MSDS696 – Data Science Practicum II - Ecoli

Eric Mwangi

7/20/2020

Load all the required libraries:

Load the libraries after installing the packages. The following commands will install these packages if they are not already installed:

Reference: Horton, P., & Nakai, K. (1996). A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins. ISMB-96 Proceedings, 109 – 115. Retrieved from https://www.aaai.org/Papers/ISMB/1996/ISMB96-012.pdf

##Getting the data.

Read the data (ecoli.data) from the local directory.

Create a data frame from the downloaded file. Attach the column header titles.

```
# Load the specified data set, here "ecoli".
ecoli_df <- as_tibble(read.table("D:/Regis Docs/MSDS696 - Data Science Practicum II/Data/ecoli.data", h
    col.names = c("seqn", "mcg", "gvh", "lip", "chg", "aac", "alm1", "alm2", "cld")))
```

Look at the data (structure of the imported data):

Display the first few rows of the data frame.

```
str(ecoli_df)
```

```
## tibble [336 x 9] (S3: tbl_df/tbl/data.frame)
##  $ seqn: chr [1:336] "AAT_ECOLI" "ACEA_ECOLI" "ACEK_ECOLI" "ACKA_ECOLI" ...
##  $ mcg : num [1:336] 0.49 0.07 0.56 0.59 0.23 0.67 0.29 0.21 0.2 0.42 ...
##  $ gvh : num [1:336] 0.29 0.4 0.4 0.49 0.32 0.39 0.28 0.34 0.44 0.4 ...
##  $ lip : num [1:336] 0.48 0.48 0.48 0.48 0.48 0.48 0.48 0.48 0.48 0.48 ...
##  $ chg : num [1:336] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ aac : num [1:336] 0.56 0.54 0.49 0.52 0.55 0.36 0.44 0.51 0.46 0.56 ...
##  $ alm1: num [1:336] 0.24 0.35 0.37 0.45 0.25 0.38 0.23 0.28 0.51 0.18 ...
##  $ alm2: num [1:336] 0.35 0.44 0.46 0.36 0.35 0.46 0.34 0.39 0.57 0.3 ...
##  $ cld : chr [1:336] "cp" "cp" "cp" "cp" ...
```

```
glimpse(ecoli_df)
```

```
## Rows: 336
## Columns: 9
## $ seqn <chr> "AAT_ECOLI", "ACEA_ECOLI", "ACEK_ECOLI", "ACKA_ECOLI", "ADI_EC...
## $ mcg  <dbl> 0.49, 0.07, 0.56, 0.59, 0.23, 0.67, 0.29, 0.21, 0.20, 0.42, 0....
```

```
## $ gvh  <dbl> 0.29, 0.40, 0.40, 0.49, 0.32, 0.39, 0.28, 0.34, 0.44, 0.40, 0....
## $ lip  <dbl> 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0....
## $ chg  <dbl> 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0....
## $ aac  <dbl> 0.56, 0.54, 0.49, 0.52, 0.55, 0.36, 0.44, 0.51, 0.46, 0.56, 0....
## $ alm1 <dbl> 0.24, 0.35, 0.37, 0.45, 0.25, 0.38, 0.23, 0.28, 0.51, 0.18, 0....
## $ alm2 <dbl> 0.35, 0.44, 0.46, 0.36, 0.35, 0.46, 0.34, 0.39, 0.57, 0.30, 0....
## $ cld  <chr> "cp", "cp", "cp", "cp", "cp", "cp", "cp", "cp", "cp", "cp", "c...
```

Summary of the data.

```
summary(ecoli_df)
```

```
##      seqn                mcg              gvh              lip        
##  Length:336         Min.   :0.0000   Min.   :0.16   Min.   :0.4800  
##  Class :character   1st Qu.:0.3400   1st Qu.:0.40   1st Qu.:0.4800  
##  Mode  :character   Median :0.5000   Median :0.47   Median :0.4800  
##                     Mean   :0.5001   Mean   :0.50   Mean   :0.4955  
##                     3rd Qu.:0.6625   3rd Qu.:0.57   3rd Qu.:0.4800  
##                     Max.   :0.8900   Max.   :1.00   Max.   :1.0000  
##       chg              aac              alm1             alm2        
##  Min.   :0.5000   Min.   :0.000   Min.   :0.0300   Min.   :0.0000  
##  1st Qu.:0.5000   1st Qu.:0.420   1st Qu.:0.3300   1st Qu.:0.3500  
##  Median :0.5000   Median :0.495   Median :0.4550   Median :0.4300  
##  Mean   :0.5015   Mean   :0.500   Mean   :0.5002   Mean   :0.4997  
##  3rd Qu.:0.5000   3rd Qu.:0.570   3rd Qu.:0.7100   3rd Qu.:0.7100  
##  Max.   :1.0000   Max.   :0.880   Max.   :1.0000   Max.   :0.9900  
##      cld           
##  Length:336         
##  Class :character   
##  Mode  :character   
##                     
##                     
##
```

```
data_prof <- funModeling::profiling_num(ecoli_df)

data_prof
```

```
##   variable      mean    std_dev variation_coef   p_01   p_05 p_25  p_50   p_75
## 1      mcg 0.5000595 0.19463398     0.38922162 0.0635 0.2075 0.34 0.500 0.6625
## 2      gvh 0.5000000 0.14815684     0.29631367 0.2600 0.2975 0.40 0.470 0.5700
## 3      lip 0.4954762 0.08849528     0.17860652 0.4800 0.4800 0.48 0.480 0.4800
## 4      chg 0.5014881 0.02727724     0.05439259 0.5000 0.5000 0.50 0.500 0.5000
## 5      aac 0.5000298 0.12237573     0.24473690 0.1875 0.3300 0.42 0.495 0.5700
## 6     alm1 0.5001786 0.21575130     0.43134855 0.1205 0.2075 0.33 0.455 0.7100
## 7     alm2 0.4997321 0.20941052     0.41904552 0.1535 0.2250 0.35 0.430 0.7100
##     p_95  p_99    skewness   kurtosis    iqr        range_98      range_80
## 1 0.7825 0.860 -0.16521389   2.135734 0.3225  [0.0635, 0.86] [0.245, 0.74]
## 2 0.8100 0.860  0.77172433   3.237767 0.1700    [0.26, 0.86] [0.34, 0.735]
## 3 0.4800 1.000  5.53449861  31.630675 0.0000       [0.48, 1]  [0.48, 0.48]
## 4 0.5000 0.500 18.24836938 334.002985 0.0000      [0.5, 0.5]    [0.5, 0.5]
## 5 0.7125 0.830  0.06258569   4.297958 0.1500  [0.1875, 0.83]  [0.37, 0.65]
## 6 0.8600 0.920  0.26136274   1.952924 0.3800  [0.1205, 0.92] [0.245, 0.79]
## 7 0.8500 0.913  0.41236856   2.057885 0.3600 [0.1535, 0.913]   [0.27, 0.8]
```

```
print.data.frame(head(ecoli_df))
```

```
##         seqn  mcg  gvh  lip chg  aac alm1 alm2 cld
## 1  AAT_ECOLI 0.49 0.29 0.48 0.5 0.56 0.24 0.35  cp
## 2 ACEA_ECOLI 0.07 0.40 0.48 0.5 0.54 0.35 0.44  cp
## 3 ACEK_ECOLI 0.56 0.40 0.48 0.5 0.49 0.37 0.46  cp
## 4 ACKA_ECOLI 0.59 0.49 0.48 0.5 0.52 0.45 0.36  cp
## 5  ADI_ECOLI 0.23 0.32 0.48 0.5 0.55 0.25 0.35  cp
## 6 ALKH_ECOLI 0.67 0.39 0.48 0.5 0.36 0.38 0.46  cp
```

View a section of the data frame.

```
library(dplyr)
ecoli_df %>% slice_head(n = 5)
```

```
## # A tibble: 5 x 9
##   seqn         mcg   gvh   lip   chg   aac  alm1  alm2 cld
##   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
## 1 AAT_ECOLI   0.49 0.290  0.48   0.5  0.56  0.24  0.35 cp
## 2 ACEA_ECOLI  0.07 0.4    0.48   0.5  0.54  0.35  0.44 cp
## 3 ACEK_ECOLI  0.56 0.4    0.48   0.5  0.49  0.37  0.46 cp
## 4 ACKA_ECOLI  0.59 0.49   0.48   0.5  0.52  0.45  0.36 cp
## 5 ADI_ECOLI   0.23 0.32   0.48   0.5  0.55  0.25  0.35 cp
```

```
print.data.frame(head(ecoli_df))
```

```
##         seqn  mcg  gvh  lip chg  aac alm1 alm2 cld
## 1  AAT_ECOLI 0.49 0.29 0.48 0.5 0.56 0.24 0.35  cp
## 2 ACEA_ECOLI 0.07 0.40 0.48 0.5 0.54 0.35 0.44  cp
## 3 ACEK_ECOLI 0.56 0.40 0.48 0.5 0.49 0.37 0.46  cp
## 4 ACKA_ECOLI 0.59 0.49 0.48 0.5 0.52 0.45 0.36  cp
## 5  ADI_ECOLI 0.23 0.32 0.48 0.5 0.55 0.25 0.35  cp
## 6 ALKH_ECOLI 0.67 0.39 0.48 0.5 0.36 0.38 0.46  cp
```

```
print.data.frame(tail(ecoli_df))
```

```
##         seqn  mcg  gvh  lip chg  aac alm1 alm2 cld
## 1 TORA_ECOLI 0.43 0.59 0.48 0.5 0.52 0.49 0.56  pp
## 2 TREA_ECOLI 0.74 0.56 0.48 0.5 0.47 0.68 0.30  pp
## 3 UGPB_ECOLI 0.71 0.57 0.48 0.5 0.48 0.35 0.32  pp
## 4 USHA_ECOLI 0.61 0.60 0.48 0.5 0.44 0.39 0.38  pp
## 5 XYLF_ECOLI 0.59 0.61 0.48 0.5 0.42 0.42 0.37  pp
## 6 YTFQ_ECOLI 0.74 0.74 0.48 0.5 0.31 0.53 0.52  pp
```

```
library(dplyr)
ecoli_df %>% slice_tail(n = 5)
```

```
## # A tibble: 5 x 9
##   seqn         mcg   gvh   lip   chg   aac  alm1  alm2 cld
##   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
```

```
## 1 TREA_ECOLI  0.74 0.56   0.48   0.5  0.47  0.68  0.3  pp
## 2 UGPB_ECOLI  0.71 0.570  0.48   0.5  0.48  0.35  0.32 pp
## 3 USHA_ECOLI  0.61 0.6    0.48   0.5  0.44  0.39  0.38 pp
## 4 XYLF_ECOLI  0.59 0.61   0.48   0.5  0.42  0.42  0.37 pp
## 5 YTFQ_ECOLI  0.74 0.74   0.48   0.5  0.31  0.53  0.52 pp
```

```
view(ecoli_df)
```

Dimensions of the data frame.

```
dim(ecoli_df)
```

```
## [1] 336   9
```

Length of the data frame.

```
length(ecoli_df)
```

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

```
nrow(ecoli_df)
```

```
## [1] 336
```

```
ncol(ecoli_df)
```

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

```
lapply(ecoli_df, summary)
```

```
## $seqn
##    Length     Class      Mode
##       336 character character
##
## $mcg
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3400  0.5000  0.5001  0.6625  0.8900
##
## $gvh
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.16    0.40    0.47    0.50    0.57    1.00
##
## $lip
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4800  0.4800  0.4800  0.4955  0.4800  1.0000
##
## $chg
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5000  0.5000  0.5000  0.5015  0.5000  1.0000
##
```

```
## $aac
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.420   0.495   0.500   0.570   0.880
##
## $alm1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0300  0.3300  0.4550  0.5002  0.7100  1.0000
##
## $alm2
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.3500  0.4300  0.4997  0.7100  0.9900
##
## $cld
##    Length     Class      Mode
##       336 character character
```

```
describe(ecoli_df)
```

```
## Warning in describe(ecoli_df): NAs introduced by coercion

## Warning in describe(ecoli_df): NAs introduced by coercion

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf

## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf
```

```
##       vars   n mean   sd median trimmed  mad  min  max range  skew kurtosis
## seqn*    1 336  NaN   NA     NA     NaN   NA  Inf -Inf  -Inf    NA       NA
## mcg      2 336  0.5 0.19   0.50    0.50 0.24 0.00 0.89  0.89 -0.16    -0.88
## gvh      3 336  0.5 0.15   0.47    0.49 0.12 0.16 1.00  0.84  0.77     0.22
## lip      4 336  0.5 0.09   0.48    0.48 0.00 0.48 1.00  0.52  5.51    28.44
## chg      5 336  0.5 0.03   0.50    0.50 0.00 0.50 1.00  0.50 18.17   329.02
## aac      6 336  0.5 0.12   0.50    0.50 0.11 0.00 0.88  0.88  0.06     1.27
## alm1     7 336  0.5 0.22   0.46    0.49 0.25 0.03 1.00  0.97  0.26    -1.06
## alm2     8 336  0.5 0.21   0.43    0.49 0.19 0.00 0.99  0.99  0.41    -0.95
## cld*     9 336  NaN   NA     NA     NaN   NA  Inf -Inf  -Inf    NA       NA
##         se
## seqn*   NA
## mcg   0.01
## gvh   0.01
## lip   0.00
## chg   0.00
## aac   0.01
## alm1  0.01
## alm2  0.01
## cld*    NA
```

```
status(ecoli_df)
```

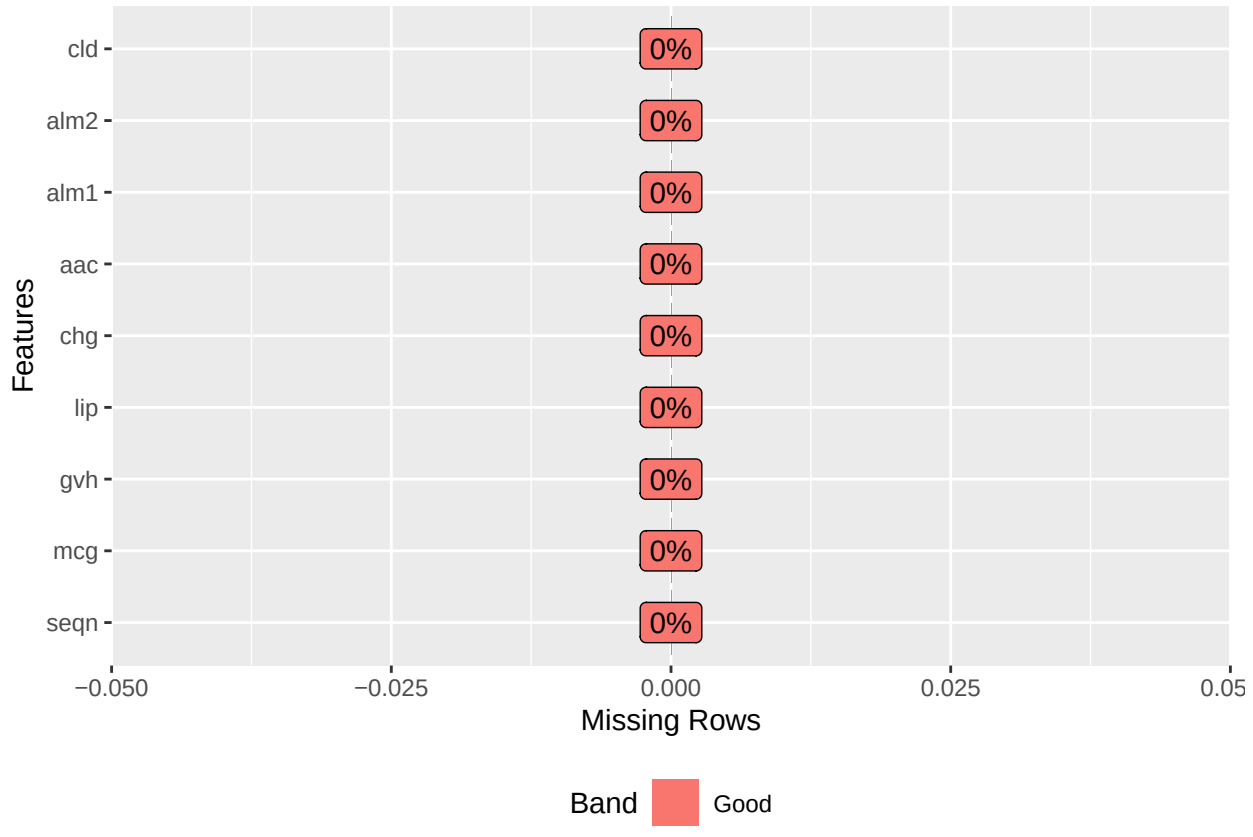
```
##   variable q_zeros     p_zeros q_na p_na q_inf p_inf      type unique
## 1     seqn       0 0.000000000    0    0     0     0 character    336
## 2      mcg       2 0.005952381    0    0     0     0   numeric     78
## 3      gvh       0 0.000000000    0    0     0     0   numeric     63
## 4      lip       0 0.000000000    0    0     0     0   numeric      2
## 5      chg       0 0.000000000    0    0     0     0   numeric      2
## 6      aac       1 0.002976190    0    0     0     0   numeric     59
## 7     alm1       0 0.000000000    0    0     0     0   numeric     82
## 8     alm2       1 0.002976190    0    0     0     0   numeric     77
## 9      cld       0 0.000000000    0    0     0     0 character      8
```

Frequencies of the data Frame.

```
#freq(ecoli_df)
```

Missing attribute values.

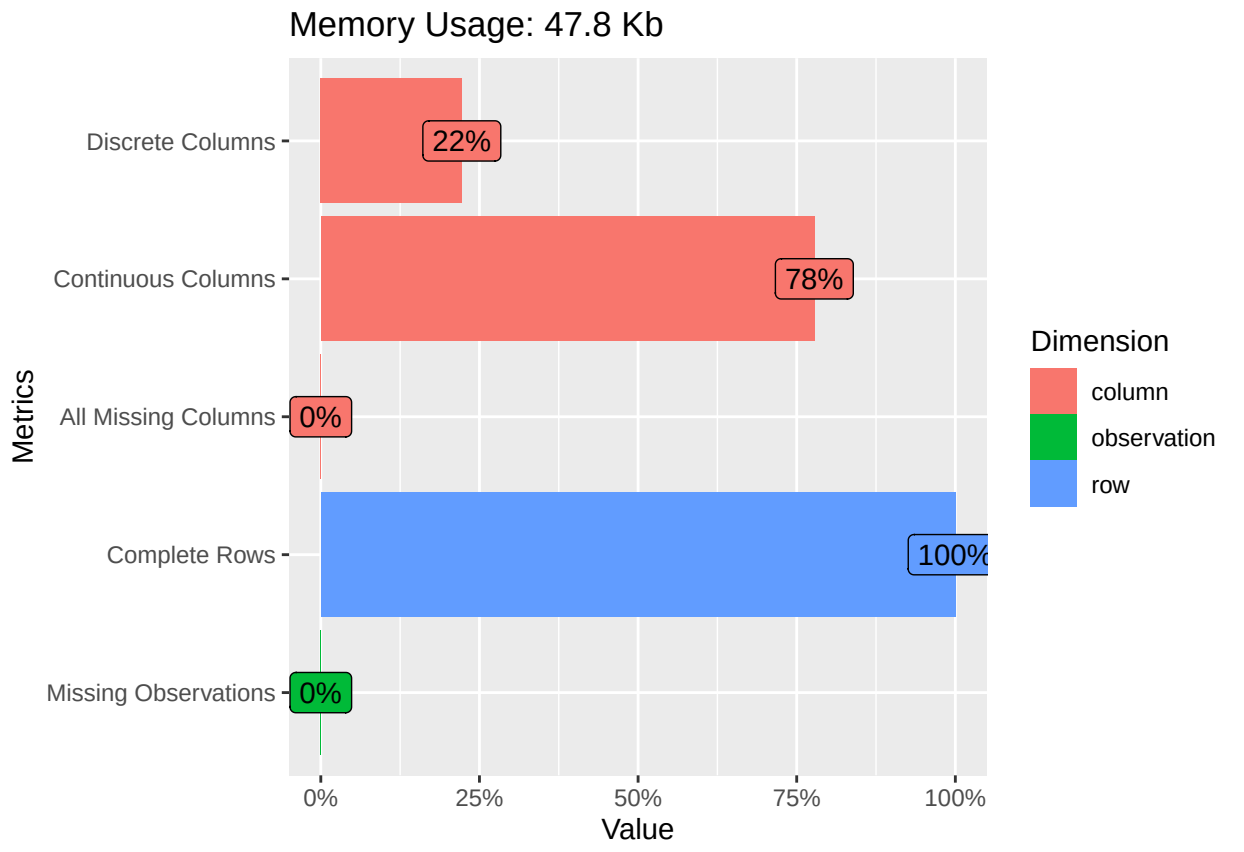
```
plot_missing(ecoli_df)
```



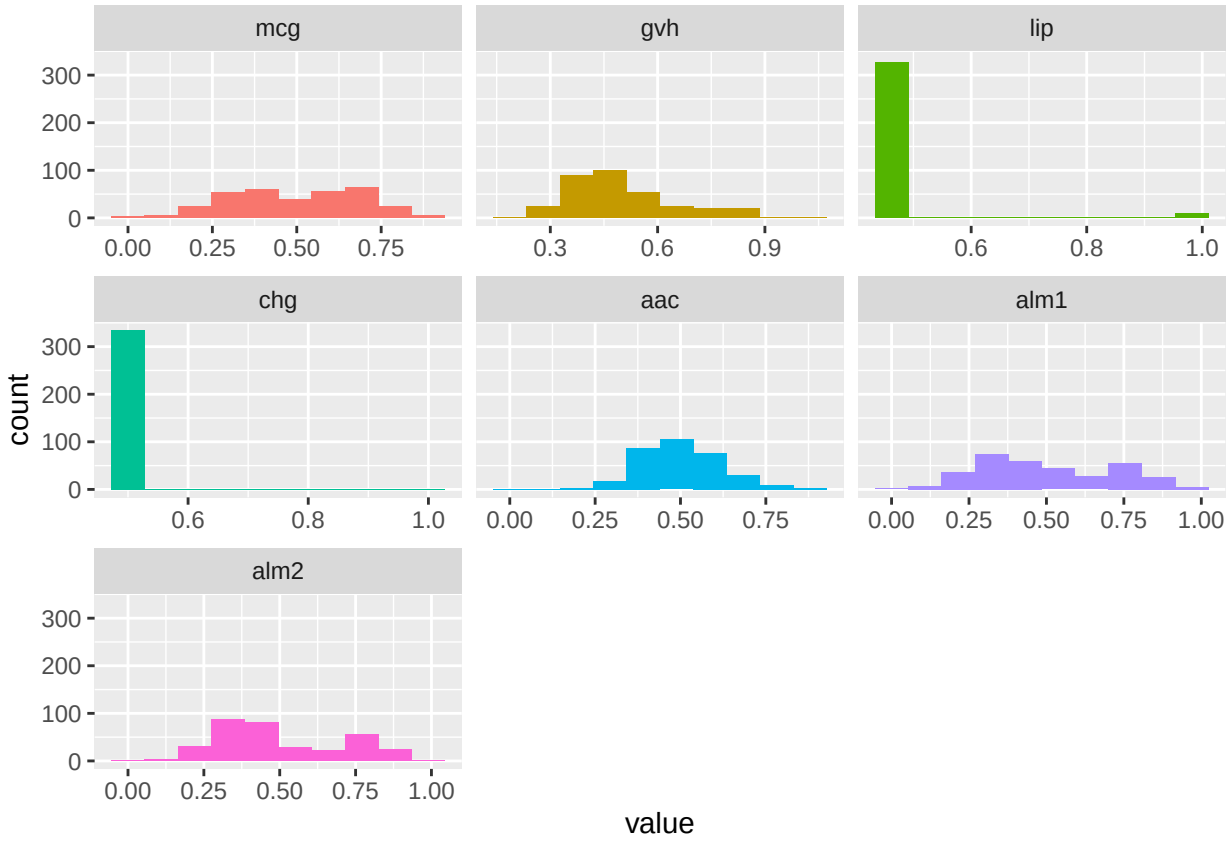

```
introduce(ecoli_df)
```

```
## # A tibble: 1 x 9
##    rows columns discrete_columns continuous_colu~ all_missing_col~
##   <int>   <int>            <int>            <int>            <int>
## 1   336       9                2                7                0
## # ... with 4 more variables: total_missing_values <int>, complete_rows <int>,
## #   total_observations <int>, memory_usage <dbl>
```

```
plot_intro(ecoli_df)
```

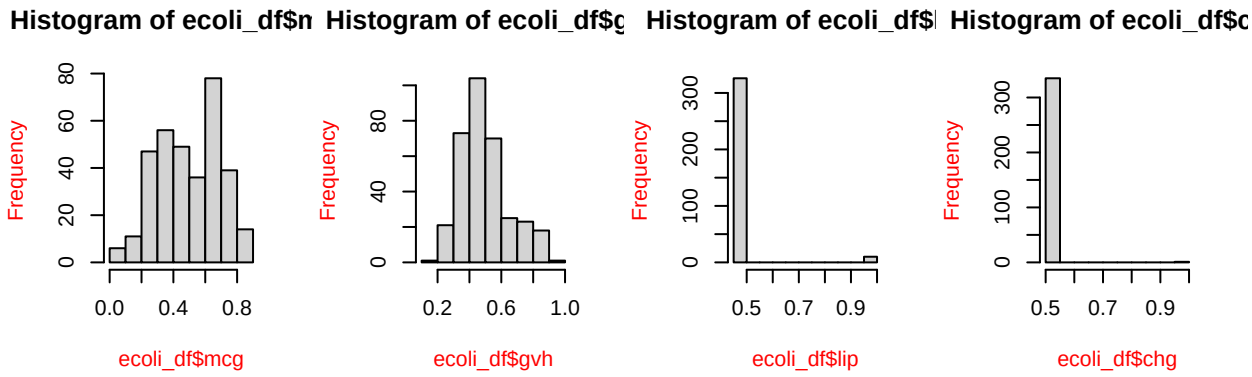


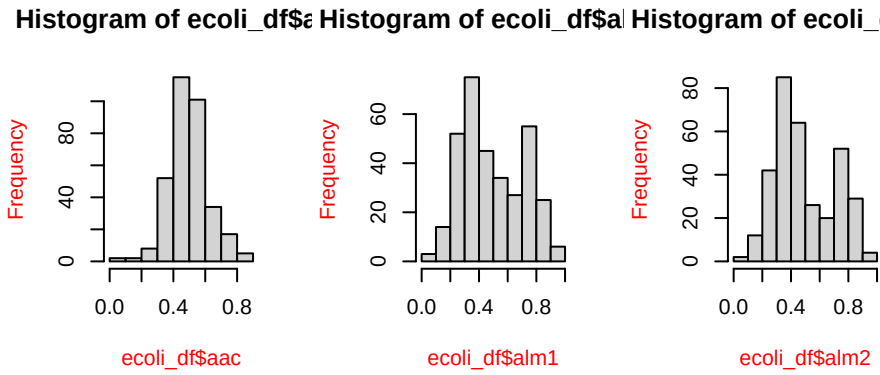

```
plot_num(ecoli_df)
```



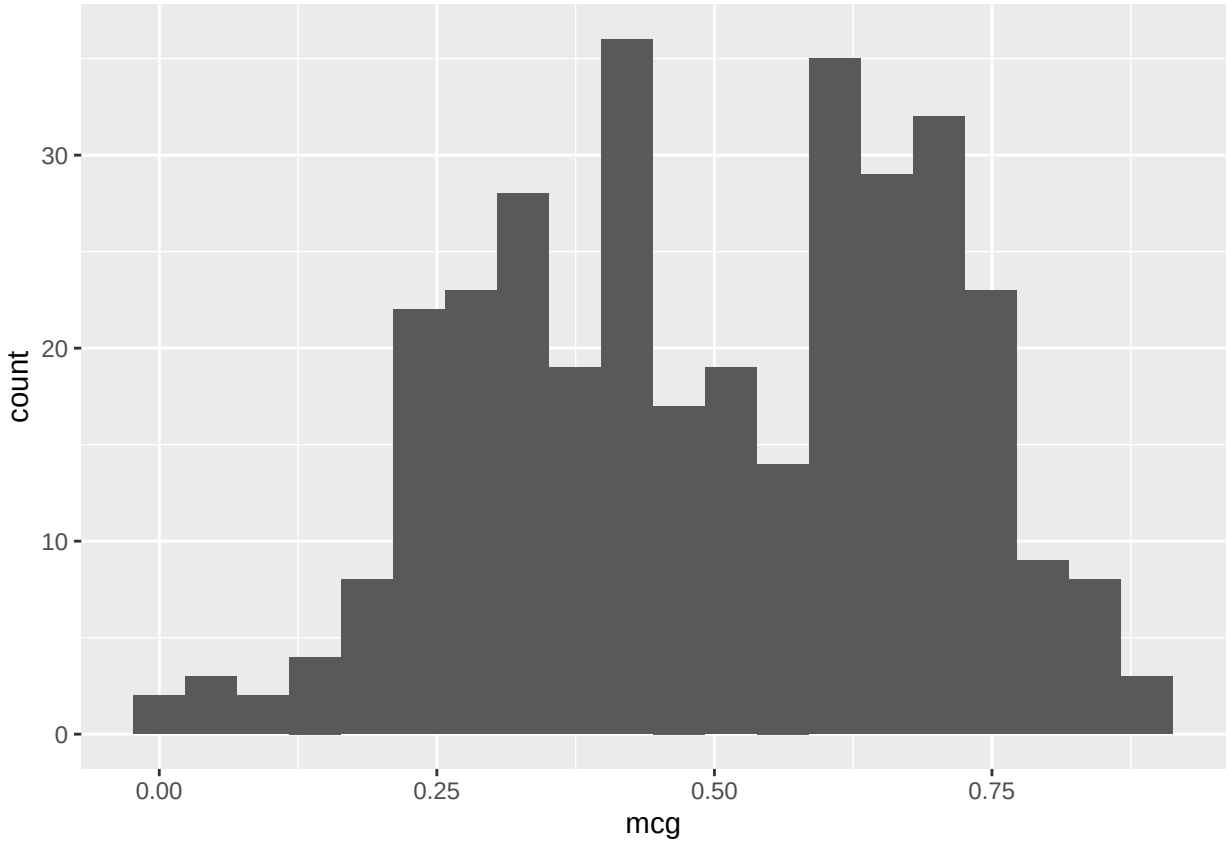
Histogram of the variables:

```
# Set a graphical parameter within the plotting function
par(mfrow=c(2, 4))  # divide graph area in 2 rows and 4 columns
hist(ecoli_df$mcg, col.lab="red")
hist(ecoli_df$gvh, col.lab="red")
hist(ecoli_df$lip, col.lab="red")
hist(ecoli_df$chg, col.lab="red")
hist(ecoli_df$aac, col.lab="red")
hist(ecoli_df$alm1, col.lab="red")
hist(ecoli_df$alm2, col.lab="red")
```

