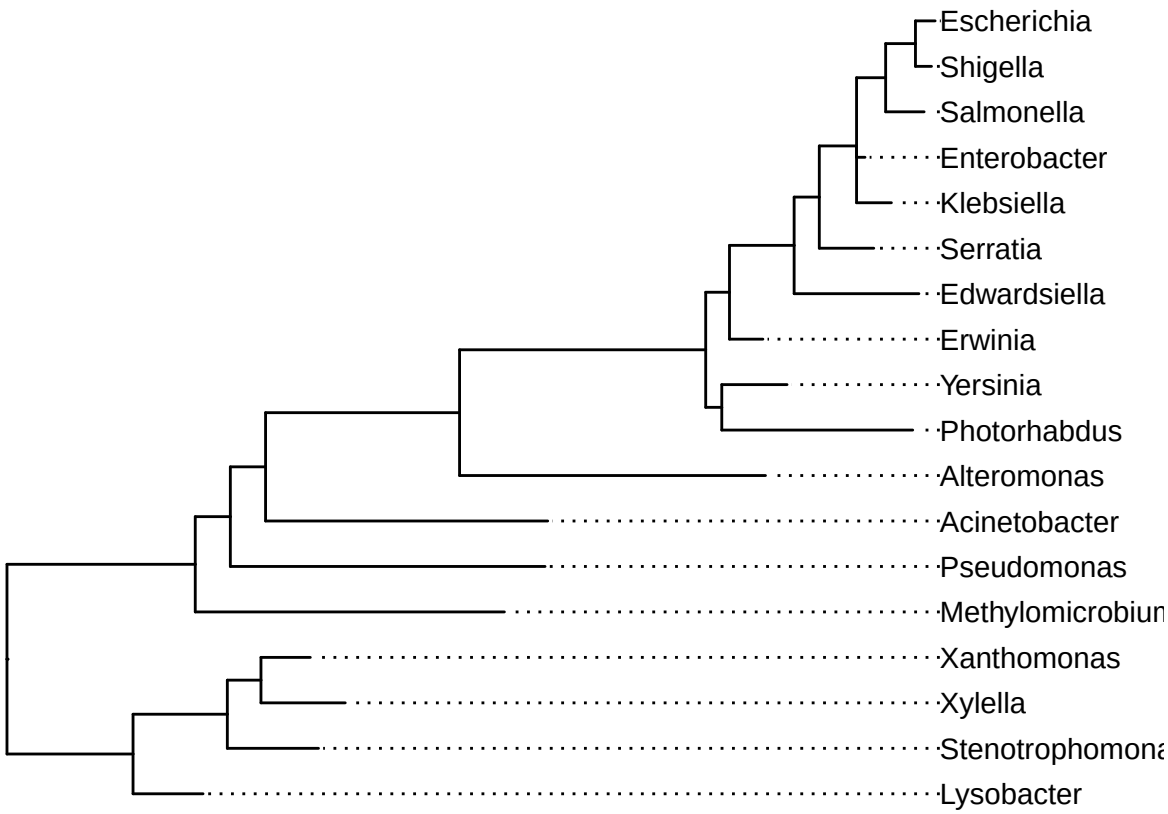
comparative_RNA-Seq_manuscript

Thomas_Nicholson

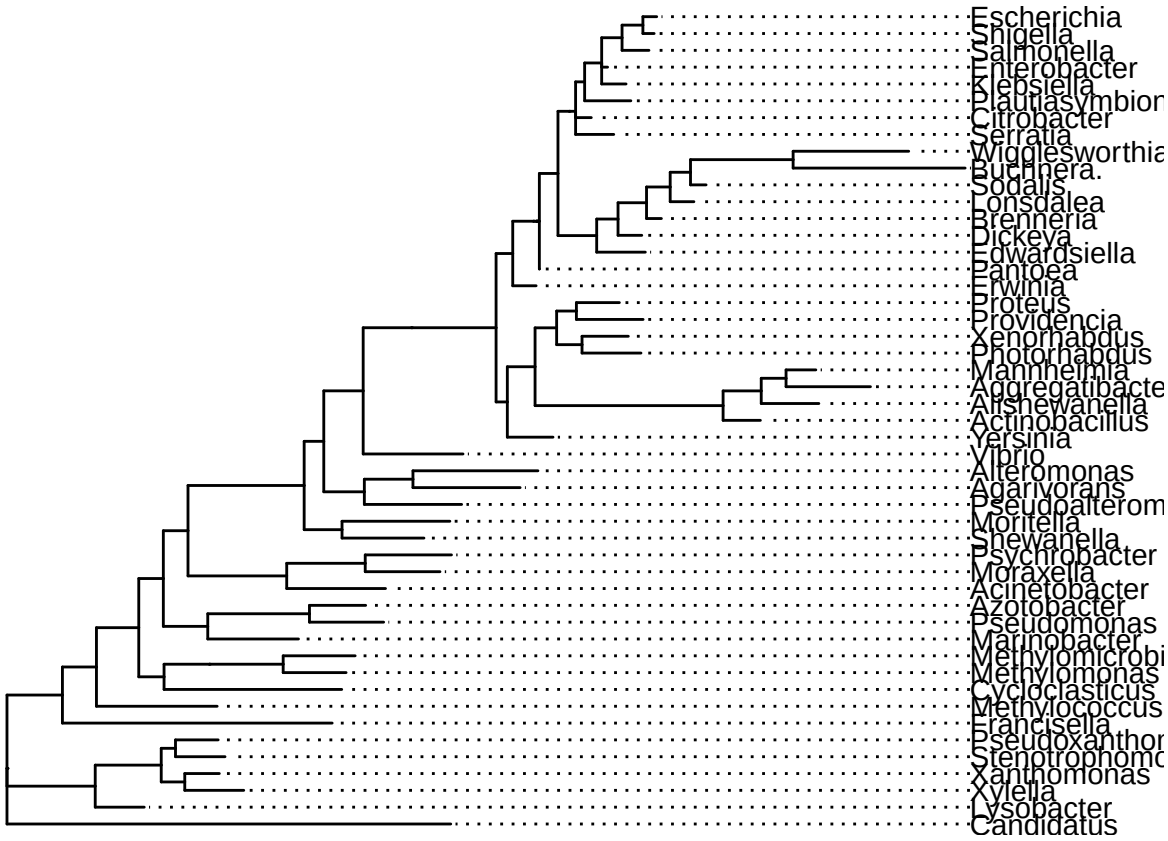
19/02/2022

UpsetR figure

```
#ML phylogenetic tree built using 16s rRNA for the Gammaproteobacteria
#The full tree was also built for all bacterial genera, but is not displayed
tree <- read.tree(paste0(data_path, 'upsetr.tree'))

  tbl_tree <- as_tibble(tree)

#list of extra Gammaproteobacteria that were
#not included in the RNA-sed analysis
to_drop <- c("Azotobacter", "Marinobacter", "Pseudoalteromonas", "Agarivorans",
             "Vibrio", "Alishewanella", "Aggregatibacter", "Mannheimia",
             "Actinobacillus", "Xenorhabdus", "Providencia", "Proteus",
             "Pantoea", "Brenneria", "Lonsdalea", "Buchnera.", "Wigglesworthia",
             "Sodalis", "Dickeya", "Citrobacter", "Plautiasymbiont",
             "Shewanella", "Moritella", "Moraxella", "Psychrobacter",
             "Methylomonas", "Cycloclasticus", "Methylococcus", "Francisella",
             "Pseudoxanthomonas", "Candidatus", "Plautia", "Methylophaga",
             "Pasteurella", "Salinivibrio")


  sub_tree <- drop.tip(tree, to_drop)

