analysis\_uown

C. Coleman

10/27/2021

# UOWN Data Analysis for Determining Correlation Between Stream Health Indicators.

*The following markdown conducts the statistical analysis of how different indicators of stream health can potentially predict each other. In the previous processing script, the main take away was that there were very few significant relationships between different chemical, fecal/microbial and biological indicators. The only significant relationships found between Turbity and E. Coli CFU. However, It struck me as peculiar to see biological indicator score and conductivity to have only one significant relationship. After further review of the exploratory figures, I now want to assess the data and relationships explored thus far for impact of outliers, determine if it is appropriate to remove outlying values, and see if this helps find more significant relationships.* *The second part will include test/train splitting to predict stream health indicator values in the MIDO data set and assess the fit of linear models to the MIDO data. This is a more depth recreation of the linear model exploration. Obviously, we found that there weren’t any significant relationships in exploratory analysis. However, this gives us a chance to assess how good a fit the models are in order to see if its even worth looking at the relationships.*

*Therefore, the following constitutes the statistical analysis protocol that will be conducted in this markdown:* #### PART 1: Outliers in stream health indicator (Biological Score, E. Coli (cfu), Conductivity, NO3 (mg/L), and Turbidity) linear relationships. *1. Assess each linear relationship previously performed for outliers in the data;* *2. Determine if it is appropriate to remove outliers;* *3. Rerun linear regressions for the different stream health indicator relationships and assess if more significant relationships exist.*

#### PART 2: Linear Model Fit Analysis of MIDO Data

*1. Conduct test/train data splitting for MIDO Data.* *2. Conduct linear modeling for relationships between stream health indicator predictors (Biological Score, E. Coli (cfu), Conductivity, NO3 (mg/L), and Turbidity) in the form of a workflow for the Test and Train MIDO data.* *3. Predict values for Test and Train MIDO data.* *4. Assess how well the MIDO linear model suite of relationships (Biological Score, E. Coli (cfu), Conductivity, NO3 (mg/L), and Turbidity) fir the test and train data using Root Mean Square Error (rmse). The result will measure how good of a fit the created models are to the data.*

#### PART 3: LASSO Modeling MIDO Data

*Conduct LASSO modeling of linear regression between Biological Score and all other variables and E. coli CFU and all other variables. Cross validation well be done by calculating RMSE for all models and compared to the RMSE for linear regression modeling of a Null model. Additionally, The best model will be selected using the R function select\_best(), for which residuals will be calculated between predicted and actual data.*

### Required packages

library(tidyverse) #Working with multiple Tidy packages

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.5 v dplyr 1.0.7  
## v tidyr 1.1.4 v stringr 1.4.0  
## v readr 2.0.1 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidymodels) #Building models

## Registered S3 method overwritten by 'tune':  
## method from   
## required\_pkgs.model\_spec parsnip

## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --

## v broom 0.7.9 v rsample 0.1.0   
## v dials 0.0.10 v tune 0.1.6   
## v infer 1.0.0 v workflows 0.2.3   
## v modeldata 0.1.1 v workflowsets 0.1.0   
## v parsnip 0.1.7 v yardstick 0.0.8   
## v recipes 0.1.17

## -- Conflicts ----------------------------------------- tidymodels\_conflicts() --  
## x scales::discard() masks purrr::discard()  
## x dplyr::filter() masks stats::filter()  
## x recipes::fixed() masks stringr::fixed()  
## x dplyr::lag() masks stats::lag()  
## x yardstick::spec() masks readr::spec()  
## x recipes::step() masks stats::step()  
## \* Learn how to get started at https://www.tidymodels.org/start/

library(dplyr) #data manipulation  
library(here) #setting pathways for saving files

## here() starts at C:/Data/Github/MADA/CARTERCOLEMAN\_MADA\_PROJECT

library(rpart) #Model fitting

##   
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':  
##   
## prune

library(ranger) #Model fitting  
library(glmnet) #Model fitting

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 4.1-3

library(knitr) #saving tables

### Load Data

*Start by loading the cleaned data labeled as MIDO, NORO, and BICO into the Global Environment.*

#Set the location of the desired dataframes  
MIDO\_location <- here::here("data","processed\_data","MIDO.RDS")  
NORO\_location <- here::here("data","processed\_data","NORO.RDS")  
BICO\_location <- here::here("data","processed\_data","BICO.RDS")  
  
#Load in the Dataframes  
MIDO <- readRDS(MIDO\_location)  
NORO <- readRDS(NORO\_location)  
BICO <- readRDS(BICO\_location)

\_Additionally, I am going to start of by log fitting the E. coli data. This is because that data can have counts several orders of magnitude higher than the variables it is being assessed with. Therefore, model fit may be artificially bad.

#Change the E.Coli (cfu) variable to log10 scale using mutate()  
MIDO %>%  
 mutate\_at(vars(e.coli.cfu), ~log10(.))

## # A tibble: 88 x 11  
## WSID biological\_score conductivity.uscm turbidity.ntu no3.mgL pH  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 MIDO301 13 53 5 0.4 6.7   
## 2 MIDO601 25 65 7 0.6 7.2   
## 3 MIDO701 10 56 7 0.7 7.3   
## 4 MIDO801 15 89 16 1 8   
## 5 MIDO802 12 92 5 3.6 7.3   
## 6 MIDO802 16 77 25 1 6.1   
## 7 MIDO601 18 59 17.9 2.3 5.7   
## 8 MIDO802 6 79 10.9 4.7 5.9   
## 9 MIDO803 5 127 6.3 8.7 6.8   
## 10 MIDO704 22 109 29.2 5 6.05  
## # ... with 78 more rows, and 5 more variables: e.coli.cfu <dbl>, month <dbl>,  
## # year <dbl>, day <dbl>, stream\_ID <chr>

NORO %>%  
 mutate\_at(vars(e.coli.cfu), ~log10(.))

## # A tibble: 35 x 11  
## WSID biological\_score conductivity.uscm turbidity.ntu no3.mgL pH  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 NORO201 21 46 8 0.6 7.2   
## 2 NORO401 22 45 11 0.6 7   
## 3 NORO503 13 54 20 0.9 7.3   
## 4 NORO601 2 685 15 5.1 6.6   
## 5 NORO602 28 44 14 0.7 7   
## 6 NORO601 1 556 9.2 23.3 6.2   
## 7 NORO503 16 65 20.3 0.3 5.5   
## 8 NORO401 31 39 6.1 0.8 6.5   
## 9 NORO503 8 55 8.5 1.1 6.3   
## 10 NORO503 15 62.5 12.5 1 6.61  
## # ... with 25 more rows, and 5 more variables: e.coli.cfu <dbl>, month <dbl>,  
## # year <dbl>, day <dbl>, stream\_ID <chr>

BICO %>%  
 mutate\_at(vars(e.coli.cfu), ~log10(.))

## # A tibble: 23 x 11  
## WSID biological\_score conductivity.uscm turbidity.ntu no3.mgL pH  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 BICO102 22 83 4 0.8 7.2   
## 2 BICO201 20 41 14.9 1.3 6.3   
## 3 BICO201 22 46.5 10.2 0.2 6.56  
## 4 BICO201 29 42.9 20 0.29 7.06  
## 5 BICO301 7 42.9 39.1 0.13 6.76  
## 6 BICO201 25 97.6 9.8 0.75 6.2   
## 7 BICO301 21 87.9 7.7 0.85 6.6   
## 8 BICO201 23 51.3 3.4 0.69 6.73  
## 9 BICO201 21 54.8 2.4 0.1 6.85  
## 10 BICO201 21 54.7 6.9 4.31 6.67  
## # ... with 13 more rows, and 5 more variables: e.coli.cfu <dbl>, month <dbl>,  
## # year <dbl>, day <dbl>, stream\_ID <chr>

### Load all the created linear models from the processing data markdown.

\_We need to load the exploratory linear models created previously in order to assess them for outliers that impact the linear regressions. This is because the cooksdistance() function calls for a linear model input.

