Frequentist Bands

Jeremy Ash

April 17, 2018

Loading in data

```
rm(list = ls())

setwd("C:/Users/Vestige/Google Drive/courses/grad_classes/ST540_Bayes/final_exam/")
load("empirical_probs.RDATA")

# All that I am using for now is the first function, which
# Computes the Frequentist confidence intervals
source("convenience_functions.r")

head(probs.Burd)
```

```
##          Observed Tree   RF        SVM        NNet KNN     PLSLDA
## 5388992         1  0.2 0.66 0.95141274 0.302391057 0.2 0.05724840
## 5388983         1  1.0 0.49 0.99655822 0.210089106 0.2 0.07885877
## 663143          1  1.0 0.56 0.99971525 0.263502387 0.6 0.24185024
## 10607           1  0.0 0.17 0.05466489 0.005392594 0.1 0.03806212
## 5388972         1  0.0 0.07 0.30697769 0.287878797 0.3 0.46096039
## 11970251        1  0.2 0.27 0.34797873 0.258992258 0.4 0.41671143
```

```
head(probs.Phar[, 1:5])
```

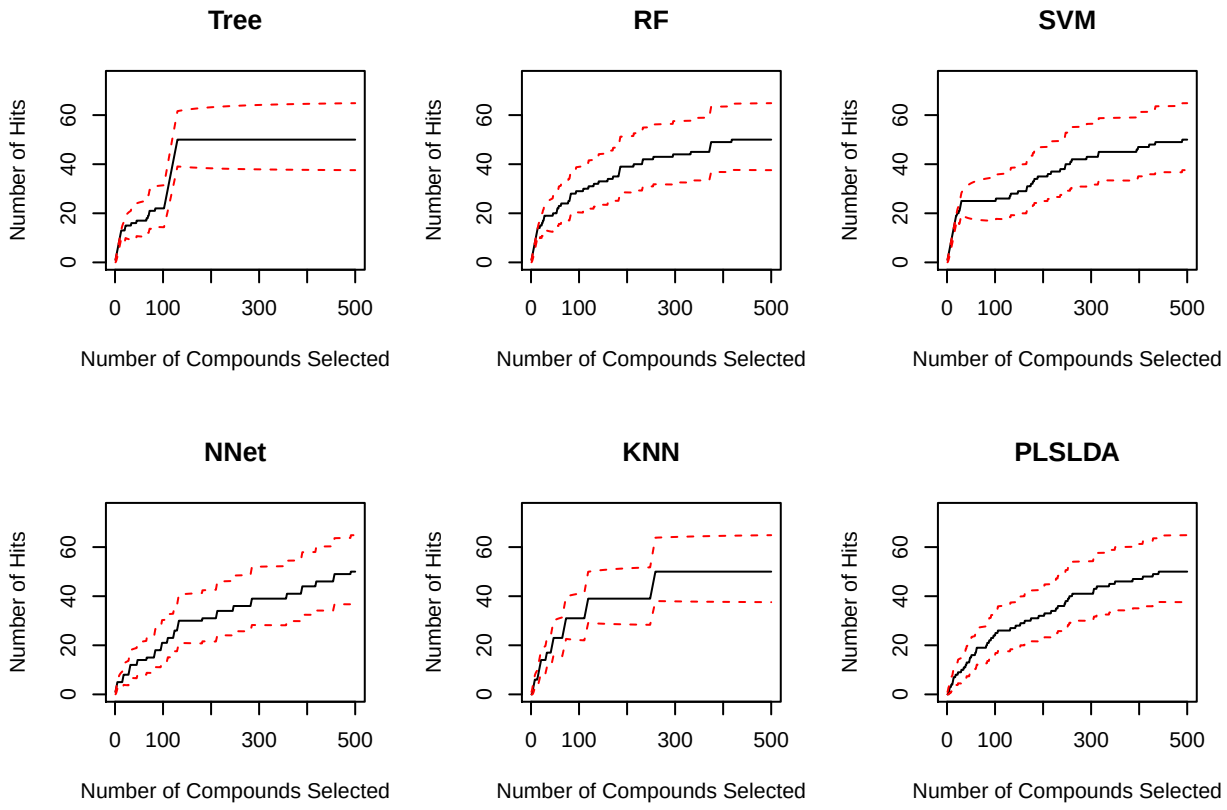
```
##          Observed       Tree   RF        SVM        NNet
## 5388992         1 0.80000000 0.68 0.08574760 0.941552309
## 5388983         1 0.80000000 0.82 0.09944312 0.000000000
## 663143          1 0.24324324 0.33 0.09902292 0.001495934
## 10607           1 0.06043956 0.19 0.09944312 1.000000000
## 5388972         1 0.80000000 0.60 0.09306289 0.000000000
## 11970251        1 0.83333333 0.63 0.09106322 0.010002460
```