    tbl_sub_tree <- as_tibble(sub_tree)


p <- ggtree(sub_tree) +
  geom_tiplab(align = T) +
  xlim(0, 0.35)

p
```

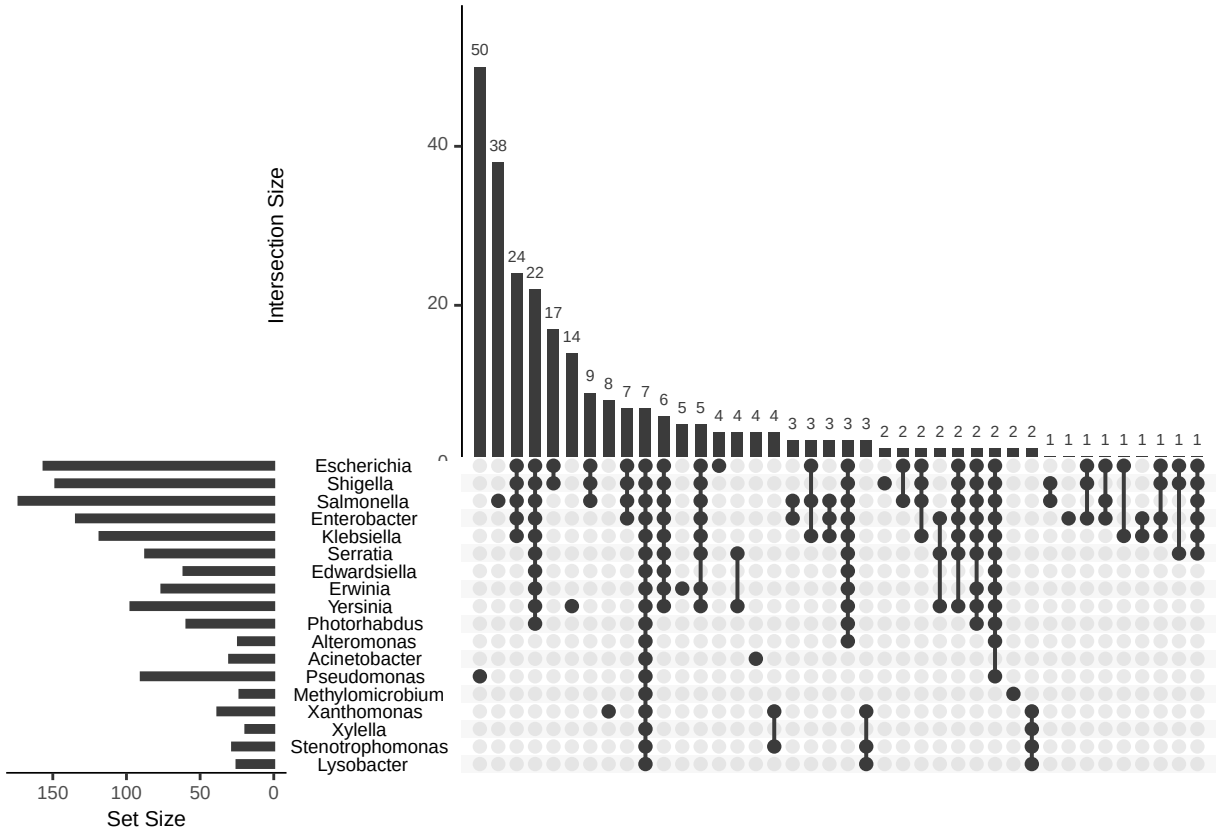



```
if(FALSE){
ggsave(filename = paste0(figure_path, "SVG/upsetr_subset_tree.svg"),
       plot = p, width = 8, height = 16)
}
p <- ggtree(tree) +
  geom_tiplab(align = T) +
  xlim(0, 0.5)

p
```

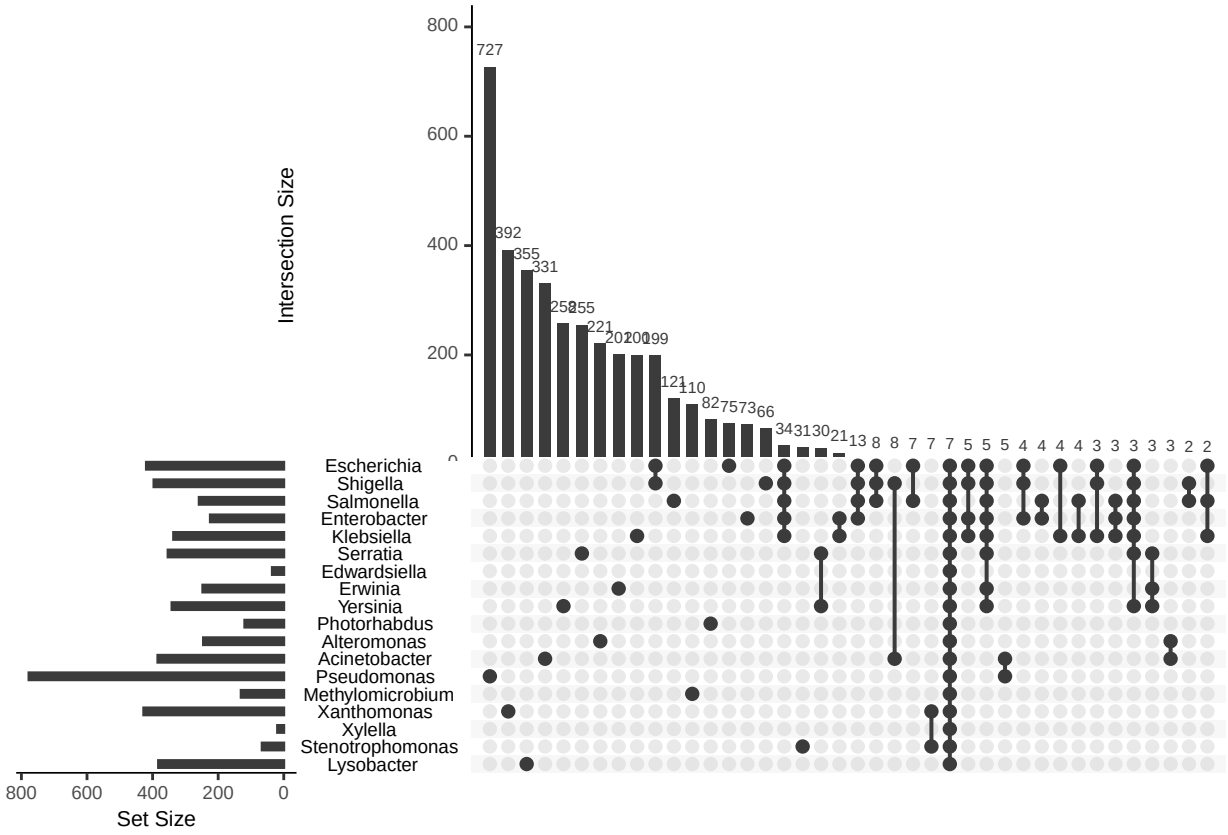



```
if(FALSE){
ggsave(filename = paste0(figure_path, "SVG/upsetr_tree.svg"),
       plot = p, width = 8, height = 16)
}
```

```
#matrix of bool values indicating for each RNA if it was found in a genus
load(paste0(data_path, "upsetSubsetPC.Rda"))
load(paste0(data_path, "upsetSubsetPredicted.Rda"))

#list of extra Gammaproteobacteria that were
#not included in the RNA-sed analysis
genera_arranged <- c("Lysobacter", "Stenotrophomonas", "Xylella", "Xanthomonas",
                     "Methylomicrobium", "Pseudomonas", "Acinetobacter",
                     "Alteromonas", "Photorhabdus", "Yersinia", "Erwinia",
                     "Edwardsiella", "Serratia", "Klebsiella", "Enterobacter",
                     "Salmonella", "Shigella", "Escherichia")


