CYO: Ecoli Classification - Data Science Capstone Harvardx

Angelita Elinon Yu

2022-07-12

EXECUTIVE SUMMARY

The project demonstrates the development and use of cross-validated K-Nearest Neighbours (CV KNN) and Random Forest machine learning algorithms to predict the localization site of E.coli proteins based on seven predictor scores.

In the first attempt to use the two models with the test data set, both models stopped working due to the same error: new levels of predictor variables were detected in the test data set not found in the train data set.

The project resolved this error by assigning "NA" to the new levels and re-ran the two models. The two models proceeded to make their predictions. However, their predictions were short of 23 instances. They only made 47 predictions but the test data set has 70 observations. Their resulting accuracies were very poor and below the baseline accuracy.

The missing 23 predictions correspond to the 23 observations in the test data set that had one or more "NA" values. The project removed the 23 observations in the test data set and re-ran the two models. Their accuracy improved significantly as there were as many predictions as there were observations in the test data set, i.e., 47.

The limitations of the project arise from the unstructured continuous values in the E.coli data set and the use of two-dimensional histograms for visualisation.

Recommendations in the Conclusion explain how to overcome these limitations, namely: structure the data as in a Likert Scale, use scatterplots, and collect more E. coli data.

INTRODUCTION

In 1996, Paul Horton and Kenta Nakai published "A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins" that described their simple model of classification which combines human-provided expert knowledge with probabilitic reasoning. Using an objective cross-validation model, without hand tuning the system to learn training data, Horton and Nakai classified 336 E.coli proteins into 8 classes with an accuracy of 81% and 1484 yeast proteins into 10 classes with an accuracy of 55%.

The Ecoli Data Set was shared by Horton and Nakai with UCI Machine Learning Repository (Center for Machine Learning and Intelligent Systems). This is the dataset used by the author for the project on hand.

Unfortunately, there are supposed to be 336 instances of observations but the author's repeated examination of the downloaded data set shows only 335 instances. The missing instance, however, is unlikely to significantly degrade the quality of the author's analysis and findings.

DATA PREPARATION

Before the Ecoli dataset can be downloaded and prepared for this project, it is important to install the packages and libraries required.

```
# Install required packages and libraries.
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

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

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

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

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(stringr))
install.packages("stringr", repos = "http://cran.us.r-project.org")
install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/aelin/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## package 'randomForest' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'randomForest'

## Warning in file.copy(savedcopy, lib, recursive =
## TRUE): problem copying C:\Users\aelin\Documents\R\win-
## library\4.1\00LOCK\randomForest\libs\x64\randomForest.dll to C:
## \Users\aelin\Documents\R\win-library\4.1\randomForest\libs\x64\randomForest.dll:
## Permission denied

## Warning: restored 'randomForest'

##
## The downloaded binary packages are in
##  C:\Users\aelin\AppData\Local\Temp\RtmpqSr5iD\downloaded_packages
```

```
library(tidyverse)
library(caret)
library(data.table)
library(stringr)
```

```
# Download the Ecoli dataset
ecolidata <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/ecoli.data"))
nrow(ecolidata)
```

```
## [1] 335
```

```
#The downloaded Ecoli dataset is already a dataframe with 335 instances and one column only.
dim(ecolidata)
```

```
## [1] 335   1
```

```
class(ecolidata)
```

```
## [1] "data.frame"
```

```
head (ecolidata)
```

```
##   AAT_ECOLI...0.49..0.29..0.48..0.50..0.56..0.24..0.35...cp
## 1 ACEA_ECOLI  0.07  0.40  0.48  0.50  0.54  0.35  0.44   cp
## 2 ACEK_ECOLI  0.56  0.40  0.48  0.50  0.49  0.37  0.46   cp
## 3 ACKA_ECOLI  0.59  0.49  0.48  0.50  0.52  0.45  0.36   cp
## 4 ADI_ECOLI   0.23  0.32  0.48  0.50  0.55  0.25  0.35   cp
## 5 ALKH_ECOLI  0.67  0.39  0.48  0.50  0.36  0.38  0.46   cp
## 6 AMPD_ECOLI  0.29  0.28  0.48  0.50  0.44  0.23  0.34   cp
```

```
#There is a need to parse the contents of the one column to bring out the attributes
#- eight predictors and one predicted value.

#To do this, the str_split_fixed function is used from the stringr package

#But the one column needs to be named in order to address it.
#The one column is named "ecolivector" for this purpose.
colnames(ecolidata) <- c("ecolivector")

#Parse ecolivector
ecolisplit<-str_split_fixed(ecolidata$ecolivector, "  ", 9)

#After parsing or splitting the column, the dataset still has 335 instances but with 9 columns.
dim(ecolisplit)
```

```
## [1] 335   9
```

```
#The parsed dataset, however, has now become a matrix with each row considered an array.
class(ecolisplit)
```

```
## [1] "matrix" "array"
```

```
#The parsed dataset is transformed back to a dataframe as follows.  It has 335 rows and 9 columns.
ecolidataframe<- as.data.frame(ecolisplit)
dim(ecolidataframe)
```

```
## [1] 335   9
```

```
class(ecolidataframe)
```

```
## [1] "data.frame"
```

```
#The columns have no names and are by default generically referred to as V1 to V9.
head(ecolidataframe)
```

```
##           V1    V2   V3   V4   V5   V6   V7   V8  V9
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44  cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46  cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36  cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35  cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46  cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34  cp
```

```
#The column names are assigned as follows:
colnames(ecolidataframe) <- c("seqname", "mcg", "gvh", "lip", "chg", "aac", "alm1", "alm2", "locsite")

#The dataset that has now been transformed back to a dataframe still has 335 rows and 9 columns.
#But this time, the columns are properly named.
dim(ecolidataframe)
```

```
## [1] 335   9
```

```
class(ecolidataframe)
```

```
## [1] "data.frame"
```

```
head(ecolidataframe)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

```
str(ecolidataframe)
```

```
## 'data.frame':    335 obs. of  9 variables:
##  $ seqname: chr  "ACEA_ECOLI" "ACEK_ECOLI" "ACKA_ECOLI" "ADI_ECOLI" ...
##  $ mcg    : chr  "0.07" "0.56" "0.59" " 0.23" ...
##  $ gvh    : chr  "0.40" "0.40" "0.49" "0.32" ...
##  $ lip    : chr  "0.48" "0.48" "0.48" "0.48" ...
##  $ chg    : chr  "0.50" "0.50" "0.50" "0.50" ...
##  $ aac    : chr  "0.54" "0.49" "0.52" "0.55" ...
##  $ alm1   : chr  "0.35" "0.37" "0.45" "0.25" ...
##  $ alm2   : chr  "0.44" "0.46" "0.36" "0.35" ...
##  $ locsite: chr  " cp" " cp" " cp" " cp" ...
```

DATA PROFILE

Predicted and Predictor Variables

The predicted variable is the locsite (localization_site). There are eight possible locsites:

1. cp (cytoplasm)
2. im (inner membrane without signal sequence)
3. pp (perisplasm)
4. imU (inner membrane, uncleavable signal sequence)
5. om (outer membrane)
6. omL (outer membrane lipoprotein)
7. imL (inner membrane lipoprotein)
8. imS (inner membrane, cleavable signal sequence)

The eight attributes that may be used as predictors are:

1. seqname (Sequence Name): accession number for the SWISS-PROT database
2. mcg: McGeoch's method for signal sequence recognition
3. gvh: von Heijne's method for signal sequence recognition
4. lip: von Heijne's Signal Peptidase II consensus sequence score; binary attribute
5. chg: Presence of charge on N-terminus of predicted lipoproteins; binary attribute
6. aac: score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins
7. alm1: score of the ALOM membrane spanning region prediction program
8. alm2: score of ALOM program after excluding putative cleavable signal regions from the sequence

Data Issues

An inspection of the distribution of the possible locsites show that the data set requires further cleaning. Remaining data issues are:

1. One instance had an alm2 reading before it, e.g. "0.39 cp". This occurred in instance 47.
2. There is a space in front of some of them, e.g. " cp", " im", " om", " pp".

```
ecolidataframe[47,]
```

```
##     seqname mcg  gvh  lip  chg  aac alm1 alm2   locsite
## 47 GT_ECOLI     0.43 0.40 0.48 0.50 0.39 0.28 0.39   cp
```

```
ecolidataframe %>% group_by(locsite) %>% summarize(count=n())
```

```
## # A tibble: 9 x 2
##   locsite     count
##   <chr>       <int>
## 1 " cp"         141
## 2 " im"          77
## 3 " om"          20
## 4 " pp"          52
## 5 "0.39   cp"     1
## 6 "imL"           2
## 7 "imS"           2
## 8 "imU"          35
## 9 "omL"           5
```

Values of row 47 are corrected as follows:

```
ecolidataframe[47,2]= 0.43
ecolidataframe[47,3]= 0.40
ecolidataframe[47,4]= 0.48
ecolidataframe[47,5]= 0.50
ecolidataframe[47,6]= 0.39
ecolidataframe[47,7]= 0.28
ecolidataframe[47,8]= 0.39
ecolidataframe[47,9]= "cp"
ecolidataframe[47,]
```

```
##     seqname  mcg gvh  lip chg  aac alm1 alm2 locsite
## 47 GT_ECOLI 0.43 0.4 0.48 0.5 0.39 0.28 0.39      cp
```

```
# Row 47 values have been corrected and the only remaining data issue is the leading space
# in certain locsites.
ecolidataframe %>% group_by(locsite) %>% summarize(count=n())
```

```
## # A tibble: 9 x 2
##   locsite count
##   <chr>   <int>
## 1 " cp"     141
## 2 " im"      77
## 3 " om"      20
## 4 " pp"      52
## 5 "cp"        1
## 6 "imL"       2
## 7 "imS"       2
## 8 "imU"      35
## 9 "omL"       5
```

The space in front of these locsites is removed as follows:

```
ecolidataframe$locsite<-trimws(ecolidataframe$locsite, which = c("left"))
ecolidataframe %>% group_by(locsite) %>% summarize(count=n())
```

```
## # A tibble: 8 x 2
##   locsite count
##   <chr>   <int>
## 1 cp        142
## 2 im         77
## 3 imL         2
## 4 imS         2
## 5 imU        35
## 6 om         20
## 7 omL         5
## 8 pp         52
```

All data issues have been resolved and no data has been lost in the process of data cleansing.

```
dim(ecolidataframe)
```

```
## [1] 335   9
```

```
class(ecolidataframe)
```

```
## [1] "data.frame"
```

```
head (ecolidataframe)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

```
str(ecolidataframe)
```

```
## 'data.frame':    335 obs. of  9 variables:
##  $ seqname: chr  "ACEA_ECOLI" "ACEK_ECOLI" "ACKA_ECOLI" "ADI_ECOLI" ...
##  $ mcg    : chr  "0.07" "0.56" "0.59" " 0.23" ...
##  $ gvh    : chr  "0.40" "0.40" "0.49" "0.32" ...
##  $ lip    : chr  "0.48" "0.48" "0.48" "0.48" ...
##  $ chg    : chr  "0.50" "0.50" "0.50" "0.50" ...
##  $ aac    : chr  "0.54" "0.49" "0.52" "0.55" ...
##  $ alm1   : chr  "0.35" "0.37" "0.45" "0.25" ...
##  $ alm2   : chr  "0.44" "0.46" "0.36" "0.35" ...
##  $ locsite: chr  "cp" "cp" "cp" "cp" ...
```

TRAIN AND TEST DATA SET PARTITIONS

The ecolidataframe is partitioned into train and test data sets as follows:

```
set.seed(1)
# Since the original data set is only 335 instances, allocate 80% for training and
# 20% for testing to allow more opportunity to test and validate the model while still using a
# significant portion to train the model.
test_index <- createDataPartition(y=ecolidataframe$locsite, times = 1, p = 0.2, list = FALSE)
ecoli_train <- ecolidataframe[-test_index,]
ecoli_test <- ecolidataframe[test_index,]
```

The ecoli train and test data sets resulting from partitioning the ecolidataframe are now ready for analysis and model development. They have the following number of instances:

```
nrow (ecoli_train)
```

```
## [1] 265
```

```
nrow (ecoli_test)
```

```
## [1] 70
```

ANALYSIS OF TRAIN DATA SET

Predicted Variable: locsite

The predicted variable is locsite. The possible locsites defined in an earlier section are distributed as follows in the train set.

```
ecoli_train %>% group_by(locsite) %>% summarize(count = n()) %>% arrange(desc(count))
```

```
## # A tibble: 8 x 2
##   locsite count
##   <chr>   <int>
## 1 cp        113
## 2 im         61
## 3 pp         41
## 4 imU        28
## 5 om         16
## 6 omL         4
## 7 imL         1
## 8 imS         1
```

The distribution shows that the top 5 locsites are cp, im, pp, imU, and om. The first in rank, cp, has almost double the frequency of the second in rank, im, indicating a significant lead.

Predictor Variables

The eight candidate predictor variables - seqname, mcg, gvh, lip, chg, aac, alm1, alm2 - are defined in an earlier section. Except for the seqname, all predictor variables represent numeric scores that indicate signal sequence, consensus sequence, presence of charge, discriminant analysis of the amino acid content, ALOM membrane, and ALOM program.

1. seqname

Upon inspection below, there are 265 distinct seqnames which corresponds to the number of rows in the train data set. This signifies that the seqname is a valid unique identifier of each instance in the train data set. For this reason, seqname will not be regarded as a predictor variable.

A check on the original ecolidataframe yielded 335 unique seqnames which is equivalent to the total number of instances in the downloaded ecoli data set. This further confirms that seqname is indeed a unique identifier and will not be a useful predictor.

```
n_distinct(ecoli_train$seqname)
```

```
## [1] 265
```

```
n_distinct(ecolidataframe$seqname)
```

```
## [1] 335
```

At this point, there are seven predictor variables: mcg, gvh, lip, chg, aac, alm1, and alm2 which will be profiled next.

2. mcg

As shown below, there are 99 possible mcg scores. The top five are: 0.63, 0.64, 0.67, 0.34, and 0.69.

```
n_distinct(ecoli_train$mcg)
```

```
## [1] 99
```

```
ecoli_train %>% group_by(mcg) %>% summarize(count = n()) %>% arrange (desc(count))
```

```
## # A tibble: 99 x 2
##    mcg   count
##    <chr> <int>
##  1 0.63     13
##  2 0.64     10
##  3 0.67      8
##  4 0.34      7
##  5 0.69      7
##  6 0.41      6
##  7 0.44      6
##  8 0.51      6
##  9 0.25      5
## 10 0.29      5
## # ... with 89 more rows
```

3. gvh

As shown below, there are 61 possible gvh scores. The top five are: 0.51, 0.49, 0.37, 0.47, and 0.40.

```
n_distinct(ecoli_train$gvh)
```

```
## [1] 61
```

```
ecoli_train %>% group_by(gvh) %>% summarize(count = n()) %>% arrange (desc(count))
```

```
## # A tibble: 61 x 2
##    gvh   count
##    <chr> <int>
##  1 0.51     16
##  2 0.49     13
##  3 0.37     12
##  4 0.47     12
##  5 0.40     10
##  6 0.44      9
##  7 0.57      9
##  8 0.42      8
##  9 0.45      8
## 10 0.46      8
## # ... with 51 more rows
```

4. lip

As shown below, there are 2 possible lip scores: 0.48 and 1.00.

```
n_distinct(ecoli_train$lip)
```

```
## [1] 2
```

```
ecoli_train %>% group_by(lip) %>% summarize(count = n()) %>% arrange (desc(count))
```

```
## # A tibble: 2 x 2
##   lip   count
##   <chr> <int>
## 1 0.48    258
## 2 1.00      7
```

5. chg

As shown below, there are 3 possible chg scores: 0.50, 0.5, and 1.00. But since 0.50 and 0.5 are the same, just with different decimal places, there are really only 2 possible chg scores: 0.50 and 1.00. This data is no longer cleaned at this point since the outlying decimal format, 0.5, occurs only once and will unlikely have any impact on locsite prediction.

```
n_distinct(ecoli_train$chg)
```

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

```
ecoli_train %>% group_by(chg) %>% summarize(count = n()) %>% arrange (desc(count))
```

```
## # A tibble: 3 x 2
##   chg   count
##   <chr> <int>
## 1 0.50    263
## 2 0.5       1
## 3 1.00      1
```

6. aac

As shown below, there are 57 possible aac scores. The top five are: 0.46, 0.54, 0.42, 0.48, and 0.41.

```
n_distinct(ecoli_train$aac)
```

```
## [1] 57
```

```
ecoli_train %>% group_by(aac) %>% summarize(count = n()) %>% arrange (desc(count))
```

```
## # A tibble: 57 x 2
##    aac   count
##    <chr> <int>
##  1 0.46     13
##  2 0.54     12
##  3 0.42     11
```

```
##  4 0.48     11
##  5 0.41     10
##  6 0.49     10
##  7 0.50     10
##  8 0.37      9
##  9 0.45      9
## 10 0.51      9
## # ... with 47 more rows
```

7. alm1

As shown below, there are 74 possible alm1 scores. The top five are: 0.33, 0.35, 0.78, 0.28, and 0.39.

```
n_distinct(ecoli_train$alm1)
```

```
## [1] 74
```

```
ecoli_train %>% group_by(alm1) %>% summarize(count = n()) %>% arrange (desc(count))
```

```
## # A tibble: 74 x 2
##    alm1  count
##    <chr> <int>
##  1 0.33      9
##  2 0.35      9
##  3 0.78      8
##  4 0.28      7
##  5 0.39      7
##  6 0.42      7
##  7 0.47      7
##  8 0.54      7
##  9 0.71      7
## 10 0.22      6
## # ... with 64 more rows
```

8. alm2

As shown below, there are 75 possible alm2 scores. The top five are: 0.33, 0.39, 0.43, 0.44, and 0.74.

```
n_distinct(ecoli_train$alm2)
```

```
## [1] 75
```

```
ecoli_train %>% group_by(alm2) %>% summarize(count = n()) %>% arrange (desc(count))
```

```
## # A tibble: 75 x 2
##    alm2  count
##    <chr> <int>
##  1 0.33     10
##  2 0.39      9
##  3 0.43      9
```

```
##  4 0.44      9
##  5 0.74      9
##  6 0.35      8
##  7 0.36      8
##  8 0.34      7
##  9 0.37      7
## 10 0.38      7
## # ... with 65 more rows
```

VISUALISATION AND INSIGHTS ON TRAIN DATA SET

To visually appreciate the contents of the train data set on which the machine learning algorithm will be modelled, the visual relationships between the predicted and predictor variables in the ecoli train data set are shown in this section.

However, this data visualisation exercise is not done for the ecoli test data set because the ecoli test data set is supposed to be of an "unknown" nature and will be used to evaluate the performance of the model developed from the ecoli train data set.

The section focuses on profiling the behaviour of each predictor variable vis-a-vis each possible locsite value, the predicted variable.

There will be 7 x 8 or 56 profiles as there are 7 predictors and 8 possible locsite values.

Data Subsets by whether Locsite was Predicted or Not

The analysis of every predictor variable will use the following data subsets. Each subset is intended to distinctly contain only the instances where the locsite of interest was predicted or not.

1. cp or not cp

```
#Subset data set that predicted locsite cp
ecoli_train_cp<- ecoli_train %>% filter(locsite == "cp")
nrow(ecoli_train_cp)
```

```
## [1] 113
```

```
head(ecoli_train_cp)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

```
#Subset data set that did not predict locsite cp
ecoli_train_not_cp<- ecoli_train %>% filter(locsite != "cp")
nrow(ecoli_train_not_cp)
```

```
## [1] 152
```

```
head(ecoli_train_not_cp)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 EMRA_ECOLI  0.06 0.61 0.48 0.50 0.49 0.92 0.37      im
## 2  AAS_ECOLI  0.44 0.52 0.48 0.50 0.43 0.47 0.54      im
## 3 AMPE_ECOLI  0.63 0.47 0.48 0.50 0.51 0.82 0.84      im
## 4 ARAE_ECOLI  0.23 0.48 0.48 0.50 0.59 0.88 0.89      im
## 5 ARAH_ECOLI  0.34 0.49 0.48 0.50 0.58 0.85 0.80      im
## 6 ATKB_ECOLI  0.46 0.61 0.48 0.50 0.48 0.86 0.87      im
```

2. im or not im

```
#Subset data set that predicted locsite im
ecoli_train_im<- ecoli_train %>% filter(locsite == "im")
nrow(ecoli_train_im)
```

```
## [1] 61
```

```
head(ecoli_train_im)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 EMRA_ECOLI  0.06 0.61 0.48 0.50 0.49 0.92 0.37      im
## 2  AAS_ECOLI  0.44 0.52 0.48 0.50 0.43 0.47 0.54      im
## 3 AMPE_ECOLI  0.63 0.47 0.48 0.50 0.51 0.82 0.84      im
## 4 ARAE_ECOLI  0.23 0.48 0.48 0.50 0.59 0.88 0.89      im
## 5 ARAH_ECOLI  0.34 0.49 0.48 0.50 0.58 0.85 0.80      im
## 6 ATKB_ECOLI  0.46 0.61 0.48 0.50 0.48 0.86 0.87      im
```

```
#Subset data set that did not predict locsite im
ecoli_train_not_im<- ecoli_train %>% filter(locsite != "im")
nrow(ecoli_train_not_im)
```

```
## [1] 204
```

```
head(ecoli_train_not_im)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

3. pp or not pp

```r
#Subset data set that predicted locsite pp
ecoli_train_pp<- ecoli_train %>% filter(locsite == "pp")
nrow(ecoli_train_pp)
```

```
## [1] 41
```

```r
head(ecoli_train_pp)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1  AGP_ECOLI  0.74 0.49 0.48 0.50 0.42 0.54 0.36      pp
## 2 AMY1_ECOLI  0.70 0.61 0.48 0.50 0.56 0.52 0.43      pp
## 3 ARAF_ECOLI  0.66 0.86 0.48 0.50 0.34 0.41 0.36      pp
## 4 ASG2_ECOLI  0.73 0.78 0.48 0.50 0.58 0.51 0.31      pp
## 5 BGLX_ECOLI  0.65 0.57 0.48 0.50 0.47 0.47 0.51      pp
## 6 C562_ECOLI  0.72 0.86 0.48 0.50 0.17 0.55 0.21      pp
```

```r
#Subset data set that did not predict locsite pp
ecoli_train_not_pp<- ecoli_train %>% filter(locsite != "pp")
nrow(ecoli_train_not_pp)
```

```
## [1] 224
```

```r
head(ecoli_train_not_pp)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

4. imU or not imU

```r
#Subset data set that predicted locsite imU
ecoli_train_imU<- ecoli_train %>% filter(locsite == "imU")
nrow(ecoli_train_imU)
```

```
## [1] 28
```

```r
head(ecoli_train_imU)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1  BCR_ECOLI  0.79 0.41 0.48 0.50 0.66 0.81 0.83     imU
## 2 CADB_ECOLI  0.83 0.48 0.48 0.50 0.65 0.76 0.79     imU
## 3 CAIT_ECOLI  0.69 0.43 0.48 0.50 0.59 0.74 0.77     imU
## 4 CPXA_ECOLI  0.79 0.36 0.48 0.50 0.46 0.82 0.70     imU
## 5 CYDB_ECOLI  0.75 0.37 0.48 0.50 0.64 0.70 0.74     imU
## 6 CYOB_ECOLI  0.59 0.29 0.48 0.50 0.64 0.75 0.77     imU
```

```
#Subset data set that did not predict locsite imU
ecoli_train_not_imU<- ecoli_train %>% filter(locsite != "imU")
nrow(ecoli_train_not_imU)
```

```
## [1] 237
```

```
head(ecoli_train_not_imU)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

5. om or not om

```
#Subset data set that predicted locsite om
ecoli_train_om<- ecoli_train %>% filter(locsite == "om")
nrow(ecoli_train_om)
```

```
## [1] 16
```

```
head(ecoli_train_om)
```

```
##      seqname  mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 FADL_ECOLI 0.78 0.68 0.48 0.50 0.83 0.40 0.29      om
## 2 FHUA_ECOLI 0.63 0.69 0.48 0.50 0.65 0.41 0.28      om
## 3 LAMB_ECOLI 0.67 0.88 0.48 0.50 0.73 0.50 0.25      om
## 4 NFRA_ECOLI 0.61 0.75 0.48 0.50 0.51 0.33 0.33      om
## 5 NMPC_ECOLI 0.67 0.84 0.48 0.50 0.74 0.54 0.37      om
## 6 OMPC_ECOLI 0.73 0.84 0.48 0.50 0.86 0.58 0.29      om
```

```
#Subset data set that did not predict locsite om
ecoli_train_not_om<- ecoli_train %>% filter(locsite != "om")
nrow(ecoli_train_not_om)
```

```
## [1] 249
```

```
head(ecoli_train_not_om)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

6. omL or not omL

```r
#Subset data set that predicted locsite omL
ecoli_train_omL<- ecoli_train %>% filter(locsite == "omL")
nrow(ecoli_train_omL)
```

```
## [1] 4
```

```r
head(ecoli_train_omL)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 MULI_ECOLI  0.77 0.57 1.00 0.50 0.37 0.54 0.01     omL
## 2 NLPB_ECOLI  0.66 0.49 1.00 0.50 0.54 0.56 0.36     omL
## 3  PAL_ECOLI  0.67 0.55 1.00 0.50 0.66 0.58 0.16     omL
## 4  SLP_ECOLI  0.68 0.49 1.00 0.50 0.62 0.55 0.28     omL
```

```r
#Subset data set that did not predict locsite omL
ecoli_train_not_omL<- ecoli_train %>% filter(locsite != "omL")
nrow(ecoli_train_not_omL)
```

```
## [1] 261
```

```r
head(ecoli_train_not_omL)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

7. imL or not imL

```r
#Subset data set that predicted locsite imL
ecoli_train_imL<- ecoli_train %>% filter(locsite == "imL")
nrow(ecoli_train_imL)
```

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

```r
head(ecoli_train_imL)
```

```
##      seqname  mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 NLPA_ECOLI 0.75 0.55 1.00 1.00 0.40 0.47 0.30     imL
```

```r
#Subset data set that did not predict locsite imL
ecoli_train_not_imL<- ecoli_train %>% filter(locsite != "imL")
nrow(ecoli_train_not_imL)
```

```
## [1] 264
```

```
head(ecoli_train_not_imL)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

8. imS or not imS

```
#Subset data set that predicted locsite imS
ecoli_train_imS<- ecoli_train %>% filter(locsite == "imS")
nrow(ecoli_train_imS)
```

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

```
head(ecoli_train_imS)
```

```
##      seqname  mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 NFRB_ECOLI 0.63 0.49 0.48 0.50 0.54 0.76 0.79     imS
```

```
#Subset data set that did not predict locsite imS
ecoli_train_not_imS<- ecoli_train %>% filter(locsite != "imS")
nrow(ecoli_train_not_imS)
```

```
## [1] 264
```

```
head(ecoli_train_not_imS)
```

```
##      seqname   mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1 ACEA_ECOLI  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2 ACEK_ECOLI  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3 ACKA_ECOLI  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  ADI_ECOLI  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5 ALKH_ECOLI  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6 AMPD_ECOLI  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

Predictor Profile vis-a-vis Predicted Locsite

At this point, ecoli train locsite-specific subsets have been created. These will now be used to profile every predictor variable against the frequency of the presence and absence of the specific locsite.

To utilise histograms, predictor scores are first converted to numeric from their original string format in the downloaded ecoli data.

For every predictor variable, two histograms are presented: first, to visualise the predictor scores against the frequency of the locsite of interest; second, to visualise the predictor scores against the frequency of locsites other than the one of interest.

From a naive point of view, where only one predictor variable is known, one may use the insights in this section as rules of thumb. These rules of thumb may have some value in cases of emergency and doctors need to triage a patient based only on one predictor variable.

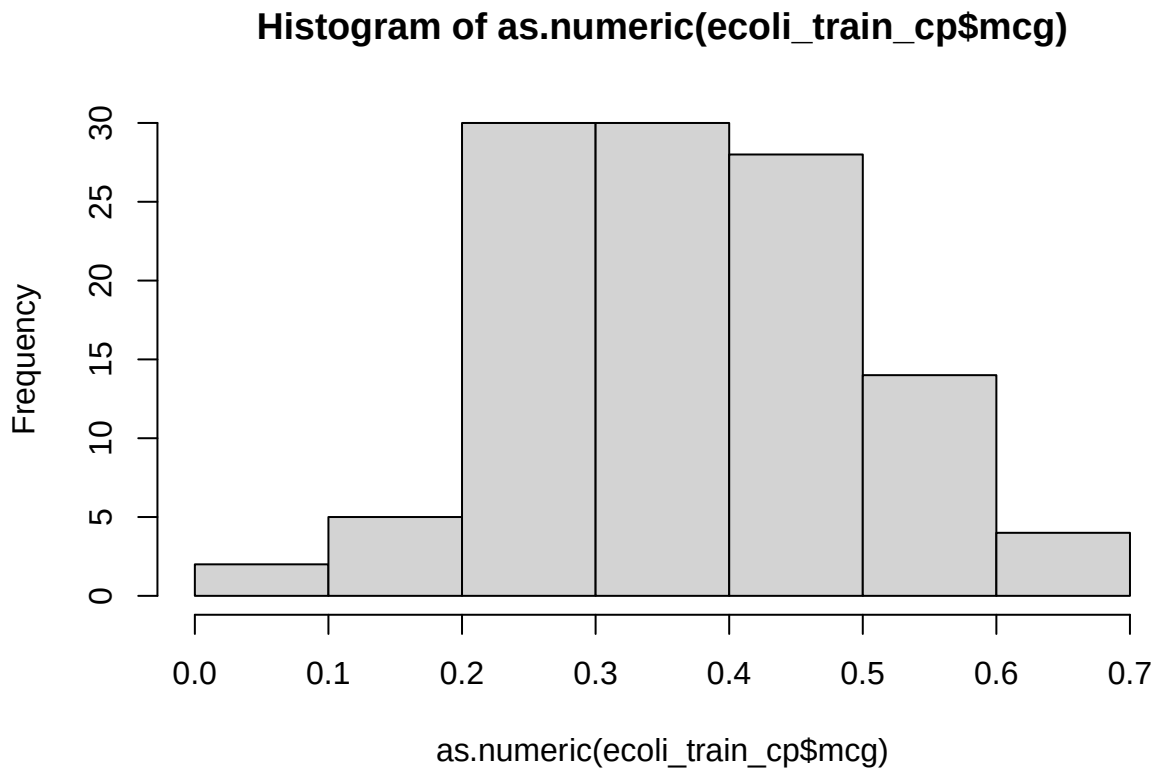
The quality of the downloaded ecoli data set is excellent because there are no missing values for all predictors. But this may not be the case in real-world data where there is a lot unknown.

1. mcg vs. cp and not cp

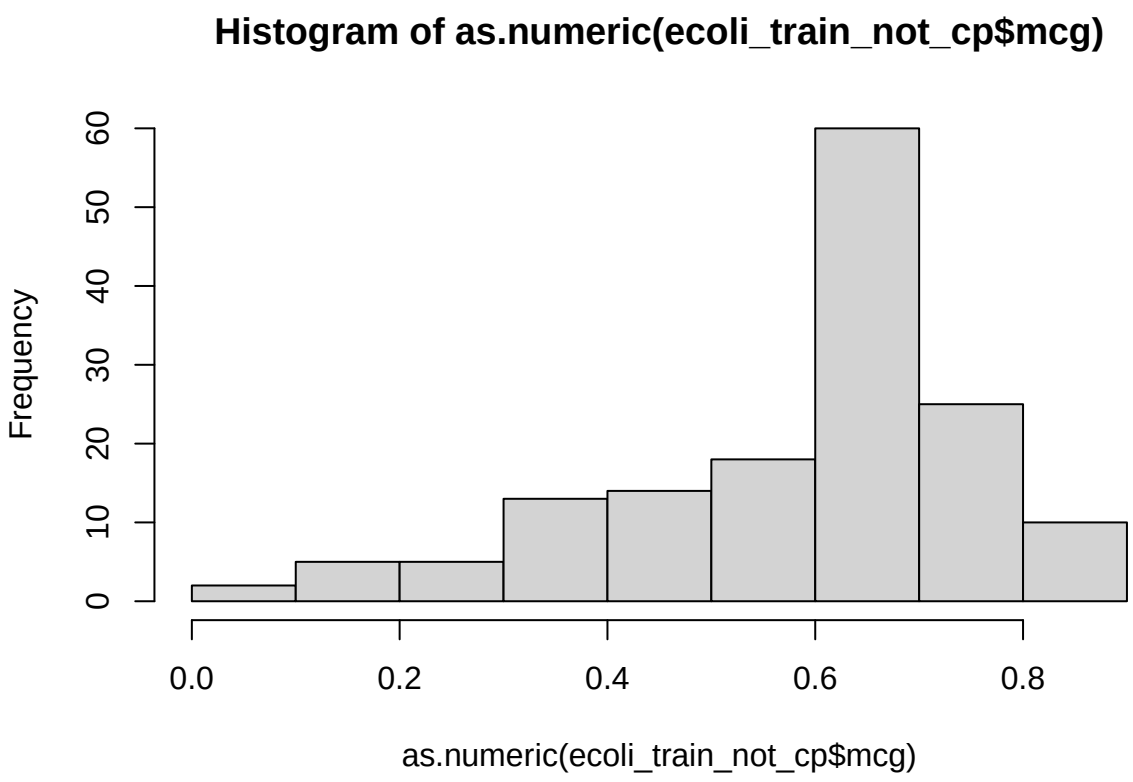
The cp histogram shows that cp is most frequently predicted when mcg score ranges from 0.2 to 0.4. The cp prediction frequencies for mcg scores higher than 0.4 are generally higher than those mcg scores lower than 0.2.

The not-cp histogram shows that when the mcg score is within 0.6 and 0.7, the predicted locsite is generally not cp.

```
hist(as.numeric(ecoli_train_cp$mcg))
```




```
hist(as.numeric(ecoli_train_not_cp$mcg))
```



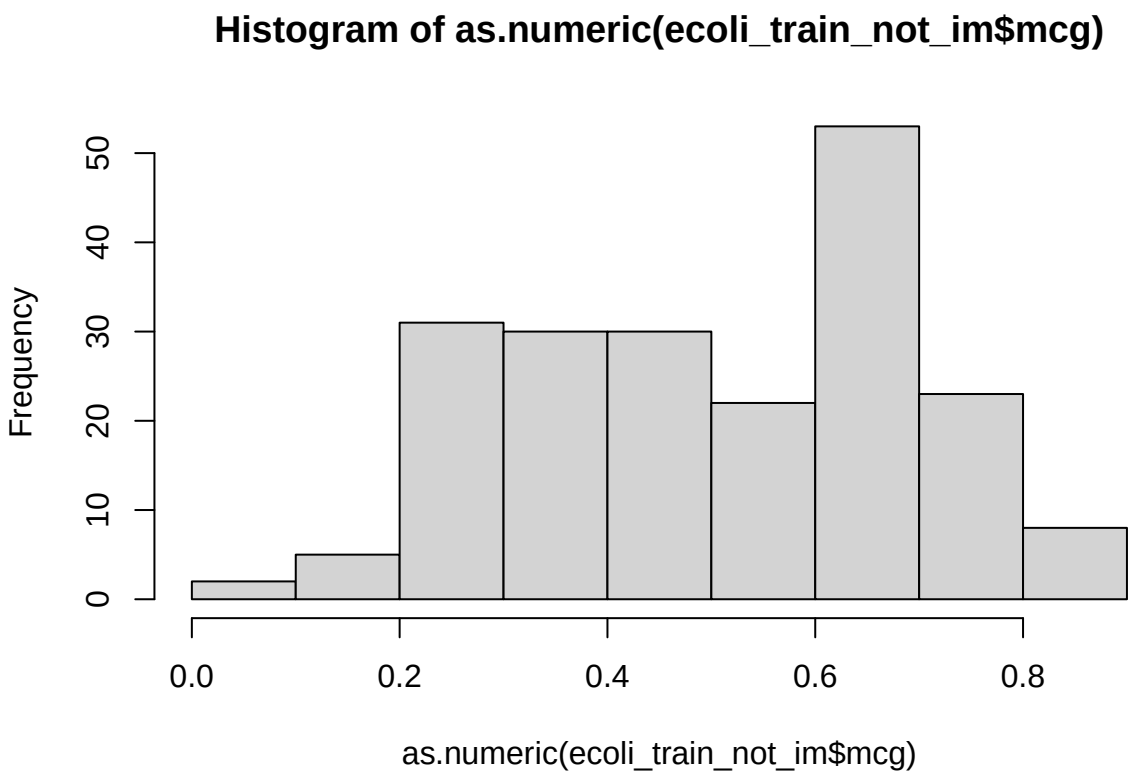
2. mcg vs. im and not ims

The im histogram shows that im is most frequently predicted when mcg score ranges from 0.3 to 0.4.

The not-im histogram shows that when the mcg score is within 0.6 and 0.7, the predicted locsite is generally not im.

```
hist(as.numeric(ecoli_train_im$mcg))
```


```
hist(as.numeric(ecoli_train_not_im$mcg))
```

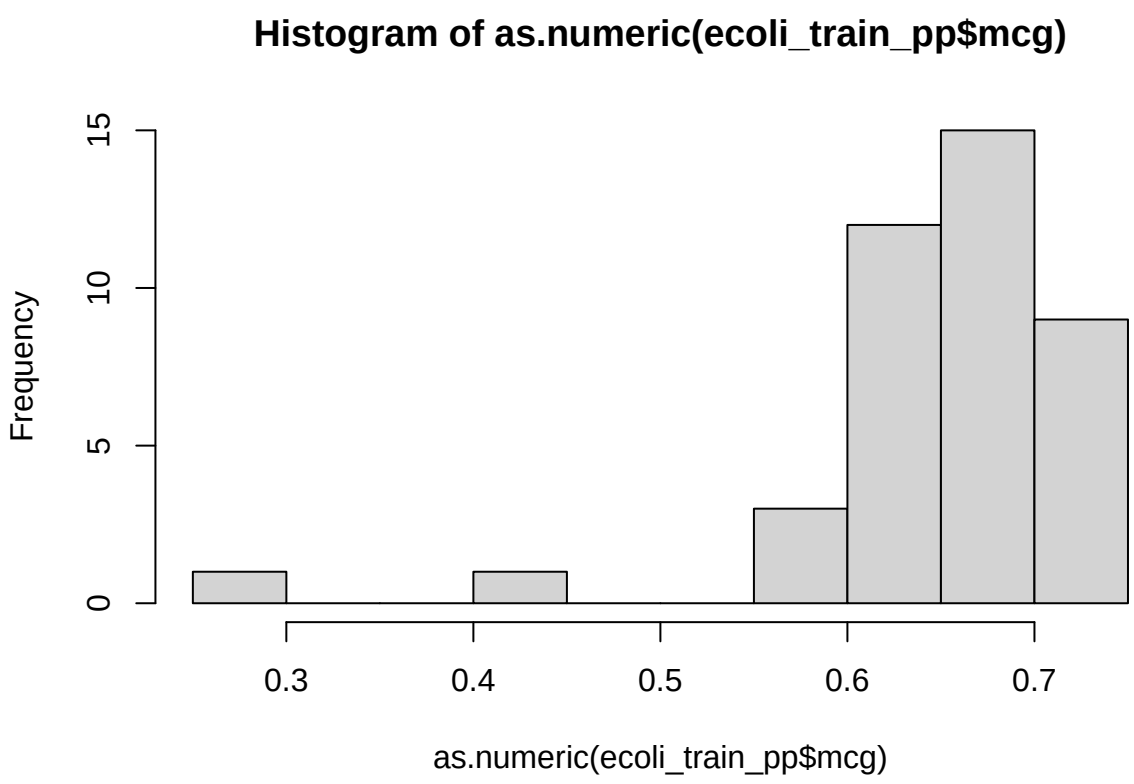


3. mcg vs. pp and not pp

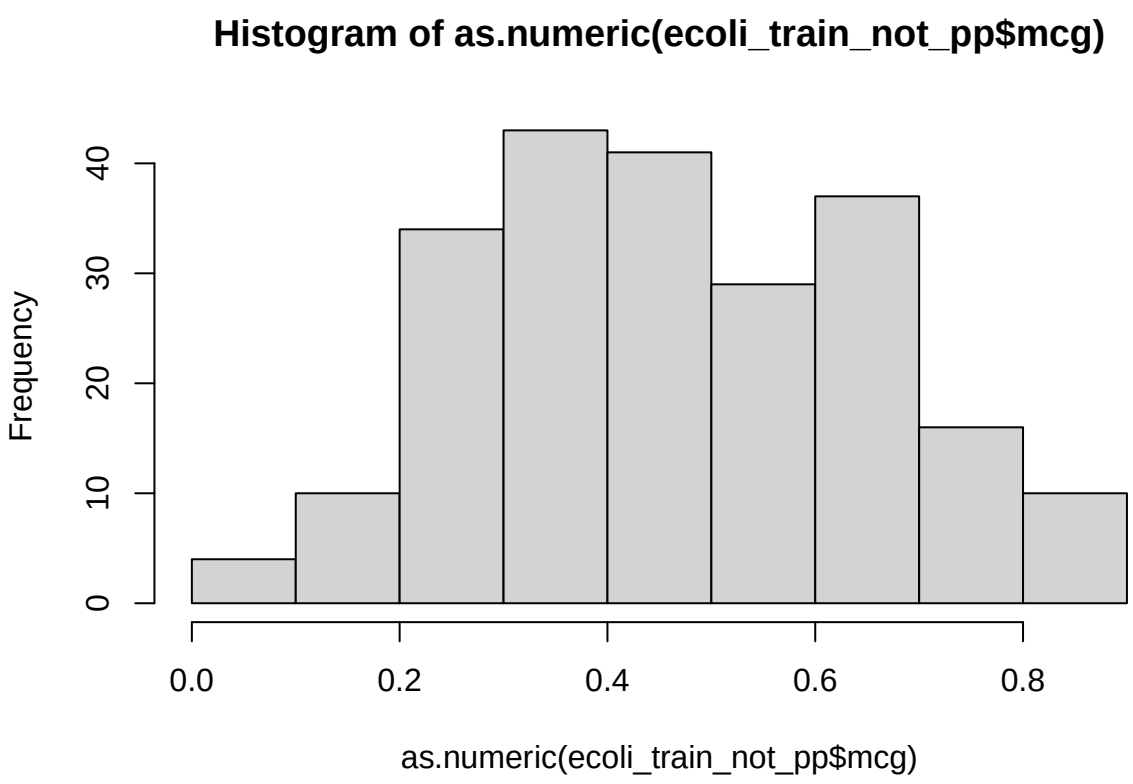
The pp histogram shows that pp is most frequently predicted when mcg score ranges from 0.65 to 0.7.

The not-pp histogram shows that when the mcg score is between 0.3 and 0.4, the predicted locsite is generally not pp.

```
hist(as.numeric(ecoli_train_pp$mcg))
```




```
hist(as.numeric(ecoli_train_not_pp$mcg))
```

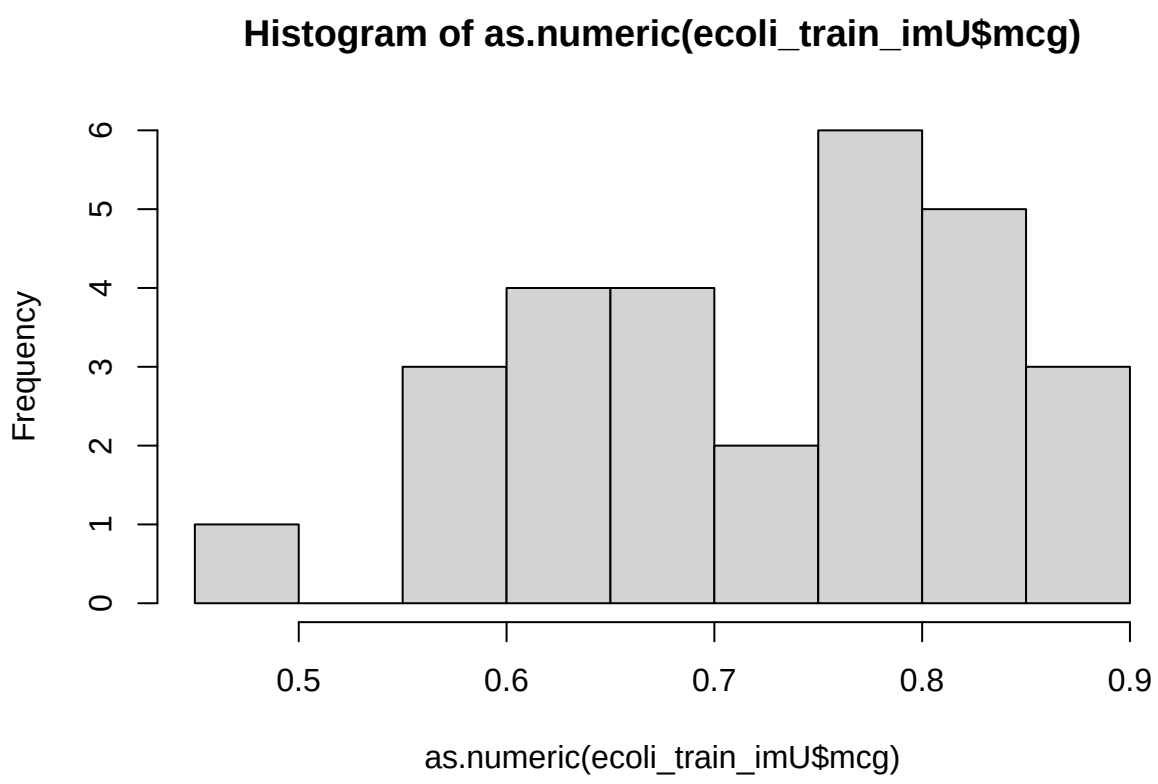


4. mcg vs. imU and not imU

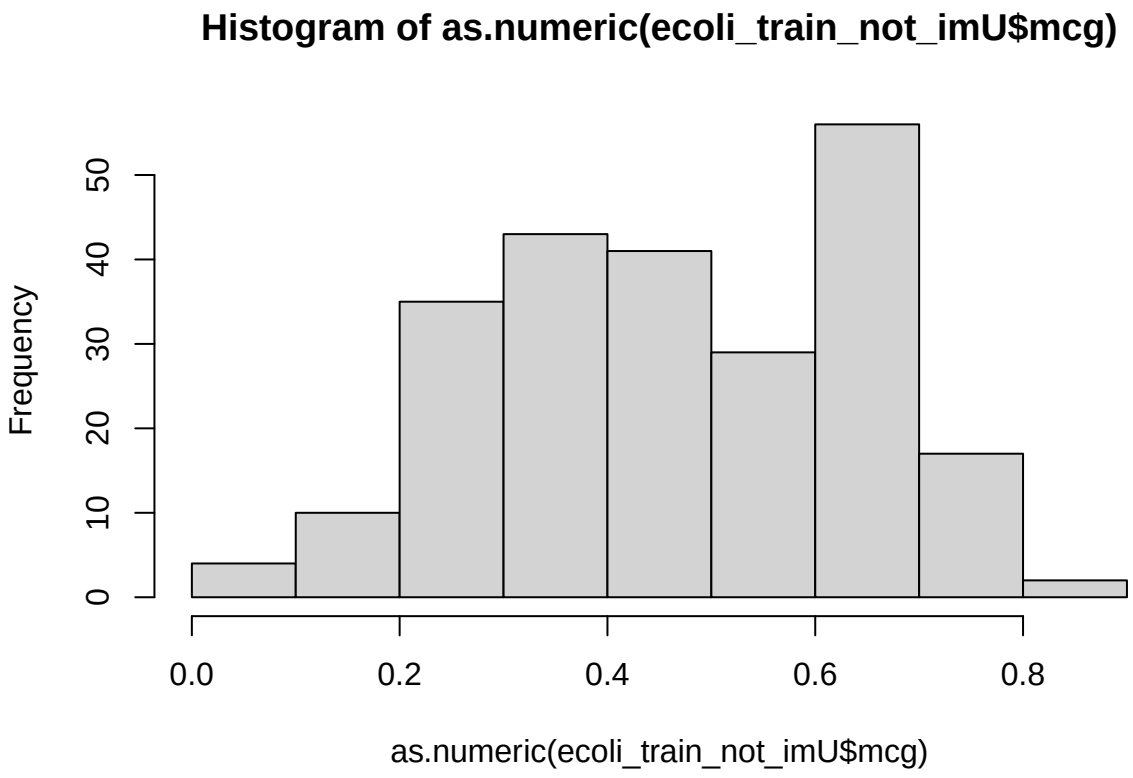
The imU histogram shows that imU is most frequently predicted when mcg score ranges from 0.75 to 0.8.

