Analysis 4: Model Selection Via K-Fold CV

Sunghwu Song

May 04, 2021

Instructions

Overview: For each question, show your R code that you used to answer each question in the provided chunks. When a written response is required, be sure to answer the entire question in complete sentences outside the code chunks. When figures are required, be sure to follow all requirements to receive full credit. Point values are assigned for every part of this analysis.

Helpful: Make sure you knit the document as you go through the assignment. Check all your results in the created PDF file.

Submission: Submit via an electronic document on Gradescope. Must be submitted as an PDF file generated in RStudio.

Introduction

The rivers of the world are home to numerous fish species whose existence is dependent on the temperature of the water. Specifically for salmonid, read this article by Katharine Carter, an environmental scientist and lover of fish from sunny California. Salmonid varieties all thrive under different temperature ranges and issues arise when river temperatures are outside these ranges.

> It's a cool place and they say it gets colder. You're bundled up now wait 'til you get older.

As we have been notified, it's getting hot in here. Global warming is happening, and these fish are getting heatstroke. You can take my snow, but hands off my fish. I need that high quality protein and omega-3 fatty acids for mad gains. Because of these "fake" facts, protectors of the water have become interested in developing predictive models for maximum water temperature.

> But the meteor men beg to differ. Judging by the hole in the satellite picture.

Below is a preview of a dataset containing close to a full year of daily observed maximum air and maximum water temperatures for 31 different rivers in Spain. The variable `D` represents the Julian day and takes values from *1* to *365*. The variable `L` identifies 31 different measurement station locations. Variables `W` and `A` are the maximum water and air temperatures, respectively. Finally, the variable named `T` for time maintains the order of the data according to when it was observed. For the sake of our sanity, all days missing important information have been removed using `na.omit()`.

```
DATA=na.omit(read_csv("AirWaterTemp.csv"))
glimpse(DATA)
```

```
## Rows: 9,857
## Columns: 5
## $ D <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2~
## $ L <dbl> 103, 103, 103, 103, 103, 103, 103, 103, 103, 103, 103, 103, 103, 103~
## $ W <dbl> 14.2, 14.4, 14.4, 10.9, 10.8, 10.7, 10.3, 10.1, 9.8, 10.5, 10.5, 9.9~
## $ A <dbl> 21.2, 16.8, 15.4, 10.8, 11.7, 12.4, 13.2, 11.8, 5.1, 3.5, 5.3, 6.2, ~
## $ T <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2~
```

> The ice we skate is getting pretty thin. The water's getting warm so you might as well swim.

What is the point of stealing your tuition, if I cannot grab some fish? The model below is an expression of a family of polynomial regression models that use `A` and `D` to explain the variation in `W`. Every choice of I and J leads to a different model. For our purpose, we would like to consider all possible subset models where $I \leq 7$ and $J \leq 7$. To choose the best model, we rely on cross-validation (CV) to estimate out-of-sample root mean squared error (RMSE). The best choice of I and J minimize prediction error. I highly recommend seeing the previous tutorials which provides necessary code to complete this assignment. If you don't look at this tutorial, you will be miserable like these fish.

$$W = a + \sum_{i=1}^{I} b_i A^i + \sum_{j=1}^{J} c_j D^j + \epsilon$$

> My world's on fire. How about yours? That's the way I like it and I'll never get bored.

Where every you see `COMPLETE`, there is code required for you to write. Lines of code where you see `#DO NOT CHANGE` are meant to not be touched. The output that results from these lines is what will be graded. As you go through the assignment, knit the document to PDF. Make sure you change `eval=F` to `eval=T`. Check your final PDF document before you submit.

Assignment

Part 1: Cross-Validated RMSE for Each Choice of I and J

Q1 *(2 Points)*

In a previous Tutorial, I demonstrated how to use 10-Fold CV to obtain an out-of-sample RMSE when $I = 4$ and $J = 3$. In this assignment, we will also use 10 folds. Using `crossv_kfold()`, create a new dataframe called `DATA2` that contains two new variables `train` and `test` that are list-columns.