#### MIDO

*Biological Score*

#Loading exploratory linear models for MIDO data with Biological Score as Outcome variable.  
BS\_Con\_location <- here::here("data","processed\_data","BS\_Con.rds")  
bs\_con\_lm <- readRDS(BS\_Con\_location)  
  
BS\_ECFU\_location <- here::here("data","processed\_data","BS\_ECFU.rds")  
bs\_ecfu\_lm <- readRDS(BS\_ECFU\_location)  
  
BS\_no3\_location <- here::here("data","processed\_data","BS\_no3.rds")  
bs\_no3\_lm <- readRDS(BS\_no3\_location)  
  
BS\_Turb\_location <- here::here("data","processed\_data","BS\_Turb.rds")  
bs\_turb\_lm <- readRDS(BS\_Turb\_location)

*ECFU*

#Loading exploratory linear models for MIDO data with E. coli (cfu) as Outcome variable.  
ECFU\_Con\_location <- here::here("data","processed\_data","BS\_Con.rds")  
ecfu\_con\_lm <- readRDS(ECFU\_Con\_location)  
  
ECFU\_no3\_location <- here::here("data","processed\_data","BS\_no3.rds")  
ecfu\_no3\_lm <- readRDS(ECFU\_no3\_location)  
  
ECFU\_Turb\_location <- here::here("data","processed\_data","BS\_Turb.rds")  
ecfu\_turb\_lm <- readRDS(ECFU\_Turb\_location)

#### BICO

*Biological Score*

#Loading exploratory linear models for BICO data with Biological Score as Outcome variable.  
B\_BS\_Con\_location <- here::here("data","processed\_data","B\_BS\_Con.rds")  
b\_bs\_con\_lm <- readRDS(B\_BS\_Con\_location)  
  
B\_BS\_ECFU\_location <- here::here("data","processed\_data","B\_BS\_ECFU.rds")  
b\_bs\_ecfu\_lm <- readRDS(B\_BS\_ECFU\_location)  
  
B\_BS\_no3\_location <- here::here("data","processed\_data","B\_BS\_no3.rds")  
b\_bs\_no3\_lm <- readRDS(B\_BS\_no3\_location)  
  
B\_BS\_Turb\_location <- here::here("data","processed\_data","B\_BS\_Turb.rds")  
b\_bs\_turb\_lm <- readRDS(B\_BS\_Turb\_location)

*ECFU*

#Loading exploratory linear models for BICO data with E. coli (cfu) as Outcome variable.  
B\_ECFU\_Con\_location <- here::here("data","processed\_data","B\_BS\_Con.rds")  
b\_ecfu\_con\_lm <- readRDS(B\_ECFU\_Con\_location)  
  
B\_ECFU\_no3\_location <- here::here("data","processed\_data","B\_BS\_no3.rds")  
b\_ecfu\_no3\_lm <- readRDS(B\_ECFU\_no3\_location)  
  
B\_ECFU\_Turb\_location <- here::here("data","processed\_data","B\_BS\_Turb.rds")  
b\_ecfu\_turb\_lm <- readRDS(B\_ECFU\_Turb\_location)

#### NORO

*Biological Score*

#Loading exploratory linear models for NORO data with Biological Score as Outcome variable.  
N\_BS\_Con\_location <- here::here("data","processed\_data","N\_BS\_Con.rds")  
n\_bs\_con\_lm <- readRDS(N\_BS\_Con\_location)  
  
N\_BS\_ECFU\_location <- here::here("data","processed\_data","N\_BS\_ECFU.rds")  
n\_bs\_ecfu\_lm <- readRDS(N\_BS\_ECFU\_location)  
  
N\_BS\_no3\_location <- here::here("data","processed\_data","N\_BS\_no3.rds")  
n\_bs\_no3\_lm <- readRDS(N\_BS\_no3\_location)  
  
N\_BS\_Turb\_location <- here::here("data","processed\_data","N\_BS\_Turb.rds")  
n\_bs\_turb\_lm <- readRDS(N\_BS\_Turb\_location)

*ECFU*

#Loading exploratory linear models for NORO data with E. coli (cfu) as Outcome variable.  
N\_ECFU\_Con\_location <- here::here("data","processed\_data","N\_BS\_Con.rds")  
n\_ecfu\_con\_lm <- readRDS(N\_ECFU\_Con\_location)  
  
N\_ECFU\_no3\_location <- here::here("data","processed\_data","N\_BS\_no3.rds")  
n\_ecfu\_no3\_lm <- readRDS(N\_ECFU\_no3\_location)  
  
N\_ECFU\_Turb\_location <- here::here("data","processed\_data","N\_BS\_Turb.rds")  
n\_ecfu\_turb\_lm <- readRDS(N\_ECFU\_Turb\_location)

### Outlier identification using Cook’s Distance.

*Outliers in these data could be potentially ttreated as errors in reporting. This is because the data is predominantly collected by non-scientist volunteers, who have had training with scientists. As such, it is likely that sampling of mis-identification of macroinvertebrates taxa may occurred. Therefore, an argument can be made that assessing for anomalies in the data could help tease apart the true relationships that exist between different stream health indicators.*

*As such, we will start by analyzing the data for outliers by evaluating relationships with cook’s distance.*

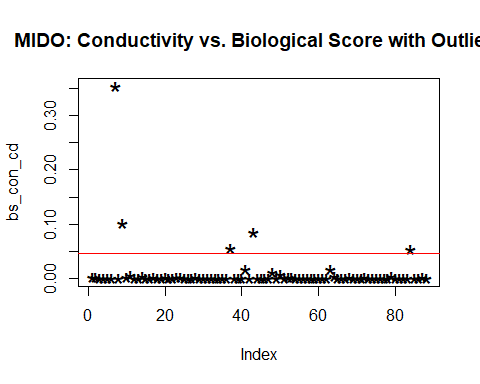
#### Set sample size for sample size parameter (4/n) for each data set (MIDO, NORO, BICO)

#This makes an argument that makes the sum number of rows in a particular data set equal to the sample size of the data set and names it accordingly.  
sample\_size\_MIDO <- nrow(MIDO)  
sample\_size\_NORO <- nrow(NORO)  
sample\_size\_BICO <- nrow(BICO)

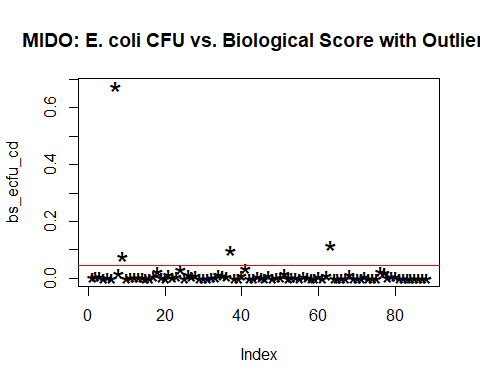
#### MIDO

*Outcome: Biological Score*

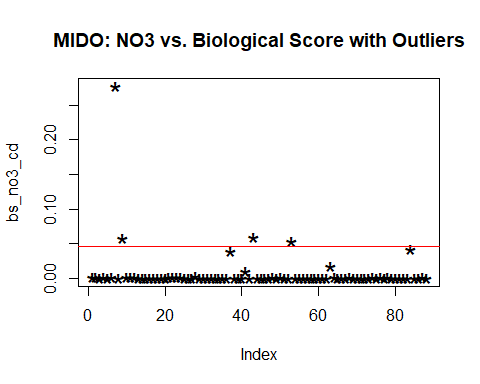
#Creates a dataframe that stores the cook's distance analysis of influential data points.  
bs\_con\_cd <- cooks.distance(bs\_con\_lm)  
plot(bs\_con\_cd, pch="\*", cex=2, main="MIDO: Conductivity vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_MIDO, col="red")



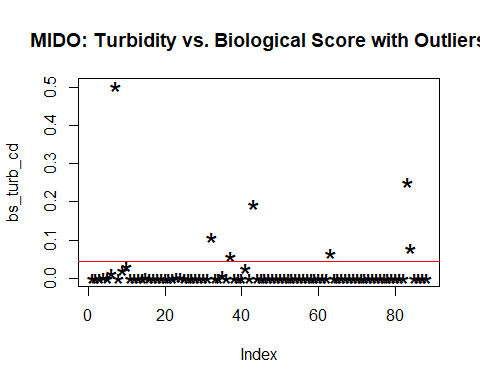
bs\_ecfu\_cd <- cooks.distance(bs\_ecfu\_lm)  
plot(bs\_ecfu\_cd, pch="\*", cex=2, main="MIDO: E. coli CFU vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_MIDO, col="red")



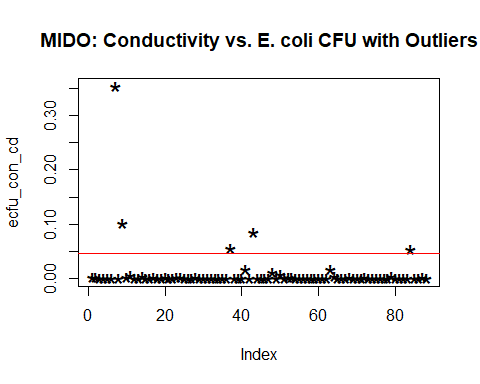
bs\_no3\_cd <- cooks.distance(bs\_no3\_lm)  
plot(bs\_no3\_cd, pch="\*", cex=2, main="MIDO: NO3 vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_MIDO, col="red")



