BLE\_Microbio1\_BackgroundPrep

Zach Brown

2024-02-22

#housekeeping code and calling things in  
library(readr)  
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.3 ✔ purrr 1.0.2  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.4 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggplot2)  
library(readxl)  
  
library(lubridate)  
library(vegan)

## Loading required package: permute  
## Loading required package: lattice  
## This is vegan 2.6-4

library(plotly) #couldn't load

##   
## Attaching package: 'plotly'  
##   
## The following object is masked from 'package:ggplot2':  
##   
## last\_plot  
##   
## The following object is masked from 'package:stats':  
##   
## filter  
##   
## The following object is masked from 'package:graphics':  
##   
## layout

library(goeveg) #couldn't load

## This is GoeVeg 0.7.2 - build: 2024-02-06

library(patchwork)  
  
#######required libraries#######################  
library(randomForest)

## randomForest 4.7-1.1  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:dplyr':  
##   
## combine  
##   
## The following object is masked from 'package:ggplot2':  
##   
## margin

library(caret)

##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:vegan':  
##   
## tolerance  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(e1071)  
library(dplyr)  
  
  
setwd("/Users/zachbrown/Desktop/SIO/Classes/2023-24/Winter2024/SIOB278\_MarineMicrobialSeminar/BLE\_LTER\_Data")  
  
  
BLE\_ASVs\_csv <- read\_csv("~/Desktop/SIO/Classes/2023-24/Winter2024/SIOB278\_MarineMicrobialSeminar/GITHUB/BLE/BLE\_ASVs.csv.gz")

## New names:  
## Rows: 73582 Columns: 651  
## ── Column specification  
## ──────────────────────────────────────────────────────── Delimiter: "," chr  
## (1): ...1 dbl (650): SRR8832739, SRR8832740, SRR8832741, SRR8832742,  
## SRR8832743, SRR88...  
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ  
## Specify the column types or set `show\_col\_types = FALSE` to quiet this message.  
## • `` -> `...1`

BLE\_ASVs\_flipped <- t(BLE\_ASVs\_csv)  
METADATA\_csv <- read\_csv("~/Desktop/SIO/Classes/2023-24/Winter2024/SIOB278\_MarineMicrobialSeminar/GITHUB/BLE/FINAL\_METADATA.csv")

## Rows: 466 Columns: 23  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (5): Run, Station, CollectionDate, Depth(m), Min\_Filter\_Size  
## dbl (18): 16s\_18s, Temp(degC), Salinity, DO(%), DO(mg/L), pH, Chl(ug/L), H2O...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

#change headers in ASV columns  
new\_headers1 <- as.character(BLE\_ASVs\_flipped[1,])  
BLE\_ASVs\_flipped2 <- BLE\_ASVs\_flipped[-1, ]  
colnames(BLE\_ASVs\_flipped2) <- new\_headers1  
  
#add in new column and insert row names  
BLE\_ASVs\_flippedfinal <- cbind(Run = rownames(BLE\_ASVs\_flipped2), BLE\_ASVs\_flipped2)  
  
#merge data  
commoncol2 = "Run"  
MergedData2 <- merge(METADATA\_csv, BLE\_ASVs\_flippedfinal, by = commoncol2)

#check histograms with env data on merged stuff  
#DataToCheck <- c("Temp(degC)", "Salinity", "DO(%)", "DO(mg/L)", "pH", "Chl(ug/L)")  
#DataToCheck2 <- colnames(MergedData2)[4:23]  
  
#for (col in DataToCheck2){  
 #if (is.numeric(MergedData2[[col]])){  
 # hist(MergedData2[[col]], main = paste("Histogram of", col), xlab = col, col = "lightblue", border = "black")  
 # } else{  
 # cat("Skipping", col, "as it is not number. \n")  
 # }  
#}

myEnvData <- c("16s\_18s", "Temp(degC)", "Salinity", "DO(%)", "DO(mg/L)", "pH", "Chl(ug/L)", "H2O-δ18O(‰)", "DOC(mg/L)", "TDN(mg/L)", "NO3(μM)", "NH4(μM)", "Bacteriax108cells/L", "POC(μg/L)", "POC-δ13C(‰)", "PN(μg/L)", "PN-δ15N(‰)", "chlorophyll\_a\_(μg/L)")  
EnvData <- MergedData2[,which(colnames(MergedData2) %in% c(myEnvData, "sample\_title", "subject"))]  
  