The not-imU histogram shows that when the mcg score is within 0.6 and 0.7, the predicted locsite is generally not imU.

```
hist(as.numeric(ecoli_train_imU$mcg))
```




```
hist(as.numeric(ecoli_train_not_imU$mcg))
```

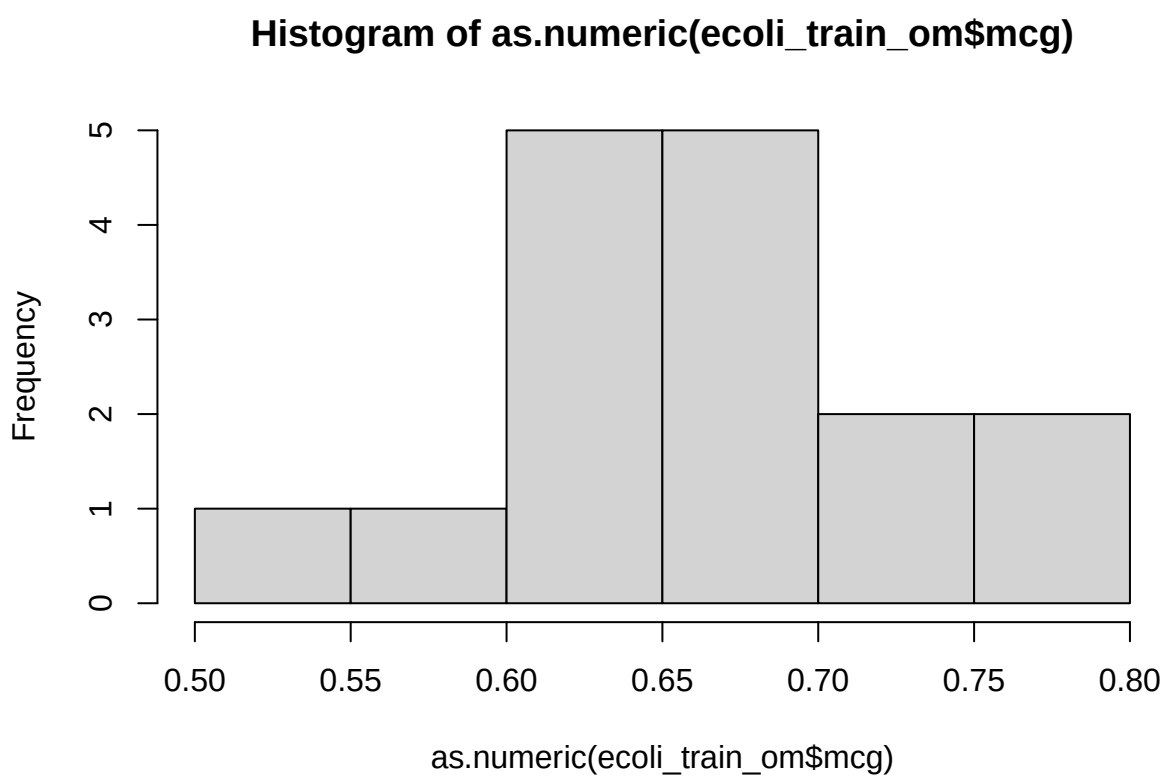


5. mcg vs. om and not om

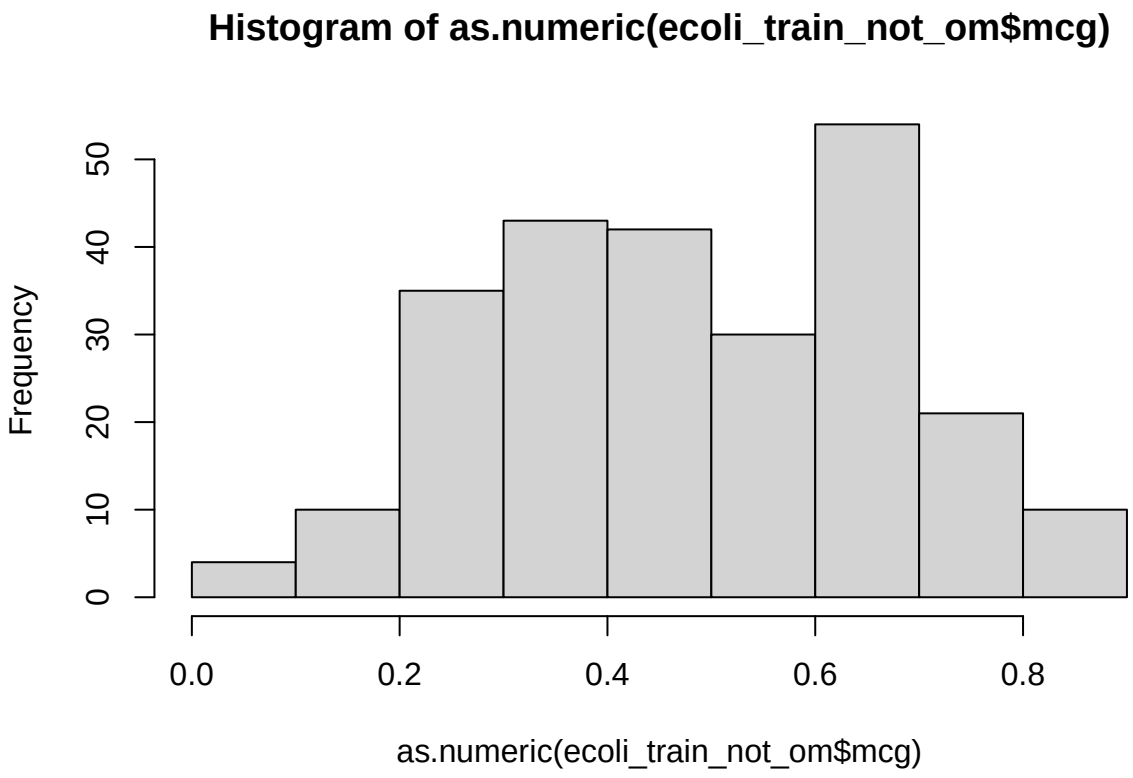
The om histogram shows that om is most frequently predicted when mcg score ranges from 0.6 to 0.7.

The not-om histogram shows that when the mcg score is within 0.6 and 0.7, the predicted locsite is generally not om.

```
hist(as.numeric(ecoli_train_om$mcg))
```




```
hist(as.numeric(ecoli_train_not_om$mcg))
```

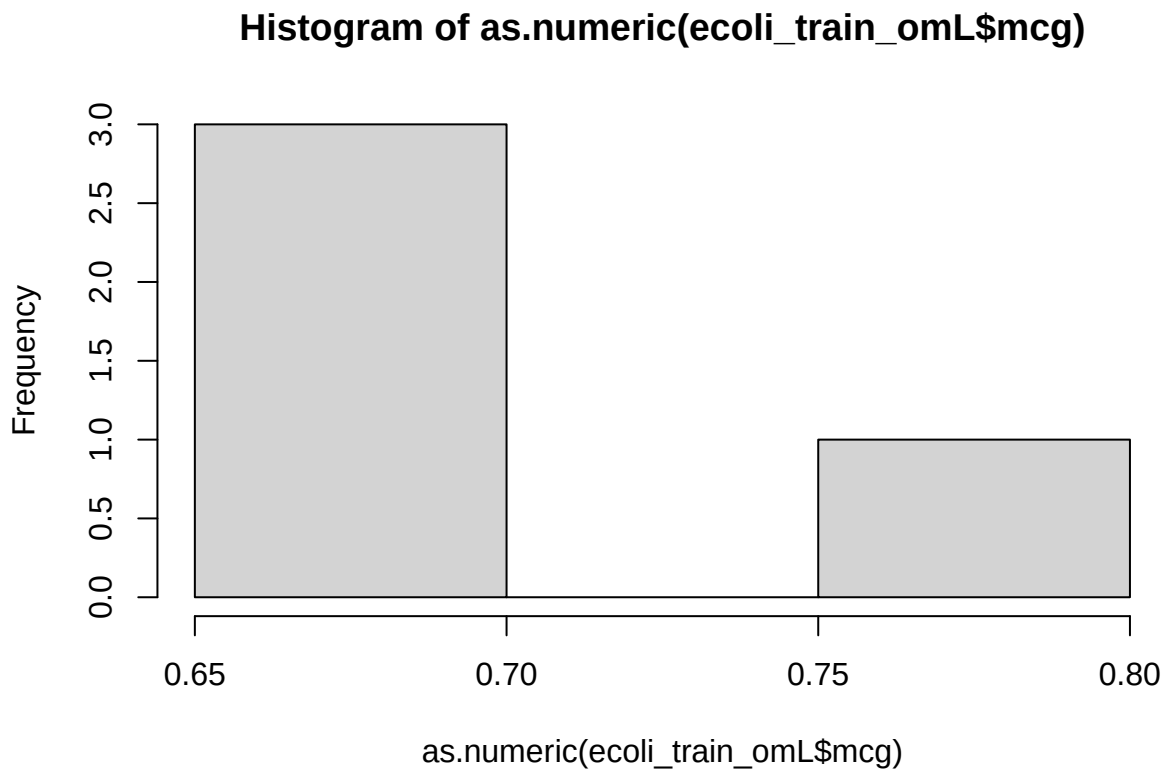


6. mcg vs. omL and not omL

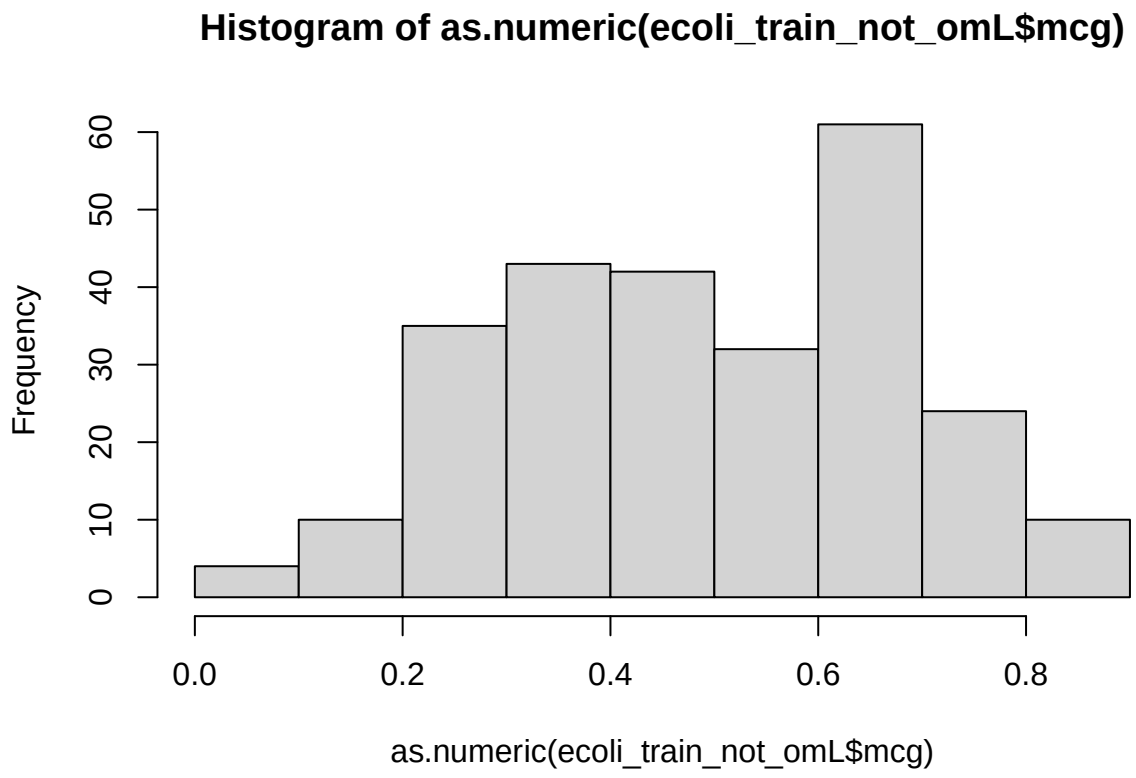
The omL histogram shows that omL is most frequently predicted when mcg score ranges from 0.65 to 0.7.

The not-omL histogram shows that when the mcg score is within 0.6 and 0.7, the predicted locsite is generally not omL.

```
hist(as.numeric(ecoli_train_omL$mcg))
```




```
hist(as.numeric(ecoli_train_not_omL$mcg))
```

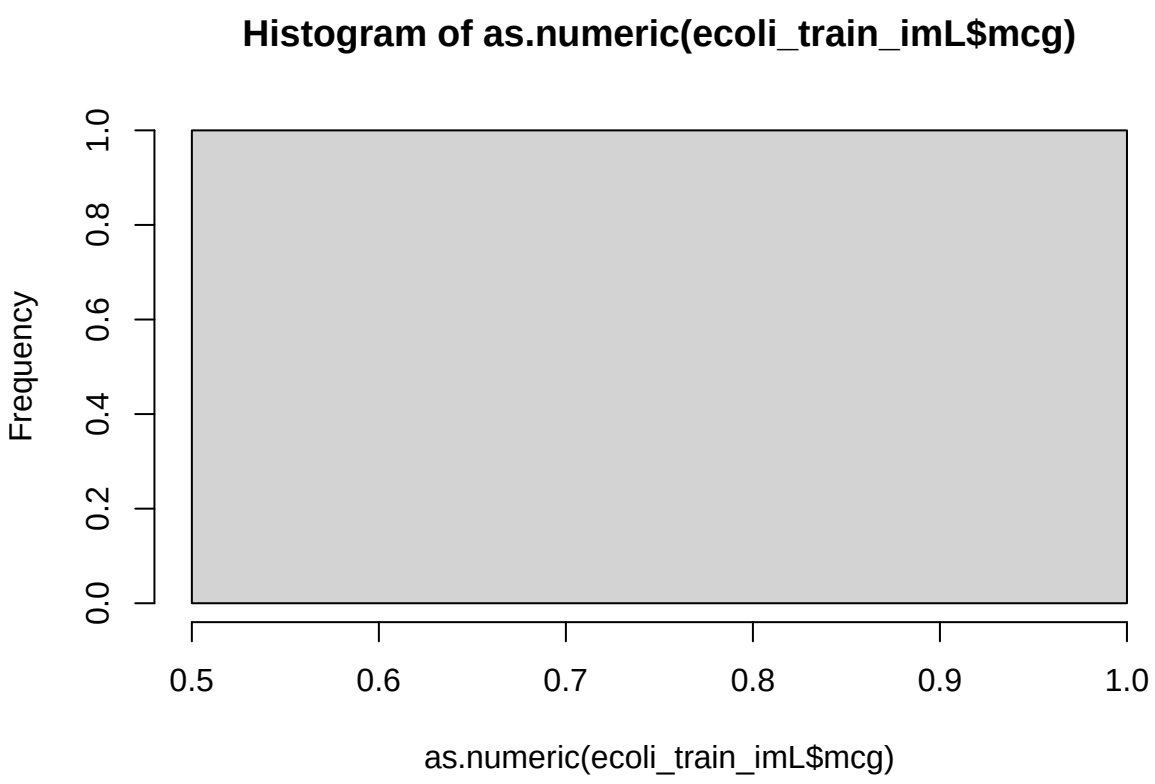


7. mcg vs. imL and not imL

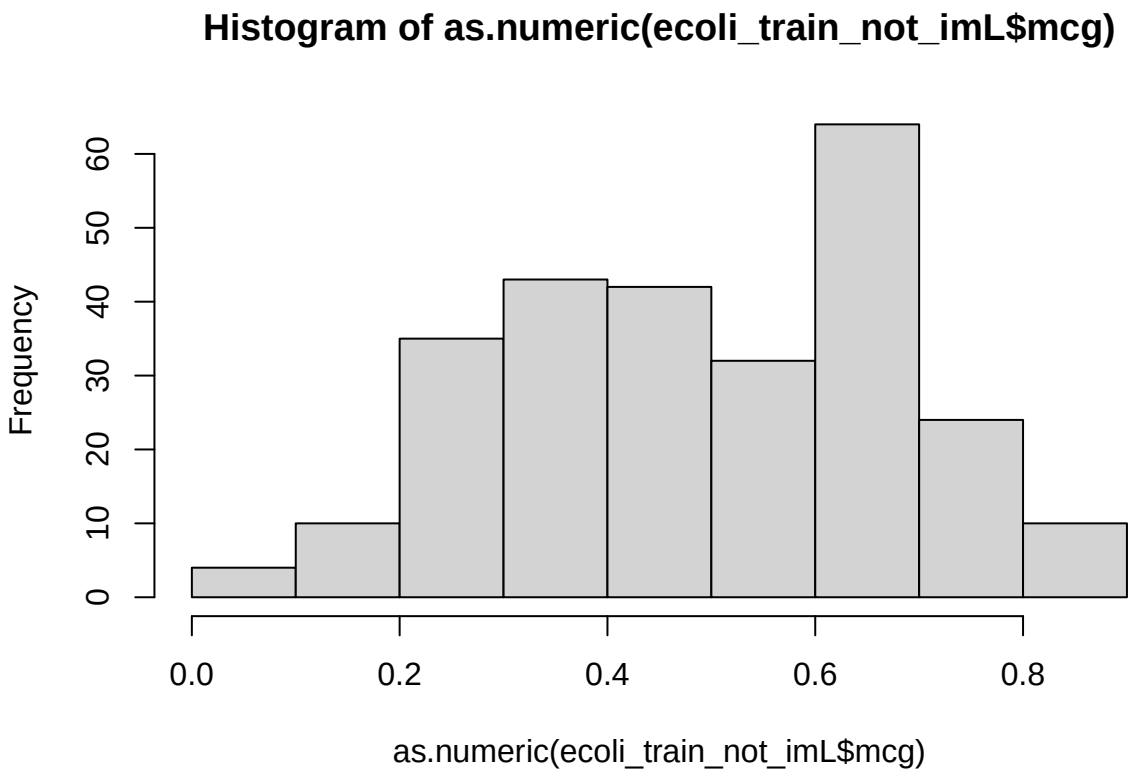
There is only one instance in the imL histogram. No specific mcg score range can be distinguished from this visualisation.

The not-imL histogram shows that when the mcg score is within 0.6 and 0.7, the predicted locsite is generally not imL.

```
hist(as.numeric(ecoli_train_imL$mcg))
```




```
hist(as.numeric(ecoli_train_not_imL$mcg))
```

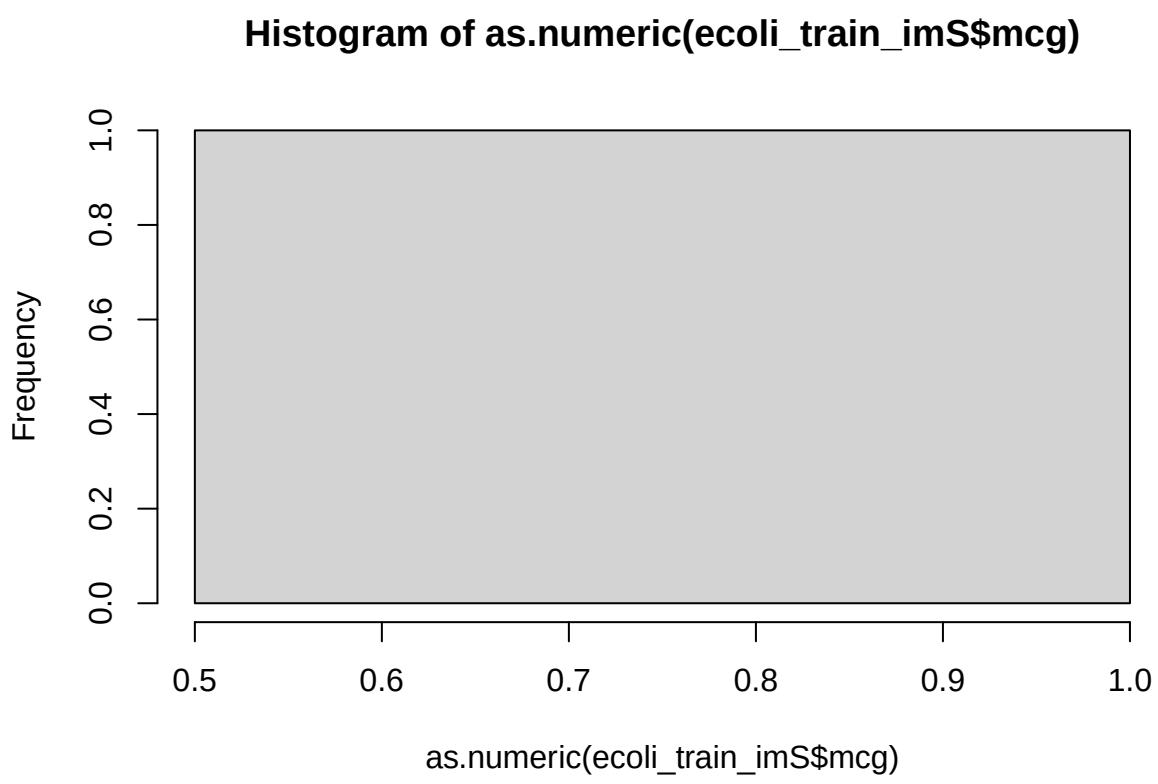


8. mcg vs. imS and not imS

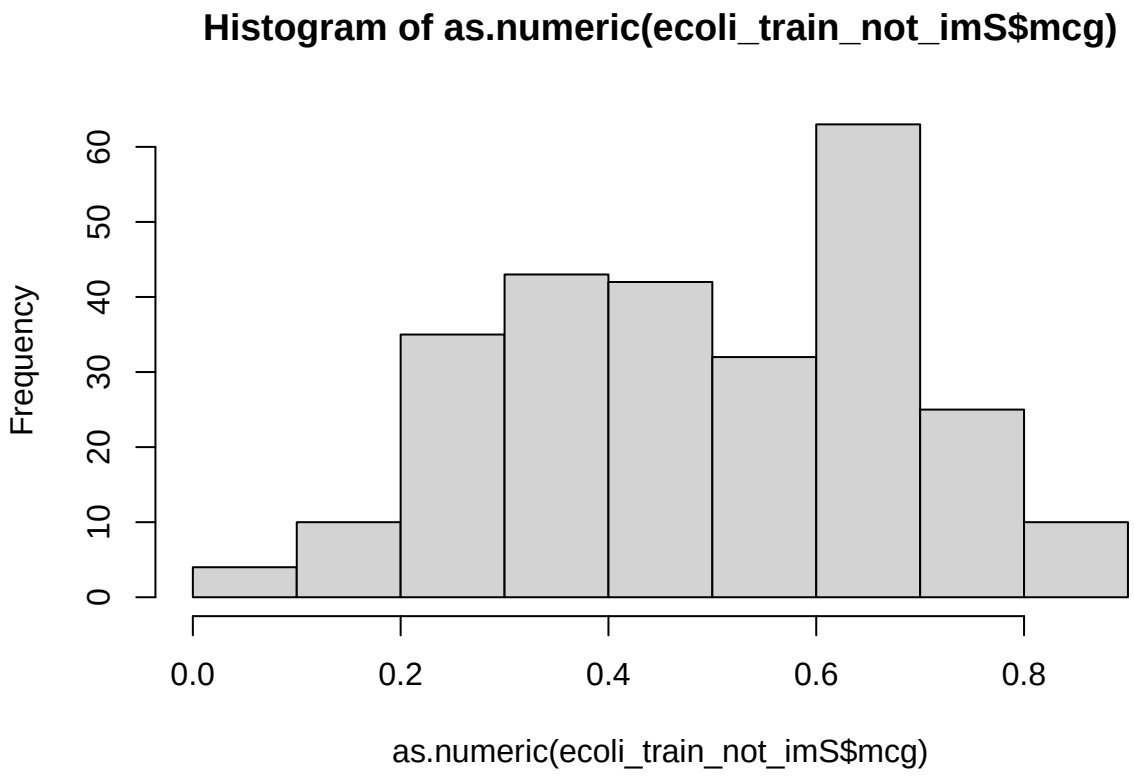
There is only one instance in the imS histogram. No specific mcg score range can be distinguished from this visualisation.

The not-imS histogram shows that when the mcg score is within 0.6 and 0.7, the predicted locsite is generally not imS.

```
hist(as.numeric(ecoli_train_imS$mcg))
```




```
hist(as.numeric(ecoli_train_not_imS$mcg))
```

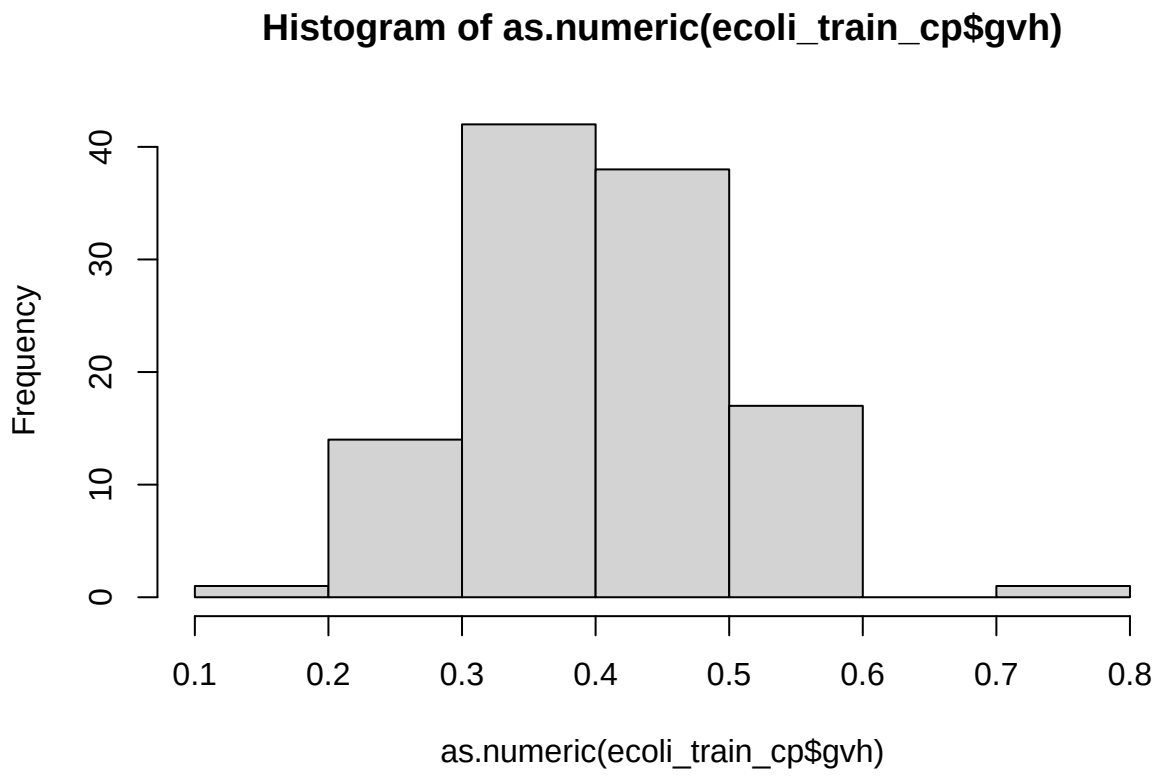


9. gvh vs. cp and not cp

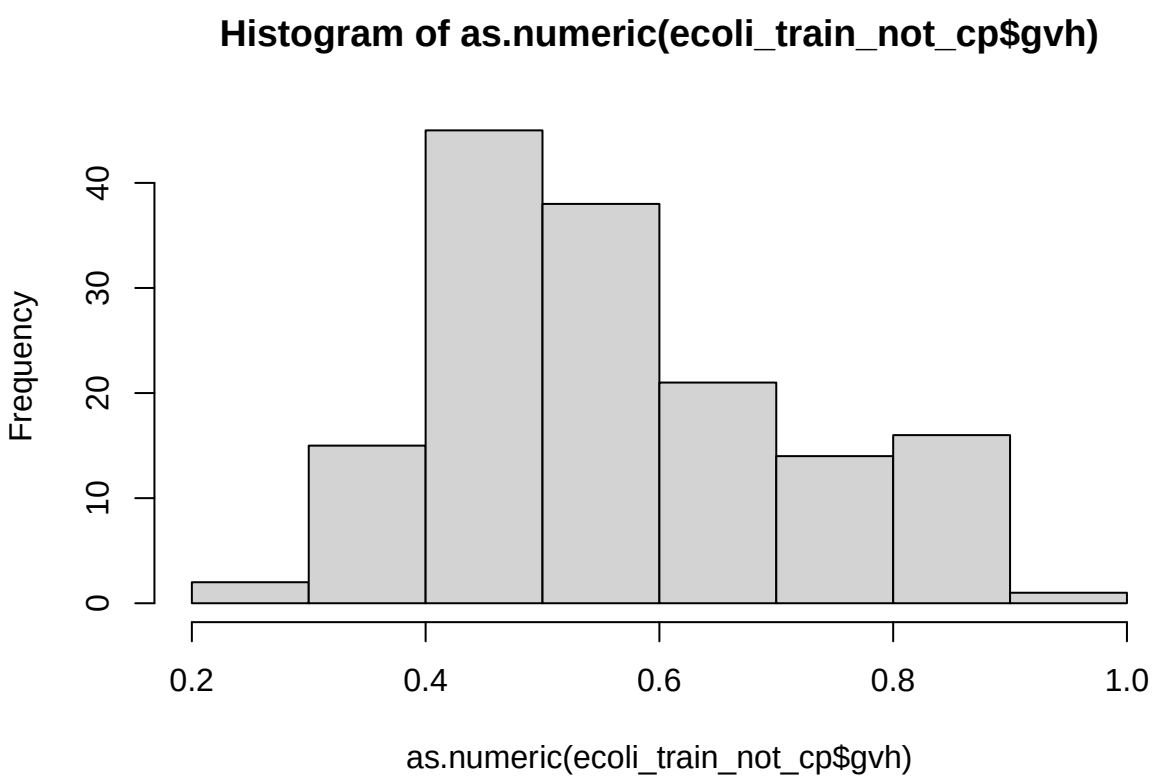
The cp histogram shows that cp is most frequently predicted when gvh score ranges from 0.3 to 0.4.

The not-cp histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not cp.

```
hist(as.numeric(ecoli_train_cp$gvh))
```




```
hist(as.numeric(ecoli_train_not_cp$gvh))
```

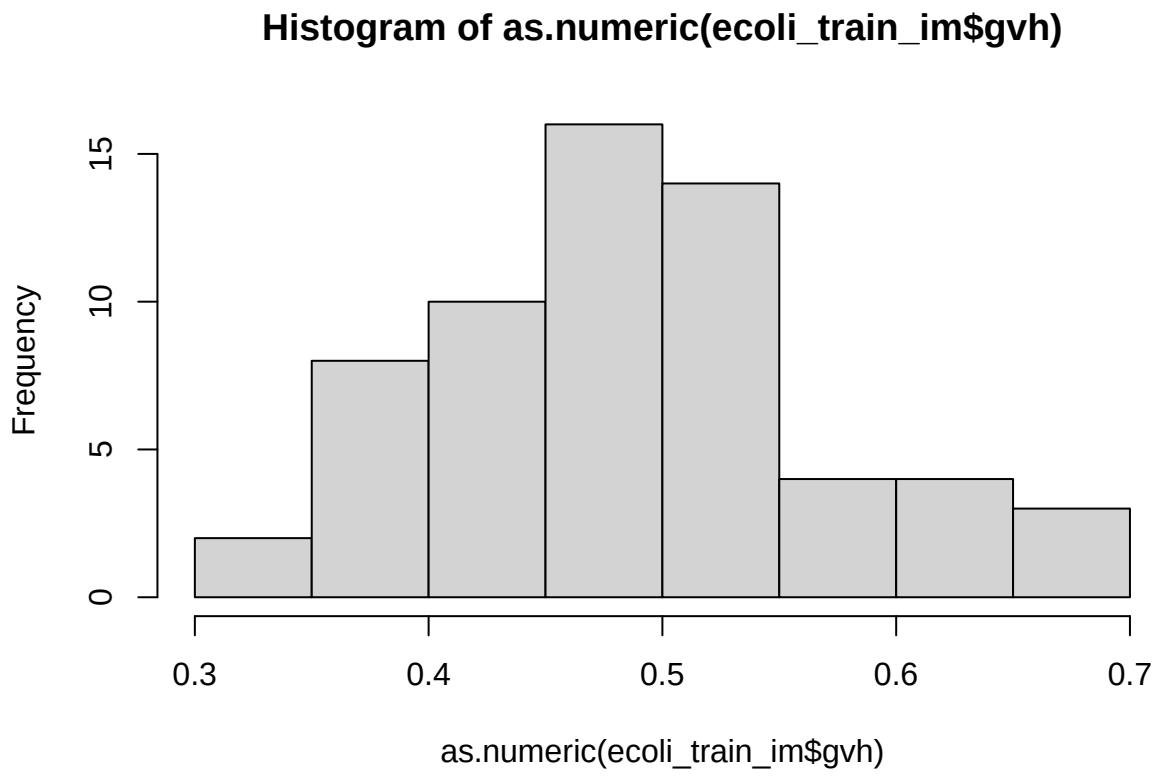


10. gvh vs. im and not im

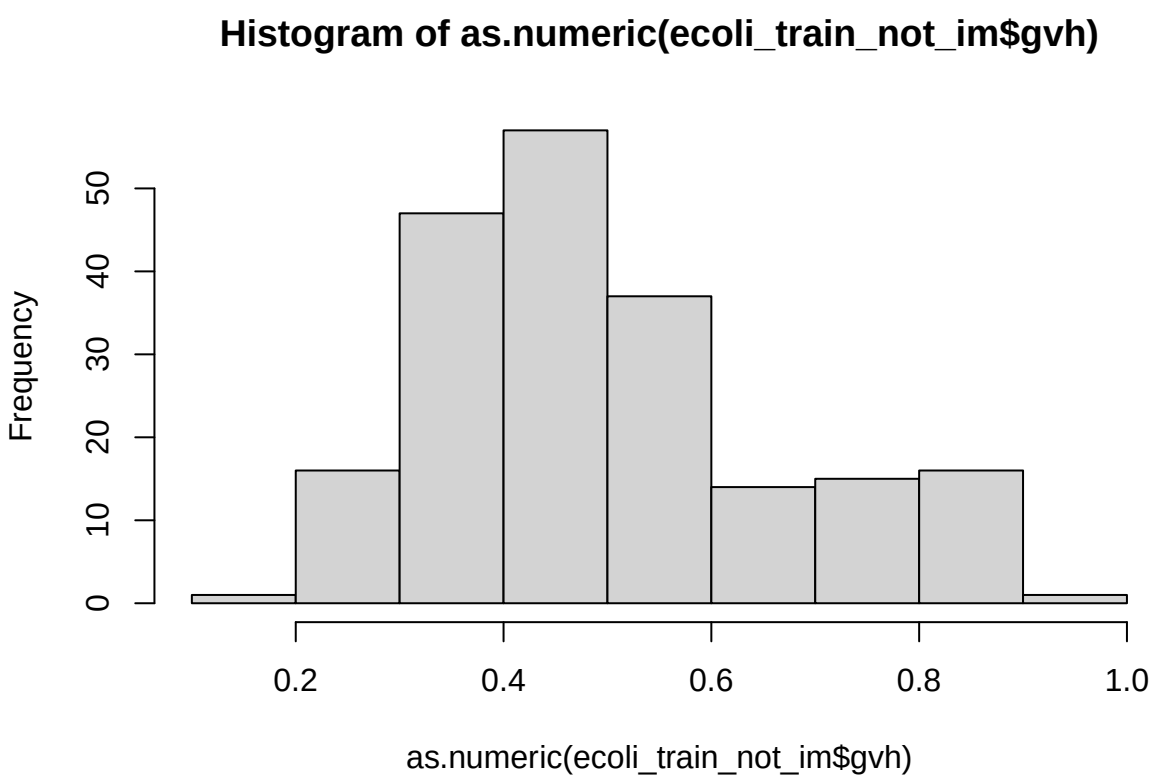
The im histogram shows that im is most frequently predicted when gvh score ranges from 0.45 to 0.5.

The not-im histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not im.

```
hist(as.numeric(ecoli_train_im$gvh))
```




```
hist(as.numeric(ecoli_train_not_im$gvh))
```

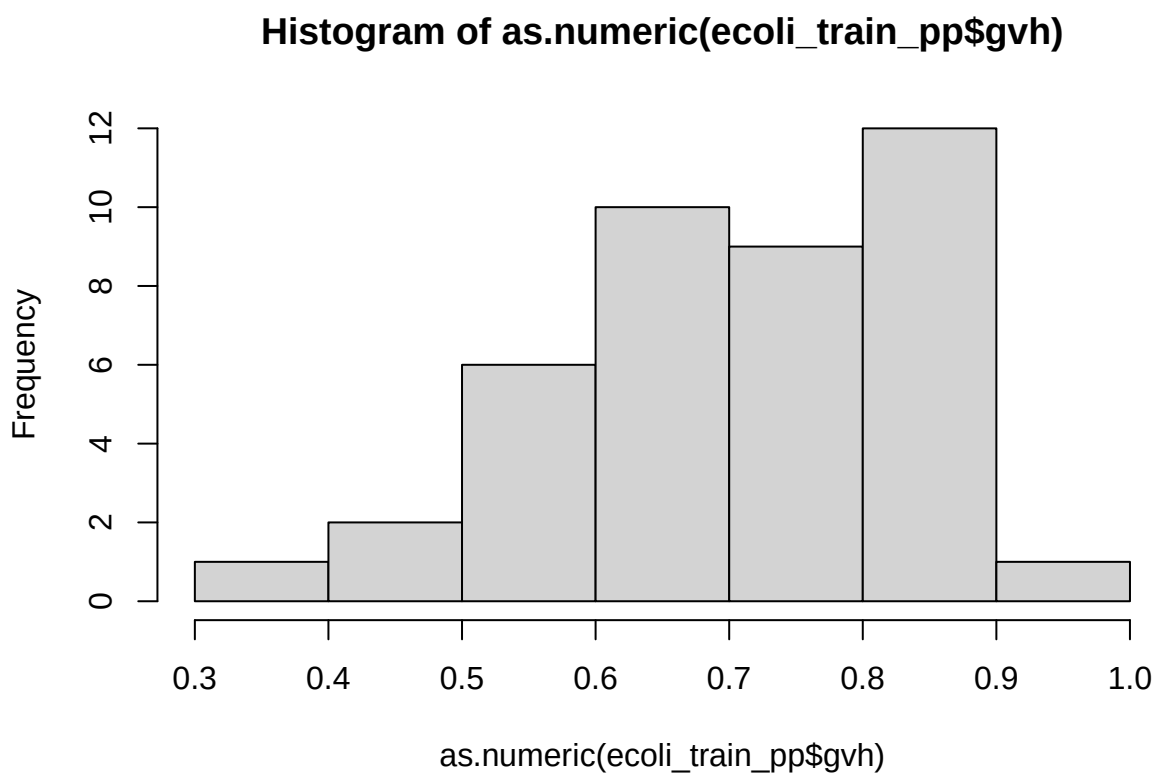


11. gvh vs. pp and not pp

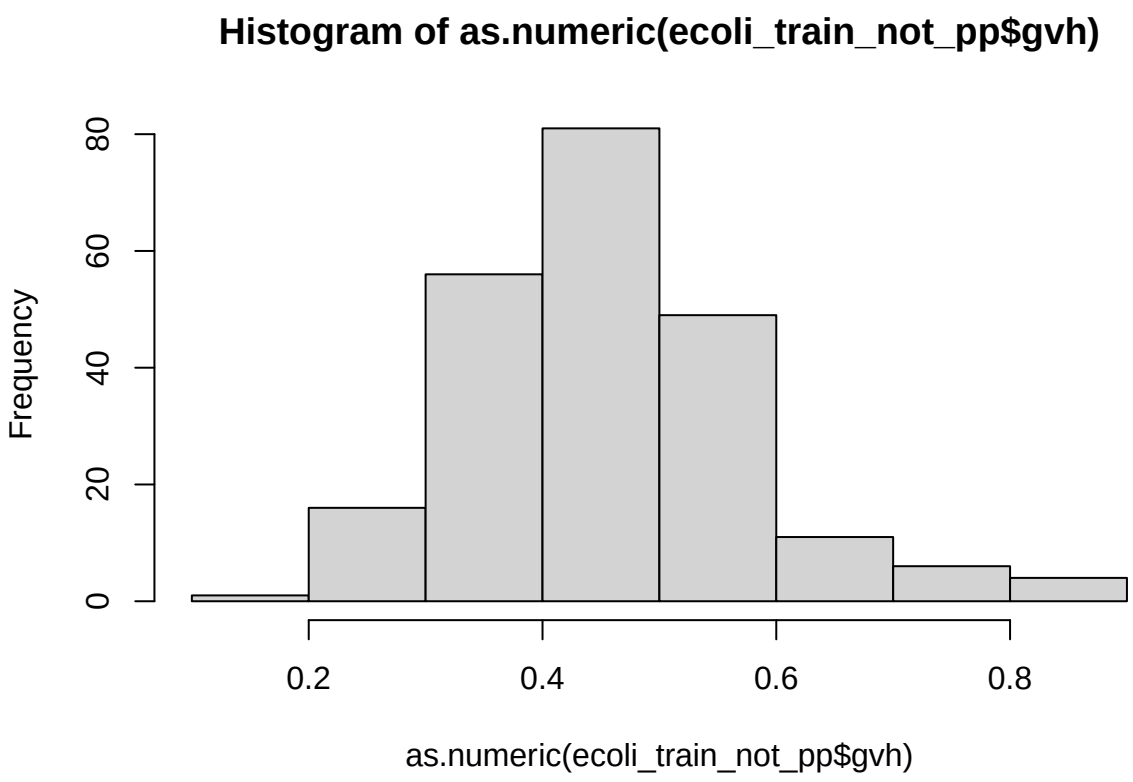
The pp histogram shows that pp is most frequently predicted when gvh score ranges from 0.8 to 0.9.

The not-pp histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not pp.

```
hist(as.numeric(ecoli_train_pp$gvh))
```




```
hist(as.numeric(ecoli_train_not_pp$gvh))
```

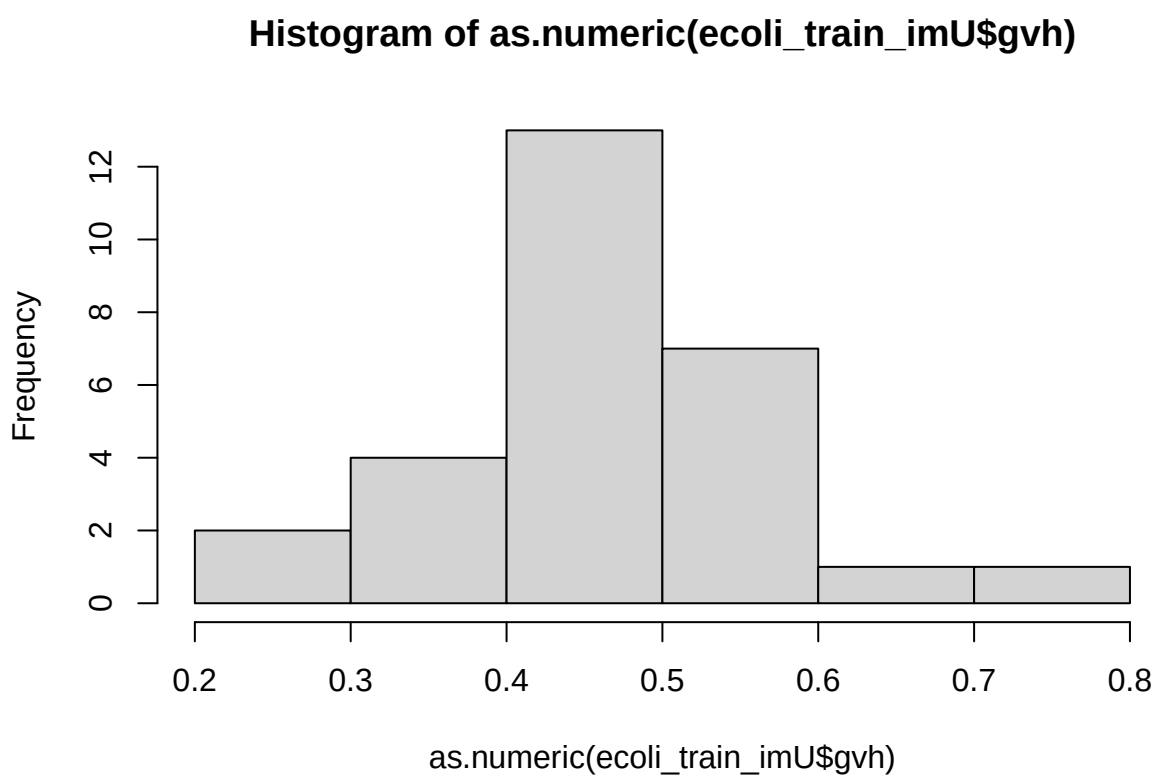


12. gvh vs. imU and not imU

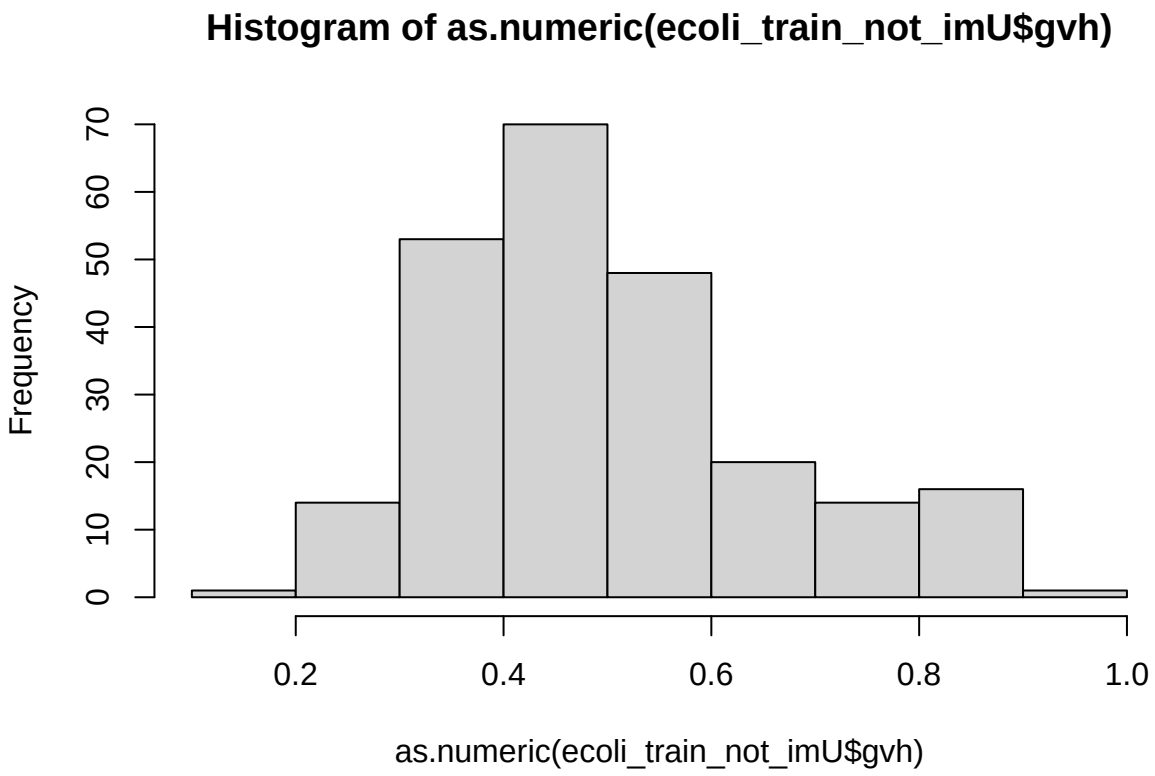
The imU histogram shows that imU is most frequently predicted when gvh score ranges from 0.4 to 0.5.