Code and Output:

```
set.seed(216) #DO NOT CHANGE

DATA2 <- DATA %>%
  na.omit() %>%
  crossv_kfold(10)

head(DATA2) #DO NOT CHANGE
```

```
## # A tibble: 6 x 3
##   train                 test                .id
```

```
##   <named list>           <named list>         <chr>
## 1 <resample [8,871 x 5]> <resample [986 x 5]> 01
## 2 <resample [8,871 x 5]> <resample [986 x 5]> 02
## 3 <resample [8,871 x 5]> <resample [986 x 5]> 03
## 4 <resample [8,871 x 5]> <resample [986 x 5]> 04
## 5 <resample [8,871 x 5]> <resample [986 x 5]> 05
## 6 <resample [8,871 x 5]> <resample [986 x 5]> 06
```

Q2 *(3 Points)*

Create a function called `RMSE.func()` that takes two vector arguments called *actual* and *predict* representing actual responses in the data and predicted responses from a model, respectively. This function should output the RMSE, which measures the overall error between actual and predicted values.

Code and Output:

```
x=c(1,3,4) #DO NOT CHANGE
y=c(0,0,0) #DO NOT CHANGE

RMSE.func <- function(actual, predict){
  return(sqrt(mean((actual-predict)^2)))
}

RMSE.func(actual=x,predict=y) #DO NOT CHANGE
```

```
## [1] 2.94392
```

Q3 *(10 Points)*

For a specific I and J, the following function fits the desired polynomial model to a given set of data. This function can be utilized to fit polynomial regression models of varying degrees.

```
train.model.func=function(data,I,J){
  mod=lm(W~poly(A,I)+poly(D,J),data=data)
  return(mod)
}
```

In the code chunk below, we begin by initiating an empty 7×7 matrix of missing values called `OUT.RMSE`. Each row corresponds to a different choice of I and each column corresponds to a different choice of J. Apply the code in the previous Tutorial in a double loop that performs 10-Fold CV to estimate out-of-sample RMSE under each polynomial model where $I \in \{1, 2, 3, \cdots, 7\}$ and $J \in \{1, 2, 3, \cdots, 7\}$ and then saves the RMSE in the (I, J)-cell of the matrix `OUT.RMSE`.

Code and Output:

```
OUT.RMSE=matrix(NA,7,7) #DO NOT CHANGE

max_i = nrow(OUT.RMSE)
max_j = ncol(OUT.RMSE)
for(i in 1:max_i){
  for(j in 1:max_j){
    DATA4=DATA2 %>%
```

```
      mutate(train2.model=map(train,train.model.func,i,j))
    DATA5 = DATA4 %>%
      mutate(predict=map2(test,train2.model,~augment(.y,newdata=.x))) %>%
      select(predict) %>%
      unnest(cols=c(predict))
    OUT.RMSE[i,j] = RMSE.func(actual=DATA5$W,predict=DATA5$.fitted)
  }
}

print(OUT.RMSE) #DO NOT CHANGE
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 3.085816 2.804976 2.750931 2.724427 2.720082 2.719978 2.719441
## [2,] 3.082147 2.766886 2.709600 2.695842 2.693852 2.691924 2.692129
## [3,] 3.071947 2.767048 2.709689 2.695323 2.693120 2.690916 2.691105
## [4,] 3.071028 2.766809 2.709543 2.695357 2.693194 2.690943 2.691113
## [5,] 3.069793 2.767196 2.709861 2.695607 2.693466 2.691157 2.691321
## [6,] 3.069761 2.767359 2.710100 2.695857 2.693761 2.691450 2.691612
## [7,] 3.070092 2.767982 2.710564 2.696379 2.694291 2.692010 2.692154
```