  
EnvData.mat <- as.matrix(EnvData[,c(1:18)])  
EnvData.mat <- na.omit(EnvData.mat)  
  
#BLE <- metaMDS(EnvData.mat, distance = "bray", k = 2) #what should k value be - tried 18, 3, and 2  
#stressplot(BLE)  
  
#BLE.nmds <- as.data.frame(BLE$points) #Using the scores function from vegan to extract the site scores and convert to a data.frame  
#BLE.nmds <- cbind(BLE.nmds, na.omit(MergedData2))  
  
#ggplot(data = BLE.nmds) +  
 #geom\_point(aes(x = MDS1, y = MDS2)) +  
 #theme\_bw()

# ASVs  
  
MergedData3 <- MergedData2[,c(25:ncol(MergedData2)-1)] # only numeric ASV data  
  
# asv.df.during <- mutate\_all(asv.df.during, function(x) as.numeric(as.character(x)))  
MergedData3 <- na.omit(MergedData3) #omit all NA that might be present in the data frame  
  
MergedData4 <- apply(MergedData3, MARGIN = 2, FUN = as.numeric) #convert data frame from character values to numerics  
str(MergedData4) #confirm dataframe structure

## num [1:466, 1:73581] 0 0 0 0 0 0 0 0 0 0 ...  
## - attr(\*, "dimnames")=List of 2  
## ..$ : NULL  
## ..$ : chr [1:73581] "CCGTTCGGTACACACCGCCCGTCGCTACTACCGATTGAATGTCATAGTGAGGTCCTCGGACTGTTTGGTTGGCGGATCACTCTGCCTGCCTGGCGGGAAGACGACCAAACT"| \_\_truncated\_\_ "CCGTTCGGTACACACCGCCCGTCGCTACTACCGATTGAGTGTTTTAGTGAGGTCCTCGGATTGCTTGTCTGGCGGTTAACGCTGCCTAGCCGGCGAAAAGACGACCAAACT"| \_\_truncated\_\_ "CCGTTCGGTACACACCGCCCGTCGCACCTACCGATTGAGTGGTCCGGTGAACCCTCGAGATTGTGATTAATTTCTTTTATTAGAATTTGATTGTGAGAACTTGTGTAAACC"| \_\_truncated\_\_ "GCCGTTCGGTACACACCGCCCGTCGCACCTACCGATTGAGTGGTCCGGTGAAGACTCGGGATTGTGGCAATCTACTTTATTGCAGTTTGTCCACGAGAACCTGTCTAAACC"| \_\_truncated\_\_ ...

#potential data modification steps, think we can omit for now, not sure why they included --> incorporated to see if it would influence insufficient data error, it didn't so changed back  
#MergedData4 <- MergedData4[which(rowSums(MergedData4) > 4000),]  
#MergedData4 <- MergedData4[,which(colSums(MergedData4) > 10)] # remove ASVs with less than 10 reads across all samples  
  
#This is all the example of NMDS plot for ASV that did not work - kept in for records  
  
#BLEASVs.mat <- as.matrix(MergedData4) #make the dataframe a matrix  
#BLEASV <- metaMDS(BLEASVs.mat, distance = "bray", k = 2, trymax = 50) #run the nonmetric multidimensional scaling analysis  
#stressplot(BLEASV)  
#BLEASV.nmds <- as.data.frame(BLEASV$points) #Using the scores function from vegan to extract the site scores and convert to a data.frame  
#BLEASV.nmds <- cbind(BLEASV.nmds, na.omit(MergedData4))  
#ggplot(data = BLEASV.nmds) +  
 #geom\_point(aes(x = MDS1, y = MDS2)) +  
 #theme\_bw()  
  
#this is bad - things would likely look better if we separated 16s and 18s entirely -- that is done in the next chunk

#separate the 16s and 18s data  
MergedData16s <- MergedData2[MergedData2$'16s\_18s' == "16", ]  
MergedData18s <- MergedData2[MergedData2$'16s\_18s' == "18", ]  
  