upset_pred <- select_columns_by_list(upsetSubsetPredicted, genera_arranged)
upset_pc <- select_columns_by_list(upsetSubsetPC, genera_arranged)

UpSetR::upset(upset_pc, sets = colnames(upset_pc),
              mb.ratio = c(0.55, 0.45), order.by = "freq", keep.order = T)
```



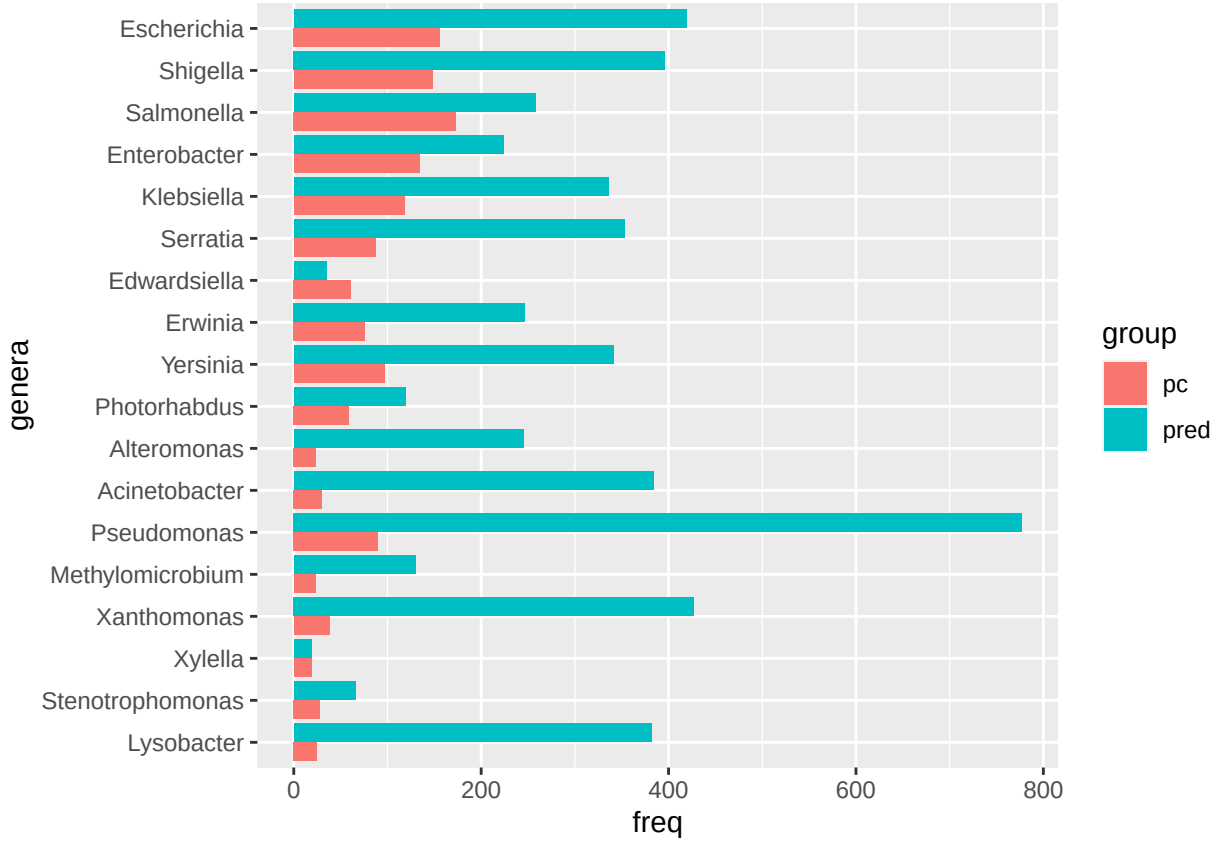

```
UpSetR::upset(upset_pred, sets = colnames(upset_pred),
              mb.ratio = c(0.55, 0.45), order.by = "freq", keep.order = T)
```




```
if(FALSE){
svglite(filename=paste0(figure_path, "SVG/upsetr_pc.svg"))
UpSetR::upset(upset_pc, sets = colnames(upset_pc),
              mb.ratio = c(0.55, 0.45), order.by = "freq", keep.order = T)
dev.off()
svglite(filename=paste0(figure_path, "SVG/upsetr_pred.svg"))
UpSetR::upset(upset_pred, sets = colnames(upset_pred),
              mb.ratio = c(0.55, 0.45), order.by = "freq", keep.order = T)
dev.off()
}
```

```
#matrix of bool values indicating for each RNA if it was found in a genus
load(paste0(data_path, "upsetSubsetPC.Rda"))
load(paste0(data_path, "upsetSubsetPredicted.Rda"))


#list of extra Gammaproteobacteria that were not included in the RNA-sed analyis