The not-imU histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not imU. This histogram has greater frequencies than the first and is likely more representative of imU behaviour than the first histogram, i.e., imU prediction.

```
hist(as.numeric(ecoli_train_imU$gvh))
```




```
hist(as.numeric(ecoli_train_not_imU$gvh))
```

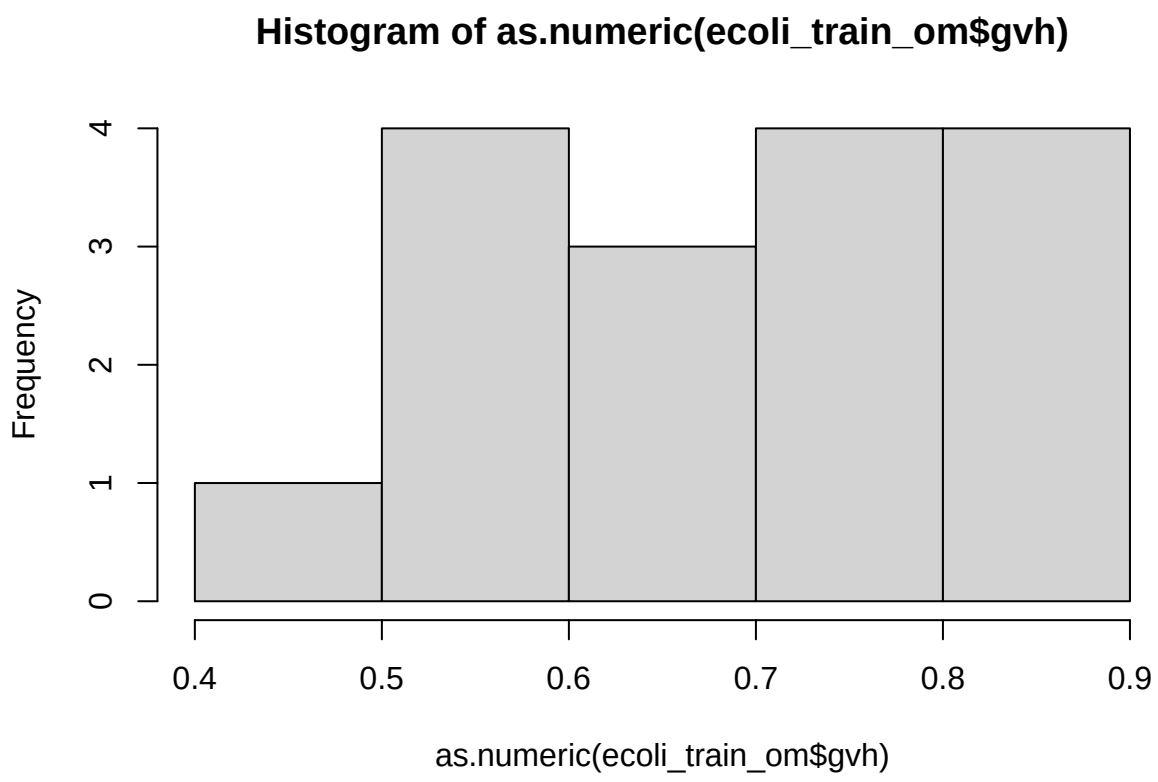


13. gvh vs. om and not om

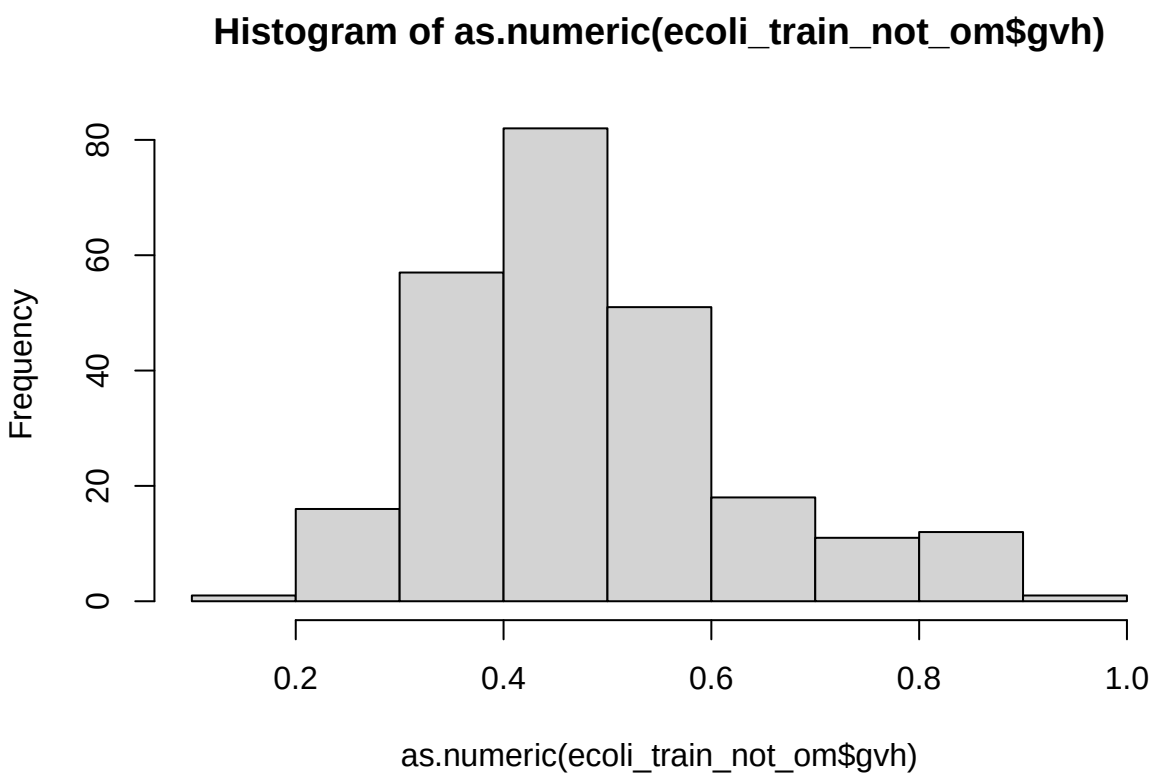
The om histogram shows that om is most frequently predicted when gvh score ranges from 0.5 to 0.6 and from 0.7 to 0.9.

The not-om histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not om.

```
hist(as.numeric(ecoli_train_om$gvh))
```




```
hist(as.numeric(ecoli_train_not_om$gvh))
```

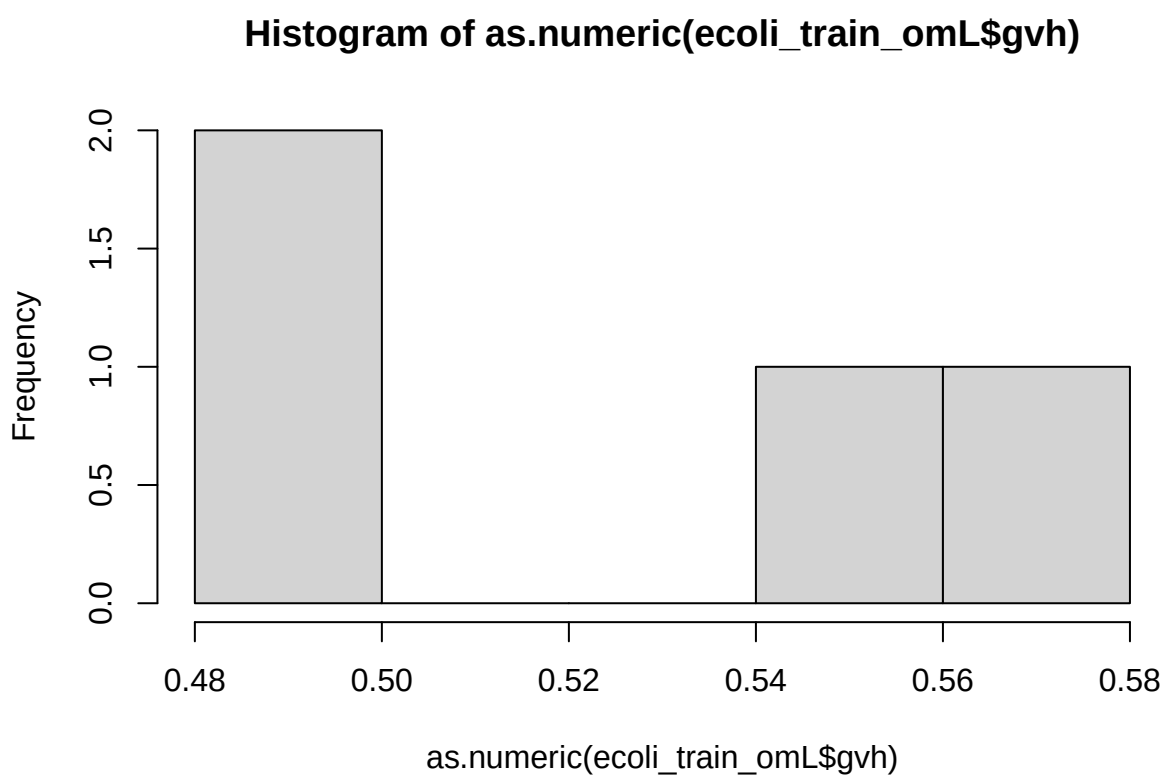


14. gvh vs. omL and not omL

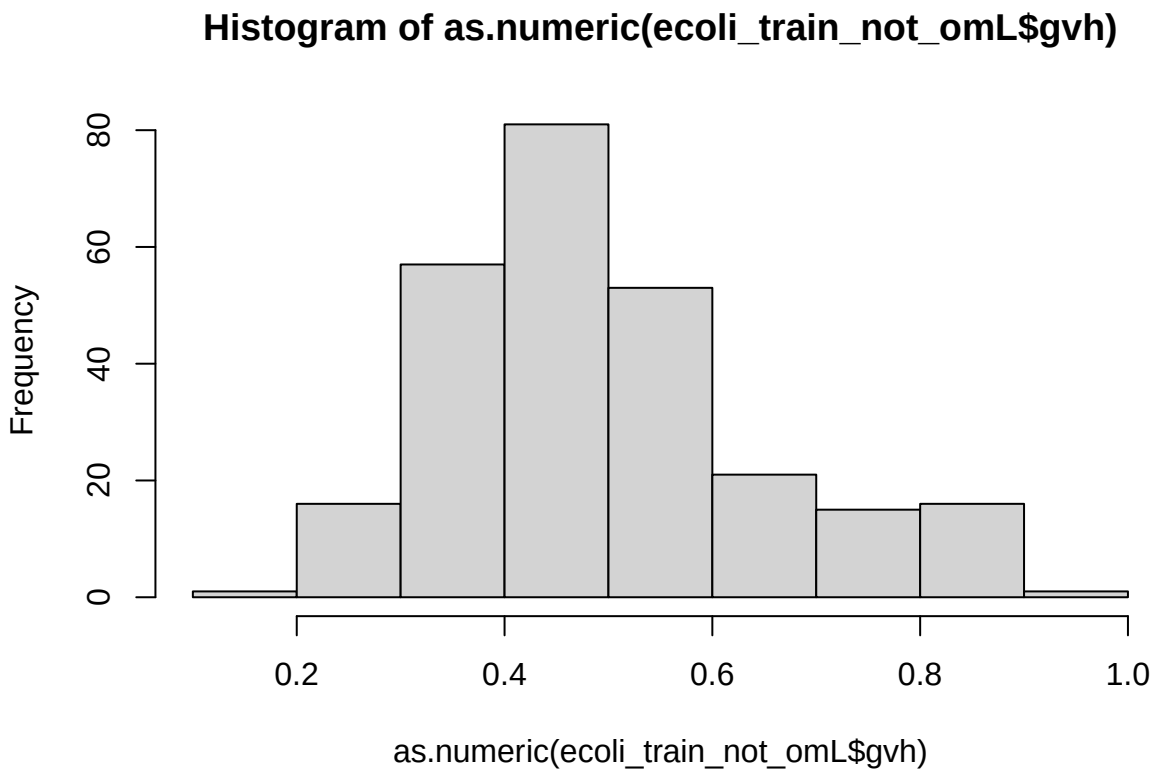
The omL histogram shows that omL is most frequently predicted when gvh score ranges from 0.48 to 0.5.

The not-omL histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not omL. This histogram has greater frequencies than the first and is likely more representative of omL behaviour than the first histogram, i.e., omL prediction.

```
hist(as.numeric(ecoli_train_omL$gvh))
```




```
hist(as.numeric(ecoli_train_not_omL$gvh))
```

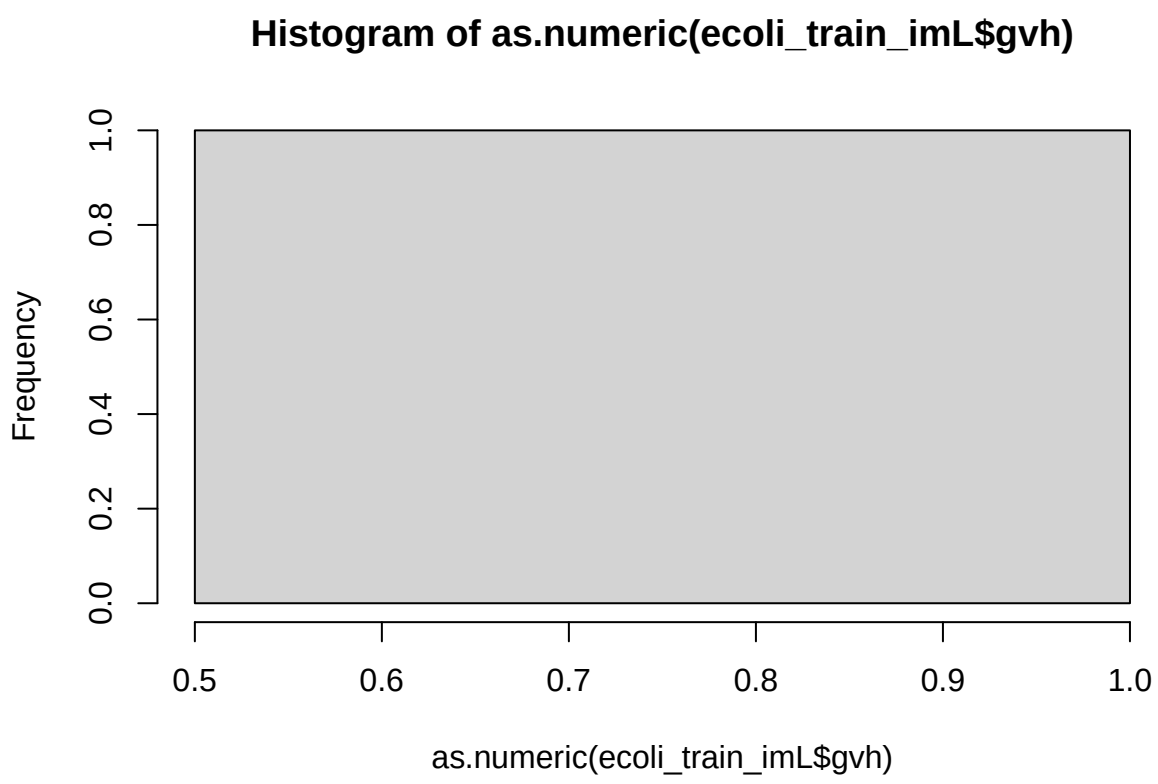


15. gvh vs. imL and not imL

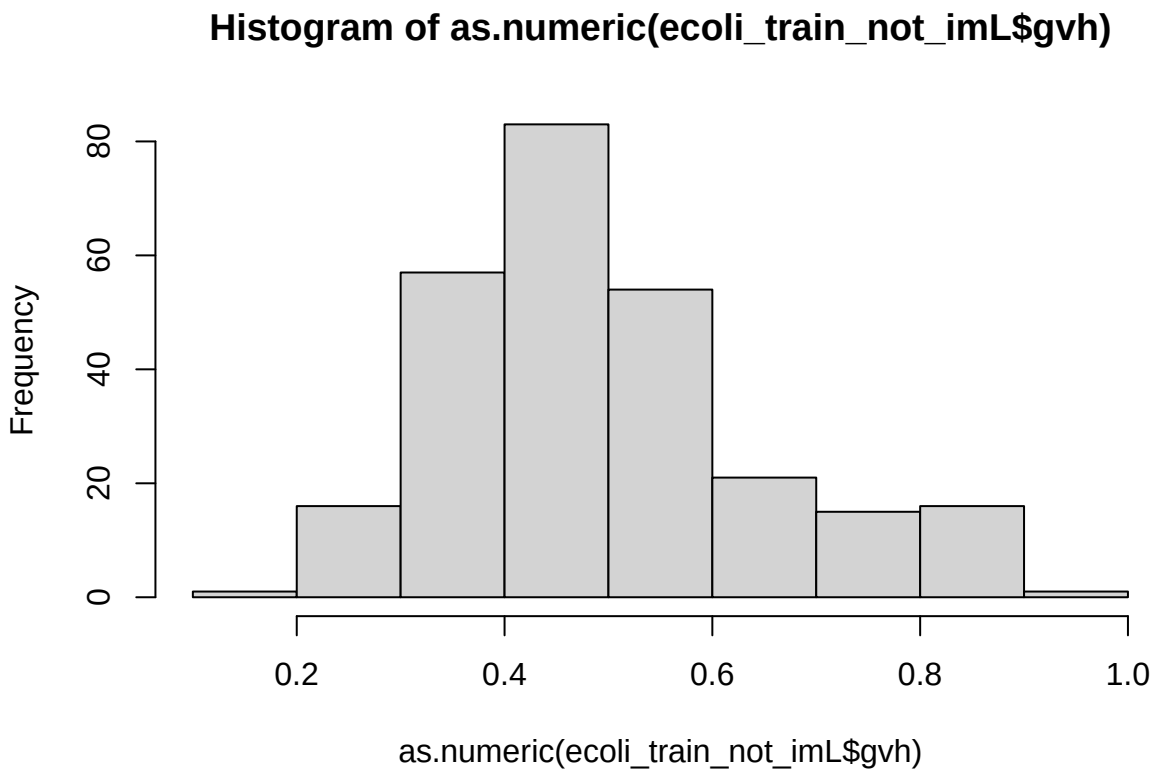
There is only one instance in the imL histogram. No specific gvh score range can be distinguished from this visualisation.

The not-imL histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not imL

```
hist(as.numeric(ecoli_train_imL$gvh))
```




```
hist(as.numeric(ecoli_train_not_imL$gvh))
```

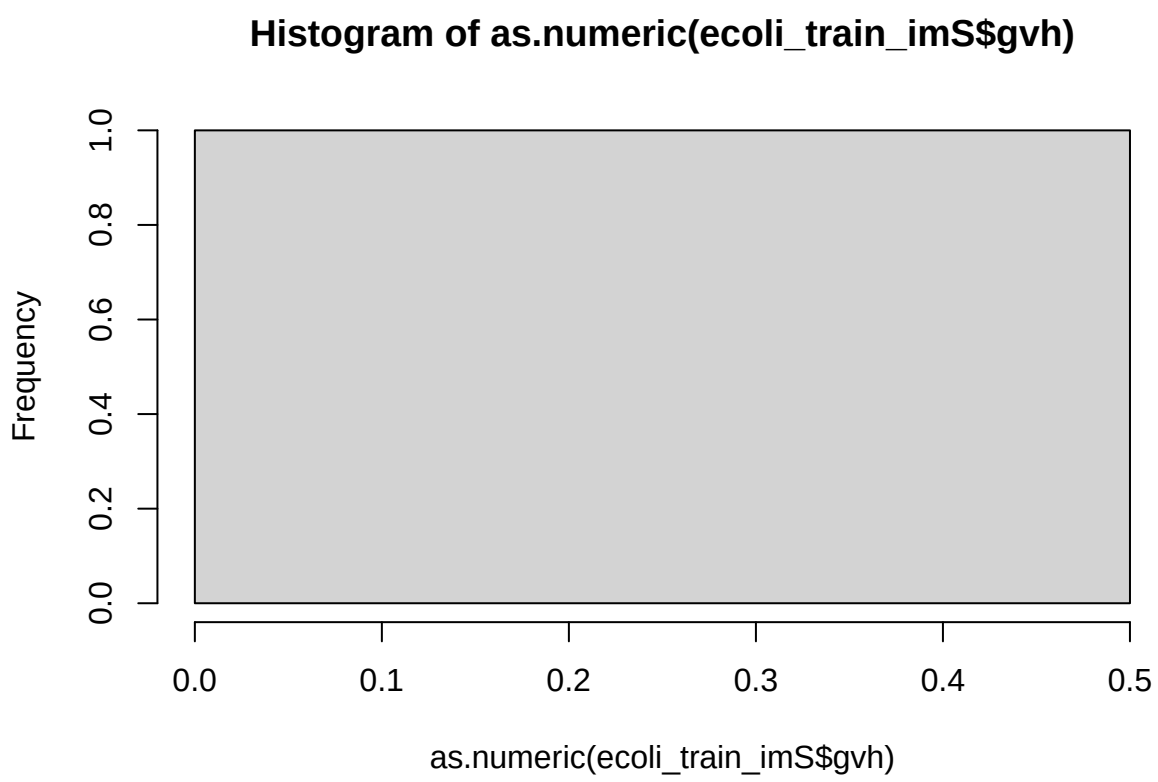


16. gvh vs. imS and not imS

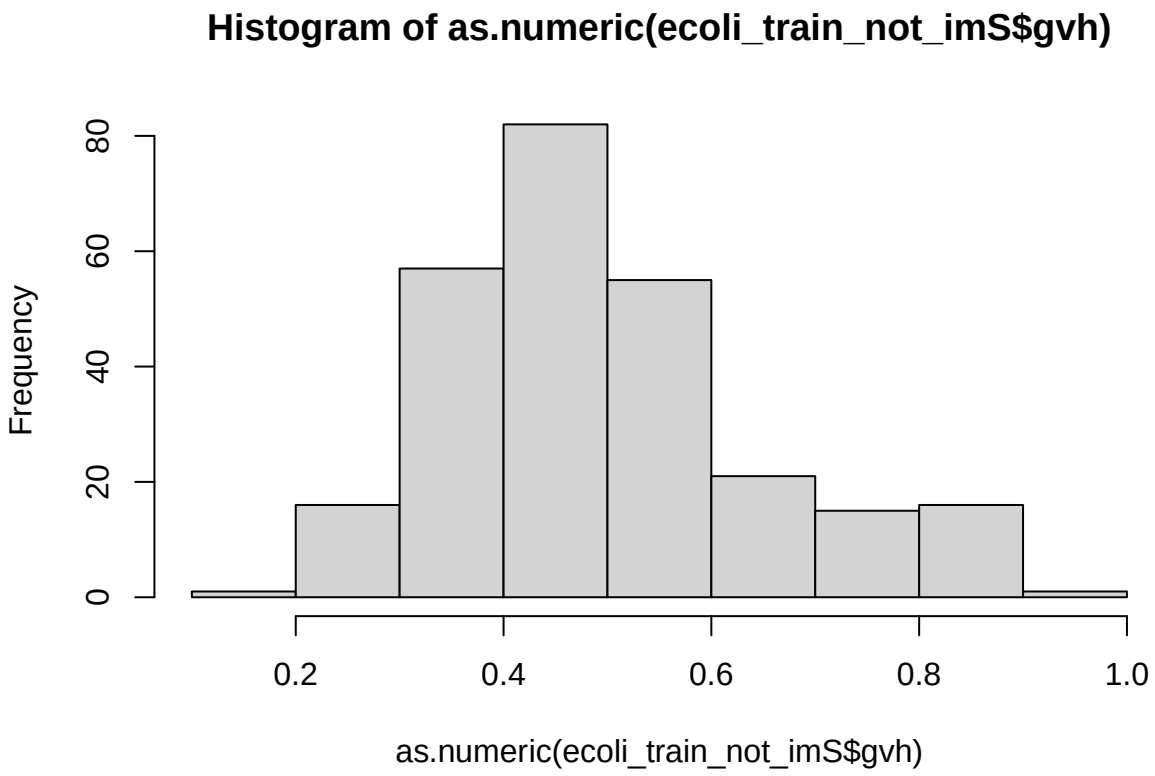
There is only one instance in the imS histogram. No specific gvh score range can be distinguished from this visualisation.

The not-imS histogram shows that when the gvh score is within 0.4 and 0.5, the predicted locsite is generally not imS.

```
hist(as.numeric(ecoli_train_imS$gvh))
```




```
hist(as.numeric(ecoli_train_not_imS$gvh))
```

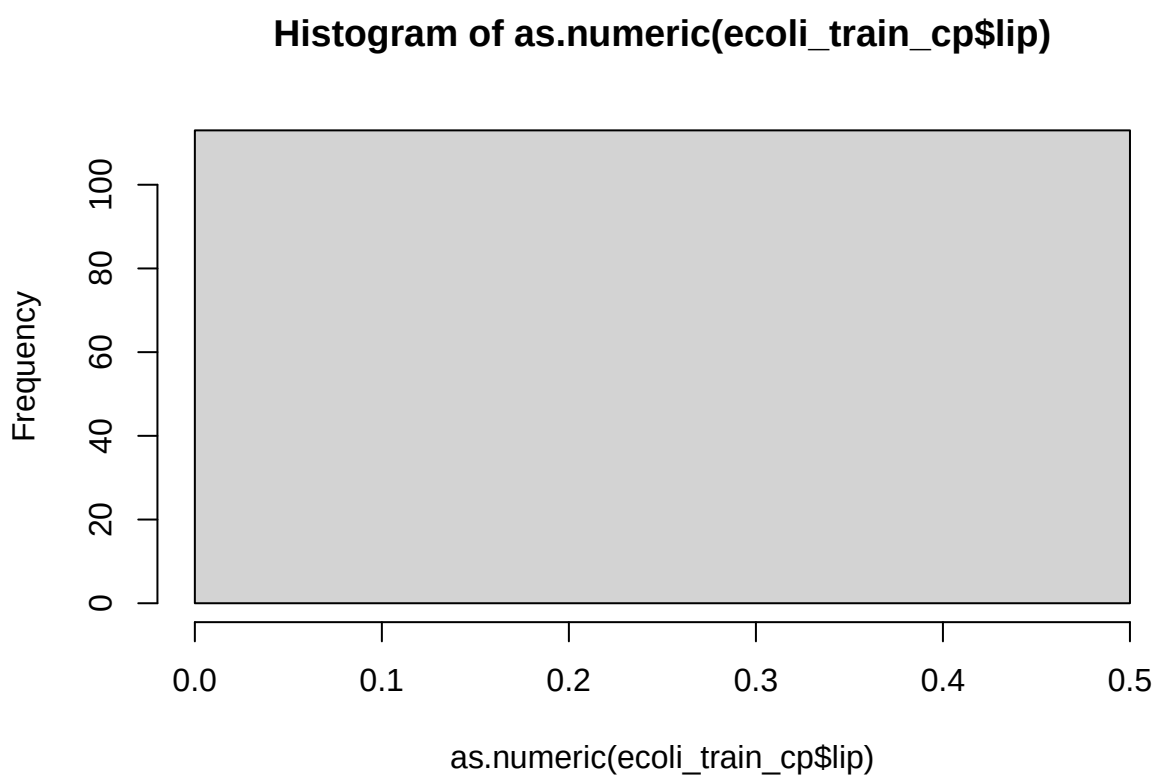


17. lip vs. cp and not cp

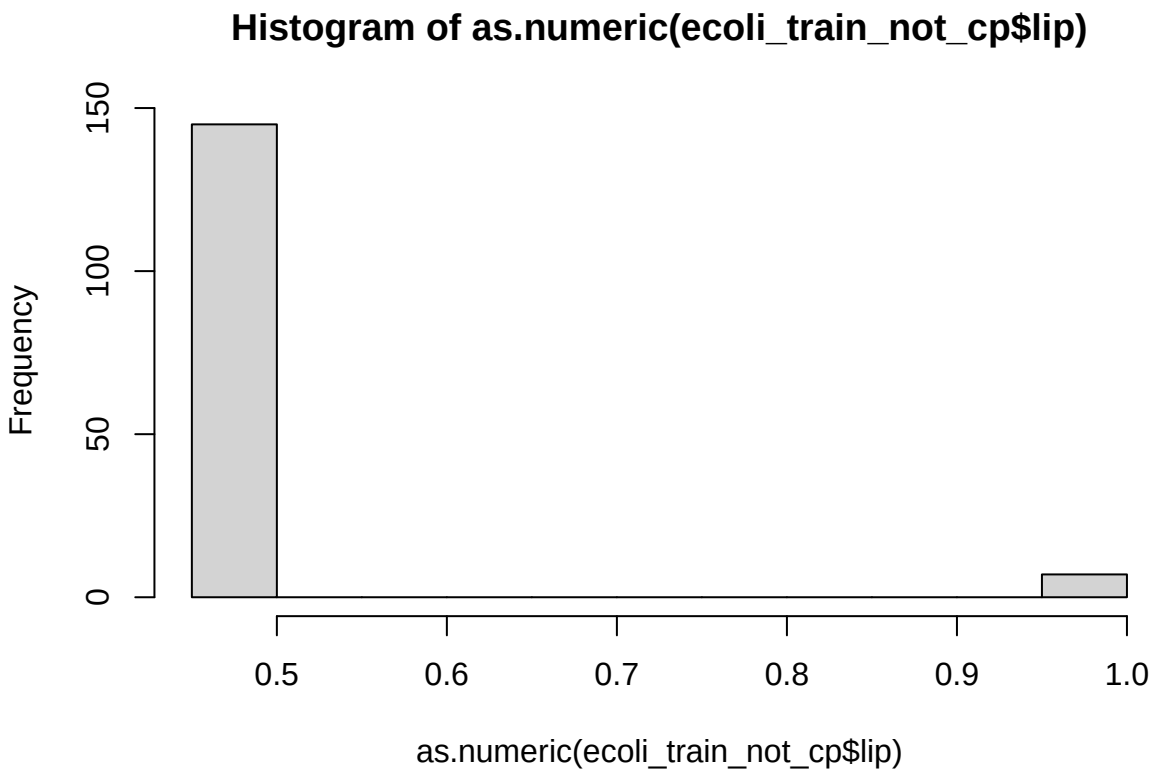
There is only one instance in the cp histogram. No specific lip score range can be distinguished from this visualisation.

The not-cp histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not cp.

```
hist(as.numeric(ecoli_train_cp$lip))
```




```
hist(as.numeric(ecoli_train_not_cp$lip))
```

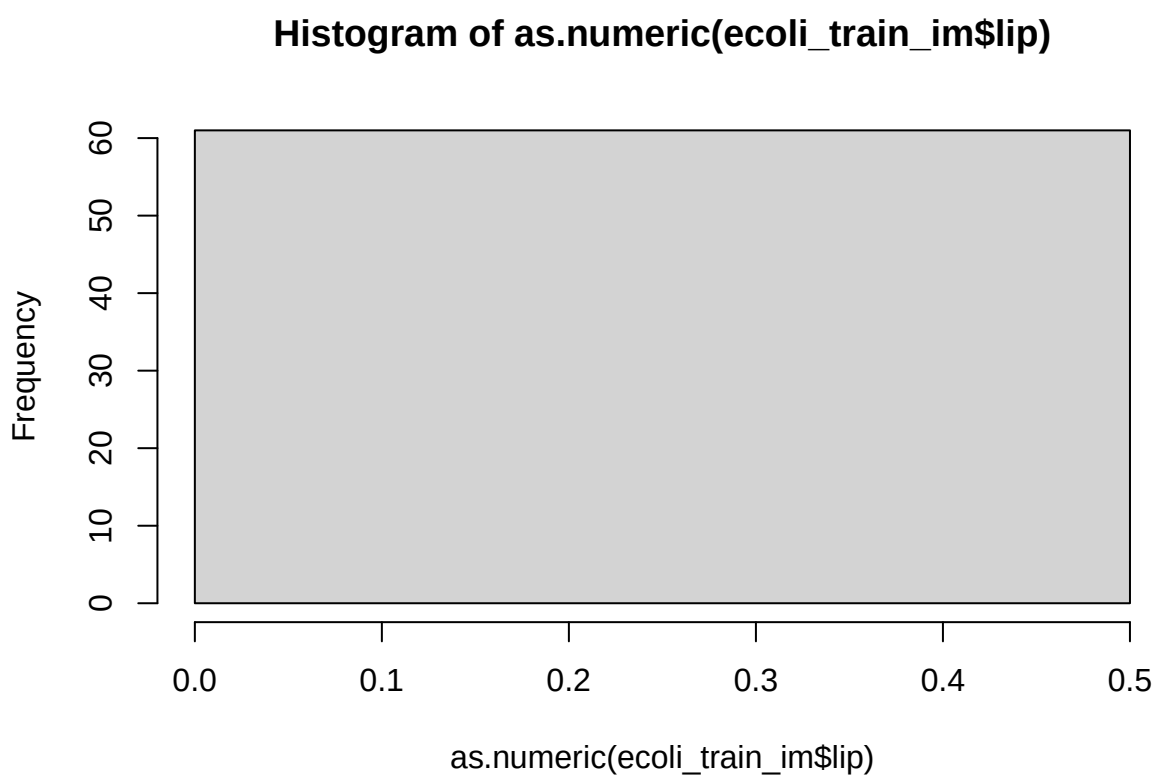


18. lip vs. im and not im

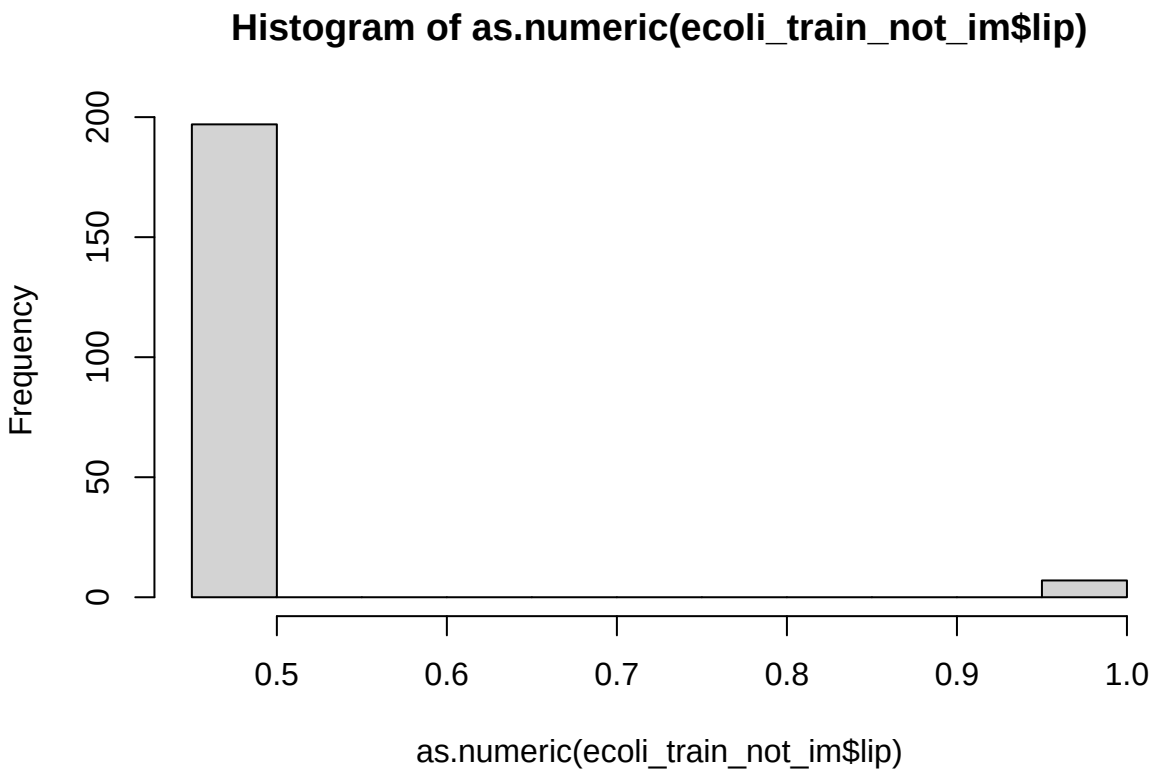
There is only one instance in the im histogram. No specific lip score range can be distinguished from this visualisation.

The not-im histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not im.

```
hist(as.numeric(ecoli_train_im$lip))
```




```
hist(as.numeric(ecoli_train_not_im$lip))
```

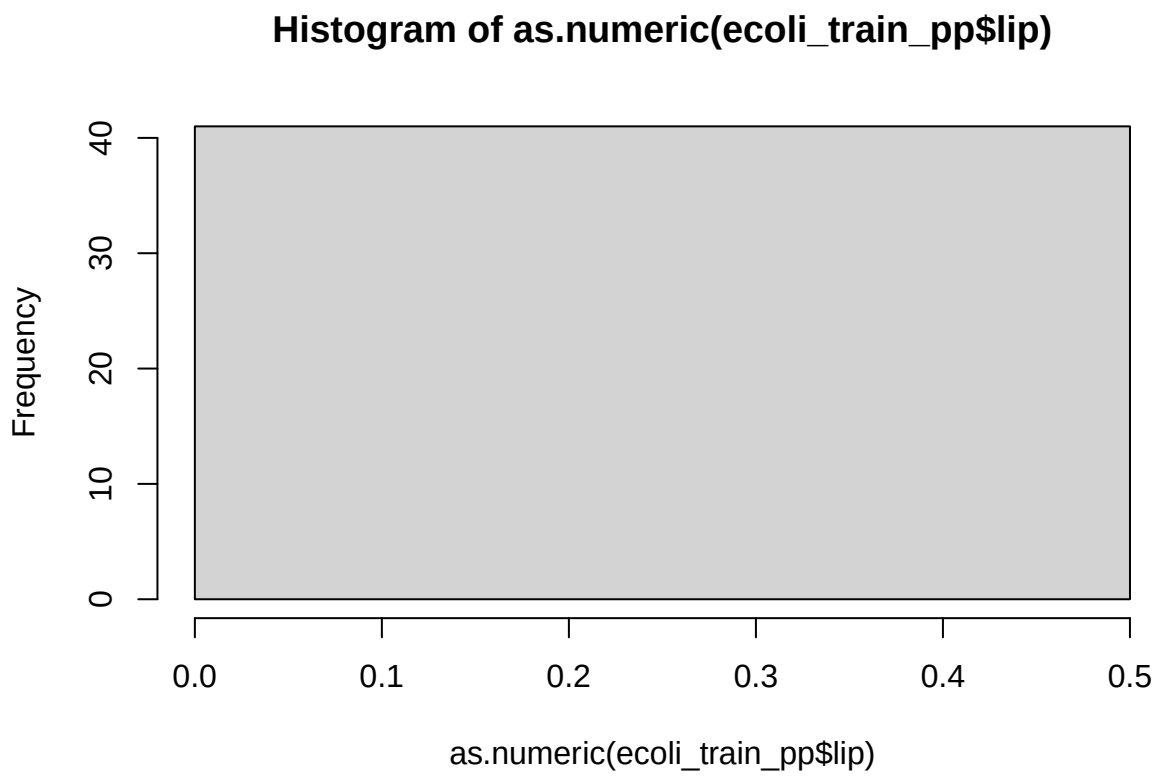


19. lip vs. pp and not pp

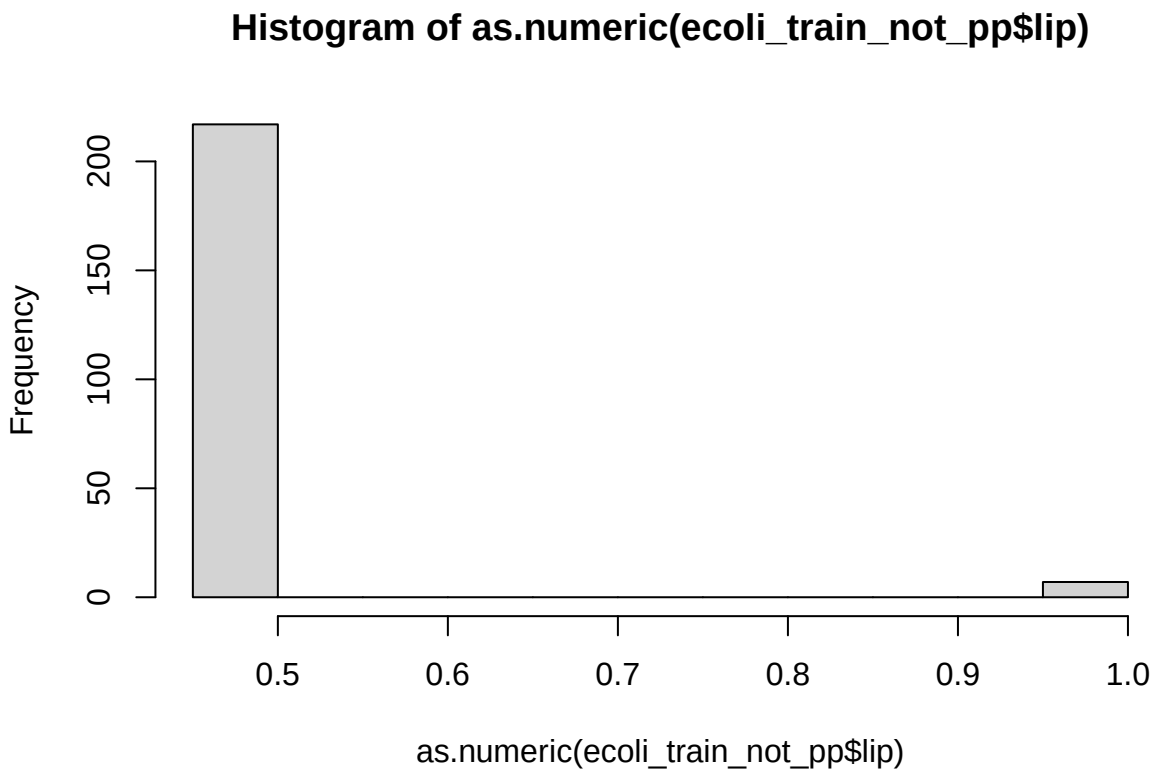
There is only one instance in the pp histogram. No specific lip score range can be distinguished from this visualisation.

The not-pp histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not pp.

```
hist(as.numeric(ecoli_train_pp$lip))
```




```
hist(as.numeric(ecoli_train_not_pp$lip))
```

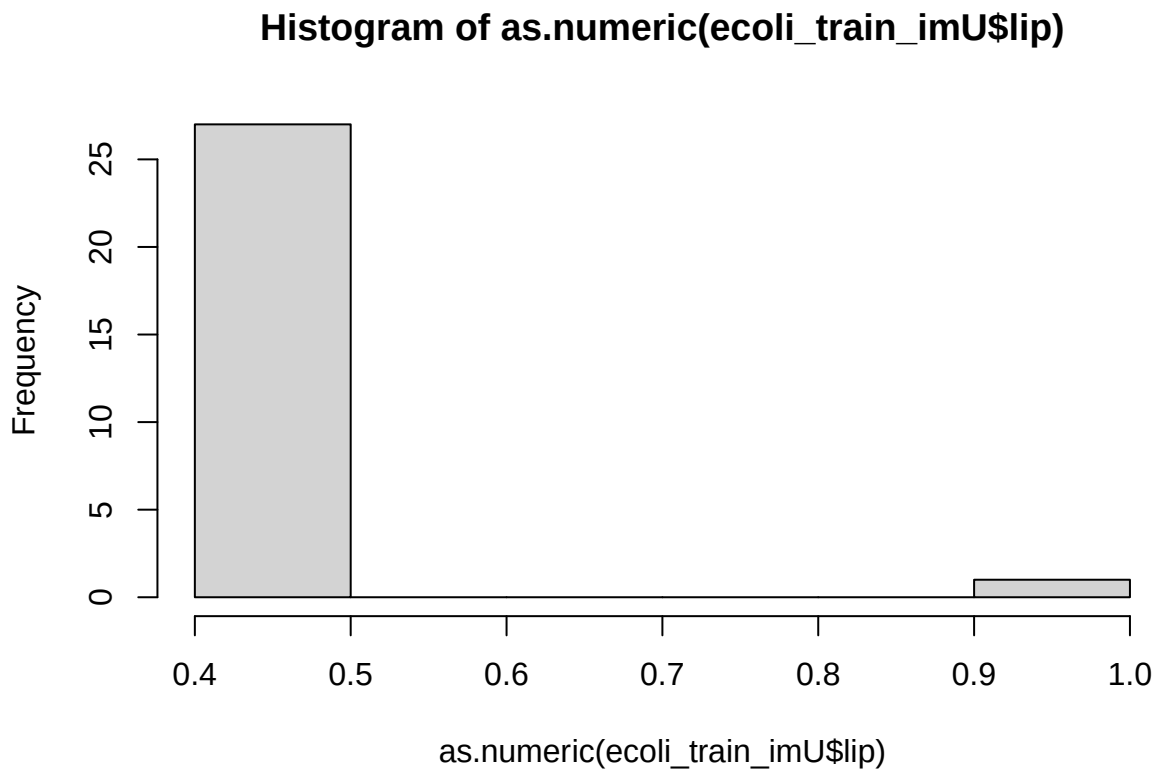


20. lip vs. imU and not imU

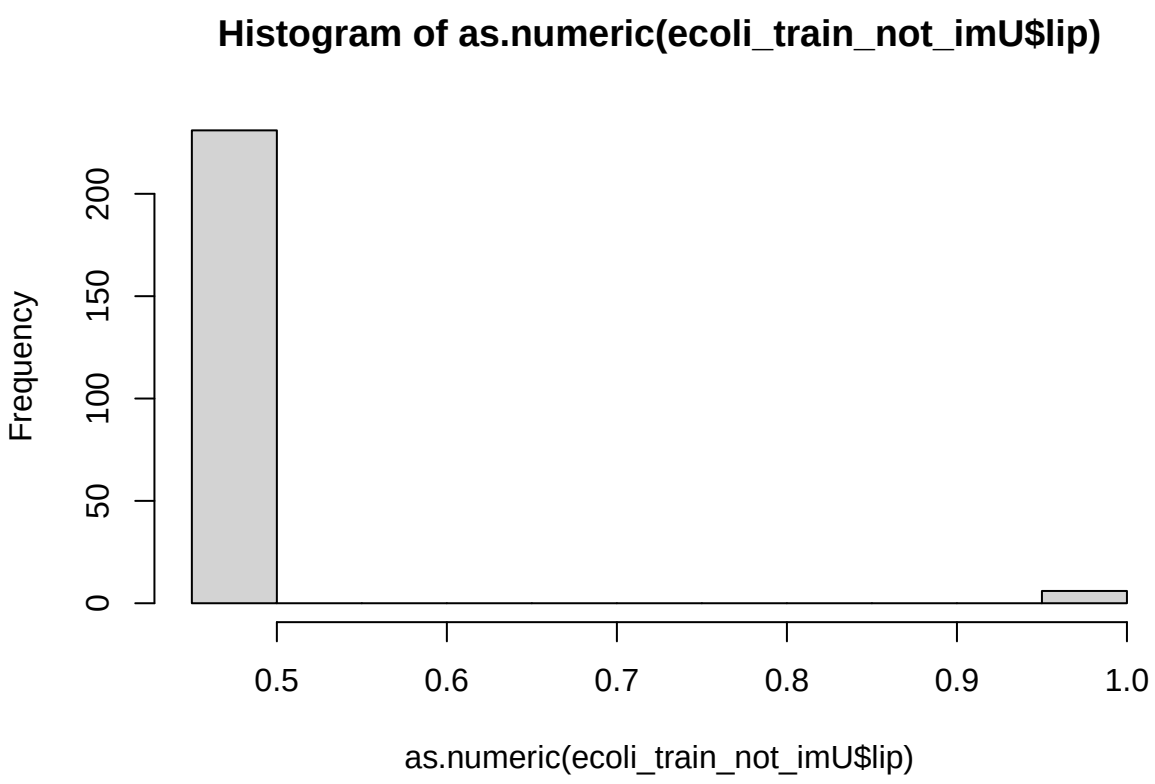
The imU histogram shows that imU is most frequently predicted when lip score ranges from 0.4 to 0.5.

The not-imU histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not imU.

```
hist(as.numeric(ecoli_train_imU$lip))
```




```
hist(as.numeric(ecoli_train_not_imU$lip))
```

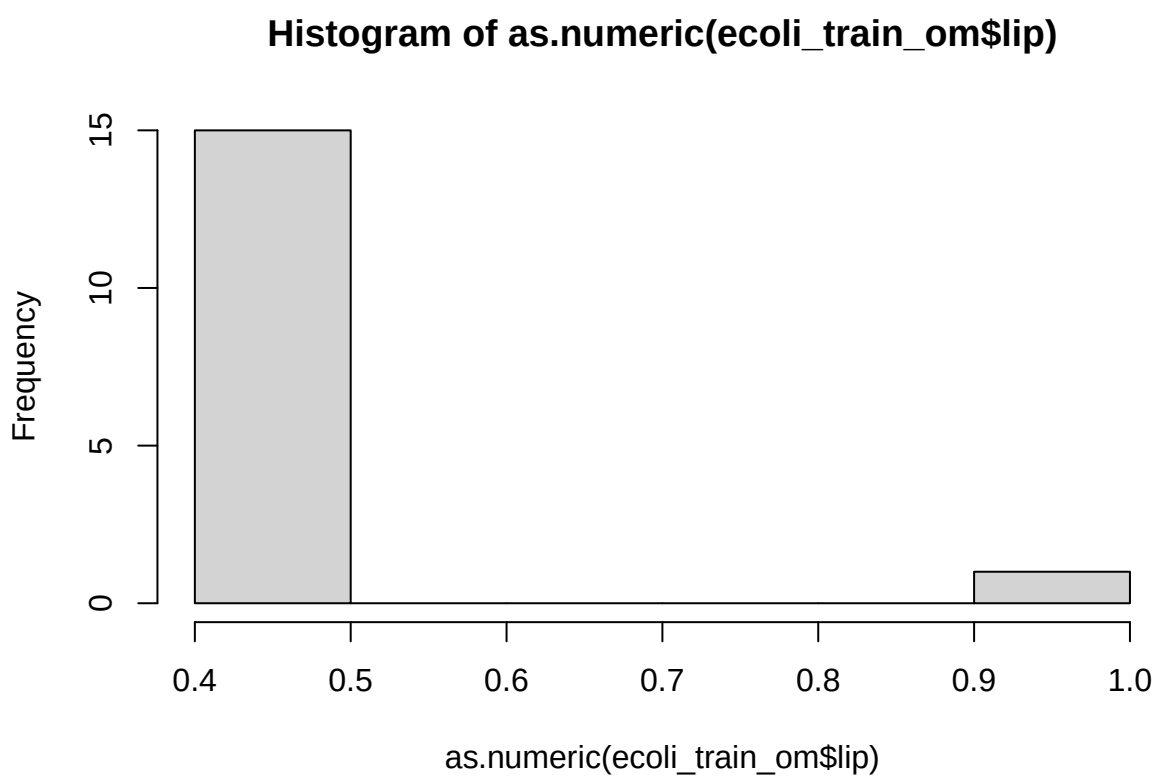


21. lip vs. om and not om

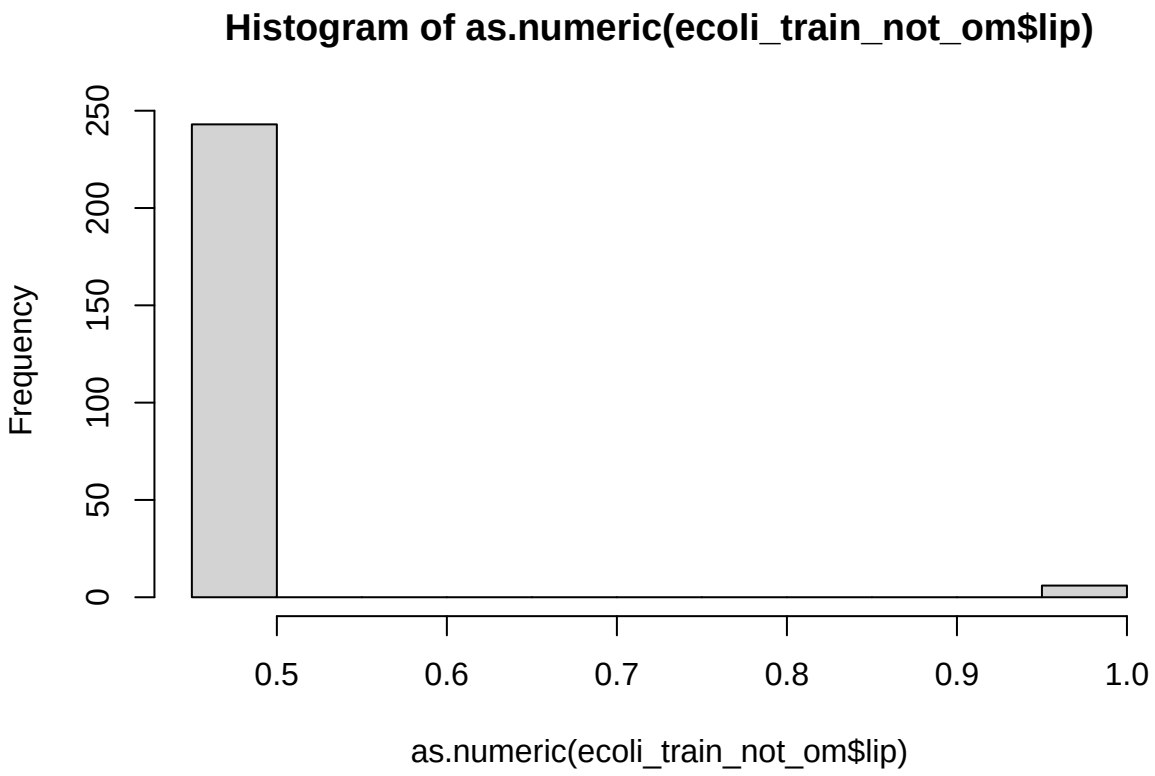
The om histogram shows that om is most frequently predicted when lip score ranges from 0.4 to 0.5.