#separate each 16s and 18s by filter size as well - probably won't use but just in case we want in the future  
#MergedData16s3um142 <- MergedData16s[MergedData16s$Min\_Filter\_Size == "142 mm 3 µm polycarbonate filter", ]  
#MergedData16s022um142 <- MergedData16s[MergedData16s$Min\_Filter\_Size == "142 mm 0.22 µm Supor filter", ]  
#MergedData16s022um <- MergedData16s[MergedData16s$Min\_Filter\_Size == "0.22 µm (Sterivex-GP)", ]  
#MergedData16s3um47 <- MergedData16s[MergedData16s$Min\_Filter\_Size == "47 mm 3 µm polycarbonate filter", ]  
#MergedData18s3um142 <- MergedData18s[MergedData18s$Min\_Filter\_Size == "142 mm 3 µm polycarbonate filter", ]  
#MergedData18s022um142 <- MergedData18s[MergedData18s$Min\_Filter\_Size == "142 mm 0.22 µm Supor filter", ]  
#MergedData18s022um <- MergedData18s[MergedData18s$Min\_Filter\_Size == "0.22 µm (Sterivex-GP)", ]  
#MergedData18s3um47 <- MergedData18s[MergedData18s$Min\_Filter\_Size == "47 mm 3 µm polycarbonate filter", ]  
  
#make ASV dataframes for 16s and 18s data, with numeric structure instead of characters  
#start with 16s  
MergedData16s2 <- MergedData16s[,c(25:ncol(MergedData16s)-1)] # only numeric ASV data  
MergedData16s2 <- na.omit(MergedData16s2) #omit all NA that might be present in the data frame  
MergedData16s3 <- apply(MergedData16s2, MARGIN = 2, FUN = as.numeric) #convert data frame from character values to numerics  
  
MergedData18s2 <- MergedData18s[,c(25:ncol(MergedData18s)-1)] # only numeric ASV data  
MergedData18s2 <- na.omit(MergedData18s2) #omit all NA that might be present in the data frame  
MergedData18s3 <- apply(MergedData18s2, MARGIN = 2, FUN = as.numeric) #convert data frame from character values to numerics  
  
#Check the ASV read numbers with colSums  
ASVreads16s <- colSums(MergedData16s3[, 25:ncol(MergedData16s3)])  
ASVreads18s <- colSums(MergedData18s3[, 25:ncol(MergedData18s3)])  
  
#visualize - commented out to make knitting easier  
  
#try 16s and 18s stressplots individually  
#BLE16sASVs.mat <- as.matrix(MergedData16s3) #make the dataframe a matrix  
##BLEASV16s <- metaMDS(BLE16sASVs.mat, distance = "bray", k = 3, trymax = 50) #run the nonmetric multidimensional scaling analysis  
#stressplot(BLEASV16s)  
#BLE16sASV.nmds <- as.data.frame(BLEASV16s$points) #Using the scores function from vegan to extract the site scores and convert to a data.frame  
#BLE16sASV.nmds <- cbind(BLE16sASV.nmds, na.omit(MergedData16s3))  
#ggplot(data = BLE16sASV.nmds) +  
# geom\_point(aes(x = MDS1, y = MDS2)) +  
# theme\_bw()

#commented out to make knitting easier  
  
#BLE18sASVs.mat <- as.matrix(MergedData18s3) #make the dataframe a matrix  
#BLEASV18s <- metaMDS(BLE18sASVs.mat, distance = "bray", k = 3, trymax = 50) #run the nonmetric multidimensional scaling analysis  
#stressplot(BLEASV18s)  
#BLE18sASV.nmds <- as.data.frame(BLEASV18s$points) #Using the scores function from vegan to extract the site scores and convert to a data.frame  
#BLE18sASV.nmds <- cbind(BLE18sASV.nmds, na.omit(MergedData18s3))  
#ggplot(data = BLE18sASV.nmds) +  
 #geom\_point(aes(x = MDS1, y = MDS2)) +  
 #theme\_bw()

#Attempt 1 RF - regression

#Should we divide up by filter size as well?  
#Start with 16s data, RF\_classification template  
library(randomForest)  
library(caret)  
library(e1071)  
library(dplyr)

Work with MergedData16sSalinity3 for RF. This chunk discusses how I tailored the dataset to meet the RF requirements, starting from confirming numeric data to converting to relative abundance to pulling the ASV and one env data predicted value (salinity) and making sure the data frame was of an okay size.

