Homework 7

Advanced Methods for Data Analysis (36-402 A)

Due Friday March 20, 2020 at 3:00 pm

You should **always show all your work**.

- Start each problem **on a new page**.
- Assignments must be submitted through Gradescope as a PDF. Follow the instructions here: `https://www.cmu.edu/teaching/gradescope/`
- Gradescope will ask you to mark which parts of your submission correspond to each homework problem. This is mandatory; if you do not, grading will be slowed down, and your assignment will be penalized.
- Make sure your work is legible in Gradescope. You may not receive credit for work the TAs cannot read. **Note:** If you submit a PDF with pages much larger than 8.5×11", they will be blurry and unreadable in Gradescope.
- For questions involving R code, we strongly recommend using R Markdown. The relevant code should be included with each question, rather than in an appendix.

1. **Optimism and Effective Degrees of Freedom.** In lecture (Lecture 1, Lecture 3, and again in Lecture 6), we presented the following result about the relationship between in-sample prediction risk and expected training error:

   Let the data consist of $(x_1, Y_1), \ldots, (x_n, Y_n)$, a set of n (predictor, response) pairs with

   (i) $Y_i = r(x_i) + \varepsilon_i$ for each i,

   (ii) $\varepsilon_1, \ldots, \varepsilon_n$ uncorrelated random variables with mean 0 and common variance σ^2,

   (iii) $x_1, \ldots, x_n$ all non-random known values, and

   (iv) $r(\cdot)$ a non-random unknown function.

   Let $\hat{r}(\cdot)$ be an estimator of $r(\cdot)$ that is determined from the data. Let Y_i' have the same distribution as Y_i for each i with $Y_1', \ldots, Y_n'$ independent of $Y_1, \ldots, Y_n$. Then

   $$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\{Y_i' - \hat{r}(x_i)\}^2\right] - \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{r}(x_i)\}^2\right] = \frac{2}{n}\sum_{i=1}^{n}\mathrm{Cov}(Y_i, \hat{r}(x_i)). \qquad (1)$$

   Prove that (1) is true. You may want to review some of the earlier lectures where prediction risk was discussed, especially the associated hand-written notes.

2. **Degrees of Freedom for Kernels and Splines.** Once more, use the housing data from Homeworks 2 and 4. Merge the traning and test data sets into a single data set as in problem 2(e) of Homework 4.

   We will be fitting models in which we try to predict median house value from mean household income. We will use both splines and normal-kernel regressions with a variety of tuning parameters while we attempt to use "effective degrees-of-freedom" to make the comparisons on a common scale.

   (a) Plot `Median_house_value` versus `Mean_household_income`.

   (b) First, fit several spline modes with different smoothing parameters. We will use `smooth.spline` with the `df` version of the smoothing parameter. (This is the same as the *effective degrees-of-freedom* that was defined in lecture. R will automatically select the penalty λ to match the number of effective degrees of freedom you select.)

   Use `df` parameter values from 2 to 27 in steps of 1. For each `df`, run five-fold cross-validation on the full data set, computing the estimated MSE of prediction on each held-out fold. Compute the sample average of the five estimated MSE of prediction values for each `df`.

   Also, for each `df`, compute an estimated standard error for the average by computing the sample standard deviation of the five estimated MSE of prediction values divided by $\sqrt{5}$. Plot the sample average of the five estimated MSE of prediction values against `df`, and find the `df` value that provides the lowest average. Compare the differences between the plotted averages to the estimated standard errors of the averages. Say why this comparison either does or does not inspire confidence that we have really found the `df` that provides the best out-of-sample prediction.

   (c) Next, perform a similar analysis for kernel regression with a normal (Gaussian) kernel. Use the `npreg` function in `library(np)` to fit the kernel regressions.

   This time, there is no `df` option for the smoothing parameter. Instead, use the `bws` parameter to set the bandwidth to each of the values from 4000 to 7000 in steps of 300. (That should be 11 different values, for a sanity check.) **Make sure that you use the same five folds for five-fold cross-validation as you used in part (b). The comparisons will not be fair if you use different folds.** In the instructions from part (b), replace `df` by `bws` and repeat all of the analyses for the kernel regression models. (This calculation is time-consuming.)