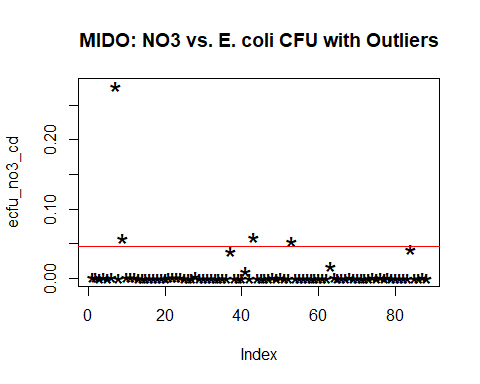
bs\_turb\_cd <- cooks.distance(bs\_turb\_lm)  
plot(bs\_turb\_cd, pch="\*", cex=2, main="MIDO: Turbidity vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_MIDO, col="red")

 *Outcome: E.coli CFU*

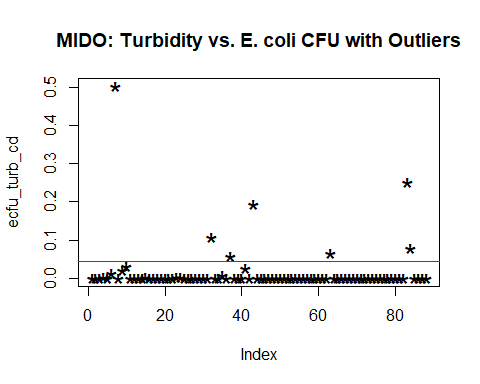
#Creates a dataframe that stores the cook's distance analysis of influential data points.  
ecfu\_con\_cd <- cooks.distance(ecfu\_con\_lm)  
plot(ecfu\_con\_cd, pch="\*", cex=2, main="MIDO: Conductivity vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_MIDO, col="red")



ecfu\_no3\_cd <- cooks.distance(ecfu\_no3\_lm)  
plot(ecfu\_no3\_cd, pch="\*", cex=2, main="MIDO: NO3 vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_MIDO, col="red")



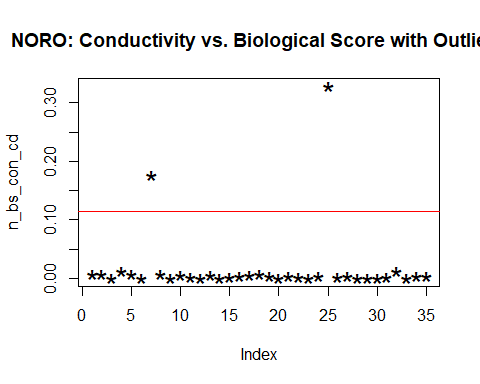
ecfu\_turb\_cd <- cooks.distance(ecfu\_turb\_lm)  
plot(ecfu\_turb\_cd, pch="\*", cex=2, main="MIDO: Turbidity vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_MIDO, col="red")



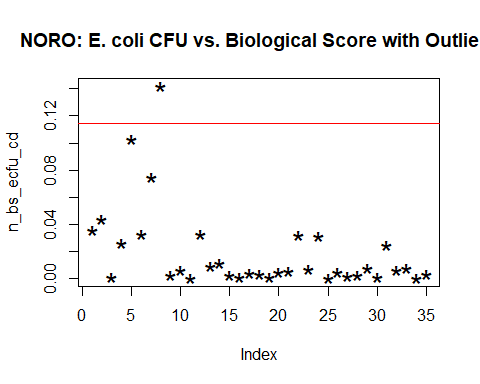
#### NORO

*Outcome: Biological Score*

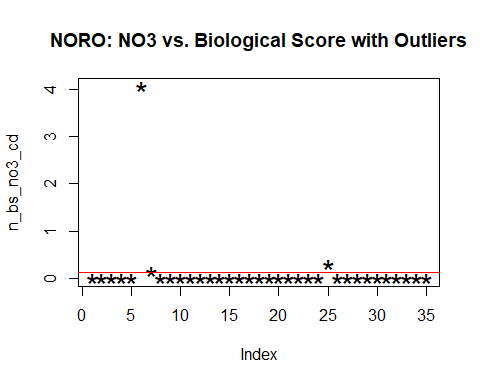
#Creates a dataframe that stores the cook's distance analysis of influential data points.  
n\_bs\_con\_cd <- cooks.distance(n\_bs\_con\_lm)  
plot(n\_bs\_con\_cd, pch="\*", cex=2, main="NORO: Conductivity vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_NORO, col="red")



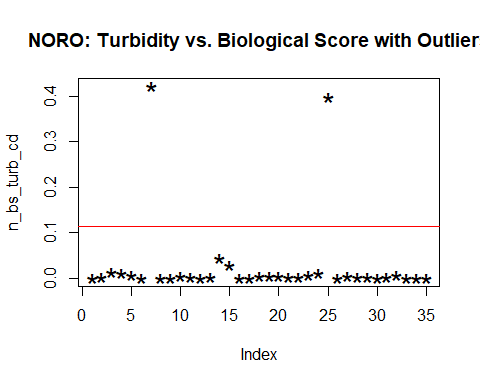
n\_bs\_ecfu\_cd <- cooks.distance(n\_bs\_ecfu\_lm)  
plot(n\_bs\_ecfu\_cd, pch="\*", cex=2, main="NORO: E. coli CFU vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_NORO, col="red")



n\_bs\_no3\_cd <- cooks.distance(n\_bs\_no3\_lm)  
plot(n\_bs\_no3\_cd, pch="\*", cex=2, main="NORO: NO3 vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_NORO, col="red")

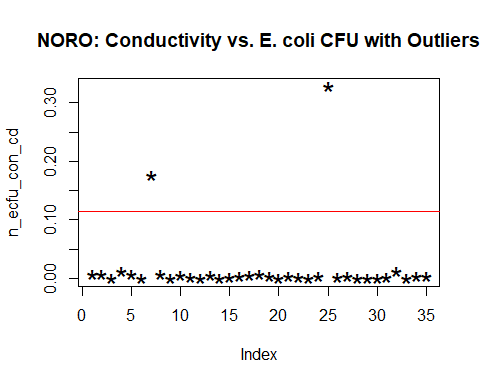


n\_bs\_turb\_cd <- cooks.distance(n\_bs\_turb\_lm)  
plot(n\_bs\_turb\_cd, pch="\*", cex=2, main="NORO: Turbidity vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_NORO, col="red")

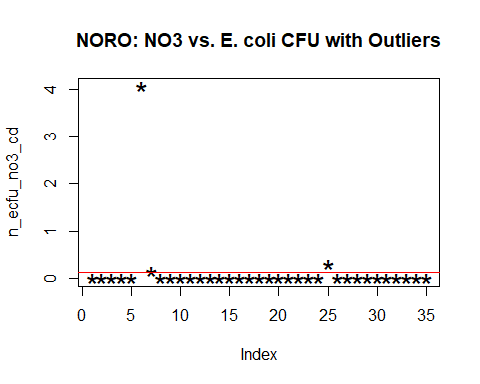


*Outcome: E.coli CFU*

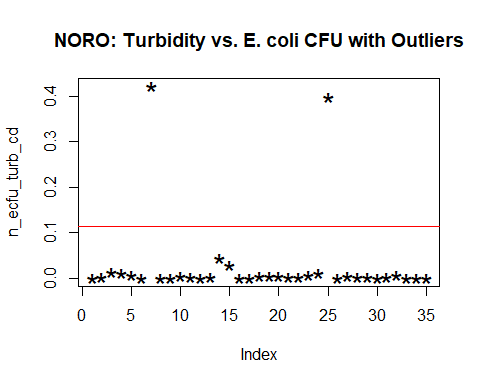
#Creates a dataframe that stores the cook's distance analysis of influential data points.  
n\_ecfu\_con\_cd <- cooks.distance(n\_ecfu\_con\_lm)  
plot(n\_ecfu\_con\_cd, pch="\*", cex=2, main="NORO: Conductivity vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_NORO, col="red")



n\_ecfu\_no3\_cd <- cooks.distance(n\_ecfu\_no3\_lm)  
plot(n\_ecfu\_no3\_cd, pch="\*", cex=2, main="NORO: NO3 vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_NORO, col="red")



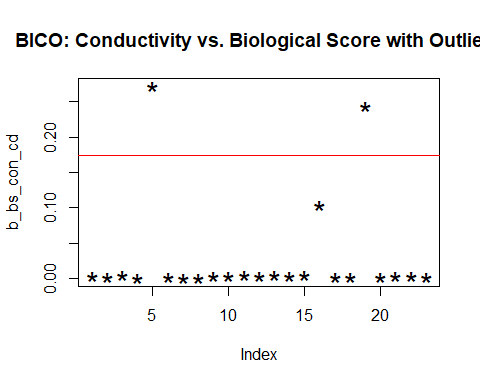
n\_ecfu\_turb\_cd <- cooks.distance(n\_ecfu\_turb\_lm)  
plot(n\_ecfu\_turb\_cd, pch="\*", cex=2, main="NORO: Turbidity vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_NORO, col="red")