The not-om histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not om.

```
hist(as.numeric(ecoli_train_om$lip))
```




```
hist(as.numeric(ecoli_train_not_om$lip))
```

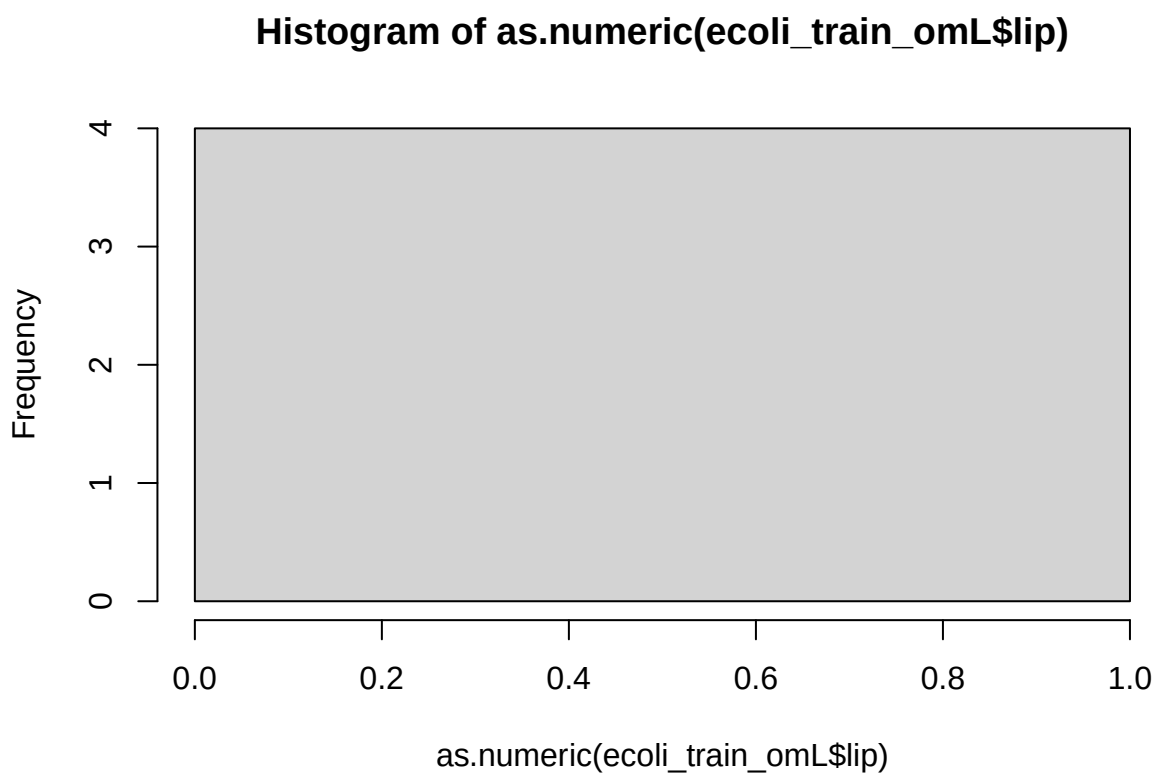


22. lip vs. omL and not omL

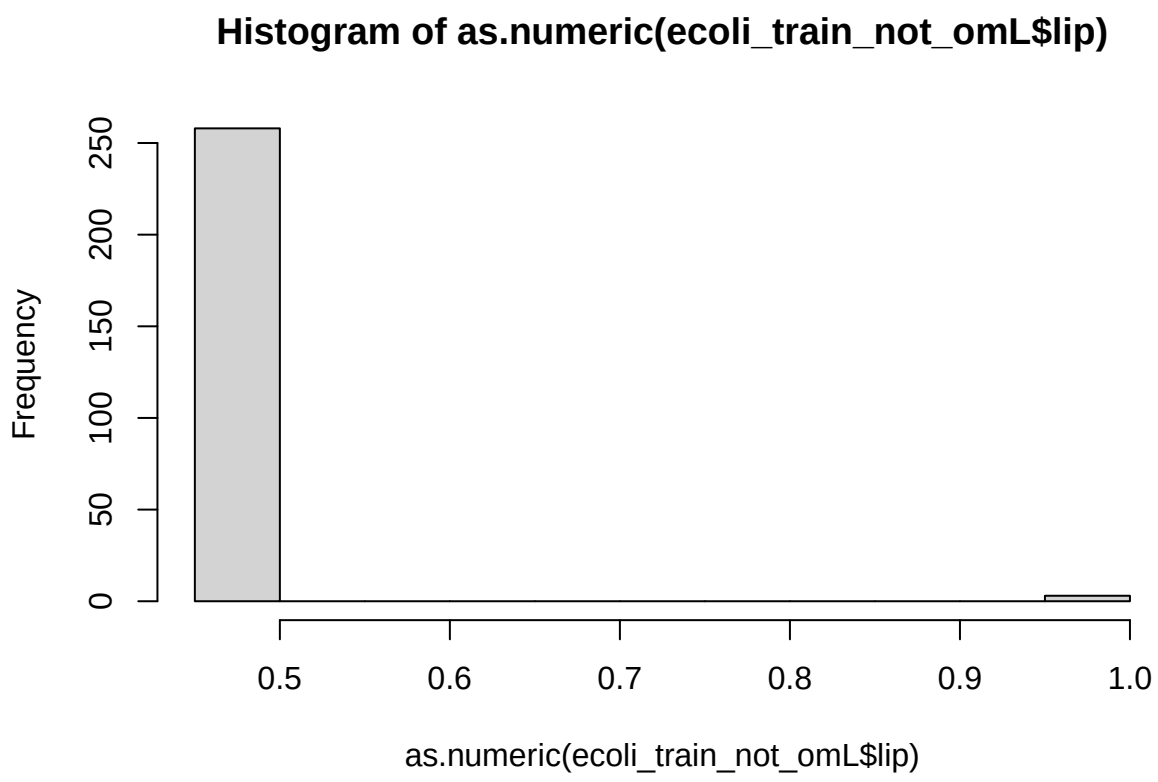
The omL histogram shows that omL is most frequently predicted when lip score ranges from 0.0 to 1.0.

The not-omL histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not omL.

```
hist(as.numeric(ecoli_train_omL$lip))
```




```
hist(as.numeric(ecoli_train_not_omL$lip))
```

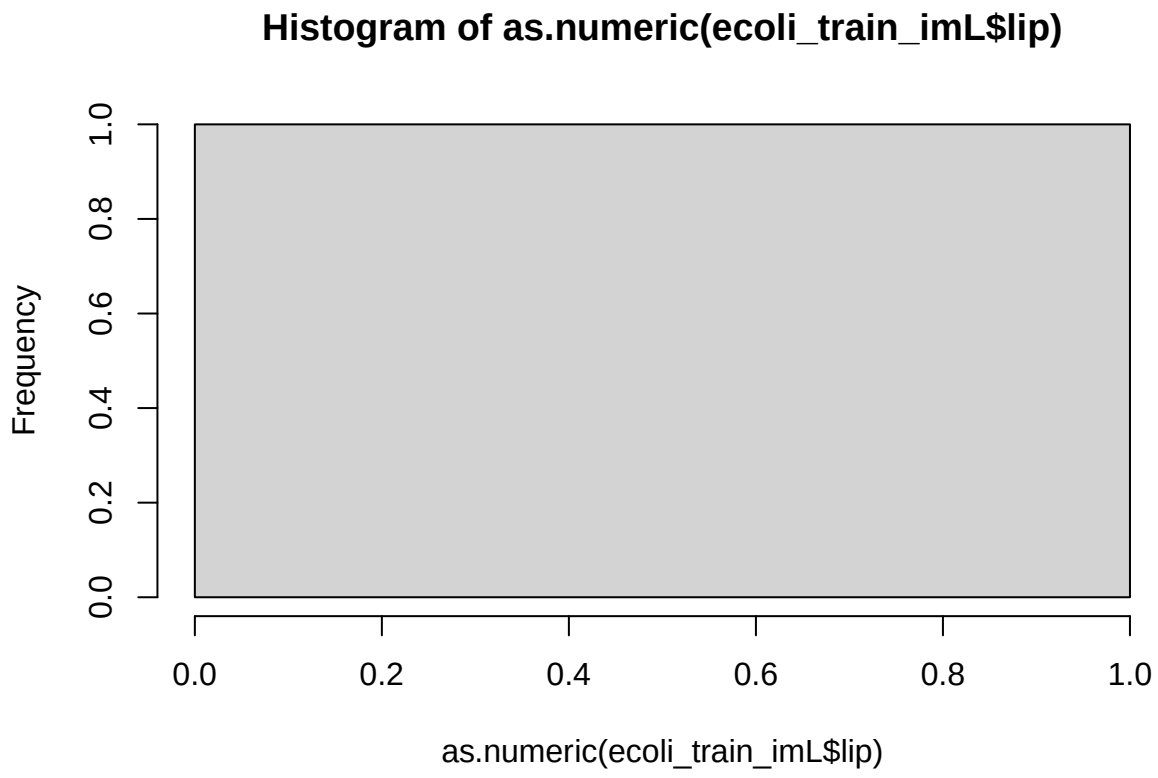


23. lip vs. imL and not imL

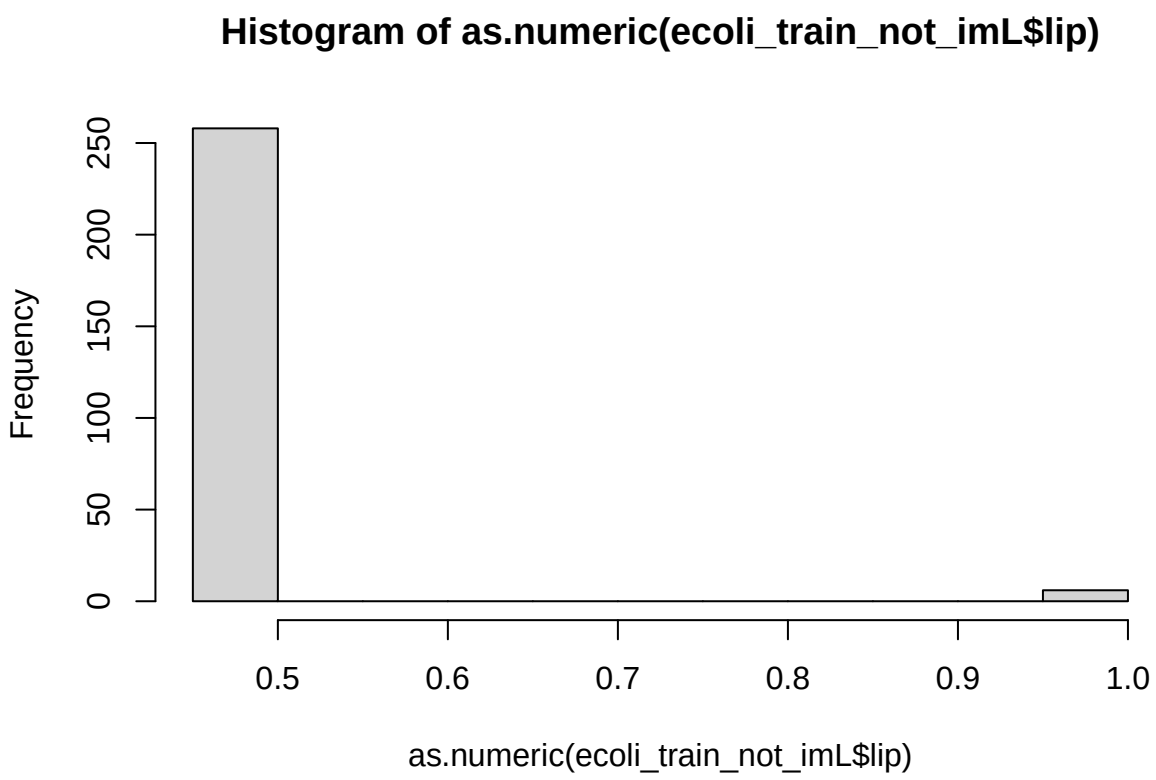
There is only one instance in the imL histogram. No specific lip score range can be distinguished from this visualisation.

The not-imL histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not imL.

```
hist(as.numeric(ecoli_train_imL$lip))
```




```
hist(as.numeric(ecoli_train_not_imL$lip))
```

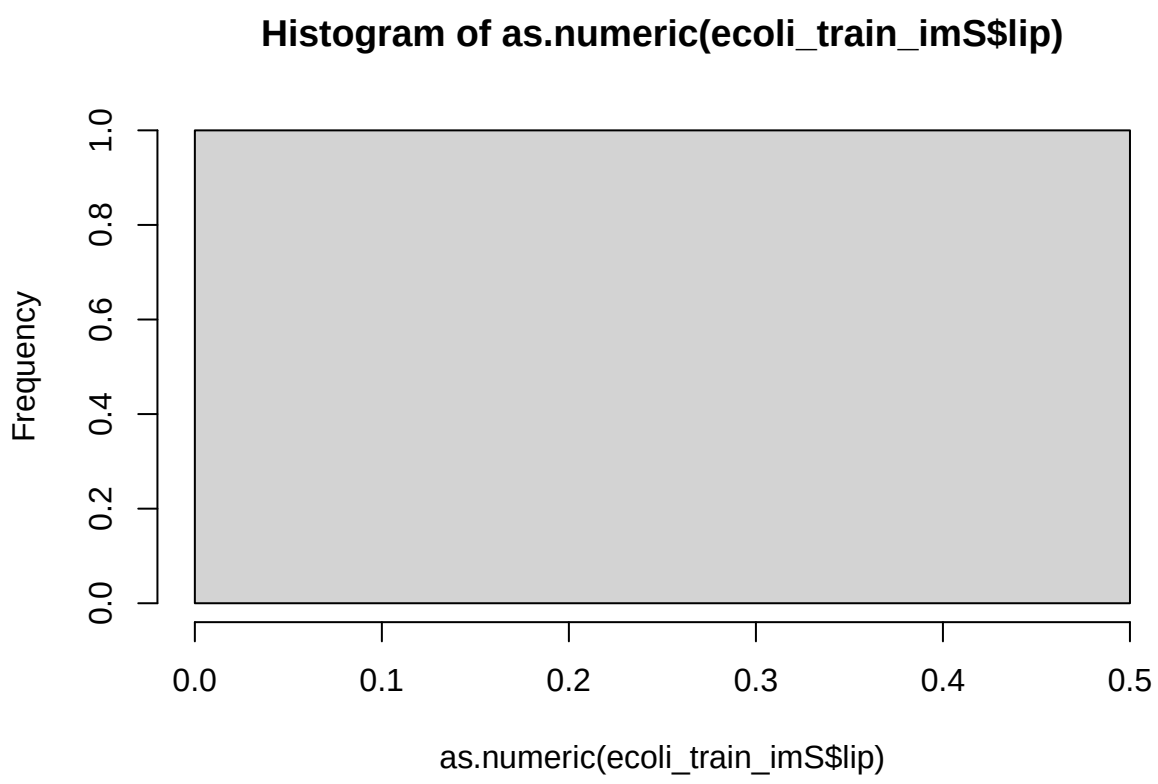


24. lip vs. imS and not imS

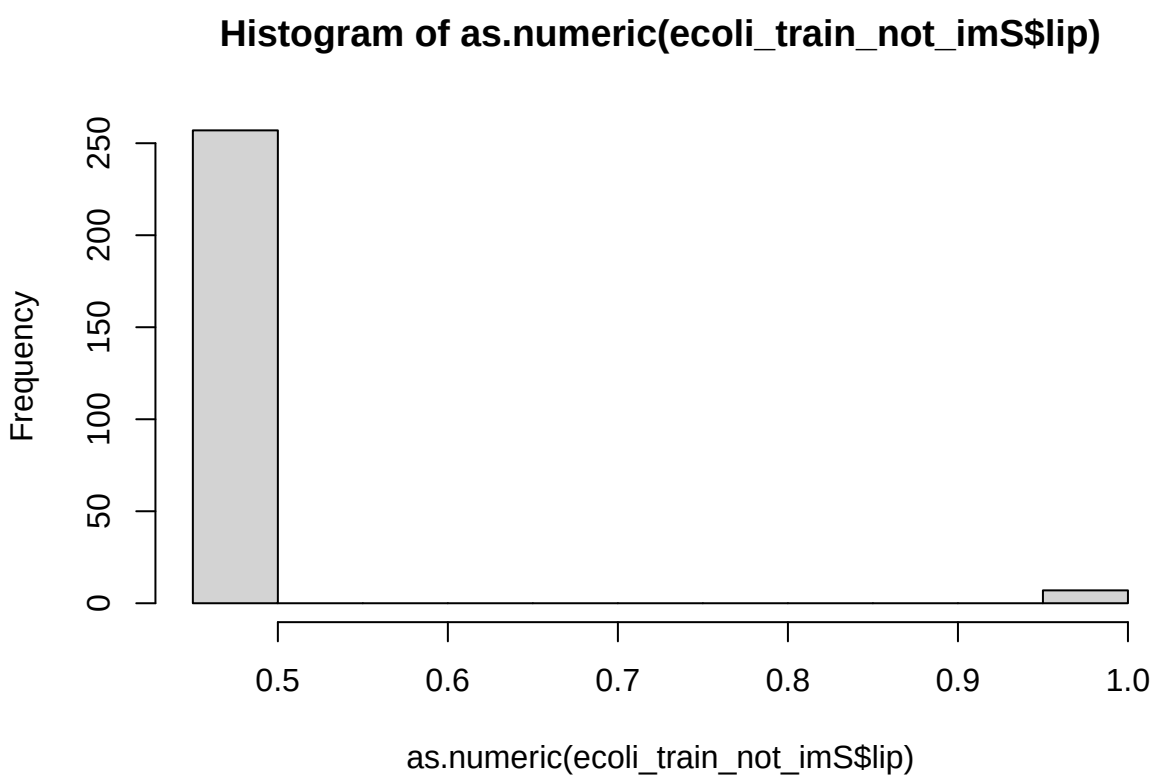
There is only one instance in the imS histogram. No specific lip score range can be distinguished from this visualisation.

The not-imS histogram shows that when the lip score is less than 0.5, the predicted locsite is generally not imS.

```
hist(as.numeric(ecoli_train_imS$lip))
```




```
hist(as.numeric(ecoli_train_not_imS$lip))
```

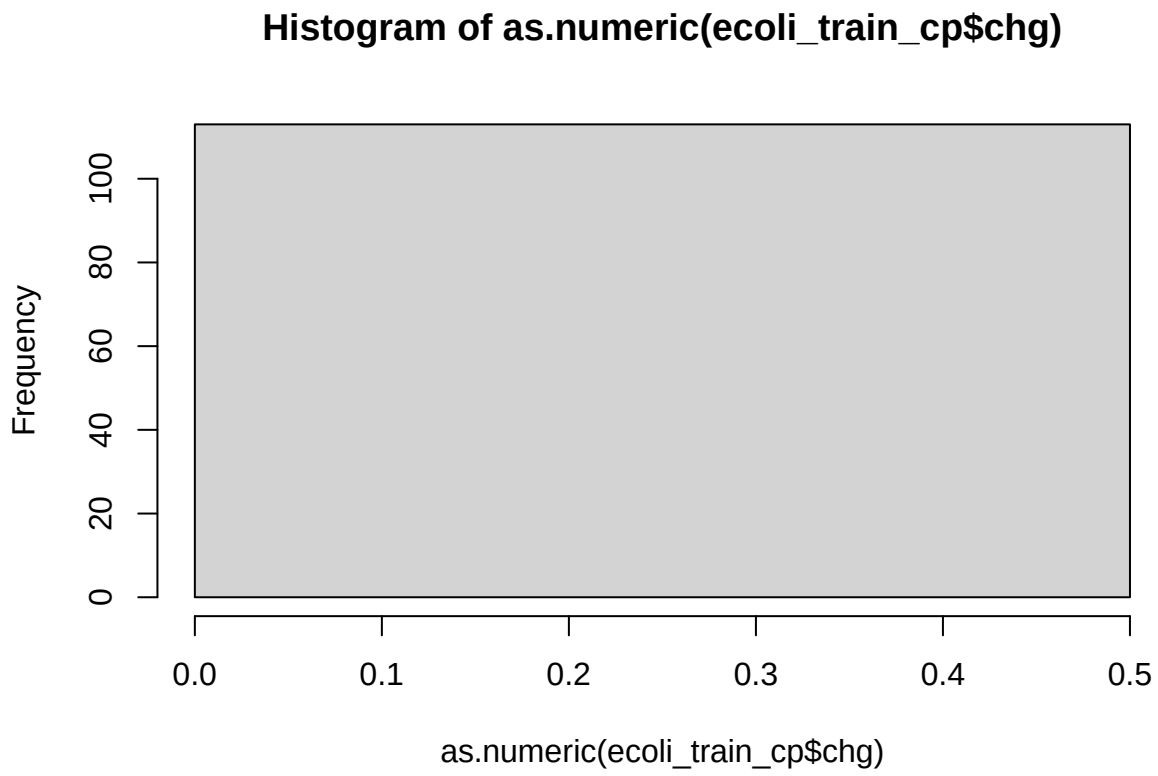


25. chg vs. cp and not cp

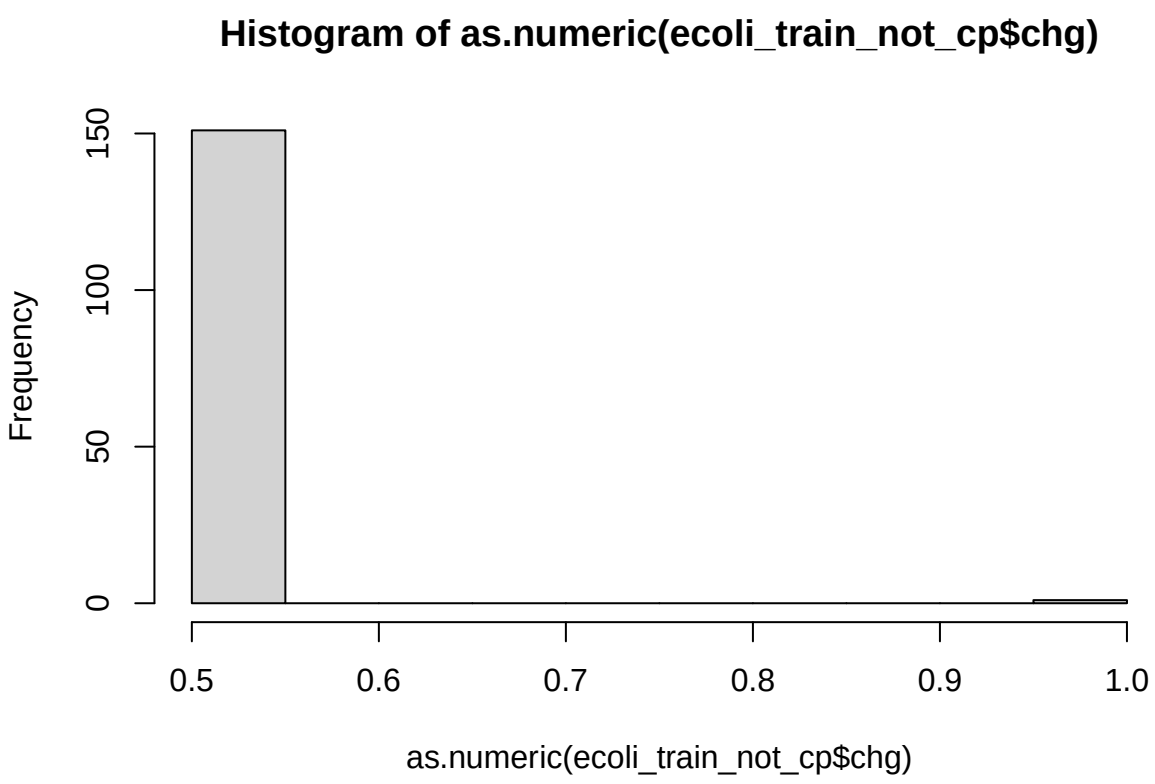
The cp histogram shows that cp is most frequently predicted when chg score ranges from 0.0 to 0.5.

The not-cp histogram shows that when the chg score is less than 0.55, the predicted locsite is generally not cp.

```
hist(as.numeric(ecoli_train_cp$chg))
```




```
hist(as.numeric(ecoli_train_not_cp$chg))
```

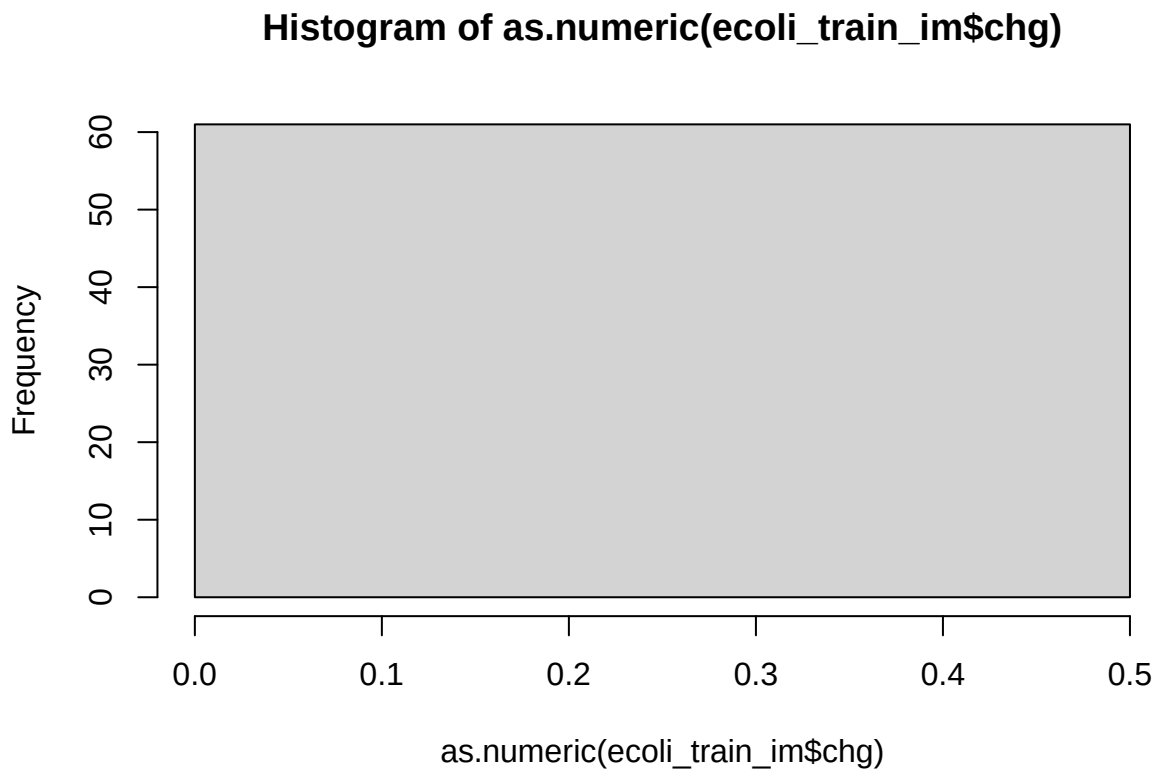


26. chg vs. im and not im

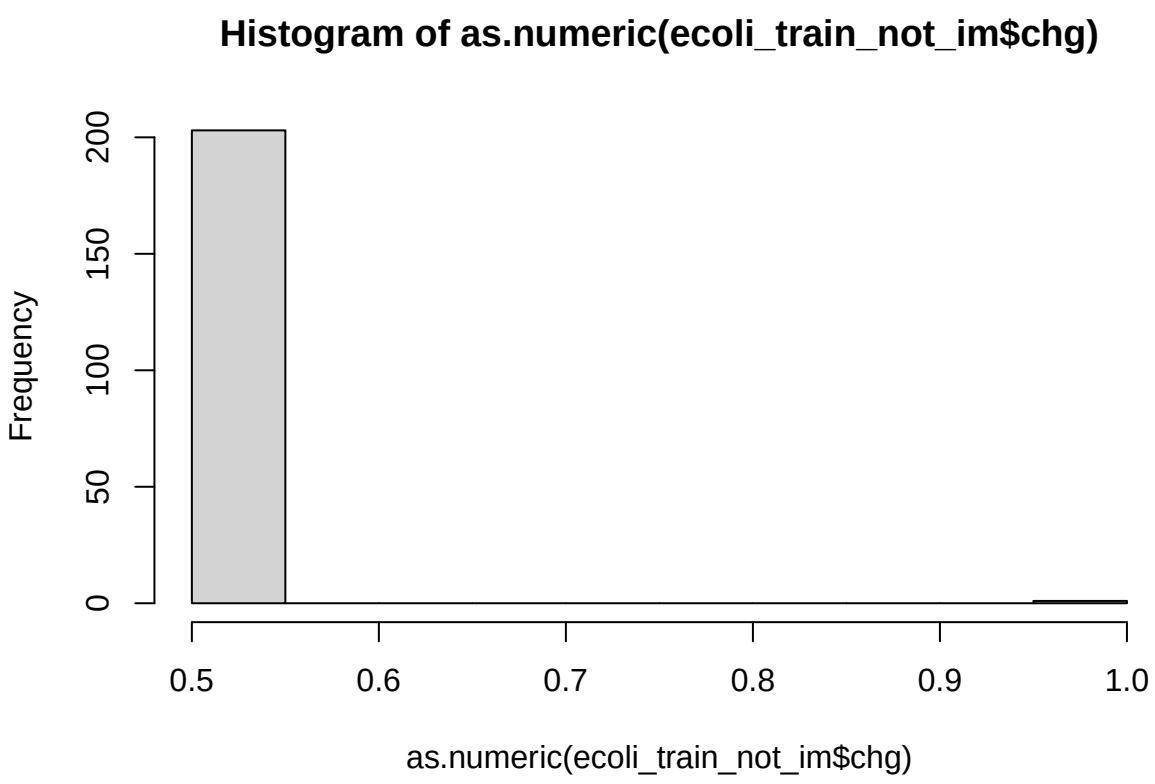
The im histogram shows that im is most frequently predicted when chg score ranges from 0.0 to 0.5.

The not-im histogram shows that when the chg score is less than 0.55, the predicted locsite is generally not im.

```
hist(as.numeric(ecoli_train_im$chg))
```




```
hist(as.numeric(ecoli_train_not_im$chg))
```

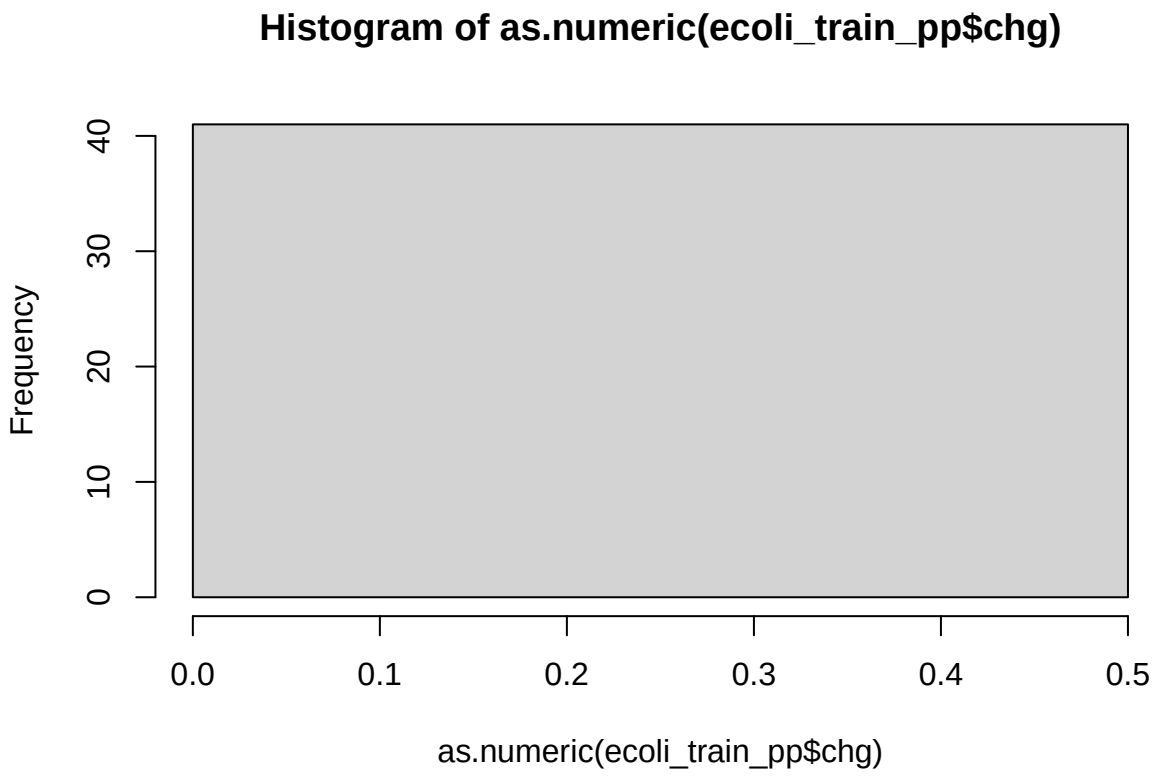


27. chg vs. pp and not pp

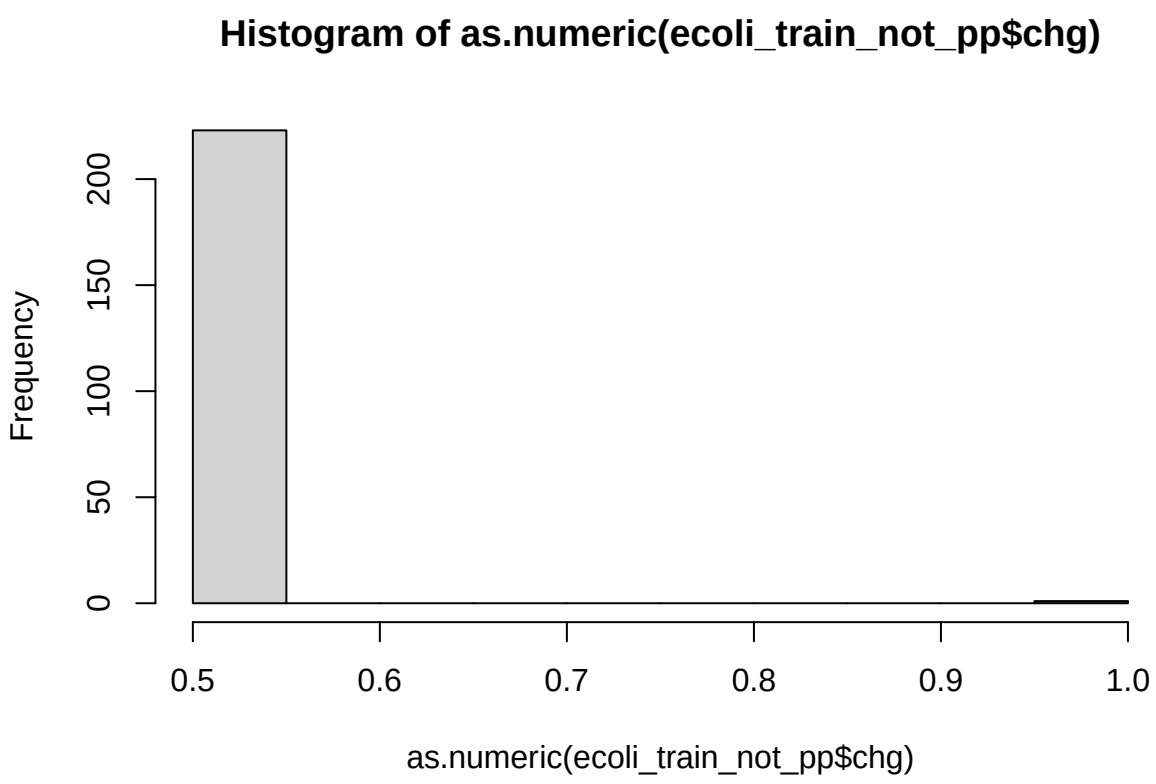
The pp histogram shows that pp is most frequently predicted when chg score ranges from 0.0 to 0.5.

The not-pp histogram shows that when the chg score is less than 0.55, the predicted locsite is generally not pp.

```
hist(as.numeric(ecoli_train_pp$chg))
```




```
hist(as.numeric(ecoli_train_not_pp$chg))
```

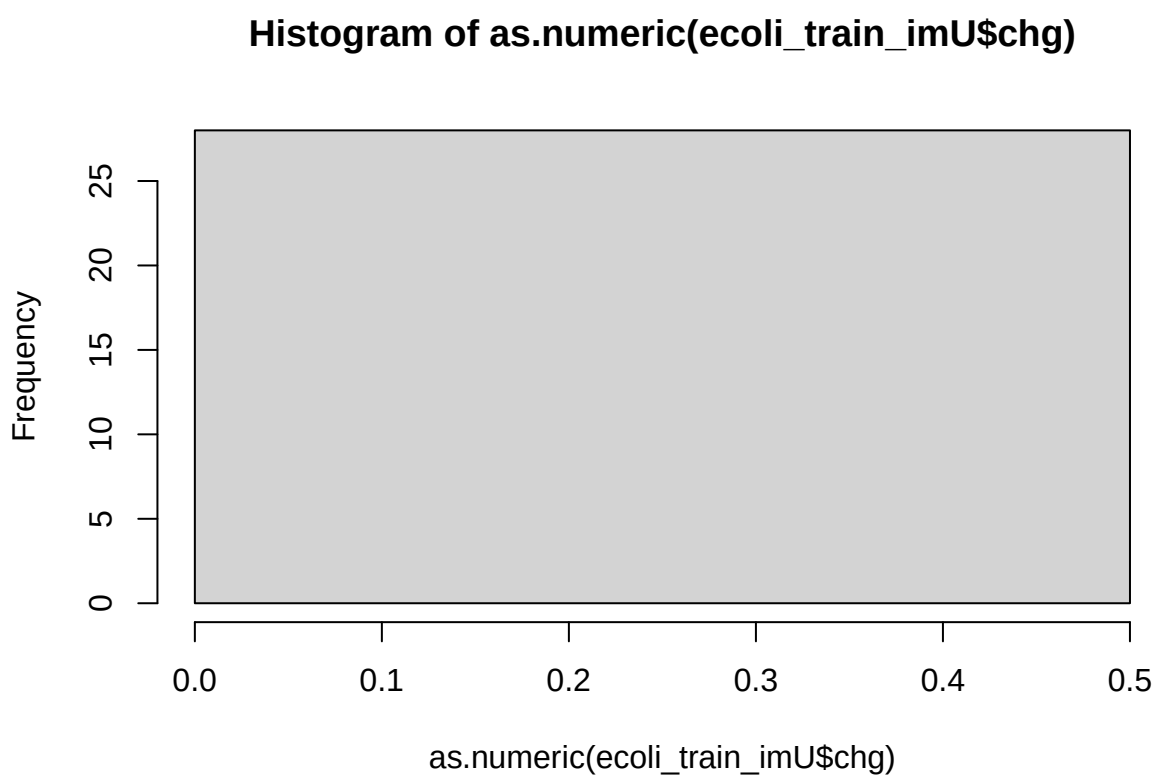


28. chg vs. imU and not imU

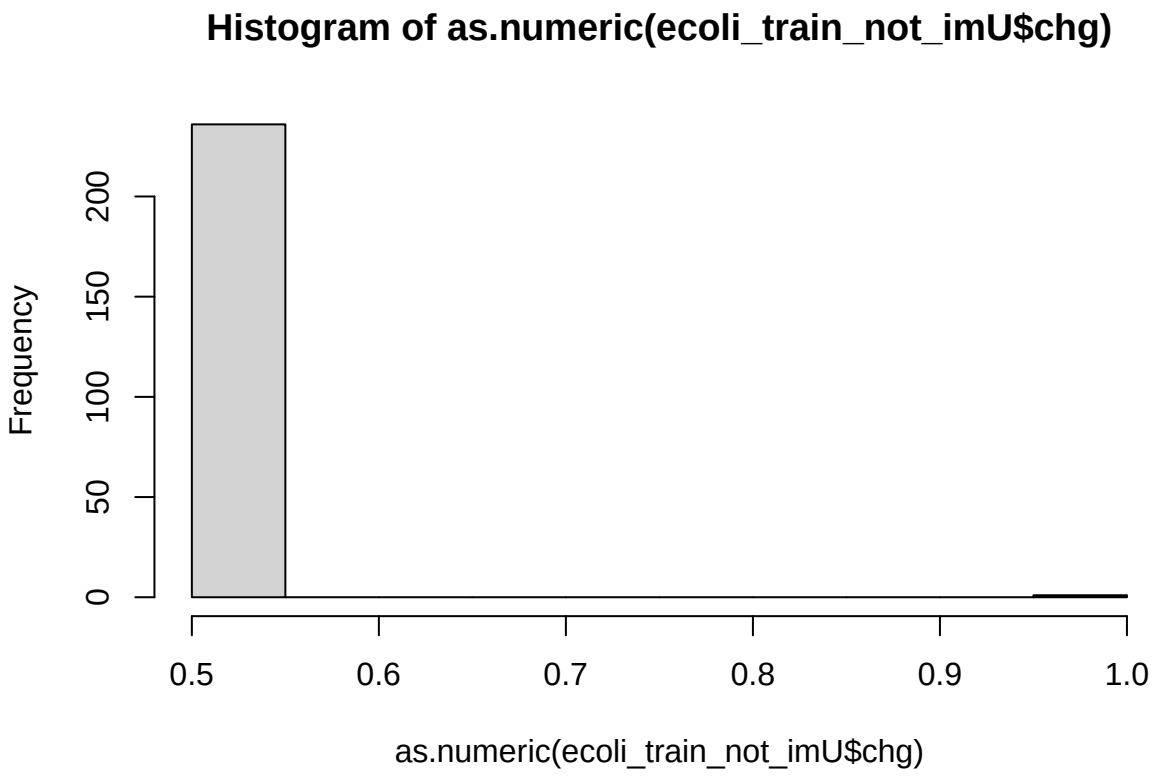
The imU histogram shows that imU is most frequently predicted when chg score ranges from 0.0 to 0.5.

The not-imU histogram shows that when the chg score is less than 0.55, the predicted locsite is generally not imU.

```
hist(as.numeric(ecoli_train_imU$chg))
```




```
hist(as.numeric(ecoli_train_not_imU$chg))
```

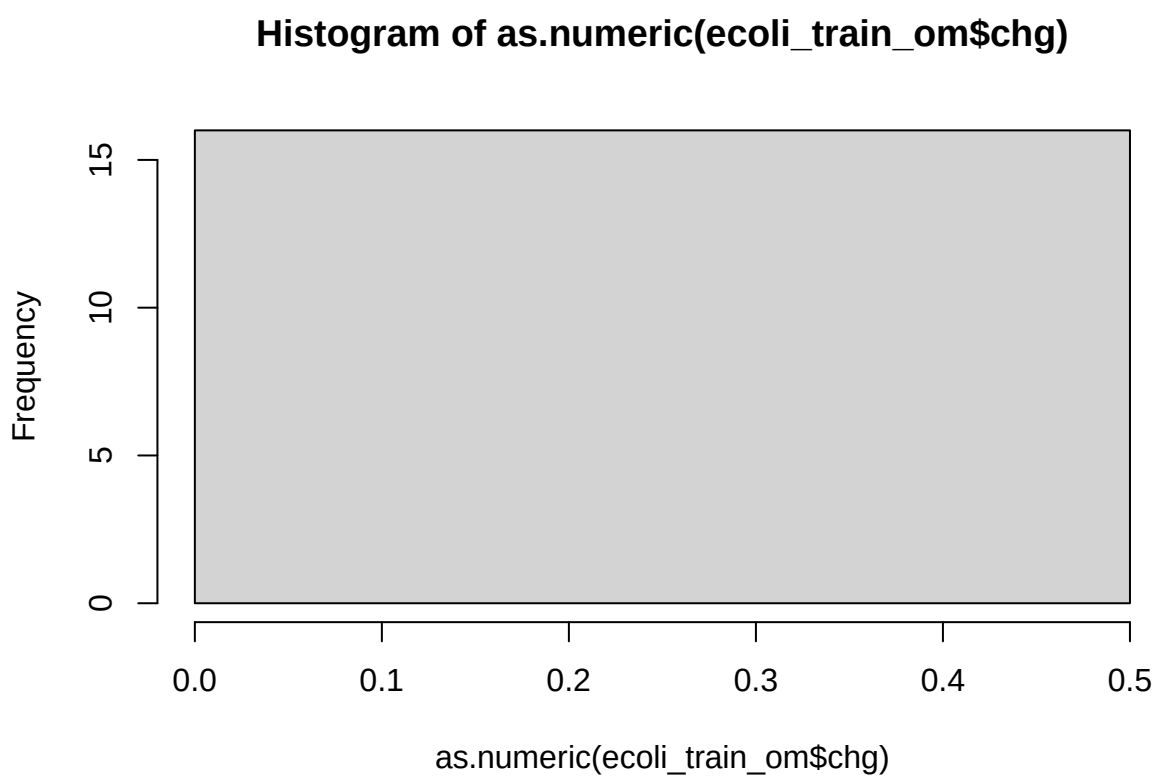


29. chg vs. om and not om

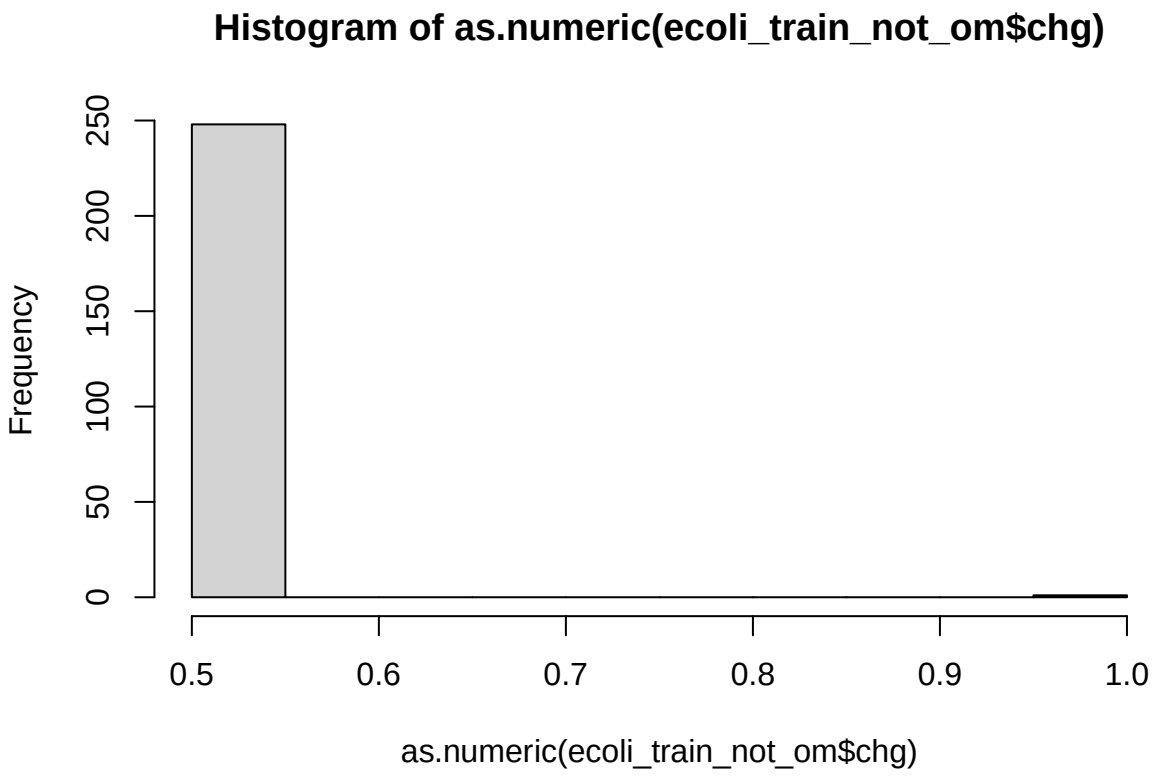
The om histogram shows that om is most frequently predicted when chg score ranges from 0.0 to 0.5.