#make sure all data is numeric  
  
#pull out the environmental metadata  
#MergedData16sEnv <- MergedData16s[, 1:23, drop = FALSE]  
#combine env data and numeric 16s ASV data  
#MergedData16s4 <- cbind(MergedData16sEnv, MergedData16s3)  
#pull all numeric environmental and ASV data  
#MergedData16s5 <- MergedData16s4[, 7:ncol(MergedData16s4), drop = FALSE]  
##remove any columns with 0 value -- cut out 40k columns and environmental data  
#MergedData16sRemoved <- MergedData16s5[, which(colSums(MergedData16s5) != 0)]  
#create a new dataframe that has just numeric and environmental data  
#MergedData16s6 <- cbind(MergedData16sEnv, MergedData16sRemoved)  
  
#MergedData16s6Trimmed <- MergedData16s6[, 1:15000, drop = FALSE] # just arbitrarily trimmed down the amount of ASV data we were using because of protection stack overflow error - which meant we had too much data to work with  
  
#IGNORE THIS - had to deal with NA error, potentially - trimmed from 233 to 222 runs for this particular variable as factor  
#MergedData16s6TrimmedSalNA <- na.omit(MergedData16s6Trimmed[, c('Salinity')])  
#MergedData16s6TrimmedSalNAdf <- MergedData16s6Trimmed[!is.na(MergedData16s6Trimmed[, 'Salinity']), ]  
  
#make a data frame with just salinity and the trimmed non-zero col ASV data  
#MergedData16sSalinity <- MergedData16s6Trimmed[, c(8, 24:15000), drop = FALSE]  
#MergedData16sSalinity2 <- MergedData16sSalinity[!is.na(MergedData16sSalinity[, 'Salinity']), ] #removed the NA values from mergeddata16ssalinity  
  
#convert community structure data to relative abundance, divide each row by sum of row (divide df by rowsum vector)  
#RelativeAbundanceASV1 <- MergedData16sSalinity2[, 2:14978, drop = FALSE] #create a dataframe that has the ASV data  
#Salinity <- MergedData16sSalinity2[,1] #pull the salinity data from the ASV data  
#Rowsums\_RelAbASV1 <- rowSums(RelativeAbundanceASV1) #create a vector with the sums of the ASV counts  
#RelativeAbundance16sASV2 <- RelativeAbundanceASV1/Rowsums\_RelAbASV1 #convert from counts to relative abundance  
#MergedData16sSalinity3 <- cbind(Salinity, RelativeAbundance16sASV2) #combine salinity with ASV relative abundance, finally have a dataset we can push through the random forest pipeline  
  
#relative abundances of all 15k ASVs (15k arbitrarily chosen) - created to make starting new models easier  
#RelativeAbundanceASV3 <- MergedData16s6Trimmed[, c(24:15000), drop=FALSE]  
#Rowsums\_RelAbASV3 <- rowSums(RelativeAbundanceASV3)  
#RelativeAbundance16sASV4 <- RelativeAbundanceASV3/Rowsums\_RelAbASV3  
  
#EnvDataWith15kRelativeASVAbundances <- cbind(MergedData16sEnv, RelativeAbundance16sASV4)  
#write.csv(EnvDataWith15kRelativeASVAbundances, "Env\_ASVData.csv")  
  
#EnvASVDataFlipped <- t(EnvDataWith15kRelativeASVAbundances)  
#write.csv(EnvASVDataFlipped, "EnvASVData2.csv")  
  
  
#MergedData16sXEnvCondition <- MergedData16s6Trimmed[, c(9, 24:15000), drop = FALSE]  
  
  
##saving the final dataset that we work with  
#write.csv(MergedData16sSalinity3, "MergedDataSalinity3.csv")  
  
  
#Set a dataframe as the dataframe we want to put through random forest  
#Data16sTry1 <- MergedData16sSalinity3  
#str(MergedData16sSalinity3)  
  
# Data Partition in 70% training dataset and 30% test dataset - directly from code  
#set.seed(123)  
#ind <- sample(2, nrow(Data16sTry1), replace = TRUE, prob = c(0.7, 0.3))  
#train16sSalinity <- Data16sTry1[ind==1,]  
#test16sSalinity <- Data16sTry1[ind==2,]  
  
#running rf - taken directly from code  
#set.seed(4444)  
#rf16sSalinity1 <- randomForest(Salinity~., #looking at phase while compared to all the columns so dot (.) is given after tilda (~)  
# data=train16sSalinity, #train data is used to train  
## ntree = 200, #number of trees to run  
# importance = TRUE, #evaluates importance of a predictor  
# proximity = TRUE) #calculates the proximity measure among the rows  
  
#getting protection stack overflow error when I run the above stuff - resolved by halving  
#now getting na.fail.default missing values in object error - removed na, but I think had to remove NA from all other env variables?  
  