Part 2: Comparing Top 5 Models

In the code chunk below, we start by making the information found the summarized information in `OUT.RMSE` tidy. There are three columns in `OUT.RMSE2` that links the cross-validated RMSE to all considered combinations of I and J. Change `eval=F` to `eval=T` before knitting to HTML.

```
OUT.RMSE2=as.tibble(OUT.RMSE) %>%
  mutate(I=1:7) %>%
  rename(`1`=V1,`2`=V2,`3`=V3,`4`=V4,`5`=V5,`6`=V6,`7`=V7) %>%
  select(I,everything()) %>%
  gather(`1`:`7`,key="J",value="RMSE",convert=T) %>%
  mutate(I=as.factor(I),J=as.factor(J))
```

```
## Warning: ‘as.tibble()‘ was deprecated in tibble 2.0.0.
## Please use ‘as_tibble()‘ instead.
## The signature and semantics have changed, see ‘?as_tibble‘.
```

```
## Warning: The ‘x‘ argument of ‘as_tibble.matrix()‘ must have unique column names if ‘.name_repair‘ is
## Using compatibility ‘.name_repair‘.
```

```
head(OUT.RMSE2)
```

```
## # A tibble: 6 x 3
##   I     J      RMSE
##   <fct> <fct> <dbl>
## 1 1     1      3.09
## 2 2     1      3.08
## 3 3     1      3.07
## 4 4     1      3.07
## 5 5     1      3.07
## 6 6     1      3.07
```

Q1 ***(2 Points)***

Create a tibble called `BEST5.RMSE` which contains the rows in `OUT.RMSE2` corresponding to the best five models according to the RMSE and sorted from best to worst.

Code and Output:

```
temp <- arrange(OUT.RMSE2, RMSE)
BEST5.RMSE <- temp[1:5,]

head(BEST5.RMSE) #DO NOT CHANGE
```

```
## # A tibble: 5 x 3
##   I     J      RMSE
##   <fct> <fct> <dbl>
## 1 3     6      2.69
## 2 4     6      2.69
## 3 3     7      2.69
## 4 4     7      2.69
## 5 5     6      2.69
```

Q2 ***(3 Points)***

Now, observe the figure below that shows the change in maximum water temperature (W) across Julian days (D). Change `eval=F` to `eval=T` before knitting to HTML.

```
ggplot(DATA) +
  geom_point(aes(x=D,y=W),alpha=0.05,stroke=0) +
  theme_minimal() +
  xlab("Julian Day")+
  ylab("Max Water Temperature")
```

Using `mutate()`, we create a tibble `BEST5.DATA` based off `DATA`. The new object `BEST5.DATA` contains 5 columns of predictions under the top 5 models based on the values of `I` and `J` in `BEST5.RMSE`. The five columns of predictions should be given names `First`, `Second`, `Third`, `Fourth`,`Fifth`, in order from best to worst. Change `eval=F` to `eval=T` before knitting to HTML.

```
BEST5.DATA=DATA %>%
            mutate(First=predict(lm(W~poly(A,as.numeric(BEST5.RMSE$I[1]))+poly(D,as.numeric(BEST5.RMSE$
                   Second=predict(lm(W~poly(A,as.numeric(BEST5.RMSE$I[2]))+poly(D,as.numeric(BEST5.RMSE
                   Third=predict(lm(W~poly(A,as.numeric(BEST5.RMSE$I[3]))+poly(D,as.numeric(BEST5.RMSE$
                   Fourth=predict(lm(W~poly(A,as.numeric(BEST5.RMSE$I[4]))+poly(D,as.numeric(BEST5.RMSE
                   Fifth=predict(lm(W~poly(A,as.numeric(BEST5.RMSE$I[5]))+poly(D,as.numeric(BEST5.RMSE$
```

Then, I want you to use the pipe `%>%` with `gather()` to create a new tibble called `BEST5.DATA2` that gathers all the predictions. A variable named `Model` should contain the values "First","Second","Third","Fourth", and "Fifth". A variable named `Predict` should contain the predictions corresponding to the appropriate models. In `gather()`, be sure to set `factor_key=T` to ensure that the new variable `Model` is a factor variable with ordered levels logically from "First" to "Fifth".

