MATH 297 Assignment 1

Zongze Liu

02 Feburary 2022

We first set up the libraries and import the data. We let n be the total number of observations and let k be $n/\log n$ rounded down. K will serve as a high degree of freedom and will be used to compute an esitmation of variance later.

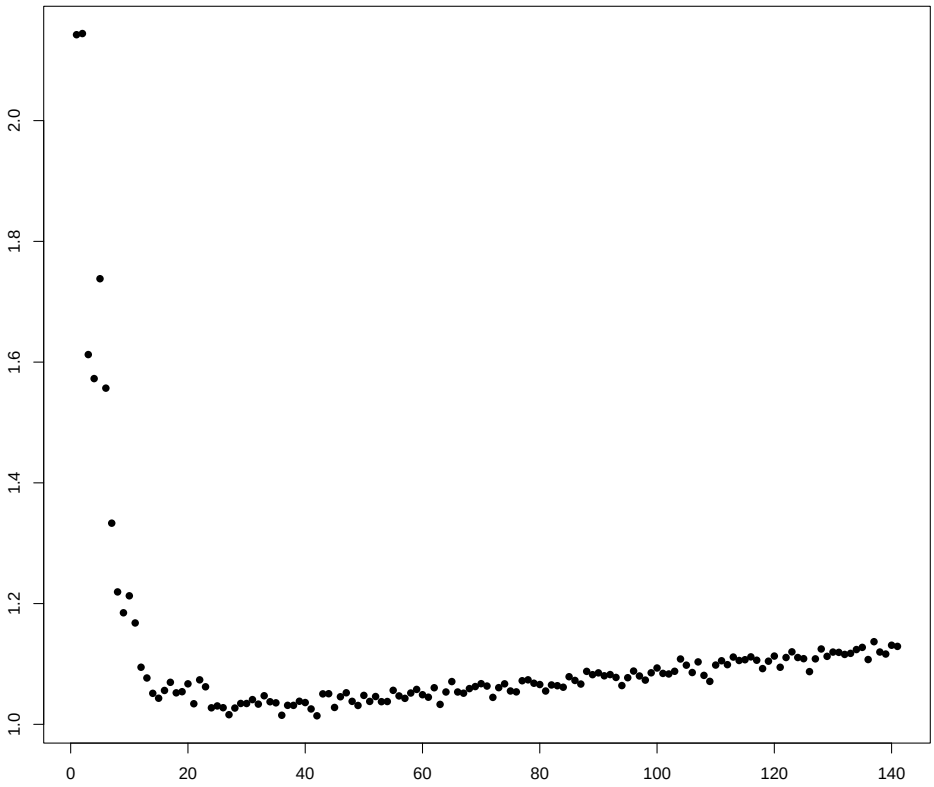
```
> library(tidyverse)
> library(HoRM)
> tmp = read_csv("C:/Users/zongze liu/Desktop/tmp-20160413.csv")
> tmp = as.data.frame(tmp)
> names(tmp) = c("xx", "yy")
> dat = tmp
> attach(dat)
> n = length(xx)
> k = round(n/log(n))
```

1 Regressogram

We will first write the function $regress_residue$ to compute the sum of squared error for a given number of bins(nbins). We note that the first five lines of codes come from the source code for regressogram function in the package HORM.

```
> regress_residue = function(x, y, nbins){
+   xy <- data.frame(x = x,y = y)
+   xy <- xy[order(xy$x),]
+   z <- cut(xy$x,breaks=seq(min(xy$x),max(xy$x),length=nbins+1),
+            labels=1:nbins,include.lowest=TRUE)
+   xyz <- data.frame(xy,z=z)
+   MEANS <- c(by(xyz$y,xyz$z,FUN=mean))
+
+   residue = 0
+   for (i in 1:n){
+     residue = residue + (xyz$y[i] - MEANS[xyz$z[i]])^2
+   }
+   return(residue)
+ }
```

Figure 1: Number of bins in regressogram vs estimated errors



We next estimate the variance of the regressogram model by the residue sum of square in a model with very high degree of freedom i.e.with a very high number of bins(nbins = n/log(n)).

```
> sigma_hat = regress_residue(dat$xx, dat$yy, k)/(n - k)
```