genera_arranged <- c("Lysobacter", "Stenotrophomonas", "Xylella", "Xanthomonas",
                     "Methylomicrobium", "Pseudomonas", "Acinetobacter",
                     "Alteromonas", "Photorhabdus", "Yersinia", "Erwinia",
                     "Edwardsiella", "Serratia", "Klebsiella", "Enterobacter",
                     "Salmonella", "Shigella", "Escherichia")

upset_pred <- select_columns_by_list(upsetSubsetPredicted, genera_arranged)
upset_pc <- select_columns_by_list(upsetSubsetPC, genera_arranged)
```

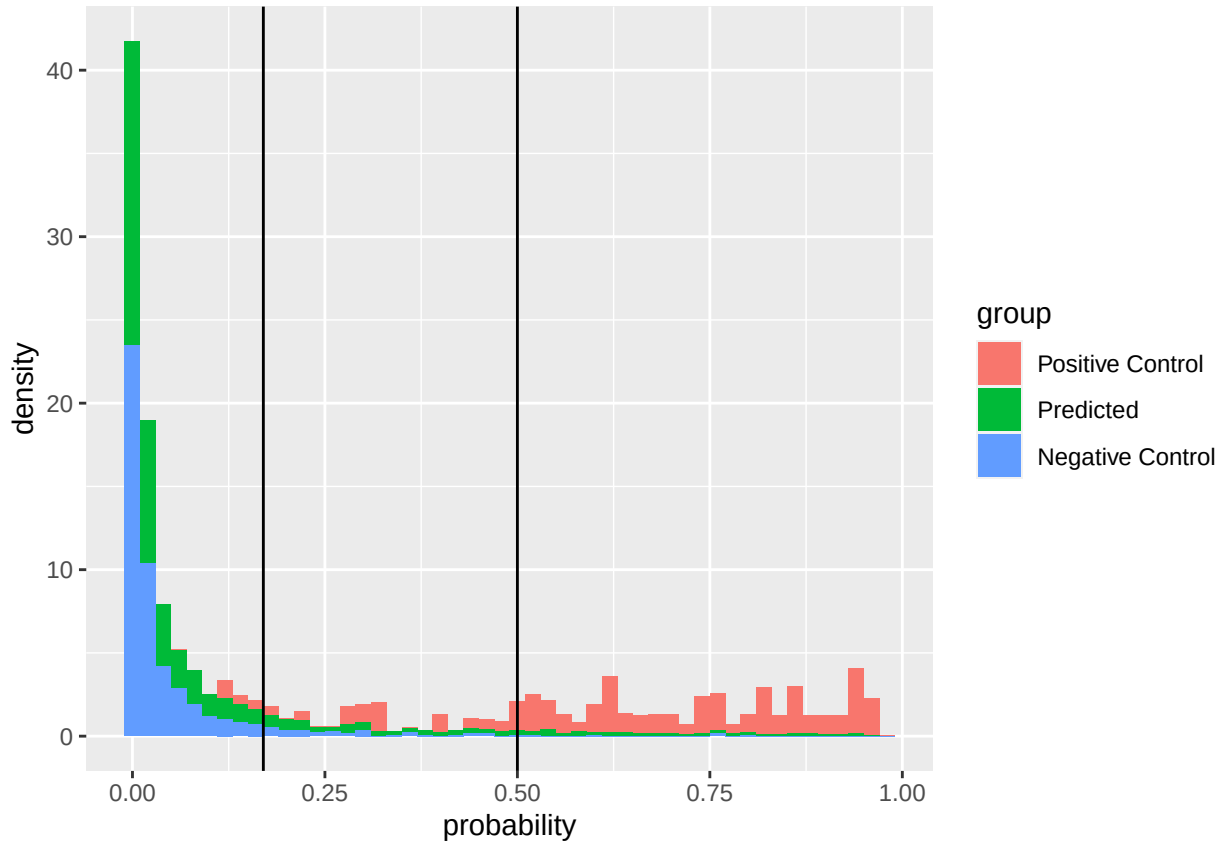
```
pc_freq <- colSums(upset_pc)
pred_freq <- colSums(upset_pred)

pc_freq <- vector_to_dataframe(pc_freq, 'genera')
pred_freq <- vector_to_dataframe(pred_freq, 'genera')

pc_freq <- pc_freq %>% dplyr::rename(freq = vec) %>% mutate(group ='pc')
pred_freq <- pred_freq %>% dplyr::rename(freq = vec) %>% mutate(group ='pred')

freq_data <-pc_freq %>% bind_rows(pred_freq)

freq_data$genera <- factor(freq_data$genera, levels = unique(freq_data$genera))

p <- ggplot() +
  geom_bar(data = freq_data,
           aes(y = genera, x = freq, group = group, fill = group),
           stat = 'identity', position = 'dodge')