Code and Output:

```
BEST5.DATA2 <- BEST5.DATA %>%
  gather('First','Second','Third','Fourth','Fifth', key="Model", value="Predict", factor_key=T)

head(BEST5.DATA2) #DO NOT CHANGE
```

```
## # A tibble: 6 x 7
##       D     L     W     A     T Model Predict
```

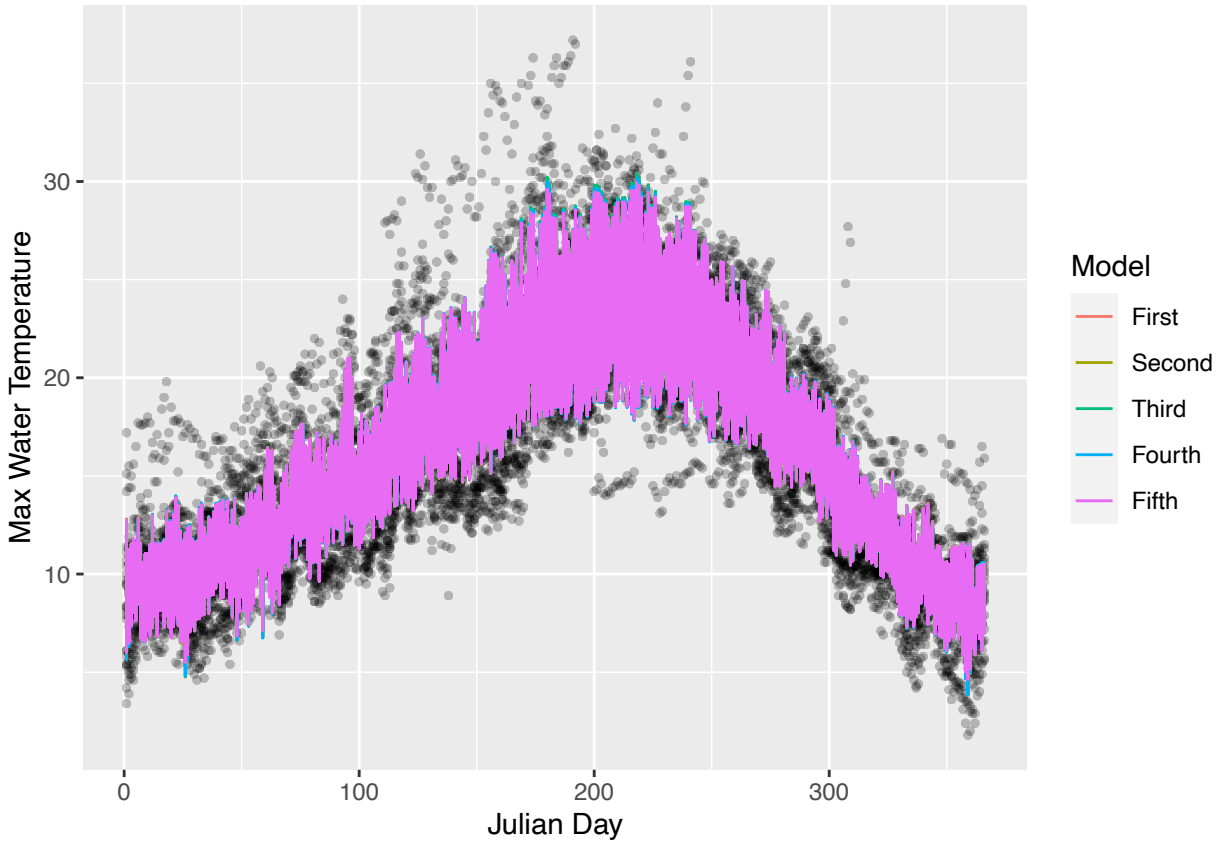
```
##   <dbl> <dbl> <dbl> <dbl> <dbl> <fct>   <dbl>
## 1     1   103  14.2  21.2     1 First   12.9
## 2     2   103  14.4  16.8     2 First   11.3
## 3     3   103  14.4  15.4     3 First   10.8
## 4     4   103  10.9  10.8     4 First    9.32
## 5     5   103  10.8  11.7     5 First    9.57
## 6     6   103  10.7  12.4     6 First    9.77
```