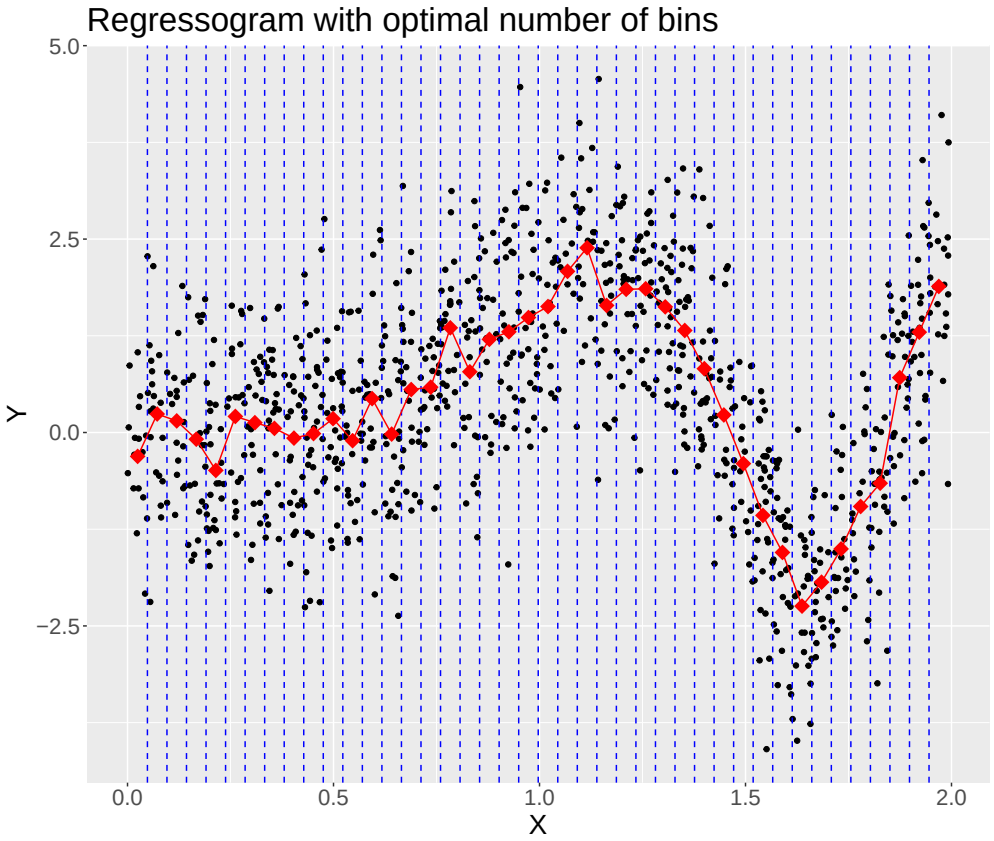
Then we use an array of size k to record the estimated errors of regressogram models with i bins where i ranges from 1 to k. The estimated error is given by the formula

$$\widehat{Err} = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2/n + 2\widehat{\sigma}i/n$$

```
> err_estimates = rep(0, times = k)
> for(i in 1:k){
+   err_estimates[i] = regress_residue(dat$xx, dat$yy, i)/n + 2*sigma_hat*i/n
+ }
```

We plot the number of bins in the regressogram models against the esitmated errors in figure 1. The optimal model will have 42 number of bins Finally, we will plot the regressogram model with the optimal number of bins.

Figure 2: Regressogram



2 Paper Summary

The paper studies the relations between two types of theories of prediction error. The first theory is covariance penalty methods including Mallow's C_p, Akaike's Information Criterion(AIC), and Stein's Unbiased Risk Estimates(SURE). The second theory includes cross-validation and related nonparametric bootstrap methods. The paper establishes a Rao-Blackwell type connection between covariance penalty and cross validation. More precisely, Theorem 1 in section 4 shows that the average of cross-validation estimates (across the non-parametric bootstrap data sets) used to calculate the conditional covariances approximates the covariance penalty (using paremetric bootstrap data sets) very well. The result implies that covariance penalties are more accurate than cross-validation, assuming that we trust the parametric bootstrap model. Theorem 2 in Section 6 gives a similar Rao-Blackwell type relation between covariance penalties with parametric bootstrap and nonparametric bootstrap techniques.