#plotting error-shows the error with growing number of trees in a forest  
#plot(rf16sSalinity1)  
  
#saving the parameters used in running RF  
#sink(paste0("RF\_parameter",".txt"))  
#rf16sSalinity1  
#sink()  
  
#predicting the dependent variable in the training set using generated RF model  
#p1 <- predict(rf16sSalinity1, train16sSalinity)  
#write.csv(p1,paste0("trainingSalinity16s",".csv"))  
  
#predicting the dependent variable in the validation set using generated RF model  
#p2 <- predict(rf16sSalinity1, test16sSalinity)  
#write.csv(p2,paste0("validationSalinity16s",".csv"))

#make sure all data is numeric  
  
#pull out the environmental metadata  
#MergedData18sEnv <- MergedData18s[, 1:23, drop = FALSE]  
#combine env data and numeric 16s ASV data  
#MergedData18s4 <- cbind(MergedData18sEnv, MergedData18s3)  
#pull all numeric environmental and ASV data  
#MergedData18s5 <- MergedData18s4[, 7:ncol(MergedData18s4), drop = FALSE]  
#remove any columns with 0 value -- cut out 40k columns and environmental data  
#MergedData18sRemoved <- MergedData18s5[, which(colSums(MergedData18s5) != 0)]  
#create a new dataframe that has just numeric and environmental data  
#MergedData18s6 <- cbind(MergedData18sEnv, MergedData18sRemoved)  
  
#MergedData16s6Trimmed <- MergedData16s6[, 1:15000, drop = FALSE] # not needed because only 14145 columns  
  
#IGNORE THIS - had to deal with NA error, potentially - trimmed from 233 to 222 runs for this particular variable as factor  
#MergedData16s6TrimmedSalNA <- na.omit(MergedData16s6Trimmed[, c('Salinity')])  
#MergedData16s6TrimmedSalNAdf <- MergedData16s6Trimmed[!is.na(MergedData16s6Trimmed[, 'Salinity']), ]  
  
#make a data frame with just salinity and the trimmed non-zero col ASV data  
#MergedData18sSalinity <- MergedData18s6[, c(8, 24:14145), drop = FALSE]  
#MergedData18sSalinity2 <- MergedData18sSalinity[!is.na(MergedData18sSalinity[, 'Salinity']), ] #removed the NA values from mergeddata16ssalinity  
  
#convert community structure data to relative abundance, divide each row by sum of row (divide df by rowsum vector)  
#RelativeAbundance18ASV1 <- MergedData18sSalinity2[, 2:14123, drop = FALSE] #create a dataframe that has the ASV data  
#Salinity18 <- MergedData18sSalinity2[,1] #pull the salinity data from the ASV data  
#Rowsums\_RelAb18ASV1 <- rowSums(RelativeAbundance18ASV1) #create a vector with the sums of the ASV counts  
#RelativeAbundance18sASV2 <- RelativeAbundance18ASV1/Rowsums\_RelAb18ASV1 #convert from counts to relative abundance  
#MergedData18sSalinity3 <- cbind(Salinity18, RelativeAbundance18sASV2) #combine salinity with ASV relative abundance, finally have a dataset we can push through the random forest pipeline  
  
#relative abundances of all 15k ASVs (15k arbitrarily chosen) - created to make starting new models easier  
#RelativeAbundance18ASV3 <- MergedData18s6[, c(24:14145), drop=FALSE]  
#Rowsums\_RelAb18ASV3 <- rowSums(RelativeAbundance18ASV3)  
#RelativeAbundance18sASV4 <- RelativeAbundance18ASV3/Rowsums\_RelAb18ASV3  
  
#EnvDataWith15kRelativeASVAbundances18 <- cbind(MergedData18sEnv, RelativeAbundance18sASV4)  
#write.csv(EnvDataWith15kRelativeASVAbundances18, "Env\_ASVData18.csv")  
  
