = data$x^2
  model = lm(y ~ x2 + x, data = data)
  w0 = model$coefficients[1]
  w2 = model$coefficients[2]
  w1 = model$coefficients[3]
  x0_hat = (-w1 + sqrt(w1^2 - 4*w2*(w0-1.3))) / (2*w2)
  x0_est[i] = x0_hat
}

library(ggplot2)
x0_est = as.data.frame(x0_est)
ggplot(x0_est, aes(x=x0_est)) +
    geom_histogram(aes(y=..density..),
                   binwidth=0.005,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")+
  geom_vline(aes(xintercept=sqrt(0.3)),    # Ignore NA values for meany
               color="red", linetype="dashed", size=1)# Overlay with transparent density plot
```



Using just your original sample, compute the bootstrap distribution of ^x0 .

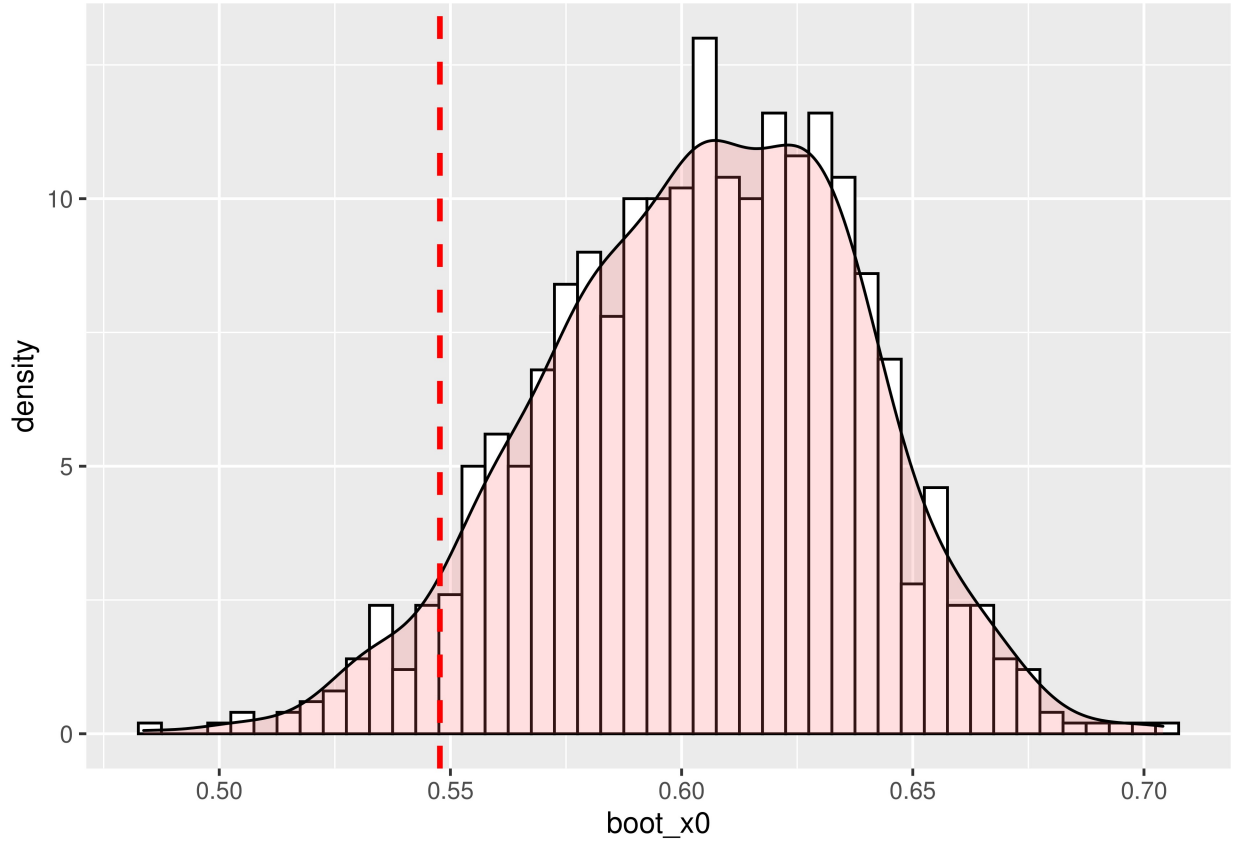
```
boot_x0 = vector()
print(sqrt(0.3))
```