The not-om histogram shows that when the chg score is less than 0.55, the predicted locsite is generally not om.

```
hist(as.numeric(ecoli_train_om$chg))
```




```
hist(as.numeric(ecoli_train_not_om$chg))
```

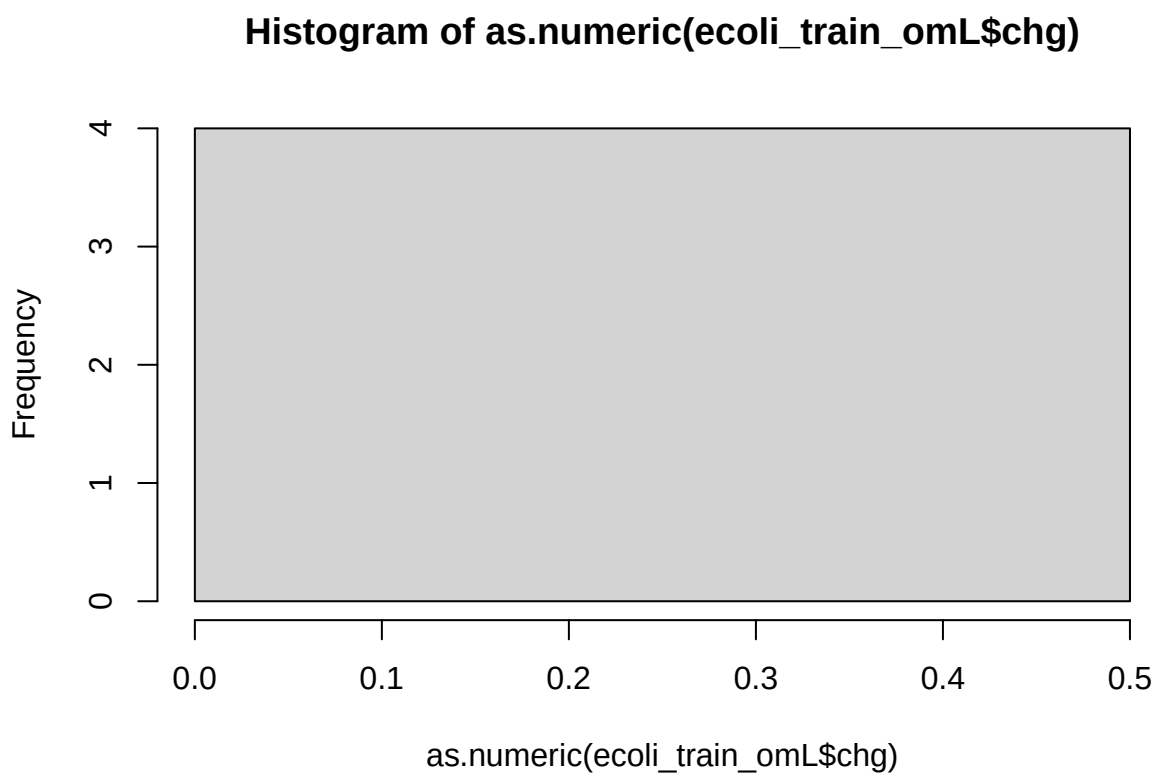


30. chg vs. omL and not omL

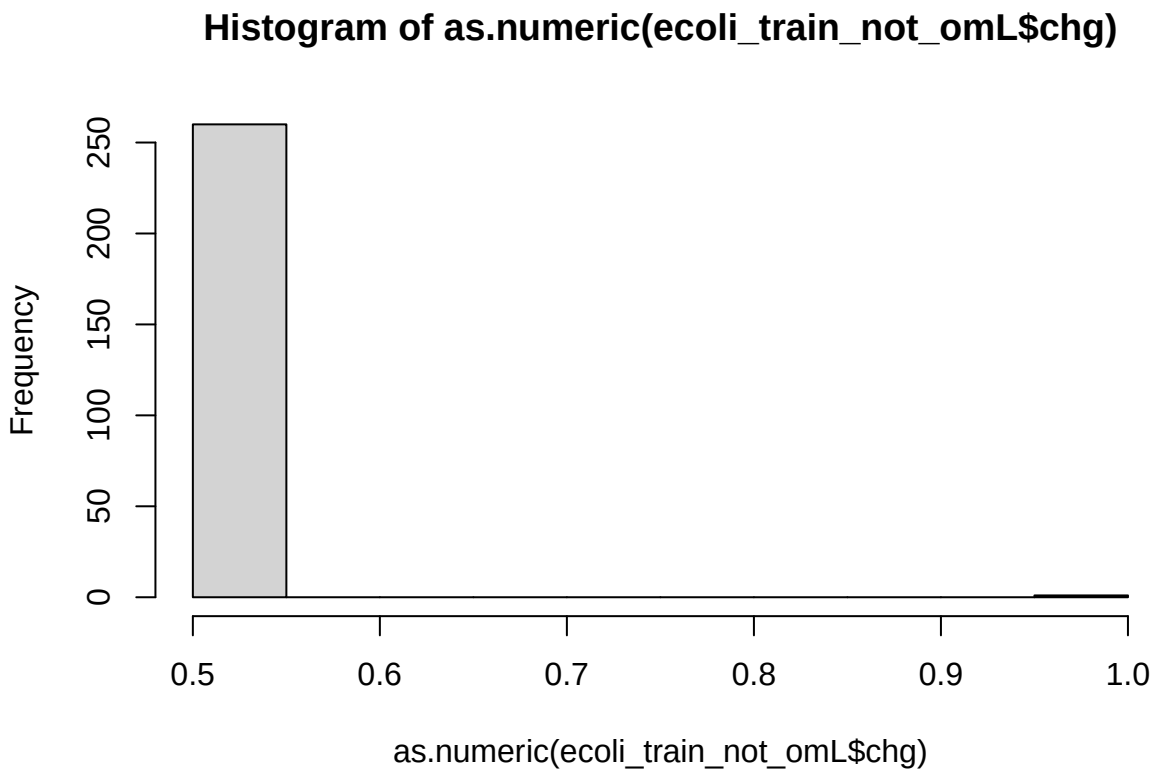
The omL histogram shows that omL is most frequently predicted when chg score ranges from 0.0 to 0.5.

The not-omL histogram shows that when the chg score is less than 0.55, the predicted locsite is generally not omL.

```
hist(as.numeric(ecoli_train_omL$chg))
```




```
hist(as.numeric(ecoli_train_not_omL$chg))
```

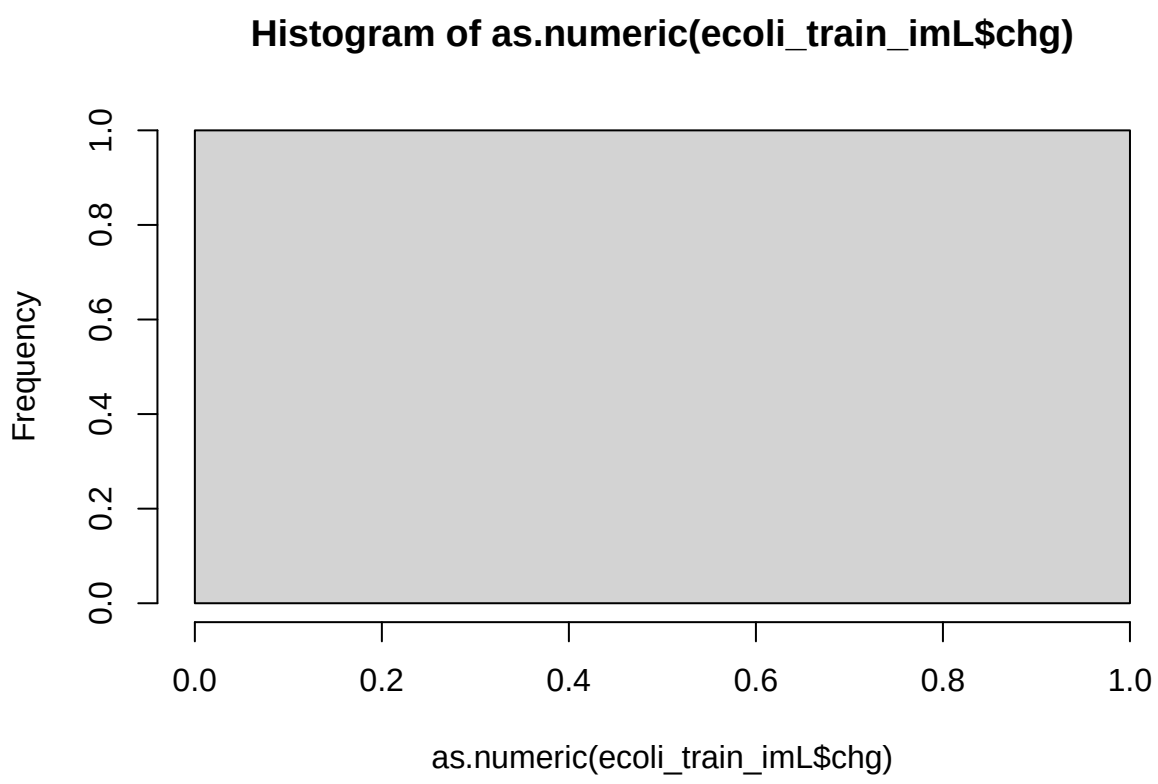


31. chg vs. imL and not imL

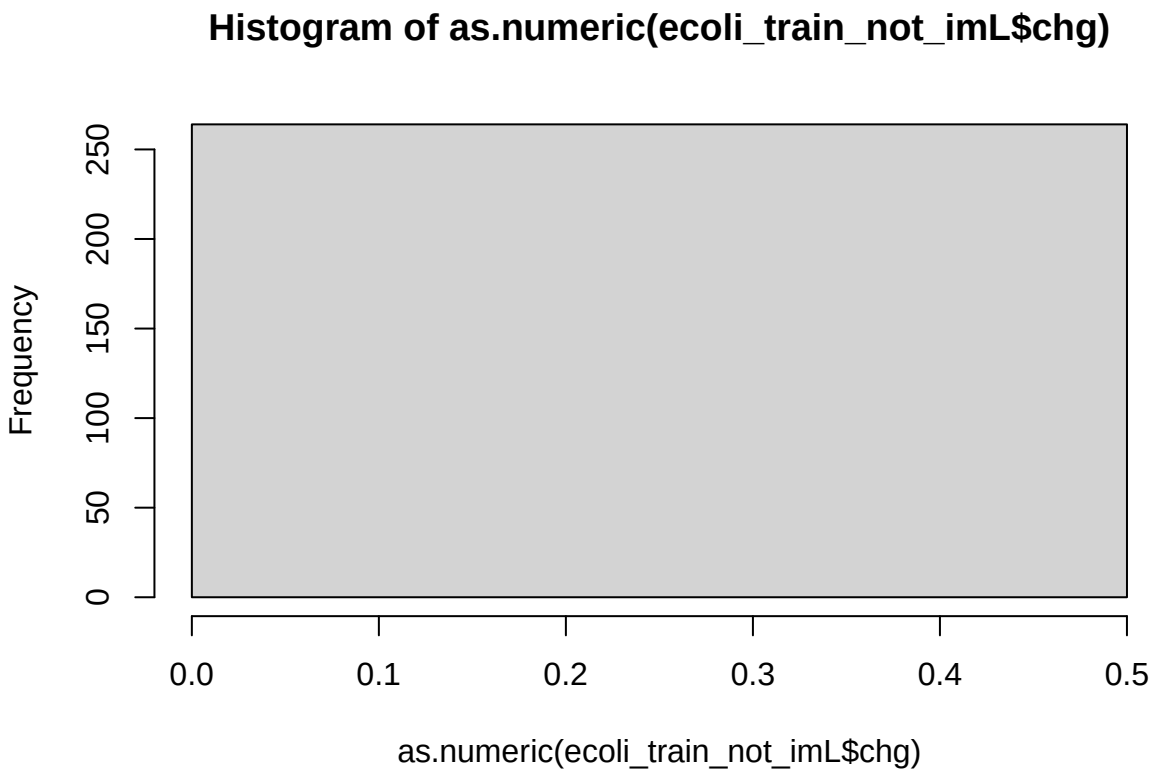
There is only one instance in the imL histogram. No specific chg score range can be distinguished from this visualisation.

The not-imL histogram shows that when the chg score is less than 0.5, the predicted locsite is generally not imL.

```
hist(as.numeric(ecoli_train_imL$chg))
```




```
hist(as.numeric(ecoli_train_not_imL$chg))
```

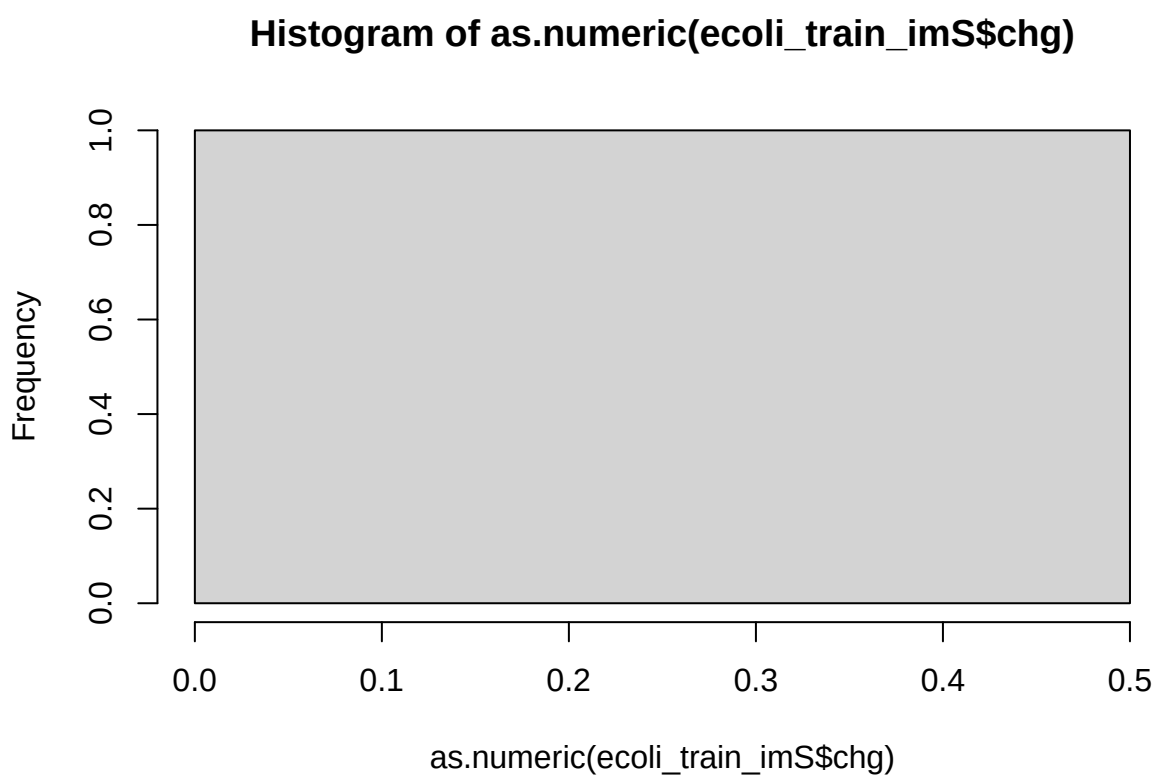


32. chg vs. imS and not imS

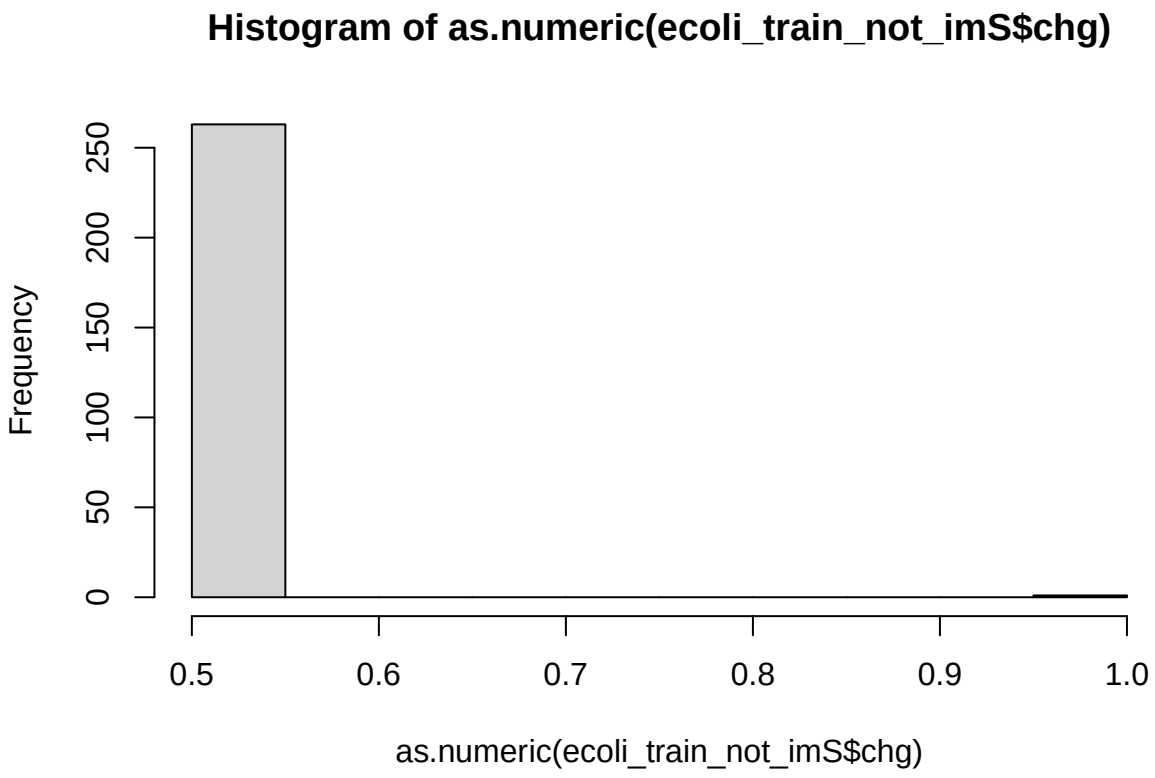
There is only one instance in the imS histogram. No specific chg score range can be distinguished from this visualisation.

The not-imS histogram shows that when the chg score is less than 0.55, the predicted locsite is generally not imS.

```
hist(as.numeric(ecoli_train_imS$chg))
```




```
hist(as.numeric(ecoli_train_not_imS$chg))
```

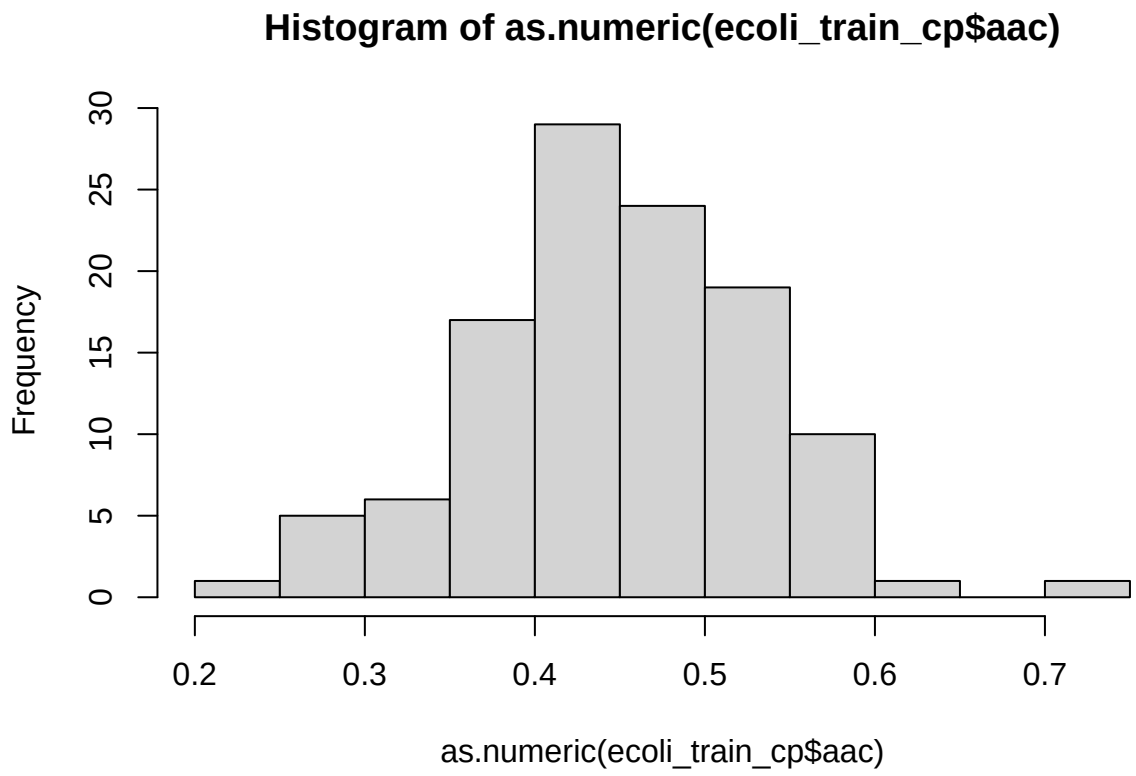


33. aac vs. cp and not cp

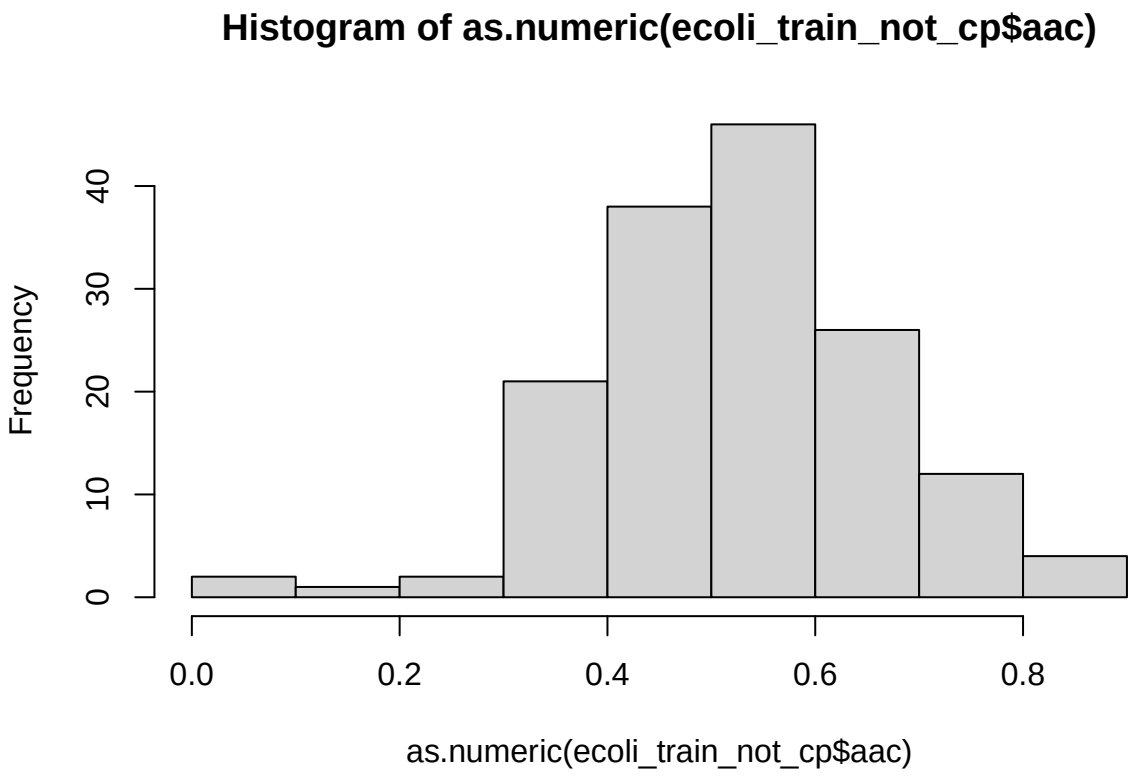
The cp histogram shows that cp is most frequently predicted when aac score ranges from 0.4 to 0.45.

The not-cp histogram shows that when the aac score is between 0.5 and 0.6, the predicted locsite is generally not cp.

```
hist(as.numeric(ecoli_train_cp$aac))
```




```
hist(as.numeric(ecoli_train_not_cp$aac))
```

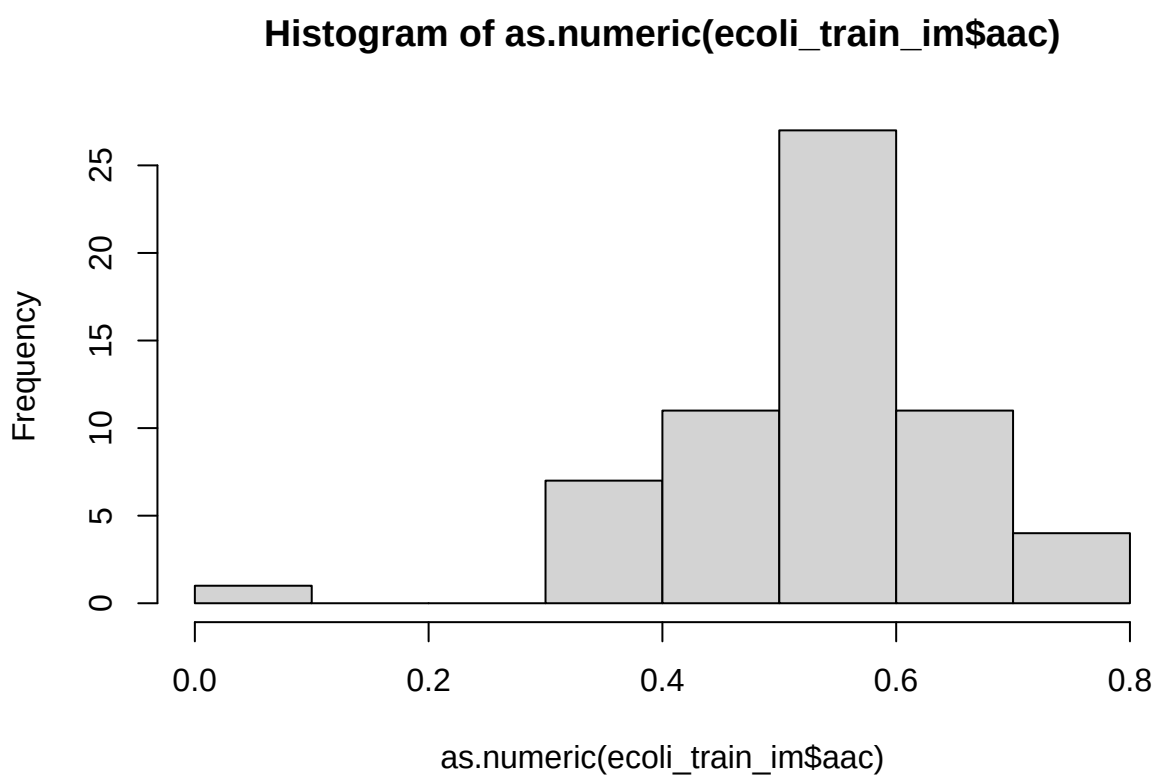


34. aac vs. im and not im

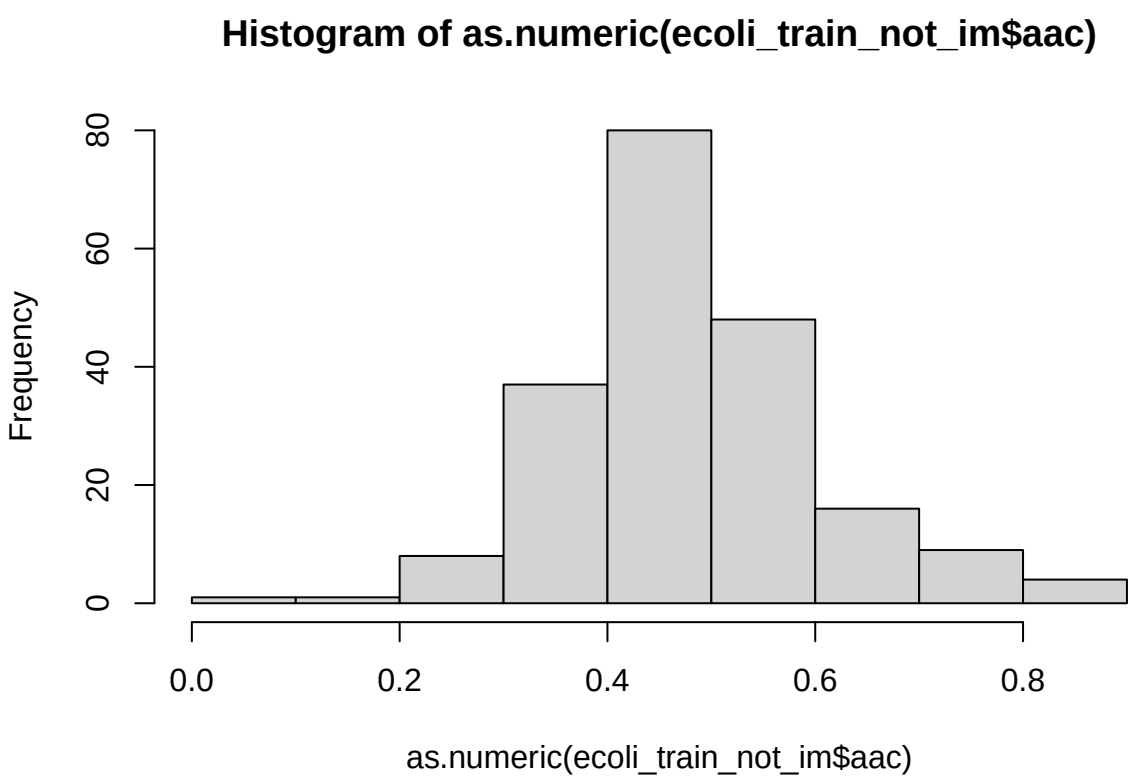
The im histogram shows that im is most frequently predicted when aac score ranges from 0.5 to 0.6.

The not-im histogram shows that when the aac score is between 0.4 and 0.5, the predicted locsite is generally not im.

```
hist(as.numeric(ecoli_train_im$aac))
```




```
hist(as.numeric(ecoli_train_not_im$aac))
```

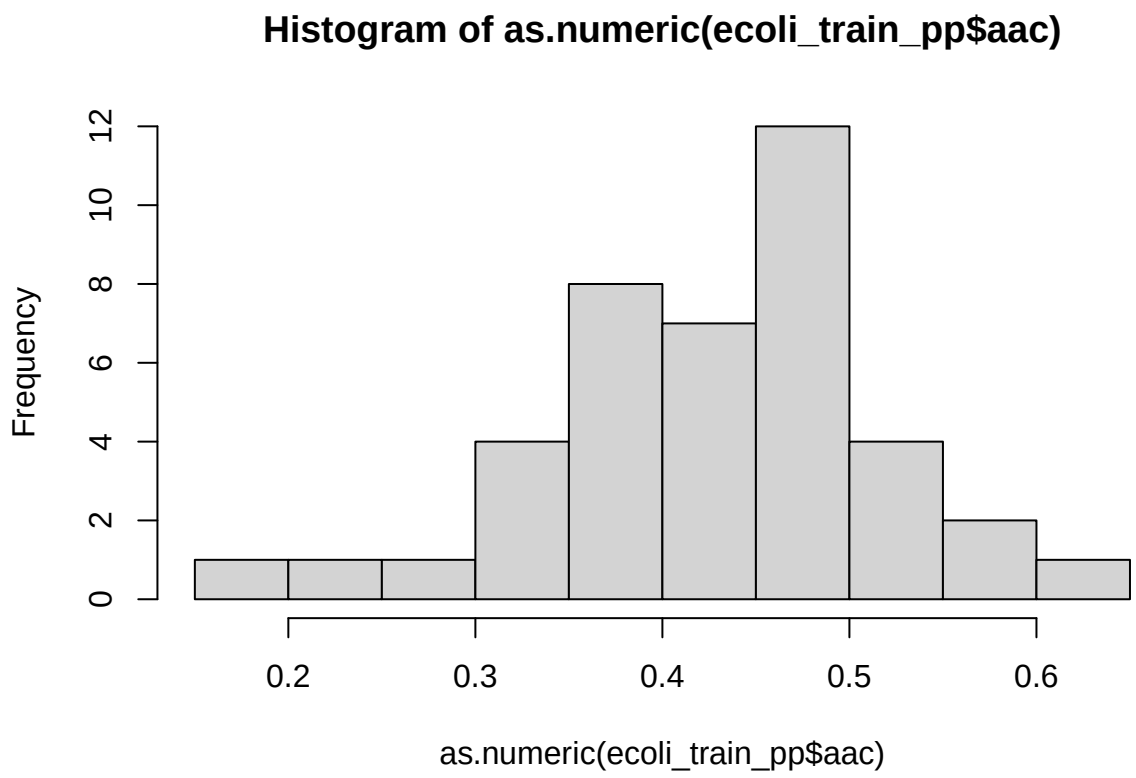


35. aac vs. pp and not pp

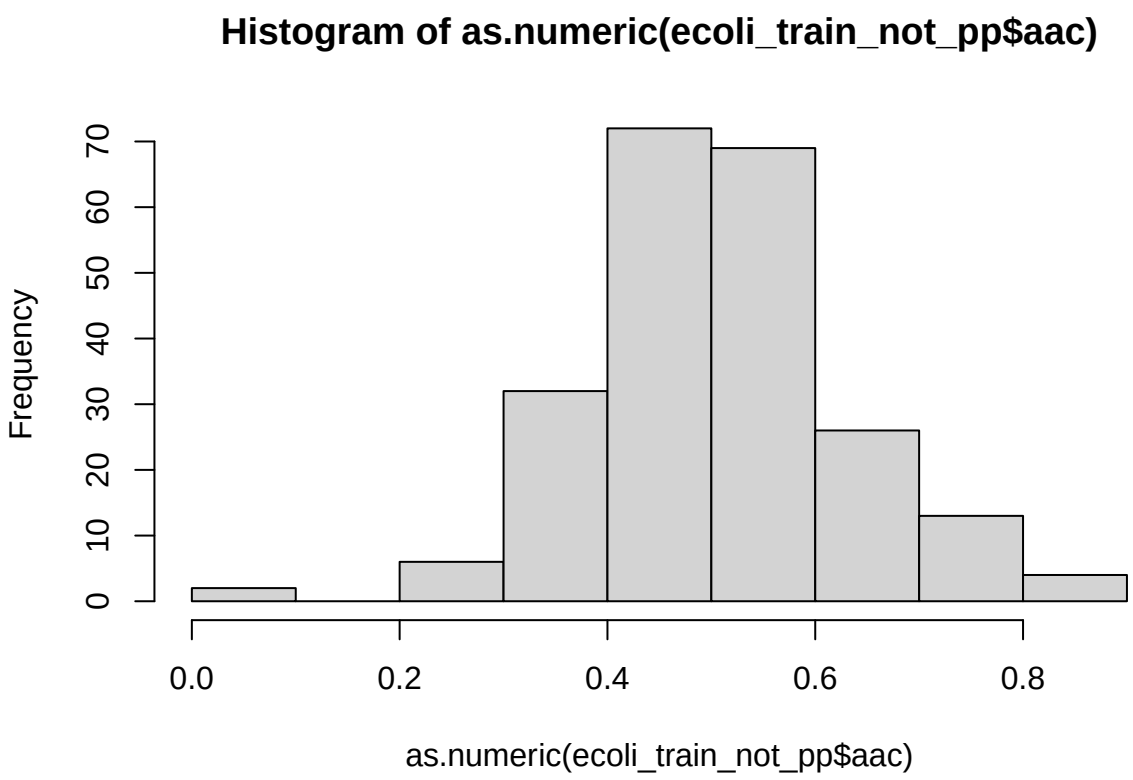
The pp histogram shows that pp is most frequently predicted when aac score ranges from 0.45 to 0.5.

The not-pp histogram shows that when the aac score is between 0.4 and 0.5, the predicted locsite is generally not pp.

```
hist(as.numeric(ecoli_train_pp$aac))
```




```
hist(as.numeric(ecoli_train_not_pp$aac))
```



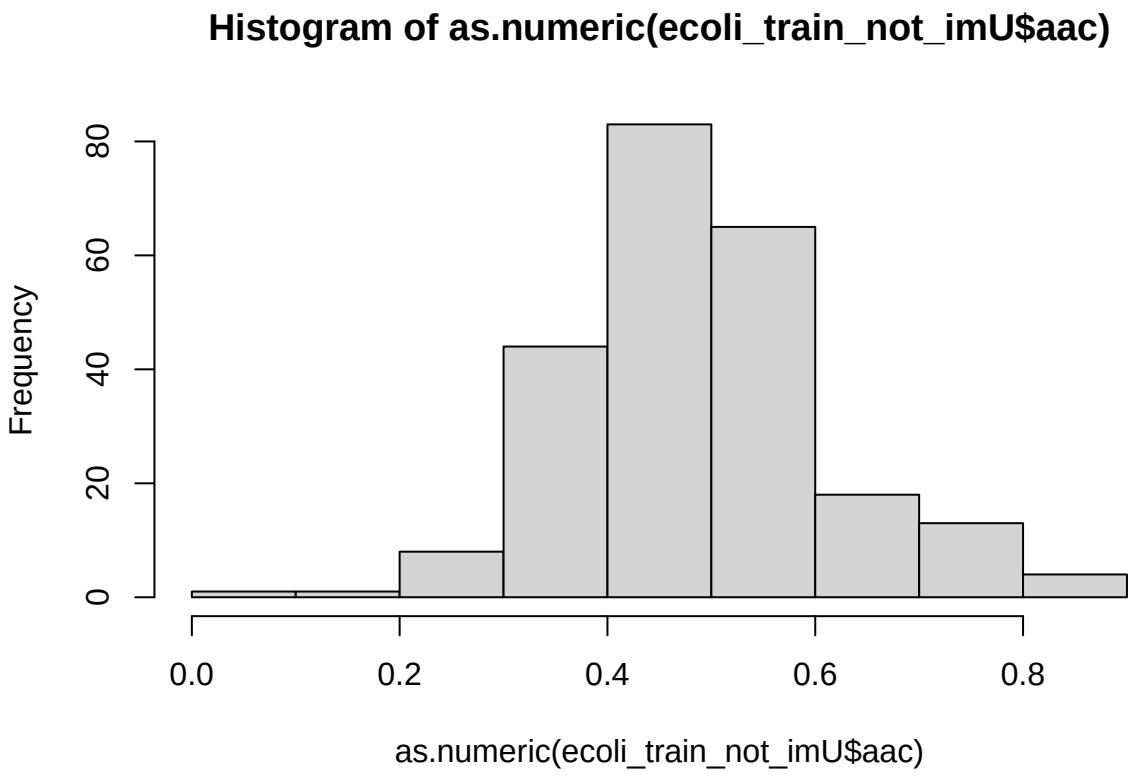
36. aac vs. imU and not imU

The imU histogram shows that imU is most frequently predicted when aac score ranges from 0.5 to 0.6.

The not-imU histogram shows that when the aac score is between 0.4 and 0.5, the predicted locsite is generally not imU.

```
hist(as.numeric(ecoli_train_imU$aac))
```


```
hist(as.numeric(ecoli_train_not_imU$aac))
```

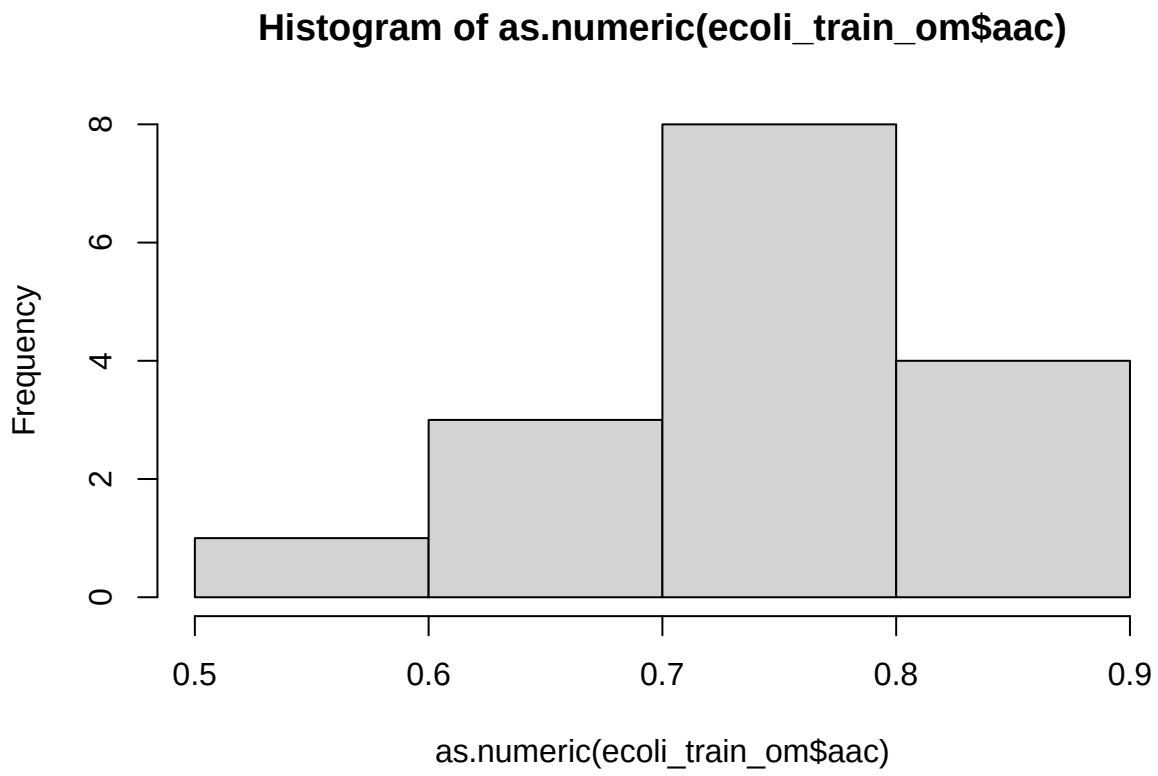


37. aac vs. om and not om

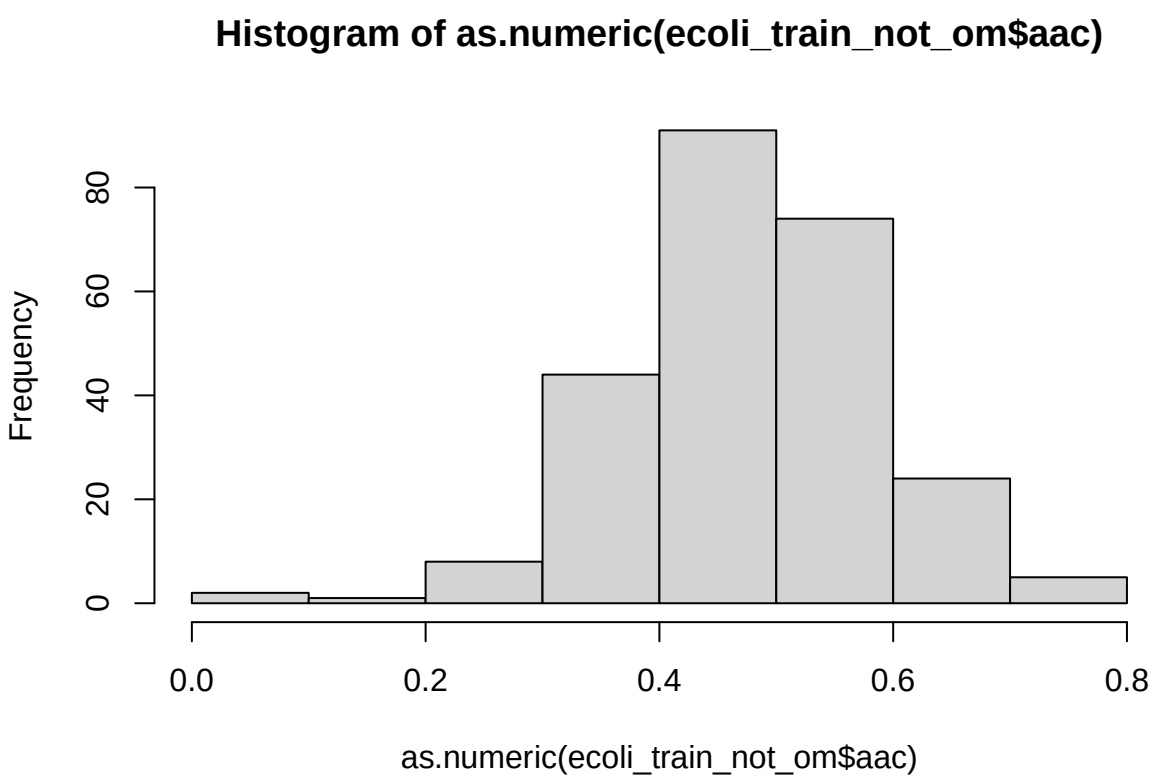
The om histogram shows that om is most frequently predicted when aac score ranges from 0.7 to 0.8.

