Problem Set1

Yile Chen

1

Regarding model complexity, the model illustrated by the upper row of plots (model1) is a simple linear model, with low complexity and flexibility; whereas the model used in the lower row (model2) has much higher complexity and flexibility. Comparing the plots column-wisely, we can see that the complex model2 fit the data much better than the simple model1, suggesting that model2 has low bias and model1 has high bias. In terms of variance, model2 tends to overfit, leading to high variance (when applied to new datasets, the model may not fit them as good as it fits the training dataset). For model1, it is likely that it tends to have relatively low variance. Though it produces high bias for each dataset, the error rates are generally more consistent than the more complex model.

2

Training error(downwards): the graph shows that in general, training error decreases when the model becomes more flexible (k approaches to 1). It is quite apparent because flexible models can take into account more localized details of the data points, and are tuned to fit the data set better.

Testing error(U-shaped): when flexibility increases, test error first decreases, but after reaches a point (around k=10), it levels up again. This is because, at first, while adding more complexity in the model, we tend to capture the relationship between input and output better and can thus make better predictions (similar to what have happened to the training data). However, when flexibility increases to a certain extent, the model would face the problem of overfitting. That is, the model adapts too well on the training set – it tends to capture random(noise) components – that when applied to new dataset, it would perform worse than to the original training set.

3

In such cases, it is likely that the performance of a flexible model is better than an inflexible method. Because a flexible model will fit the dataset closer and, by leveraging the large sample size, it would capture the relationship between X and Y more accurately (much lower bias) without sacrificing variance too much.

4

In such cases, it is likely that the performance of a flexible model is worse than an inflexible method. Because a flexible model will have the large number of parameters tuned to closely fit the small dataset. This can easily lead to overfitting and result in high variance, so that a simpler model with much lower virance could perform better.

5

In such cases, the performance of a flexible model would be better than an inflexible method. Since the relationship between X and Y is highly non-linear, a simple model would not able to accurately capture such complex relationship. A more flexible model would outperform a simpler one by signinficantly lowering bias.

6

While minimizing the training MSE, we are reducing training error and tuning the model to better fit the training data. However, if the model works too closely with the training data (fits it too well), it may capture too much elements, including noises, in it. Such detailed patterns may only exist in the training dataset, so that when the model is applied to new data, it could perform worse In other words, such model would have low bias but high virance, which suggest overfitting.

7

```
knitr::opts_chunk$set(echo = TRUE)
library(ISLR)
library(boot)
library(here)
```

```
## here() starts at /Users/miao/Desktop
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
anes <- read.csv(here("data", "anes_pilot_2016.csv"))
```

```
anes <- anes %>%
  mutate(fttrump = replace(fttrump, fttrump > 100 | fttrump < 0, NA),
         ftobama = replace(ftobama, ftobama > 100 | ftobama < 0, NA))%>%
  na.omit()
sum(is.na(anes))
```

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

```
NESdta_short <- anes %>%
  select(fttrump, ftobama)

skimr::skim(NESdta_short)
```

Table 1: Data summary

Name	NESdta_short
Number of rows	88
Number of columns	2
Column type frequency:	
numeric	2
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
fttrump	0	1	44.68	36.01	0	5.75	50.0	75.75	100	
ftobama	0	1	43.76	38.78	0	3.00	39.5	81.75	100	

```
median(NESdta_short$ftobama, na.rm = TRUE)
```

```
## [1] 39.5
```

The following code compute two bootstrapped samples of Obama.

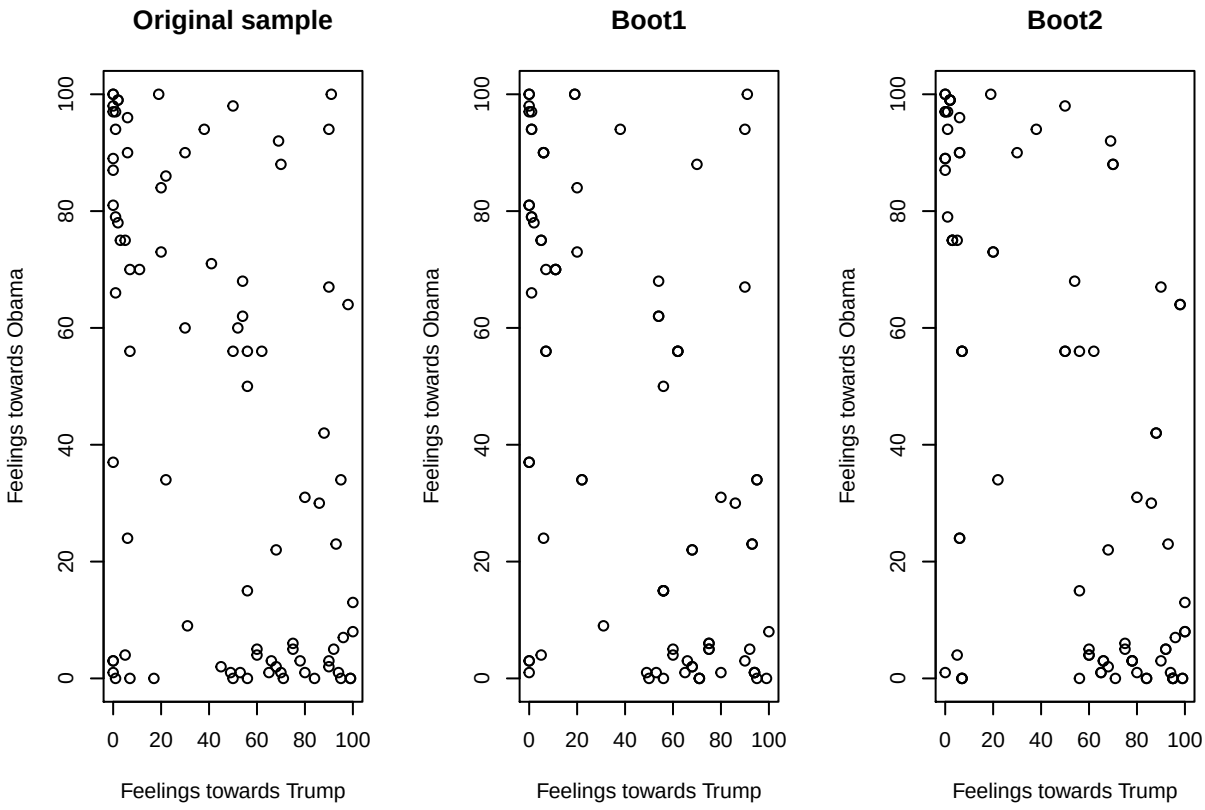
```
set.seed(12345)
#bootstrapped sample for ftobama
boot_obama1 <- sample(NESdta_short$ftobama, replace = TRUE)
#take a look at the sample
boot_obama2 <- sample(NESdta_short$ftobama, replace = TRUE)
```