   (d) For each bandwidth h, the effective degrees-of-freedom for a kernel regression is

$$\mathrm{df}(h) = K(0) \sum_{i=1}^{n} \frac{1}{\sum_{j=1}^{n} K([x_i - x_j]/h)}.$$

   (This calculation is time-consuming.)

   For each bandwidth (`bws`) in part (c), compute the corresponding effective degrees-of-freedom. Draw a single plot that contains the pairs

   (effective degrees-of-freedom, estimated MSE of prediction)

   for both the spline fits and the kernel fits, labeled well enough to be able to tell which are which. Comparing the differences between the estimated MSE's for

splines and kernels to the estimated standard errors, say why this comparison does or does not inspire confidence that we can tell which method provides better out-of-sample prediction.

(e) Finally redraw the plot from part (a) and add

(i) a line for the spline fit corresponding to the best smoothing parameter value, and

(ii) a line for the kernel regression fit with the best smothing parameter value.

For the two lines, use a common set of 201 predictor values that is equally-spaced over the observed range of `Mean_household_income` and extending 5% past each end. Comment on how well or badly each of the two fitted curves seems to fit the data. Also offer a reason for why the two fitted curves behave so differently at and beyond the extremes of the predictor.

1. **Optimism and Effective Degrees of Freedom.** In lecture (Lecture 1, Lecture 3, and again in Lecture 6), we presented the following result about the relationship between in-sample prediction risk and expected training error:

   Let the data consist of $(x_1, Y_1), \ldots, (x_n, Y_n)$, a set of n (predictor, response) pairs with

   (i) $Y_i = r(x_i) + \varepsilon_i$ for each i,

   (ii) $\varepsilon_1, \ldots, \varepsilon_n$ uncorrelated random variables with mean 0 and common variance σ^2,

   (iii) $x_1, \ldots, x_n$ all non-random known values, and

   (iv) $r(\cdot)$ a non-random unknown function.

   Let $\hat{r}(\cdot)$ be an estimator of $r(\cdot)$ that is determined from the data. Let Y_i' have the same distribution as Y_i for each i with $Y_1', \ldots, Y_n'$ independent of $Y_1, \ldots, Y_n$. Then

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\{Y_i' - \hat{r}(x_i)\}^2\right] - \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{r}(x_i)\}^2\right] = \frac{2}{n}\sum_{i=1}^{n}\mathrm{Cov}(Y_i, \hat{r}(x_i)). \qquad (1)$$

   Prove that (1) is true. You may want to review some of the earlier lectures where prediction risk was discussed, especially the associated hand-written notes.

Let $\bar{r}(x_i) = E(\hat{r}(x_i))$, $Y_i' = r(x_i) + \varepsilon_i'$

$$E\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \hat{r}(x_i))^2\right] - E\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{r}(x_i))^2\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} E\left[(Y_i' - \hat{r}(x_i))^2 - (Y_i - \hat{r}(x_i))^2\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} E\left[(Y_i' - r(x_i) + r(x_i) - \bar{r}(x_i) + \bar{r}(x_i) - \hat{r}(x_i))^2 - (Y_i - r(x_i) + r(x_i) - \bar{r}(x_i) + \bar{r}(x_i) - \hat{r}(x_i))^2\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} E\left[(\varepsilon_i' + r(x_i) - \bar{r}(x_i) + \bar{r}(x_i) - \hat{r}(x_i))^2 - (\varepsilon_i + r(x_i) - \bar{r}(x_i) + \bar{r}(x_i) - \hat{r}(x_i))^2\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} E\left[\varepsilon_i'^2 + 2(r(x_i) - \bar{r}(x_i))\varepsilon_i' + 2(\bar{r}(x_i) - \hat{r}(x_i))\varepsilon_i' + (r(x_i) - \bar{r}(x_i))^2 + 2(r(x_i) - \bar{r}(x_i))(\bar{r}(x_i) - \hat{r}(x_i)) + (\bar{r}(x_i) - \hat{r}(x_i))^2 - \varepsilon_i^2 - 2(r(x_i) - \bar{r}(x_i))\varepsilon_i - 2(\bar{r}(x_i) - \hat{r}(x_i))\varepsilon_i - (r(x_i) - \bar{r}(x_i))^2 - 2(r(x_i) - \bar{r}(x_i))(\bar{r}(x_i) - \hat{r}(x_i)) - (\bar{r}(x_i) - \hat{r}(x_i))^2\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n} E[\varepsilon_i'^2] + 2(r(x_i) - \bar{r}(x_i))E(\varepsilon_i') + 2E[(\bar{r}(x_i) - \hat{r}(x_i))\varepsilon_i'] - E[\varepsilon_i^2] - 2(r(x_i) - \bar{r}(x_i))E(\varepsilon_i) - 2E[(\bar{r}(x_i) - \hat{r}(x_i))\varepsilon_i]$$