Psuedo code

- For each D-M combination
  - Order the predicted probabilities in decreasing order
  - Order the true activity vector in the same way
  - Compute the number of hits at each number of tests
  - Create a Clopper Pearson interval
  - Plot these intervals on the accumulation curve

Confidence intervals for the accumulation curves for Burden Number descriptors Models

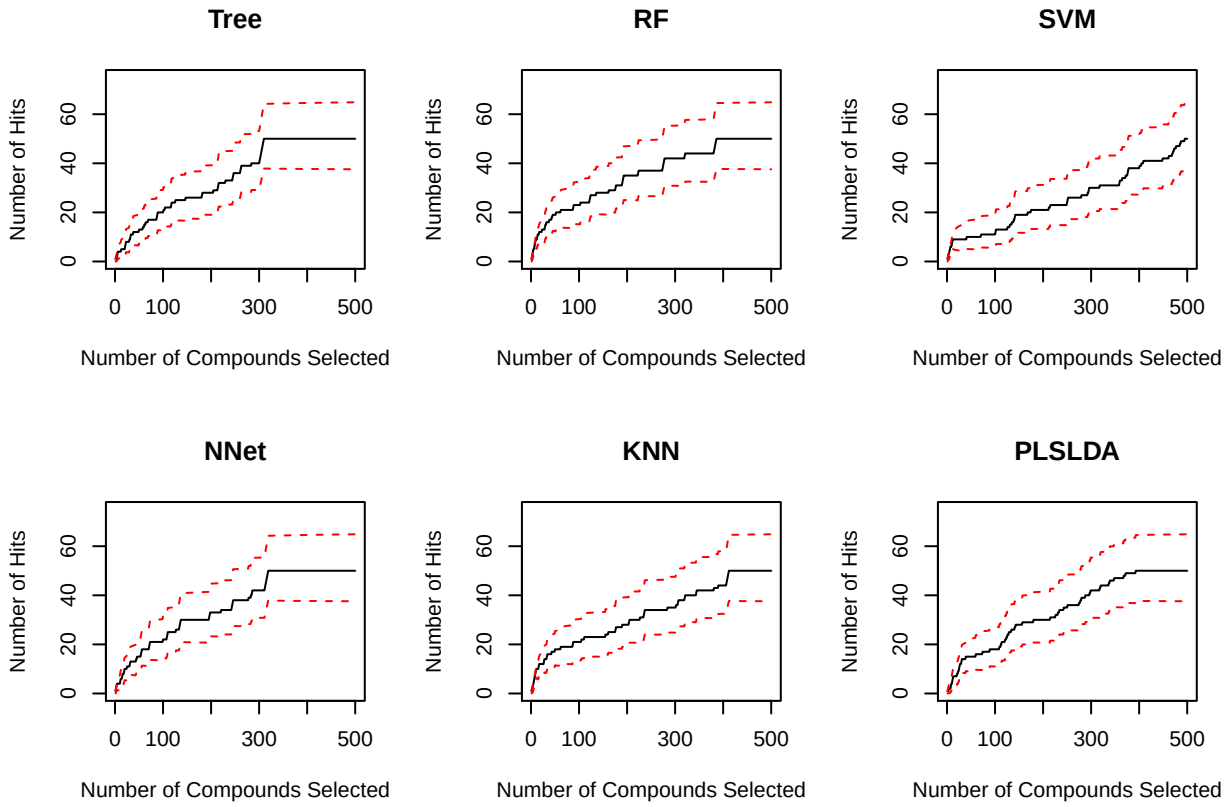
```
int.list.burd <- list(length = ncol(probs.Burd) - 1)
for(i in 2:ncol(probs.Burd)) {
  probs <- probs.Burd[, i]
  hit.vec <- probs.Burd$Observed

  order.idx <- order(probs, decreasing = T)
  probs <- probs[order.idx]
  hit.vec <- hit.vec[order.idx]

  m <- length(probs)
  # Matrix containing the number of hits and lower and upper bounds for 95%
  # confidence intervals for each number of tests
  int.mat <- matrix(ncol = 3, nrow = m)
  colnames(int.mat) <- c("NHits", "LB", "UB")
  for(j in 1:m) {
    int.mat[j, 1] <- sum(hit.vec[1:j])
    # All you need to know here is that I am providing my success probabilities
    # For the j compounds priotized for testing and the number of hits
    # And using this to compute a confidence interval using a
    # Frequentist method.  Similar to getting 95%
    # CI for a normally distributed variable.
    # We will estimate these confidence interval using Bayesian
    # methods you are familiar with.
    int.mat[j, 2:3] <- j * CPInt(x = sum(hit.vec[1:j]), p.vec = probs[1:j])
  }
  int.list.burd[[i-1]] <- int.mat
}

# Plotting the accumulation curves and confidence band for each modeling method
par(mfrow = c(2, 3))
for (i in seq_along(int.list.burd)) {
  plot(int.list.burd[[i]][, 1], type = "l", ylim = c(0, 75),
       main = colnames(probs.Burd)[i+1], ylab = "Number of Hits",
       xlab = "Number of Compounds Selected")
  lines(int.list.burd[[i]][, 2], type = "l", lty = "dashed", col = "red")
  lines(int.list.burd[[i]][, 3], type = "l", lty = "dashed", col = "red")
}
```