The not-om histogram shows that when the aac score is between 0.4 and 0.5, the predicted locsite is generally not om.

```
hist(as.numeric(ecoli_train_om$aac))
```




```
hist(as.numeric(ecoli_train_not_om$aac))
```

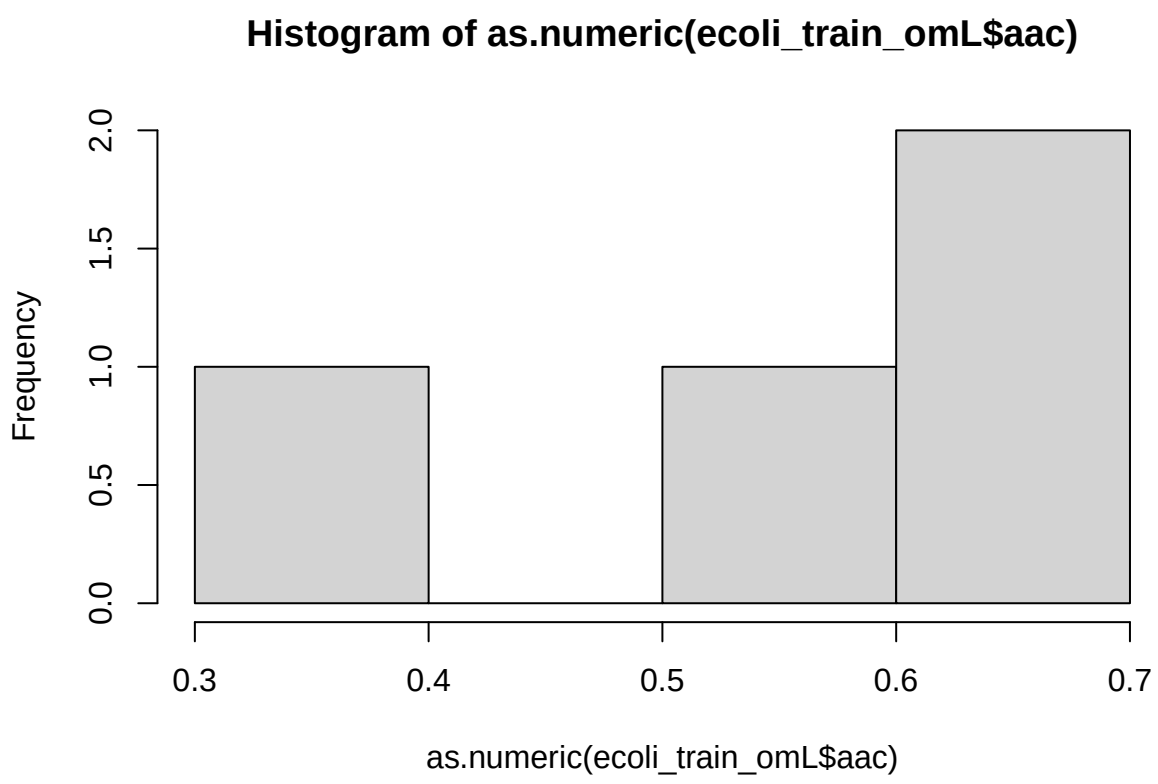


38. aac vs. omL and not omL

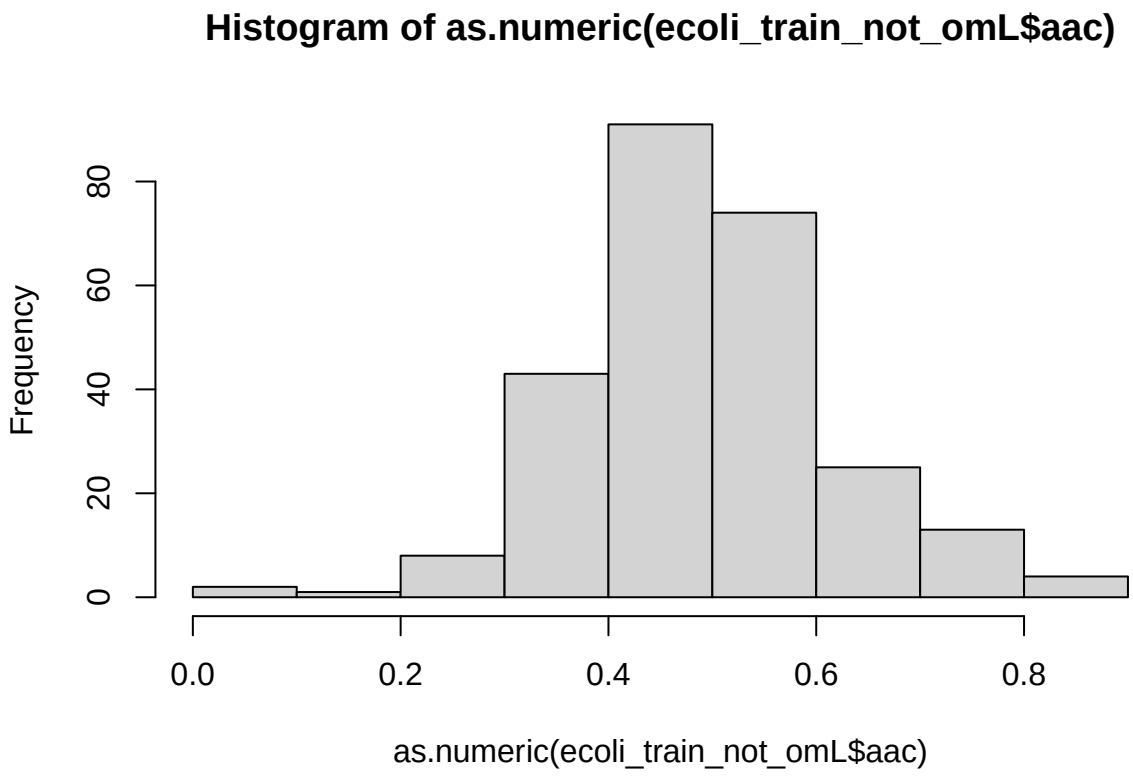
The omL histogram shows that omL is most frequently predicted when aac score ranges from 0.6 to 0.7.

The not-omL histogram shows that when the aac score is between 0.4 and 0.5, the predicted locsite is generally not omL.

```
hist(as.numeric(ecoli_train_omL$aac))
```




```
hist(as.numeric(ecoli_train_not_omL$aac))
```

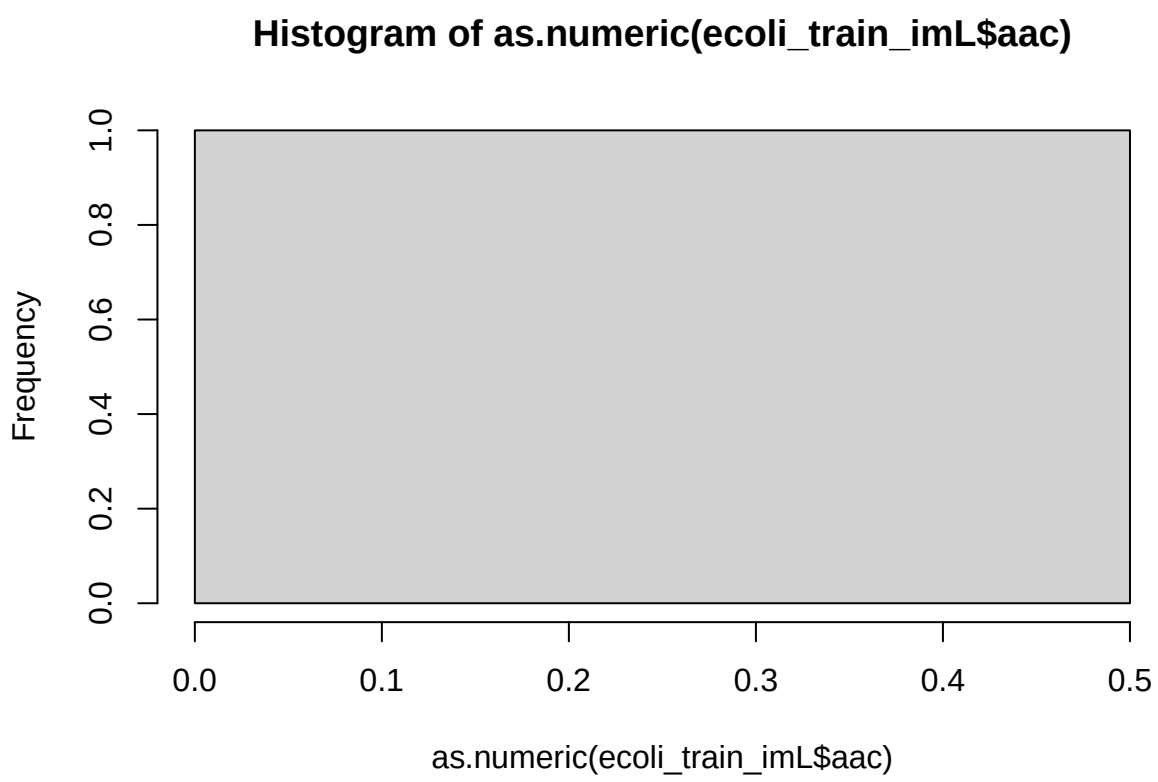


39. aac vs. imL and not imL

There is only one instance in the imL histogram. No specific aac score range can be distinguished from this visualisation.

The not-imL histogram shows that when the aac score is between 0.4 and 0.5, the predicted locsite is generally not imL.

```
hist(as.numeric(ecoli_train_imL$aac))
```



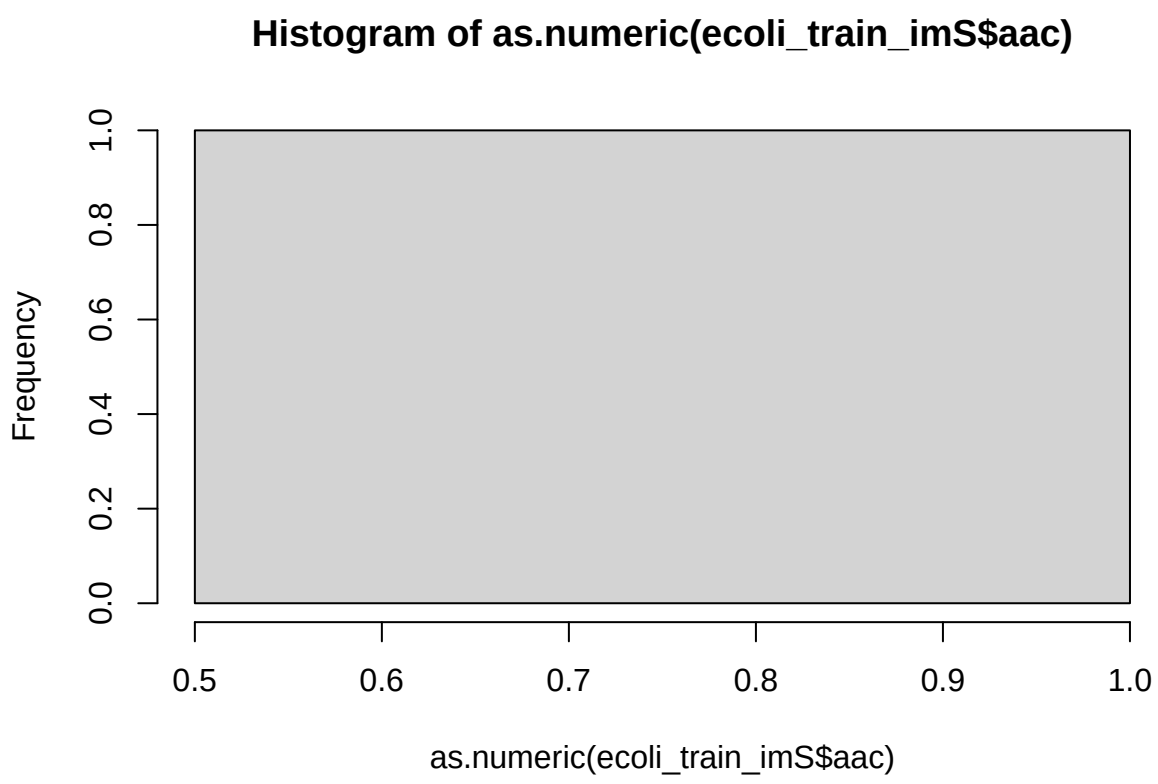

```
hist(as.numeric(ecoli_train_not_imL$aac))
```

40. aac vs. imS and not imS

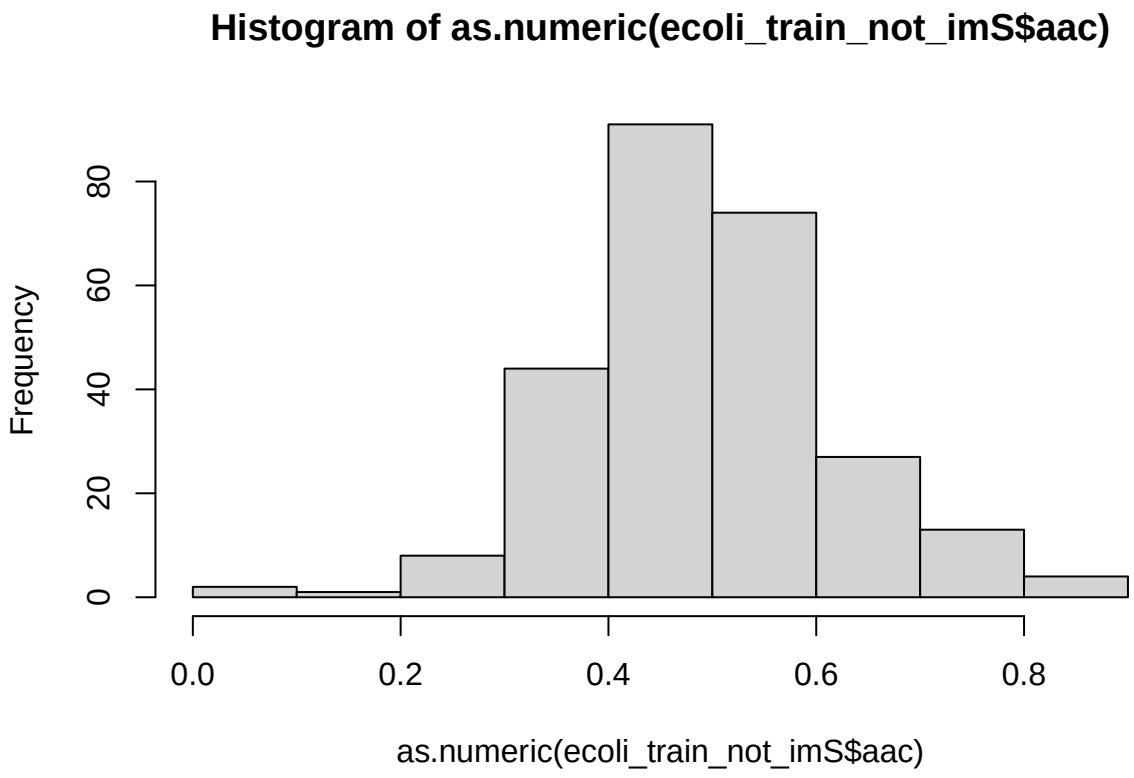
There is only one instance in the imS histogram. No specific aac score range can be distinguished from this visualisation.

The not-imS histogram shows that when the aac score is between 0.4 and 0.5, the predicted locsite is generally not imS.

```
hist(as.numeric(ecoli_train_imS$aac))
```




```
hist(as.numeric(ecoli_train_not_imS$aac))
```

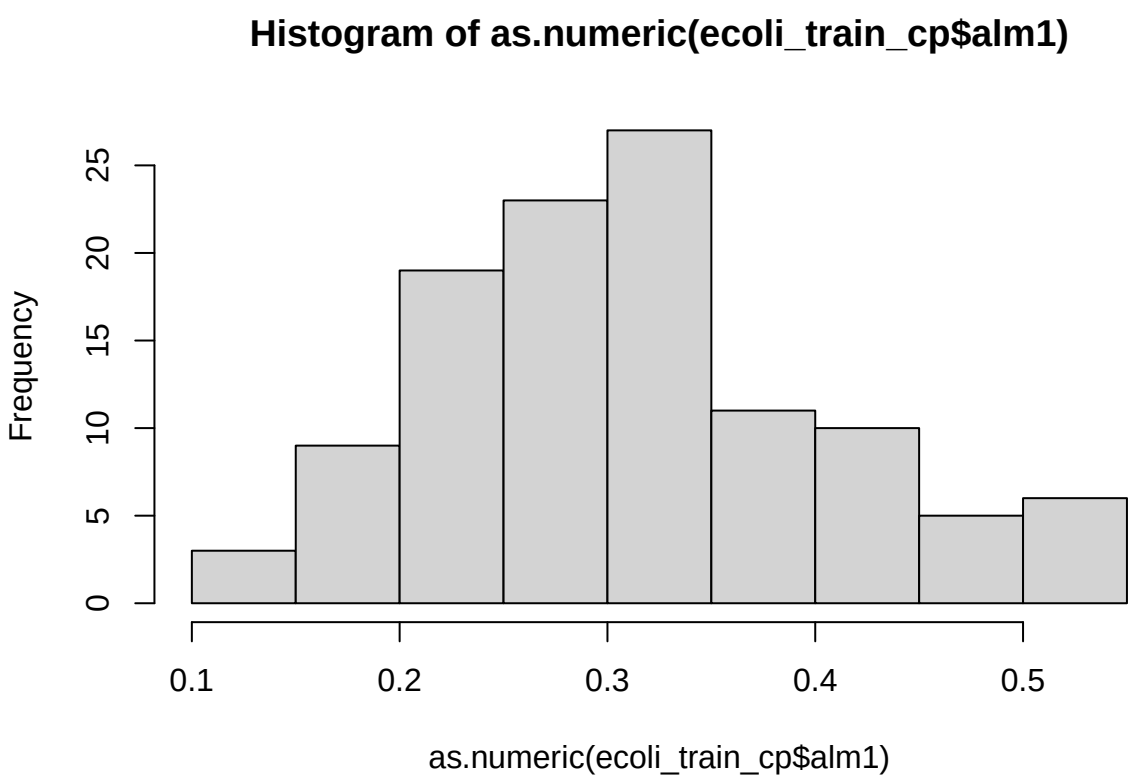


41. alm1 vs. cp and not cp

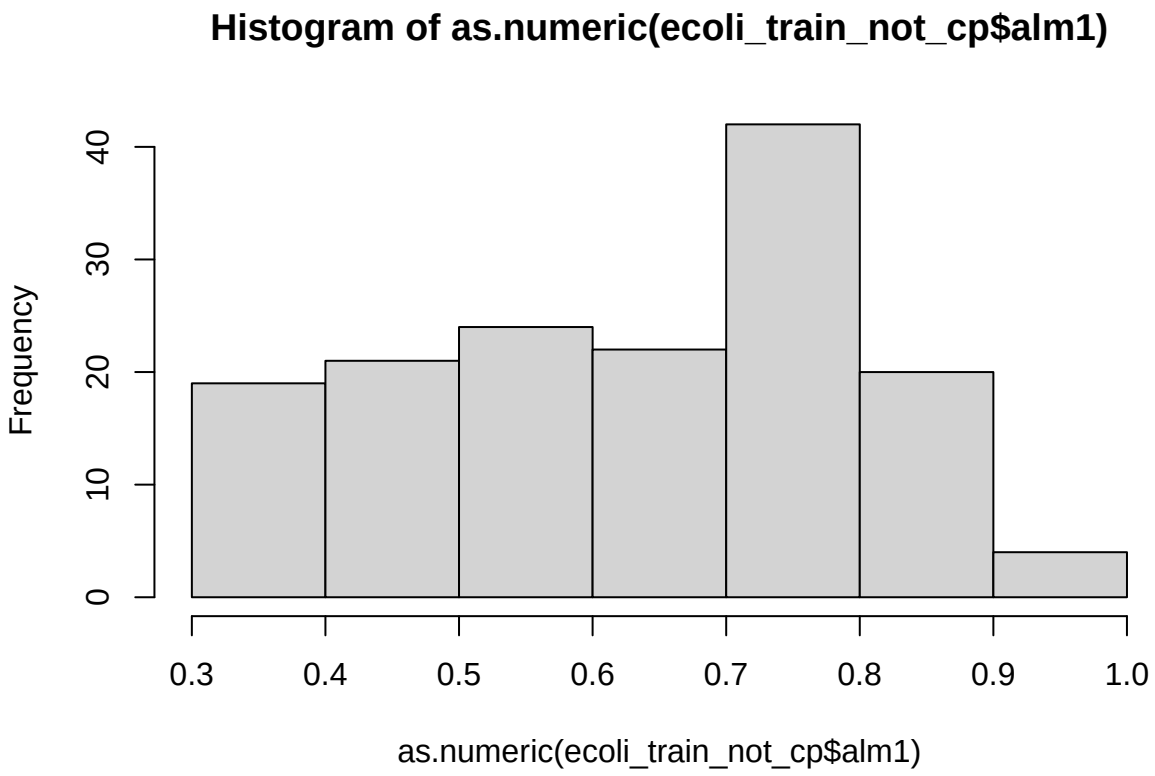
The cp histogram shows that cp is most frequently predicted when alm1 score ranges from 0.3 to 0.35.

The not-cp histogram shows that when the alm1 score is between 0.7 and 0.8, the predicted locsite is generally not cp.

```
hist(as.numeric(ecoli_train_cp$alm1))
```




```
hist(as.numeric(ecoli_train_not_cp$alm1))
```

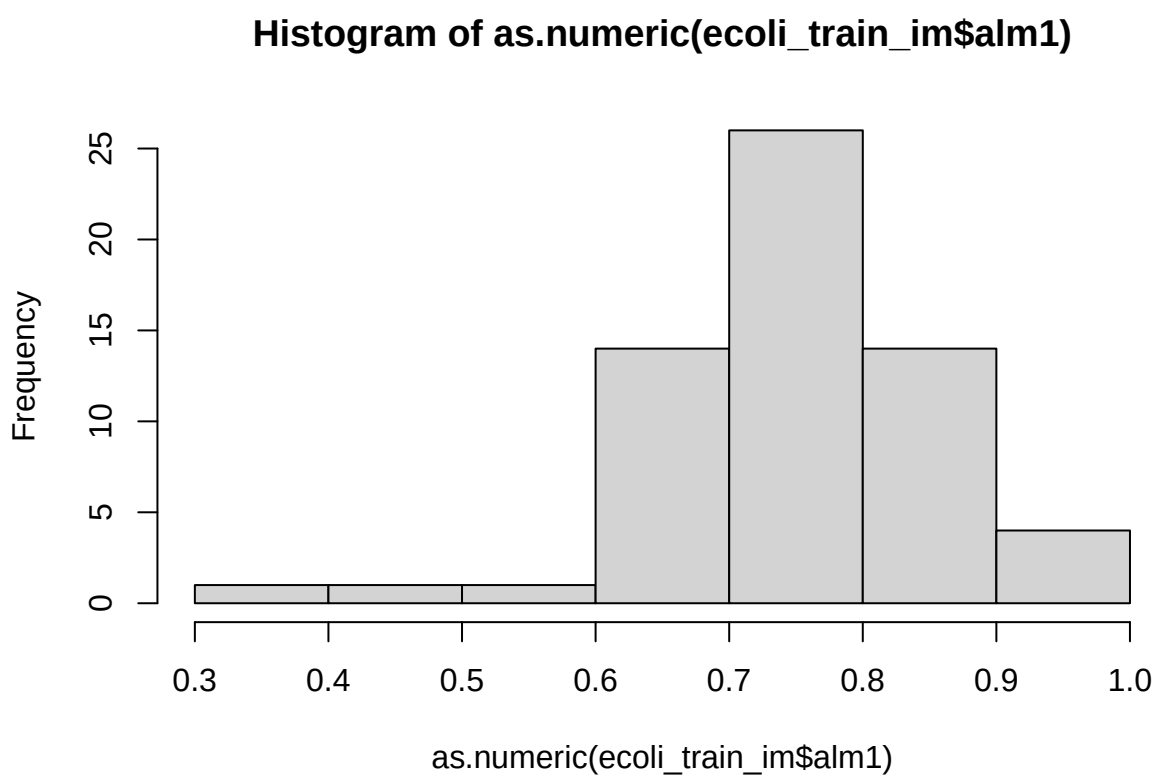


42. alm1 vs. im and not im

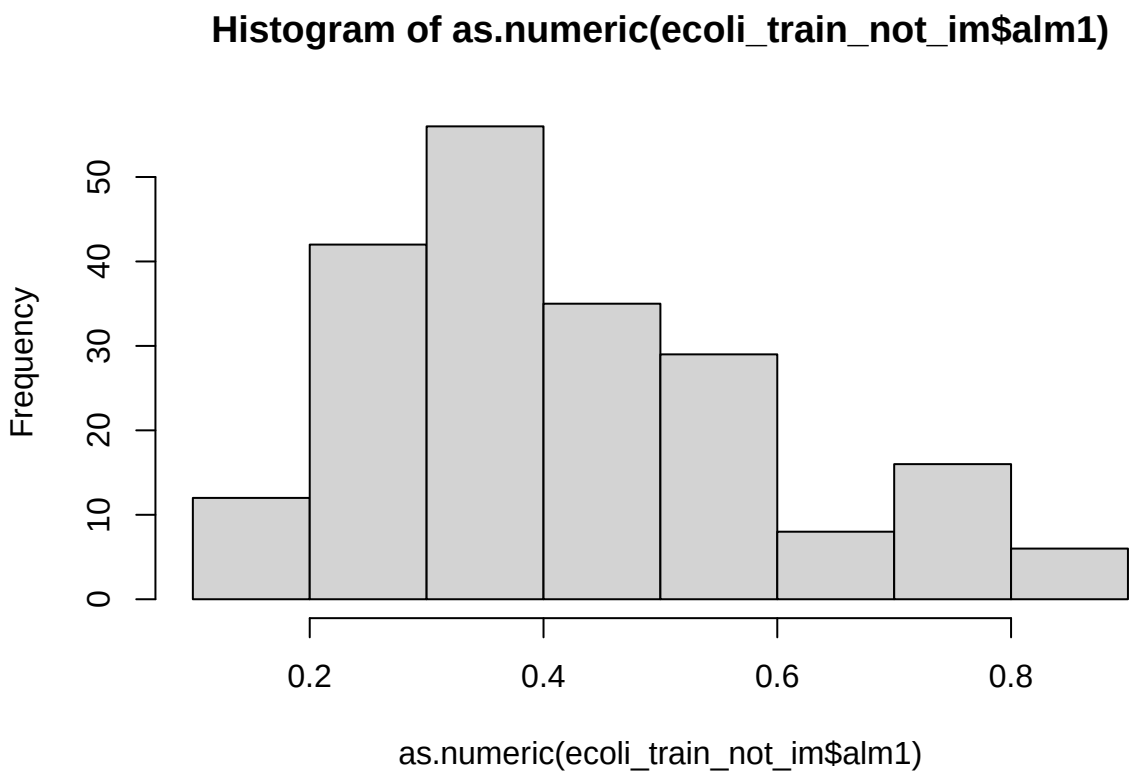
The im histogram shows that im is most frequently predicted when alm1 score ranges from 0.7 to 0.8.

The not-cp histogram shows that when the alm1 score is between 0.3 and 0.4, the predicted locsite is generally not im.

```
hist(as.numeric(ecoli_train_im$alm1))
```




```
hist(as.numeric(ecoli_train_not_im$alm1))
```

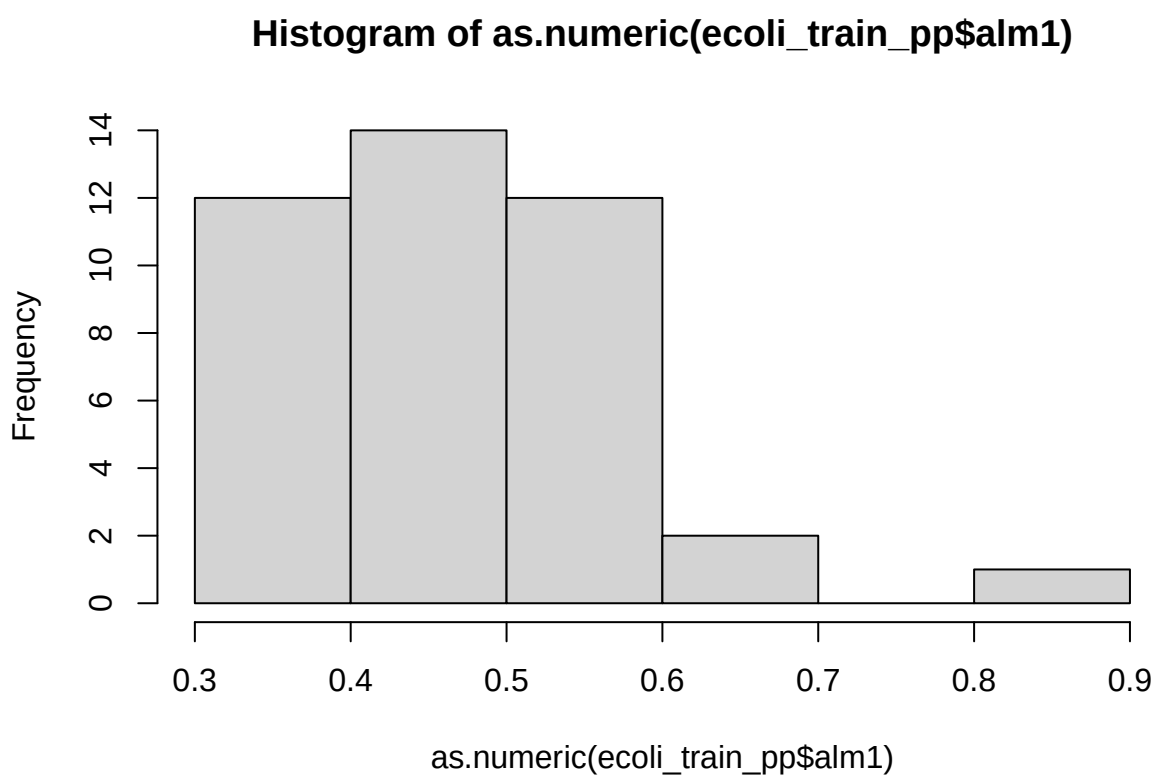


43. alm1 vs. pp and not pp

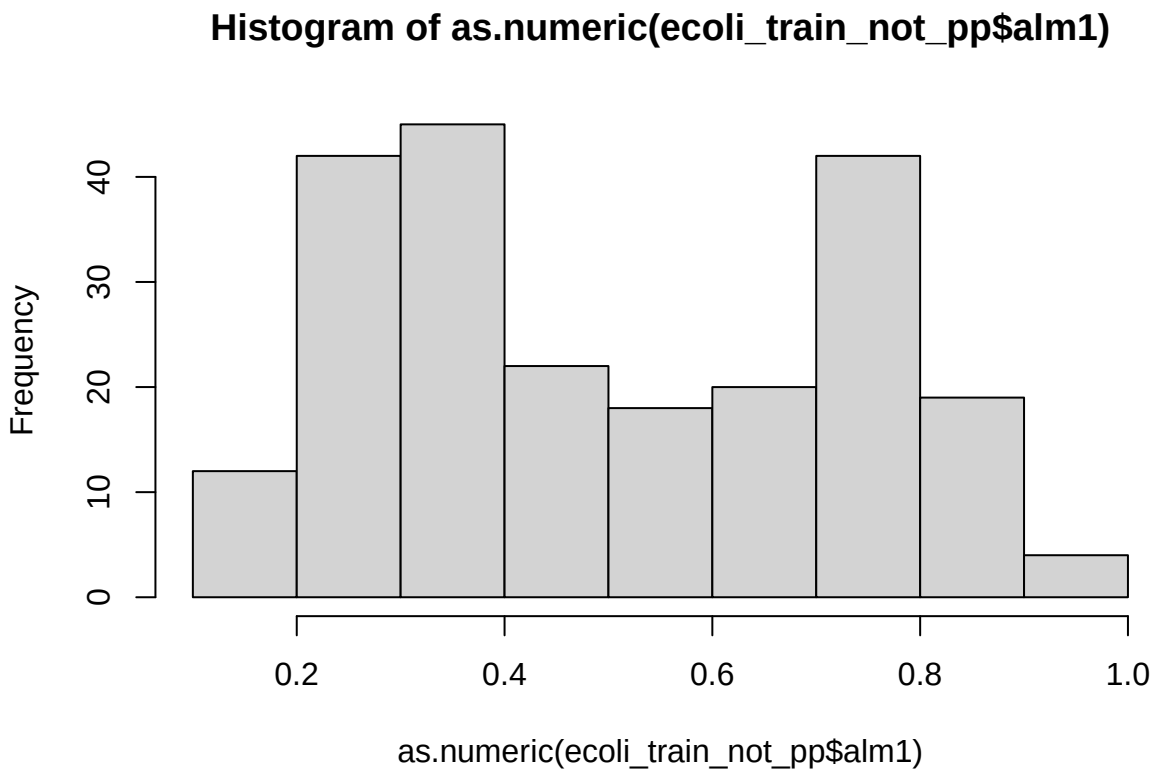
The pp histogram shows that pp is most frequently predicted when alm1 score ranges from 0.4 to 0.5.

The not-pp histogram shows that when the alm1 score is between 0.3 and 0.4, the predicted locsite is generally not pp.

```
hist(as.numeric(ecoli_train_pp$alm1))
```




```
hist(as.numeric(ecoli_train_not_pp$alm1))
```

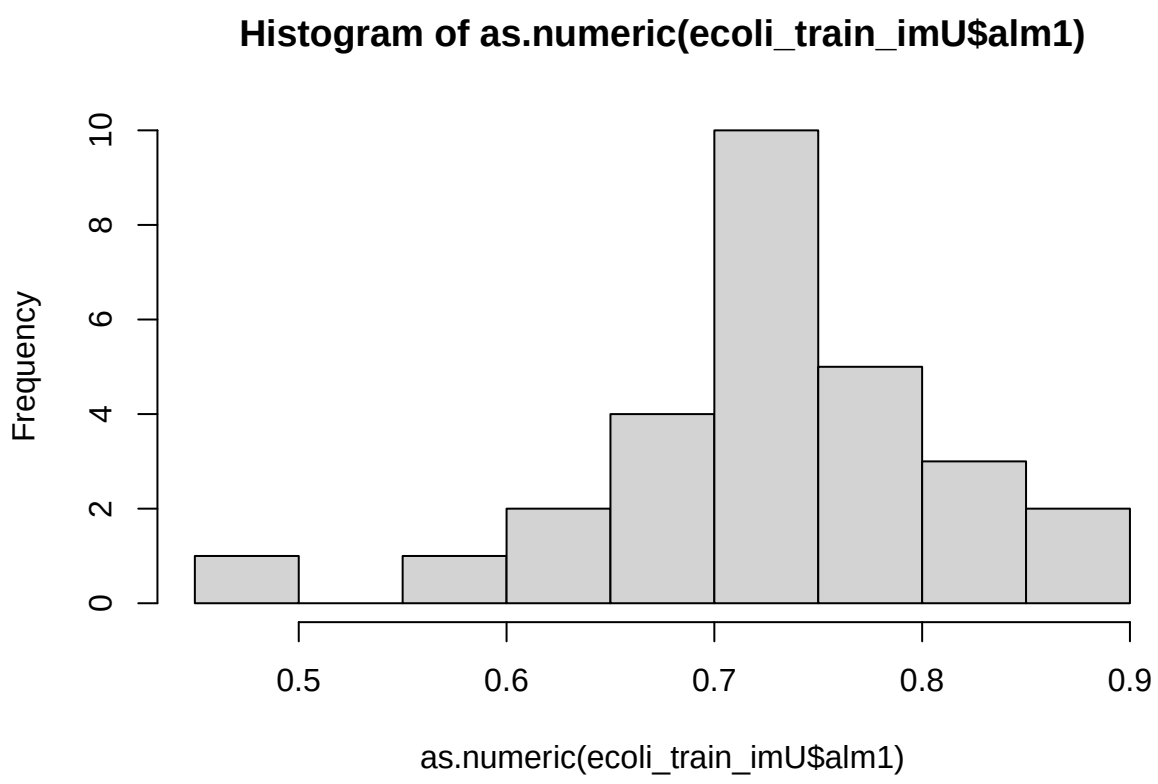


44. alm1 vs. imU and not imU

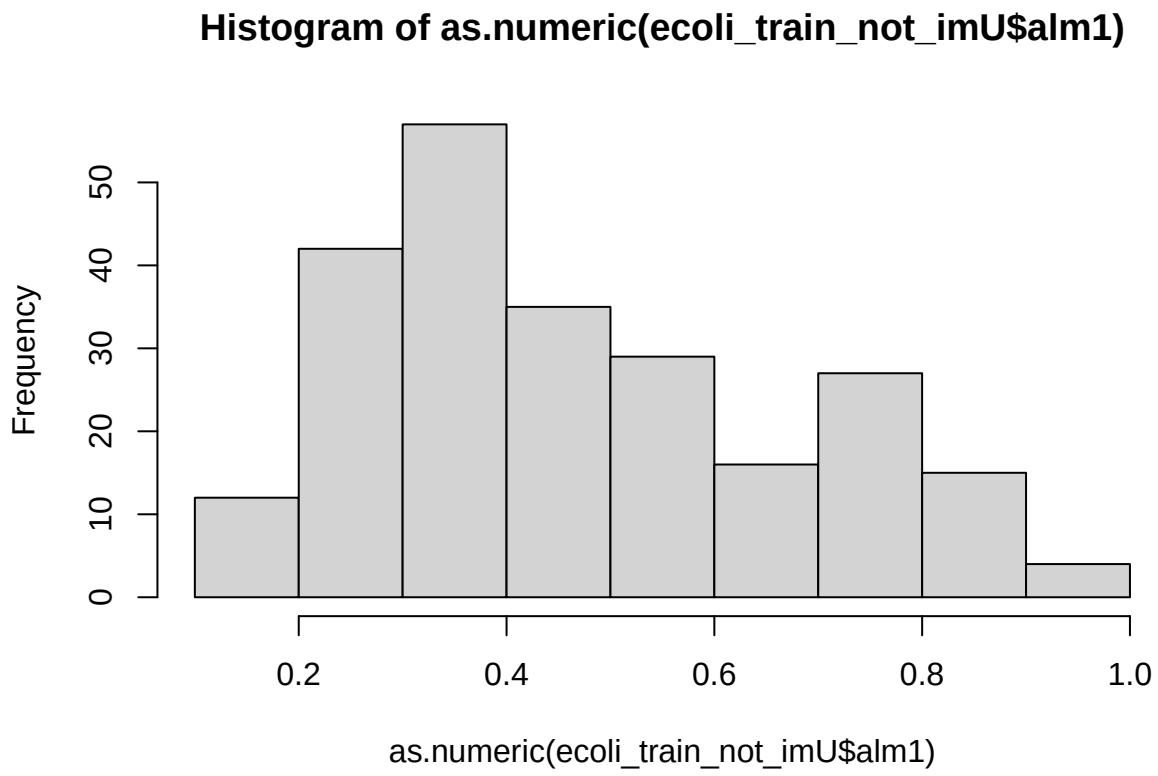
The imU histogram shows that imU is most frequently predicted when alm1 score ranges from 0.7 to 0.75.

The not-imU histogram shows that when the alm1 score is between 0.3 and 0.4, the predicted locsite is generally not imU.

```
hist(as.numeric(ecoli_train_imU$alm1))
```




```
hist(as.numeric(ecoli_train_not_imU$alm1))
```

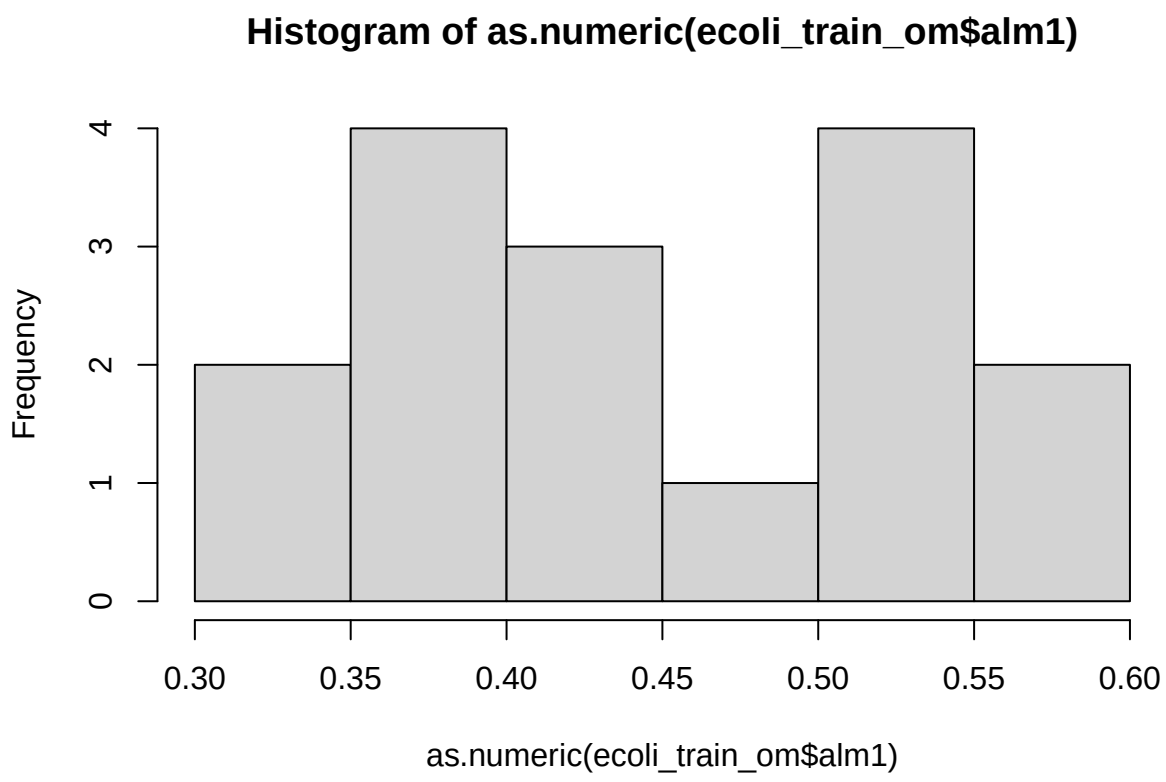


45. alm1 vs. om and not om

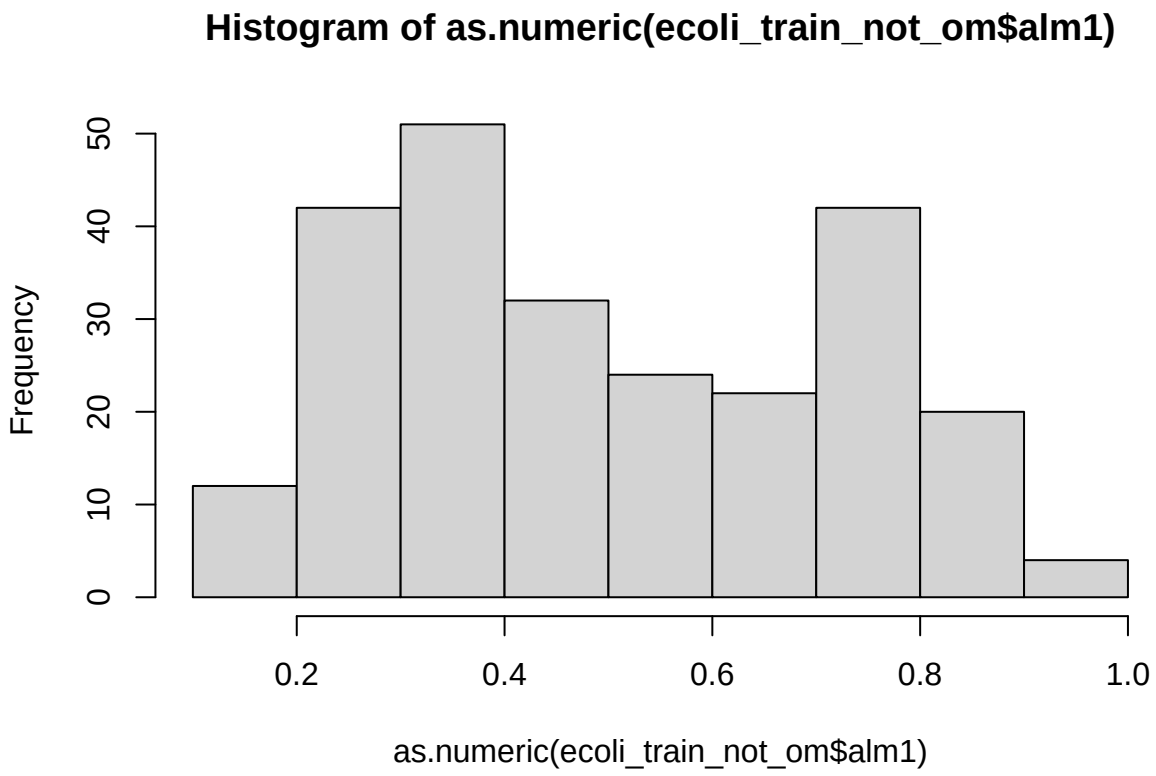
The om histogram shows that om is most frequently predicted when alm1 score ranges from 0.35 to 0.4 and from 0.5 to 0.55.

The not-om histogram shows that when the alm1 score is between 0.3 and 0.4, the predicted locsite is generally not om.

```
hist(as.numeric(ecoli_train_om$alm1))
```




```
hist(as.numeric(ecoli_train_not_om$alm1))
```

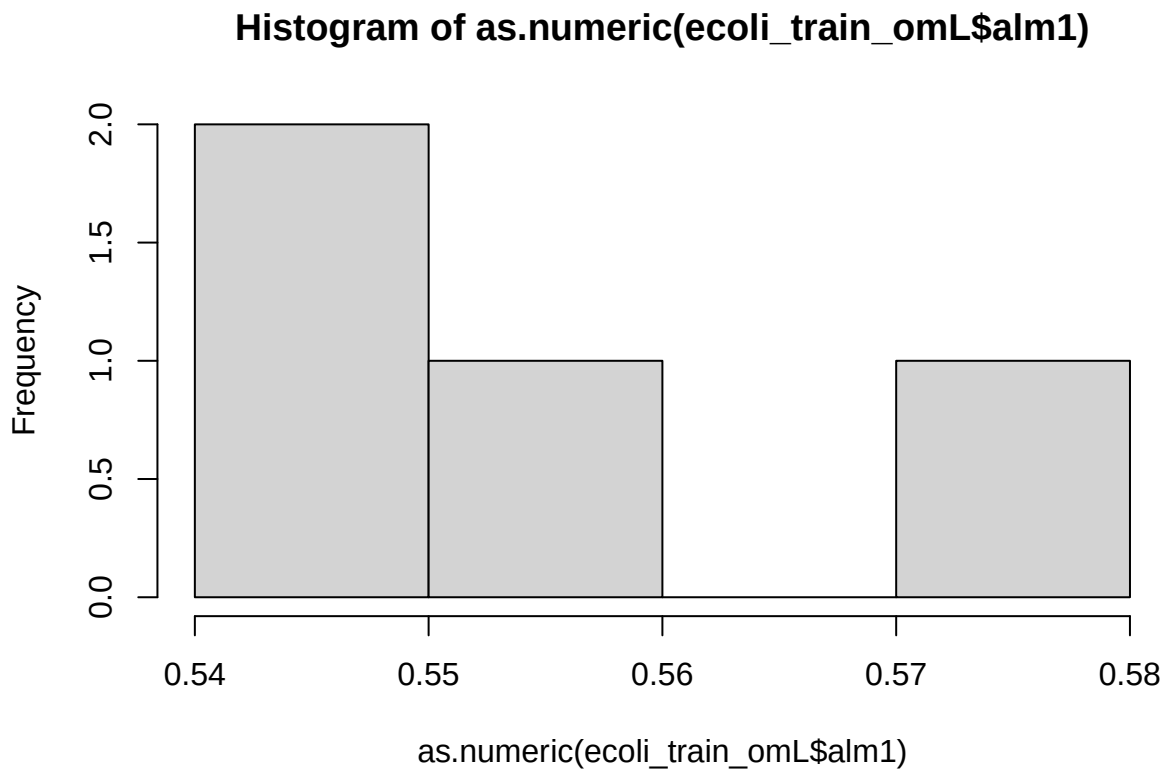


46. alm1 vs. omL and not omL

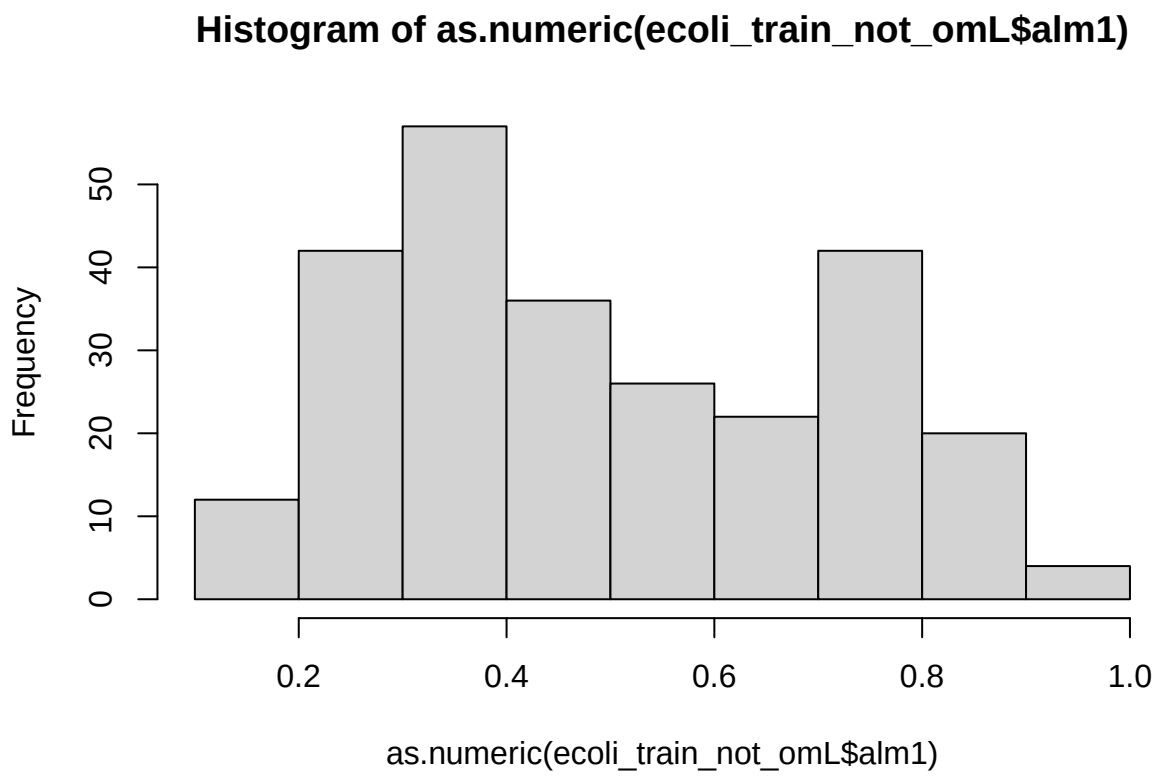
The omL histogram shows that omL is most frequently predicted when alm1 score ranges from 0.54 to 0.55.