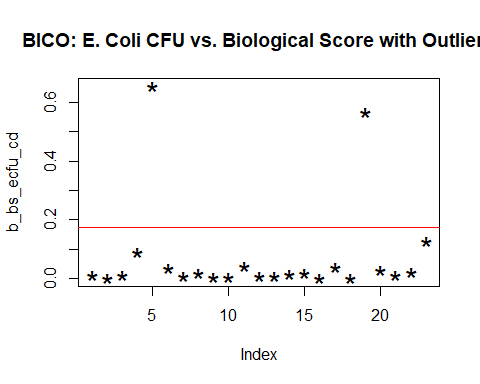
#### BICO

*Outcome: Biological Score*

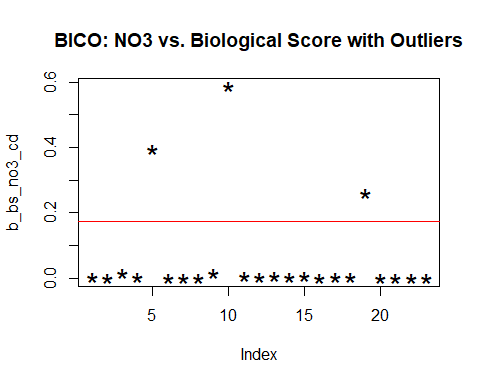
#Creates a dataframe that stores the cook's distance analysis of influential data points.  
b\_bs\_con\_cd <- cooks.distance(b\_bs\_con\_lm)  
plot(b\_bs\_con\_cd, pch="\*", cex=2, main="BICO: Conductivity vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_BICO, col="red")



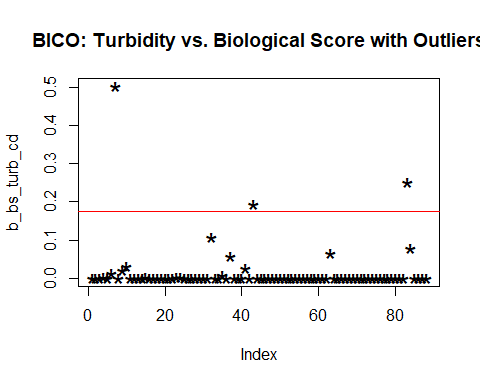
b\_bs\_ecfu\_cd <- cooks.distance(b\_bs\_ecfu\_lm)  
plot(b\_bs\_ecfu\_cd, pch="\*", cex=2, main="BICO: E. Coli CFU vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_BICO, col="red")



b\_bs\_no3\_cd <- cooks.distance(b\_bs\_no3\_lm)  
plot(b\_bs\_no3\_cd, pch="\*", cex=2, main="BICO: NO3 vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_BICO, col="red")

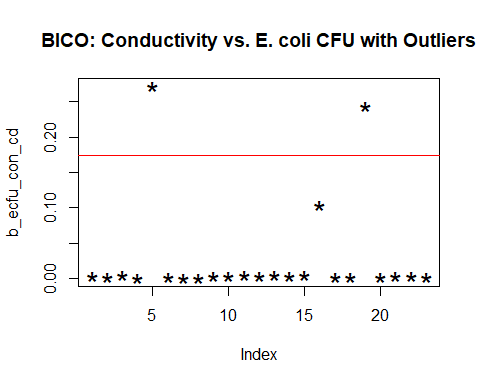


b\_bs\_turb\_cd <- cooks.distance(bs\_turb\_lm)  
plot(b\_bs\_turb\_cd, pch="\*", cex=2, main="BICO: Turbidity vs. Biological Score with Outliers")  
abline(h = 4/sample\_size\_BICO, col="red")

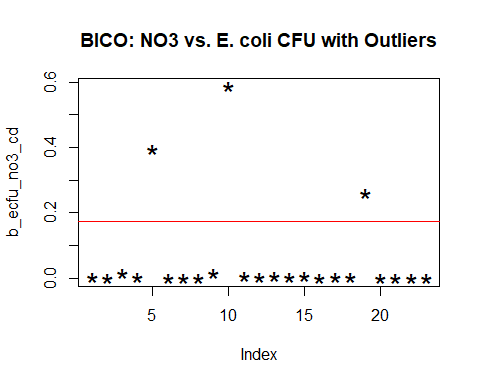


*Outcome: E.coli CFU*

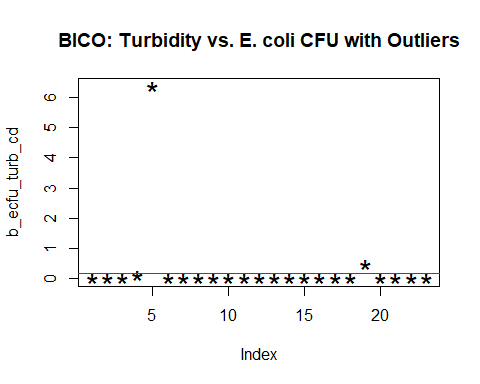
#Creates a dataframe that stores the cook's distance analysis of influential data points.  
b\_ecfu\_con\_cd <- cooks.distance(b\_ecfu\_con\_lm)  
plot(b\_ecfu\_con\_cd, pch="\*", cex=2, main="BICO: Conductivity vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_BICO, col="red")



b\_ecfu\_no3\_cd <- cooks.distance(b\_ecfu\_no3\_lm)  
plot(b\_ecfu\_no3\_cd, pch="\*", cex=2, main="BICO: NO3 vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_BICO, col="red")



b\_ecfu\_turb\_cd <- cooks.distance(b\_ecfu\_turb\_lm)  
plot(b\_ecfu\_turb\_cd, pch="\*", cex=2, main="BICO: Turbidity vs. E. coli CFU with Outliers")  
abline(h = 4/sample\_size\_BICO, col="red")



*The results of the Cook’s Distance show that while there are outliers, very few of them actually exceed a Cook’s Distance value (CDV) of 1. The CDV value of 1 is significant because it is the statistical community consensus value that indicates an influential outlier. As such, it is my educated opinion to leave all data points in, as I do not believe they are skewing liner modle results in a significant enough way to warrant removal.*

# Part 2: Linear Model Fit Assessment for MIDO Data

*At this point,the main take-aways from data analysis thus far is that correlation of stream health indicators showed that most interactions were non-significant, with the exception of E. coli (cfu) and turbidity. Additionally, the data set is most likely not skewed by influential outliers.*

*Moving forward, I want to focus in on just the MIDO dataset. As you can tell, working with many variables in three different data sets can be quite cumbersome to read through. For ease of interpretation, the MIDO data set will serve as the data frame where I test how robust the linear fit models actually are for the data. MIDO is the only data set large enough that when we split the data set into a “test” and “train” data set, I actually have confidence in the resulting model fit analysis. For comparison, MIDO has an n = 88; NORO has an n = 35; and BICO has an n = 23. The only thing we are missing by dropping the other sites is assessing site-by-site comparison, which I am choosing not to focus on anyway.*

#### Splitting MIDO Data

*As discussed above, we need to split the MIDO data into a test and train data set. We do this to see if we get approximately the same linear relationship between the larger (75% of MIDO) train data and the smaller (25% of MIDO) test data.*

#Split the data with 1:4 ratio.  
data\_split <- initial\_split(MIDO, prop = 3/4)  
  
# Create data frames for the two sets:  
train\_data <- training(data\_split)  
test\_data <- testing(data\_split)

*Create a recipe for each linear model relationship in the MIDO dataframe.*

Recipe\_bs\_con <- recipe(biological\_score ~ conductivity.uscm, data = MIDO)  
Recipe\_bs\_ecfu <- recipe(biological\_score ~ e.coli.cfu, data = MIDO)  
Recipe\_bs\_no3 <- recipe(biological\_score ~ no3.mgL, data = MIDO)  
Recipe\_bs\_turb <- recipe(biological\_score ~ turbidity.ntu, data = MIDO)  
Recipe\_ecfu\_con <- recipe(e.coli.cfu ~ e.coli.cfu, data = MIDO)  
Recipe\_ecfu\_no3 <- recipe(e.coli.cfu ~ no3.mgL, data = MIDO)  
Recipe\_ecfu\_turb <- recipe(e.coli.cfu ~ turbidity.ntu, data = MIDO)

*With our seven linear model recipes created, we will define of linear regression model pipe.*

linear\_mod <- linear\_reg() %>%   
 set\_engine("lm") %>%  
 set\_mode("regression")