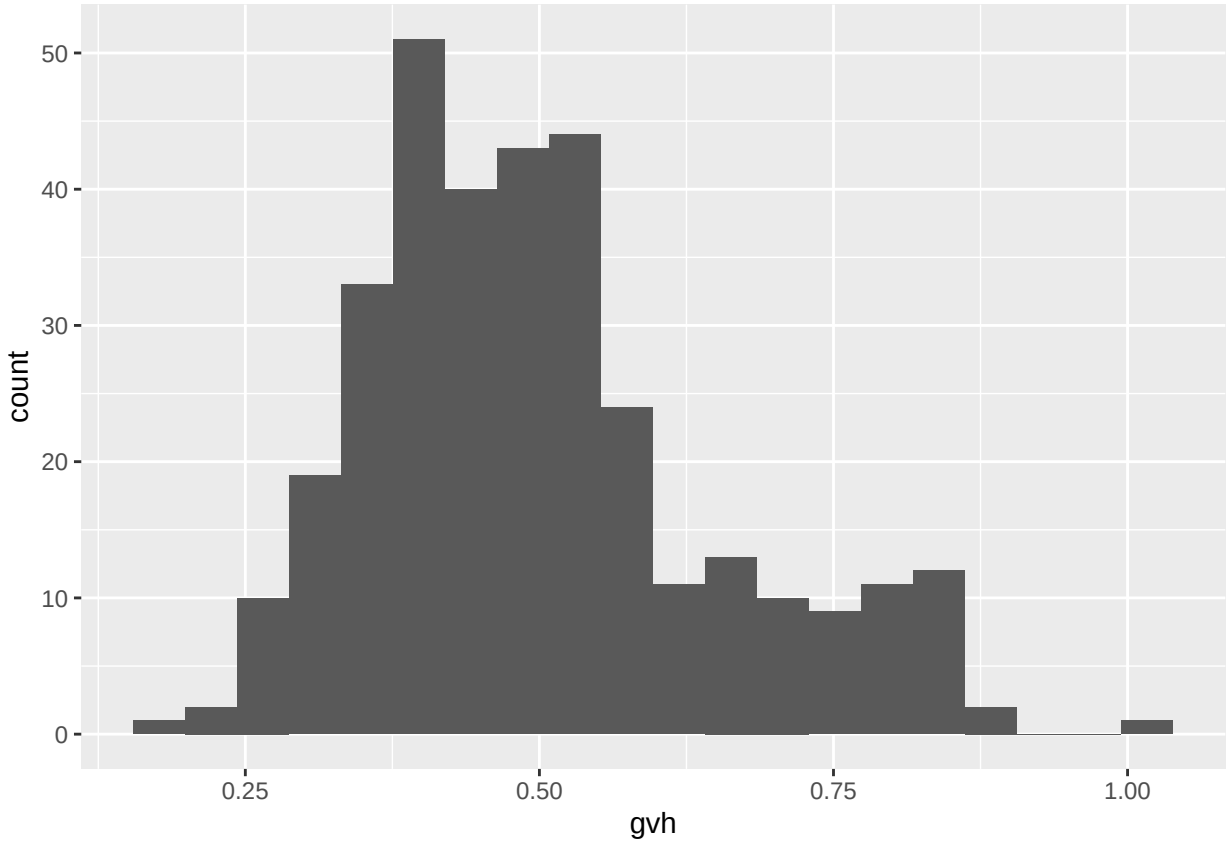




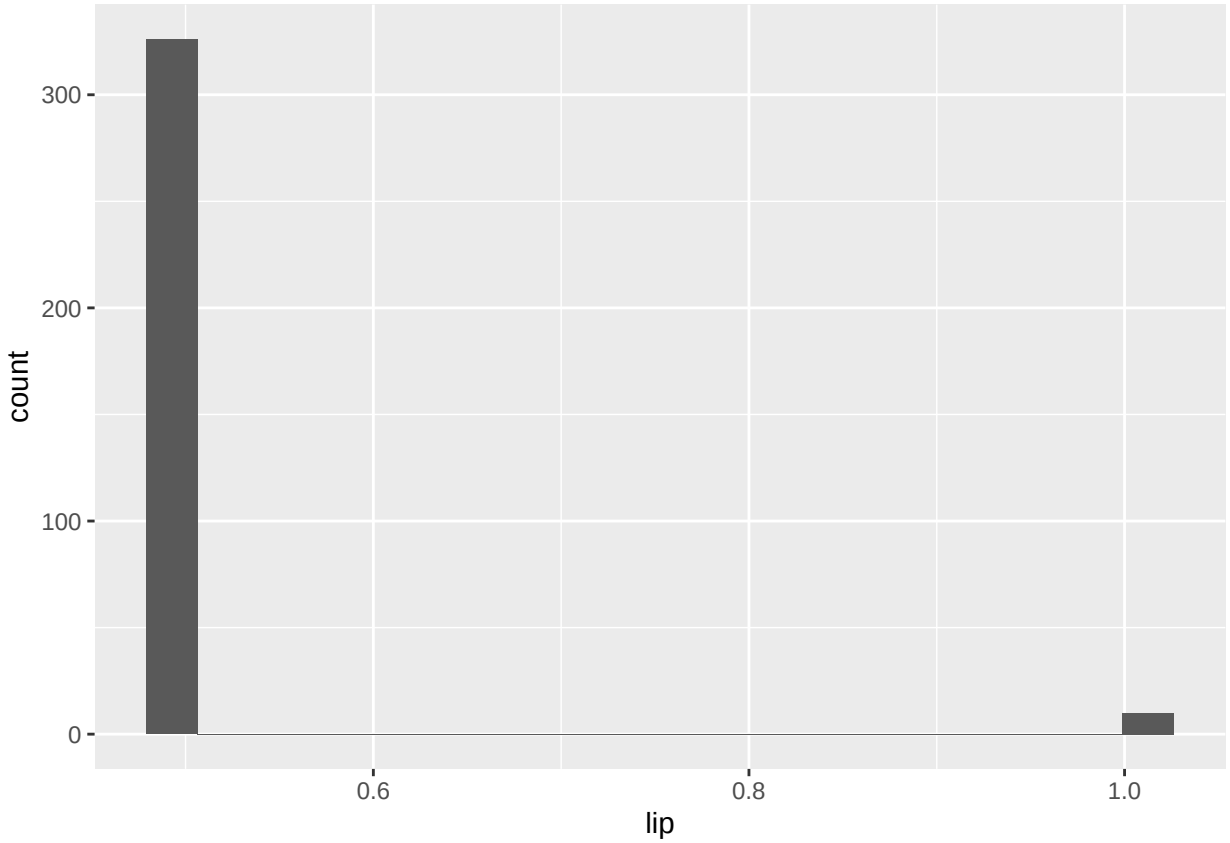

```
ecoli_df %>%
  ggplot(aes(x=mcg)) +
         geom_histogram(bins=20)
```



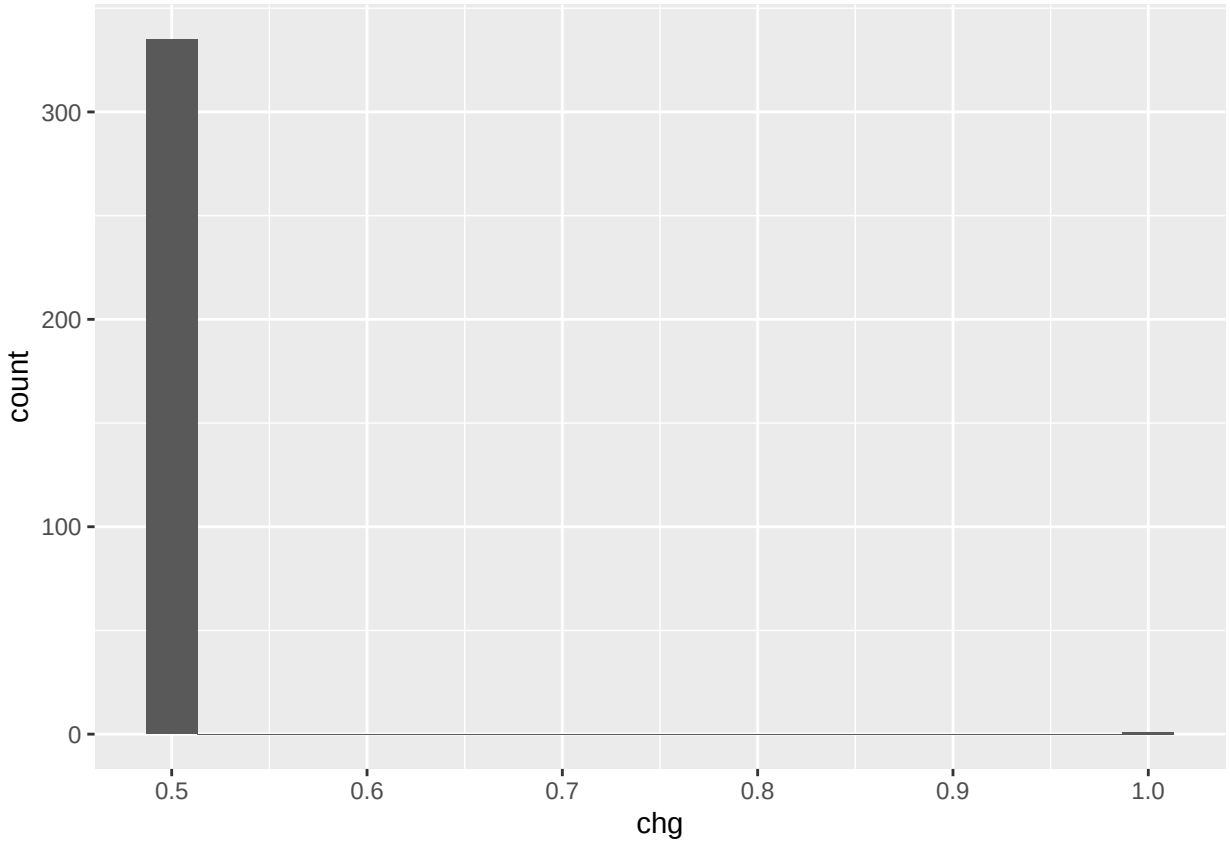

```
ecoli_df %>%
  ggplot(aes(x=gvh)) +
      geom_histogram(bins=20)
```



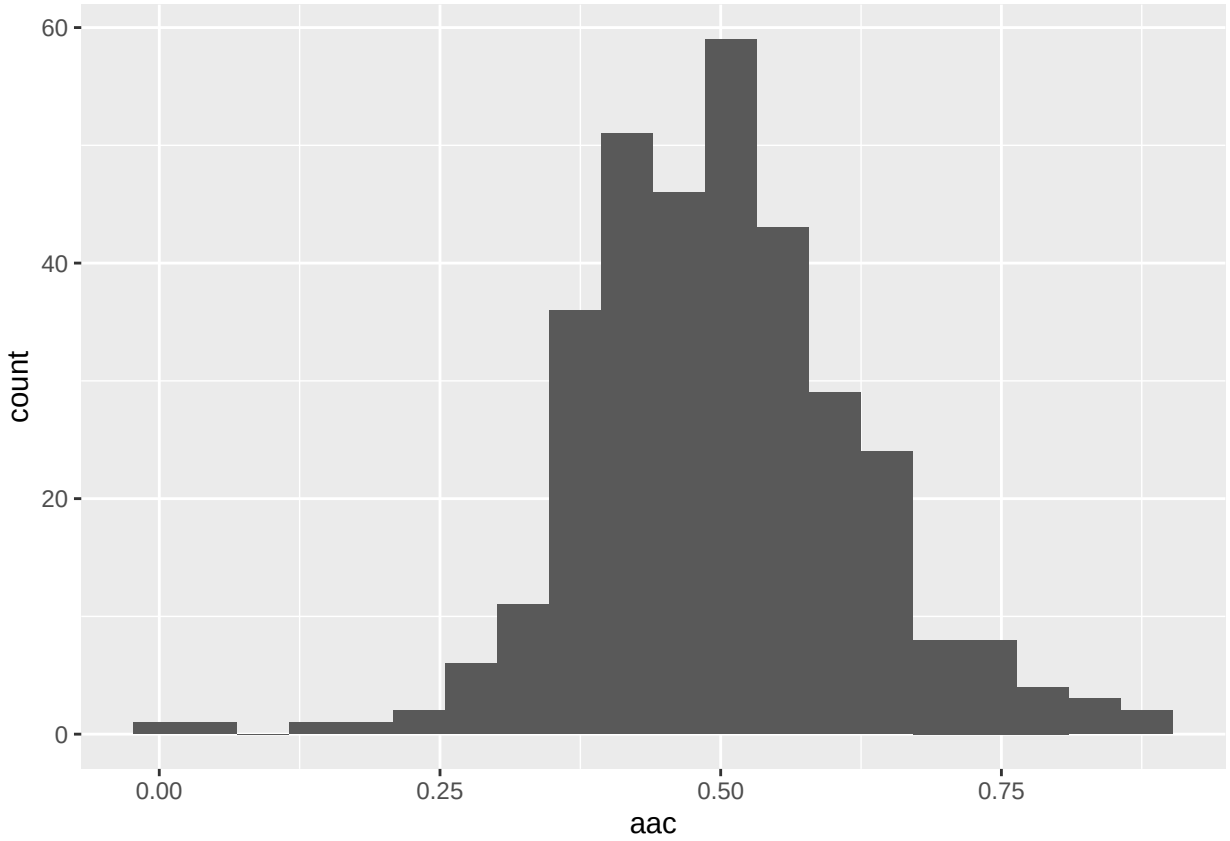

```
ecoli_df %>%
  ggplot(aes(x=lip)) +
      geom_histogram(bins=20)
```



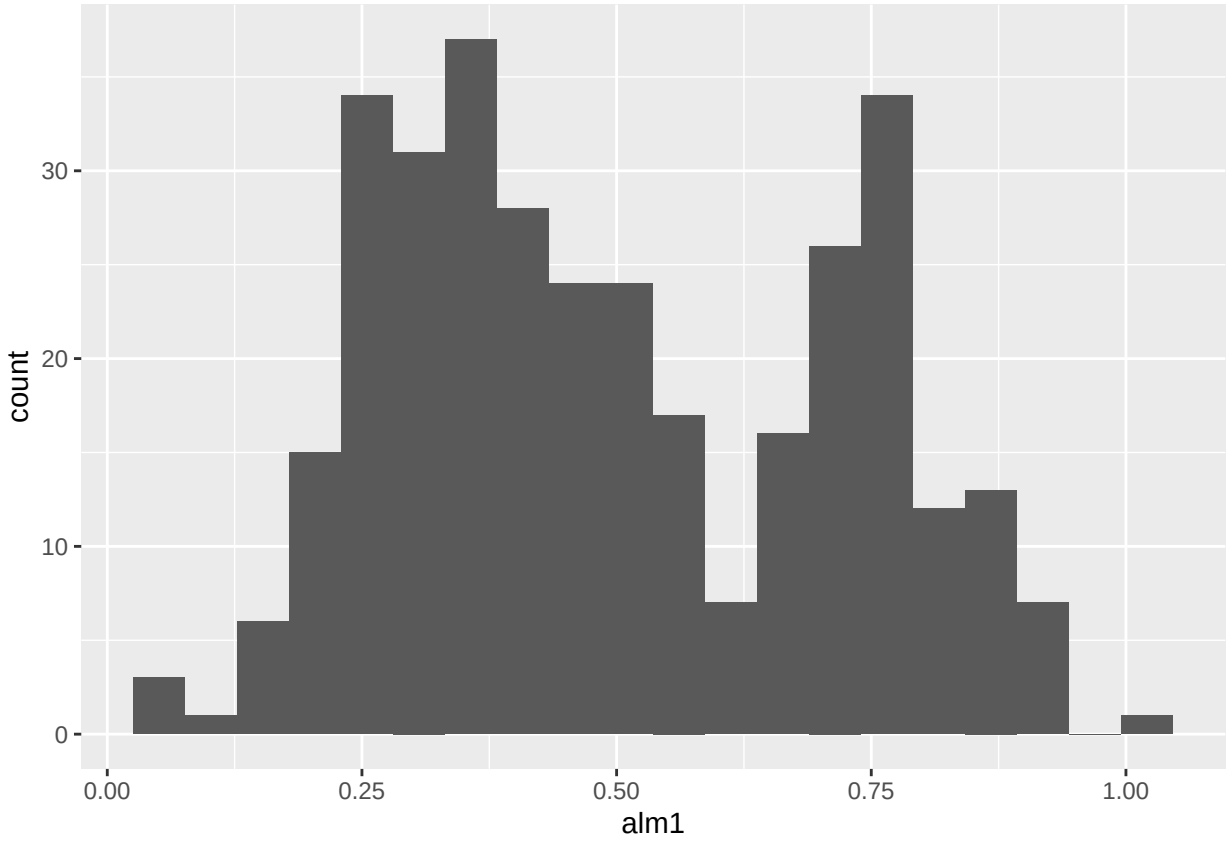

```
ecoli_df %>%
  ggplot(aes(x=chg)) +
      geom_histogram(bins=20)
```



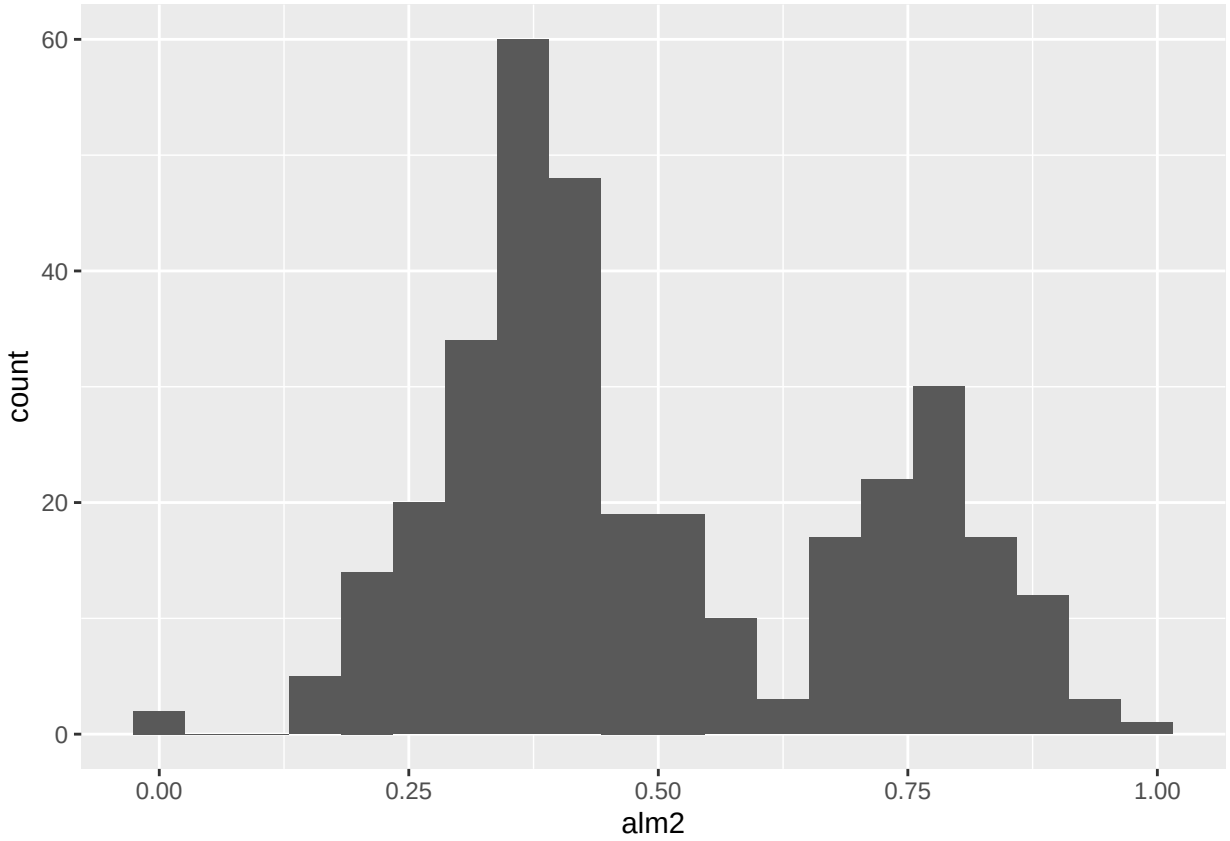

```
ecoli_df %>%
  ggplot(aes(x=aac)) +
      geom_histogram(bins=20)
```



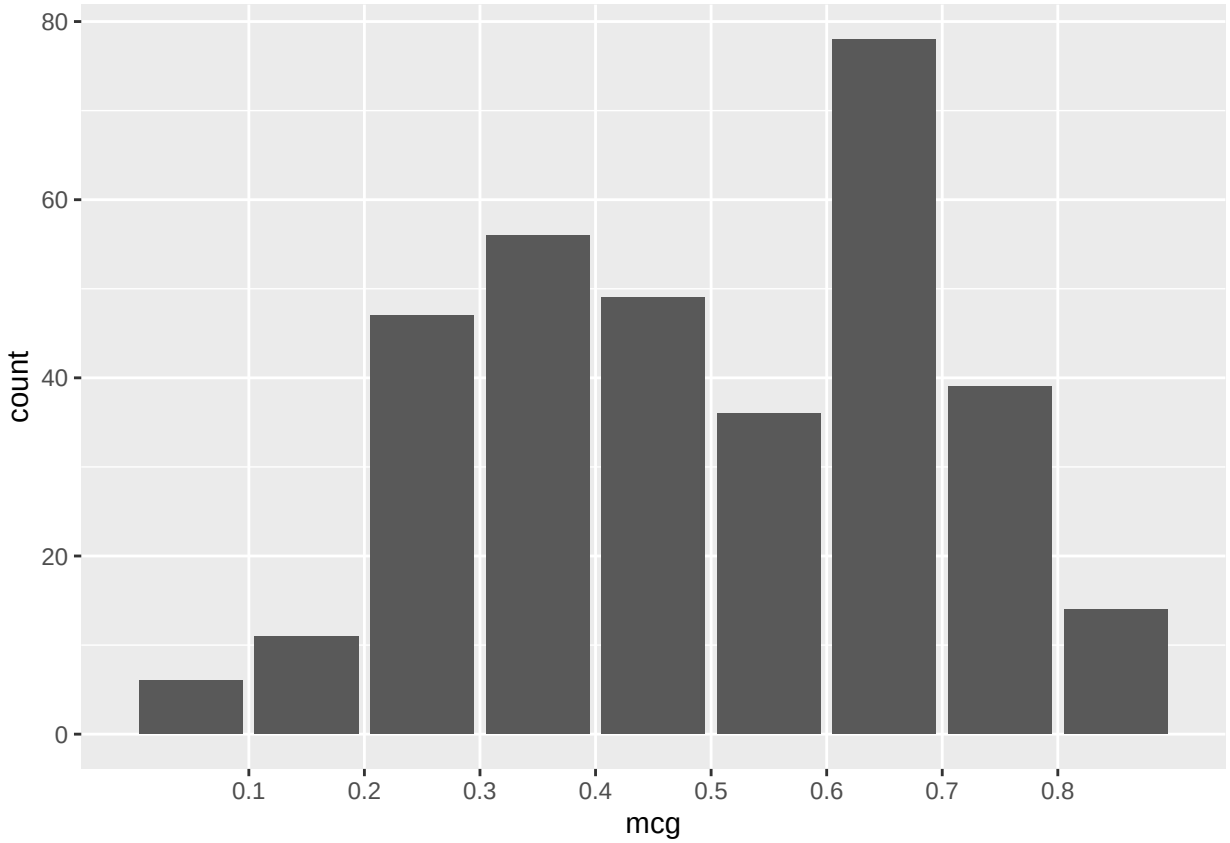

```
ecoli_df %>%
  ggplot(aes(x=alm1)) +
      geom_histogram(bins=20)
```



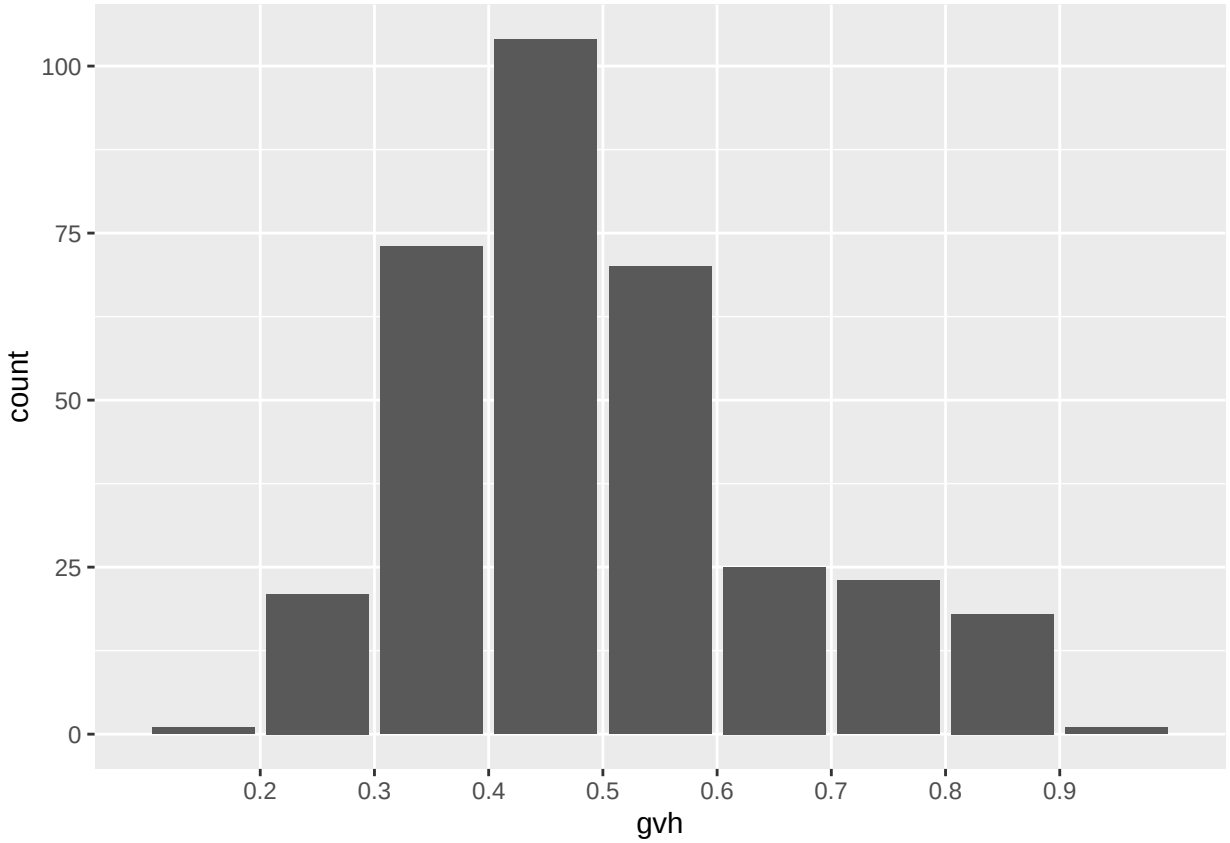

```
ecoli_df %>%
  ggplot(aes(x=alm2)) +
      geom_histogram(bins=20)
```



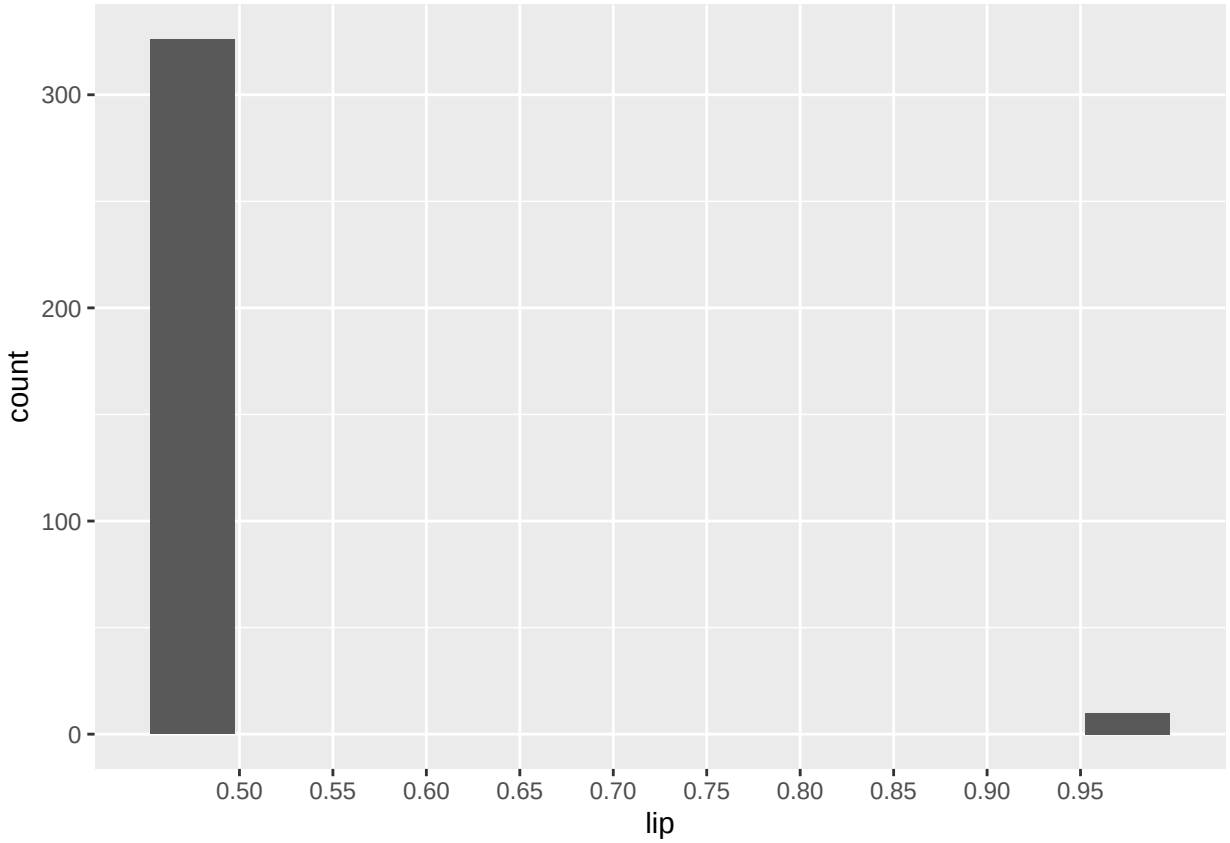

```
ecoli_df %>%
  ggplot(aes(x=mcg)) +
         geom_bar() +
         scale_x_binned()
```



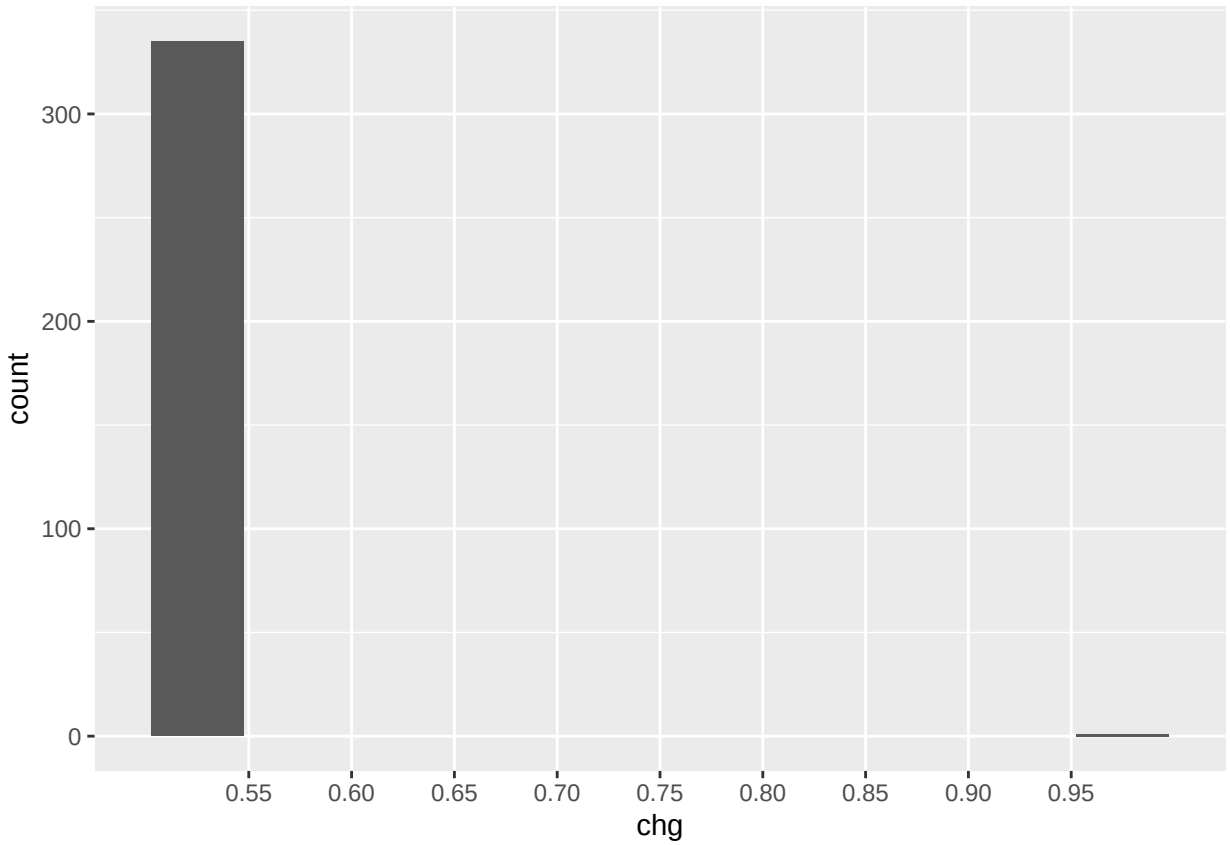

```
ecoli_df %>%
  ggplot(aes(x=gvh)) +
         geom_bar() +
         scale_x_binned()
```