($E[\varepsilon_i'^2] = \sigma^2$, $E[\varepsilon_i^2] = \sigma^2$, $E(\varepsilon_i') = 0$, $E(\varepsilon_i) = 0$)

$$= \frac{2}{n}\sum_{i=1}^{n} E\left[(Y_i' - r(x_i))(\bar{r}(x_i) - \hat{r}(x_i)) - (Y_i - r(x_i))(\bar{r}(x_i) - \hat{r}(x_i))\right]$$

$$= \frac{2}{n}\sum_{i=1}^{n} E[Y_i' - r(x_i)]E[\bar{r}(x_i) - \hat{r}(x_i)] + E\left[(Y_i - E[Y_i])(\hat{r}(x_i) - \bar{r}(x_i))\right]$$

since $\hat{r}$ is estimated using (x_i, Y_i)s and they are independent of (x_i, Y_i')s and $Y_i' - r(x_i) = \varepsilon_i'$ and has mean 0 so $Y_i' - r(x_i)$ and $\bar{r}(x_i) - \hat{r}(x_i)$ aren't correlated ($E[Y_i' - r(x_i)] = 0$)

$$= \frac{2}{n}\sum_{i=1}^{n} \mathrm{Cov}(Y_i, \hat{r}(x_i))$$

402 HW7

Alvin Pan

Name: Qingkai Pan Andrew ID: qpan
Collaborated with:

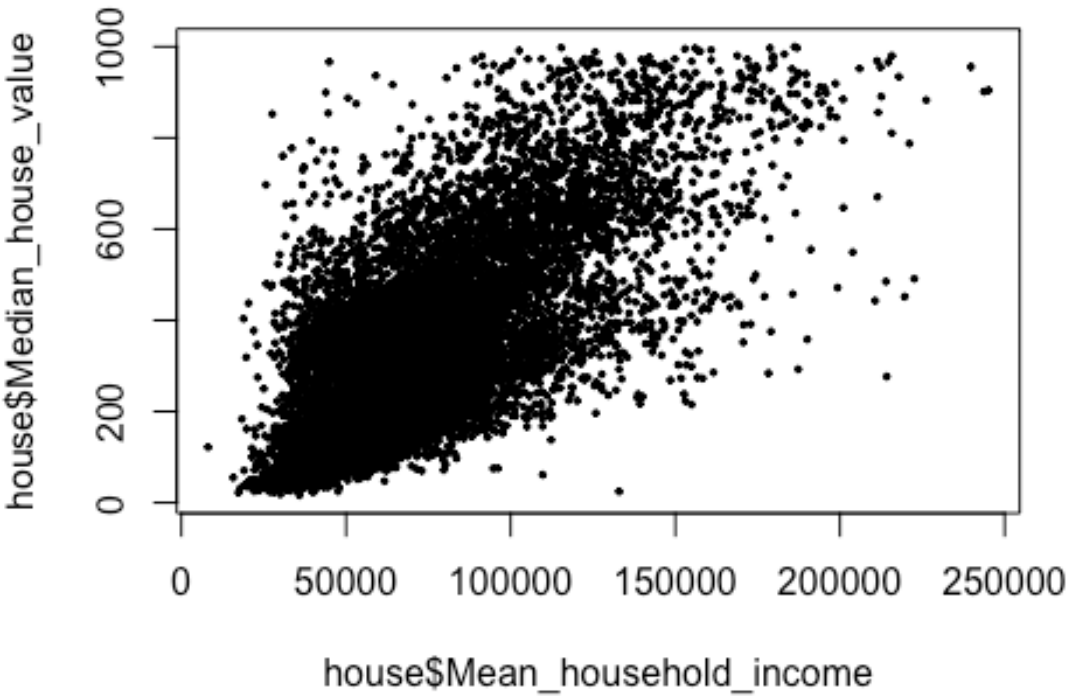
```
library(np)

## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-9)
## [vignette("np_faq",package="np") provides answers to frequently asked
questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

Q2