Q3 ***(5 Points)***

Create a figure that overlays the raw and predicted maximum water temperatures for the top 5 models given the Julian Day. The raw data needs to be shown in a scatter plot using `geom_point()` with `alpha=0.05` and `stroke=0` . The predictions should be created using `geom_line()` with different colors for each of the Models. Label the x-axis "Julian Day" and the y-axis "Max Water Temperature". In a complete sentence, explain how the predicted maximum water temperatures differ for the top five models.

Code and Output *(3 Points)*:

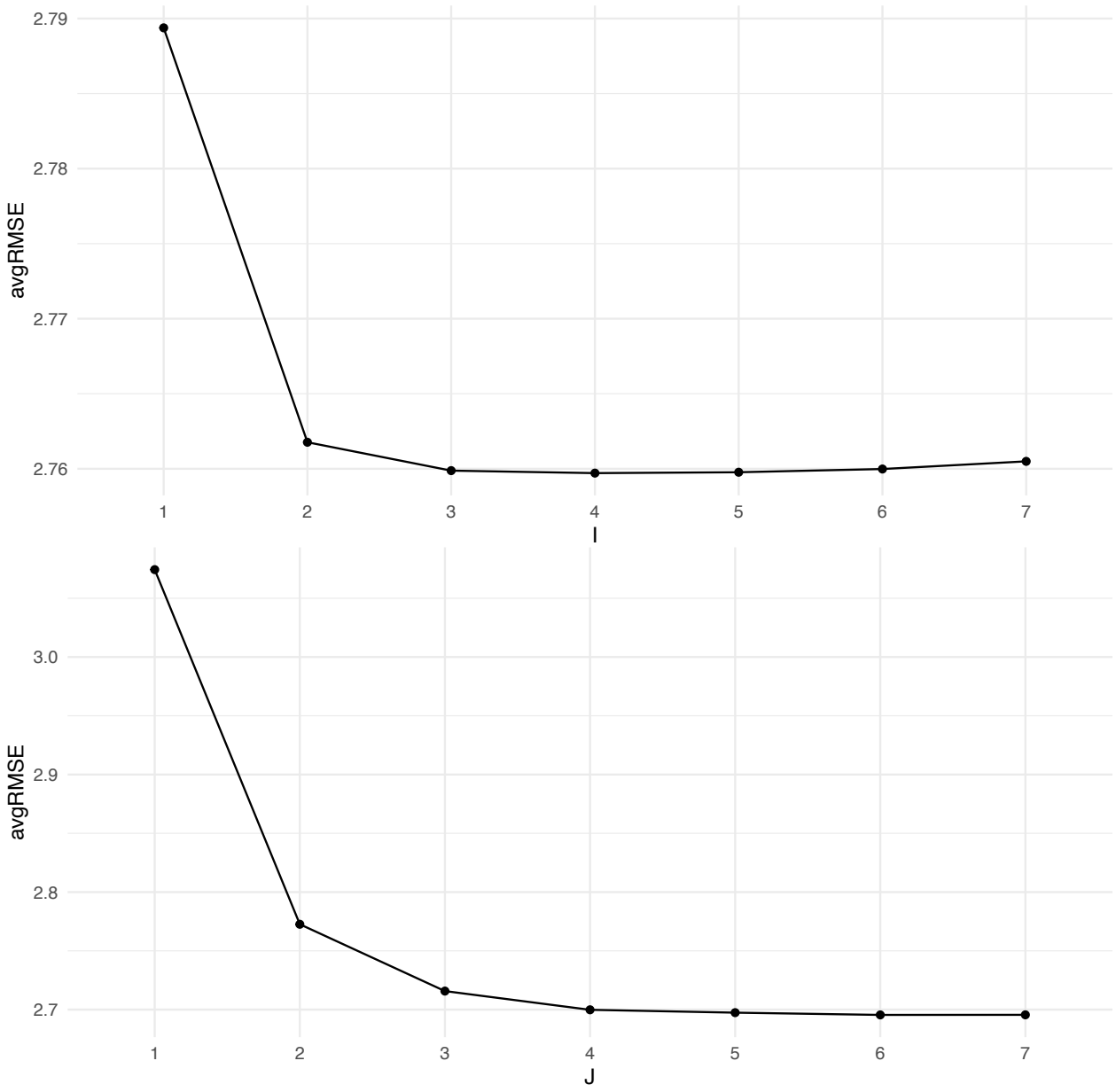
```
ggplot(BEST5.DATA2) +
  geom_point(aes(x = D,y = W), alpha = 0.05, stroke = 0) +
  geom_line(aes(x = D, y = Predict, color = Model)) +
  labs(x = "Julian Day", y = "Max Water Temperature")
```