*Next, create a workflow that adds our seven recipes and model together.*

#Biological Score and Conductivity  
bs\_con\_wflow <-   
 workflow() %>%   
 add\_model(linear\_mod) %>%   
 add\_recipe(Recipe\_bs\_con)  
  
#Biological Score and E. coli(cfu)  
bs\_ecfu\_wflow <-   
 workflow() %>%   
 add\_model(linear\_mod) %>%   
 add\_recipe(Recipe\_bs\_ecfu)  
  
#Biological Score and NO3  
bs\_no3\_wflow <-   
 workflow() %>%   
 add\_model(linear\_mod) %>%   
 add\_recipe(Recipe\_bs\_no3)  
  
#Biological Score and Turbidity  
bs\_turb\_wflow <-   
 workflow() %>%   
 add\_model(linear\_mod) %>%   
 add\_recipe(Recipe\_bs\_turb)  
  
#E. Coli (cfu) and Conductivity  
ecfu\_con\_wflow <-   
 workflow() %>%   
 add\_model(linear\_mod) %>%   
 add\_recipe(Recipe\_ecfu\_con)  
  
#E. Coli (cfu) and NO3  
ECFU\_no3\_wflow <-   
 workflow() %>%   
 add\_model(linear\_mod) %>%   
 add\_recipe(Recipe\_ecfu\_no3)  
  
#E. Coli (cfu) and Turbidity  
ecfu\_turb\_wflow <-   
 workflow() %>%   
 add\_model(linear\_mod) %>%   
 add\_recipe(Recipe\_ecfu\_turb)

## Modeling Train Data Predictions Using Workflows

*Using the workflow above, lets now fit each pf the seven stream health indicator linear models to the train data set.*

*Defining a command that runs model fitting to each of the seven models*

bs\_con\_fit <-   
 bs\_con\_wflow %>%   
 fit(data = train\_data)  
  
bs\_ecfu\_fit <-   
 bs\_ecfu\_wflow %>%   
 fit(data = train\_data)  
  
bs\_no3\_fit <-   
 bs\_no3\_wflow %>%   
 fit(data = train\_data)  
  
bs\_turb\_fit <-   
 bs\_turb\_wflow %>%   
 fit(data = train\_data)  
  
ecfu\_con\_fit <-   
 ecfu\_con\_wflow %>%   
 fit(data = train\_data)  
  
ecfu\_no3\_fit <-   
 ECFU\_no3\_wflow %>%   
 fit(data = train\_data)  
  
ecfu\_turb\_fit <-   
 ecfu\_turb\_wflow %>%   
 fit(data = train\_data)

*Pull linear regression fit models using parsnip().*

bs\_con\_fit %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 20.7 2.55 8.11 2.06e-11  
## 2 conductivity.uscm -0.0387 0.0324 -1.19 2.37e- 1

bs\_ecfu\_fit %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 18.3 0.950 19.2 2.70e-28  
## 2 e.coli.cfu -0.000799 0.000811 -0.984 3.29e- 1

bs\_no3\_fit %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 17.5 0.903 19.4 1.70e-28  
## 2 no3.mgL 0.132 0.149 0.883 3.81e- 1

bs\_turb\_fit %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 17.2 1.15 14.9 1.98e-22  
## 2 turbidity.ntu 0.0675 0.0820 0.823 4.13e- 1

Ecoli\_con\_lm <- ecfu\_con\_fit %>%   
 extract\_fit\_parsnip() %>%   
 tidy()  
  
ecfu\_no3\_fit %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 588. 139. 4.24 0.0000740  
## 2 no3.mgL -10.0 22.9 -0.437 0.663

ecfu\_turb\_fit %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 387. 175. 2.22 0.0303  
## 2 turbidity.ntu 18.2 12.4 1.47 0.146

#### Using Workflows to Make Predictions in the train data set.

*This is important because we are basically going to try and recreate the actual MIDO data set using the test and trained data and our linear models. We will eventually compare the test data predictions with the train data predictions to assess if our model is actually a good estimator of the relationships between stream health indicators. Additionally, this prediction step allows us to “artificially” create a larger n for our data, allowing fit to be assessed more accurately. This will be done using the augment() function to make predicted outcomes in the test and train data.*

bs\_con\_aug <-   
 augment(bs\_con\_fit, train\_data)  
  
bs\_ecfu\_aug <-   
 augment(bs\_ecfu\_fit, train\_data)  
  
bs\_no3\_aug <-   
 augment(bs\_no3\_fit, train\_data)  
  
bs\_turb\_aug <-   
 augment(bs\_turb\_fit, train\_data)  
  
ecfu\_con\_aug <-   
 augment(ecfu\_con\_fit, train\_data)  
  
ecfu\_no3\_aug <-   
 augment(ecfu\_no3\_fit, train\_data)  
  
ecfu\_turb\_aug <-   
 augment(ecfu\_turb\_fit, train\_data)

#### Using Root Mean Squared Error (RMSE) to Assess Model Fit.

*Assessing fit for train data predictions by calculating RMSE for the test data predictions of stream health indicator linear relationships.*

bs\_con\_aug %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.63

bs\_ecfu\_aug %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.66

bs\_no3\_aug %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.67

bs\_turb\_aug %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.67

Ecoli\_con\_RMSE <- ecfu\_con\_aug %>%  
 rmse(truth = e.coli.cfu, .pred)  
   
ecfu\_no3\_aug %>%  
 rmse(truth = e.coli.cfu, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 1024.

ecfu\_turb\_aug %>%  
 rmse(truth = e.coli.cfu, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 1009.

## Modeling Test Data Predictions Using Workflows

*Using the workflow above, lets now fit each pf the seven stream health indicator linear models to the test data set.*

*Defining a command that runs model fitting to each of the seven models*

bs\_con\_fit\_test <-   
 bs\_con\_wflow %>%   
 fit(data = test\_data)  
  
bs\_ecfu\_fit\_test <-   
 bs\_ecfu\_wflow %>%   
 fit(data = test\_data)  
  
bs\_no3\_fit\_test <-   
 bs\_no3\_wflow %>%   
 fit(data = test\_data)  
  
bs\_turb\_fit\_test <-   
 bs\_turb\_wflow %>%   
 fit(data = test\_data)  
  
ecfu\_con\_fit\_test <-   
 ecfu\_con\_wflow %>%   
 fit(data = test\_data)  
  
ecfu\_no3\_fit\_test <-   
 ECFU\_no3\_wflow %>%   
 fit(data = test\_data)  
  
ecfu\_turb\_fit\_test <-   
 ecfu\_turb\_wflow %>%   
 fit(data = test\_data)

*Pull linear regression fit models using parsnip().*

bs\_con\_fit\_test %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 16.6 3.19 5.20 0.0000439  
## 2 conductivity.uscm 0.0119 0.0377 0.317 0.755

bs\_ecfu\_fit\_test %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 16.9 2.08 8.11 0.0000000949  
## 2 e.coli.cfu 0.00154 0.00379 0.407 0.688

bs\_no3\_fit\_test %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 15.7 1.83 8.57 0.0000000395  
## 2 no3.mgL 1.43 0.942 1.51 0.146

bs\_turb\_fit\_test %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 14.7 1.93 7.63 0.000000239  
## 2 turbidity.ntu 0.358 0.179 2.00 0.0588

Ecoli\_con\_test\_lm <- ecfu\_con\_fit\_test %>%   
 extract\_fit\_parsnip() %>%   
 tidy()  
  
ecfu\_no3\_fit\_test %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 526. 104. 5.03 0.0000637  
## 2 no3.mgL -103. 53.8 -1.91 0.0701

ecfu\_turb\_fit\_test %>%   
 extract\_fit\_parsnip() %>%   
 tidy()

## # A tibble: 2 x 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) 261. 117. 2.23 0.0373  
## 2 turbidity.ntu 17.1 10.9 1.57 0.131

#### Using Workflows to Make Predictions in the test data set.

*This is important because we are basically going to try and recreate the actual MIDO data set using the test and trained data and our linear models. This prediction step allows us to “artificially” create a larger n for our data, allowing fit to be assessed more accurately for data predicted from split data. This will be done using the augment() function to make predicted outcomes in the test and train data.*

bs\_con\_aug\_test <-   
 augment(bs\_con\_fit, test\_data)  
  
bs\_ecfu\_aug\_test <-   
 augment(bs\_ecfu\_fit, test\_data)  
  
bs\_no3\_aug\_test <-   
 augment(bs\_no3\_fit, test\_data)  
  
bs\_turb\_aug\_test <-   
 augment(bs\_turb\_fit, test\_data)  
  
ecfu\_con\_aug\_test <-   
 augment(ecfu\_con\_fit, test\_data)  
  
ecfu\_no3\_aug\_test <-   
 augment(ecfu\_no3\_fit, test\_data)  
  
ecfu\_turb\_aug\_test <-   
 augment(ecfu\_turb\_fit, test\_data)