```
house.train = read.csv("housetrain.csv", header = TRUE)
house.test = read.csv("housetest.csv", header = TRUE)

house <- rbind(house.train, house.test)
```

(a)

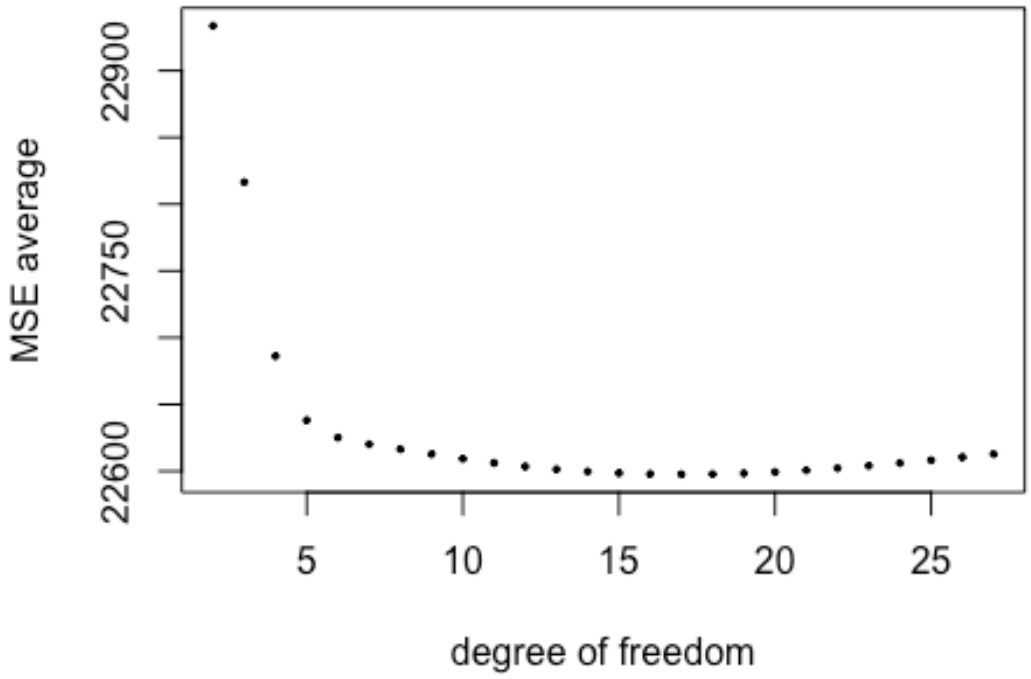
```
plot(house$Mean_household_income, house$Median_house_value, pch = 20, cex =
0.5)
```



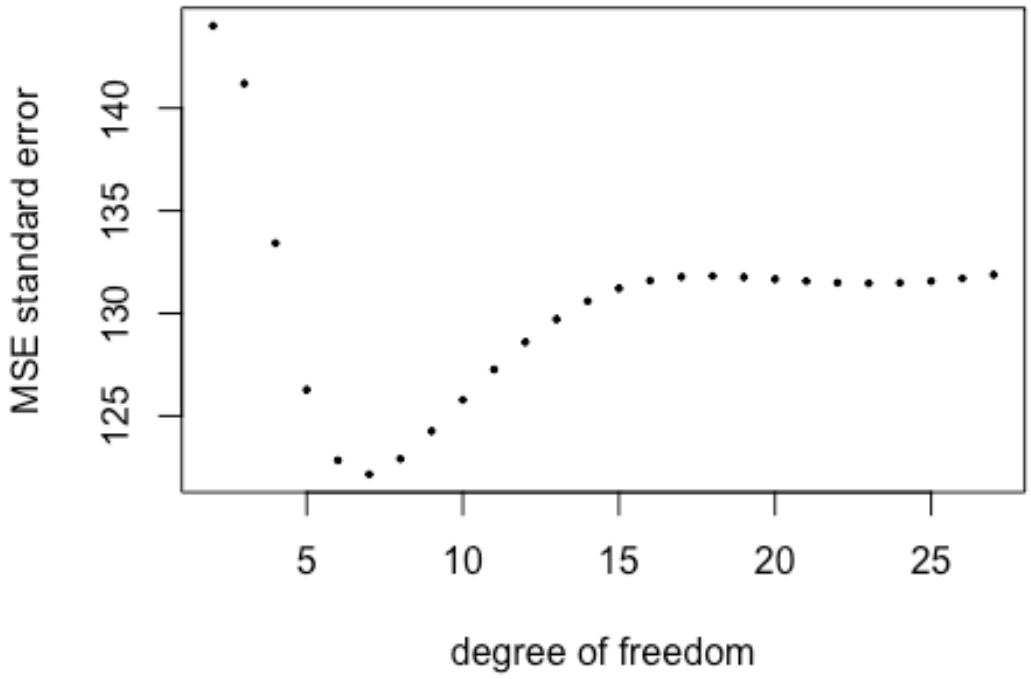
(b)