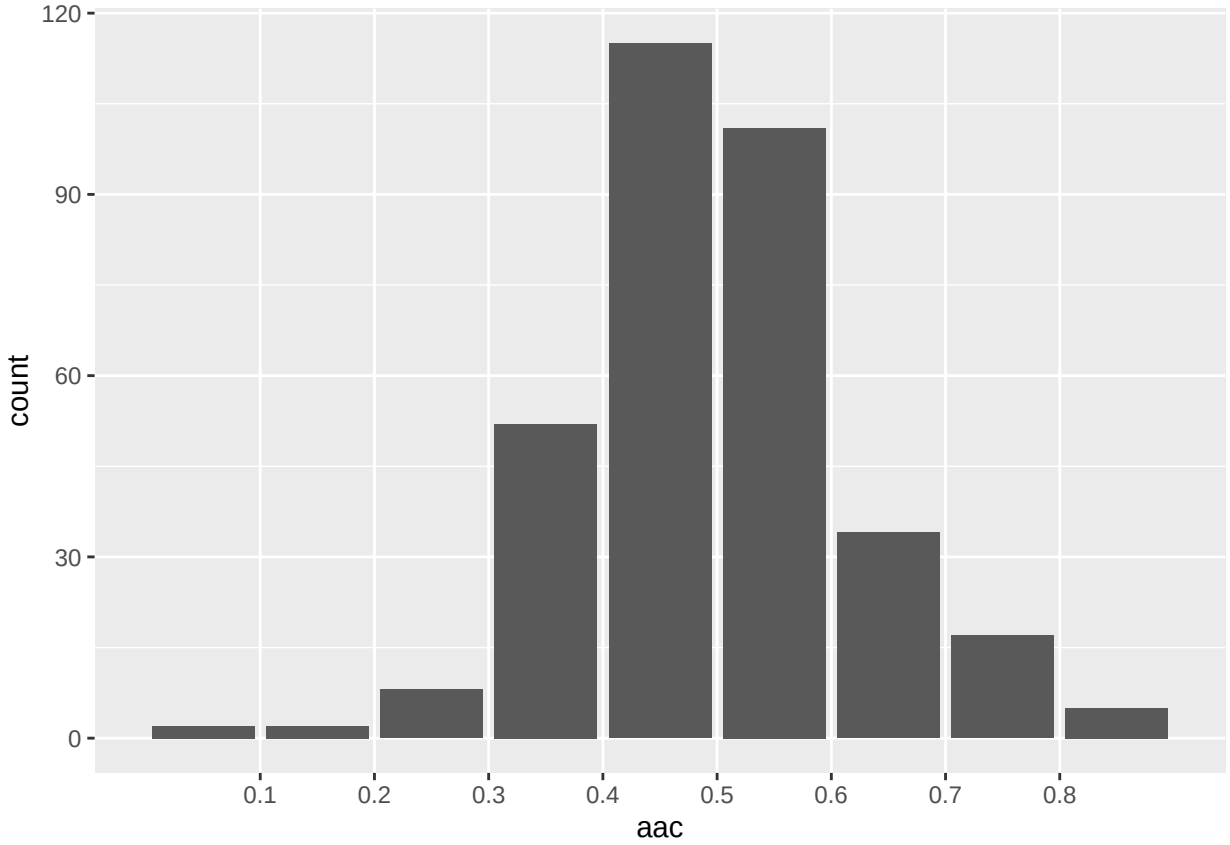

```
ecoli_df %>%
  ggplot(aes(x=lip)) +
         geom_bar() +
         scale_x_binned()
```



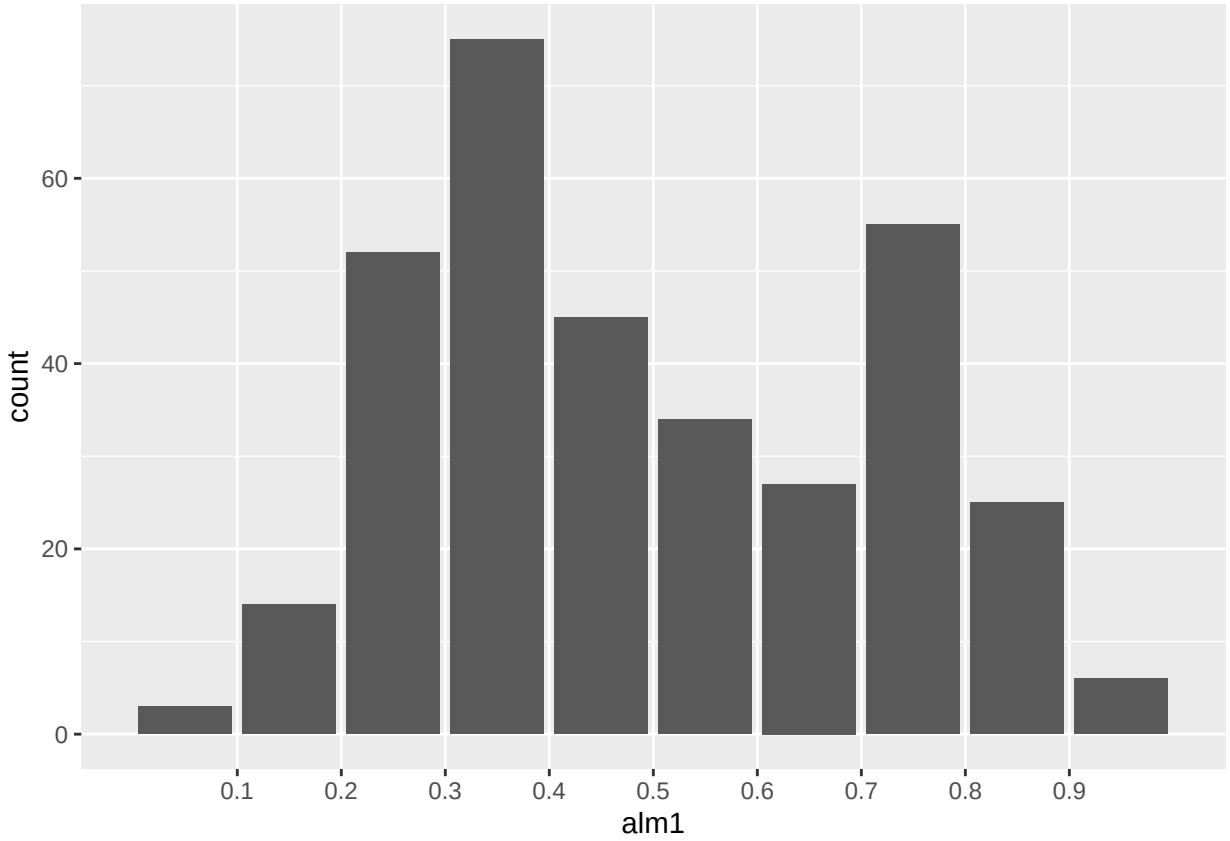

```
ecoli_df %>%
  ggplot(aes(x=chg)) +
         geom_bar() +
         scale_x_binned()
```



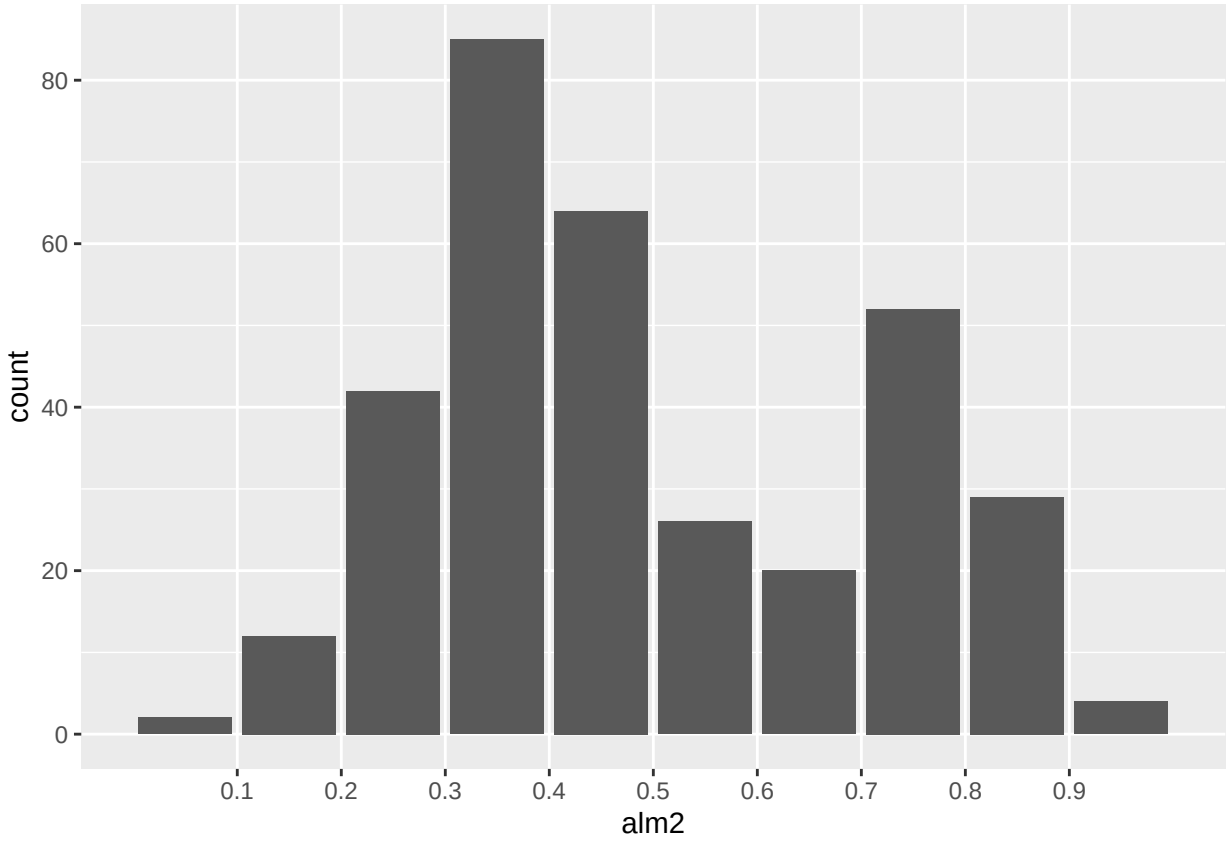

```
ecoli_df %>%
  ggplot(aes(x=aac)) +
         geom_bar() +
         scale_x_binned()
```




```
ecoli_df %>%
  ggplot(aes(x=alm1)) +
         geom_bar() +
         scale_x_binned()
```



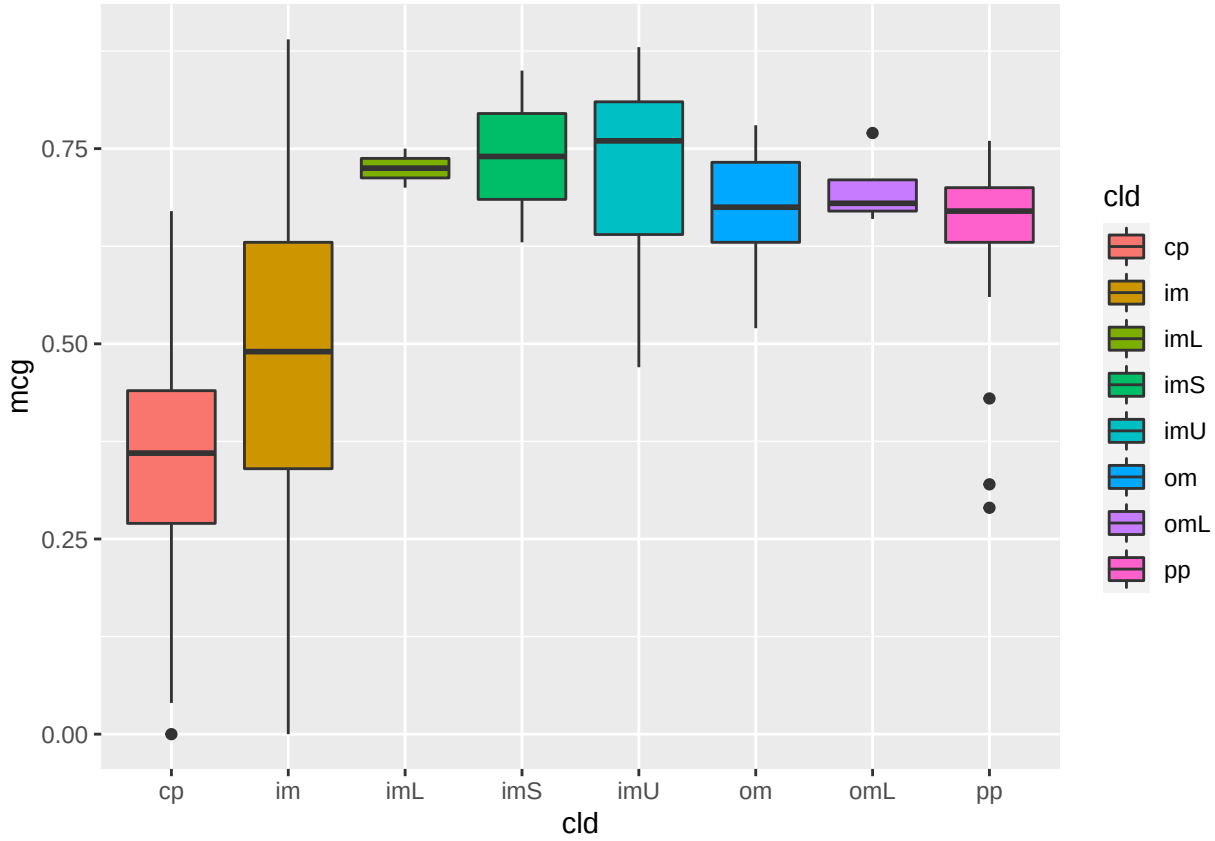

```
ecoli_df %>%
  ggplot(aes(x=alm2)) +
         geom_bar() +
         scale_x_binned()
```



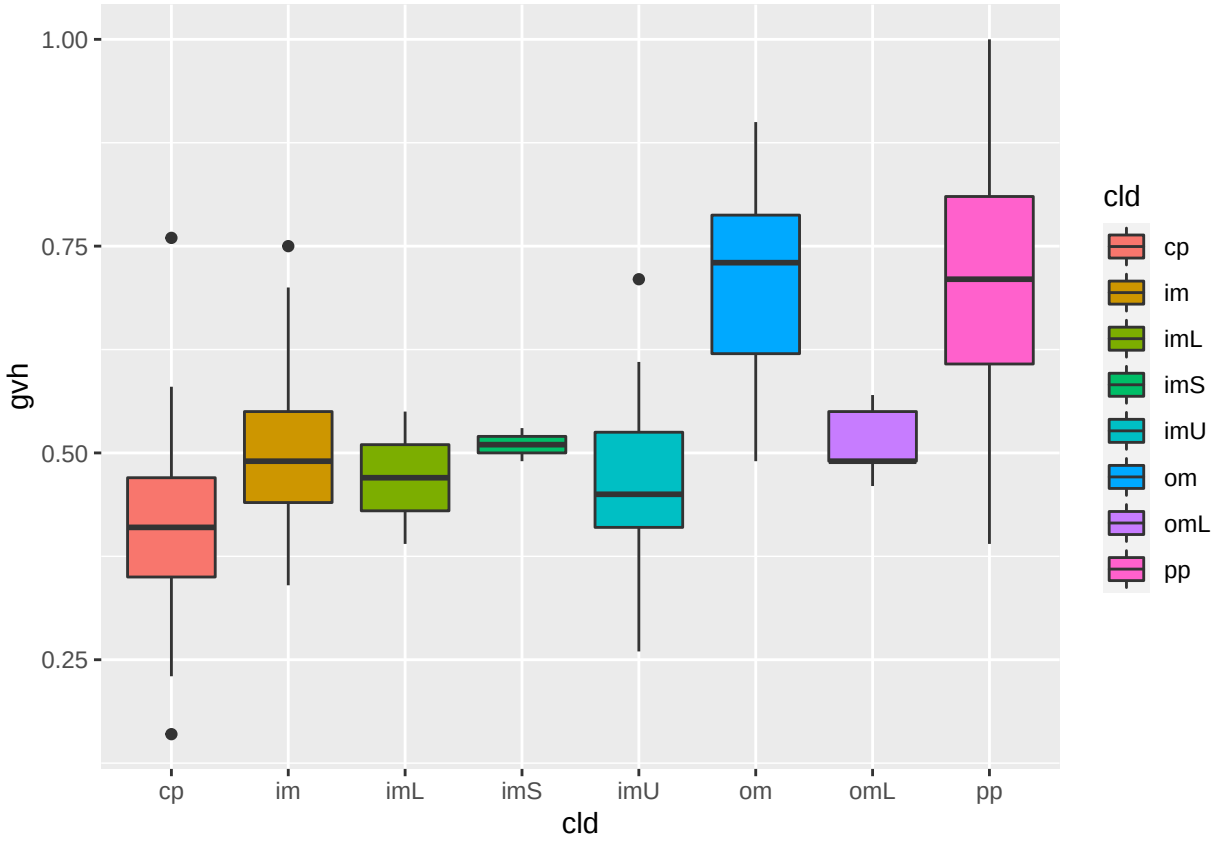
Shape of Data:

Box and whisker plots.

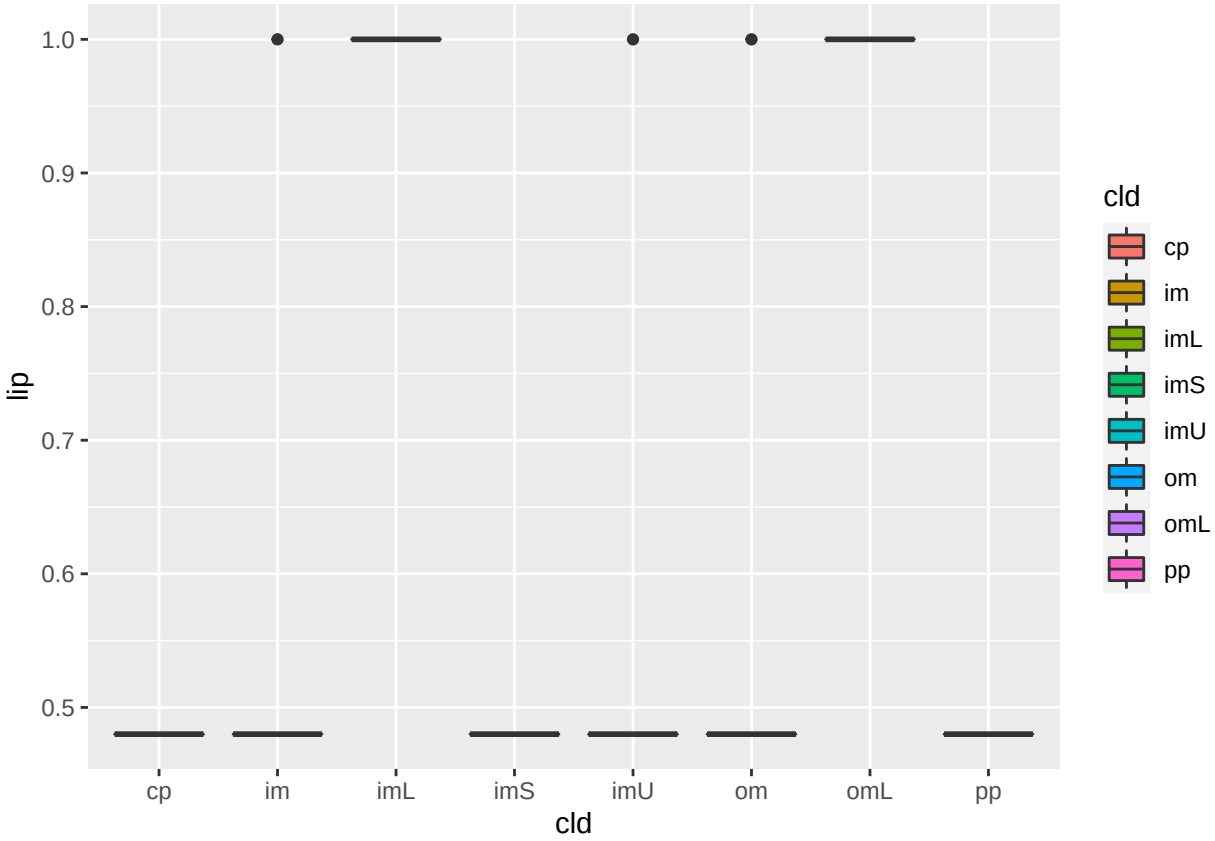
```
qplot(cld, mcg, data=ecoli_df, geom="boxplot", fill=cld)
```



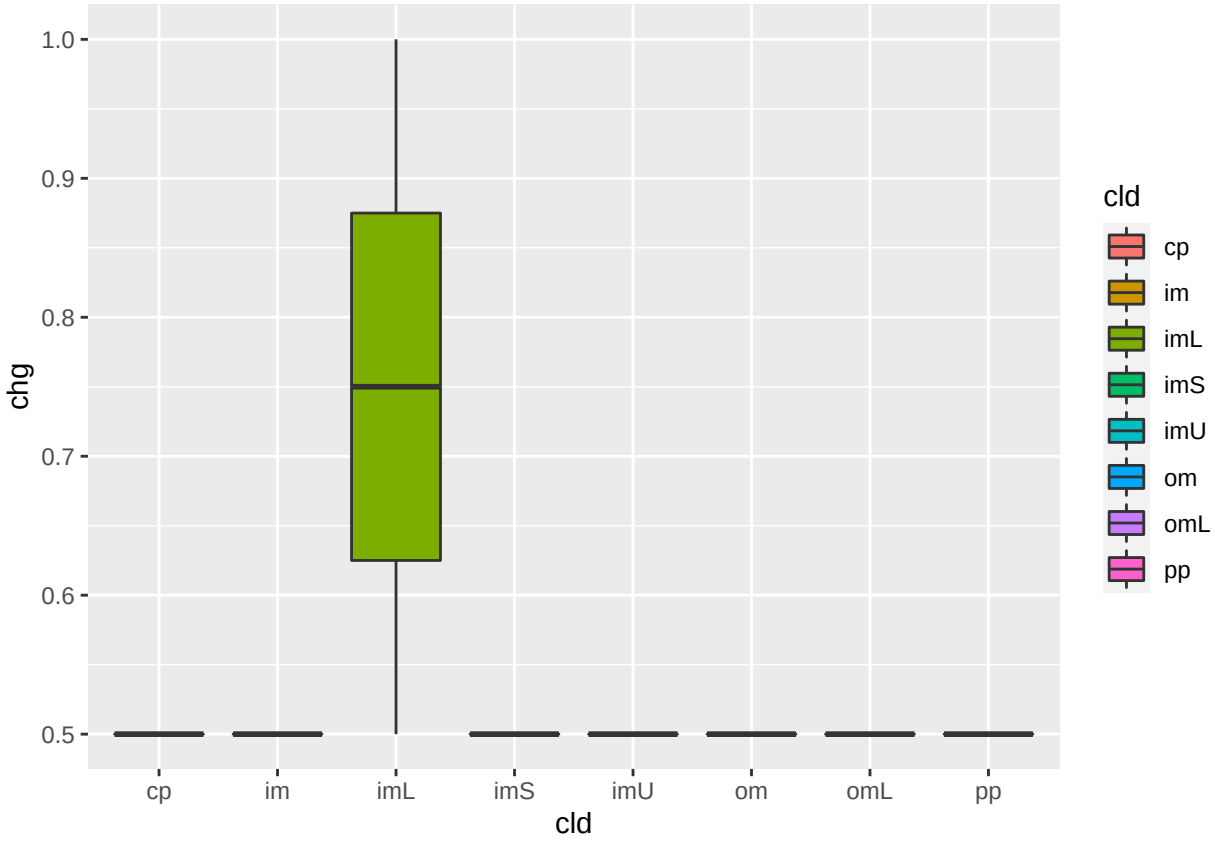

```
qplot(cld, gvh, data=ecoli_df, geom="boxplot", fill=cld)
```



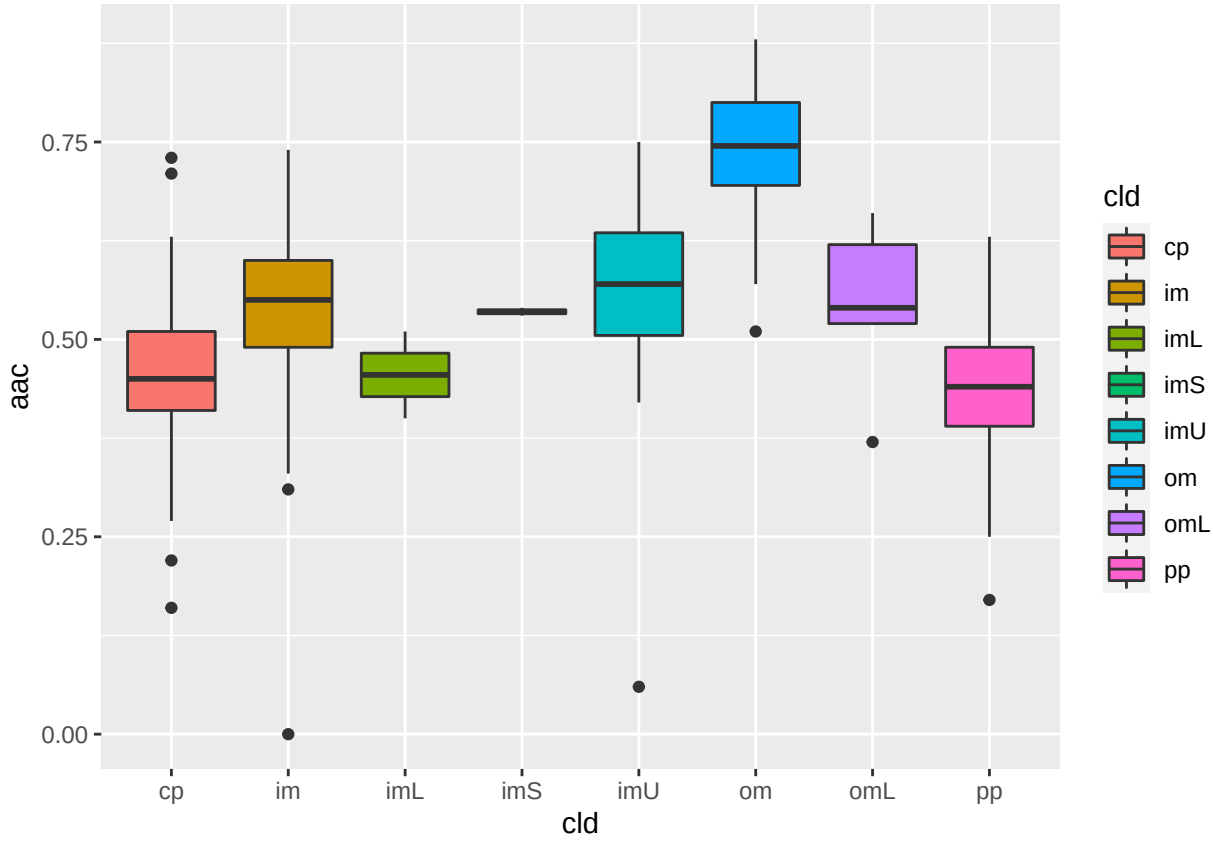

```
qplot(cld, lip, data=ecoli_df, geom="boxplot", fill=cld)
```



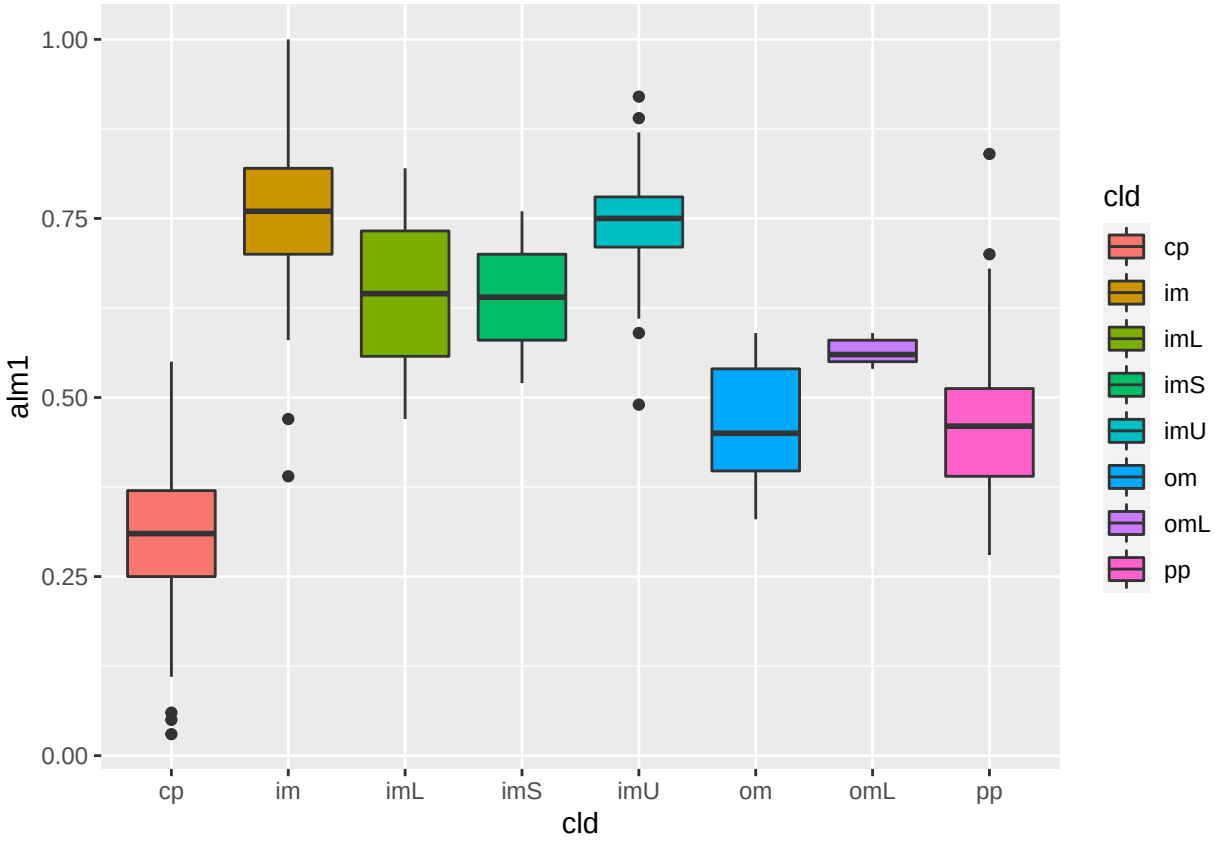

```
qplot(cld, chg, data=ecoli_df, geom="boxplot", fill=cld)
```