The following code compute two bootstrapped samples of Trump.

```
set.seed(12345)
#Bootstrapped sample1 for fttrump
boot_trump1 <- sample(NESdta_short$fttrump, replace = TRUE)
#Bootstrapped sample2 for fttrump
boot_trump2 <- sample(NESdta_short$fttrump, replace = TRUE)
```

plot feelings toward trump (x-axis) versus feelings toward obama (y axis) for both the original sample and the boot strapped samples.

```
par(mfrow = c(1,3))
plot(NESdta_short$fttrump, NESdta_short$ftobama, main = "Original sample",
     xlab = "Feelings towards Trump", ylab = "Feelings towards Obama")
plot(boot_trump1, boot_obama1,  main = "Boot1",
     xlab = "Feelings towards Trump", ylab = "Feelings towards Obama")
plot(boot_trump2, boot_obama2,  main = "Boot2",
     xlab = "Feelings towards Trump", ylab = "Feelings towards Obama")
```



8

In general, the bootstrapped plots look similar to each other and are also similar to that of the original sample (Data points concentrate in the upper left and lower right corner.). I think the similarity can be explained by the fact that all bootstrapped samples are generated from their original sample, so that it is not surprising that they would somehow resemble it. As for the minor differences, because we sampled with replacement (one obeservation has the chance to appear in a bootstrapped sample multiple times), the bootstrapped samples are not simple copies(or simple subsets) of the original datasets but are different from them; and because of the randomness of the draws, the resamples are also not exactly similar to each other.

9

```r
set.seed(12345)
#create a set of median values of ftobama by bootstrapping 1000 times from the original data
median_boot_obama <- replicate(1000, median(sample(NESdta_short$ftobama, replace = TRUE), na.rm = TRUE))
#take a look
skimr::skim(median_boot_obama)
```

Table 3: Data summary

Name	median_boot_obama
Number of rows	1000
Number of columns	1
Column type frequency:	
numeric	1

Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
data	0	1	40.96	13.69	5	32	38.75	56	75	

```
#standard error of the mediann values
se_median_boot_obama <- sd(median_boot_obama)
se_median_boot_obama
```

```
## [1] 13.69111
```

```
#Confidence interval of the median values
quantile(median_boot_obama, prob = 0.025)
```

```
## 2.5%
##   13
```

```
quantile(median_boot_obama, prob = 0.975)
```

```
##  97.5%
## 62.025
```

```
#the median value of the original sample
median(NESdta_short$ftobama, na.rm = TRUE)
```

```
## [1] 39.5
```

From the above results, we can see that the standard error of the bootstrapped median values is 13.69, and the 95% confidence interval is (13, 62.025). The raletively large standard error and wide confidence interval sugggest that the bootstrapped median values of ftrump distribute quite loosely around the mean. Bootstrapping, in this case, helps to quantify uncertainty associated with the statistic of interest (the median value) and this uncertainty is shown by the standard error and confidence interval.

10

Similarities:

Both cross validation and bootstrapping are resampling methods, which are essential to test and evaluate statistical models. They both repeatedly draw samples from a data set and refit a model on each sample.

Differences:

The two method are different in terms of sampling method and the size of resampling dataset (compared to the original one) – Cross validation uses random sampling without replacement to split the available dataset into smaller sets, whereas the Bootstrapping method uses the original dataset to create multiple datasets of the same size by resampling with replacement.

Why any of this matter?

For social science, researchers may often encounter unsatisfactory samples (e.g.,small samples) which alone may not be able to produce robust results. Resampling methods could ease the problem by producing 'more data'. To be more specific, bootstrapping may fit such circumstances better, since it can help to quantify uncertainty by sampling without replacement and construct a large number of bootstrapped datasets, which are of the same size as the original sample. Similarly, for computational modelling, both resampling method, by creating more training sets and testing sets, can help to validate models for more accurate prediction. Additionally, a more robust modelling process could also facilitate social science inferences.