```r
get.pred.error <- function(model, test) {
  test.pred = predict(model, newdata = test)
  return (mean((test.pred-test$Median_house_value)^2))
}

err.mat = matrix(rep(0, 5*26), ncol = 5)

set.seed(0)
samp <- sample(rep(1:5, 2121), replace = FALSE)

dfs = 2:27

for (n in 2:27) {
  #samp <- sample(rep(1:5, 2121), replace = FALSE)
  for (k in 1:5) {
    testd <- house[samp == k, ]
    traind <- house[!(samp == k), ]

    train.rows = nrow(traind)

    sm =
```

```
smooth.spline(traind$Median_house_value~traind$Mean_household_income, df=n)
    test.pred = predict(sm, x = testd$Mean_household_income)$y
    err.mat[n-1,k] = mean((test.pred-testd$Median_house_value)^2)
  }
}

df.err.means = apply(err.mat, 1, mean)

df.err.ses = apply(err.mat, 1, sd)/sqrt(5)

plot(2:27, df.err.means, xlab = "degree of freedom", ylab = "MSE average",
pch = 20, cex = 0.5)
```




```
best.df = dfs[which.min(df.err.means)]
best.df

## [1] 17

plot(2:27, df.err.ses, xlab = "degree of freedom", ylab = "MSE standard
error", pch = 20, cex = 0.5)