```
## [1] 0.5477226
```

```
for (i in 1:1000){
  boot_x0[i] = bootstrap(data)
```

```
}
```

```
library(ggplot2)
boot_x0 = as.data.frame(boot_x0)
ggplot(boot_x0, aes(x=boot_x0)) +
    geom_histogram(aes(y=..density..),      # Histogram with density instead of count on y-axis
                   binwidth=0.005,
                   colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666")+
  geom_vline(aes(xintercept=sqrt(0.3)),    # Ignore NA values for mean
               color="red", linetype="dashed", size=1)# Overlay with transparent density plot
```



Compare these two distributions, and in particular their standard deviations.

```
print(sqrt(var(as.vector(x0_est))))
```

```
##            x0_est
## x0_est 0.04009306
```

```
print(sqrt(var(as.vector(boot_x0))))
```

```
##            boot_x0
## boot_x0 0.03355649
```

Here, we can see that the bootstrap result has less variance than the simulation result does, though with larger bias.

Problem 3. Decision trees

$(m^*, j^*, t^*_{m^*,j^*}) = \arg\min_{m,j,t_{m,j}} \min_{c^l, c^r} \left(\sum_{i:x_i \in R_m} (y_i - c^l)^2 + \sum (y_i - c^r)^2 - \sum_{i:x_i \in R_m} (y_i - \bar{y}_m)^2 \right)$.

1. Show that under this squared-error loss criterion, give $t_{j,m}$, $c^l = \bar{y}^l_m$, $c^r = \bar{y}^r_m$ are the response averages.

To get the minimization of Loss, we have

$\frac{\partial Loss}{\partial c^l} = -2 \sum_{i:x_i \in t_{m,j}} (y_i - c^l) = 0$. Hence, $c^l = \frac{\sum_{i:x_{i,j} \in t_{j,m}} y_i}{N^l_m}$.

Similarly, we have $c^r = \frac{\sum_{i:x_{i,j} \in \bar{t}_{j,m}} y_i}{N^r_m}$.

2. Show that the optimal rule $(m^*, j^*, t^*_{m^*,j^*})$ is a solution to the following problem:

$(m^*, j^*, t^*_{m^*,j^*}) = \arg\max_{m,j,t_{m,j}} \frac{N^l_m N^r_m}{N^l_m + N^r_m} (\bar{y}^l_m - \bar{y}^r_m)^2$

$\bar{y}_m = \frac{N^l_m \bar{y}^l_m + N^r_m \bar{y}^r_m}{N^l_m + N^r_m}$

$Loss = \sum_{left} \left((y_i - \bar{y}^l_m)^2 - (y_i - \bar{y}_m)^2 \right) + \sum_{right} \left((y_i - \bar{y}^r_m)^2 - (y_i - \bar{y}_m)^2 \right)$

$$= \sum_{\text{left}} \left(2(\bar{y}_m - \bar{y}_m^l) y_i + \bar{y}_m^{l\,2} - \bar{y}_m^2\right) + \sum_{\text{right}} \left(2(\bar{y}_m - \bar{y}_m^r) y_i + \bar{y}_m^{r\,2} - \bar{y}_m^2\right)$$

$$= 2(\bar{y}_m - \bar{y}_m^l) N_m^l \bar{y}_m^l + \bar{y}_m^{l\,2} N_m^l - \bar{y}_m^2 N_m^l + 2(\bar{y}_m - \bar{y}_m^r) N_m^r \bar{y}_m^r + \bar{y}_m^{r\,2} N_m^r - \bar{y}_m^2 N_m^r$$