In the process of showing the above theorems, the paper developed explicit formulas for covariance penalty and error estimation. Since degrees of freedom in model estimation rules are defined in terms of covariance penalty, we can find the optimal degrees of freedom for various models, including linear spline models and kernel regression models, using the covariance penalty formulas. We start with a homoskedastic model

$$\boldsymbol{y} \sim (\boldsymbol{\mu}, \sigma^2 I)$$

where components of y are uncorrelated and y_i has mean μ_i and covariance σ^2. We let $\widehat{\boldsymbol{\mu}} = m(\boldsymbol{y})$ be a general estimation rule and we use squared error $Q(y_i, \widehat{\mu})$ for prediction error:

$$Q(y_i, \widehat{\mu}) = (y_i - \widehat{\mu})^2$$

Assuming $\widehat{\boldsymbol{f}}$ is the assumed density of $\boldsymbol{y}$, in the Gaussian case, we let $\widehat{\boldsymbol{f}} = N(\widehat{\boldsymbol{\mu}}, \widehat{\sigma}^2 I)$ and we obtain $\widehat{\sigma}^2$ from the residuals of some big model with negligible bias. Then we generate a large number 'B' of simulated observations and estimates from $\widehat{\boldsymbol{f}}$. Finally we can estimate the covariance penalty $cov_i := cov(\widehat{\mu}_i, y_i)$ from the observed bootstrap covariance:

$$\widehat{cov}_i = \sum_{b=1}^{B} \widehat{\mu}^{*b}(y_i^{*b} - y_i^{*\cdot})/(B-1) \quad where \quad y_i^{*\cdot} = \sum_b y_i^{*b}/B$$

This gives us the estimate of prediction error:

$$\widehat{Err} = err + 2\sum_{i=1}^{n} \widehat{cov}_i$$

In the case of a linear estimation rule $\widehat{\boldsymbol{\mu}} = M\boldsymbol{y}$, the degree of freedom is defined as $trace(M)$, for a general estimation rule $\widehat{\boldsymbol{\mu}} = m(\boldsymbol{y})$, we extend the definition as

$$df = \sum_{i=1}^{n} cov(\widehat{\mu}_i, y_i)/\sigma^2$$

And this is the base for problem 1,3,4,5 in the assignment. Furthermore, we can extend the squared prediction error to a wide class of error measures as shown in Section 3 of the paper. For any concave function $q(x)$ with a real-valued argument, we can define the q-class error measures for outcome y and prediction $\widehat{\mu}$ by

$$Q(y, \widehat{\mu}) = q(\widehat{\mu}) + \dot{q}(\widehat{\mu})(y - \widehat{\mu}) - q(y) \quad where \quad \dot{q}(\widehat{\mu}) = dq/d\mu|_{\widehat{\mu}}$$

Then by the Optimism Theorem in Section 3, we can have explicit formulas for covariance penalty and prediction error estimations for the q-class error measures.

3 Linear Spline Model

We will fit the given data to a linear spline model. We find the optimal number of knots in the linear spline model by using the formulas for covariance penalty and error estimation mentioned in part 2. To use parametric bootstrap in the process, we assume

$$y_i|x_i \sim N(g(x_i), \sigma^2)$$

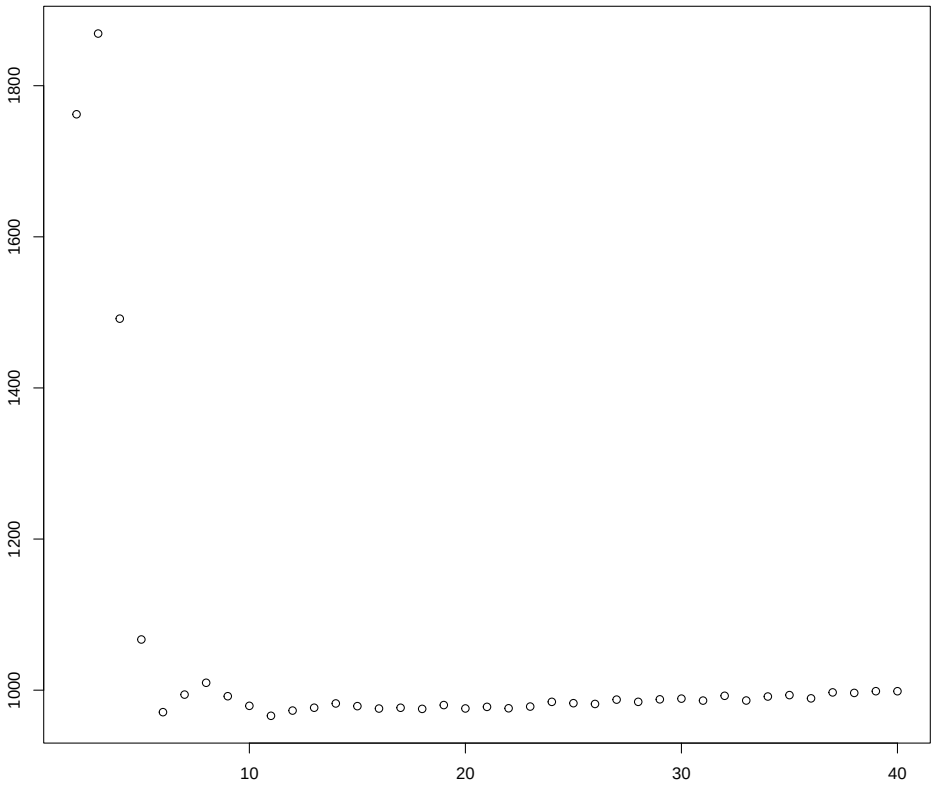
We need to estimate the variance σ^2. We can do this by fitting the data to a linear spline model with number of knots equal to $n/\log n$ and use the residue sum of square of the model to estimate the variance. We can also use sigmahat = 0.986199246346988 from part11 which is the residue sum of square of the regressogram model with $n/\log n$ number of bins. From the data we will generate a large number B of simulated observations y^{*b} for b in 1,..,B. For the bootstrapped observations, we have