```

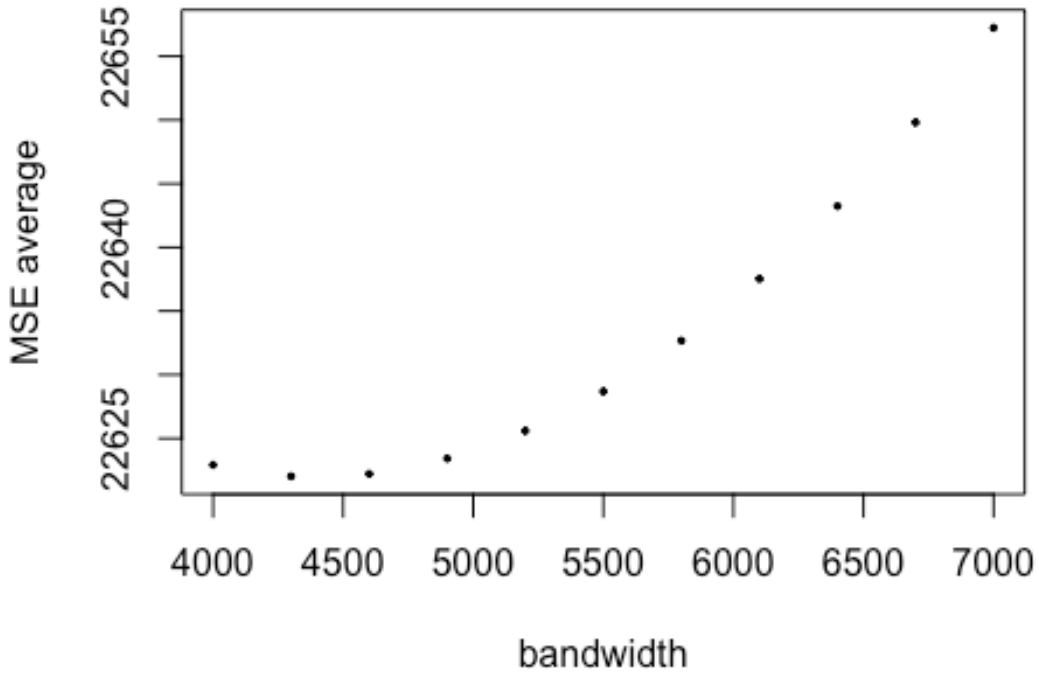


The two plots shows different minimums for the average of the MSEs and standard deviation of the MSEs. But since the MSE deviation is all between 120 and 135 for degree of freedom greater or equal to 5, which means the variation of the standard deviation is not high. So it does inspire that we do have found the ideal df.

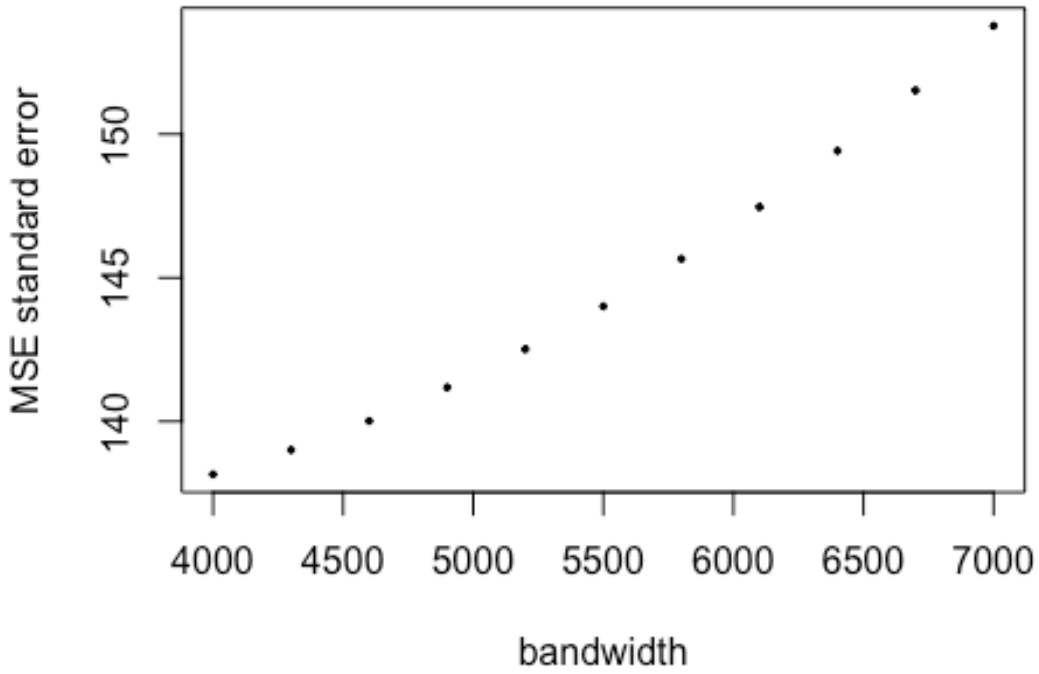
(c)

```
gaus.err.mat = matrix(rep(0, 5*11), ncol = 5)

bws1 = seq(4000, 7000, by=300)

for (i in 1:length(bws1)) {
  for (k in 1:5) {
    testd <- house[samp == k, ]
    traind <- house[!(samp == k), ]

    train.rows = nrow(traind)


    gm = npreg(Median_house_value ~ Mean_household_income,
               data = traind, bws = bws1[i])
    gaus.err.mat[i,k] = get.pred.error(gm, testd)
```

```
  }
}

gaus.err.means = apply(gaus.err.mat, 1, mean)

gaus.err.ses = apply(gaus.err.mat, 1, sd)/sqrt(5)

plot(bws1, gaus.err.means, xlab = "bandwidth", ylab = "MSE average", pch =
20, cex = 0.5)
```