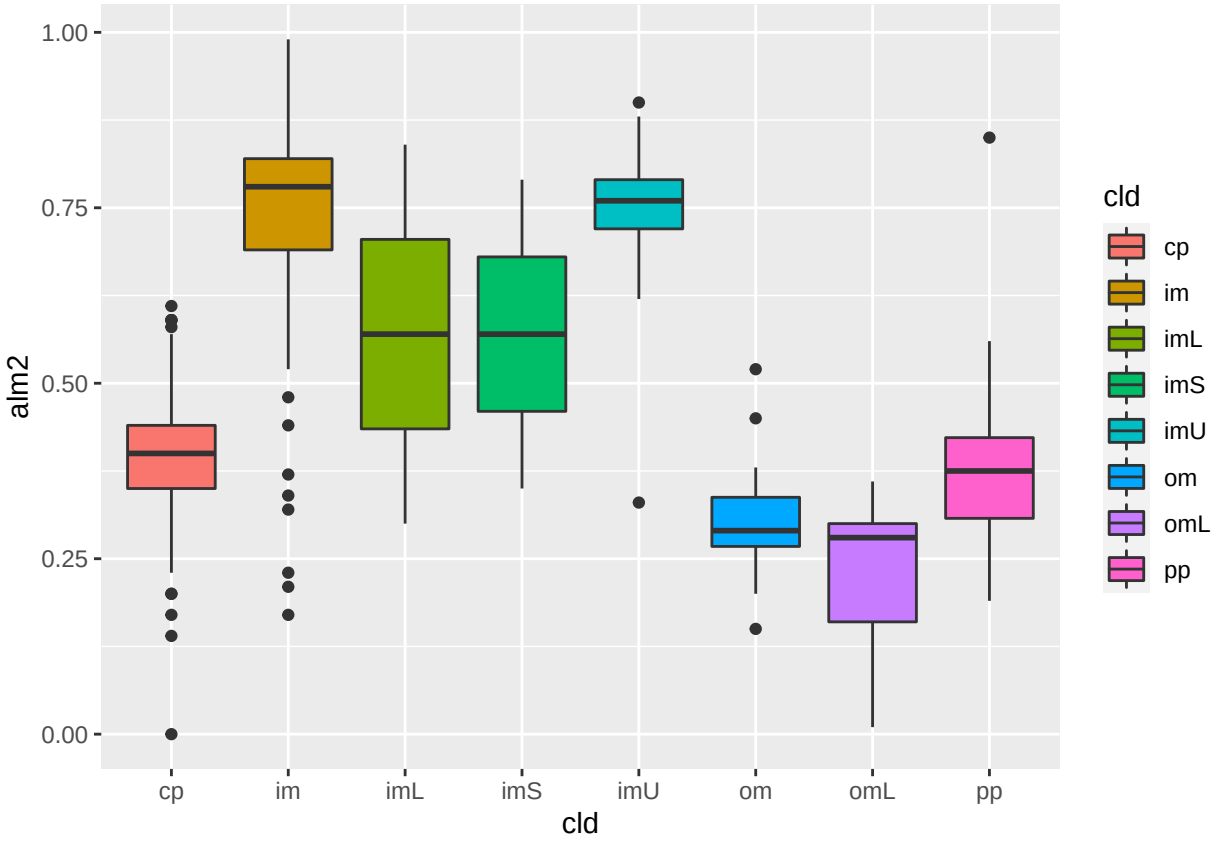

```
qplot(cld, aac, data=ecoli_df, geom="boxplot", fill=cld)
```




```
qplot(cld, alm1, data=ecoli_df, geom="boxplot", fill=cld)
```

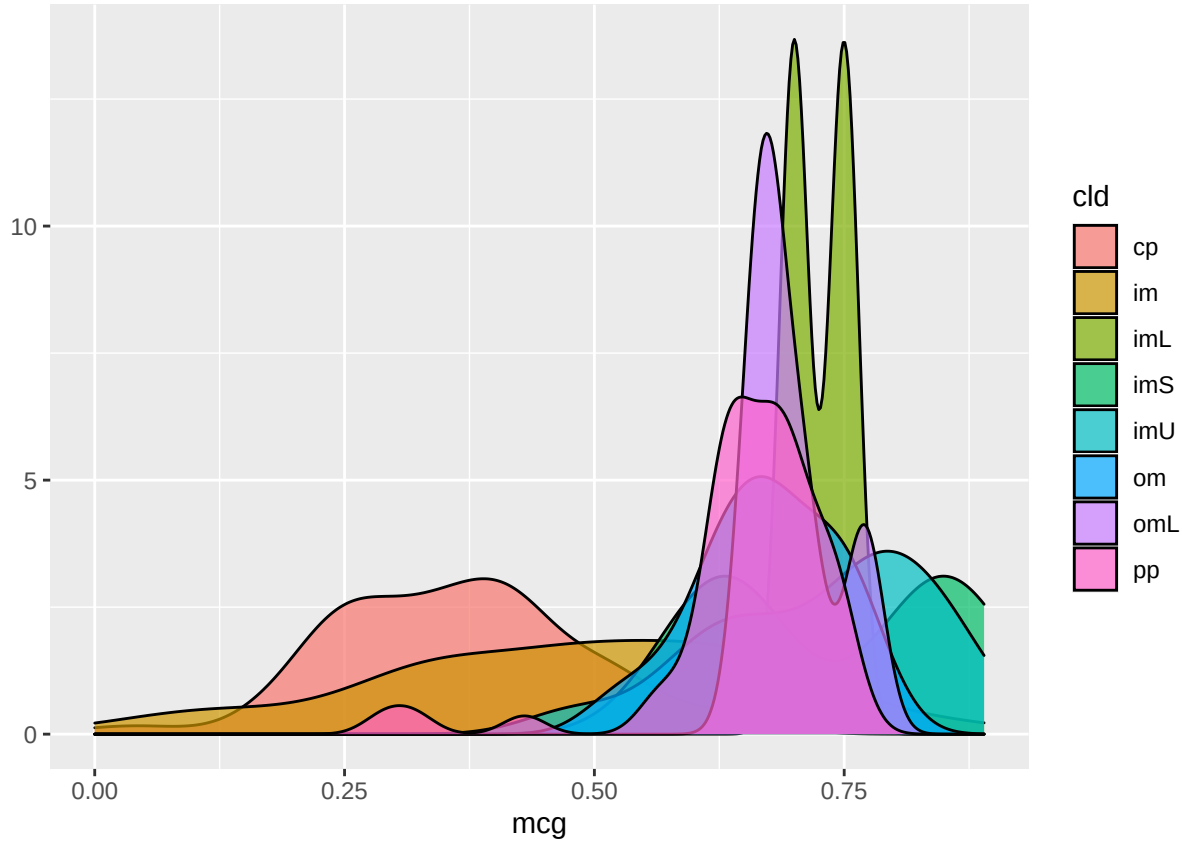



```
qplot(cld, alm2, data=ecoli_df, geom="boxplot", fill=cld)
```

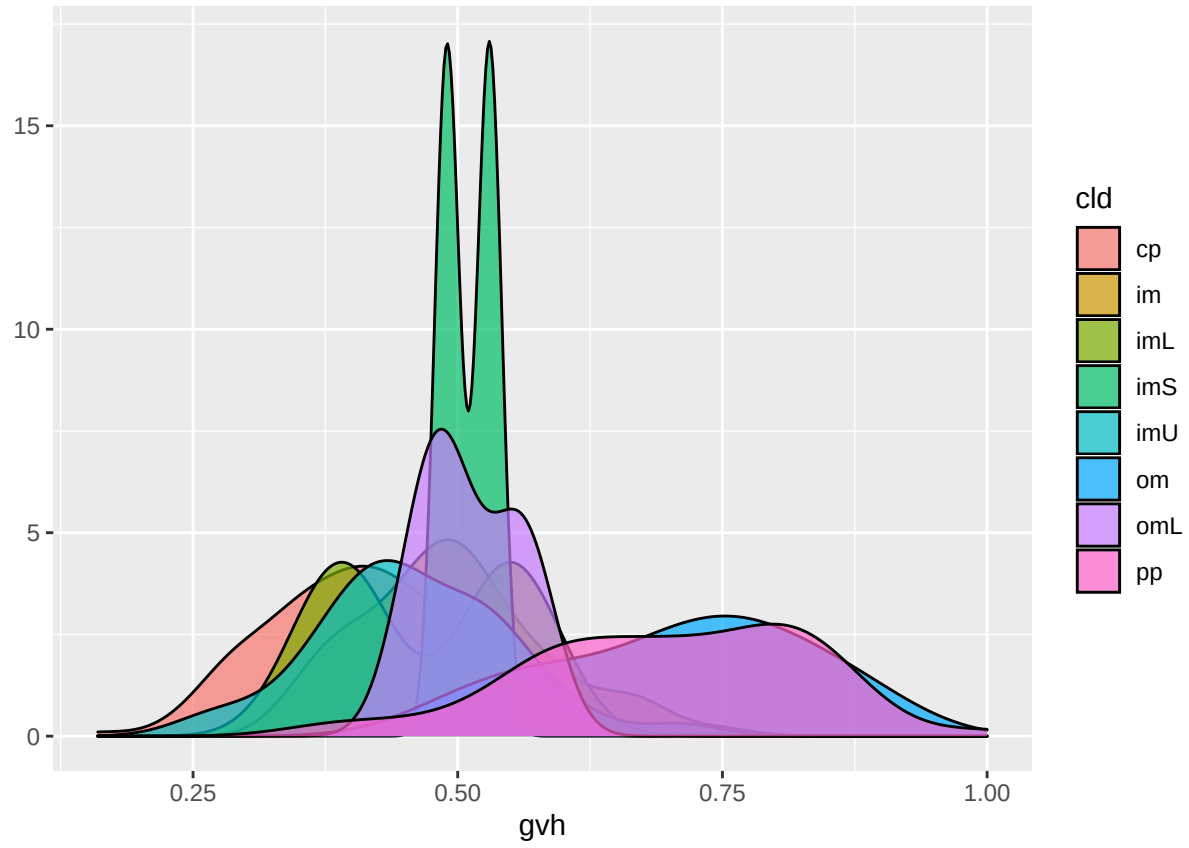


Overlapping density plot: Comparing distributions between the different categories of proteins and Class Distributions.

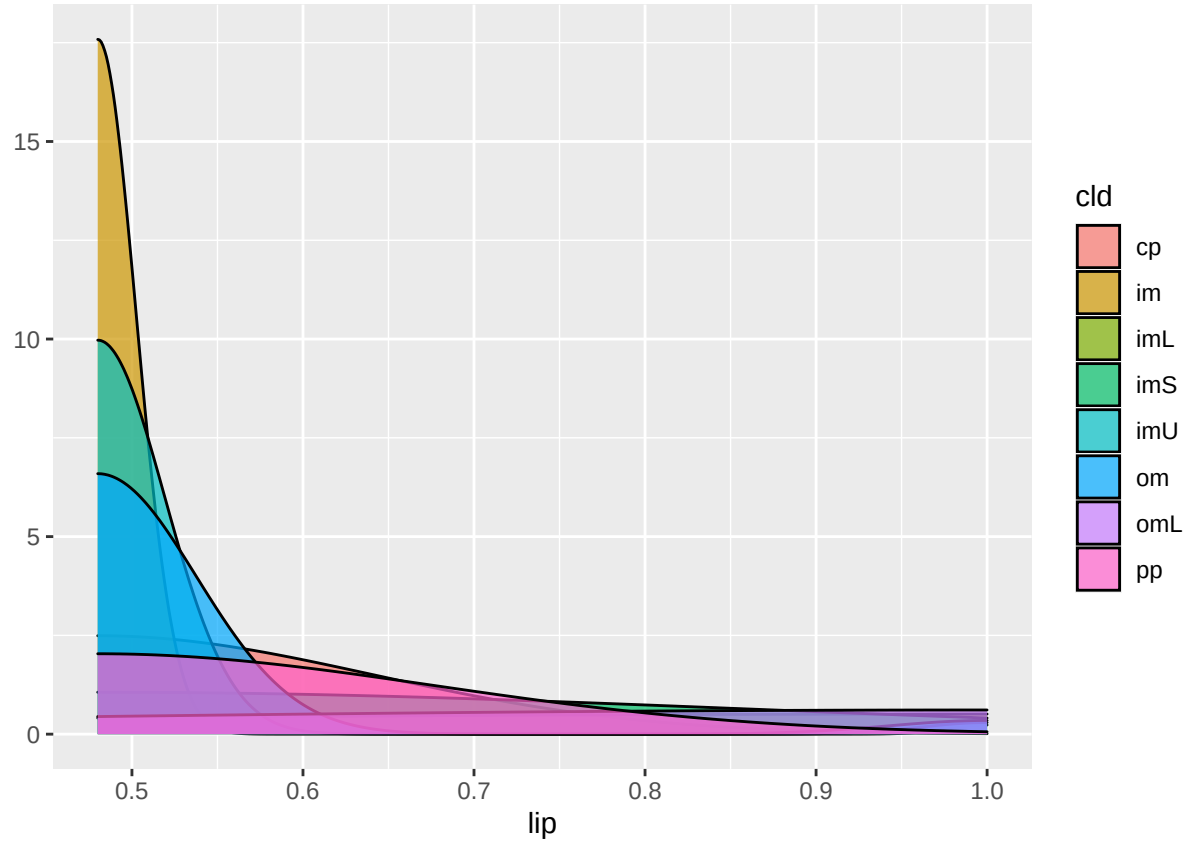
```
qplot(mcg, data=ecoli_df, geom="density", alpha=I(.7), fill=cld)
```



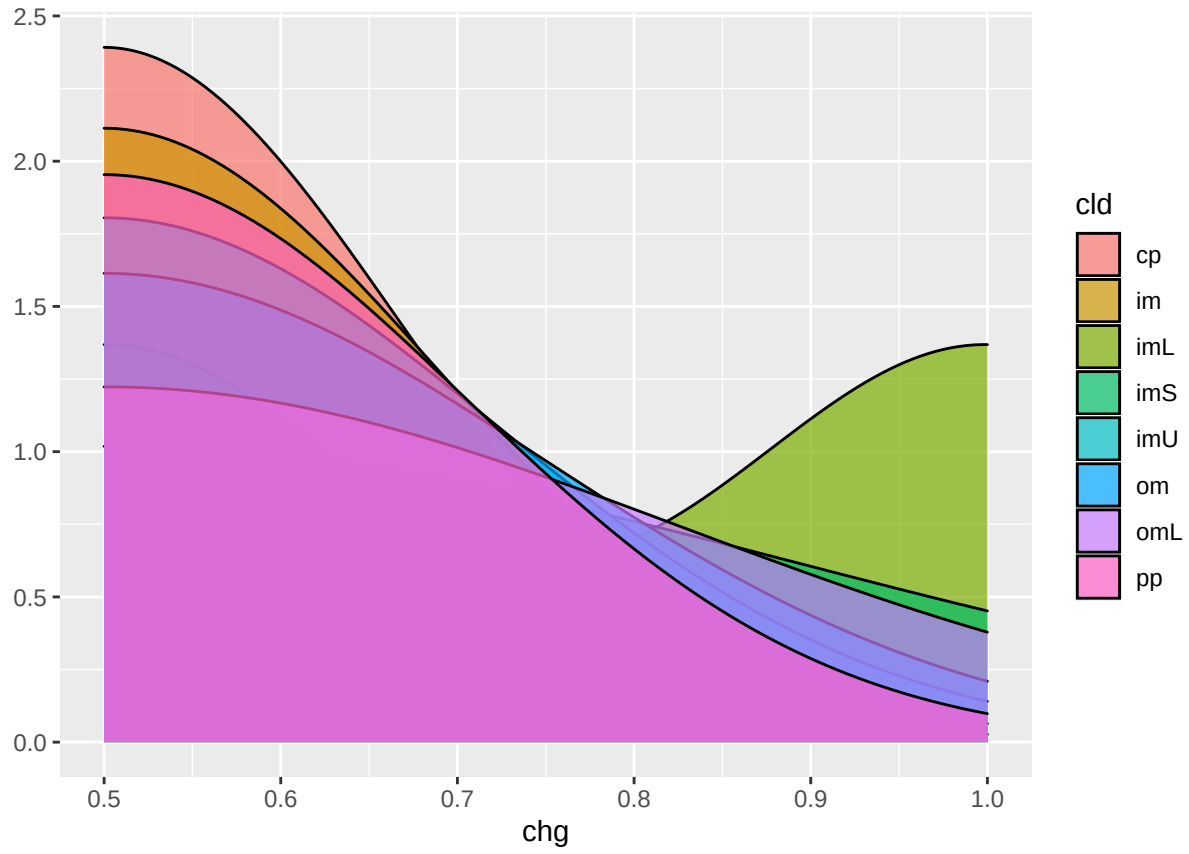

```
qplot(gvh, data=ecoli_df, geom="density", alpha=I(.7), fill=cld)
```



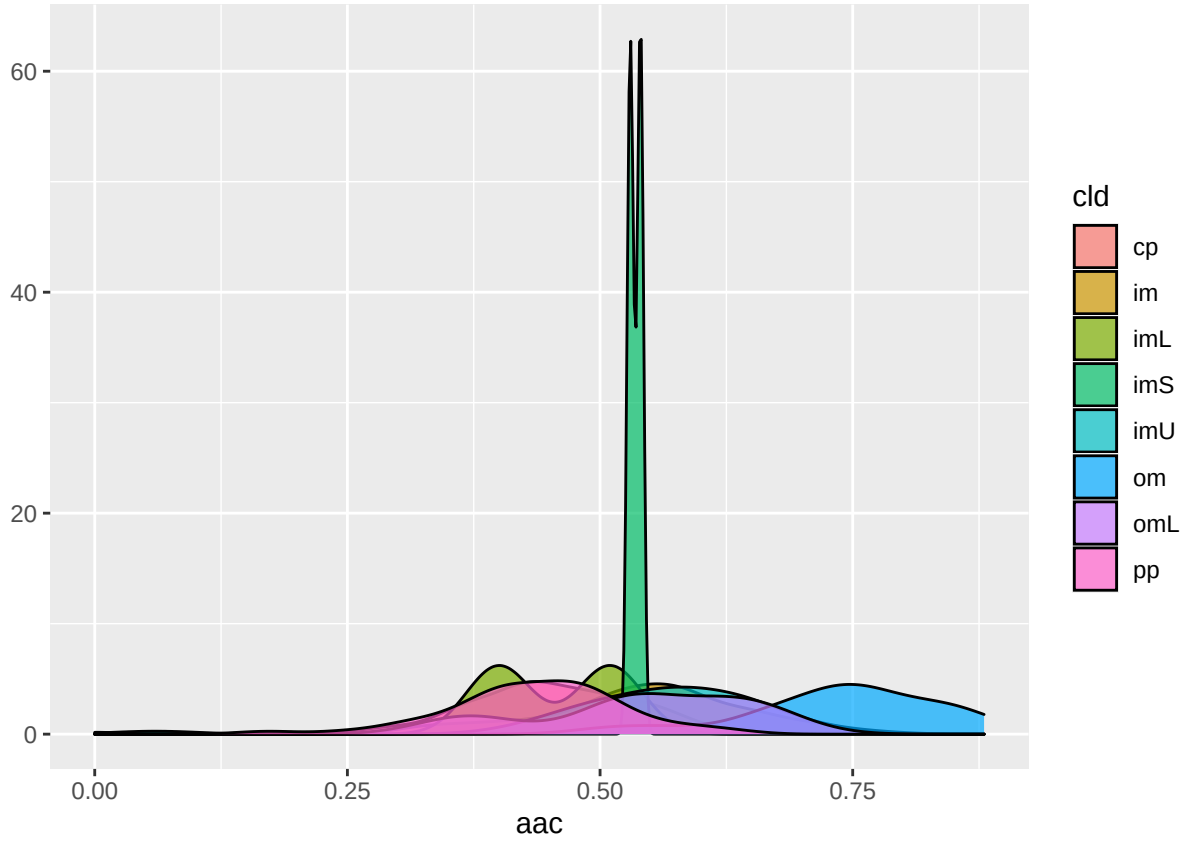

```
qplot(lip, data=ecoli_df, geom="density", alpha=I(.7), fill=cld)
```



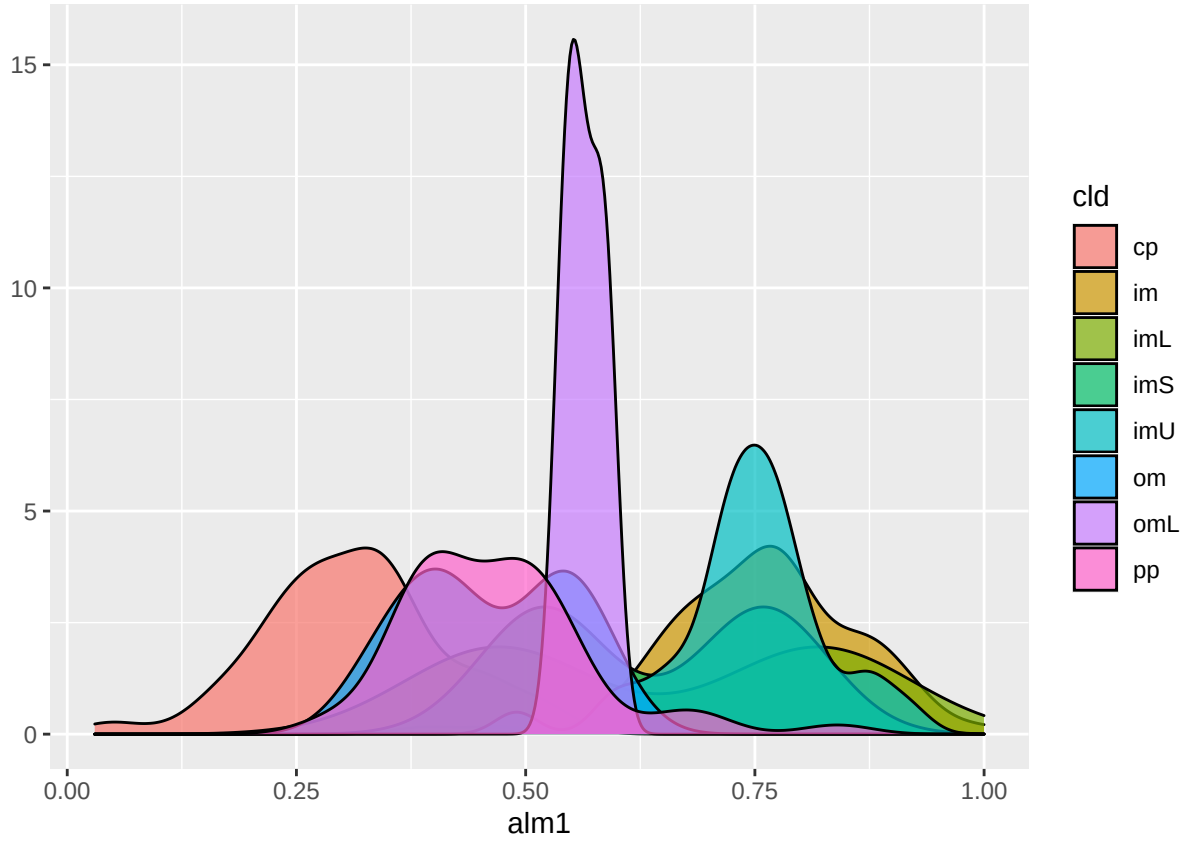

```
qplot(chg, data=ecoli_df, geom="density", alpha=I(.7), fill=cld)
```



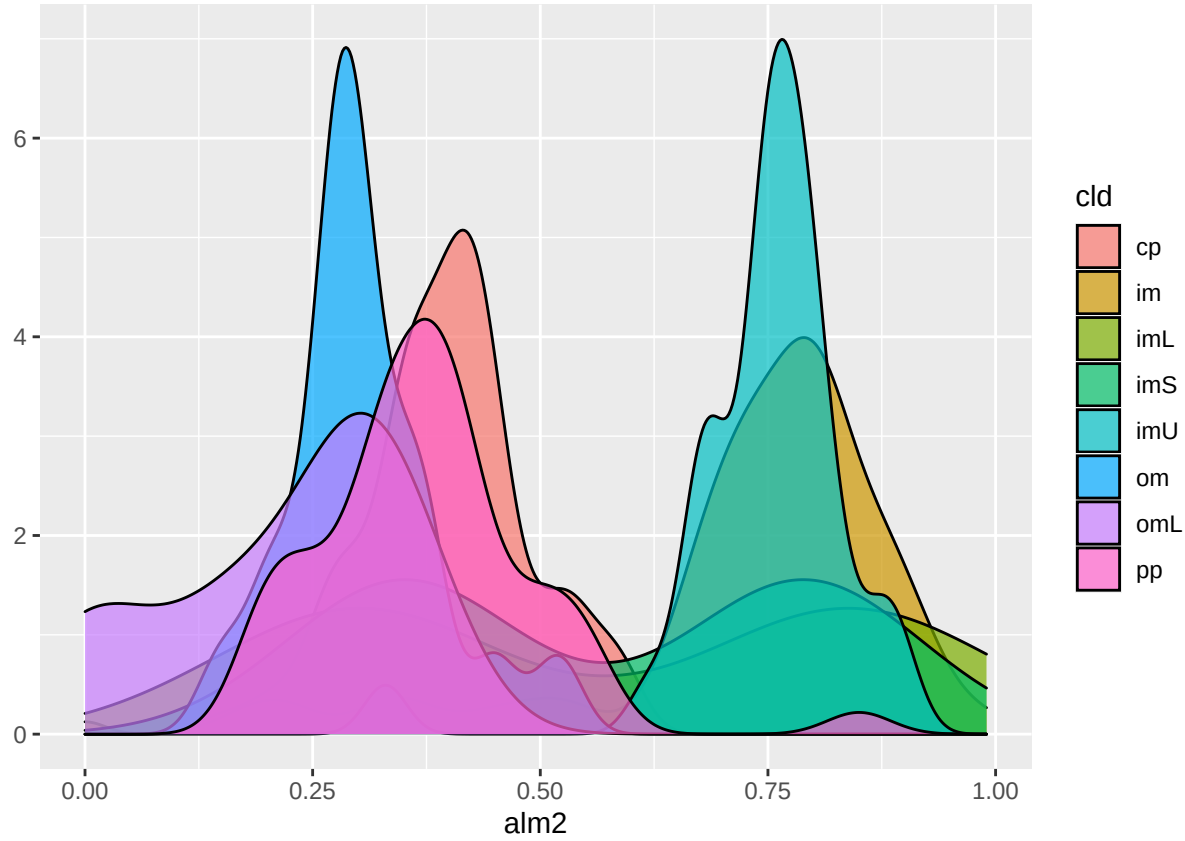

```
qplot(aac, data=ecoli_df, geom="density", alpha=I(.7), fill=cld)
```




```
qplot(alm1, data=ecoli_df, geom="density", alpha=I(.7), fill=cld)
```

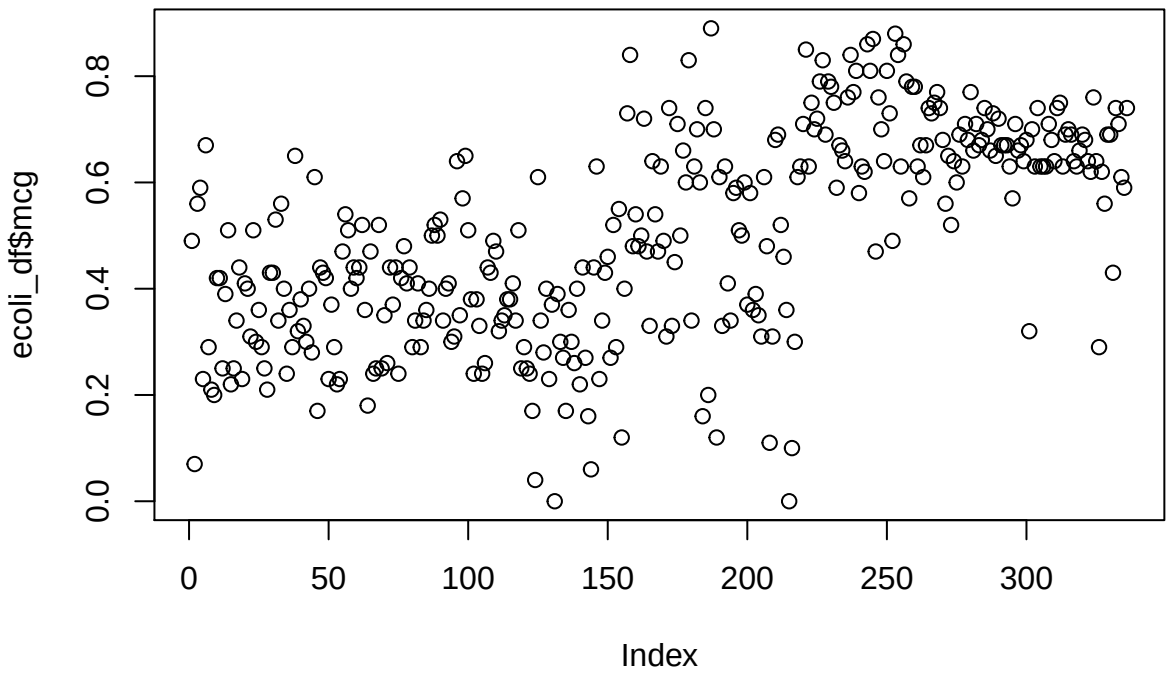



```
qplot(alm2, data=ecoli_df, geom="density", alpha=I(.7), fill=cld)
```

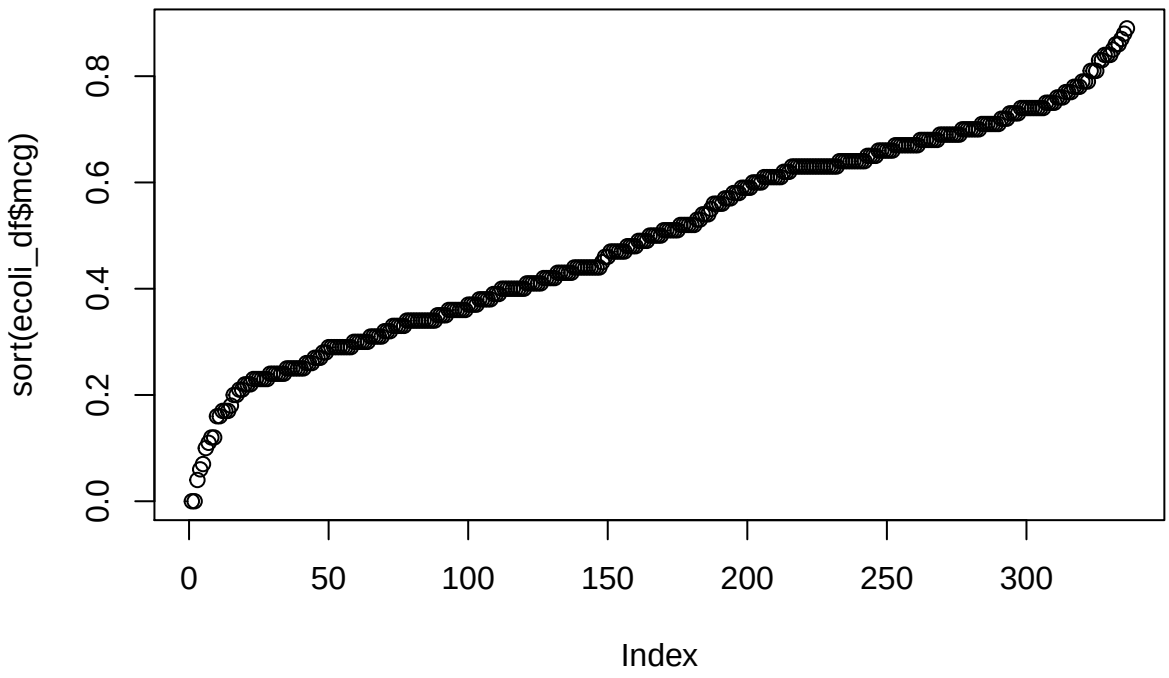


Plotting a Vectors: Print the elements of the vectors (proteins) according to their index.