#### Using Root Mean Squared Error (RMSE) to Assess Model Fit.

*Assessing fit for train data predictions by calculating RMSE for the test data predictions of stream health indicator linear relationships.*

bs\_con\_aug\_test %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.80

bs\_ecfu\_aug\_test %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.57

bs\_no3\_aug\_test %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.46

bs\_turb\_aug\_test %>%  
 rmse(truth = biological\_score, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 6.33

Ecoli\_con\_RMSE\_test <- ecfu\_con\_aug\_test %>%  
 rmse(truth = e.coli.cfu, .pred)  
   
ecfu\_no3\_aug\_test %>%  
 rmse(truth = e.coli.cfu, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 419.

ecfu\_turb\_aug\_test %>%  
 rmse(truth = e.coli.cfu, .pred)

## # A tibble: 1 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 rmse standard 386.

#### Save significant Test and Train linear models to the results folder.

#Create a table for E. coli (cfu) as a function of Conductivity Linear Fit Model in the MIDO train data.  
MIDO\_ECFU\_Con\_fit <- kable(as.tibble(Ecoli\_con\_lm),  
 caption = "Linear Fit Model: E. Coli CFU predicted by Conductivity in the Trained MIDO Data Set")

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

MIDO\_ECFU\_Con\_fit

Linear Fit Model: E. Coli CFU predicted by Conductivity in the Trained MIDO Data Set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 564.4167 | 127.2408 | 4.435816 | 3.62e-05 |

#Create a table for E. coli (cfu) as a function of Conductivity Linear Fit Model in the MIDO test data.  
  
MIDO\_ECFU\_Con\_fit\_test <- kable(as.tibble(Ecoli\_con\_test\_lm),  
 caption = "Linear Fit Model: E. Coli CFU predicted by Conductivity in the Test MIDO Data Set")  
  
MIDO\_ECFU\_Con\_fit\_test

Linear Fit Model: E. Coli CFU predicted by Conductivity in the Test MIDO Data Set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 394.4773 | 83.6566 | 4.715435 | 0.0001177 |

#### Save significant Test and Train RMSE to the results folder.

#Create a table for E. coli (cfu) as a function of Conductivity Linear Fit Model RMSE in the MIDO train data.  
MIDO\_ECFU\_Con\_RMSE <- kable(as.tibble(Ecoli\_con\_RMSE),  
 caption = "Root Mean Square Error: E. Coli CFU predicted by Conductivity in the Trained MIDO Data Set")  
  
MIDO\_ECFU\_Con\_RMSE

Root Mean Square Error: E. Coli CFU predicted by Conductivity in the Trained MIDO Data Set

|  |  |  |
| --- | --- | --- |
| .metric | .estimator | .estimate |
| rmse | standard | 1025.848 |

#Create a table for E. coli (cfu) as a function of Conductivity Linear Fit Model RMSE in the MIDO test data.  
  
MIDO\_ECFU\_Con\_RMSE\_test <- kable(as.tibble(Ecoli\_con\_RMSE),  
 caption = "Root Mean Square Error: E. Coli CFU predicted by Conductivity in the Test MIDO Data Set")  
  
MIDO\_ECFU\_Con\_RMSE\_test

Root Mean Square Error: E. Coli CFU predicted by Conductivity in the Test MIDO Data Set

|  |  |  |
| --- | --- | --- |
| .metric | .estimator | .estimate |
| rmse | standard | 1025.848 |

# Part 3: LASSO Modeling for Ideal Combination of Stream Health Indicators to Predict Biological SCore and E. Coli Colony Forming Units in MIDO Data.

# Biological Score

## Data Set Up:

*Take ID and time variables out of MIDO Data frame. This will help reduce errors when we try to run the Model.*

MIDO\_LASSO <- MIDO %>%  
 select(-"WSID", -"stream\_ID", -"month", -"year", -"day", -"pH")

*log transform E. coli data to fix bias in E. coli data*

MIDO\_LASSO <- MIDO\_LASSO %>%  
 mutate(e.coli.cfu = log10(MIDO\_LASSO$e.coli.cfu))

*Set seed: This sets a random number generator with initial (pseudo)random values set as “123”. We will need a series of random numbers created for our machine learning analysis.*

#Set random number generator.  
set.seed(123)

## Null Model For MIDO Data

*5-fold cross validation, 5 times repeated for MIDO data: Here we are setting a cross-validation of the machine learning models. Cross-validation is used to measure how the results of our machine learning models will generalize to an independent data set. As such, the folds created will be be 5 random sub-samples of the train data set to test the validity of our models within the train data set. The 5x5 structure is arbitrary.*

#Create folds for LASSO Cross Validation.  
fold\_BS <- vfold\_cv(MIDO\_LASSO, v = 5, repeats = 5, strata = biological\_score)

*Creating the recipe for Biological Score vs all predictors*

BS.recipe <-   
 recipe(biological\_score ~ ., data = MIDO\_LASSO) %>%  
 step\_zv(all\_predictors()) %>%  
 step\_normalize(all\_predictors())  
   
BS.recipe

## Recipe  
##   
## Inputs:  
##   
## role #variables  
## outcome 1  
## predictor 4  
##   
## Operations:  
##   
## Zero variance filter on all\_predictors()  
## Centering and scaling for all\_predictors()

*Setting linear regression model to assess relationship between Biological Score (outcome) and all other predictor variables.*

lm\_mod <- linear\_reg() %>% set\_engine("lm") %>% set\_mode("regression")

*However, first we need to create our null model to test against.*

### Null Model:

*Creates null model recipe. When we call this term, it will indicate in our workflow that Biological Score will be predicted by a value of 1 (NULL).*

Null\_recipe\_lm <- recipe(biological\_score ~ 1, data = MIDO\_LASSO)

*Creating the Workflow: this creates a set workflow for running a null linear regression model with biological score as the outcome.*

null\_wf <- workflow() %>% add\_model(lm\_mod) %>% add\_recipe(Null\_recipe\_lm)

*Here, I am going to fit the null model created in the above workflow to the folds made from the train data set.*

null\_lm <- fit\_resamples(null\_wf, resamples = fold\_BS)

## ! Fold1, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat5: internal: A correlation computation is required, but `estimate` is const...

*Calculate RMSE for the Null linear model.*

Null\_Met <- collect\_metrics(null\_lm)  
  
Null\_Met <- kable(Null\_Met,  
 caption = "Root Mean Squared Error and Standard Deviation for Biological Score Null Linear Model")  
  
Null\_Met

Root Mean Squared Error and Standard Deviation for Biological Score Null Linear Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| .metric | .estimator | mean | n | std\_err | .config |
| rmse | standard | 6.628388 | 25 | 0.1313342 | Preprocessor1\_Model1 |
| rsq | standard | NaN | 0 | NA | Preprocessor1\_Model1 |

*RMSE = 6.63, with a standard deviation of 0.13. This will serve as our check to test our models against latter on.*

## LASSO Model

*Specifying The Model: LASSO*

lasso\_mod <-   
 linear\_reg(penalty = tune(), mixture = 1) %>%   
 set\_engine("glmnet")

*Creating a Workflow: LASSO*

lasso\_wf <- workflow() %>%  
 add\_model(lasso\_mod) %>%  
 add\_recipe(BS.recipe)

*Create Tuning Grid: LASSO*

LASSO\_grid <- grid\_regular(penalty(), levels = 50)

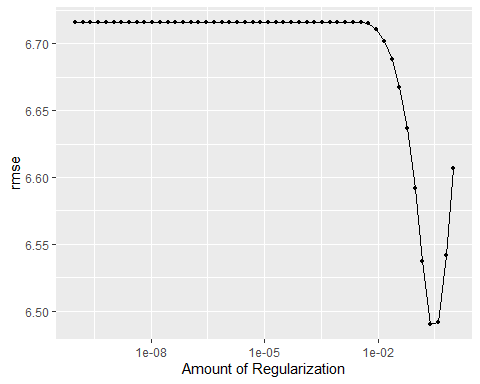
*Cross Validation with tune\_grid(): LASSO*

lasso\_resample <-   
 lasso\_wf %>%  
 tune\_grid(resamples = fold\_BS,  
 grid = LASSO\_grid,  
 control = control\_grid(verbose = FALSE, save\_pred = TRUE),  
 metrics = metric\_set(rmse))  
  
lasso\_resample %>%  
 collect\_metrics()