The not-omL histogram shows that when the alm1 score is between 0.3 and 0.4, the predicted locsite is generally not omL.

```
hist(as.numeric(ecoli_train_omL$alm1))
```




```
hist(as.numeric(ecoli_train_not_omL$alm1))
```

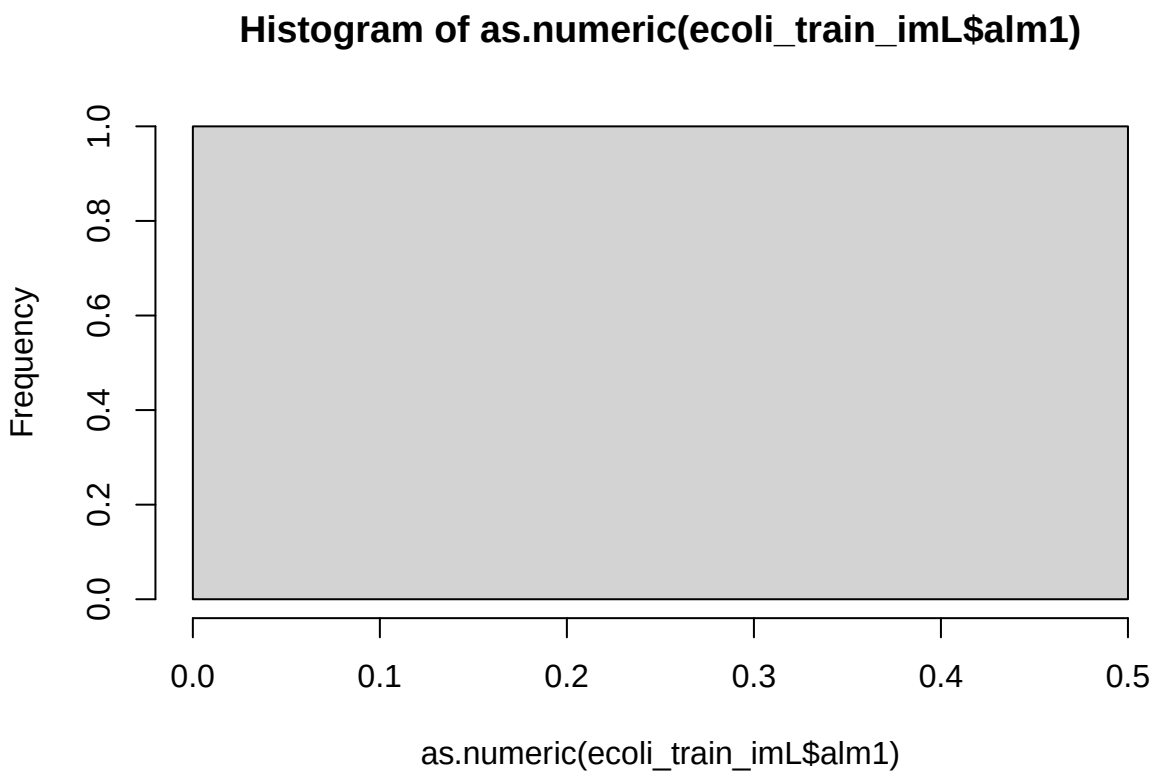


47. alm1 vs. imL and not imL

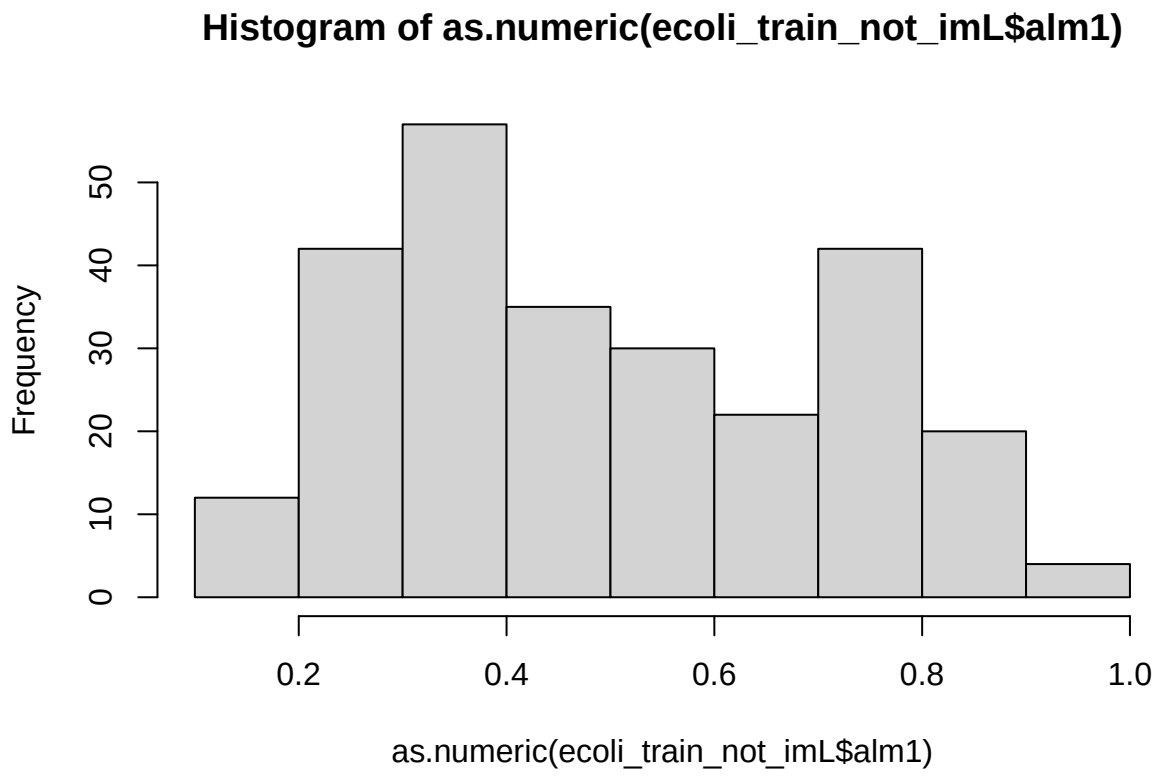
There is only one instance in the imL histogram. No specific alm1 score range can be distinguished from this visualisation.

The not-imL histogram shows that when the alm1 score is between 0.3 and 0.4, the predicted locsite is generally not imL.

```
hist(as.numeric(ecoli_train_imL$alm1))
```




```
hist(as.numeric(ecoli_train_not_imL$alm1))
```

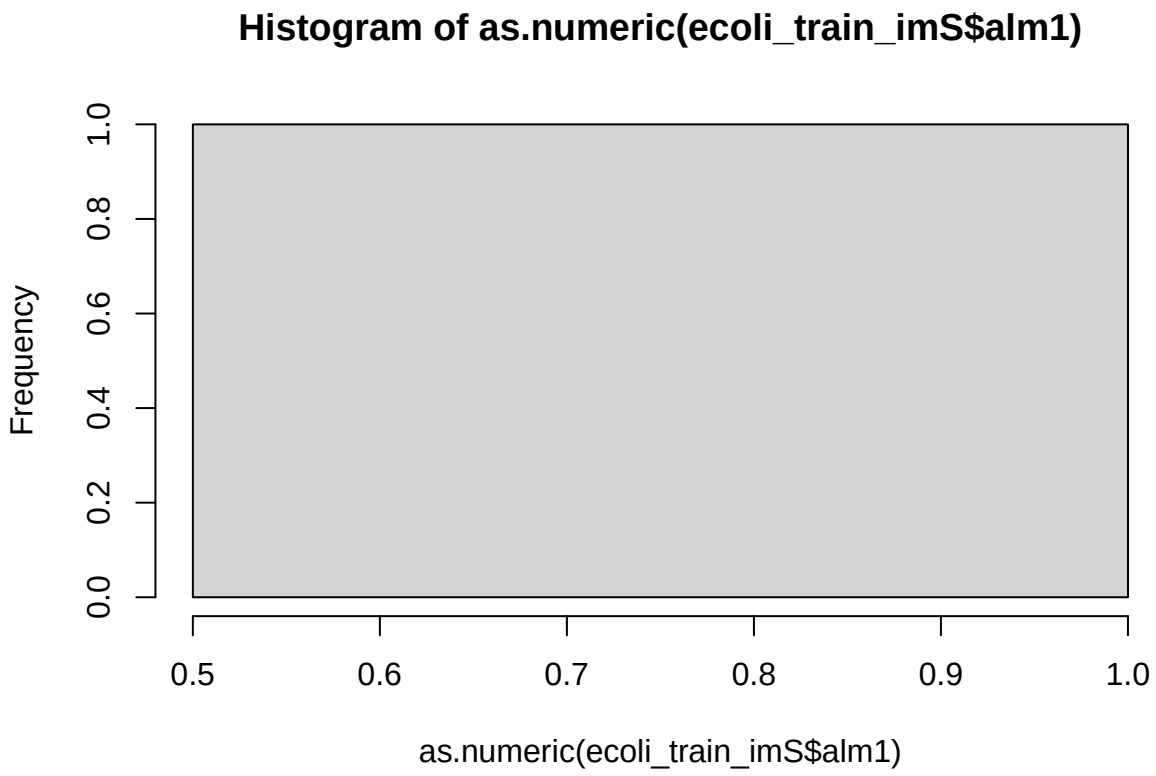


48. alm1 vs. imS and not imS

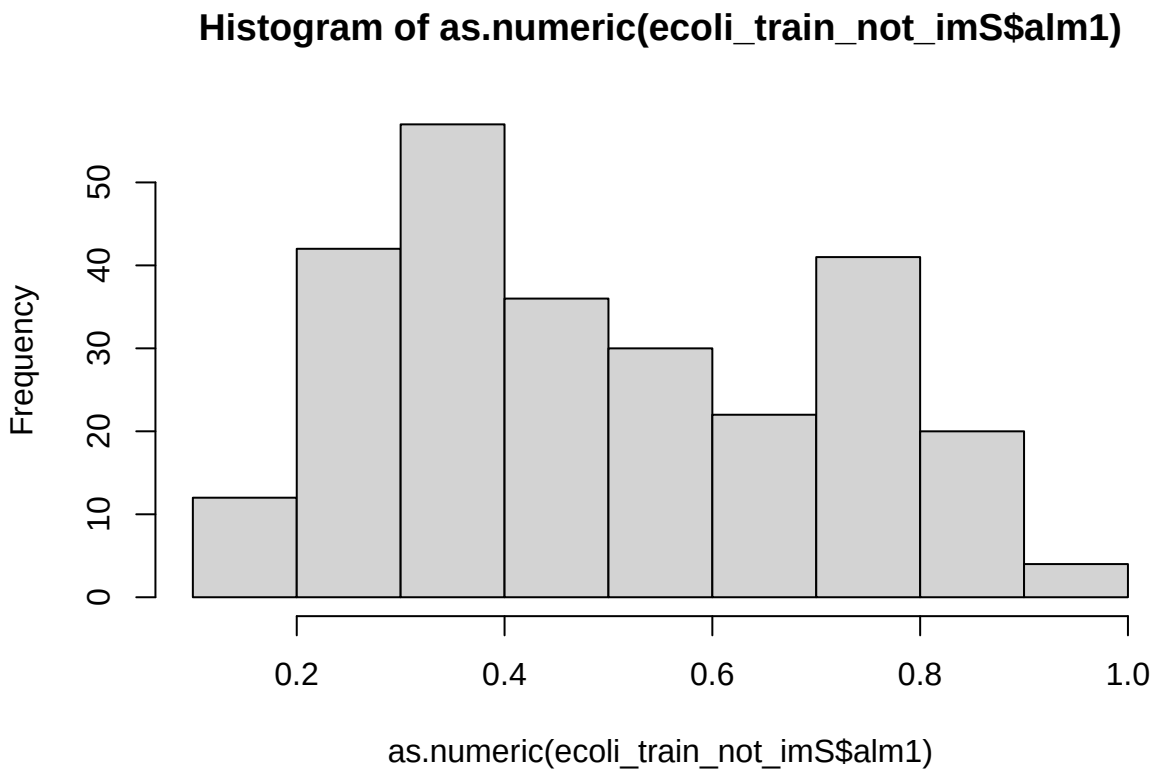
There is only one instance in the imS histogram. No specific alm1 score range can be distinguished from this visualisation.

The not-imL histogram shows that when the alm1 score is between 0.3 and 0.4, the predicted locsite is generally not imS.

```
hist(as.numeric(ecoli_train_imS$alm1))
```




```
hist(as.numeric(ecoli_train_not_imS$alm1))
```

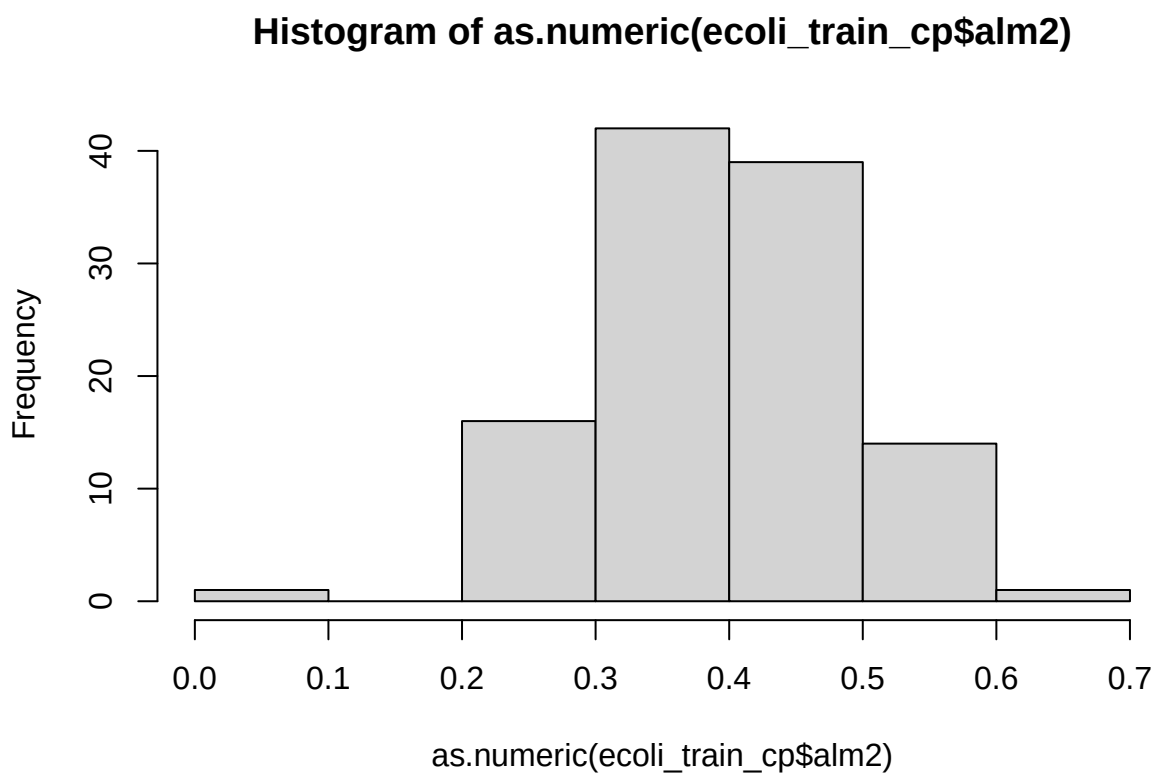


49. alm2 vs. cp and not cp

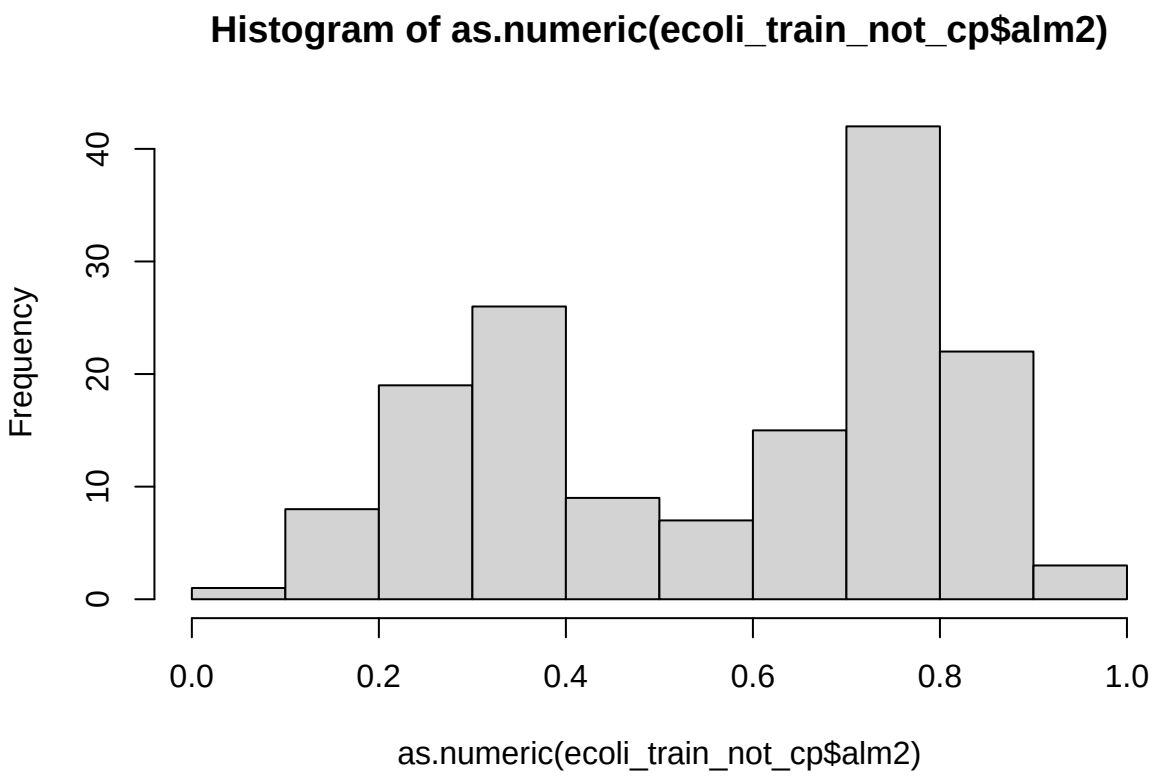
The cp histogram shows that cp is most frequently predicted when alm2 score ranges from 0.3 to 0.4.

The not-cp histogram shows that when the alm2 score is between 0.7 and 0.8, the predicted locsite is generally not cp.

```
hist(as.numeric(ecoli_train_cp$alm2))
```




```
hist(as.numeric(ecoli_train_not_cp$alm2))
```

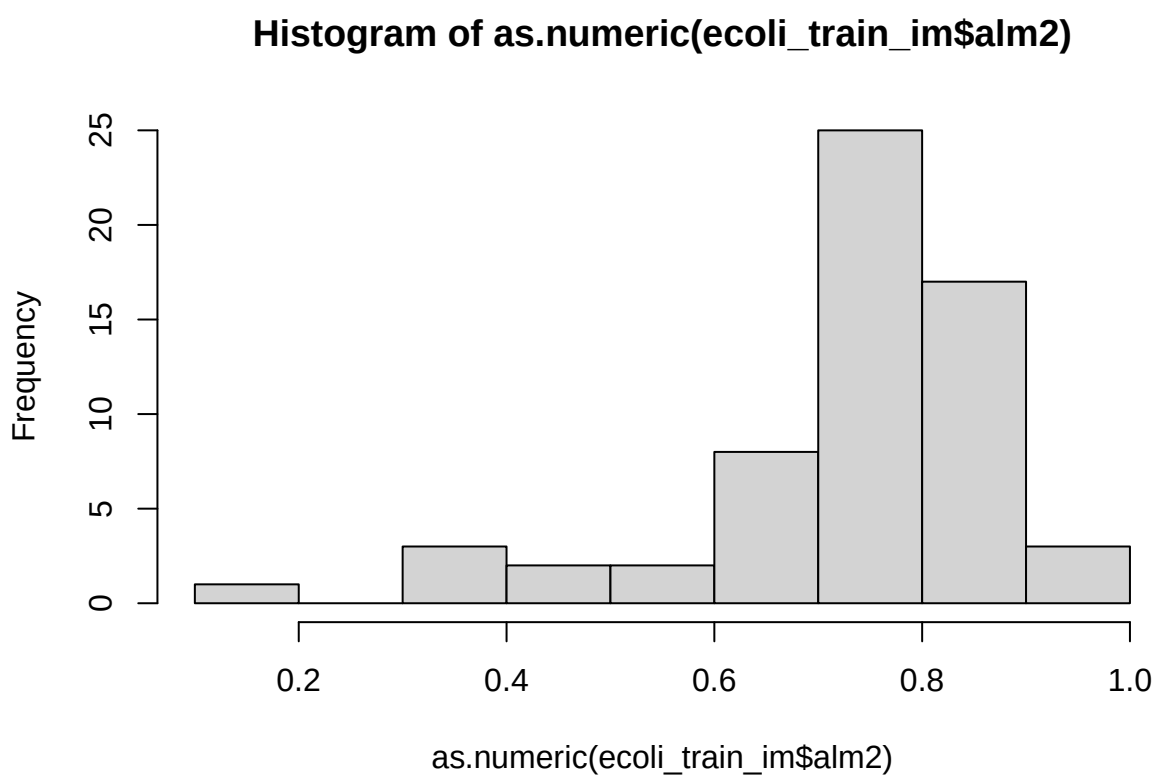


50. alm2 vs. im and not im

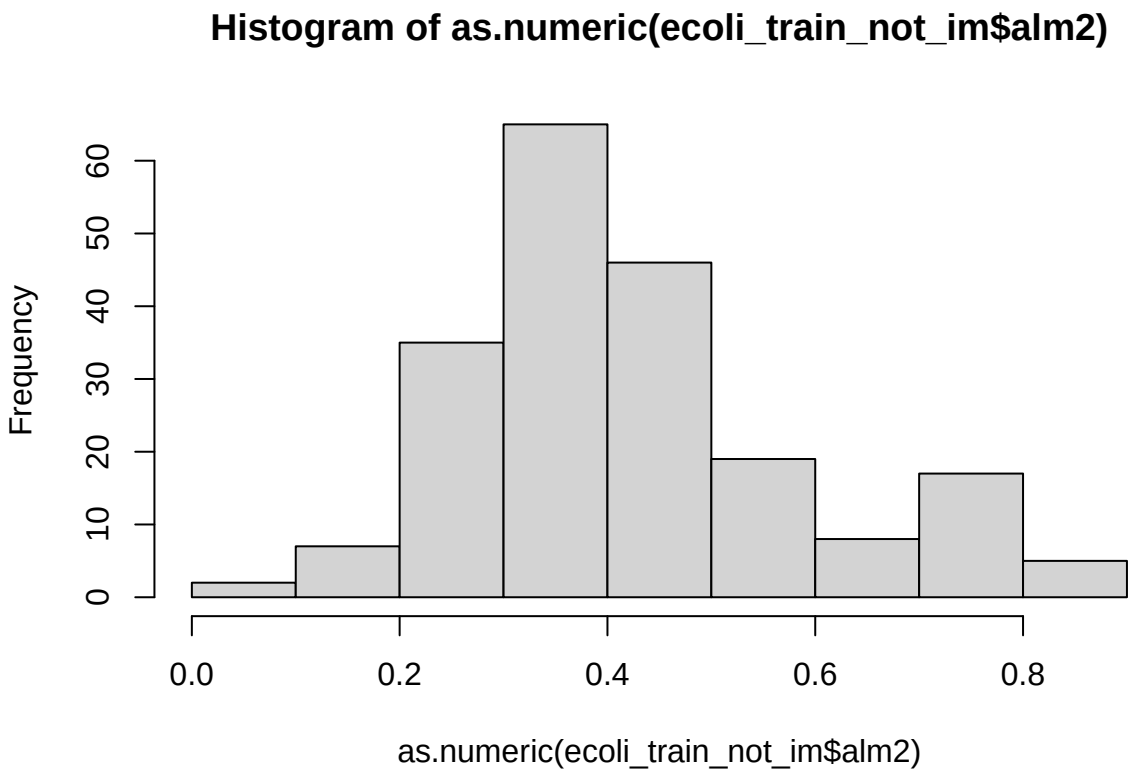
The im histogram shows that im is most frequently predicted when alm2 score ranges from 0.7 to 0.8.

The not-im histogram shows that when the alm2 score is between 0.3 and 0.4, the predicted locsite is generally not im.

```
hist(as.numeric(ecoli_train_im$alm2))
```




```
hist(as.numeric(ecoli_train_not_im$alm2))
```

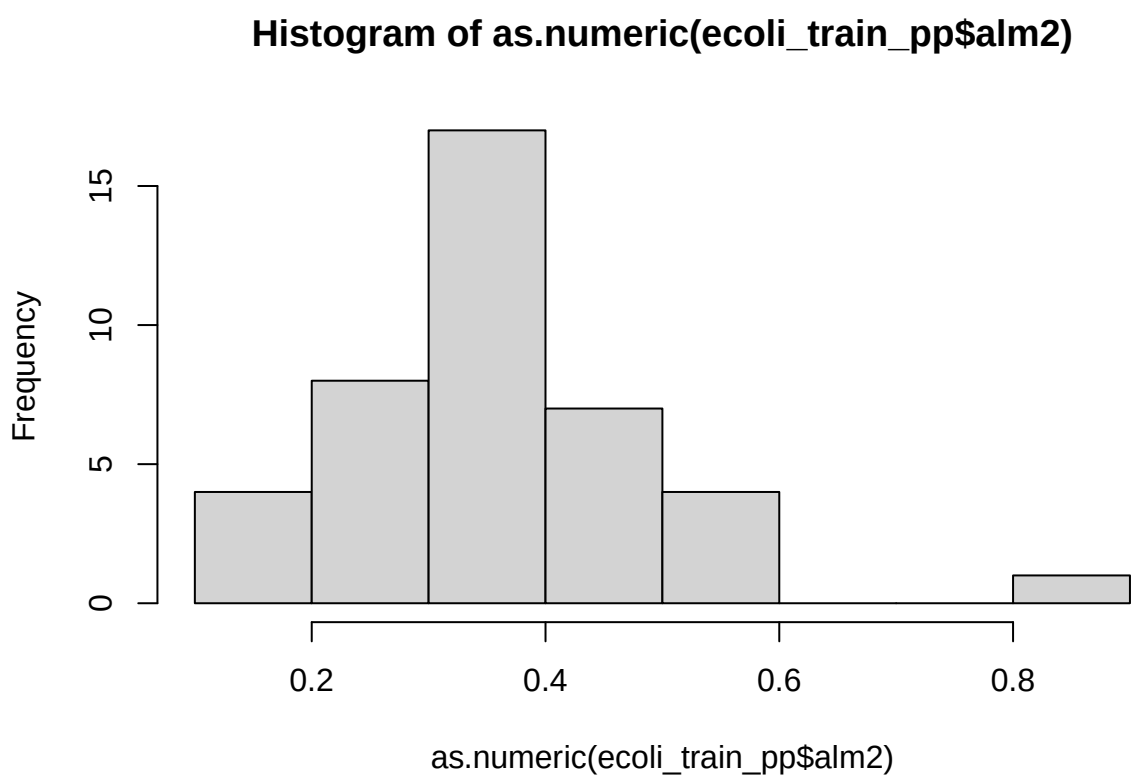


51. alm2 vs. pp and not pp

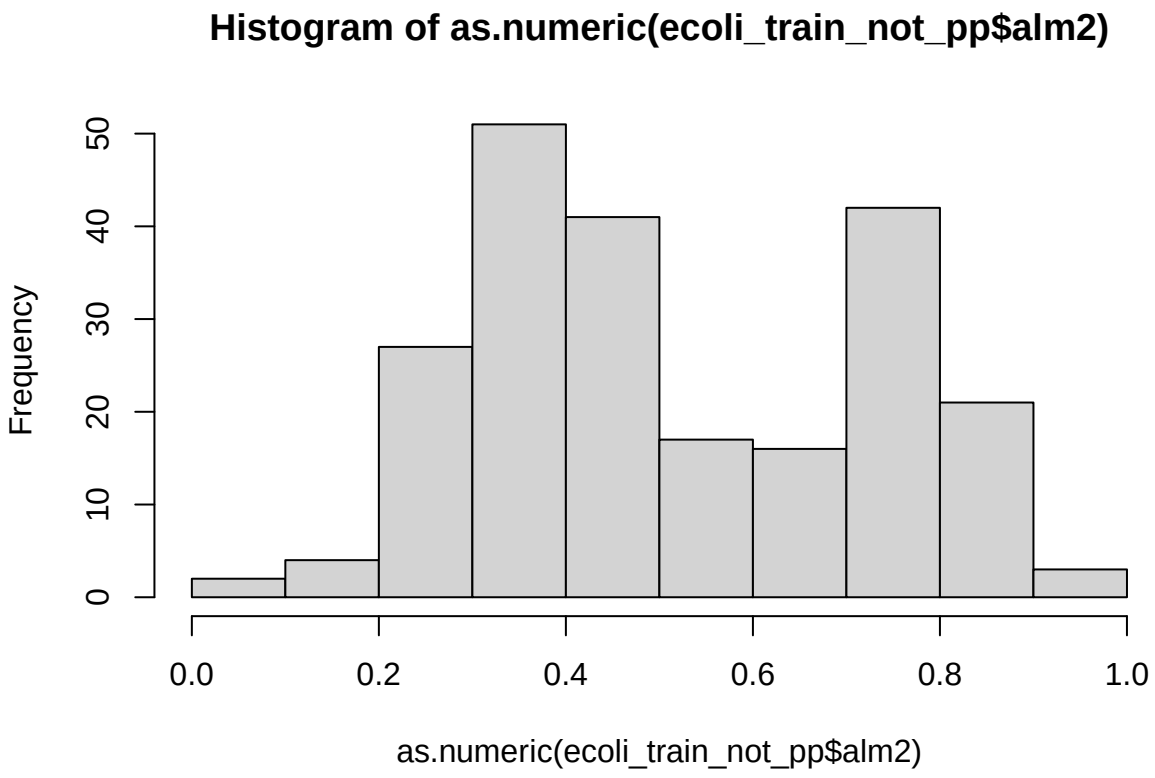
The pp histogram shows that pp is most frequently predicted when alm2 score ranges from 0.3 to 0.4.

The not-pp histogram shows that when the alm2 score is between 0.3 and 0.4, the predicted locsite is generally not pp. This histogram has a higher frequency than the first, hence more likely to be credible pp behavior.

```
hist(as.numeric(ecoli_train_pp$alm2))
```




```
hist(as.numeric(ecoli_train_not_pp$alm2))
```

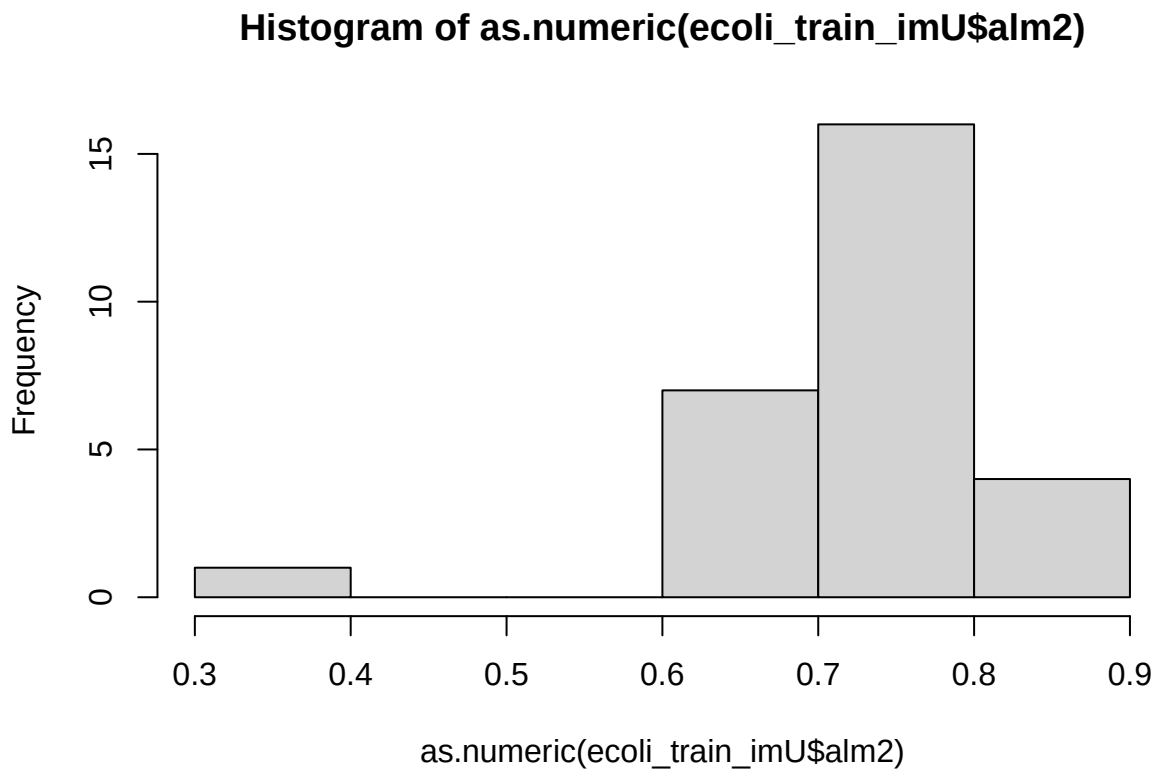


52. alm2 vs. imU and not imU

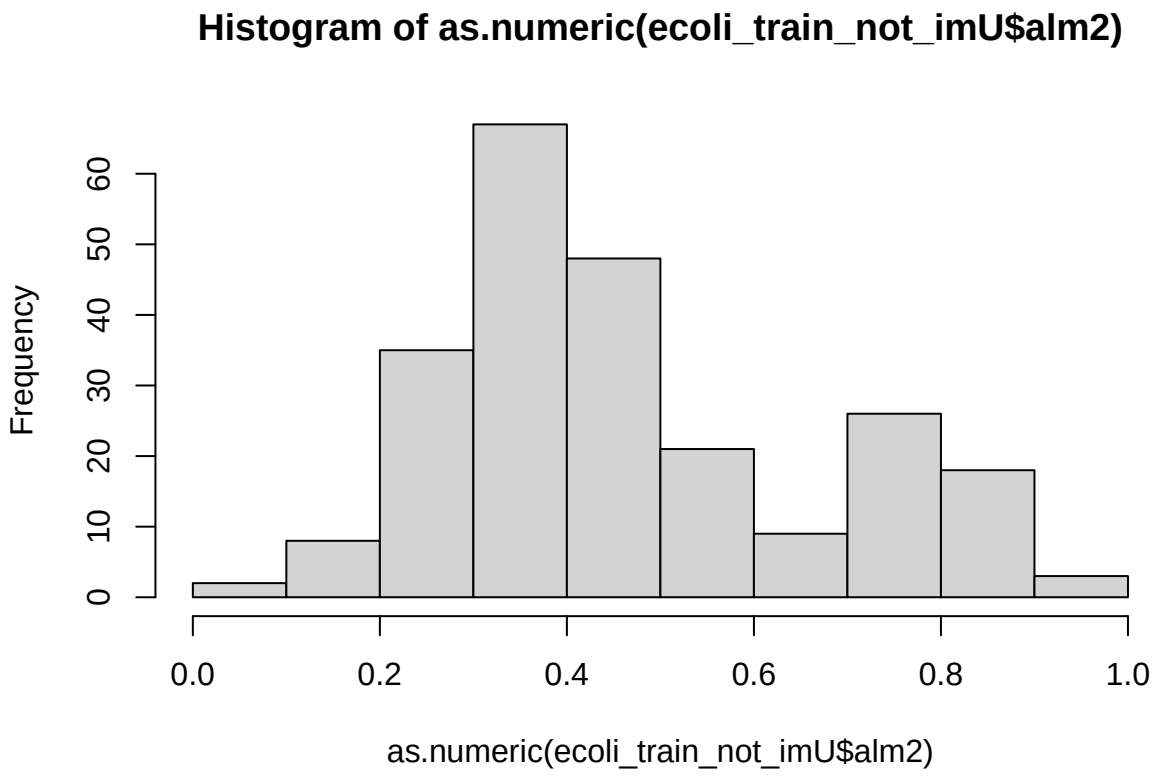
The imU histogram shows that imU is most frequently predicted when alm2 score ranges from 0.7 to 0.8.

The not-imU histogram shows that when the alm2 score is between 0.3 and 0.4, the predicted locsite is generally not imU.

```
hist(as.numeric(ecoli_train_imU$alm2))
```




```
hist(as.numeric(ecoli_train_not_imU$alm2))
```

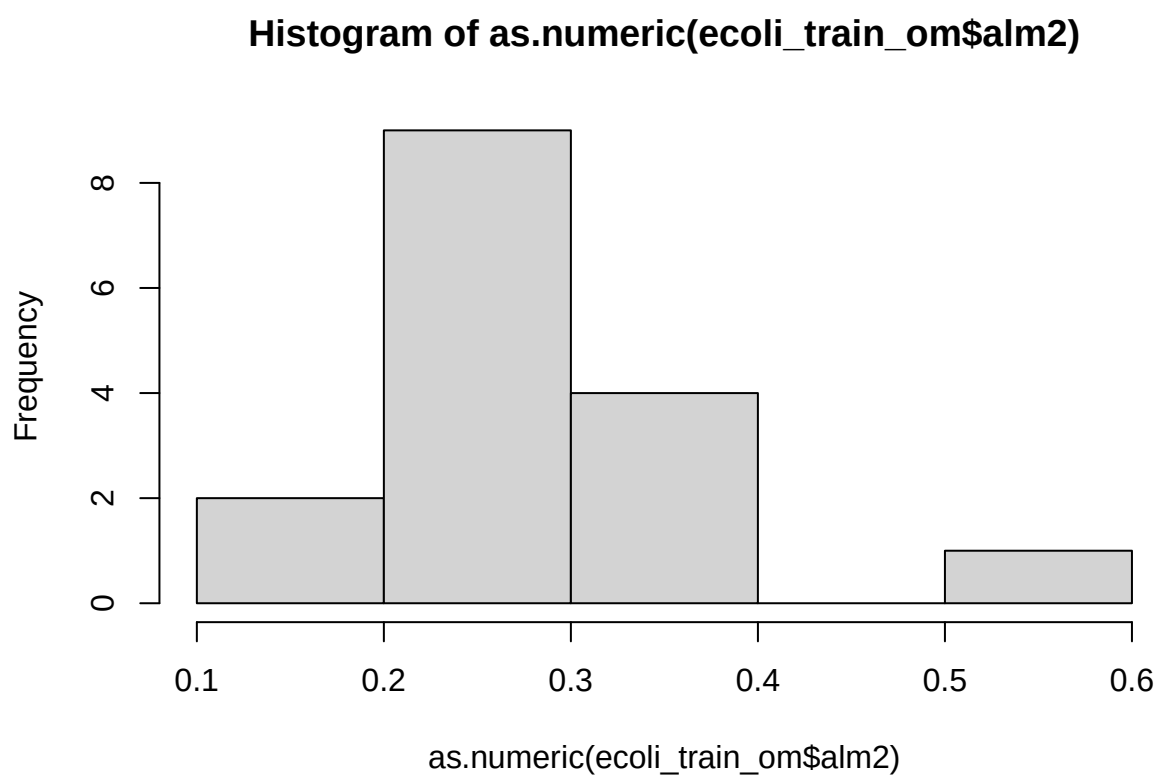


53. alm2 vs. om and not om

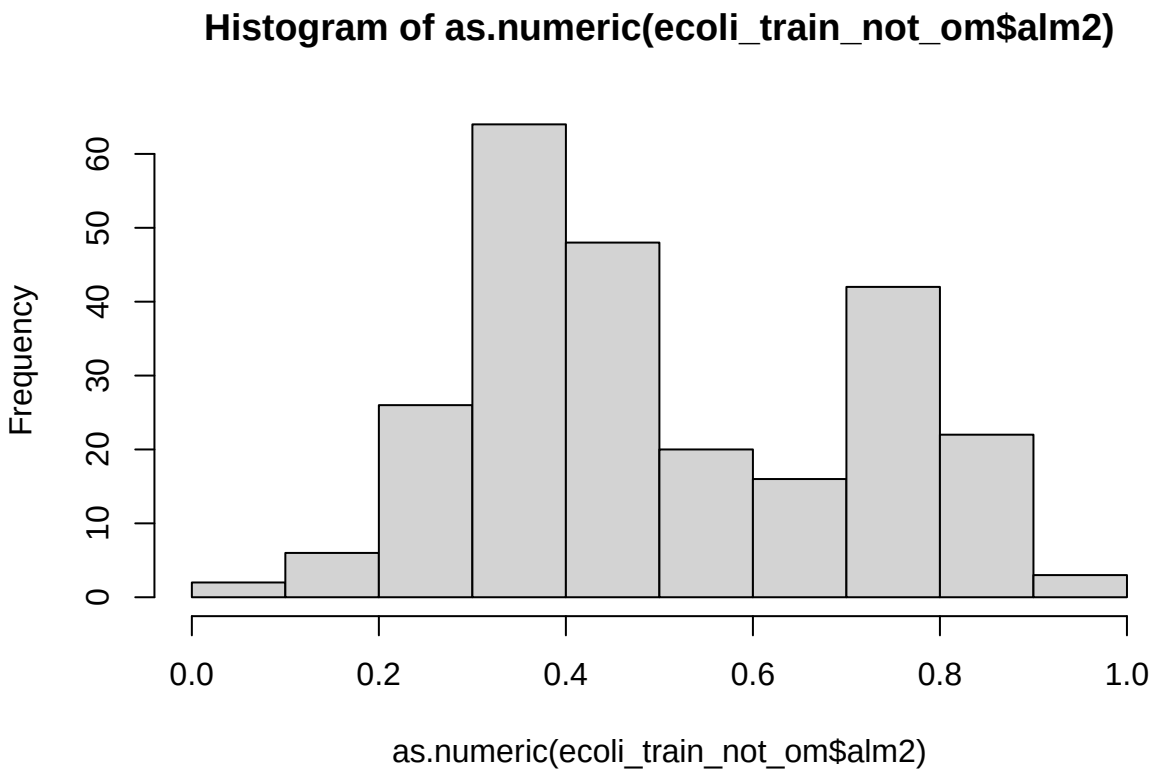
The om histogram shows that om is most frequently predicted when alm2 score ranges from 0.2 to 0.3.

The not-om histogram shows that when the alm2 score is between 0.3 and 0.4, the predicted locsite is generally not om.

```
hist(as.numeric(ecoli_train_om$alm2))
```




```
hist(as.numeric(ecoli_train_not_om$alm2))
```

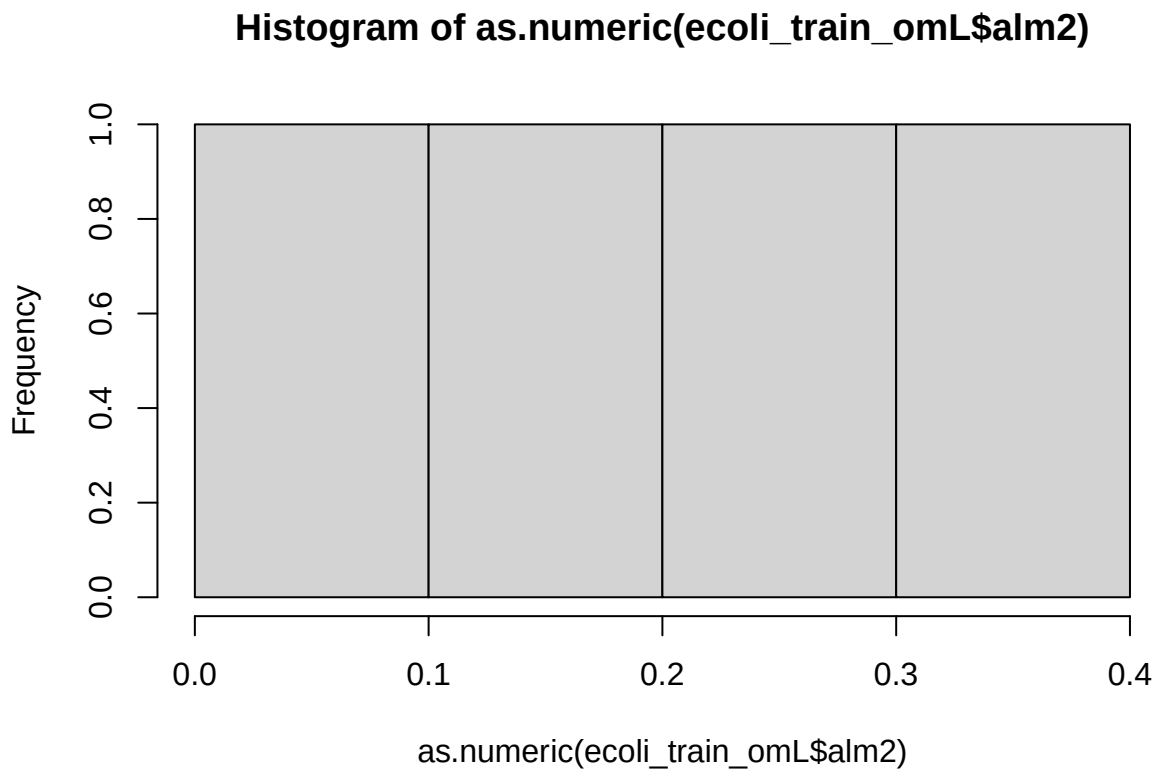


54. alm2 vs. omL and not omL

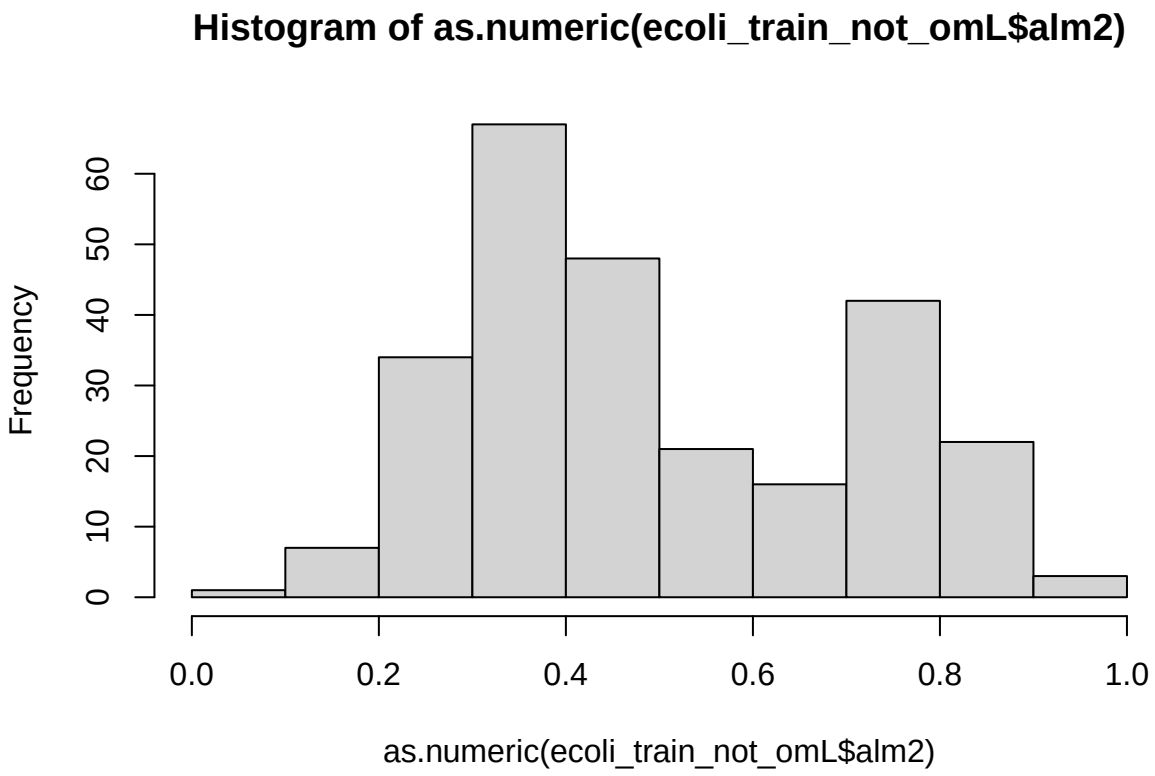
There is only one instance in the omL histogram. No specific alm2 score range can be distinguished from this visualisation.