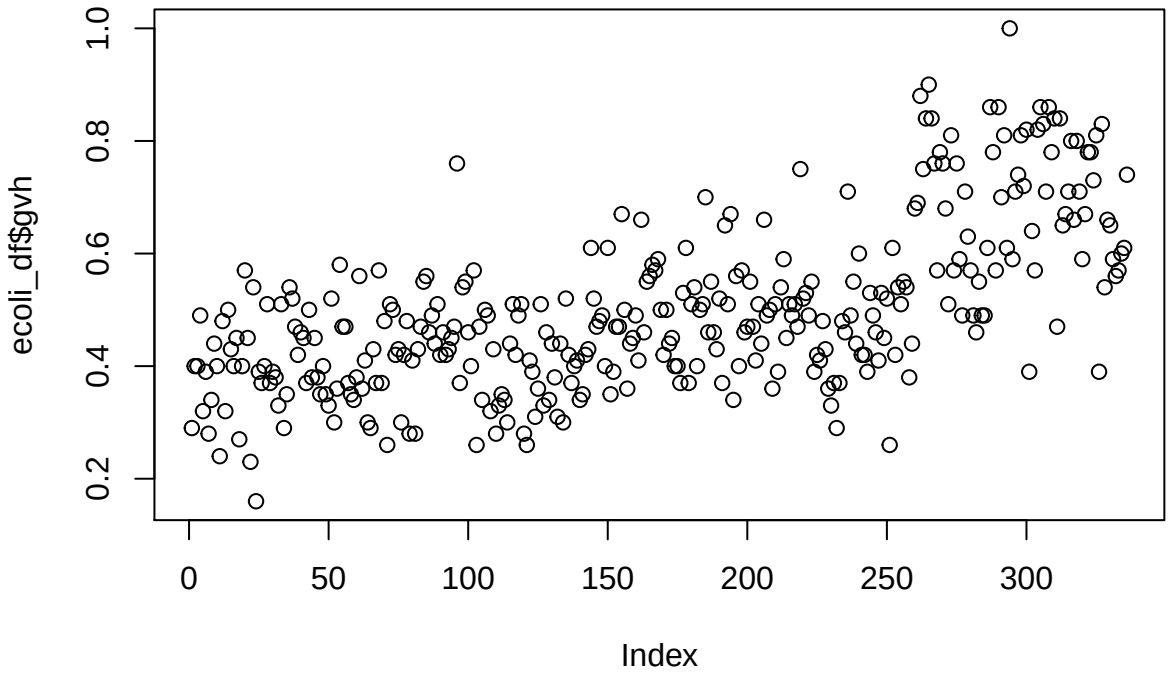
```
plot(ecoli_df$mcg) # Plot proteins for each observation.
```



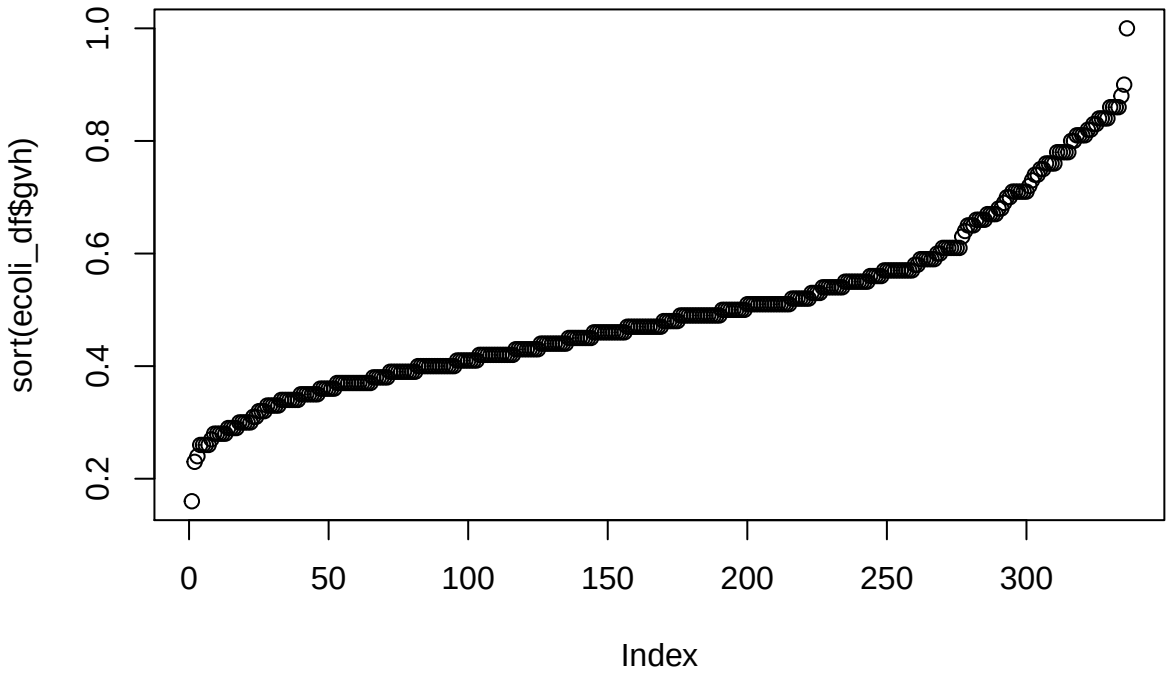

```
plot(sort(ecoli_df$mcg)) # Plot values against their ranks.
```



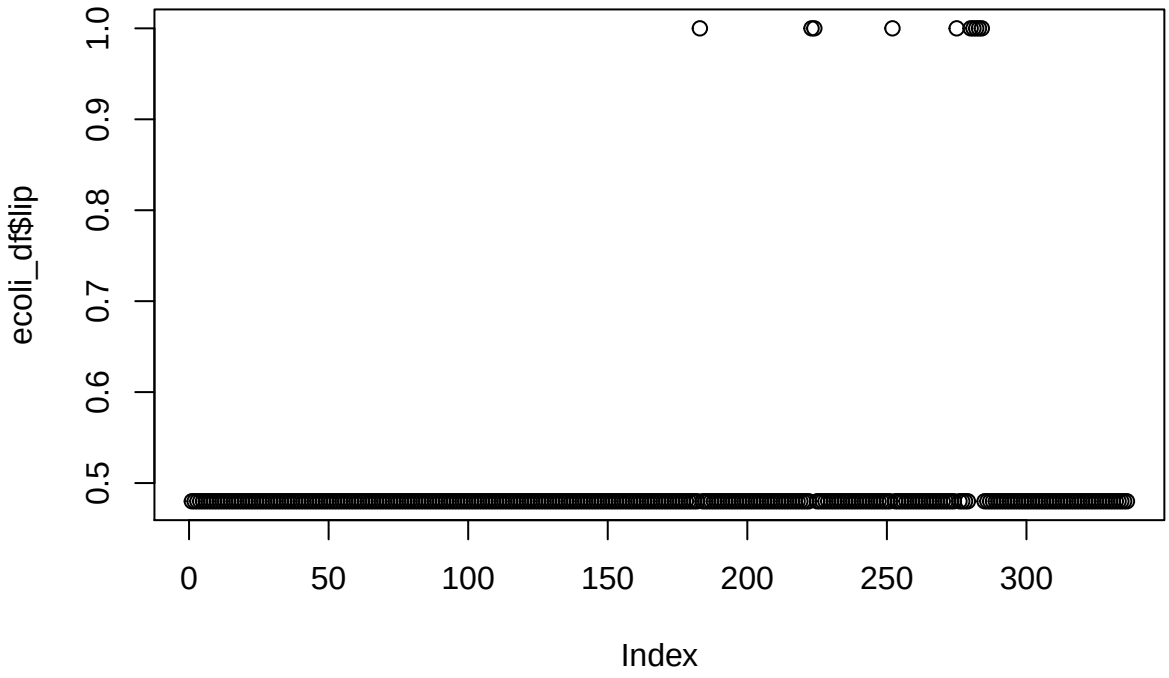

```
plot(ecoli_df$gvh)
```



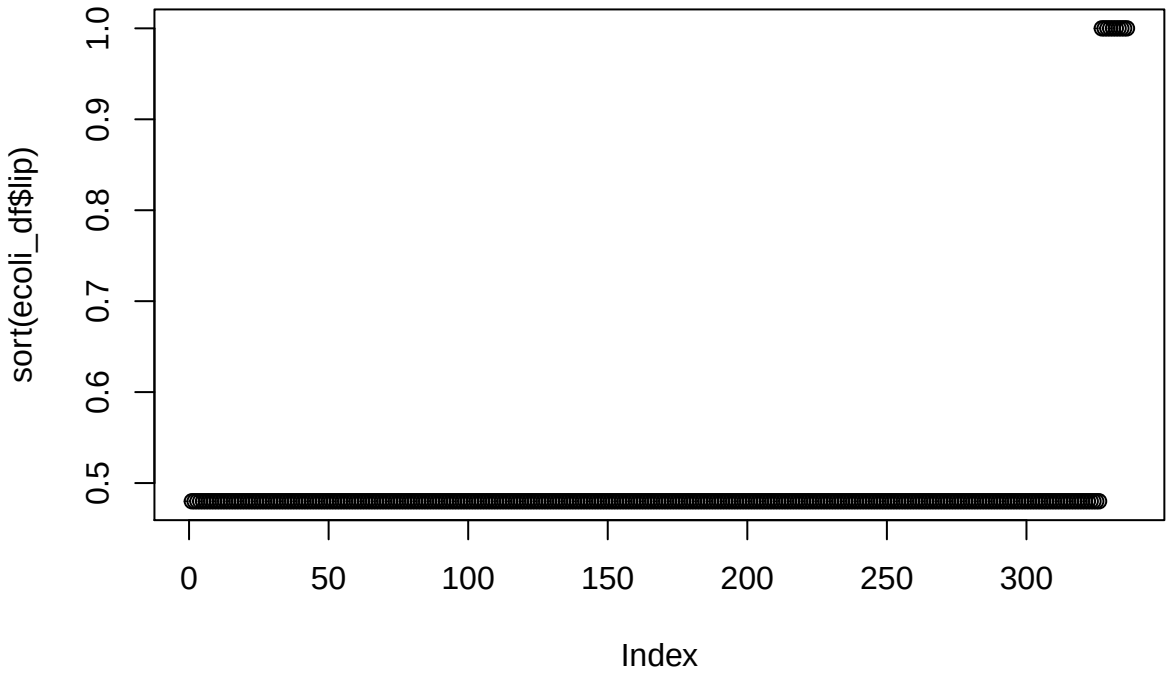

```
plot(sort(ecoli_df$gvh))
```



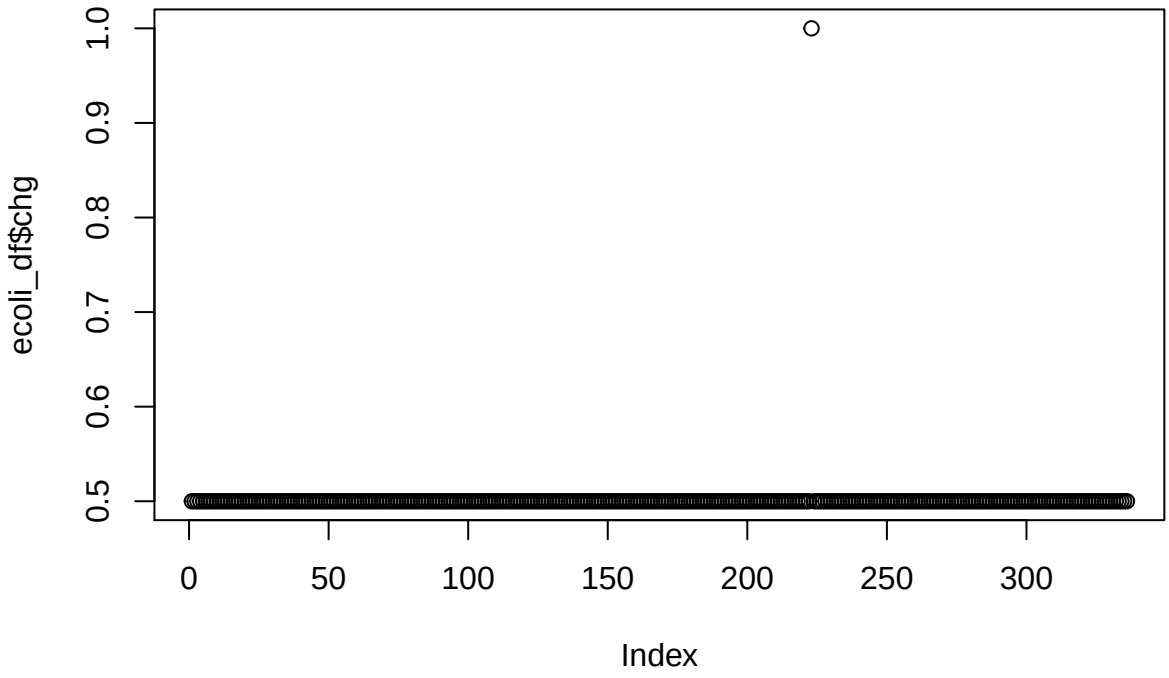

```
plot(ecoli_df$lip)
```



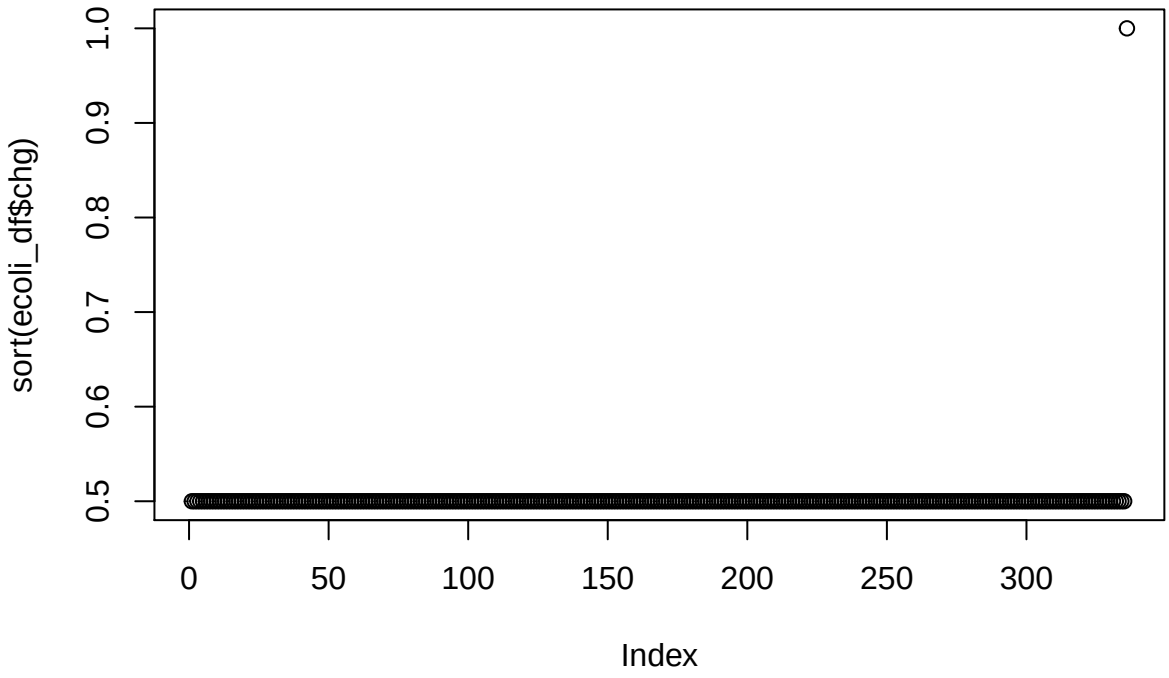

```
plot(sort(ecoli_df$lip))
```



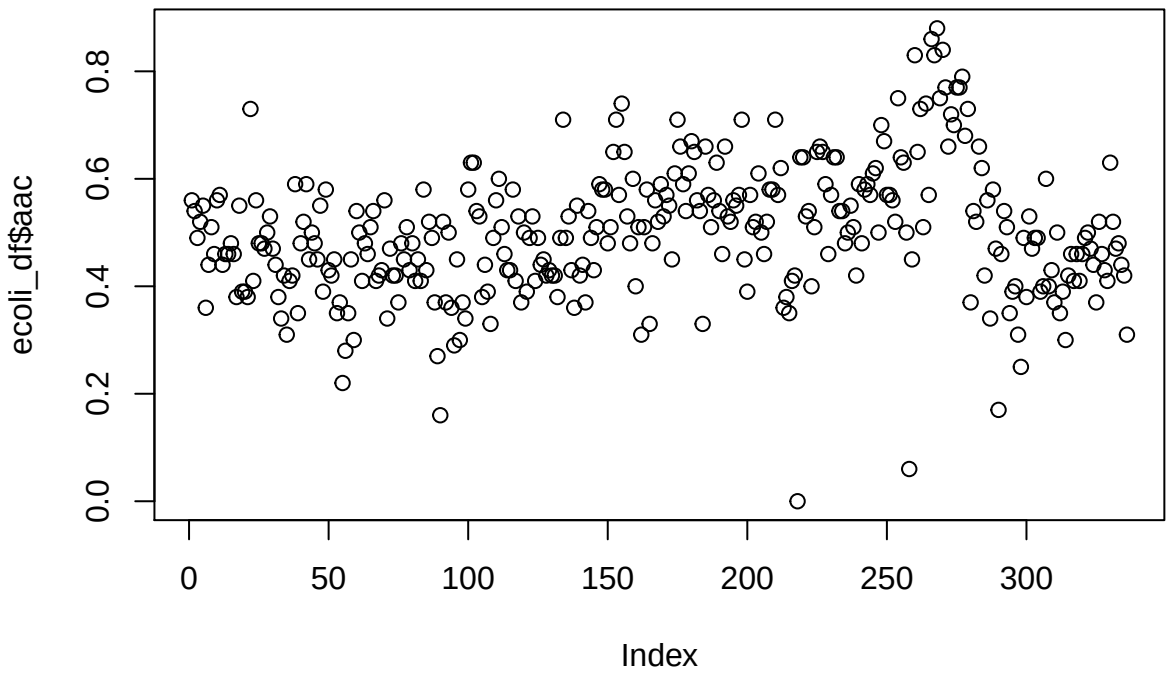

```
plot(ecoli_df$chg)
```



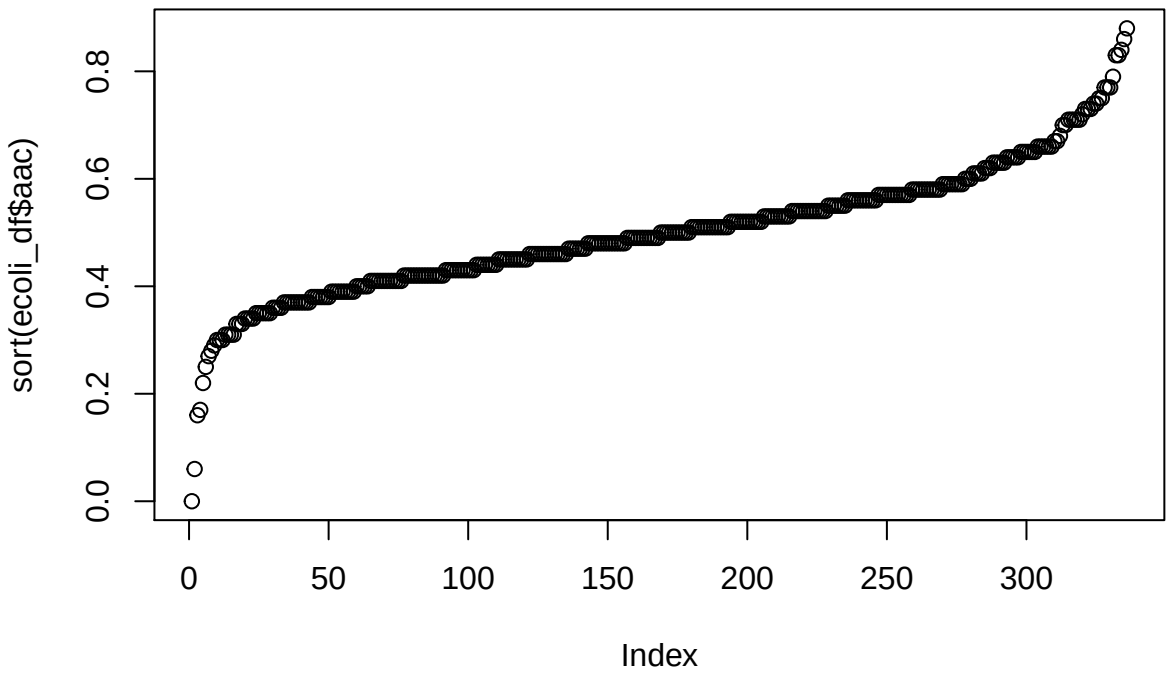

```
plot(sort(ecoli_df$chg))
```



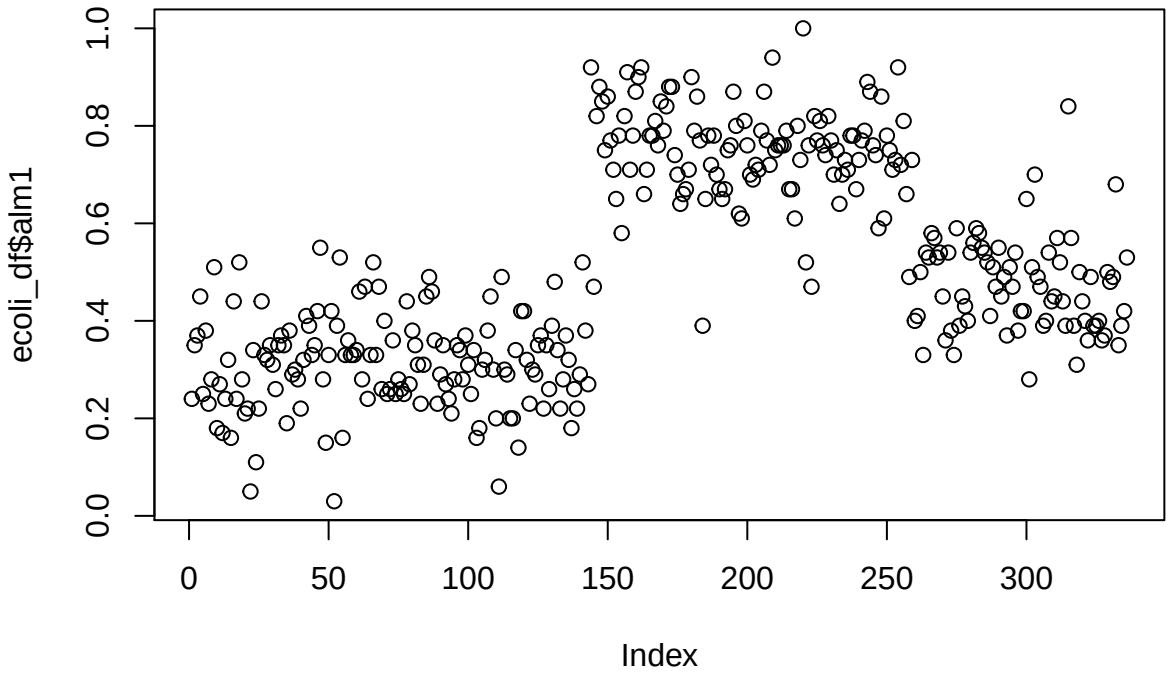

```
plot(ecoli_df$aac)
```



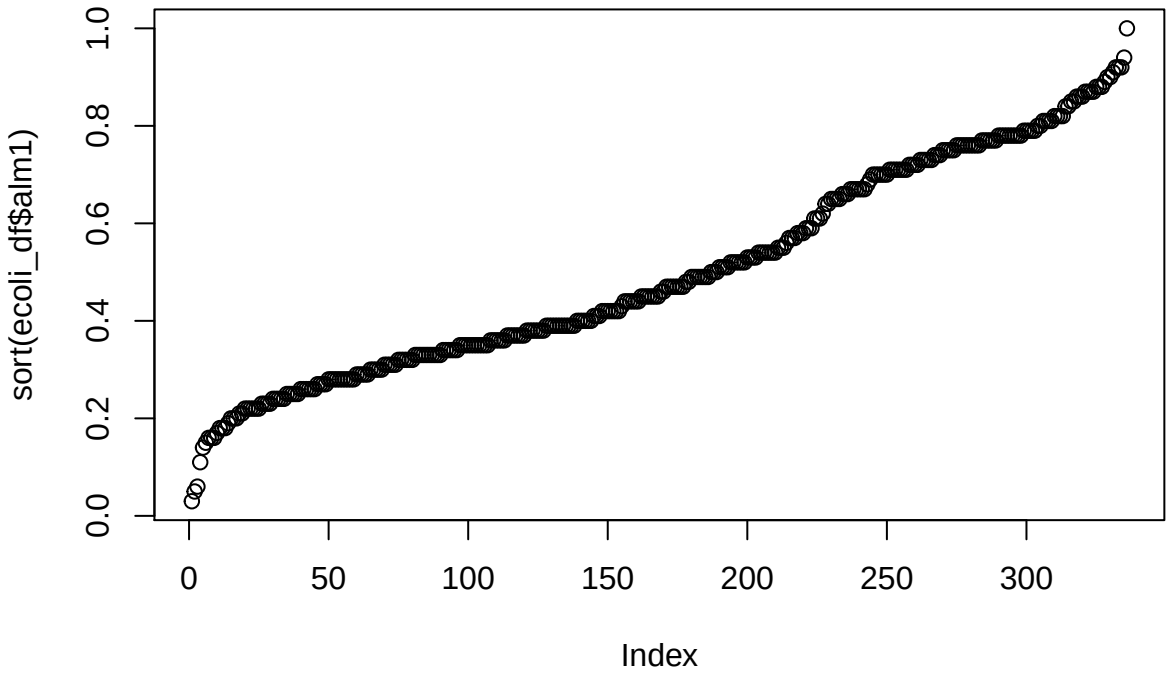

```
plot(sort(ecoli_df$aac))
```



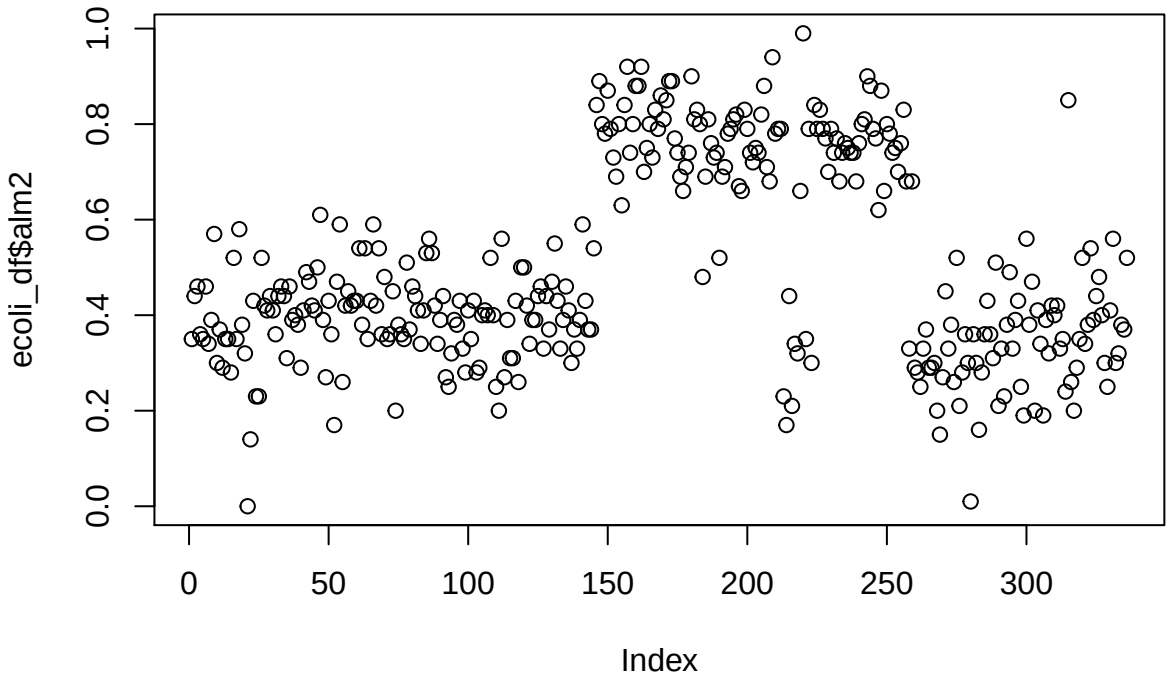

```
plot(ecoli_df$alm1)
```



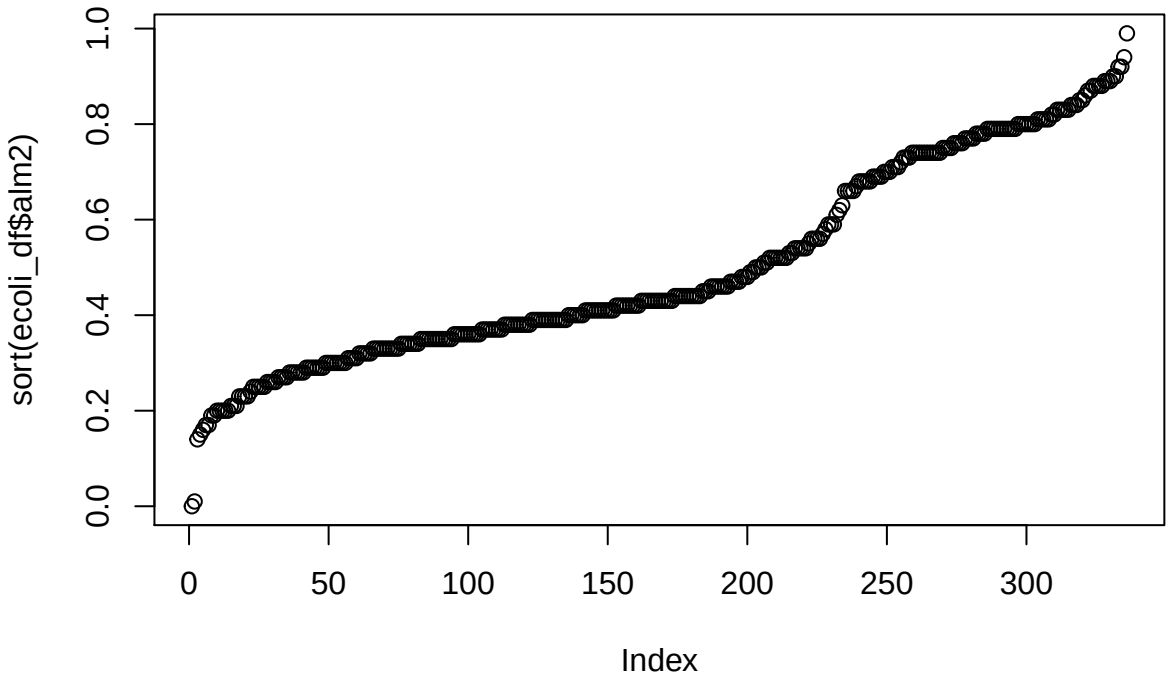

```
plot(sort(ecoli_df$alm1))
```




```
plot(ecoli_df$alm2)
```