Confidence intervals for the accumulation curves for Pharmamocophores descriptors Models

```
int.list.phar <- list(length = ncol(probs.Burd) - 1)
for(i in 2:ncol(probs.Phar)) {
  probs <- probs.Phar[, i]
  hit.vec <- probs.Phar$Observed

  order.idx <- order(probs, decreasing = T)
  probs <- probs[order.idx]
  hit.vec <- hit.vec[order.idx]

  m <- length(probs)
  int.mat <- matrix(ncol = 3, nrow = m)
  colnames(int.mat) <- c("NHits", "LB", "UB")
  for(j in 1:m) {
    int.mat[j, 1] <- sum(hit.vec[1:j])
    int.mat[j, 2:3] <- j * CPInt(x = sum(hit.vec[1:j]), p.vec = probs[1:j])
  }
  int.list.phar[[i-1]] <- int.mat
}

par(mfrow = c(2, 3))
```

```
for (i in seq_along(int.list.phar)) {
  plot(int.list.phar[[i]][, 1], type = "l", ylim = c(0, 75),
       main = colnames(probs.Burd)[i+1], ylab = "Number of Hits",
       xlab = "Number of Compounds Selected")
  lines(int.list.phar[[i]][, 2], type = "l", lty = "dashed", col = "red")
  lines(int.list.phar[[i]][, 3], type = "l", lty = "dashed", col = "red")
}
```



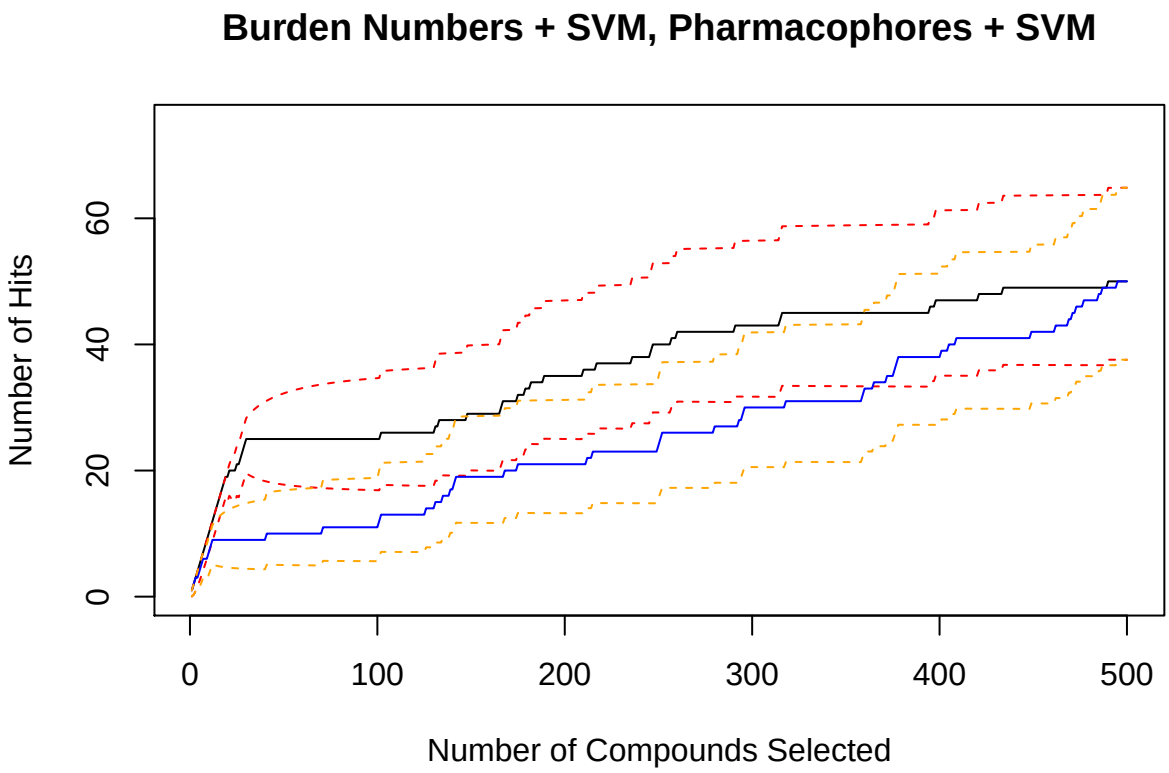
Only a few curves with significant differences

```
colnames(probs.Burd)[4]
```

```
## [1] "SVM"
```