## # A tibble: 50 x 7  
## penalty .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 1 e-10 rmse standard 6.72 25 0.218 Preprocessor1\_Model01  
## 2 1.60e-10 rmse standard 6.72 25 0.218 Preprocessor1\_Model02  
## 3 2.56e-10 rmse standard 6.72 25 0.218 Preprocessor1\_Model03  
## 4 4.09e-10 rmse standard 6.72 25 0.218 Preprocessor1\_Model04  
## 5 6.55e-10 rmse standard 6.72 25 0.218 Preprocessor1\_Model05  
## 6 1.05e- 9 rmse standard 6.72 25 0.218 Preprocessor1\_Model06  
## 7 1.68e- 9 rmse standard 6.72 25 0.218 Preprocessor1\_Model07  
## 8 2.68e- 9 rmse standard 6.72 25 0.218 Preprocessor1\_Model08  
## 9 4.29e- 9 rmse standard 6.72 25 0.218 Preprocessor1\_Model09  
## 10 6.87e- 9 rmse standard 6.72 25 0.218 Preprocessor1\_Model10  
## # ... with 40 more rows

*Plot model performance using autoplot()*

#Plot of MIDO data LASSO model performance  
lasso\_resample %>%  
 autoplot(main="MIDO Data LASSO Performance")



*Showing and selecting best performing Models*

#Showing best performing LASSO models  
top\_five\_bs <- lasso\_resample %>%  
 show\_best()  
   
#Selects best performing model  
best\_lasso <- lasso\_resample %>%  
 select\_best()  
  
Best\_LASSO\_BS <- kable(top\_five\_bs,  
 caption = "Top 5 best performing LASSO Models for Biological Score as the Outcome")  
  
Best\_LASSO\_BS

Top 5 best performing LASSO Models for Biological Score as the Outcome

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| penalty | .metric | .estimator | mean | n | std\_err | .config |
| 0.2442053 | rmse | standard | 6.490085 | 25 | 0.1632713 | Preprocessor1\_Model47 |
| 0.3906940 | rmse | standard | 6.491326 | 25 | 0.1502093 | Preprocessor1\_Model48 |
| 0.1526418 | rmse | standard | 6.537139 | 25 | 0.1744540 | Preprocessor1\_Model46 |
| 0.6250552 | rmse | standard | 6.541565 | 25 | 0.1382506 | Preprocessor1\_Model49 |
| 0.0954095 | rmse | standard | 6.592059 | 25 | 0.1863224 | Preprocessor1\_Model45 |

*This shows that model 47 is the best performing models (RMSE = 6.50; STE = 0.16). However, it doesn’t really perform any better than the null model, making it a bad fit to the data.*

*Creating final fit based on best model permutation and plotting predicted values from that final fit model*

lasso\_final\_wf <-   
 lasso\_wf %>%   
 finalize\_workflow(best\_lasso)  
  
lasso\_final\_wf

## == Workflow ====================================================================  
## Preprocessor: Recipe  
## Model: linear\_reg()  
##   
## -- Preprocessor ----------------------------------------------------------------  
## 2 Recipe Steps  
##   
## \* step\_zv()  
## \* step\_normalize()  
##   
## -- Model -----------------------------------------------------------------------  
## Linear Regression Model Specification (regression)  
##   
## Main Arguments:  
## penalty = 0.244205309454865  
## mixture = 1  
##   
## Computational engine: glmnet

#Create workflow for fitting model to train\_data2 predictions  
lasso\_final\_fit <-   
 lasso\_final\_wf %>%  
 fit(MIDO\_LASSO)

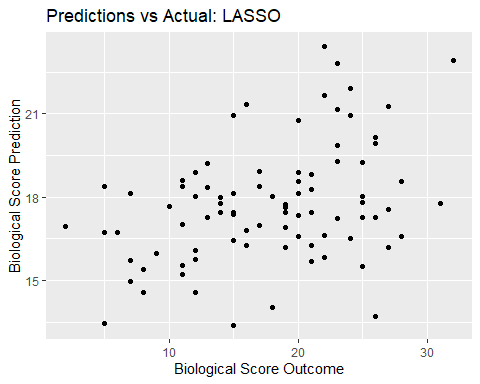
*Calculating residuals:*

#Manually calculate residuals for MIDO data LASSO models between real and predicted values.  
lasso\_residuals <- lasso\_final\_fit %>%  
 augment(MIDO\_LASSO) %>% #use augment() to make predictions from train data  
 select(c(.pred, biological\_score)) %>%  
 mutate(.resid = biological\_score - .pred) #calculate residuals and make new row.  
  
lasso\_residuals

## # A tibble: 88 x 3  
## .pred biological\_score .resid  
## <dbl> <dbl> <dbl>  
## 1 19.2 13 -6.22  
## 2 19.2 25 5.76  
## 3 17.7 10 -7.67  
## 4 20.9 15 -5.94  
## 5 15.8 12 -3.75  
## 6 21.3 16 -5.34  
## 7 14.0 18 3.96  
## 8 16.7 6 -10.7   
## 9 13.5 5 -8.47  
## 10 23.4 22 -1.43  
## # ... with 78 more rows

*model predictions from tuned model vs actual outcomes*

lasso\_pred\_plot <- ggplot(lasso\_residuals,   
 aes(x = biological\_score,   
 y = .pred)) +   
 geom\_point() +   
 labs(title = "Predictions vs Actual: LASSO",   
 x = "Biological Score Outcome",   
 y = "Biological Score Prediction")  
lasso\_pred\_plot

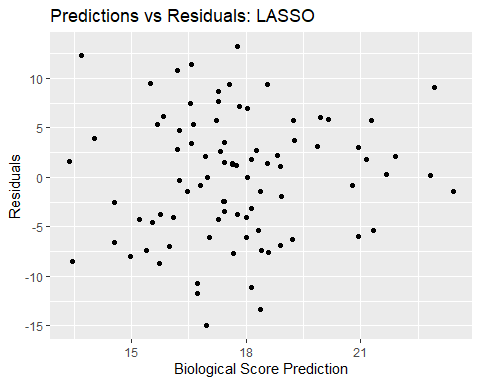


#save plot  
lasso\_pred\_plot\_BS\_location <- here::here("results", "lasso\_prediction\_plot\_BS.png")  
  
ggsave(lasso\_pred\_plot\_BS\_location)

## Saving 5 x 4 in image

*plot residuals vs predictions*

lasso\_residual\_plot <- ggplot(lasso\_residuals,   
 aes(y = .resid,   
 x = .pred)) +   
 geom\_point() +   
 labs(title = "Predictions vs Residuals: LASSO",   
 x = "Biological Score Prediction",   
 y = "Residuals")  
plot(lasso\_residual\_plot) #view plot



# LASSO: E. coli cfu

## Data Set Up:

*Set seed: This sets a random number generator with initial (pseudo)random values set as “123”. We will need a series of random numbers created for our machine learning analysis.*

#Set random number generator.  
set.seed(123)

## Null Model For MIDO Data

*5-fold cross validation, 5 times repeated for MIDO data: Here we are setting a cross-validation of the machine learning models. Cross-validation is used to measure how the results of our machine learning models will generalize to an independent data set. As such, the folds created will be be 5 random sub-samples of the train data set to test the validity of our models within the train data set. The 5x5 structure is arbitrary.*

#Create folds for LASSO Cross Validation.  
fold\_EC <- vfold\_cv(MIDO\_LASSO, v = 5, repeats = 5, strata = e.coli.cfu)

*Creating the recipe for E. coli cfu vs all predictors*

EC.recipe <-   
 recipe(e.coli.cfu ~ ., data = MIDO\_LASSO) %>%  
 step\_zv(all\_predictors()) %>%  
 step\_normalize(all\_predictors())  
   
EC.recipe

## Recipe  
##   
## Inputs:  
##   
## role #variables  
## outcome 1  
## predictor 4  
##   
## Operations:  
##   
## Zero variance filter on all\_predictors()  
## Centering and scaling for all\_predictors()

*Setting linear regression model to assess relationship between E. coli cfu (outcome) and all other predictor variables.*

lm\_mod <- linear\_reg() %>% set\_engine("lm") %>% set\_mode("regression")

*However, first we need to create our null model to test against.*

### Null Model:

*Creates null model recipe. When we call this term, it will indicate in our workflow that E. Coli cfu will be predicted by a value of 1 (NULL).*

Null\_recipe\_lm\_EC <- recipe(e.coli.cfu ~ 1, data = MIDO\_LASSO)

*Creating the Workflow: this creates a set workflow for running a null linear regression model with E. coli cfu as the outcome.*

null\_wf\_EC <- workflow() %>% add\_model(lm\_mod) %>% add\_recipe(Null\_recipe\_lm\_EC)

*Here, I am going to fit the null model created in the above workflow to the folds made from the train data set.*

null\_lm\_EC <- fit\_resamples(null\_wf\_EC, resamples = fold\_EC)

## ! Fold1, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat1: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat2: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat3: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat4: internal: A correlation computation is required, but `estimate` is const...

## ! Fold1, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold2, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold3, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold4, Repeat5: internal: A correlation computation is required, but `estimate` is const...