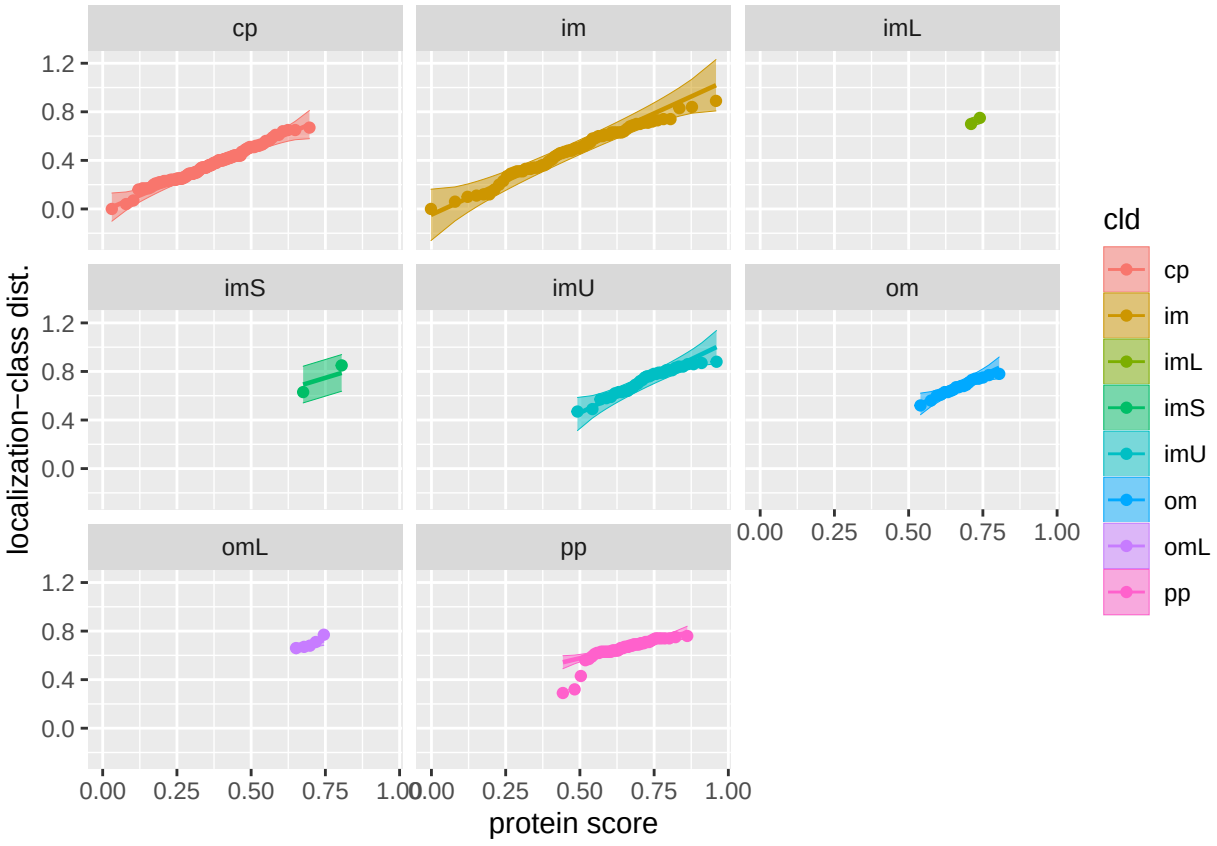



```
plot(sort(ecoli_df$alm2))
```

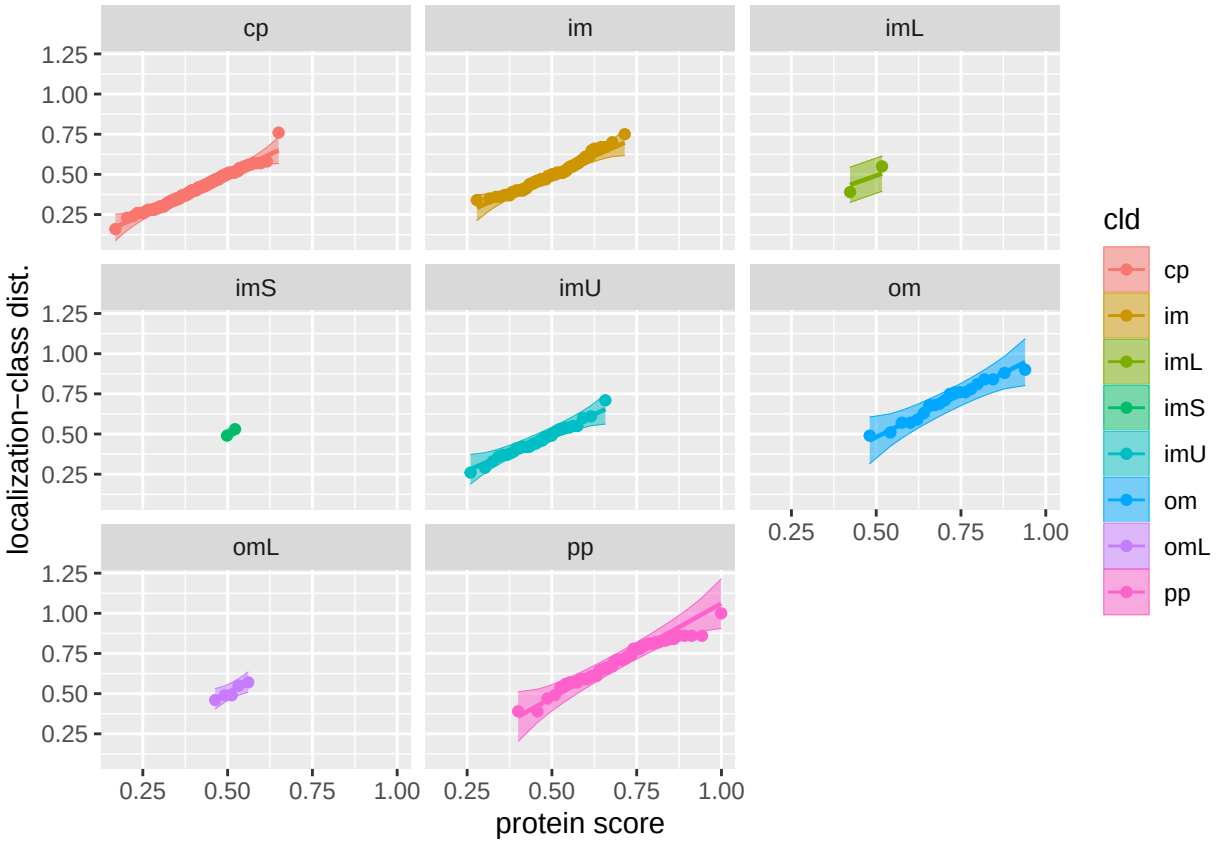


quantile-quantile (Q-Q) and probability-probability (P-P) points, lines, and confidence bands.

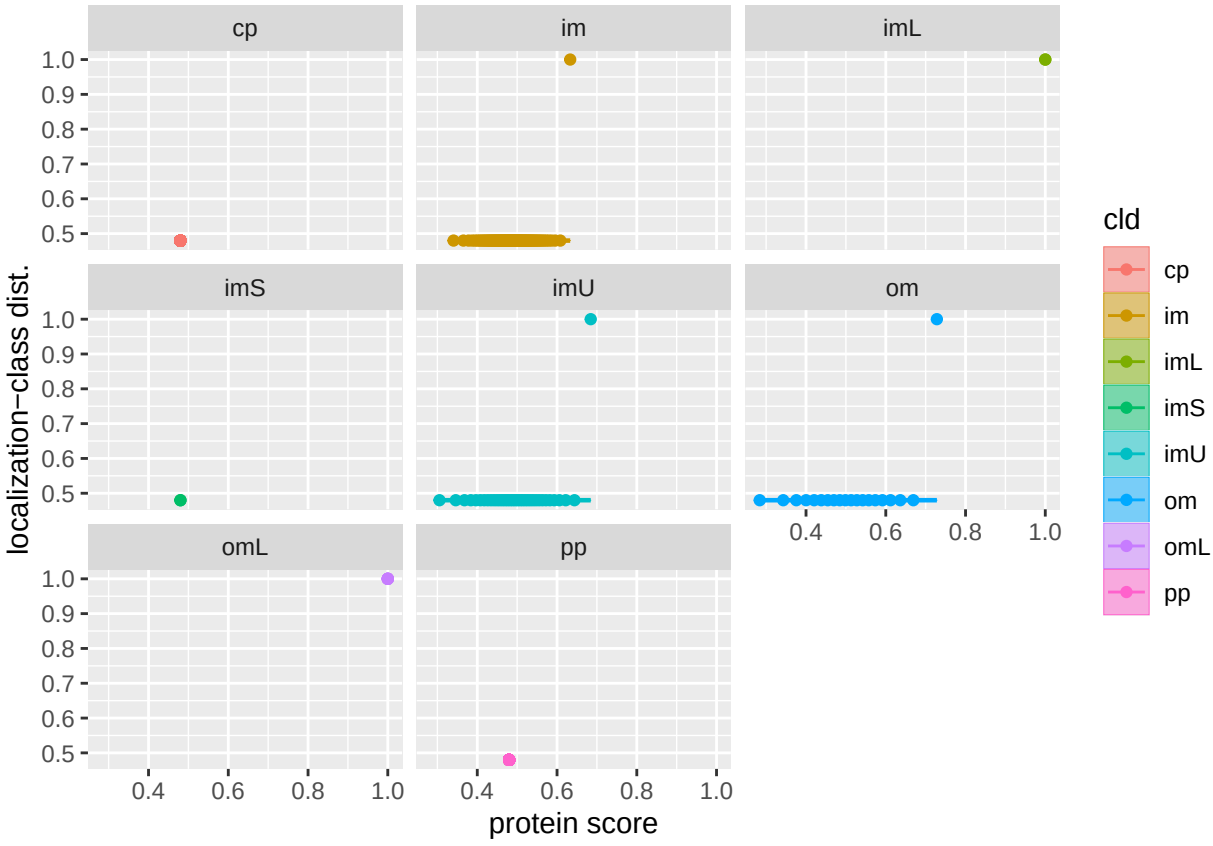
```
gg_mcg <- ggplot(data = ecoli_df, mapping = aes(sample = mcg, color = cld, fill = cld)) +
    stat_qq_band(alpha=0.5) +
    stat_qq_line() +
    stat_qq_point() +
    facet_wrap(~ cld) +
    labs(x = "protein score", y = "localization-class dist.")
gg_mcg
```



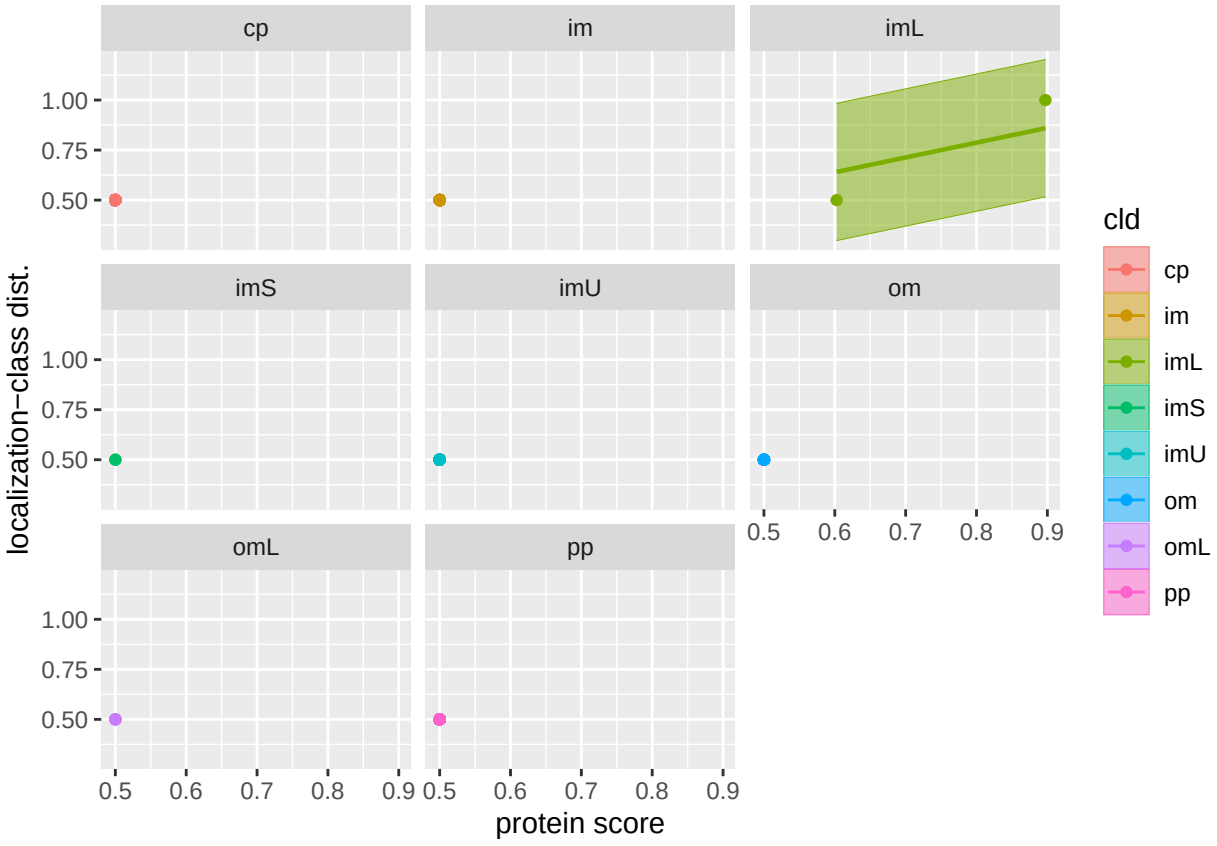

```
gg_gvh <- ggplot(data = ecoli_df, mapping = aes(sample = gvh, color = cld, fill = cld)) +
    stat_qq_band(alpha=0.5) +
    stat_qq_line() +
    stat_qq_point() +
    facet_wrap(~ cld) +
    labs(x = "protein score", y = "localization-class dist.")
gg_gvh
```



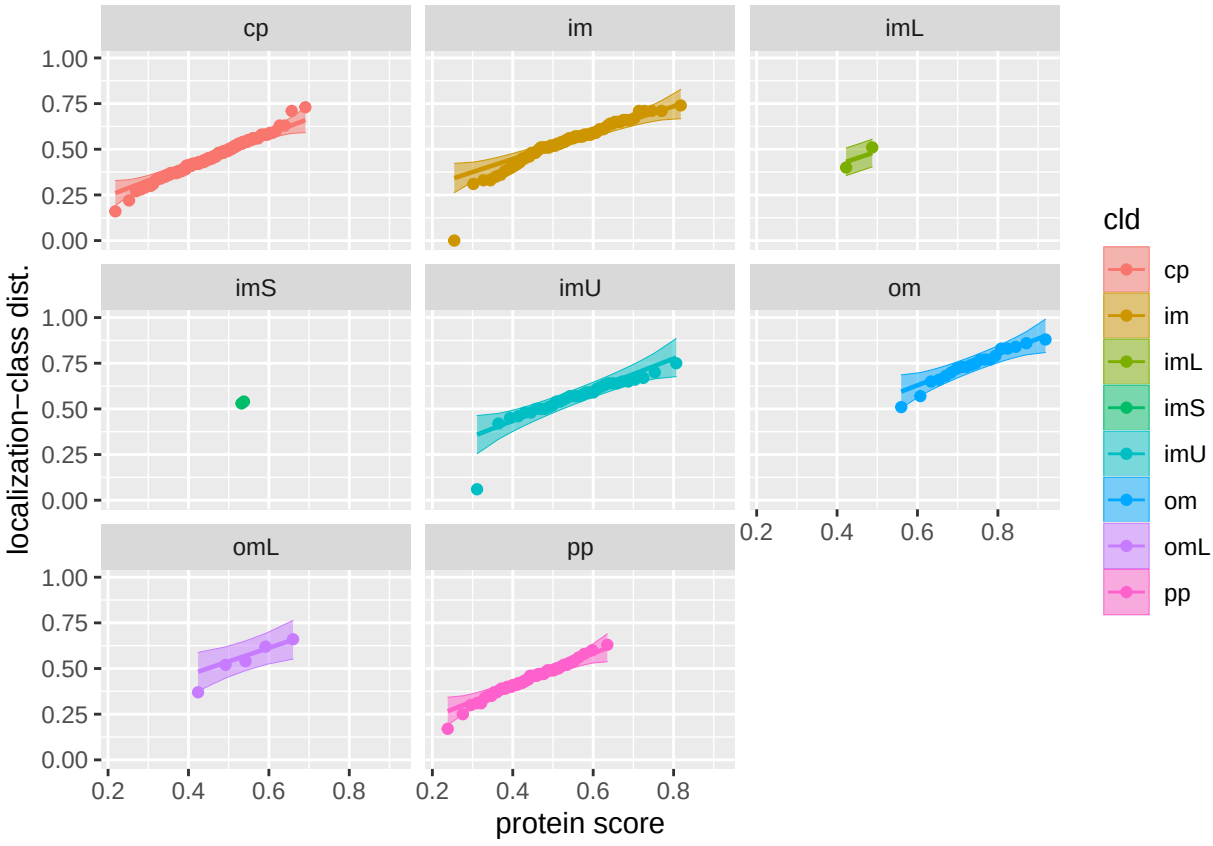

```
gg_lip <- ggplot(data = ecoli_df, mapping = aes(sample = lip, color = cld, fill = cld)) +
    stat_qq_band(alpha=0.5) +
    stat_qq_line() +
    stat_qq_point() +
    facet_wrap(~ cld) +
    labs(x = "protein score", y = "localization-class dist.")
gg_lip
```



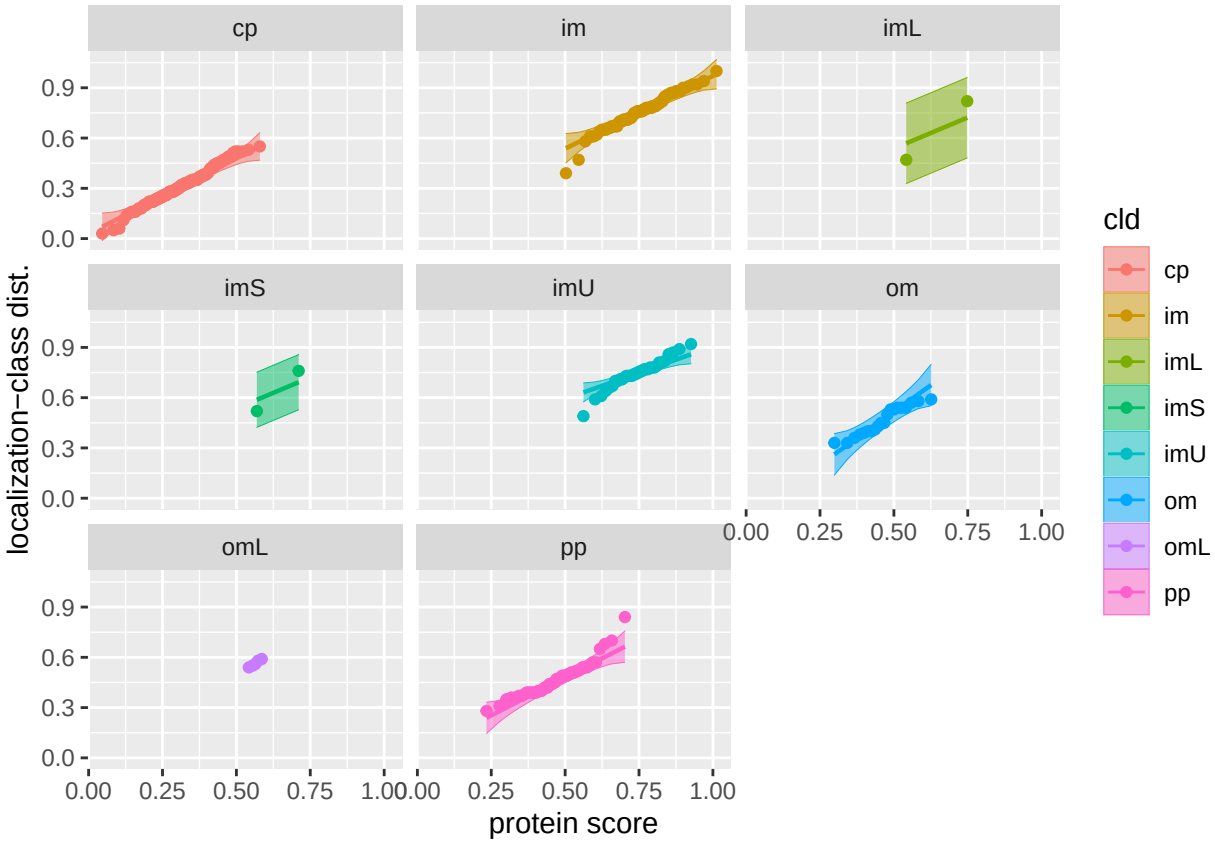

```
gg_chg <- ggplot(data = ecoli_df, mapping = aes(sample = chg, color = cld, fill = cld)) +
    stat_qq_band(alpha=0.5) +
    stat_qq_line() +
    stat_qq_point() +
    facet_wrap(~ cld) +
    labs(x = "protein score", y = "localization-class dist.")
gg_chg
```



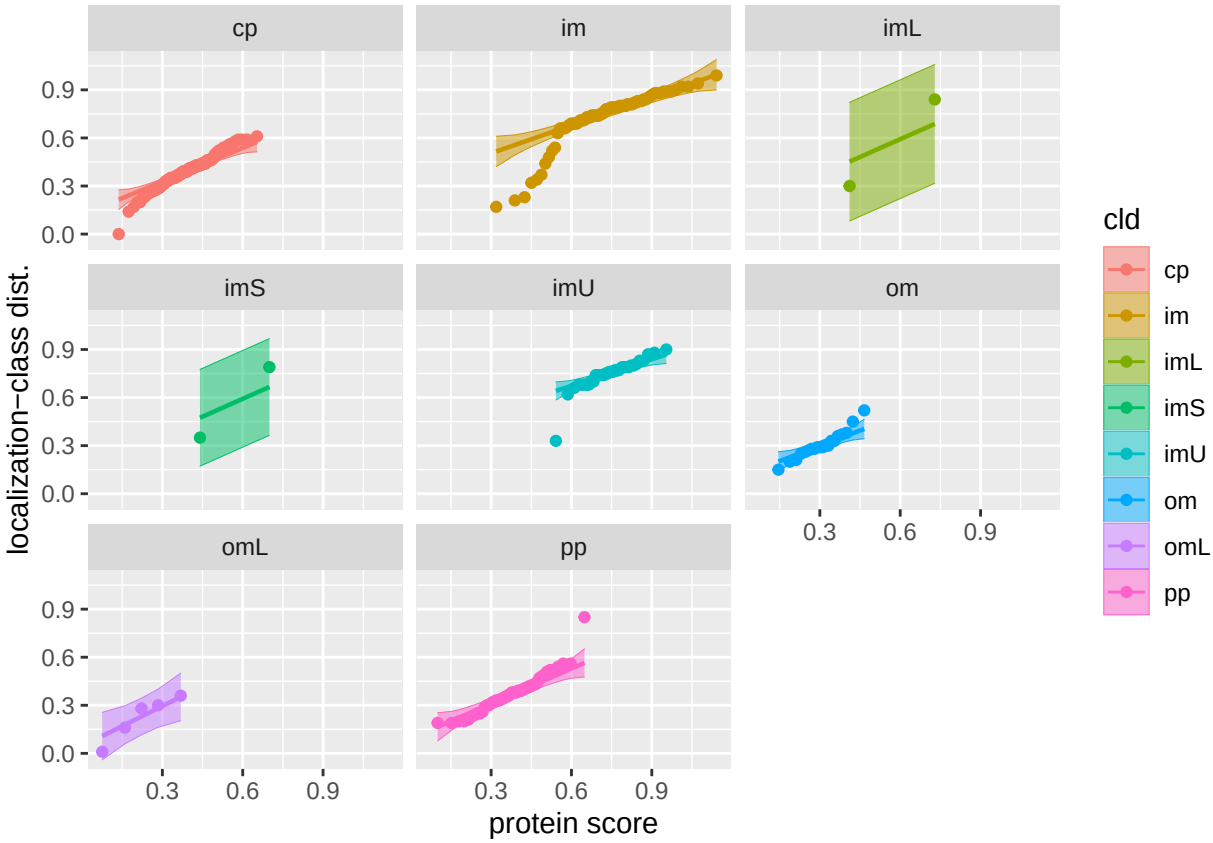

```
gg_aac <- ggplot(data = ecoli_df, mapping = aes(sample = aac, color = cld, fill = cld)) +
    stat_qq_band(alpha=0.5) +
    stat_qq_line() +
    stat_qq_point() +
    facet_wrap(~ cld) +
    labs(x = "protein score", y = "localization-class dist.")
gg_aac
```




```
gg_alm1 <- ggplot(data = ecoli_df, mapping = aes(sample = alm1, color = cld, fill = cld)) +
    stat_qq_band(alpha=0.5) +
    stat_qq_line() +
    stat_qq_point() +
    facet_wrap(~ cld) +
    labs(x = "protein score", y = "localization-class dist.")
gg_alm1
```

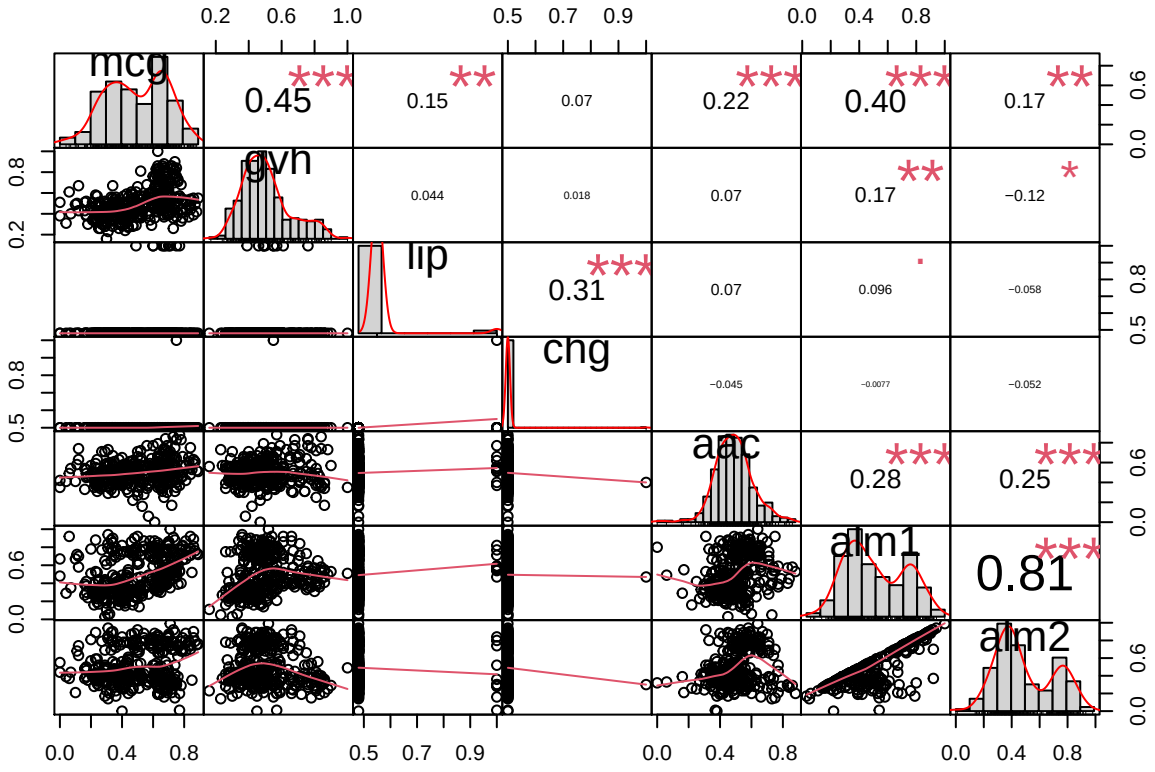



```
gg_alm2 <- ggplot(data = ecoli_df, mapping = aes(sample = alm2, color = cld, fill = cld)) +
    stat_qq_band(alpha=0.5) +
    stat_qq_line() +
    stat_qq_point() +
    facet_wrap(~ cld) +
    labs(x = "protein score", y = "localization-class dist.")
gg_alm2
```



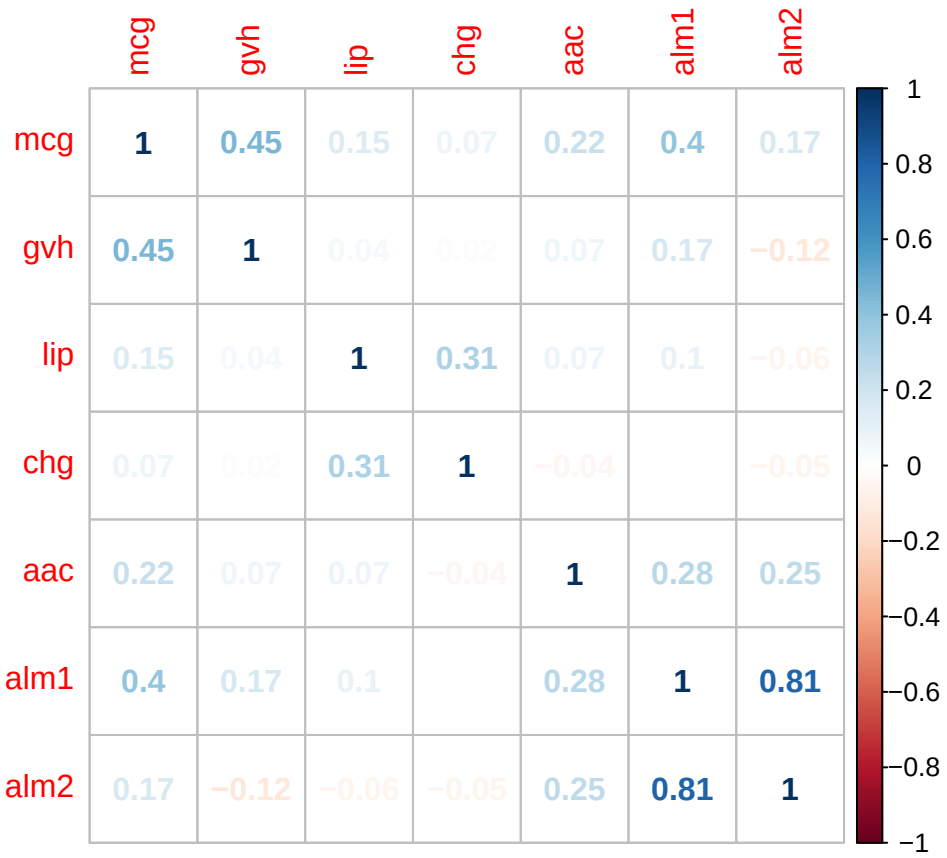
Correlation matrix using pairs plots: A quick way of finding out which variables in a data set are correlated with each other.

```
#library("PerformanceAnalytics")
ecoli_dfc1 <- ecoli_df[, c(2,3,4,5,6,7,8)] # Concatenate the columns we want to display.
chart.Correlation(ecoli_dfc1, histogram=TRUE, pch=19) # plot the graph
```



Attempt analysis of these raw numbers.

```
corrmatrix <- cor(ecoli_dfc1)
corrplot(corrmatrix, method = 'number')
```



We see that the variables alm1 and alm2 are highly correlated but the other variables are not highly correlated.

```
getAnywhere(correlation)
```

```
## A single object matching 'correlation' was found
## It was found in the following places
##   namespace:lava
## with value
##
## function (x, ...)
## UseMethod("correlation")
## <bytecode: 0x0000000033f771e8>
## <environment: namespace:lava>
```

Including Plots

```
ecoli_dfc1 <- ecoli_df[, c(2,3,4,5,6,7,8)] # Concatenate the columns we want to display.
lava:::correlation(ecoli_dfc1, histogram=TRUE, pch=19) # plot the graph
```

```
##                Estimate
## mcg~~gvh    0.454805259
## mcg~~lip    0.146841483
## mcg~~chg    0.070265602
```

```
## mcg~~aac    0.220699039
## mcg~~alm1   0.396978631
## mcg~~alm2   0.167086173
## gvh~~lip    0.043804474
## gvh~~chg    0.018466013
## gvh~~aac    0.069824280
## gvh~~alm1   0.173491832
## gvh~~alm2  -0.120199249
## lip~~chg    0.311951010
## lip~~aac    0.070190017
## lip~~alm1   0.095787600
## lip~~alm2  -0.057570478
## chg~~aac   -0.044725866
## chg~~alm1  -0.007653676
## chg~~alm2  -0.052188439
## aac~~alm1   0.279482496
## aac~~alm2   0.252674569
## alm1~~alm2  0.809323614
##                   2.5%
## mcg~~gvh    0.3796662188
## mcg~~lip    0.0831673262
## mcg~~chg    0.0013128903
## mcg~~aac    0.1200317074
## mcg~~alm1   0.2996162484
## mcg~~alm2   0.0630608363
## gvh~~lip   -0.0278036088
## gvh~~chg   -0.0008124890
## gvh~~aac   -0.0570292074
## gvh~~alm1   0.0874995183
## gvh~~alm2  -0.2122982473
## lip~~chg    0.0004492448
## lip~~aac   -0.0294391271
## lip~~alm1   0.0334965063
## lip~~alm2  -0.1926295017
## chg~~aac   -0.0888691927
## chg~~alm1  -0.0170930825
## chg~~alm2  -0.1032176344
## aac~~alm1   0.1781111423
## aac~~alm2   0.1411281868
## alm1~~alm2  0.7472103284
##                  97.5%
## mcg~~gvh    0.523983126
## mcg~~lip    0.209321514
## mcg~~chg    0.138553322
## mcg~~aac    0.316874083
## mcg~~alm1   0.486157706
## mcg~~alm2   0.267519923
## gvh~~lip    0.114965270
## gvh~~chg    0.037730795
## gvh~~aac    0.194459057
## gvh~~alm1   0.256917642
## gvh~~alm2  -0.025983696
## lip~~chg    0.568268205
## lip~~aac    0.168438263
```

```
## lip~~alm1   0.157337392
## lip~~alm2   0.079629209
## chg~~aac   -0.000407188
## chg~~alm1   0.001787094
## chg~~alm2  -0.000885243
## aac~~alm1   0.374983946
## aac~~alm2   0.357885744
## alm1~~alm2  0.857420400
##                 P-value
## mcg~~gvh   4.507778e-26
## mcg~~lip   7.088069e-06
## mcg~~chg   4.580130e-02
## mcg~~aac   2.256324e-05
## mcg~~alm1  1.172855e-13
## mcg~~alm2  1.731474e-03
## gvh~~lip   2.304745e-01
## gvh~~chg   6.046731e-02
## gvh~~aac   2.805471e-01
## gvh~~alm1  8.708098e-05
## gvh~~alm2  1.251336e-02
## lip~~chg   4.968147e-02
## lip~~aac   1.671642e-01
## lip~~alm1  2.616264e-03
## lip~~alm2  4.111118e-01
## chg~~aac   4.793328e-02
## chg~~alm1  1.120700e-01
## chg~~alm2  4.617915e-02
## aac~~alm1  1.480885e-07
## aac~~alm2  1.321152e-05
## alm1~~alm2 5.060937e-44
##
##          mcg      gvh
## gvh  0.45481
## lip  0.14684  0.04380
## chg  0.07027  0.01847
## aac  0.22070  0.06982
## alm1 0.39698  0.17349
## alm2 0.16709 -0.12020
##           lip       chg
## gvh
## lip
## chg   0.31195
## aac   0.07019 -0.044726
## alm1  0.09579 -0.007654
## alm2 -0.05757 -0.052188
##         aac   alm1
## gvh
## lip
## chg
## aac
## alm1 0.2795
## alm2 0.2527 0.8093
```

#Reshaping the data frame:

Drop the id feature altogether. As it is located in the first column, we can exclude it by making a copy of the wbcd data frame without column 1:

Classification models require the data be numeric and for SVM, the column does not add value.

```
ecoli_df <- ecoli_df[-1]
```

convert "cld" field as the class factor.