p
```




```
if(FALSE){
  ggsave(filename = paste0(figure_path, 'SVG/rnas_frequency.svg'),
  plot = p, width = 8, height = 16)
}
```

Random forest interpretation figure

```
load("~/bin/PhD/Chapter_4/chapter_4_files/predDat.Rda")
load("~/bin/PhD/Chapter_4/chapter_4_files/validatation2.Rda")

#select the desired columns from the predicted data and validatation data
probDat <- predDat %>%
  select(probability, ID, group, srna.counts.2) %>%
  bind_rows(validation2 %>% select(probability, ID, group, srna.counts.2))

#not sure if setting factors will break anything so using another data frame
plotDat <- probDat
plotDat$group <- factor(plotDat$group,
                        levels = c('Positive Control',
                                   'Predicted',
                                   'Negative Control'))

#plot histogram of the probabilities
p <- ggplot() +
  geom_histogram(data = plotDat,
                 aes(x = probability,
                     y = ..density..,
                     group = group,
                     fill = group),
                 binwidth = 0.02) +
  geom_vline(xintercept = 0.17) +
  geom_vline(xintercept = 0.5)
p
```

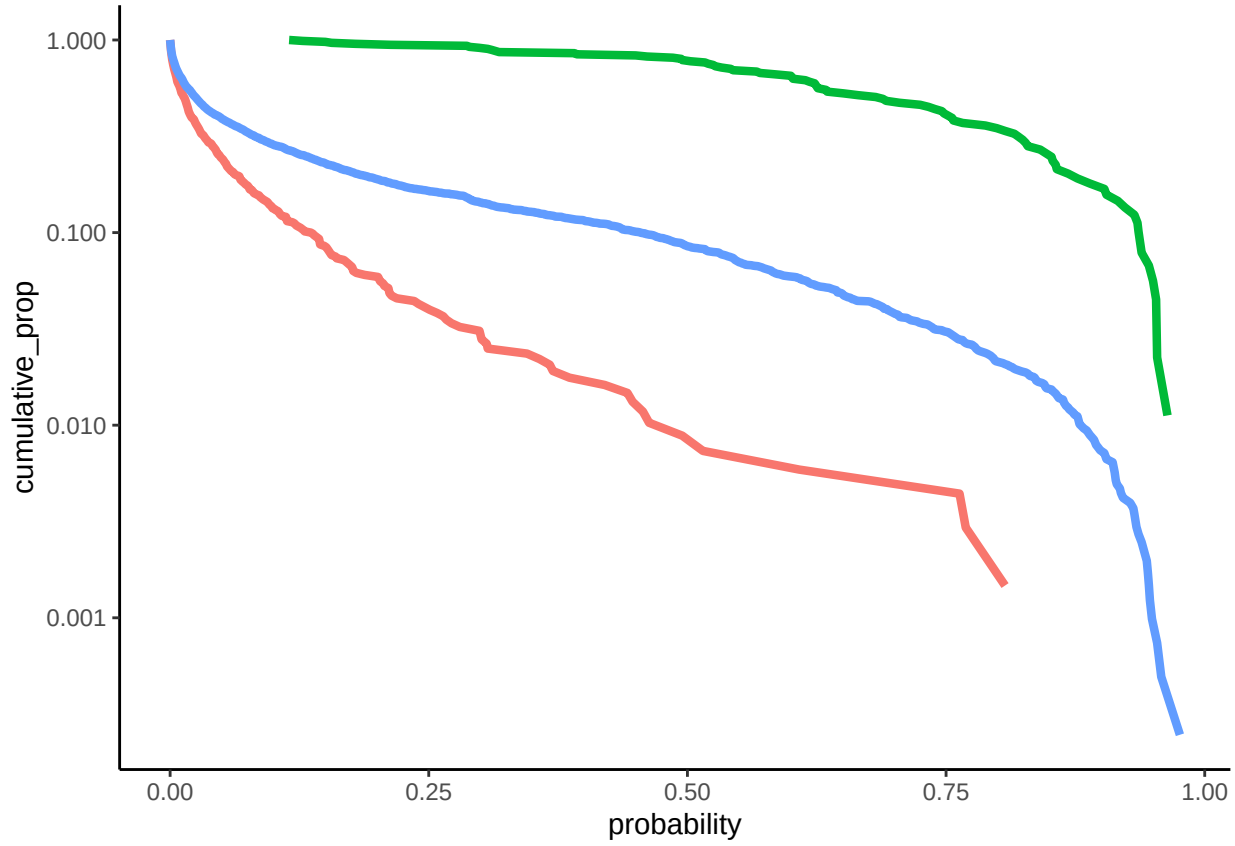



```
if(FALSE){
  ggsave(filename = paste0(figure_path, "SVG/histogram_probabilities.svg"),
         plot = p, width = 178, height = 155, units = "mm")
}

#get the cumulative counts of the number of alignments as probability increases
countsCumul <- cumulativeCounts(dists = probDat,
                                smooth = F,
                                target_column = 'probability')
```

```
## `summarise()` has grouped output by 'group'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'group'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'group'. You can override using the
## `.groups` argument.
```

```
##produces plot to use for figure showing probability results
p <- ggplot() +
  geom_line(data = countsCumul, aes(x= probability,
                                    y = cumulative_prop,
                                    group = group,
                                    colour = group),
            size = 1.5,
            show.legend = F) +
```

```
  scale_y_continuous(trans = 'log10')
p + theme_classic()
```




```
if(FALSE){