$$y^{*b} \sim N(g(x_i), \widehat{\sigma}^2)$$

Then we can find $\widehat{\mu}_i^{*b}$ by fitting the bootstrapped data y^{*b} to the linear spline model. We do this for each candidate number of knots and calculate the covariance penalty and estimated error based on formulas in part 2.

```
> require(lspline)
> require(splines)
> B = 200
> max_knot = 40
> square_error = rep(0, times = max_knot)
> error_estimates = rep(0, times = max_knot)
> for (num_knot in 2:max_knot){
+   lspmodel = lm(yy ~ elspline(xx, num_knot), data = dat)
+   y_hat = fitted(lspmodel)
+   square_error[num_knot] = sum(resid(lspmodel)^2)
+   total_boot = matrix(0,nrow = n, ncol = B)
+   mu_hat_star = matrix(0, nrow = n, ncol = B)
+   cov_hat = rep(0, times = n)
+
+   for(b in 1:B){
```

Figure 3: number of knots vs estimated error




```
+     total_boot[,b] = y_hat + rnorm(n, 0, sigma_hat)
+     bootstrap_dat = data.frame(x = xx, y = total_boot[,b])
+     boot_lsp = lm(y ~ elspline(x, num_knot), data = bootstrap_dat)
+     mu_hat_star[,b] = fitted(boot_lsp)
+   }
+   for(i in 1:n){
+     y_i_star = sum(total_boot[i,])/B
+     for(b in 1:B){
+       cov_hat[i] = cov_hat[i] + mu_hat_star[i,b]*(total_boot[i,b] - y_i_star)/(B - 1
+     }
+   }
+   error_estimates[num_knot] = square_error[num_knot] + 2*sum(cov_hat)
+ }
```

The optimal number of knots is 11

4 Kernel Regression Model

In this section, we fit the data to kernel regression models. We will try to find the optimal bandwidth to for the kernel regression model. Our minimal bandwidth is 0.05 and our maximal bandwidth is 1.5. And we try kernel regression models with 20 different bandwidths in this range.

```
> sorted_dat = dat[order(xx),]
> B = 200
> max_band = 20
> square_error = rep(0, times = max_band)
> error_estimates = rep(0, times = max_band)
> band_width = seq(from = 0.05, to = 1.5, length.out = max_band)
> for (k in 1:max_band){
+   knmodel = ksmooth(sorted_dat$xx, sorted_dat$yy, 'normal', bandwidth = band_width[k
+   y_hat = knmodel$y
+   square_error[k] = sum((y_hat - yy)^2)
+   total_boot = matrix(0,nrow = n, ncol = B)
+   mu_hat_star = matrix(0, nrow = n, ncol = B)
+   cov_hat = rep(0, times = n)
+
+   for(b in 1:B){
+     total_boot[,b] = y_hat + rnorm(n, 0, sigma_hat)
+     bootstrap_dat = data.frame(x = sorted_dat$xx, y = total_boot[,b])
+     boot_kn = ksmooth(bootstrap_dat$x, bootstrap_dat$y, 'normal', bandwidth = band_w
+     mu_hat_star[,b] = boot_kn$y
+   }
+   for(i in 1:n){
+     y_i_star = sum(total_boot[i,])/B
+     for(b in 1:B){
+       cov_hat[i] = cov_hat[i] + mu_hat_star[i,b]*(total_boot[i,b] - y_i_star)/(B - 1
+     }
+   }
+   error_estimates[k] = square_error[k] + 2*sum(cov_hat)
+ }
```