The Class Distribution ("cld") is the outcome we hope to classify. This feature indicates the protein classification.

Many R machine learning classifiers require that the target feature is coded as a factor, so we will need to recode the cld variable.

```
ecoli_df$cld <- as.factor(ecoli_df$cld)
ecoli_df
```

```
## # A tibble: 336 x 8
##      mcg   gvh   lip   chg   aac  alm1  alm2 cld
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>
##  1 0.49  0.290  0.48   0.5  0.56  0.24 0.35  cp
##  2 0.07  0.4    0.48   0.5  0.54  0.35 0.44  cp
##  3 0.56  0.4    0.48   0.5  0.49  0.37 0.46  cp
##  4 0.59  0.49   0.48   0.5  0.52  0.45 0.36  cp
##  5 0.23  0.32   0.48   0.5  0.55  0.25 0.35  cp
##  6 0.67  0.39   0.48   0.5  0.36  0.38 0.46  cp
##  7 0.290 0.28   0.48   0.5  0.44  0.23 0.34  cp
##  8 0.21  0.34   0.48   0.5  0.51  0.28 0.39  cp
##  9 0.2   0.44   0.48   0.5  0.46  0.51 0.570 cp
## 10 0.42  0.4    0.48   0.5  0.56  0.18 0.3   cp
## # ... with 326 more rows
```

```
str(ecoli_df)
```

```
## tibble [336 x 8] (S3: tbl_df/tbl/data.frame)
##  $ mcg : num [1:336] 0.49 0.07 0.56 0.59 0.23 0.67 0.29 0.21 0.2 0.42 ...
##  $ gvh : num [1:336] 0.29 0.4 0.4 0.49 0.32 0.39 0.28 0.34 0.44 0.4 ...
##  $ lip : num [1:336] 0.48 0.48 0.48 0.48 0.48 0.48 0.48 0.48 0.48 0.48 ...
##  $ chg : num [1:336] 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
##  $ aac : num [1:336] 0.56 0.54 0.49 0.52 0.55 0.36 0.44 0.51 0.46 0.56 ...
##  $ alm1: num [1:336] 0.24 0.35 0.37 0.45 0.25 0.38 0.23 0.28 0.51 0.18 ...
##  $ alm2: num [1:336] 0.35 0.44 0.46 0.36 0.35 0.46 0.34 0.39 0.57 0.3 ...
##  $ cld : Factor w/ 8 levels "cp","im","imL",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
#summary(ecoli_df)
```

#SVM - Standardization:

Scale each feature to a fairly small interval.

We Apply normalization to rescale the features to a standard range of values.

We use the Min-Max Normalization:

```
normalize <- function(x) {
return((x - min(x)) / (max(x) - min(x)))
}
```

After executing this code, our normalize() function can be applied to every column in the ecoli data frame using the lapply() function.

```
ecoli_norm <- as.data.frame(lapply(ecoli_df[1:7], normalize))
```

To confirm that the normalization worked, we can see that the minimum and maximum strength are now 0 and 1, respectively. To confirm that the transformation was applied correctly, let's look at the summary statistics:

```
summary(ecoli_norm)
```

```
##       mcg              gvh              lip               chg
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.3820   1st Qu.:0.2857   1st Qu.:0.00000   1st Qu.:0.000000
##  Median :0.5618   Median :0.3690   Median :0.00000   Median :0.000000
##  Mean   :0.5619   Mean   :0.4048   Mean   :0.02976   Mean   :0.002976
##  3rd Qu.:0.7444   3rd Qu.:0.4881   3rd Qu.:0.00000   3rd Qu.:0.000000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.000000
##       aac              alm1             alm2
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.4773   1st Qu.:0.3093   1st Qu.:0.3535
##  Median :0.5625   Median :0.4381   Median :0.4343
##  Mean   :0.5682   Mean   :0.4847   Mean   :0.5048
##  3rd Qu.:0.6477   3rd Qu.:0.7010   3rd Qu.:0.7172
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

Each of the columns now have values that range from 0 to 1. The function appears to be working correctly. Despite the fact that the values in the vectors varied, after normalization, they all appear exactly the same.

```
str(ecoli_norm)
```

```
## 'data.frame':    336 obs. of  7 variables:
##  $ mcg : num  0.5506 0.0787 0.6292 0.6629 0.2584 ...
##  $ gvh : num  0.155 0.286 0.286 0.393 0.19 ...
##  $ lip : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ chg : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ aac : num  0.636 0.614 0.557 0.591 0.625 ...
##  $ alm1: num  0.216 0.33 0.351 0.433 0.227 ...
##  $ alm2: num  0.354 0.444 0.465 0.364 0.354 ...
```

Checking the variables, we see that lip and chg are constants. The "lip" column is 0.48 except for 10 datasets that are 1.00. The "chg" variable is 0.5 except for 1 dataset that is 1.0. When normalized, the two columns are 0. Since they don't add any meaningful information, we drop the two columns.

```
drop <- c('lip', 'chg')
ecoli_norm <- ecoli_norm[,!(names(ecoli_norm) %in% drop)]
```

View the normalized data frame.

```
view(ecoli_norm)
```

The eighth column "cld" was dropped from this data frame. We add it back by using the following code:

Add back "cld" (class distribiution) column:

```
ecoli_norm$cld <- ecoli_df$cld
```

Verify the first few rows of the data frame:

```
print.data.frame(head(ecoli_norm))
```

```
##          mcg       gvh       aac      alm1      alm2 cld
## 1 0.55056180 0.1547619 0.6363636 0.2164948 0.3535354  cp
## 2 0.07865169 0.2857143 0.6136364 0.3298969 0.4444444  cp
## 3 0.62921348 0.2857143 0.5568182 0.3505155 0.4646465  cp
## 4 0.66292135 0.3928571 0.5909091 0.4329897 0.3636364  cp
## 5 0.25842697 0.1904762 0.6250000 0.2268041 0.3535354  cp
## 6 0.75280899 0.2738095 0.4090909 0.3608247 0.4646465  cp
```

Verify the last few rows of the data frame:

```
print.data.frame(tail(ecoli_norm))
```

```
##           mcg       gvh       aac      alm1      alm2 cld
## 331 0.4831461 0.5119048 0.5909091 0.4742268 0.5656566  pp
## 332 0.8314607 0.4761905 0.5340909 0.6701031 0.3030303  pp
## 333 0.7977528 0.4880952 0.5454545 0.3298969 0.3232323  pp
## 334 0.6853933 0.5238095 0.5000000 0.3711340 0.3838384  pp
## 335 0.6629213 0.5357143 0.4772727 0.4020619 0.3737374  pp
## 336 0.8314607 0.6904762 0.3522727 0.5154639 0.5252525  pp
```

View the structure of the data frame:

```
str(ecoli_norm)
```

```
## 'data.frame':    336 obs. of  6 variables:
##  $ mcg : num  0.5506 0.0787 0.6292 0.6629 0.2584 ...
##  $ gvh : num  0.155 0.286 0.286 0.393 0.19 ...
##  $ aac : num  0.636 0.614 0.557 0.591 0.625 ...
##  $ alm1: num  0.216 0.33 0.351 0.433 0.227 ...
##  $ alm2: num  0.354 0.444 0.465 0.364 0.354 ...
##  $ cld : Factor w/ 8 levels "cp","im","imL",..: 1 1 1 1 1 1 1 1 1 1 ...
```

#Data preparation – creating training and test datasets.

Split into training and testing, ratio of 70:30.

```r
set.seed(101)
train_index <- sample(1:nrow(ecoli_norm), 0.7 * nrow(ecoli_norm))
ecolinorm_train <- ecoli_norm[train_index, ]
ecolinorm_test <- ecoli_norm[-train_index, ]
```

View the dimensions of the split data sets:

```r
dim(ecolinorm_train)
```

```
## [1] 235   6
```

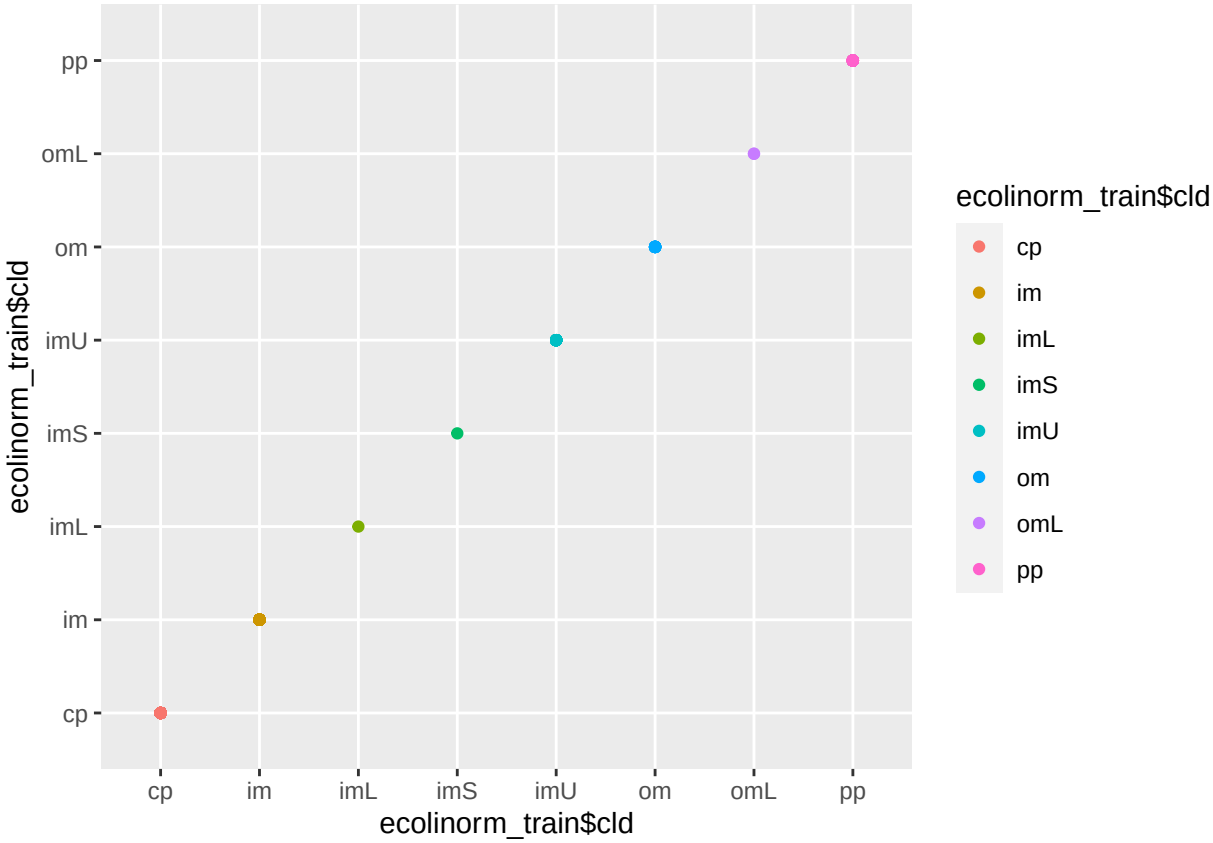
```r
dim(ecolinorm_test)
```

```
## [1] 101   6
```

```r
str(ecolinorm_train)
```

```
## 'data.frame':    235 obs. of  6 variables:
##  $ mcg : num  0.775 0.708 0.348 0.348 0.719 ...
##  $ gvh : num  0.595 0.583 0.369 0.238 0.595 ...
##  $ aac : num  0.466 0.443 0.33 0.659 0.466 ...
##  $ alm1: num  0.485 0.423 0.258 0.938 0.371 ...
##  $ alm2: num  0.253 0.354 0.394 0.949 0.202 ...
##  $ cld : Factor w/ 8 levels "cp","im","imL",..: 8 8 1 2 8 8 5 1 1 8 ...
```

```r
ecolinormNew_train <- tibble::enframe(name = NULL, ecolinorm_train$cld)
ggplot(ecolinormNew_train, aes(x = ecolinorm_train$cld, y = ecolinorm_train$cld, col = ecolinorm_train$
```



#Training a svm LINEAR model on the data:

#svm linear model:

```
ecoli.linmodel <- svm(cld ~ ., data = ecolinorm_train, kernel='linear', cost=1, scale=FALSE)
```

```
summary(ecoli.linmodel)
```

```
##
## Call:
## svm(formula = cld ~ ., data = ecolinorm_train, kernel = "linear",
##     cost = 1, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  147
##
##  ( 31 33 39 25 15 2 1 1 )
##
##
## Number of Classes:  8
##
```

```
## Levels:
##  cp im imL imS imU om omL pp
```

#Evaluating svm LINEAR model performance:

```
str(ecolinorm_test$cld)
```

```
##  Factor w/ 8 levels "cp","im","imL",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
length(ecolinorm_test$cld)
```

```
## [1] 101
```

```
str(ecolinorm_test)
```

```
## 'data.frame':    101 obs. of  6 variables:
##  $ mcg : num  0.663 0.236 0.281 0.258 0.573 ...
##  $ gvh : num  0.393 0.214 0.286 0.286 0.452 ...
##  $ aac : num  0.591 0.58 0.523 0.443 0.466 ...
##  $ alm1: num  0.433 0.258 0.423 0.258 0.32 ...
##  $ alm2: num  0.364 0.394 0.525 0.384 0.434 ...
##  $ cld : Factor w/ 8 levels "cp","im","imL",..: 1 1 1 1 1 1 1 1 1 1 ...
```

#PREDICT svm LINEAR MODEL: #svm linear model predict.

```
ecoli.linmodelpred <- predict(ecoli.linmodel, ecolinorm_test,  type="C-classification")
```

Add Class = to avoid error in confusion matrix: Error in table(data, reference, dnn = dnn, ...) : all arguments must have the same length.

```
summary(ecoli.linmodelpred)
```

```
##  cp  im imL imS imU  om omL  pp
##  46  33   0   0   0   4   0  18
```

Use the table function to generate a classification table with the prediction result and labels of the testing data set.

```
ecoli.linmodeltable <- table(ecoli.linmodelpred, ecolinorm_test$cld)

ecoli.linmodeltable
```

```
##
## ecoli.linmodelpred cp im imL imS imU om omL pp
##                 cp  44  1   0   0   0  0   0  1
##                 im   1 20   1   1  10  0   0  0
##                 imL  0  0   0   0   0  0   0  0
##                 imS  0  0   0   0   0  0   0  0
##                 imU  0  0   0   0   0  0   0  0
##                 om   0  0   0   0   0  4   0  0
##                 omL  0  0   0   0   0  0   0  0
##                 pp   0  1   0   0   0  1   3 13
```

```r
classAgreement(ecoli.linmodeltable)
```

```
## $diag
## [1] 0.8019802
##
## $kappa
## [1] 0.7168092
##
## $rand
## [1] 0.889901
##
## $crand
## [1] 0.7423167
```

Confusion Matrix of the linear predicted model.

```r
confusionMatrix(ecoli.linmodelpred, ecolinorm_test$cld)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction cp im imL imS imU om omL pp
##        cp  44  1   0   0   0  0   0  1
##        im   1 20   1   1  10  0   0  0
##        imL  0  0   0   0   0  0   0  0
##        imS  0  0   0   0   0  0   0  0
##        imU  0  0   0   0   0  0   0  0
##        om   0  0   0   0   0  4   0  0
##        omL  0  0   0   0   0  0   0  0
##        pp   0  1   0   0   0  1   3 13
##
## Overall Statistics
##
##                Accuracy : 0.802
##                  95% CI : (0.7109, 0.8746)
##     No Information Rate : 0.4455
##     P-Value [Acc > NIR] : 2.268e-13
##
##                   Kappa : 0.7168
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: cp Class: im Class: imL Class: imS Class: imU
## Sensitivity             0.9778    0.9091   0.000000   0.000000    0.00000
## Specificity             0.9643    0.8354   1.000000   1.000000    1.00000
## Pos Pred Value          0.9565    0.6061        NaN        NaN        NaN
## Neg Pred Value          0.9818    0.9706   0.990099   0.990099    0.90099
## Prevalence              0.4455    0.2178   0.009901   0.009901    0.09901
## Detection Rate          0.4356    0.1980   0.000000   0.000000    0.00000
## Detection Prevalence    0.4554    0.3267   0.000000   0.000000    0.00000
```

```
## Balanced Accuracy       0.9710    0.8723   0.500000   0.500000    0.50000
##                       Class: om Class: omL Class: pp
## Sensitivity              0.8000     0.0000    0.9286
## Specificity              1.0000     1.0000    0.9425
## Pos Pred Value           1.0000        NaN    0.7222
## Neg Pred Value           0.9897     0.9703    0.9880
## Prevalence               0.0495     0.0297    0.1386
## Detection Rate           0.0396     0.0000    0.1287
## Detection Prevalence     0.0396     0.0000    0.1782
## Balanced Accuracy        0.9000     0.5000    0.9356
```

```
str(ecoli.linmodelpred)
```

```
##  Factor w/ 8 levels "cp","im","imL",..: 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "names")= chr [1:101] "4" "8" "16" "19" ...
```

```
length(ecoli.linmodelpred)
```

```
## [1] 101
```

#(Load the e1071 package - already done in the beginnig)

#Improving model performance. We use the radial kernel (similar to Gaussian RBF kernel:

#Training a svm RADIAL model on the data: Train the support vector machine using the svm function with trainset (ecolinorm_train) as the input dataset, and use cld as the classification category.

#svm RADIAL MODEL: #svm radial model (with a cost of 1).

```
ecoli.radmodel <- svm(cld ~., data = ecolinorm_train, kernel="radial", cost=1, gamma = 1/ncol(ecolinorm
```

```
length(ecoli.radmodel)
```

```
## [1] 30
```

```
length(ecolinorm_train)
```

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

Obtain overall information about the built model with summary.

```
summary(ecoli.radmodel)
```

```
##
## Call:
## svm(formula = cld ~ ., data = ecolinorm_train, kernel = "radial",
##     cost = 1, gamma = 1/ncol(ecolinorm_train))
##
##
## Parameters:
##    SVM-Type:  C-classification
```

```
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  123
##
##  ( 24 24 37 23 11 2 1 1 )
##
##
## Number of Classes:  8
##
## Levels:
##  cp im imL imS imU om omL pp
```

#Evaluating svm RADIAL model performance:

#PREDICT svm RADIAL MODEL: #svm radial model predict. Predict the label of the testing dataset based on the fitted SVM and attributes of the testing dataset.

```
#aba.radmodelpred <- predict(aba.radmodel, abadataNew_test[, -1],

ecoli.radmodelpred <- predict(ecoli.radmodel, ecolinorm_test, type="C-classification")
```

```
summary(ecoli.radmodelpred)
```

```
##  cp  im imL imS imU  om omL  pp
##  47  22   0   0  11   6   0  15
```

Use the table function to generate a classification table with the prediction result and labels of the testing data set.

```
ecoli.radmodeltable <- table(ecoli.radmodelpred, ecolinorm_test$cld)

ecoli.radmodeltable
```

```
##
## ecoli.radmodelpred cp im imL imS imU om omL pp
##                 cp  45  1   0   0   0  0   0  1
##                 im   0 17   0   1   4  0   0  0
##                 imL  0  0   0   0   0  0   0  0
##                 imS  0  0   0   0   0  0   0  0
##                 imU  0  4   1   0   6  0   0  0
##                 om   0  0   0   0   0  5   0  1
##                 omL  0  0   0   0   0  0   0  0
##                 pp   0  0   0   0   0  0   3 12
```

Use classAgreement to calculate coefficients compared to the classification agreement.

```
classAgreement(ecoli.radmodeltable)
```

```
## $diag
## [1] 0.8415842
##
```

```
## $kappa
## [1] 0.7771649
##
## $rand
## [1] 0.9221782
##
## $crand
## [1] 0.8083106
```

Use confusionMatrix to measure the prediction performance based on the classification table.

```
confusionMatrix(ecoli.radmodelpred, ecolinorm_test$cld)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction cp im imL imS imU om omL pp
##        cp  45  1   0   0   0  0   0  1
##        im   0 17   0   1   4  0   0  0
##        imL  0  0   0   0   0  0   0  0
##        imS  0  0   0   0   0  0   0  0
##        imU  0  4   1   0   6  0   0  0
##        om   0  0   0   0   0  5   0  1
##        omL  0  0   0   0   0  0   0  0
##        pp   0  0   0   0   0  0   3 12
##
## Overall Statistics
##
##                Accuracy : 0.8416
##                  95% CI : (0.7555, 0.9067)
##     No Information Rate : 0.4455
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7772
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: cp Class: im Class: imL Class: imS Class: imU
## Sensitivity             1.0000    0.7727   0.000000   0.000000    0.60000
## Specificity             0.9643    0.9367   1.000000   1.000000    0.94505
## Pos Pred Value          0.9574    0.7727        NaN        NaN    0.54545
## Neg Pred Value          1.0000    0.9367   0.990099   0.990099    0.95556
## Prevalence              0.4455    0.2178   0.009901   0.009901    0.09901
## Detection Rate          0.4455    0.1683   0.000000   0.000000    0.05941
## Detection Prevalence    0.4653    0.2178   0.000000   0.000000    0.10891
## Balanced Accuracy       0.9821    0.8547   0.500000   0.500000    0.77253
##                      Class: om Class: omL Class: pp
## Sensitivity            1.00000     0.0000    0.8571
## Specificity            0.98958     1.0000    0.9655
## Pos Pred Value         0.83333        NaN    0.8000
## Neg Pred Value         1.00000     0.9703    0.9767
```

```
## Prevalence             0.04950     0.0297    0.1386
## Detection Rate         0.04950     0.0000    0.1188
## Detection Prevalence   0.05941     0.0000    0.1485
## Balanced Accuracy      0.99479     0.5000    0.9113
```

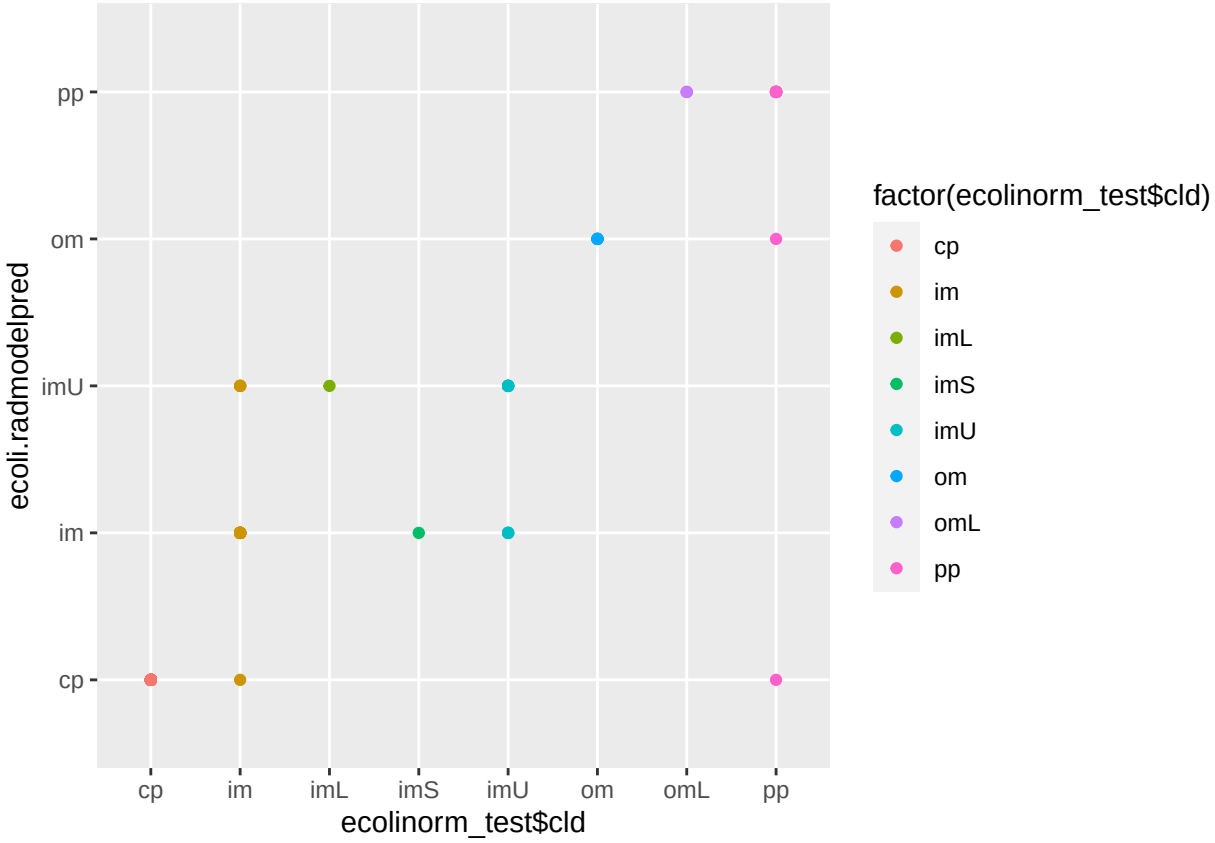
```r
length(ecolinorm_test$cld)
```

```
## [1] 101
```

```r
length(ecoli.radmodelpred)
```

```
## [1] 101
```

```r
ggplot(NULL, aes(x = ecolinorm_test$cld, y = ecoli.radmodelpred,
                 col = factor(ecolinorm_test$cld))) + geom_point()
```



#k-fold Cross Validation for svm model:

#svm TUNED MODEL: We use tune.svm to perform the 10-fold cross-validation and obtain the optimum classification model.

```r
tuned_svm <- tune.svm(cld~., data = ecolinorm_train, gamma = 10^(-6:-1), cost = 10^(1:2), tunecontrol=tu
```

```
#t_svm <- tune.svm(cld~., data = ecolinorm_train, gamma = 10^(-5:-1), cost #= 10^(-3:1))

#t_svm <- tune.svm(cld~., data = ecolinorm_train, gamma = 10^(-5:-1), cost #= 10^(-3:1), scale = FALSE,

#best.svm(cld~., data = ecolinorm_train, tunecontrol = #tune.control(cross=10))
```

Summary of the tunned SVM model.