#EnvASVData18Flipped <- t(EnvDataWith15kRelativeASVAbundances18)  
#write.csv(EnvASVData18Flipped, "EnvASV18Data2.csv")  
  
#saving the final dataset that we work with  
#twrite.csv(MergedData16sSalinity3, "MergedDataSalinity3.csv")

#read in data  
#data <- MergedData16sSalinity3 #define the dataframe we used for training  
#data2 <- read.csv(paste0("trainingSalinity16s",".csv")) #pull in data from RF  
#colnames(data2) <- c("SampleNumber", "ActualValue") #and rename it  
  
#pull real data for samples that line up with the training dataset  
#df2 <- data %>% select(1, Salinity) #pull all salinity values  
#colnames(df2) <- c("SampleNumber", "ActualValue") #change the column names  
#df2 <- data.frame(blank\_column = NA, df2) #create a blank column aligning with salinity values  
#df2$blank\_column <- 1:222 #label blank column with the number of rows there are   
  
#call in the the trained data with the predicted values  
  
#tr <- read.csv(paste0("trainingSalinity16s",".csv")) #call in the csv saved at the end of the last chunk  
#colnames(tr) <- c("SampleNumber", "PredictedValue") #rename the column names  
#trainingdatapoints <- tr[,1] #make a vector that has the salinity values that were predicted in the above random forest model  
  
#From the training data, pull the associated   
#indices <- match(trainingdatapoints, df2$blank\_column) #make a list of the row numbers that match with the trained data  
#df2\_2 <- df2[indices, ] #make a new dataframe that pulls the correct rows out of the dataframe that had all of the real data  
##df2\_2 <- df2\_2[1:155, ]#remove any NA rows  
#colnames(df2\_2) <- c("SampleNumber", "ActualValue")  
  
#Finally merge the training data and the real values into a new dataframe by the sample number  
#total1 <- merge(tr,df2\_2,by="SampleNumber")  
#write.csv(total1,"predicted\_vs\_actual\_in\_training.csv", row.names = FALSE) #save the dataframe as a CSV  
  
  
#plotting predicted vs actual value  
  
#pdf(paste0("Training\_plot",".pdf"),8,4) - don't need to run this I don't think  
#ggplot(total1, aes(y=ActualValue, x=PredictedValue)) + geom\_point() + geom\_smooth(method=lm)+  
 # ggtitle(paste0("Model RM"," training plot")) +  
 # xlab("Predicted Salinity") + ylab("Actual Salinity")   
#dev.off() - also don't need to run this  
  
#running linear model for training - did not change any of this from the example  
#fit <- lm(ActualValue ~ PredictedValue, data = total1)  
#sink(paste0("training","\_accuracy.txt"))  
#print(summary(fit))  
#sink()

#pr <- read.csv(paste0("validationSalinity16s",".csv"))  
#colnames(pr) <- c("SampleNumber", "PredictedValue")  
#validationdatapoints <- pr[,1] #make a vector that has the salinity values that were predicted in the above random forest model  
#From the training data, pull the associated   
#indices2 <- match(validationdatapoints, df2$blank\_column) #make a list of the row numbers that match with the trained data  
#df2\_3 <- df2[indices2, ] #make a new dataframe that pulls the correct rows out of the dataframe that had all of the real data  
#df2\_3 <- df2\_3[1:57, ]#remove any NA rows  
#colnames(df2\_3) <- c("SampleNumber", "ActualValue")  
  
#total2 <- merge(pr,df2\_3,by="SampleNumber")  
#write.csv(total2,"predicted\_vs\_actual\_in\_validation.csv", row.names = FALSE)  
  
#plotting predicted vs actual value  
  
#pdf(paste0("Validation\_plot",x,y,z,".pdf"),8,4)  
#ggplot(total2, aes(y=ActualValue, x=PredictedValue)) + geom\_point() + geom\_smooth(method=lm)+   
 #ggtitle(paste0("Model RM"," validation plot")) +  
 # xlab("Predicted Salinity") + ylab("Actual Salinity")   
#dev.off()  
  
#running linear model for validation  
#fit2 <- lm(ActualValue ~ PredictedValue, data = total2)  
  
#sink(paste0("validation","\_accuracy.txt"))  
#print(summary(fit2))  
#sink()