Answer *(2 Points)*: We can assume by how the values align that the predicted temperatures and the top five models are very close to each other.

Q4 ***(3 Points)***

The following two figures show the marginal change in the average out-of-sample RMSE as I and J increase. Based on these figures, what I and J would you recommend going forward? Critically, think about what these figures are telling us. Give a reason for your answer based on these graphics. Answer the question below the two figures in complete sentences. Change `eval=F` to `eval=T` before knitting to HTML.



Answer: I would use the I=4 and J=7 values because we want the points when we have the lowest y-value since that is the point of lowest error.

Part 3: Plots of Best Model

Q1 ***(4 Points)***

I want you to create a simple function called `BEST.func()` that outputs a vector of length 2 where the first element is the I and the second element is the J that corresponds to the lowest RMSE. The output vector should be a numeric vector and not a tibble with factor variables. The only argument called "data" should be a data frame that is identical in format to `OUT.RMSE2`. The last line will process the function when `data=OUT.RMSE2` and save the best choices of `I` and `J` to an object called `BEST.CHOICE` and then print out the vector of length 2 with the ideal I and J leading to the smallest RMSE. I advise using the function `which.min()`.

Code and Output:

```
BEST.func=function(data){
  temp = data[which.min(data$RMSE),1:2]
  as.numeric(temp)
}

BEST.CHOICE=BEST.func(data=OUT.RMSE2) #DO NOT CHANGE
print(BEST.CHOICE) #DO NOT CHANGE
```

```
## [1] 3 6
```