Plug the $\bar{y}_m$ in, we have Loss =

$$= -\frac{N_m^l N_m^r}{N_m^l + N_m^r} \bar{y}_m^{l\,2} - \frac{N_m^l N_m^r}{N_m^l + N_m^r} \bar{y}_m^{r\,2} + \frac{N_m^l N_m^r}{N_m^l + N_m^r} \bar{y}_m^l \bar{y}_m^r$$

$$= -\frac{N_m^l N_m^r}{N_m^l + N_m^r} (\bar{y}_m^l - y_m^r)^2$$

Hence, $\text{argmin Loss} = \underset{m,j.t_{m,j}}{\text{argmax}} \frac{N_m^l N_m^r}{N_m^l + N_m^r} (\bar{y}_m^l - y_m^r)^2$

3. In (2), we have

$$(m^*, j^*, t^*_{m^*,j^*}) = \underset{m,j,t_{m,j}}{\operatorname{argmax}} \frac{N^l_m N^r_m}{N_m} (\bar{y}^l_m - \bar{y}^r_m)^2,$$

where $t_{m,j} = (\bigcap_{j' \neq j} S_{j',m}) \cap [a_{j,m}, s_{j,m})$,

$$\bar{t}_{m,j} = (\bigcap_{j' \neq j} S_{j',m}) \cap [s_{j,m}, b_{j,m}).$$

Hence for any j, m, we only need to consider values in $\{x_{i,j} : x_j \in R_m\}$ to find $s^*_{j,m}$. We can simply order values in $\{x_{i,j} : x_j \in R_m\}$ and try each value as the partition point. There are N_m choices for partition. Also, for calculating $\bar{y}^l_m$ and $\bar{y}^r_m$, once we move $s^*_{j,m}$ to the nearest right value in $\{x_{i,j} : x_j \in R_m\}$, we have

$$\bar{y}^l_{m(new)} = \frac{\bar{y}^l_{m(old)} \cdot N^l_{m(old)} + y_i}{N^l_{m(old)} + 1}$$

$$\bar{y}^r_{m(new)} = \frac{\bar{y}^r_{m(old)} \cdot N^r_{m(old)} - y_i}{N^l_{m(old)} - 1}$$

Hence, for each step it takes $O(1)$ complexity.

The whole algorithm takes $O(N_m)$.

4. The total computational complexity for determine $(m^*, j^*, s^*_{m,j})$ is $\sum_{n=1}^{m} \sum_{j=1}^{p} O(N_m) \leq O(pN)$

Problem 4. Show that
$E_p(RSS_j^* - RSS) = 2\hat{\beta}_j^2$

$RSS = (Y - X\hat{\beta})^T (Y - X\hat{\beta})$
$RSS_j^* = (Y - X_j\hat{\beta})^T (Y - X_j\hat{\beta})$, where X_j is the value of X but with j-th variable permuted.

$RSS_j^* - RSS = 2Y(X - X_j)\hat{\beta} + \hat{\beta}^T(X_j^T X_j - X^T X)\hat{\beta}$

$2Y^T(X - X_j)\hat{\beta} = 2\hat{\beta}_j Y^T(X_j - X_j^*)$ since X and X_j are identical except the j-th column. Here, x_j^* denotes the permutation of original X_j.
Since $\hat{\beta} = (X^T X)^{-1} X^T Y = I X^T Y = X^T Y$. Also, $E_p(X_j^*) = \bar{X}_j = 0$
$E_p(2\hat{\beta}_j Y^T(X_j - X_j^*)) = 2\hat{\beta}_j Y^T X_j = 2\hat{\beta}_j \hat{\beta}_j^T = 2\hat{\beta}_j^2$. Moreover,

$E(\hat{\beta}^T(X_j^T X_j - X^T X)\hat{\beta}) = 0$ under the assumption of $X^T X = I$. Hence, $E_p(RSS_j^* - RSS) = 2\hat{\beta}_j^2$.