```
best.bw = bws1[which.min(gaus.err.means)]
best.bw

## [1] 4300

plot(bws1, gaus.err.ses, xlab = "bandwidth", ylab = "MSE standard error", pch
= 20, cex = 0.5)
```



The two plots shows different minimums for the average of the MSEs and standard deviation of the MSEs and both are monotonically increasing(avergae MSE monotonically increases for bandwidths >= 4300). But since the MSE standard deviation for the bandwidth with lowest average MSE is pretty close the bandwidth with the lowest standard deviation in terms of vertical distance, it does inspire that we do have found the ideal df.

(d)

```
calc.df <- function(x, bw) {
  n = length(x)
  res = 0
  for (i in 1:n) {
    dinom = 0
    for (j in 1:n) {
      dinom = dinom + dnorm((x[i]-x[j])/bw)
    }
    res = res + 1/dinom
  }
  return (dnorm(0) * res)
}

eff.dfs = rep(0, length(bws1))
```

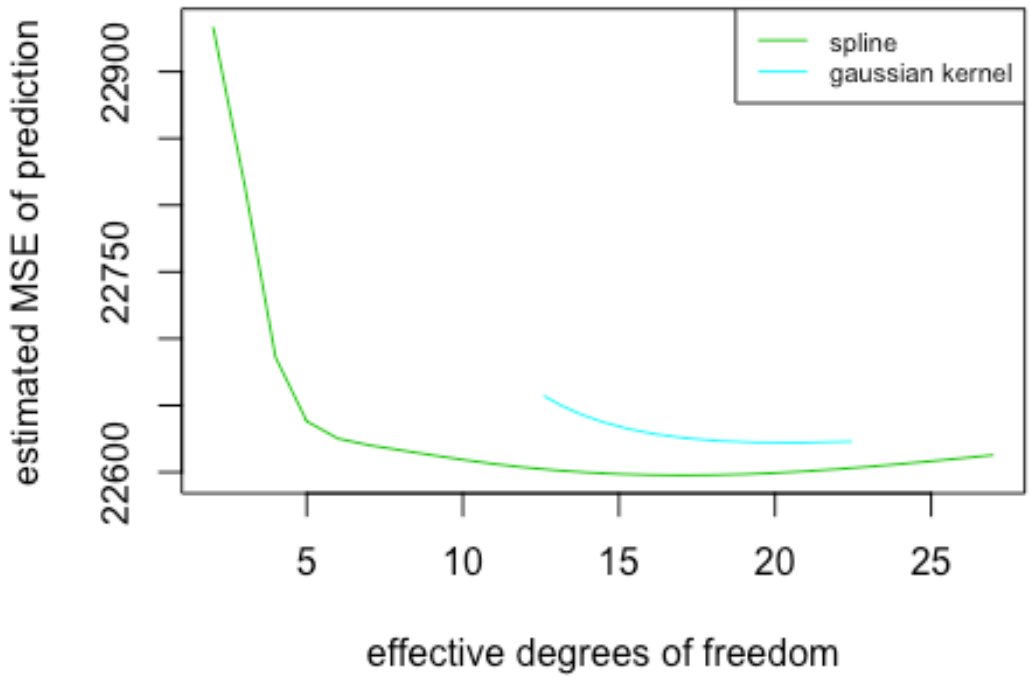
```
for (i in 1:length(bws1)) {
  eff.dfs[i] = calc.df(house$Mean_household_income, bws1[i])
}
```

Since it takes a long time, just copy down the result since rmarkdown takes ages.

```
eff.dfs = c(22.46228, 20.83822, 19.42009, 18.17743, 17.08388, 16.11645,
15.25544, 14.48435, 13.78948, 13.15955, 12.58522)
eff.dfs
```

```
##  [1] 22.46228 20.83822 19.42009 18.17743 17.08388 16.11645 15.25544
##  [8] 14.48435 13.78948 13.15955 12.58522
```