Q2 ***(4 Points)***

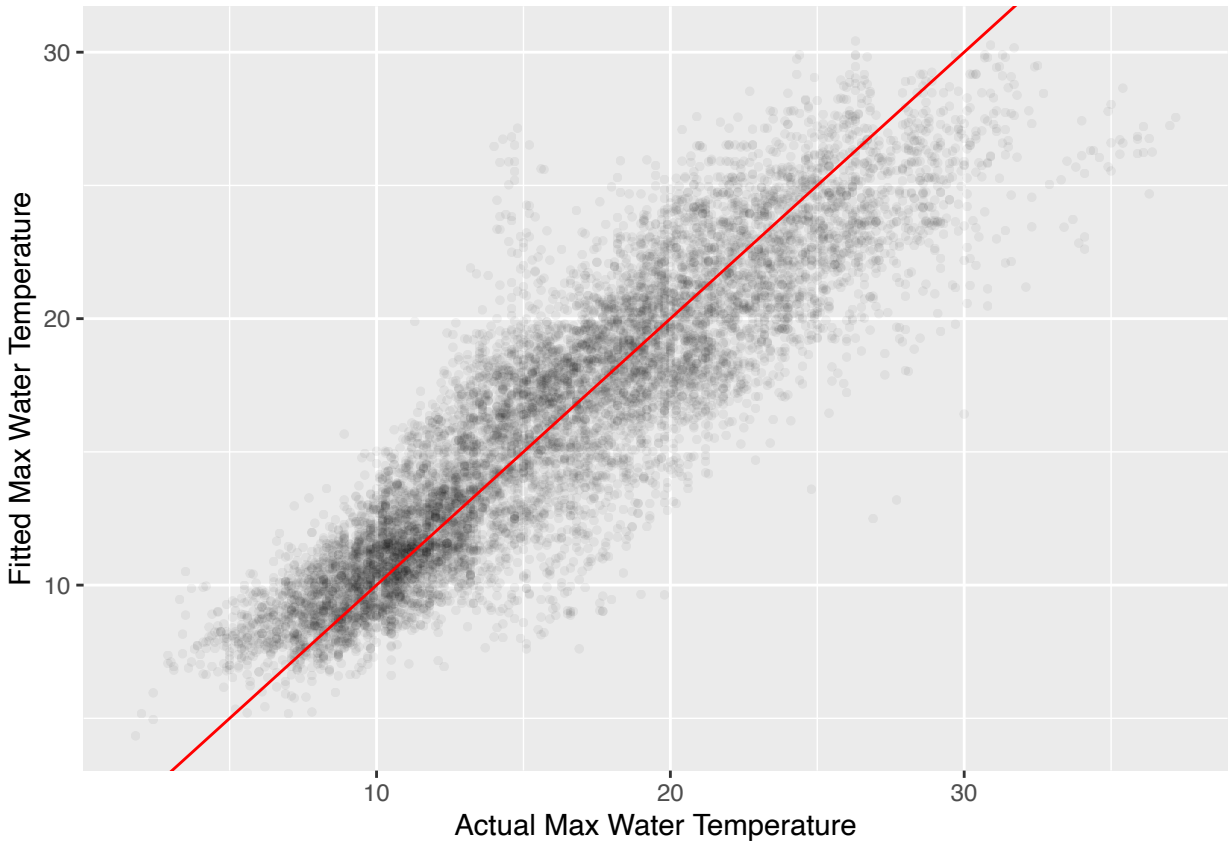
Based on the best model, I want you to obtain the fitted values for all observations in `DATA`. Then, I want you to create a figure that shows the relationship between the fitted maximum water temperatures under the best model and the actual maximum water temperatures. In this figure, you should label the x-axis "Actual Max Water Temperature" and the y-axis "Fitted Max Water Temperature". There should also be a 45 degree reference line with y-intercept equal to 0 indicating perfect prediction. Make this reference line the color "red". The function `predict()` or `augment()` could be helpful here along with the previously created function `train.model.func()`.

Code and Output:

```
ggplot(DATA) +
  geom_point(aes(x = W, y + predict(train.model.func(DATA, 3, 6))), alpha = 0.05, stroke = 0) +
  geom_abline(intercept = 0, slope = 1, color="red") +
  labs(x = "Actual Max Water Temperature", y = "Fitted Max Water Temperature")
```

```
## Warning in y + predict(train.model.func(DATA, 3, 6)): longer object length is
## not a multiple of shorter object length

## Warning in y + predict(train.model.func(DATA, 3, 6)): longer object length is
## not a multiple of shorter object length
```