  ggsave(filename = paste0(figure_path, "SVG/cumulative_probabilities.svg"),
                           plot = p, width = 178, height = 155, units = "mm")
}
```

```
load("~/bin/PhD/Chapter_4/chapter_4_files/predDat.Rda")
load("~/bin/PhD/Chapter_4/chapter_4_files/validatation2.Rda")

#select the desired columns from the predicted data and validatation data
probDat <- predDat %>%
  select(probability, ID, group, srna.counts.2) %>%
  bind_rows(validation2 %>%
              select(probability, ID, group, srna.counts.2))

#not sure if setting factors will break anything so using another data frame
plotDat <- probDat
plotDat$group <- factor(plotDat$group,
                        levels = c('Positive Control',
                                   'Predicted',
                                   'Negative Control'))

#get the FNR and PPV values at each threshold
```

```
if(FALSE){
  for(i in seq(0,1, by=0.01)){
    scores <- scoreProbabities(plotDat,
                               threshold = i,
                               target_column = 'probability')
    print(paste0(i, ': ', scores$fnr, ', ', scores$ppv))
  }
}


scores <- scoreProbabities(plotDat,
                           threshold = 0.17,
                           target_column = 'probability')
printListSubset(scores,
                vec = c('ppv', 'fnr', 'pred_pos', 'pred_pct'),
                startText = 'p > 0.17', round_val = 3)
```

```
## p > 0.17
##   ppv: 0.644
##   fnr: 0.045
##   pred_pos: 851
##   pred_pct: 0.21
##
```

```
scores <- scoreProbabities(plotDat,
                           threshold = 0.5,
                           target_column = 'probability')
printListSubset(scores,
                vec = c('ppv', 'fnr', 'pred_pos', 'pred_pct', 'pc_pct'),
                startText = 'p > 0.5', round_val = 3)
```

```
## p > 0.5
##   ppv: 0.932
##   fnr: 0.225
##   pred_pos: 342
##   pred_pct: 0.084
##   pc_pct: 0.775
##
```

```
scores <- scoreProbabities(plotDat,
                           threshold = 0.81,
                           target_column = 'probability')
printListSubset(scores,
                vec = c('ppv', 'fnr', 'pred_pos', 'pred_pct', 'pc_pct'),
                startText = 'p > 0.81', round_val = 3)
```

```
## p > 0.81
##   ppv: 1
##   fnr: 0.674
##   pred_pos: 82
##   pred_pct: 0.02
##   pc_pct: 0.326
##
```