The not-omL histogram shows that when the alm2 score is between 0.3 and 0.4, the predicted locsite is generally not omL.

```
hist(as.numeric(ecoli_train_omL$alm2))
```




```
hist(as.numeric(ecoli_train_not_omL$alm2))
```

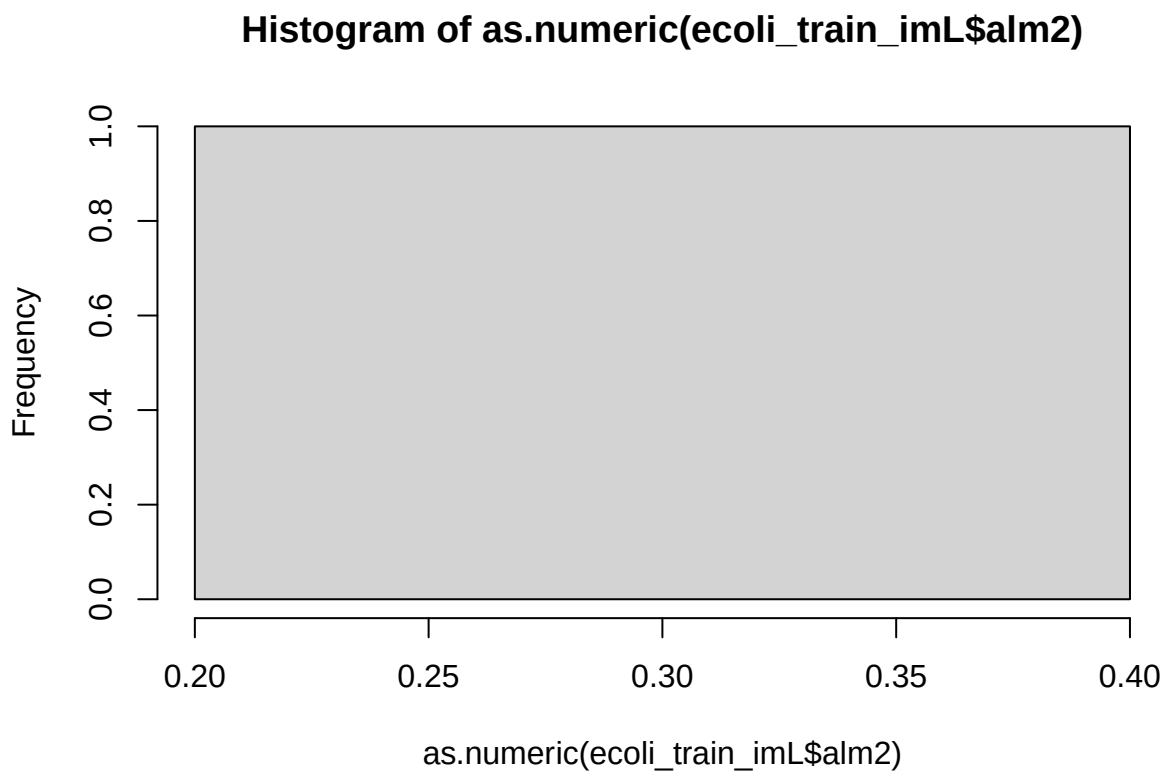


55. alm2 vs. imL and not imL

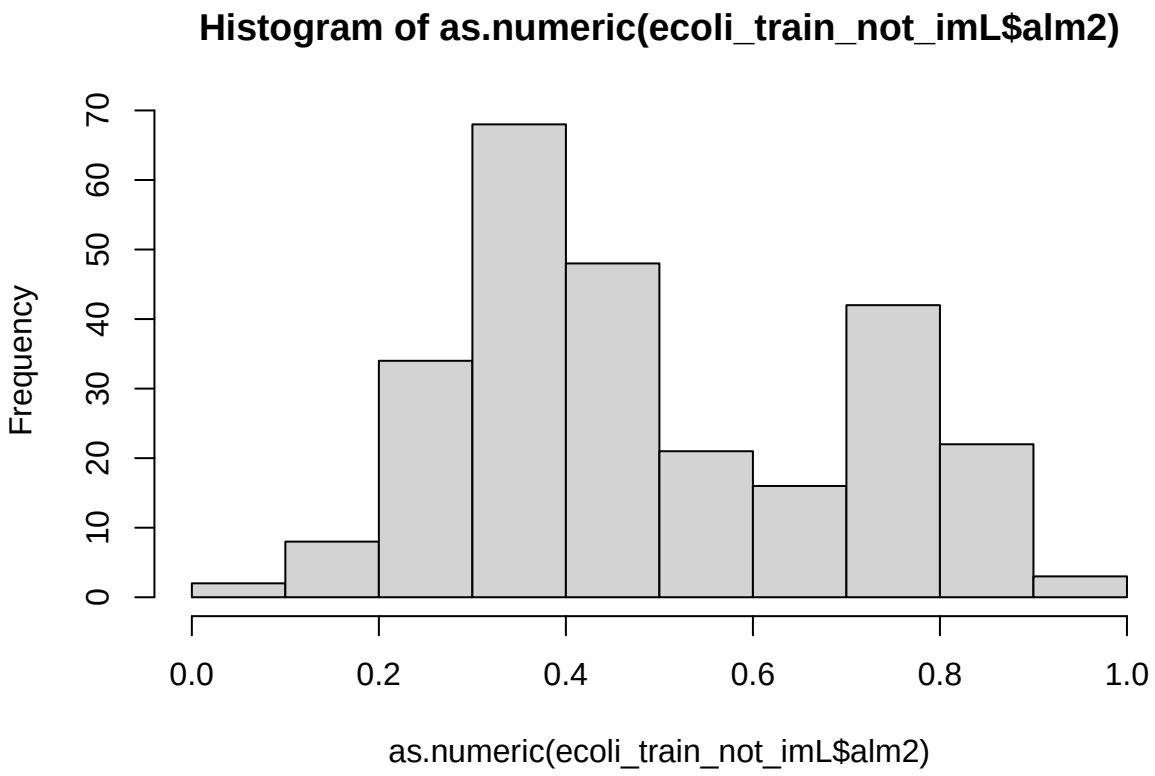
There is only one instance in the imL histogram. No specific alm2 score range can be distinguished from this visualisation.

The not-imL histogram shows that when the alm2 score is between 0.3 and 0.4, the predicted locsite is generally not imL.

```
hist(as.numeric(ecoli_train_imL$alm2))
```




```
hist(as.numeric(ecoli_train_not_imL$alm2))
```

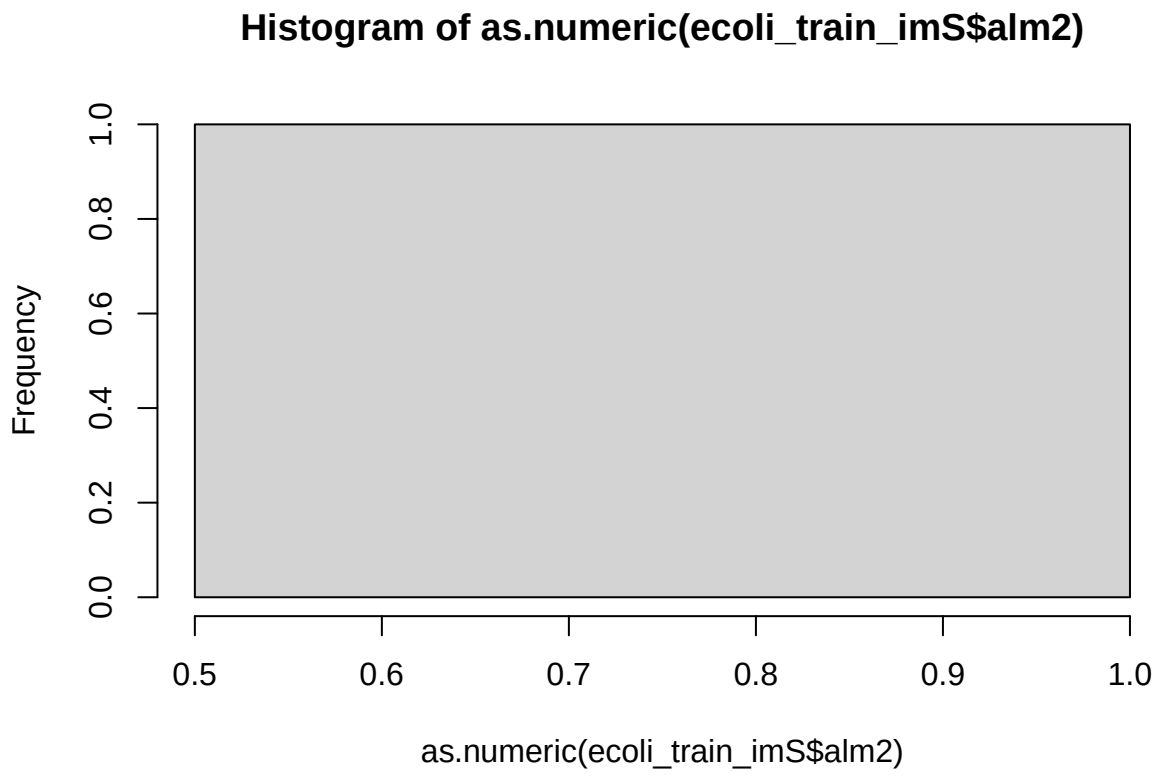


56. alm2 vs. imS and not imS

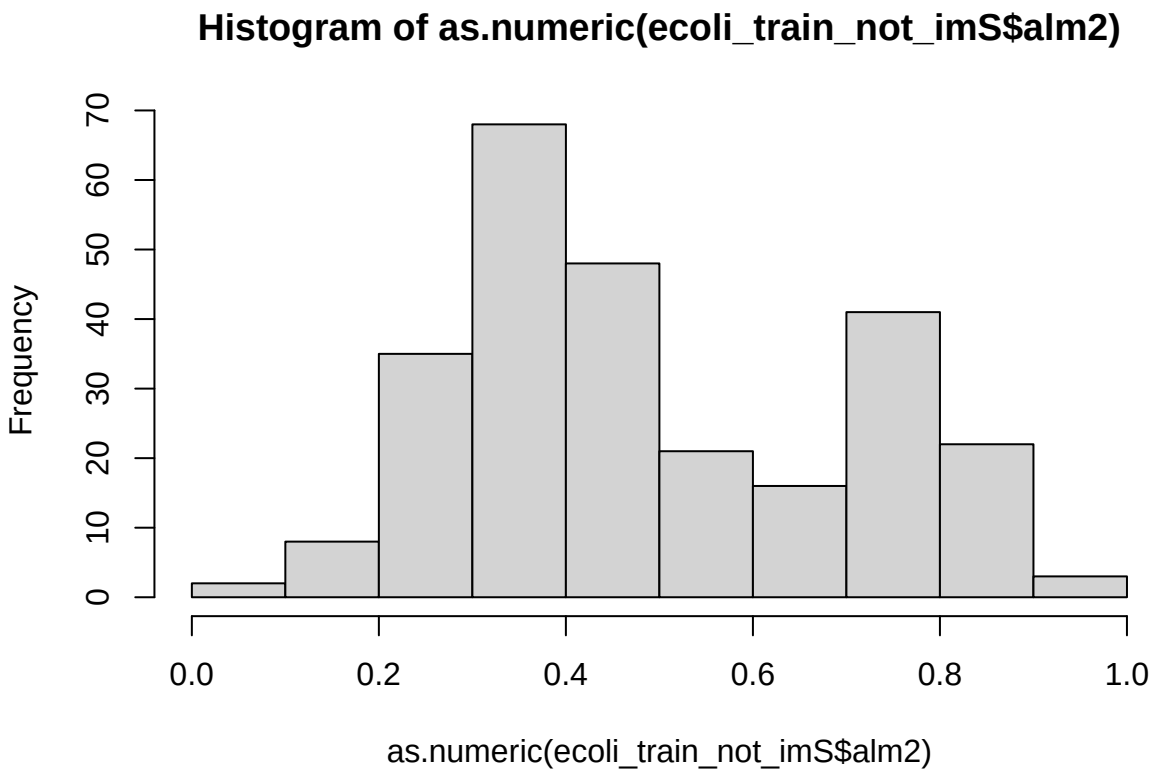
There is only one instance in the imS histogram. No specific alm2 score range can be distinguished from this visualisation.

The not-imS histogram shows that when the alm2 score is between 0.3 and 0.4, the predicted locsite is generally not imS.

```
hist(as.numeric(ecoli_train_imS$alm2))
```




```
hist(as.numeric(ecoli_train_not_imS$alm2))
```



PROPOSED MODELING APPROACHES

Baseline Accuracy

The ecoli data set was used by Paul Horton and Kenta Nakai in their published work, "A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins." In that study, they achieved an accuracy of 81% in classifying 336 E coli proteins into 8 classes.

81% is the baseline accuracy for this project which the proposed modeling approaches in the next sections - KNN and random forest - will aim to improve on.

Remove seqname from Ecoli Train and Test Data Sets

The ecoli train data set is the only data set used to develop the Cross-Validated KNN and Random Forest models.

Since it was shown in the Data Profile of Predictor Variables that seqname is a unique identifier for each instance in the ecoli train and test data sets, seqname is removed from these data sets as follows:

```
ecoli_train_seqname_out <- subset(ecoli_train, select = -c(seqname))

ecoli_test_seqname_out <- subset(ecoli_test, select = -c(seqname))

head(ecoli_train_seqname_out)
```

```
##     mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 1  0.07 0.40 0.48 0.50 0.54 0.35 0.44      cp
## 2  0.56 0.40 0.48 0.50 0.49 0.37 0.46      cp
## 3  0.59 0.49 0.48 0.50 0.52 0.45 0.36      cp
## 4  0.23 0.32 0.48 0.50 0.55 0.25 0.35      cp
## 5  0.67 0.39 0.48 0.50 0.36 0.38 0.46      cp
## 6  0.29 0.28 0.48 0.50 0.44 0.23 0.34      cp
```

```
head(ecoli_test_seqname_out)
```

```
##      mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 7   0.21 0.34 0.48 0.50 0.51 0.28 0.39      cp
## 14  0.22 0.43 0.48 0.50 0.48 0.16 0.28      cp
## 21  0.31 0.23 0.48 0.50 0.73 0.05 0.14      cp
## 33  0.40 0.29 0.48 0.50 0.42 0.35 0.44      cp
## 34  0.24 0.35 0.48 0.50 0.31 0.19 0.31      cp
## 35  0.36 0.54 0.48 0.50 0.41 0.38 0.46      cp
```

Model 1: Cross-Validated K Nearest Neighbours (CV KNN)

Rationale

KNN is recommended when data is properly labeled, data is noise-free, and the data set is small (Kumar 2020). All these three characteristics describe the ecoli data set.

The KNN algorithm, which is a type of lazy learning where the computation for predictions is deferred until classification, makes highly accurate predictions compared with other accurate models (IBM 2021).

The ecoli dataset is used for medical research that can have impact on life-changing decisions by doctors and medical researchers. In such circumstances, it is crucial to use a prediction model with reliable and high accuracy such as KNN.

The computational cost of KNN may be high due to lazy learning but with a reasonably small data set such as the ecoli data set, such increased cost may pale relative to the increased prediction accuracy that the KNN offers.

Model Development

The train function with knn method is applied to the ecoli train data set without the seqname vector below. Cross validation is also applied using the trainControl function at the same time.

However, when the model was applied to the ecoli test data set without the seqname vector, the prediction code encounters an error.

```
set.seed(8)
train_knn_cv <- train(locsite ~.,
                      method = "knn",
                      data = ecoli_train_seqname_out,
                      tuneGrid = data.frame(k = seq(3, 51, 2)),
                      trControl = trainControl(method = "cv", number = 10, p = 0.9))
```

```
## Warning: predictions failed for Fold07: k= 3 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent
```

```
## Warning: predictions failed for Fold07: k= 5 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k= 7 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k= 9 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=11 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=13 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=15 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=17 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=19 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=21 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=23 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=25 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=27 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=29 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=31 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=33 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=35 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=37 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent
```

```
## Warning: predictions failed for Fold07: k=39 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=41 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=43 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=45 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=47 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=49 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold07: k=51 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k= 3 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k= 5 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k= 7 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k= 9 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=11 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=13 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=15 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=17 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=19 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=21 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent
```

```
## Warning: predictions failed for Fold10: k=23 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=25 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=27 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=29 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=31 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=33 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=35 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=37 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=39 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=41 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=43 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=45 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=47 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=49 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning: predictions failed for Fold10: k=51 Error in dimnames(x) <- dn :
##   length of 'dimnames' [2] not equal to array extent

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
#The best value for k that maximizes accuracy is:
train_knn_cv$bestTune
```

```
##   k
## 4 9
```

```
#It has been found that below is an erroneous prediction code:
#knn_cv_preds <- predict(train_knn_cv, ecoli_test_seqname_out)
```

The code: "knn_cv_preds <- predict(train_knn_cv, ecoli_test_seqname_out)" results into an error that reveals that the ecoli_test_seqname_out has factors, such as mcg, that contain new levels that are not in ecoli_train_seqname_out.

The error encountered is: "Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels)" factor mcg has new levels 0.00, 0.26, 0.36, 0.37, 0.39, 0.62, 0.69, 0.76, 0.10, 0.28, 0.32, 0.85, 0.89 Calls: ... predict. train -> model.frame -> model.frame.default Execution halted."

To correct this error, set all observations in the ecoli test data set that represent the level that does not exist in the ecoli train data set to "NA"

```
ecoli_test_seqname_out_new<- ecoli_test_seqname_out

ecoli_test_seqname_out_new$mcg[which(!(ecoli_test_seqname_out_new$mcg %in% unique (ecoli_train_seqname_

ecoli_test_seqname_out_new$gvh[which(!(ecoli_test_seqname_out_new$gvh %in% unique (ecoli_train_seqname_

ecoli_test_seqname_out_new$lip[which(!(ecoli_test_seqname_out_new$lip %in% unique (ecoli_train_seqname_

ecoli_test_seqname_out_new$chg[which(!(ecoli_test_seqname_out_new$chg %in% unique (ecoli_train_seqname_

ecoli_test_seqname_out_new$aac[which(!(ecoli_test_seqname_out_new$aac %in% unique (ecoli_train_seqname_

ecoli_test_seqname_out_new$alm1[which(!(ecoli_test_seqname_out_new$alm1 %in% unique (ecoli_train_seqnam

ecoli_test_seqname_out_new$alm2[which(!(ecoli_test_seqname_out_new$alm2 %in% unique (ecoli_train_seqnam

# As shown below, the ecoli_test_seqname_out_new now contains the <NA> that replaced the values of the

ecoli_test_seqname_out_new
```

```
##       mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 7    0.21 0.34 0.48 0.50 0.51 0.28 0.39      cp
## 14   0.22 0.43 0.48 0.50 0.48 0.16 0.28      cp
## 21   0.31 <NA> 0.48 0.50 0.73 <NA> <NA>      cp
## 33   0.40 0.29 0.48 0.50 0.42 0.35 0.44      cp
## 34   0.24 0.35 0.48 0.50 0.31 <NA> 0.31      cp
## 35   0.36 0.54 0.48 0.50 0.41 0.38 0.46      cp
## 37   0.65 0.47 0.48 0.50 0.59 0.30 0.40      cp
## 42   0.40 0.50 0.48 0.50 0.45 0.39 0.47      cp
## 43   <NA> 0.38 0.48 0.50 0.50 0.33 0.42      cp
## 44   0.61 0.45 0.48 0.50 0.48 0.35 0.41      cp
## 51   0.29 0.30 0.48 0.50 0.45 <NA> 0.17      cp
## 68   0.25 0.37 0.48 0.50 0.43 0.26 0.36      cp
```

```
## 70   0.26 0.26 0.48 0.50 0.34 0.25 0.35      cp
## 73   0.44 0.42 0.48 0.50 0.42 0.25 0.20      cp
## 74   0.24 0.43 0.48 0.50 0.37 0.28 0.38      cp
## 79   0.29 0.41 0.48 0.50 0.48 0.38 0.46      cp
## 84   <NA> 0.56 0.48 0.50 0.43 0.45 0.53      cp
## 85   0.40 0.46 0.48 0.50 0.52 0.49 0.56      cp
## 89   0.53 0.42 0.48 0.50 <NA> 0.29 0.39      cp
## 101  0.24 0.57 0.48 0.50 0.63 0.34 0.43      cp
## 105  <NA> 0.50 0.48 0.50 0.44 0.32 0.41      cp
## 106  0.44 0.49 0.48 0.50 0.39 0.38 0.40      cp
## 110  <NA> 0.33 0.48 0.50 0.60 <NA> 0.20      cp
## 128  0.23 0.34 0.48 0.50 0.43 0.26 0.37      cp
## 129  <NA> 0.44 0.48 0.50 0.42 0.39 0.47      cp
## 130  <NA> 0.38 0.48 0.50 0.42 0.48 <NA>      cp
## 131  <NA> 0.31 0.48 0.50 0.38 0.34 0.43      cp
## 134  0.17 0.52 0.48 0.50 0.49 0.37 0.46      cp
## 135  0.36 0.42 0.48 0.50 0.53 0.32 0.41      cp
## 148  0.43 0.40 0.48 0.50 0.58 0.75 0.78      im
## 156  0.73 0.36 0.48 0.50 0.53 <NA> 0.92      im
## 162  0.72 0.46 0.48 0.50 0.51 0.66 0.70      im
## 166  0.54 0.57 0.48 0.50 0.56 0.81 0.83      im
## 167  0.47 0.59 0.48 0.50 0.52 0.76 0.79      im
## 170  0.31 0.50 0.48 0.50 0.57 0.84 0.85      im
## 174  0.71 0.40 0.48 0.50 0.71 0.70 0.74      im
## 180  0.63 0.54 0.48 0.50 0.65 0.79 0.81      im
## 181  0.70 0.40 0.48 0.50 0.56 0.86 0.83      im
## 182  0.60 0.50 1.00 0.50 0.54 0.77 0.80      im
## 184  0.74 0.70 0.48 0.50 0.66 0.65 0.69      im
## 186  <NA> 0.55 0.48 0.50 0.51 0.72 0.76      im
## 193  0.34 0.67 0.48 0.50 0.52 0.76 0.79      im
## 212  0.46 0.59 0.48 0.50 0.36 0.76 0.23      im
## 215  <NA> 0.49 0.48 0.50 0.41 0.67 0.21      im
## 218  0.63 0.75 0.48 0.50 0.64 0.73 0.66      im
## 220  <NA> 0.53 0.48 0.50 0.53 0.52 0.35     imS
## 223  0.70 0.39 1.00 0.50 0.51 0.82 0.84     imL
## 224  0.72 0.42 0.48 0.50 0.65 0.77 0.79     imU
## 229  0.78 0.33 0.48 0.50 0.57 0.77 0.79     imU
## 237  0.77 0.55 0.48 0.50 0.51 0.78 0.74     imU
## 241  <NA> 0.42 0.48 0.50 0.58 0.79 0.81     imU
## 242  0.86 0.39 0.48 0.50 0.59 <NA> 0.90     imU
## 245  0.47 0.46 0.48 0.50 0.62 0.74 0.77     imU
## 253  0.84 0.54 0.48 0.50 0.75 0.92 0.70     imU
## 264  0.74 <NA> 0.48 0.50 0.57 0.53 0.29      om
## 266  0.75 0.76 0.48 0.50 0.83 0.57 0.30      om
## 270  0.56 0.68 0.48 0.50 0.77 0.36 0.45      om
## 277  0.71 0.71 0.48 0.50 <NA> <NA> 0.36      om
## 281  0.71 0.46 1.00 0.50 0.52 0.59 0.30     omL
## 296  0.66 0.74 0.48 0.50 0.31 0.38 0.43      pp
## 300  <NA> 0.39 0.48 0.50 0.53 0.28 0.38      pp
## 303  0.74 0.82 0.48 0.50 0.49 0.49 0.41      pp
## 306  0.63 0.71 0.48 0.50 0.60 0.40 0.39      pp
## 308  0.68 0.78 0.48 0.50 0.43 0.44 0.42      pp
## 312  0.63 0.65 0.48 0.50 0.39 0.44 0.35      pp
## 319  <NA> 0.59 0.48 0.50 0.46 0.44 0.52      pp
```

```
## 322  0.62 0.78 0.48 0.50 0.47 0.49 0.54      pp
## 323  <NA> <NA> 0.48 0.50 0.44 0.39 0.39      pp
## 331  0.74 0.56 0.48 0.50 0.47 <NA> 0.30      pp
## 333  0.61 0.60 0.48 0.50 0.44 0.39 0.38      pp
```

The same prediction code that resulted in an error earlier is tried again below with the updated test data set. No error was generated.

```
knn_cv_preds <- predict(train_knn_cv, ecoli_test_seqname_out_new)
knn_cv_preds
```

```
##  [1] cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp
## [20] cp  cp  cp  im  im  im  cp  im  im  im  cp  imU cp  im  im  im  cp  cp  cp
## [39] cp  cp  cp  cp  pp  cp  cp  cp  cp
## Levels: cp im imL imS imU om omL pp
```

```
#Accurancy of CV KNN Model
mean(knn_cv_preds == ecoli_test_seqname_out_new$locsite)
```

```
## Warning in `==.default`(knn_cv_preds, ecoli_test_seqname_out_new$locsite):
## longer object length is not a multiple of shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
## [1] 0.3714286
```

Unfortunately, the accuracy of the CV KNN Model is very poor and has not exceeded the baseline accuracy of the original study by Horton and Nakai.

Why this could have happened is explored in the Limitations section in the Conclusion.

Model 2: Random Forest

Rationale

Random forests are bagged decision tree models that split on a subset of features on each split (Kho 2018). The split happens such that from the set of available features n, a subset of m features are selected at random. This way, variance can be averaged away.

In the ecoli data set, it is possible that some predictors are very strong. Examples of these can be seen in the Visualisation and Insights section presented earlier. Without using random forest, these predictors will always be chosen at the top level of the trees such that trees will be similarly structured or highly correlated.

Using random forest prevents this from happening because of the random selection of features that may not all be strong predictors in one instance of selection.

Kho (2018) lists other relevant advantages of using random forest as:

1. Random forest provides high performance without requiring a lot of interpretation.
2. Random forest is great with high dimensional data since it works with subsets of data.

3. Random forest is faster to train than decision trees since it works only on a subset of features in the model. Prediction speed is significantly faster than training speed because generated forests can be used for future use.
4. Random forest is robust to outliers by binning them and is indifferent to non-linear features.
5. Random forest has methods to balance error in unbalanced data sets.
6. Each decision tree has a high variance but low bias. But because all trees in the random forest are averaged, the variance is averaged as well - hence, low bias and moderate variance model.

All the advantages listed above are potentially relevant to the multidimensional multivariate ecoli data set.

The loss of interpretability can be regarded as a disadvantage of random forests (Irizarry 2022). But this can be overcome by examining variable importance - i.e., the count of how often a predictor is used in the individual trees.

Model Development

```
set.seed(14)
train_rf <- train(locsite ~.,
                  data = ecoli_train_seqname_out,
                  method = "rf",
                  ntree = 100,
                  tuneGrid = data.frame(mtry = seq(1:7)))
```

```
## Warning: model fit failed for Resample01: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample01: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample01: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample01: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample01: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample01: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample01: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample03: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample03: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.
```

```
## Warning: model fit failed for Resample03: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample03: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample03: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample03: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample03: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample04: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample04: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample04: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample04: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample04: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample04: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample04: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample07: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample07: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample07: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample07: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample07: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.
```

```
## Warning: model fit failed for Resample07: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample07: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample09: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample09: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample09: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample09: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample09: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample09: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample09: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample12: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample12: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample12: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample12: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample12: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample12: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample12: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample13: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.
```

```
## Warning: model fit failed for Resample13: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample13: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample13: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample13: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample13: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample13: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample14: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample14: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample14: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample14: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample14: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample14: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample14: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample15: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample15: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample15: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample15: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.
```

```
## Warning: model fit failed for Resample15: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample15: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample15: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample16: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample16: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample16: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample16: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample16: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample16: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample16: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample18: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample18: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample18: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample18: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample18: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample18: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample18: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.
```

```
## Warning: model fit failed for Resample20: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample20: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample20: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample20: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample20: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample20: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample20: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample24: mtry=1 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample24: mtry=2 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample24: mtry=3 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample24: mtry=4 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample24: mtry=5 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample24: mtry=6 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning: model fit failed for Resample24: mtry=7 Error in randomForest.default(x, y, mtry = min(para
##   Can't have empty classes in y.

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
#The best value for mtry that maximizes accuracy is:
train_rf$bestTune
```

```
##   mtry
## 7    7
```

Apply the random forest model, train_rf, to the ecoli test data set.

```
#The ecoli_test_seqname_out_new is used also for the random forest prediction otherwise
#the same error is encountered.  The error detects that there are new levels in the test
#data set that are not in the train data set.

#There is also a need to set these new levels to NA as was done in ecoli_test_seqname_out_new
#earlier generated for the CV KNN model development.

rf_preds <- predict(train_rf, ecoli_test_seqname_out_new)
rf_preds
```

```
##  [1] cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp
## [20] cp  cp  im  cp  cp  im  cp  im  im  im  cp  imU im  im  im  im  cp  cp  cp
## [39] cp  cp  cp  cp  pp  cp  cp  cp  cp
## Levels: cp im imL imS imU om omL pp
```

```
#Accuracy of random forest
mean(rf_preds == ecoli_test_seqname_out$locsite)
```

```
## Warning in `==.default`(rf_preds, ecoli_test_seqname_out$locsite): longer object
## length is not a multiple of shorter object length
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
## [1] 0.4
```

```
#Most important variable
varImp(train_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 364)
##
##          Overall
## mcg0.63   100.00
## alm10.78   89.43
## alm10.71   78.03
## lip1.00    72.84
## alm20.74   70.79
## mcg0.69    67.69
## alm10.73   66.32
## mcg0.64    64.44
## mcg0.67    64.29
## alm10.54   63.30
## alm10.67   62.13
## alm20.79   57.76
## alm20.80   55.86
## gvh0.81    53.54
## alm20.68   52.43
## gvh0.86    51.15
## gvh0.84    50.84
```

```
## alm20.76   49.08
## gvh0.61    48.84
## alm10.76   48.31
```

Unfortunately, the accuracy level achieved by the random forest model falls below the baseline accuracy achieved by Horton and Nakai in the original study.

The possible reasons for this low accuracy are explained in the Limitations section in the Conclusion.

Comparison of Modeling Approaches

1. Accuracy

The random forest model has a slightly higher accuracy than the CV KNN model.

2. Error Generated

Both modeling approaches, however, generated an error due to having new levels in certain factors or vectors in the test data set that are not found in the train data set. These levels had to be transformed into NA for the two modeling approaches to proceed in prediction generation.

3. Predictions vs. Actual Test Data

Both the CV KNN and Random Forest models generated only 47 predictions. But the actual test data had 70 observations. What accounts for the discrepancy of 23 observations?

```
# CV KNN Predictions
knn_cv_preds
```

```
##  [1] cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp
## [20] cp  cp  cp  im  im  im  cp  im  im  im  cp  imU cp  im  im  im  cp  cp  cp
## [39] cp  cp  cp  cp  pp  cp  cp  cp  cp
## Levels: cp im imL imS imU om omL pp
```

```
# Random Forest Predictions
rf_preds
```

```
##  [1] cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp
## [20] cp  cp  im  cp  cp  im  cp  im  im  im  cp  imU im  im  im  im  cp  cp  cp
## [39] cp  cp  cp  cp  pp  cp  cp  cp  cp
## Levels: cp im imL imS imU om omL pp
```

```
#Actual Test Data
ecoli_test_seqname_out_new$locsite
```

```
##  [1] "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"
## [13] "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"
## [25] "cp"  "cp"  "cp"  "cp"  "cp"  "im"  "im"  "im"  "im"  "im"  "im"  "im"
## [37] "im"  "im"  "im"  "im"  "im"  "im"  "im"  "im"  "im"  "imS" "imL" "imU"
## [49] "imU" "imU" "imU" "imU" "imU" "imU" "om"  "om"  "om"  "om"  "omL" "pp"
## [61] "pp"  "pp"  "pp"  "pp"  "pp"  "pp"  "pp"  "pp"  "pp"  "pp"
```

```
#Full Actual Test Data to show NA replacement
ecoli_test_seqname_out_new
```

```
##       mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 7    0.21 0.34 0.48 0.50 0.51 0.28 0.39      cp
## 14   0.22 0.43 0.48 0.50 0.48 0.16 0.28      cp
## 21   0.31 <NA> 0.48 0.50 0.73 <NA> <NA>      cp
## 33   0.40 0.29 0.48 0.50 0.42 0.35 0.44      cp
## 34   0.24 0.35 0.48 0.50 0.31 <NA> 0.31      cp
## 35   0.36 0.54 0.48 0.50 0.41 0.38 0.46      cp
## 37   0.65 0.47 0.48 0.50 0.59 0.30 0.40      cp
## 42   0.40 0.50 0.48 0.50 0.45 0.39 0.47      cp
## 43   <NA> 0.38 0.48 0.50 0.50 0.33 0.42      cp
## 44   0.61 0.45 0.48 0.50 0.48 0.35 0.41      cp
## 51   0.29 0.30 0.48 0.50 0.45 <NA> 0.17      cp
## 68   0.25 0.37 0.48 0.50 0.43 0.26 0.36      cp
## 70   0.26 0.26 0.48 0.50 0.34 0.25 0.35      cp
## 73   0.44 0.42 0.48 0.50 0.42 0.25 0.20      cp
## 74   0.24 0.43 0.48 0.50 0.37 0.28 0.38      cp
## 79   0.29 0.41 0.48 0.50 0.48 0.38 0.46      cp
## 84   <NA> 0.56 0.48 0.50 0.43 0.45 0.53      cp
## 85   0.40 0.46 0.48 0.50 0.52 0.49 0.56      cp
## 89   0.53 0.42 0.48 0.50 <NA> 0.29 0.39      cp
## 101  0.24 0.57 0.48 0.50 0.63 0.34 0.43      cp
## 105  <NA> 0.50 0.48 0.50 0.44 0.32 0.41      cp
## 106  0.44 0.49 0.48 0.50 0.39 0.38 0.40      cp
## 110  <NA> 0.33 0.48 0.50 0.60 <NA> 0.20      cp
## 128  0.23 0.34 0.48 0.50 0.43 0.26 0.37      cp
## 129  <NA> 0.44 0.48 0.50 0.42 0.39 0.47      cp
## 130  <NA> 0.38 0.48 0.50 0.42 0.48 <NA>      cp
## 131  <NA> 0.31 0.48 0.50 0.38 0.34 0.43      cp
## 134  0.17 0.52 0.48 0.50 0.49 0.37 0.46      cp
## 135  0.36 0.42 0.48 0.50 0.53 0.32 0.41      cp
## 148  0.43 0.40 0.48 0.50 0.58 0.75 0.78      im
## 156  0.73 0.36 0.48 0.50 0.53 <NA> 0.92      im
## 162  0.72 0.46 0.48 0.50 0.51 0.66 0.70      im
## 166  0.54 0.57 0.48 0.50 0.56 0.81 0.83      im
## 167  0.47 0.59 0.48 0.50 0.52 0.76 0.79      im
## 170  0.31 0.50 0.48 0.50 0.57 0.84 0.85      im
## 174  0.71 0.40 0.48 0.50 0.71 0.70 0.74      im
## 180  0.63 0.54 0.48 0.50 0.65 0.79 0.81      im
## 181  0.70 0.40 0.48 0.50 0.56 0.86 0.83      im
## 182  0.60 0.50 1.00 0.50 0.54 0.77 0.80      im
## 184  0.74 0.70 0.48 0.50 0.66 0.65 0.69      im
## 186  <NA> 0.55 0.48 0.50 0.51 0.72 0.76      im
## 193  0.34 0.67 0.48 0.50 0.52 0.76 0.79      im
## 212  0.46 0.59 0.48 0.50 0.36 0.76 0.23      im
## 215  <NA> 0.49 0.48 0.50 0.41 0.67 0.21      im
## 218  0.63 0.75 0.48 0.50 0.64 0.73 0.66      im
## 220  <NA> 0.53 0.48 0.50 0.53 0.52 0.35     imS
## 223  0.70 0.39 1.00 0.50 0.51 0.82 0.84     imL
## 224  0.72 0.42 0.48 0.50 0.65 0.77 0.79     imU
## 229  0.78 0.33 0.48 0.50 0.57 0.77 0.79     imU
```

```
## 237  0.77 0.55 0.48 0.50 0.51 0.78 0.74     imU
## 241  <NA> 0.42 0.48 0.50 0.58 0.79 0.81     imU
## 242  0.86 0.39 0.48 0.50 0.59 <NA> 0.90     imU
## 245  0.47 0.46 0.48 0.50 0.62 0.74 0.77     imU
## 253  0.84 0.54 0.48 0.50 0.75 0.92 0.70     imU
## 264  0.74 <NA> 0.48 0.50 0.57 0.53 0.29      om
## 266  0.75 0.76 0.48 0.50 0.83 0.57 0.30      om
## 270  0.56 0.68 0.48 0.50 0.77 0.36 0.45      om
## 277  0.71 0.71 0.48 0.50 <NA> <NA> 0.36      om
## 281  0.71 0.46 1.00 0.50 0.52 0.59 0.30     omL
## 296  0.66 0.74 0.48 0.50 0.31 0.38 0.43      pp
## 300  <NA> 0.39 0.48 0.50 0.53 0.28 0.38      pp
## 303  0.74 0.82 0.48 0.50 0.49 0.49 0.41      pp
## 306  0.63 0.71 0.48 0.50 0.60 0.40 0.39      pp
## 308  0.68 0.78 0.48 0.50 0.43 0.44 0.42      pp
## 312  0.63 0.65 0.48 0.50 0.39 0.44 0.35      pp
## 319  <NA> 0.59 0.48 0.50 0.46 0.44 0.52      pp
## 322  0.62 0.78 0.48 0.50 0.47 0.49 0.54      pp
## 323  <NA> <NA> 0.48 0.50 0.44 0.39 0.39      pp
## 331  0.74 0.56 0.48 0.50 0.47 <NA> 0.30      pp
## 333  0.61 0.60 0.48 0.50 0.44 0.39 0.38      pp
```

```
nrow(ecoli_test_seqname_out_new)
```

```
## [1] 70
```

From a review of the full actual test data set, there were 23 rows that had one or more "NA" in certain factors. These rows (observation number) are 3(21), 5(34), 9(43), 11(51), 17(84), 19(89), 21(105), 23(110), 25(129), 26(130), 27(131), 31(156), 41(186), 44(215), 46(220), 51(241), 52(242), 55(264), 58(277), 61(300), 66(319), 68(323), and 69(331). Apparently, these 23 rows were ignored by both the CV KNN and Random Forest models.

Because the calculation of Accuracy, which uses the mean function, for both models compared their predictions (47) with actual test data observations (70), the resulting accuracies were very low. Both models missed 23 predictions which were considered null and therefore not equal to their counterparts in the actual test data set.

Re-calculation of CV KNN and Random Forest Accuracy Based on Removal of 23 NA Rows

```
NA_removed_test <- ecoli_test_seqname_out_new[-c(3,5,9,11,17,19,21,23,25,26,27,31,41,44,46,51,52,55,58,
NA_removed_test
```

```
##       mcg  gvh  lip  chg  aac alm1 alm2 locsite
## 7    0.21 0.34 0.48 0.50 0.51 0.28 0.39      cp
## 14   0.22 0.43 0.48 0.50 0.48 0.16 0.28      cp
## 33   0.40 0.29 0.48 0.50 0.42 0.35 0.44      cp
## 35   0.36 0.54 0.48 0.50 0.41 0.38 0.46      cp
## 37   0.65 0.47 0.48 0.50 0.59 0.30 0.40      cp
## 42   0.40 0.50 0.48 0.50 0.45 0.39 0.47      cp
## 44   0.61 0.45 0.48 0.50 0.48 0.35 0.41      cp
```

```
## 68   0.25 0.37 0.48 0.50 0.43 0.26 0.36      cp
## 70   0.26 0.26 0.48 0.50 0.34 0.25 0.35      cp
## 73   0.44 0.42 0.48 0.50 0.42 0.25 0.20      cp
## 74   0.24 0.43 0.48 0.50 0.37 0.28 0.38      cp
## 79   0.29 0.41 0.48 0.50 0.48 0.38 0.46      cp
## 85   0.40 0.46 0.48 0.50 0.52 0.49 0.56      cp
## 101  0.24 0.57 0.48 0.50 0.63 0.34 0.43      cp
## 106  0.44 0.49 0.48 0.50 0.39 0.38 0.40      cp
## 128  0.23 0.34 0.48 0.50 0.43 0.26 0.37      cp
## 134  0.17 0.52 0.48 0.50 0.49 0.37 0.46      cp
## 135  0.36 0.42 0.48 0.50 0.53 0.32 0.41      cp
## 148  0.43 0.40 0.48 0.50 0.58 0.75 0.78      im
## 162  0.72 0.46 0.48 0.50 0.51 0.66 0.70      im
## 166  0.54 0.57 0.48 0.50 0.56 0.81 0.83      im
## 167  0.47 0.59 0.48 0.50 0.52 0.76 0.79      im
## 170  0.31 0.50 0.48 0.50 0.57 0.84 0.85      im
## 174  0.71 0.40 0.48 0.50 0.71 0.70 0.74      im
## 180  0.63 0.54 0.48 0.50 0.65 0.79 0.81      im
## 181  0.70 0.40 0.48 0.50 0.56 0.86 0.83      im
## 182  0.60 0.50 1.00 0.50 0.54 0.77 0.80      im
## 184  0.74 0.70 0.48 0.50 0.66 0.65 0.69      im
## 193  0.34 0.67 0.48 0.50 0.52 0.76 0.79      im
## 212  0.46 0.59 0.48 0.50 0.36 0.76 0.23      im
## 218  0.63 0.75 0.48 0.50 0.64 0.73 0.66      im
## 223  0.70 0.39 1.00 0.50 0.51 0.82 0.84     imL
## 224  0.72 0.42 0.48 0.50 0.65 0.77 0.79     imU
## 229  0.78 0.33 0.48 0.50 0.57 0.77 0.79     imU
## 237  0.77 0.55 0.48 0.50 0.51 0.78 0.74     imU
## 245  0.47 0.46 0.48 0.50 0.62 0.74 0.77     imU
## 253  0.84 0.54 0.48 0.50 0.75 0.92 0.70     imU
## 266  0.75 0.76 0.48 0.50 0.83 0.57 0.30      om
## 270  0.56 0.68 0.48 0.50 0.77 0.36 0.45      om
## 281  0.71 0.46 1.00 0.50 0.52 0.59 0.30     omL
## 296  0.66 0.74 0.48 0.50 0.31 0.38 0.43      pp
## 303  0.74 0.82 0.48 0.50 0.49 0.49 0.41      pp
## 306  0.63 0.71 0.48 0.50 0.60 0.40 0.39      pp
## 308  0.68 0.78 0.48 0.50 0.43 0.44 0.42      pp
## 312  0.63 0.65 0.48 0.50 0.39 0.44 0.35      pp
## 322  0.62 0.78 0.48 0.50 0.47 0.49 0.54      pp
## 333  0.61 0.60 0.48 0.50 0.44 0.39 0.38      pp
```

```
nrow(NA_removed_test)
```

```
## [1] 47
```

```
# CV KNN
knn_cv_preds_NA_removed <- predict(train_knn_cv, NA_removed_test)
knn_cv_preds_NA_removed
```

```
##  [1] cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp
## [20] im  cp  cp  im  im  im  cp  im  im  im  cp  imU im  im  im  im  cp  cp  pp
## [39] cp  cp  cp  cp  pp  cp  cp  cp  cp
## Levels: cp im imL imS imU om omL pp
```

```
mean(knn_cv_preds_NA_removed == NA_removed_test$locsite)
```

```
## [1] 0.5531915
```

```
# Random Forest
rf_preds_NA_removed <- predict(train_rf, NA_removed_test)
rf_preds_NA_removed
```

```
##  [1] cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp  cp
## [20] cp  cp  im  cp  cp  im  cp  im  im  im  cp  imU im  im  im  im  cp  cp  cp
## [39] cp  cp  cp  cp  pp  cp  cp  cp  cp
## Levels: cp im imL imS imU om omL pp
```

```
#Accuracy of random forest
mean(rf_preds_NA_removed == NA_removed_test$locsite)
```

```
## [1] 0.5106383
```

```
#Actual Test Data NA Removed
NA_removed_test$locsite
```

```
##  [1] "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "cp"
## [13] "cp"  "cp"  "cp"  "cp"  "cp"  "cp"  "im"  "im"  "im"  "im"  "im"  "im"
## [25] "im"  "im"  "im"  "im"  "im"  "im"  "im"  "imL" "imU" "imU" "imU" "imU"
## [37] "imU" "om"  "om"  "omL" "pp"  "pp"  "pp"  "pp"  "pp"  "pp"  "pp"
```

The accuracy measures for both CV KNN and Random Forest significantly improved relative to their initial values but still below the baseline accuracy achieved by Horton and Nakai in their original study.

This time, however, CV KNN accuracy is higher than that of Random Forest. Both accuracies are now higher than 50% - i.e., half of the time, both models can be relied on to predict the locsite.

CONCLUSION

Summary

The Ecoli Dataset consisting of 335 observations was used for the purpose of developing a machine learning algorithm model that can predict the localization site (locsite) of E. coli proteins based on scores.

There are eight possible localization sites, the predicted variable: cp, im, pp, imU, om, omL, imL, and imS.

Seven scores are used as predictors: mcg, gvh, lip, chg, aac, alm1, and alm2.

The seqname is also given for every observation but was not used because it is not a score. Instead, it is the unique identifier of every observation and as such cannot be reasonably regarded as a predictor. A person's Tax Identification Number (TIN), a unique identifier, cannot be assumed to be able to predict the location of the person. The TIN does not influence where a person is located. But the selected seven predictor scores can be reasonably assumed to influence or provide information on the localization of the E.coli protein.

The downloaded Ecoli data set had a number of data issues which were resolved by the project: 1) transformation of data set from matrix to data frame, assigning names to unnamed columns; 2) data error combining a score and a locsite value in one column; and, 3) space in front of some locsite values.

After all data issues were corrected, the ecoli data set was partitioned into train and test data sets. The train data set has 265 observations and the test data set has 70 observations in an 80-20 split.

Profiling the train data set, histograms and insights were presented to show the behaviour of every predictor vis-a-vis the presence or absence of every locsite.

To start the model development, the baseline accuracy was set to that achieved by Horton and Nakai's original study, 81%.

Two models were developed: 1) cross-validated K nearest neighbours (CV KNN); and, 2) random forest.

In their first attempt to predict, both models generated an error. The error resulted from the test data set having new levels in certain factors or predictor variables that were not present in the train data set. This error was resolved by assigning "NA" to the new levels in the test data set.

After "NA" was assigned to the new levels in the test data set that were not found in the train data set, the two models proceeded to predict. However, the accuracies of their predictions were very poor - 0.37 (CV KNN) and 0.4 (Random Forest).

An inspection of their predictions showed that both models generated only 47 predictions. But there are 70 observations in the test data set. This means that 23 predictions are missing.

A closer review of the test data set show that there are 23 rows that have one or more "NA" values that were set earlier to resolve the error and allow both models to proceed with their predictions.

Apparently, both models ignored these 23 rows and made only 47 predictions using those 47 test data observations that did not have "NA" for any factor or predictor variable.

But because the accuracy calculation considers the mean of predictions matching the test data set, both models lost 23 matches due to having null predictions for these observations. This explains why the accuracies of both models were very low.

The project proceeded to remove the 23 NA rows from the test data set and applied the two models. This intended to level the comparison: 47 predictions vs. 47 test data set observations. The new accuracies of both models improved significantly relative to their initial values: 0.55 (CV KNN) and 0.51 (Random Forest). But still, these accuracies are below the baseline accuracy of 0.81.

Limitations

The first limitation is the inability of both CV KNN and Random Forest to deal with new levels of predictor variables in the test data set. What the project did to repair the test data set, i.e., using NA then removing NA rows, is not a sustainable practice. The test data set represents the future unknown and being able to tamper with the future unknown just to make use of a technique is an unrealistic expectation. Further, predictions will likely lose credibility.

The second limitation is the use of histograms to reflect the behaviour of a single predictor with the predicted locsite. It is only two-dimensional and is not able to reflect the dynamism of the multivariate relationships amongst the different predictors as they together interact to predict the locsite.

Recommendations for Future Work

1. Structured Data

Future work can define ways to capture ecoli data in a structured way to avoid surprise "new levels" in the test data set. A suggested way of structuring predictor scores is to use something like a Likert Scale system with a standardised odd number of possible values, e.g. 1, 2, 3, 4, 5. Each value can represent a score range, e.g. 1 ~ 0.0 <= score <= 0.1; 2 ~ 0.1 < score <= 0.2; 3 ~ 0.2 < score <=0.3, etc. Alternatively, each value can represent a score judgment, e.g. 1 ~ very low; 2 ~ low; 3 ~ medium; 4 ~ high; 5 ~ very high. If only levels

1 - 5 are allowed for the scores in both train and test data sets. then there will be no need to "NA" any surprise new level and CV KNN and Random Forest can readily proceed to prediction without any error.

2. Multidimensional Visualisation: Scatterplot

Instead of two-dimensional histogram, multidimensional scatterplot can be used. However, with seven predictors, the visual might look cluttered and be difficult to interpret.

3. More Data

It may help improve the accuracy of the CV KNN and Random Forest if there had been more observations available.

REFERENCES

Horton, Paul and Nakai, Kenta. A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins, Retrieved June 24, 2022, from https://www.aaai.org/Papers/ISMB/1996/ISMB96-012.pdf.

IBM (2021, December 17). Usage of KNN in IBM Integrated Analytics System. Retrieved June 29, 2022, from https://www.ibm.com/docs/en/ias?topic=knn-usage.

Irizarry, R. A. (2022, January 13). Data Analysis and Prediction Algorithms with R in Introduction to Data Science. Retrieved April 06, 2022, from https://rafalab.github.io/dsbook/introduction-to-machine-learning.html.

Kho, Julia (October 20, 2018). Why Random Forest is My Favorite Machine Learning Model. Retrieved June 30, 2022, from https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706#:~:text=Random%20forests%20is%20great%20with,working%20with%20subsets%20of%20data.&text=It%20i

Kumar, Aditya (May 25, 2020). KNN Algorithm: When? Why? How?. Retrieved June 29, 2022, from https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f.

Nakai, Kenta and Horton, Paul. UCI Machine Learning Repository: Ecoli Data Set, Retrieved June 14, 2022, from https://archive.ics.uci.edu/ml/datasets/Ecoli.

Statistics Globe. R Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels):factor X has new levels Y, Retrieved July 3, 2022, from https://statisticsglobe.com/r-error-model-frame-default-factor-x-has-new-levels.