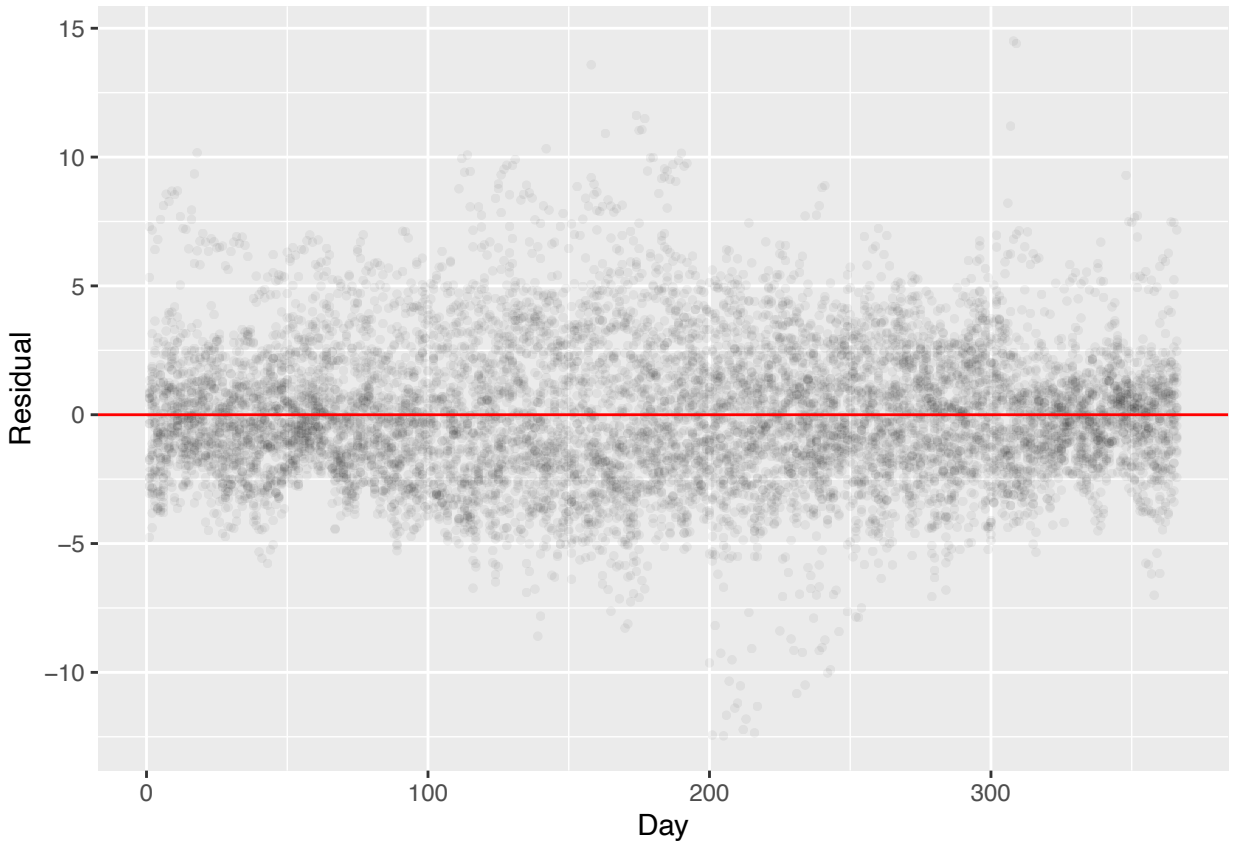


Q3 ***(4 Points)***

Based on the best model, I want you to obtain the residuals for all observations in `DATA`. Then, I want you to create a scatterplot that shows the different residuals over day `D`. In this figure, you should label the x-axis "Day" and the y-axis "Residual". There should also be a horizontal reference line at 0 indicating perfect prediction. Make this reference line the color "red". The function `residuals()`, works similar to `predict()`, but outputs the residuals and not the fitted values.

Code and Output:

```
ggplot(DATA) +
  geom_point(aes(x = D, y = residuals(train.model.func(DATA,3,6))), alpha = 0.05, stroke = 0) +
  geom_abline(intercept=0,slope=0,color="red") +
  labs(x = "Day", y = "Residual")
```


15
10
5
0
−5
−10
Residual
0
100
200
300
Day