## ! Fold5, Repeat5: internal: A correlation computation is required, but `estimate` is const...

*Calculate RMSE for the train data linear model.*

Null\_Met\_EC <- collect\_metrics(null\_lm\_EC)  
  
  
Null\_Met\_EC <- kable(Null\_Met\_EC,  
 caption = "Root Mean Squared Error and Standard Deviation for E. coli cfu Null Linear Model")  
  
Null\_Met\_EC

Root Mean Squared Error and Standard Deviation for E. coli cfu Null Linear Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| .metric | .estimator | mean | n | std\_err | .config |
| rmse | standard | 0.5244775 | 25 | 0.0138556 | Preprocessor1\_Model1 |
| rsq | standard | NaN | 0 | NA | Preprocessor1\_Model1 |

*RMSE = 0.52, with a standard deviation of 0.014. This will serve as our check to test our models against latter on.*

## LASSO Model

*Specifying The Model: LASSO*

lasso\_mod <-   
 linear\_reg(penalty = tune(), mixture = 1) %>%   
 set\_engine("glmnet")

*Creating a Workflow: LASSO*

lasso\_wf\_EC <- workflow() %>%  
 add\_model(lasso\_mod) %>%  
 add\_recipe(EC.recipe)

*Create Tuning Grid: LASSO*

LASSO\_grid <- grid\_regular(penalty(), levels = 50)

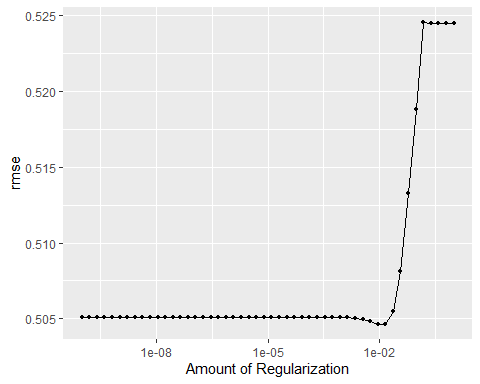
*Cross Validation with tune\_grid(): LASSO*

lasso\_resample\_EC <-   
 lasso\_wf\_EC %>%  
 tune\_grid(resamples = fold\_EC,  
 grid = LASSO\_grid,  
 control = control\_grid(verbose = FALSE, save\_pred = TRUE),  
 metrics = metric\_set(rmse))  
  
lasso\_resample\_EC %>%  
 collect\_metrics()

## # A tibble: 50 x 7  
## penalty .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 1 e-10 rmse standard 0.505 25 0.0118 Preprocessor1\_Model01  
## 2 1.60e-10 rmse standard 0.505 25 0.0118 Preprocessor1\_Model02  
## 3 2.56e-10 rmse standard 0.505 25 0.0118 Preprocessor1\_Model03  
## 4 4.09e-10 rmse standard 0.505 25 0.0118 Preprocessor1\_Model04  
## 5 6.55e-10 rmse standard 0.505 25 0.0118 Preprocessor1\_Model05  
## 6 1.05e- 9 rmse standard 0.505 25 0.0118 Preprocessor1\_Model06  
## 7 1.68e- 9 rmse standard 0.505 25 0.0118 Preprocessor1\_Model07  
## 8 2.68e- 9 rmse standard 0.505 25 0.0118 Preprocessor1\_Model08  
## 9 4.29e- 9 rmse standard 0.505 25 0.0118 Preprocessor1\_Model09  
## 10 6.87e- 9 rmse standard 0.505 25 0.0118 Preprocessor1\_Model10  
## # ... with 40 more rows

*Plot model performance using autoplot()*

#Plot of MIDO data LASSO model performance  
lasso\_resample\_EC %>%  
 autoplot()



*Showing and selecting best performing Models*

#Showing best performing LASSO models  
top\_five\_ec <- lasso\_resample\_EC %>%  
 show\_best()  
   
#Selects best performing model  
best\_lasso\_EC <- lasso\_resample\_EC %>%  
 select\_best()  
  
Best\_LASSO\_EC <- kable(top\_five\_ec,  
 caption = "Top 5 best performing LASSO Models for E. coli cfu as the Outcome")  
  
Best\_LASSO\_EC

Top 5 best performing LASSO Models for E. coli cfu as the Outcome

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| penalty | .metric | .estimator | mean | n | std\_err | .config |
| 0.0145635 | rmse | standard | 0.5046378 | 25 | 0.0122163 | Preprocessor1\_Model41 |
| 0.0091030 | rmse | standard | 0.5046412 | 25 | 0.0120305 | Preprocessor1\_Model40 |
| 0.0056899 | rmse | standard | 0.5048005 | 25 | 0.0119313 | Preprocessor1\_Model39 |
| 0.0035565 | rmse | standard | 0.5049614 | 25 | 0.0118755 | Preprocessor1\_Model38 |
| 0.0022230 | rmse | standard | 0.5050356 | 25 | 0.0118479 | Preprocessor1\_Model37 |

*This shows that model 41 is the best performing models (RMSE = 0.50; STE = 0.012). However, it doesn’t really perform any better than the null model, making it a bad fit to the data.*

*Creating final fit based on best model permutation and plotting predicted values from that final fit model*

lasso\_final\_wf\_EC <-   
 lasso\_wf\_EC %>%   
 finalize\_workflow(best\_lasso\_EC)  
  
lasso\_final\_wf\_EC

## == Workflow ====================================================================  
## Preprocessor: Recipe  
## Model: linear\_reg()  
##   
## -- Preprocessor ----------------------------------------------------------------  
## 2 Recipe Steps  
##   
## \* step\_zv()  
## \* step\_normalize()  
##   
## -- Model -----------------------------------------------------------------------  
## Linear Regression Model Specification (regression)  
##   
## Main Arguments:  
## penalty = 0.0145634847750124  
## mixture = 1  
##   
## Computational engine: glmnet

#Create workflow for fitting model to train\_data2 predictions  
lasso\_final\_fit\_EC <-   
 lasso\_final\_wf\_EC %>%  
 fit(MIDO\_LASSO)

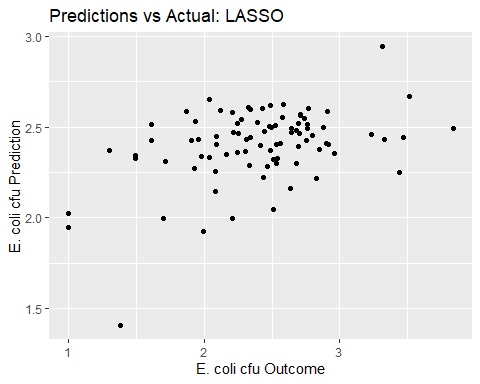
*Calculating residuals:*

#Manually calculate residuals for MIDO data LASSO models between real and predicted values.  
lasso\_residuals\_EC <- lasso\_final\_fit\_EC %>%  
 augment(MIDO\_LASSO) %>% #use augment() to make predictions from train data  
 select(c(.pred, e.coli.cfu)) %>%  
 mutate(.resid = e.coli.cfu - .pred) #calculate residuals and make new row.  
  
lasso\_residuals\_EC

## # A tibble: 88 x 3  
## .pred e.coli.cfu .resid  
## <dbl> <dbl> <dbl>  
## 1 2.53 1.93 -0.597  
## 2 2.27 1.93 -0.344  
## 3 2.60 2.43 -0.172  
## 4 2.51 1.61 -0.900  
## 5 2.43 2.75 0.322  
## 6 2.59 1.87 -0.719  
## 7 2.49 3.84 1.35   
## 8 2.60 2.77 0.161  
## 9 2.43 3.33 0.899  
## 10 2.37 1.30 -1.07   
## # ... with 78 more rows

*model predictions from tuned model vs actual outcomes*

lasso\_pred\_plot\_EC <- ggplot(lasso\_residuals\_EC,   
 aes(x = e.coli.cfu,   
 y = .pred)) +   
 geom\_point() +   
 labs(title = "Predictions vs Actual: LASSO",   
 x = "E. coli cfu Outcome",   
 y = "E. coli cfu Prediction")  
lasso\_pred\_plot\_EC



#save plot  
lasso\_pred\_plot\_EC\_location <- here::here("results", "lasso\_prediction\_plot\_EC.png")  
  
ggsave(lasso\_pred\_plot\_EC\_location)

## Saving 5 x 4 in image

*plot residuals vs predictions*

lasso\_residual\_plot\_EC <- ggplot(lasso\_residuals\_EC,   
 aes(y = .resid,   
 x = .pred)) +   
 geom\_point() +   
 labs(title = "Predictions vs Residuals: LASSO",   
 x = "E. coli cfu Prediction",   
 y = "Residuals")  
plot(lasso\_residual\_plot\_EC) #view plot