```
summary(tuned_svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
##   0.01   10
##
## - best performance: 0.136413
##
## - Detailed performance results:
##    gamma cost     error dispersion
## 1  1e-06   10 0.5840580 0.10310077
## 2  1e-05   10 0.5840580 0.10310077
## 3  1e-04   10 0.5757246 0.11518093
## 4  1e-03   10 0.1923913 0.09592444
## 5  1e-02   10 0.1364130 0.06729801
## 6  1e-01   10 0.1409420 0.06897711
## 7  1e-06  100 0.5840580 0.10310077
## 8  1e-05  100 0.5757246 0.11518093
## 9  1e-04  100 0.1923913 0.09592444
## 10 1e-03  100 0.1364130 0.06729801
## 11 1e-02  100 0.1365942 0.06777934
## 12 1e-01  100 0.1668478 0.09836306
```

Structure of the tuned model (gives more statistics):

```
str(tuned_svm)
```

```
## List of 8
##  $ best.parameters :'data.frame':    1 obs. of  2 variables:
##   ..$ gamma: num 0.01
##   ..$ cost : num 10
##   ..- attr(*, "out.attrs")=List of 2
##   .. ..$ dim     : Named int [1:2] 6 2
##   .. .. ..- attr(*, "names")= chr [1:2] "gamma" "cost"
##   .. ..$ dimnames:List of 2
##   .. .. ..$ gamma: chr [1:6] "gamma=1e-06" "gamma=1e-05" "gamma=1e-04" "gamma=1e-03" ...
##   .. .. ..$ cost : chr [1:2] "cost= 10" "cost=100"
##  $ best.performance: num 0.136
##  $ method          : chr "svm"
```

```
##  $ nparcomb        : int 12
##  $ train.ind       :List of 10
##   ..$ (0.766,24.4]: int [1:211] 79 13 15 104 127 9 39 221 68 35 ...
##   ..$ (24.4,47.8] : int [1:212] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (47.8,71.2] : int [1:211] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (71.2,94.6] : int [1:212] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (94.6,118]  : int [1:211] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (118,141]   : int [1:212] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (141,165]   : int [1:212] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (165,188]   : int [1:211] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (188,212]   : int [1:212] 225 111 143 64 119 26 110 169 38 106 ...
##   ..$ (212,235]   : int [1:211] 225 111 143 64 119 26 110 169 38 106 ...
##   ..- attr(*, "dim")= int 10
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ : chr [1:10] "(0.766,24.4]" "(24.4,47.8]" "(47.8,71.2]" "(71.2,94.6]" ...
##  $ sampling        : chr "10-fold cross validation"
##  $ performances    :'data.frame':    12 obs. of  4 variables:
##   ..$ gamma     : num [1:12] 1e-06 1e-05 1e-04 1e-03 1e-02 1e-01 1e-06 1e-05 1e-04 1e-03 ...
##   ..$ cost      : num [1:12] 10 10 10 10 10 10 100 100 100 100 ...
##   ..$ error     : num [1:12] 0.584 0.584 0.576 0.192 0.136 ...
##   ..$ dispersion: num [1:12] 0.1031 0.1031 0.1152 0.0959 0.0673 ...
##  $ best.model      :List of 30
##   ..$ call          : language best.svm(x = cld ~ ., data = ecolinorm_train, gamma = 10^(-6:-1), co
##   ..$ type          : num 0
##   ..$ kernel        : num 2
##   ..$ cost          : num 10
##   ..$ degree        : num 3
##   ..$ gamma         : num 0.01
##   ..$ coef0         : num 0
##   ..$ nu            : num 0.5
##   ..$ epsilon       : num 0.1
##   ..$ sparse        : logi FALSE
##   ..$ scaled        : logi [1:5] TRUE TRUE TRUE TRUE TRUE
##   ..$ x.scale       :List of 2
##   .. ..$ scaled:center: Named num [1:5] 0.564 0.4 0.574 0.482 0.496
##   .. .. ..- attr(*, "names")= chr [1:5] "mcg" "gvh" "aac" "alm1" ...
##   .. ..$ scaled:scale : Named num [1:5] 0.223 0.179 0.135 0.225 0.214
##   .. .. ..- attr(*, "names")= chr [1:5] "mcg" "gvh" "aac" "alm1" ...
##   ..$ y.scale       : NULL
##   ..$ nclasses      : int 8
##   ..$ levels        : chr [1:8] "cp" "im" "imL" "imS" ...
##   ..$ tot.nSV       : int 114
##   ..$ nSV           : int [1:8] 22 24 33 23 8 2 1 1
##   ..$ labels        : int [1:8] 8 1 2 5 6 7 4 3
##   ..$ SV            : num [1:114, 1:5] 0.646 0.697 0.999 1.15 0.546 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:114] "313" "317" "315" "288" ...
##   .. .. ..$ : chr [1:5] "mcg" "gvh" "aac" "alm1" ...
##   ..$ index         : int [1:114] 2 5 6 10 15 26 43 68 83 85 ...
##   ..$ rho           : num [1:28] 0.0415 0.5494 0.6385 -0.1489 -0.954 ...
##   ..$ compprob      : logi FALSE
##   ..$ probA         : NULL
##   ..$ probB         : NULL
##   ..$ sigma         : NULL
```

```
##   ..$ coefs          : num [1:114, 1:7] 10 5.89 0 0 10 ...
##   ..$ na.action      : NULL
##   ..$ fitted         : Factor w/ 8 levels "cp","im","imL",..: 8 8 1 2 8 2 2 1 1 8 ...
##   .. ..- attr(*, "names")= chr [1:235] "329" "313" "95" "209" ...
##   ..$ decision.values: num [1:235, 1:28] 1.353 0.713 -1.984 -1.303 1 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:235] "329" "313" "95" "209" ...
##   .. .. ..$ : chr [1:28] "pp/cp" "pp/im" "pp/imU" "pp/om" ...
##   ..$ terms          :Classes 'terms', 'formula'  language cld ~ mcg + gvh + aac + alm1 + alm2
##   .. .. ..- attr(*, "variables")= language list(cld, mcg, gvh, aac, alm1, alm2)
##   .. .. ..- attr(*, "factors")= int [1:6, 1:5] 0 1 0 0 0 0 0 0 1 0 ...
##   .. .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. .. ..$ : chr [1:6] "cld" "mcg" "gvh" "aac" ...
##   .. .. .. .. ..$ : chr [1:5] "mcg" "gvh" "aac" "alm1" ...
##   .. .. ..- attr(*, "term.labels")= chr [1:5] "mcg" "gvh" "aac" "alm1" ...
##   .. .. ..- attr(*, "order")= int [1:5] 1 1 1 1 1
##   .. .. ..- attr(*, "intercept")= num 0
##   .. .. ..- attr(*, "response")= int 1
##   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. .. ..- attr(*, "predvars")= language list(cld, mcg, gvh, aac, alm1, alm2)
##   .. .. ..- attr(*, "dataClasses")= Named chr [1:6] "factor" "numeric" "numeric" "numeric" ...
##   .. .. .. ..- attr(*, "names")= chr [1:6] "cld" "mcg" "gvh" "aac" ...
##   ..- attr(*, "class")= chr [1:2] "svm.formula" "svm"
##  - attr(*, "class")= chr "tune"
```

The best tuned model has a gamma of 0.001 and a cost of 10.

Select the best model from the tuned model.

```
tuned_svmfit <- tuned_svm$best.model
```

Summary of the best tuned model.

```
summary(tuned_svmfit)
```

```
##
## Call:
## best.svm(x = cld ~ ., data = ecolinorm_train, gamma = 10^(-6:-1),
##     cost = 10^(1:2), tunecontrol = tune.control(cross = 10))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  114
##
##  ( 22 24 33 23 8 2 1 1 )
##
##
## Number of Classes:  8
##
```

```
## Levels:
##  cp im imL imS imU om omL pp
```

#TUNED svm RADIAL MODEL: create the tuned radial model.

After retrieving the best performance parameter from tuning the result, we retrain the support vector machine with the best performance parameter:

```
tuned_radmodel <- svm(cld ~ ., data = ecolinorm_train, gamma = tuned_svm$best.parameters$gamma, cost = t
```

Summary of the retrained tuned model.

```
summary(tuned_radmodel)
```

```
##
## Call:
## svm(formula = cld ~ ., data = ecolinorm_train, gamma = tuned_svm$best.parameters$gamma,
##     cost = tuned_svm$best.parameters$cost)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  114
##
##  ( 22 24 33 23 8 2 1 1 )
##
##
## Number of Classes:  8
##
## Levels:
##  cp im imL imS imU om omL pp
```

#Create the tuned predicted model: We use the predict function to predict labels based on the fitted SVM.

```
tuned_svmpred <- predict(tuned_radmodel, ecolinorm_test[, !names(ecolinorm_test) %in% c("cld")])
```

Summary information of the tuned model: Do a summary of the tuned predicted model.

```
summary(tuned_svmpred)
```

```
##  cp  im imL imS imU  om omL  pp
##  47  22   0   0  11   6   0  15
```

Create a table for the tuned predicted model and display the table.

```
tuned_svmpredtable <- table(tuned_svmpred, ecolinorm_test$cld)

tuned_svmpredtable
```

```
##
## tuned_svmpred cp im imL imS imU om omL pp
##           cp  45  1   0   0   0  0   0  1
##           im   0 17   0   1   4  0   0  0
##           imL  0  0   0   0   0  0   0  0
##           imS  0  0   0   0   0  0   0  0
##           imU  0  4   1   0   6  0   0  0
##           om   0  0   0   0   0  5   0  1
##           omL  0  0   0   0   0  0   0  0
##           pp   0  0   0   0   0  0   3 12
```

Generate a class agreement for the tunned predicted table.

```
classAgreement(tuned_svmpredtable)
```

```
## $diag
## [1] 0.8415842
##
## $kappa
## [1] 0.7771649
##
## $rand
## [1] 0.9221782
##
## $crand
## [1] 0.8083106
```

The kappa statistic adjusts the accuracy relative to the expected agreement. The kappa of 0.78 indicates that we have a "good agreement" in predicting the protein class classification and the actual values.

Generate a confusion Matrix.

```
confusionMatrix(tuned_svmpredtable)
```

```
## Confusion Matrix and Statistics
##
##
## tuned_svmpred cp im imL imS imU om omL pp
##           cp  45  1   0   0   0  0   0  1
##           im   0 17   0   1   4  0   0  0
##           imL  0  0   0   0   0  0   0  0
##           imS  0  0   0   0   0  0   0  0
##           imU  0  4   1   0   6  0   0  0
##           om   0  0   0   0   0  5   0  1
##           omL  0  0   0   0   0  0   0  0
##           pp   0  0   0   0   0  0   3 12
##
## Overall Statistics
##
##                Accuracy : 0.8416
##                  95% CI : (0.7555, 0.9067)
##     No Information Rate : 0.4455
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.7772
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: cp Class: im Class: imL Class: imS Class: imU
## Sensitivity             1.0000    0.7727   0.000000   0.000000    0.60000
## Specificity             0.9643    0.9367   1.000000   1.000000    0.94505
## Pos Pred Value          0.9574    0.7727        NaN        NaN    0.54545
## Neg Pred Value          1.0000    0.9367   0.990099   0.990099    0.95556
## Prevalence              0.4455    0.2178   0.009901   0.009901    0.09901
## Detection Rate          0.4455    0.1683   0.000000   0.000000    0.05941
## Detection Prevalence    0.4653    0.2178   0.000000   0.000000    0.10891
## Balanced Accuracy       0.9821    0.8547   0.500000   0.500000    0.77253
##                      Class: om Class: omL Class: pp
## Sensitivity            1.00000     0.0000    0.8571
## Specificity            0.98958     1.0000    0.9655
## Pos Pred Value         0.83333        NaN    0.8000
## Neg Pred Value         1.00000     0.9703    0.9767
## Prevalence             0.04950     0.0297    0.1386
## Detection Rate         0.04950     0.0000    0.1188
## Detection Prevalence   0.05941     0.0000    0.1485
## Balanced Accuracy      0.99479     0.5000    0.9113
```

```
#confusionMatrix(tuned_sumpred, ecolinorm_test$cld)
```

The model has an accuracy of 84.2% with a confidence interval of 95%.

Generate a cross table confusion Matrix if you want to view more statistics.

```
CrossTable(x = ecolinorm_test$cld, y = tuned_svmpred, prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  101
##
##
##                     | tuned_svmpred
## ecolinorm_test$cld |        cp |        im |       imU |        om |        pp | Row Total |
## -------------------|-----------|-----------|-----------|-----------|-----------|-----------|
##                 cp |        45 |         0 |         0 |         0 |         0 |        45 |
##                    |     1.000 |     0.000 |     0.000 |     0.000 |     0.000 |     0.446 |
```

```
##                    |      0.957 |      0.000 |      0.000 |      0.000 |      0.000 |            |
##                    |      0.446 |      0.000 |      0.000 |      0.000 |      0.000 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##                 im |          1 |         17 |          4 |          0 |          0 |         22 |
##                    |      0.045 |      0.773 |      0.182 |      0.000 |      0.000 |      0.218 |
##                    |      0.021 |      0.773 |      0.364 |      0.000 |      0.000 |            |
##                    |      0.010 |      0.168 |      0.040 |      0.000 |      0.000 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##                imL |          0 |          0 |          1 |          0 |          0 |          1 |
##                    |      0.000 |      0.000 |      1.000 |      0.000 |      0.000 |      0.010 |
##                    |      0.000 |      0.000 |      0.091 |      0.000 |      0.000 |            |
##                    |      0.000 |      0.000 |      0.010 |      0.000 |      0.000 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##                imS |          0 |          1 |          0 |          0 |          0 |          1 |
##                    |      0.000 |      1.000 |      0.000 |      0.000 |      0.000 |      0.010 |
##                    |      0.000 |      0.045 |      0.000 |      0.000 |      0.000 |            |
##                    |      0.000 |      0.010 |      0.000 |      0.000 |      0.000 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##                imU |          0 |          4 |          6 |          0 |          0 |         10 |
##                    |      0.000 |      0.400 |      0.600 |      0.000 |      0.000 |      0.099 |
##                    |      0.000 |      0.182 |      0.545 |      0.000 |      0.000 |            |
##                    |      0.000 |      0.040 |      0.059 |      0.000 |      0.000 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##                 om |          0 |          0 |          0 |          5 |          0 |          5 |
##                    |      0.000 |      0.000 |      0.000 |      1.000 |      0.000 |      0.050 |
##                    |      0.000 |      0.000 |      0.000 |      0.833 |      0.000 |            |
##                    |      0.000 |      0.000 |      0.000 |      0.050 |      0.000 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##                omL |          0 |          0 |          0 |          0 |          3 |          3 |
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      1.000 |      0.030 |
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.200 |            |
##                    |      0.000 |      0.000 |      0.000 |      0.000 |      0.030 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##                 pp |          1 |          0 |          0 |          1 |         12 |         14 |
##                    |      0.071 |      0.000 |      0.000 |      0.071 |      0.857 |      0.139 |
##                    |      0.021 |      0.000 |      0.000 |      0.167 |      0.800 |            |
##                    |      0.010 |      0.000 |      0.000 |      0.010 |      0.119 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##       Column Total |         47 |         22 |         11 |          6 |         15 |        101 |
##                    |      0.465 |      0.218 |      0.109 |      0.059 |      0.149 |            |
## -------------------|------------|------------|------------|------------|------------|------------|
##
##
```

class distribution in the test dataset:

```
agreement <- tuned_svmpred == ecolinorm_test$cld

table(agreement)
```

```
## agreement
## FALSE  TRUE
##    16    85
```

Using the table() function, we see that the classifier correctly identified the protein class distribution in 85 out of the 101 test records.

Agreement as a %:

```
prop.table(table(agreement))
```

```
## agreement
##     FALSE      TRUE
## 0.1584158 0.8415842
```

In percentage terms, the accuracy is about 84 percent.

#The random forest approach: We use the random forest classification method for predicting the protein localization in ecoli cells into the 8 levels (class distributions) based on their sites. We'll use the randomForest() library to classify.

First install and load the randomForest package (already done).

#Fit the random forest classifier with a training set: Separate our data into testing and training sets. We take the clean data (normalized) set we used for svm. The set.seed() function ensures that the result can be replicated. Split into training and testing, ratio of 70:30.

```
set.seed(123)
samp <- sample(nrow(ecoli_norm), 0.7 * nrow(ecoli_norm))
ecoli_trainsetrf <- ecoli_norm[samp, ]
ecoli_testsetrf <- ecoli_norm[-samp, ]
```

This will place 70% of the observations in the original dataset into train and the remaining 30% of the observations into test.

#Build our model. Fit the random forest classifier with a training set:

We use the random forest method to train a classification model. We set importance = T, which will ensure that the importance of the predictor is assessed.

```
ecoli_modelrf <- randomForest(cld ~ ., data = ecoli_trainsetrf, importance = T, keep.forest = TRUE)

ecoli_modelrf
```

```
##
## Call:
##  randomForest(formula = cld ~ ., data = ecoli_trainsetrf, importance = T,      keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 15.74%
## Confusion matrix:
##      cp im imL imS imU om omL pp class.error
## cp  101  0   0   0   0  0   0  0   0.0000000
## im    1 46   0   0   9  0   0  0   0.1785714
## imL   0  1   0   0   0  0   0  1   1.0000000
## imS   0  1   0   0   0  0   0  1   1.0000000
## imU   0 12   0   0  12  0   0  0   0.5000000
```

```
## om    0  0   0   0   0 11   1  3   0.2666667
## omL   0  0   0   0   0  2   0  0   1.0000000
## pp    3  2   0   0   0  0   0 28   0.1515152
```

Let's take a look at the model. We can see that 500 trees (the default) were built, and the model randomly sampled 2 predictors at each split. It also shows a matrix containing prediction vs actual, as well as classification error for each class.

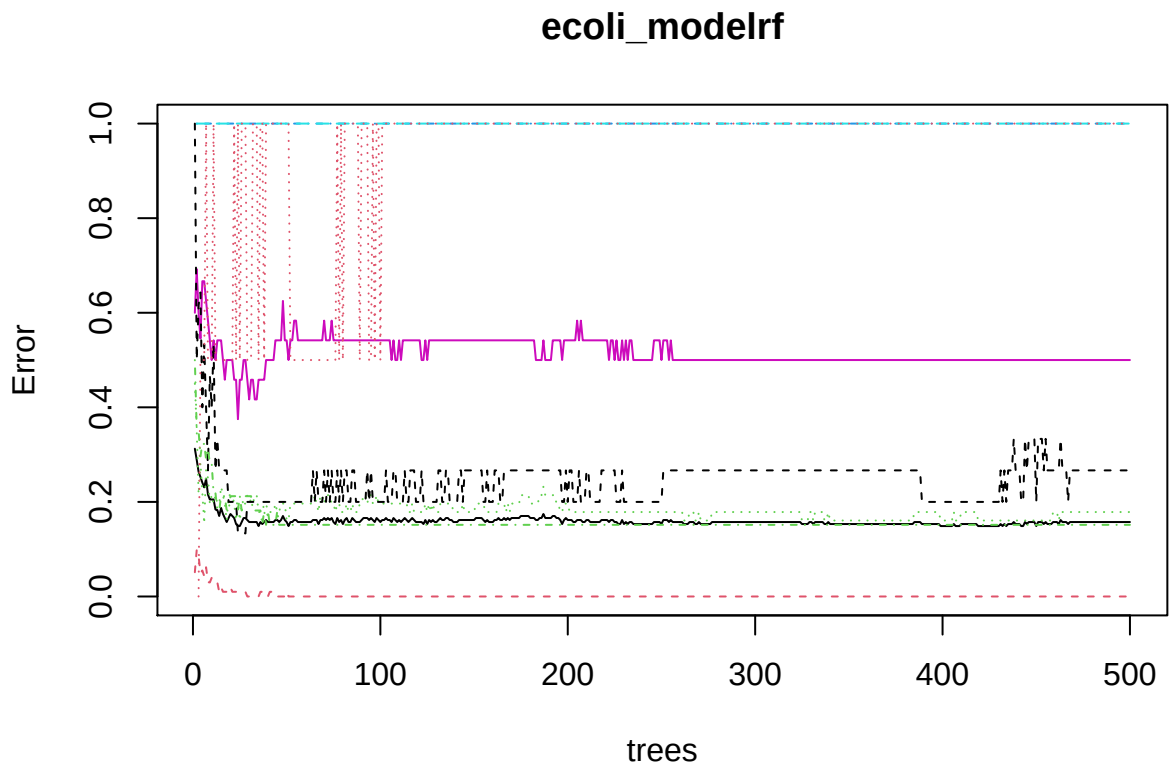
We can test the accuracy as follows:

```
(114 + 55 + 0 + 0 + 16 + 14 + 0 + 32) / nrow(ecoli_trainsetrf)
```

```
## [1] 0.9829787
```

Use the plot function to plot the mean square error of the forest object:

```
plot(ecoli_modelrf)
```



Flat line - even if run more trees, won't make a difference in the model.

Examine the importance of each attribute within the fitted classifier:

```
#importance(ecoli_modelrf)
```

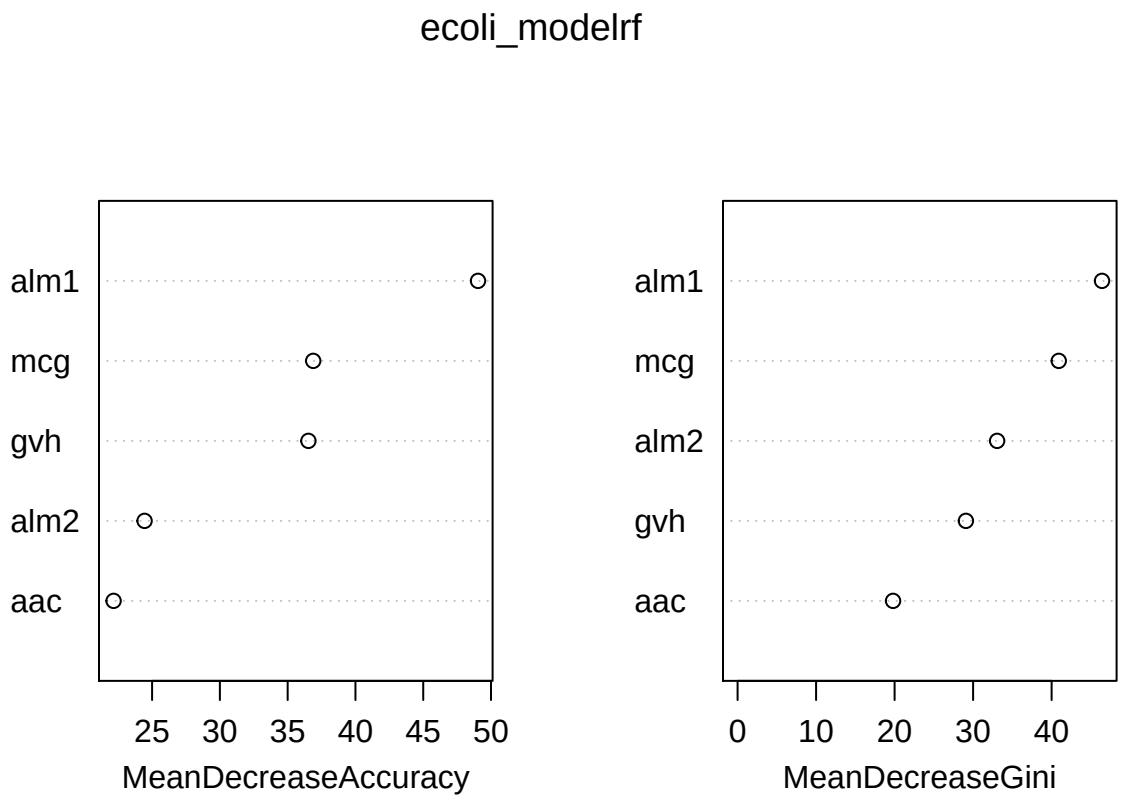
Examine the importance of each attribute within the fitted classifier:

```
varImp(ecoli_modelrf)
```

```
##             cp         im       imL       imS       imU        om       omL
## mcg  26.595751  1.0221190  0.000000  0.000000 29.736392 10.000719  4.814659
## gvh  26.987389  1.9770233  0.000000  0.000000 -0.130240  8.674469 -0.500125
## aac   8.061594 -0.6912611  0.000000  0.000000 -1.678627 32.017933  1.226586
## alm1 57.113244 26.0453340 -1.737270 -3.353721 16.817018  4.719797  2.051225
## alm2 22.718373  7.8589428 -1.001002 -1.417051 17.812548 14.607314  4.025460
##            pp
## mcg  18.88976
## gvh  36.43085
## aac  13.69222
## alm1 20.61824
## alm2 13.02525
```

Use the varImpPlot function to obtain the plot of variable importance using either mean decrease accuracy or mean decrease gini.

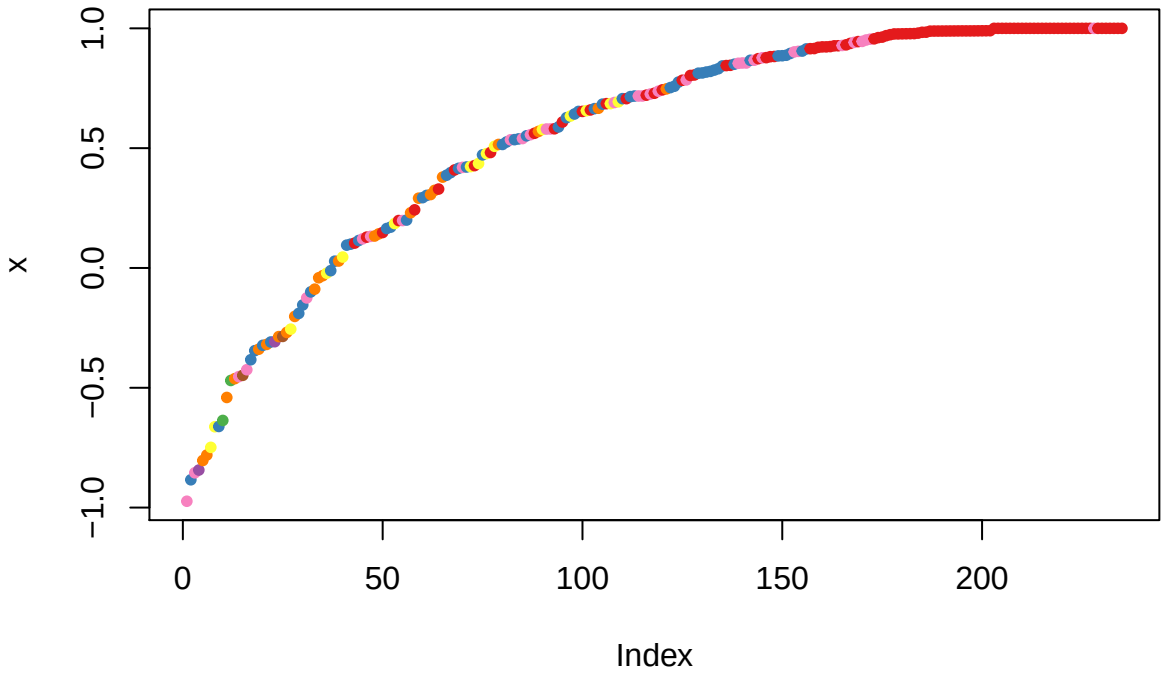
```
varImpPlot(ecoli_modelrf)
```



Use the margin function to calculate the margins of the forest object. Plot the margin cumulative distribution:

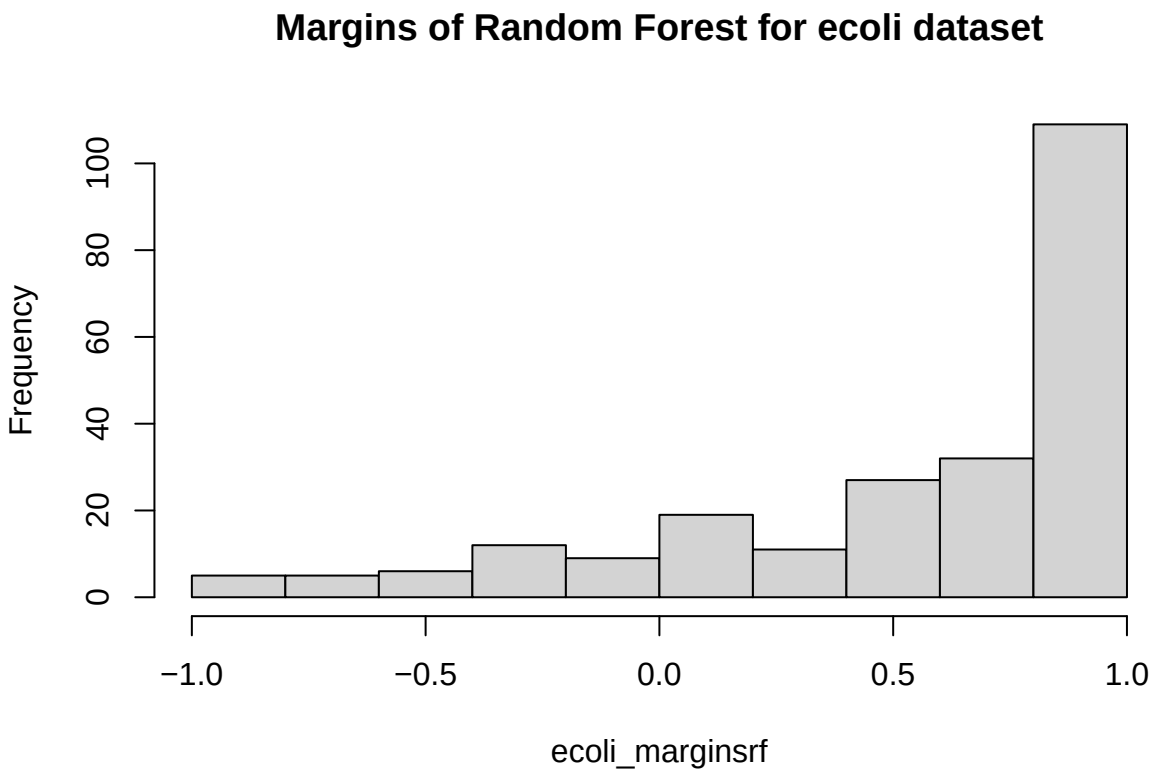
```
ecoli_marginsrf <- margin(ecoli_modelrf, ecoli_trainsetrf)

plot(ecoli_marginsrf)
```



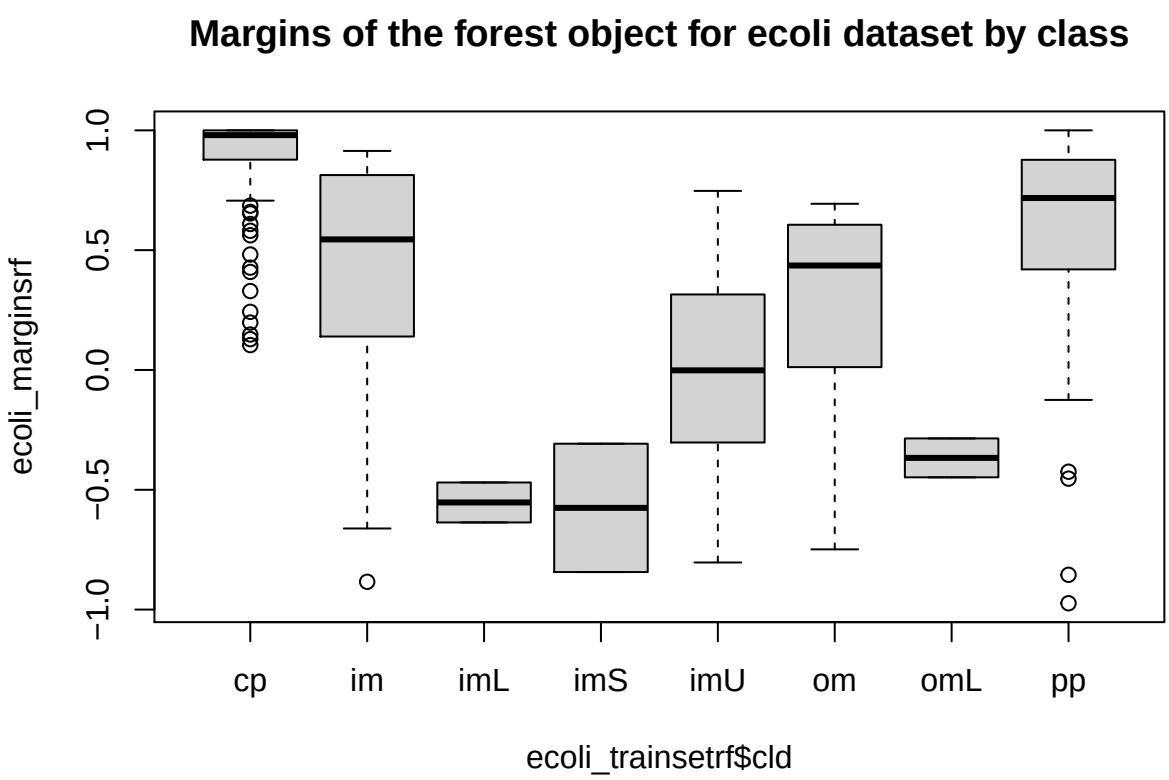
Use a histogram to visualize the margin margins of the forest object to the proportion of correctly classified observations:

```
hist(ecoli_marginsrf, main="Margins of Random Forest for ecoli dataset")
```



Use boxplot to visualize the margins of the forest object to the proportion of correctly classified observations by class:

```
boxplot(ecoli_marginsrf ~ ecoli_trainsetrf$cld, main="Margins of the forest object for ecoli dataset by
```



Let's test the model on the test data set.

```
ecoli_predictrf <- predict(ecoli_modelrf, ecoli_testsetrf)
```

Obtain the classification table:

```
table(ecoli_predictrf, ecoli_testsetrf$cld)
```

```
##
## ecoli_predictrf cp im imL imS imU om omL pp
##              cp  40  1   0   0   1  0   0  1
##              im   0 19   0   0   4  0   0  1
##              imL  0  0   0   0   0  0   0  0
##              imS  0  0   0   0   0  0   0  0
##              imU  0  1   0   0   6  0   0  0
##              om   0  0   0   0   0  5   0  0
##              omL  0  0   0   0   0  0   0  0
##              pp   2  0   0   0   0  0   3 17
```

We can test the accuracy as follows:

```
(40 + 19 + 0 + 0 + 6 + 5 + 0 + 17) / nrow(ecoli_testsetrf)
```

```
## [1] 0.8613861
```

We achieved ~86.1% accuracy with a very simple model.