```
plot(dfs, df.err.means, xlab = "effective degrees of freedom", ylab =
"estimated MSE of prediction", pch = 20, cex = 0.5, col = 3, type = 'l')

lines(eff.dfs, gaus.err.means, pch = 20, cex = 0.5, col = 5)

legend("topright", legend=c("spline", "gaussian kernel"), col=c(3,5), lty=1,
cex=0.7)
```

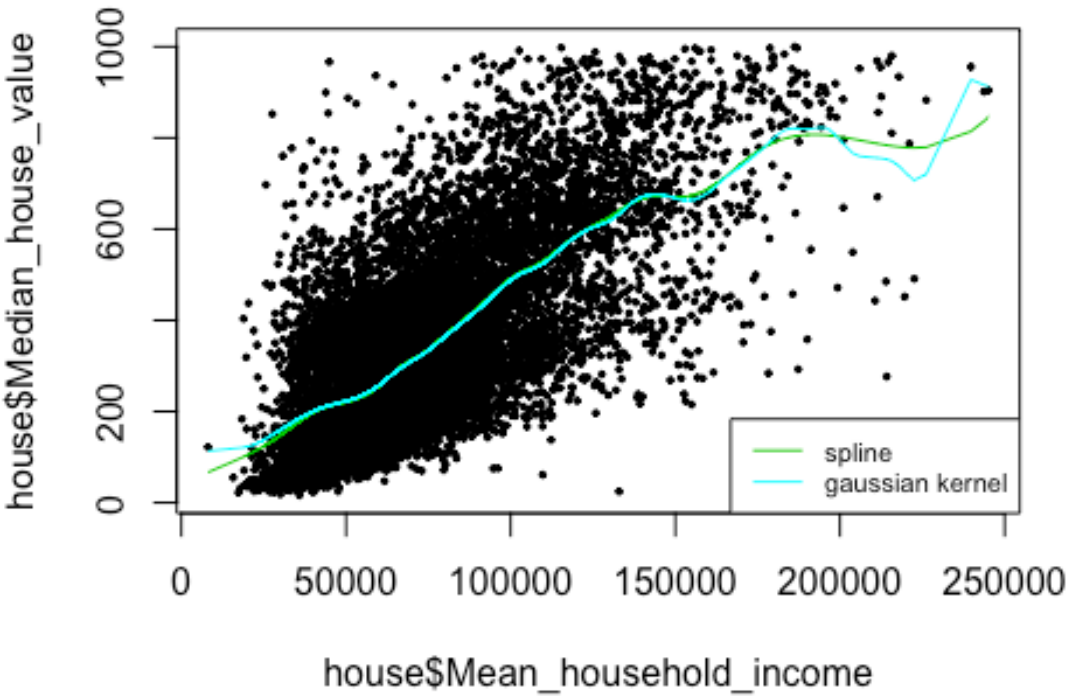


The MSEs for spline are lower than gaussian kernel between df = 10 and 25 The comparison doesn't seem to be reasonable since the range and number of bandwidths for the two

methods are different. Spline has more than double the number of degrees of freedom analyzed than gaussian kernel, so the conclusion would be not comprehensive.

(e)

```
plot(house$Mean_household_income, house$Median_house_value, pch = 20, cex =
0.5)

best.sm = smooth.spline(house$Median_house_value~house$Mean_household_income,
df=best.df)
best.sm.pred = predict(best.sm, x = house$Mean_household_income)$y

best.gm = npreg(Median_house_value ~ Mean_household_income,
                data = house, bws = best.bw)

best.gm.pred = predict(best.gm, newdata = house)

d <- data.frame(cbind(Mean_household_income = house$Mean_household_income,
best.sm.pred, best.gm.pred))

d <- d[order(d$Mean_household_income), ]

with(d, lines(Mean_household_income, best.sm.pred, xlab = "effective degrees
of freedom", ylab = "estimated MSE of prediction", pch = 20, cex = 0.5, col =
3))


with(d, lines(Mean_household_income, best.gm.pred, pch = 20, cex = 0.5, col =
5))

legend("bottomright", legend=c("spline", "gaussian kernel"), col=c(3,5),
lty=1, cex=0.7)
```



Gaussian kernel fit seems to be more influenced by the outlier at the 2 end of the plot and the rest of the plot is almost identical, and gaussian kernel in particular shows higher variance. This is because splines are restricted to have less degree of freedom at the extremes, enable it to not over-influenced by the outliers.