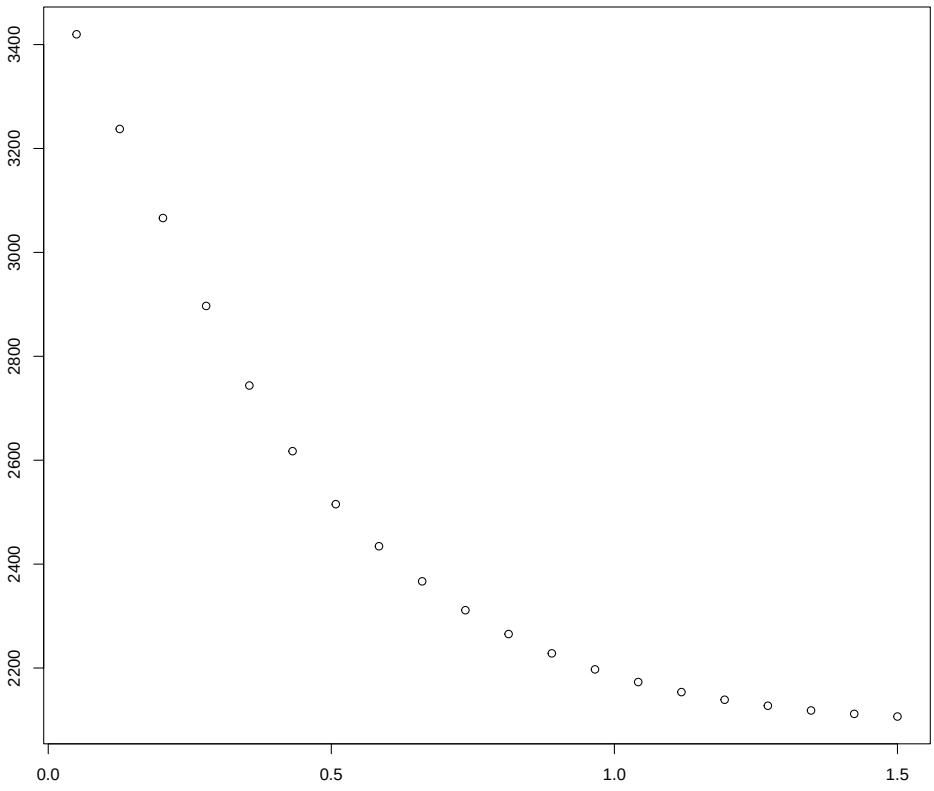
The optimal bandwidth for the kernel regression model is 1.5

5 Lowess Model

We fit the data to a Lowess model and we want to find the optimal span for the lowess model. We use similar methods as in part 3 and part 4.

```
> B = 200
> max_span = 30
```

Figure 4: bandwidth for kernel regression vs estimated error



```
> square_error = rep(0, times = max_span)
> error_estimates = rep(0, times = max_span)
> spans = seq(from = 0.05, to = 1, length.out = max_span)
> for (k in 1:max_span){
+   lowessmodel = loess(yy~xx, dat, span = spans[k])
+   y_hat = lowessmodel$y
+   square_error[k] = sum(residuals(lowessmodel)^2)
+   total_boot = matrix(0,nrow = n, ncol = B)
+   mu_hat_star = matrix(0, nrow = n, ncol = B)
+   cov_hat = rep(0, times = n)
+
+   for(b in 1:B){
+     total_boot[,b] = y_hat + rnorm(n, 0, sigma_hat)
+     bootstrap_dat = data.frame(x = xx, y = total_boot[,b])
+     boot_lowess = loess(y~x, bootstrap_dat,span = spans[k])
+     mu_hat_star[,b] = boot_lowess$y
+   }
+   for(i in 1:n){
+     y_i_star = sum(total_boot[i,])/B
+     for(b in 1:B){
+       cov_hat[i] = cov_hat[i] + mu_hat_star[i,b]*(total_boot[i,b] - y_i_star)/(B - 1
+     }
+   }
+
+   error_estimates[k] = square_error[k] + 2*sum(cov_hat)
+ }
```

We find that the optimal span for Lowess model is 0.05

Figure 5: Span for Lowess model vs estimated error