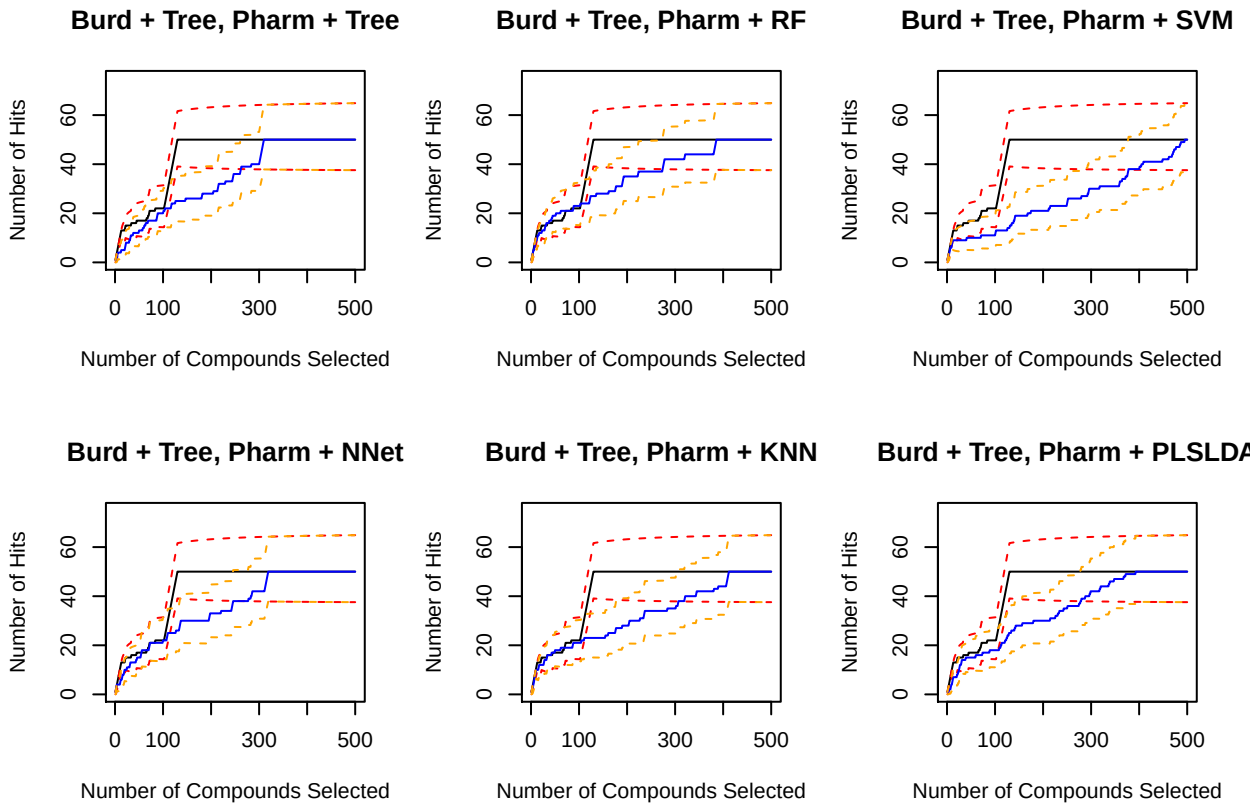
```
i <- 3
j <- 3
plot(int.list.burd[[i]][, 1], type = "l", ylim = c(0, 75),
     main = paste0("Burden Numbers + ", colnames(probs.Burd)[i+1],
                   ", Pharmacophores + ", colnames(probs.Phar)[j+1]),
     ylab = "Number of Hits",
     xlab = "Number of Compounds Selected")
lines(int.list.burd[[i]][, 2], type = "l", lty = "dashed", col = "red")
lines(int.list.burd[[i]][, 3], type = "l", lty = "dashed", col = "red")

lines(int.list.phar[[j]][, 1], type = "l", col = "blue")
```

```
lines(int.list.phar[[j]][, 2], type = "l", lty = "dashed", col = "orange")
lines(int.list.phar[[j]][, 3], type = "l", lty = "dashed", col = "orange")
```



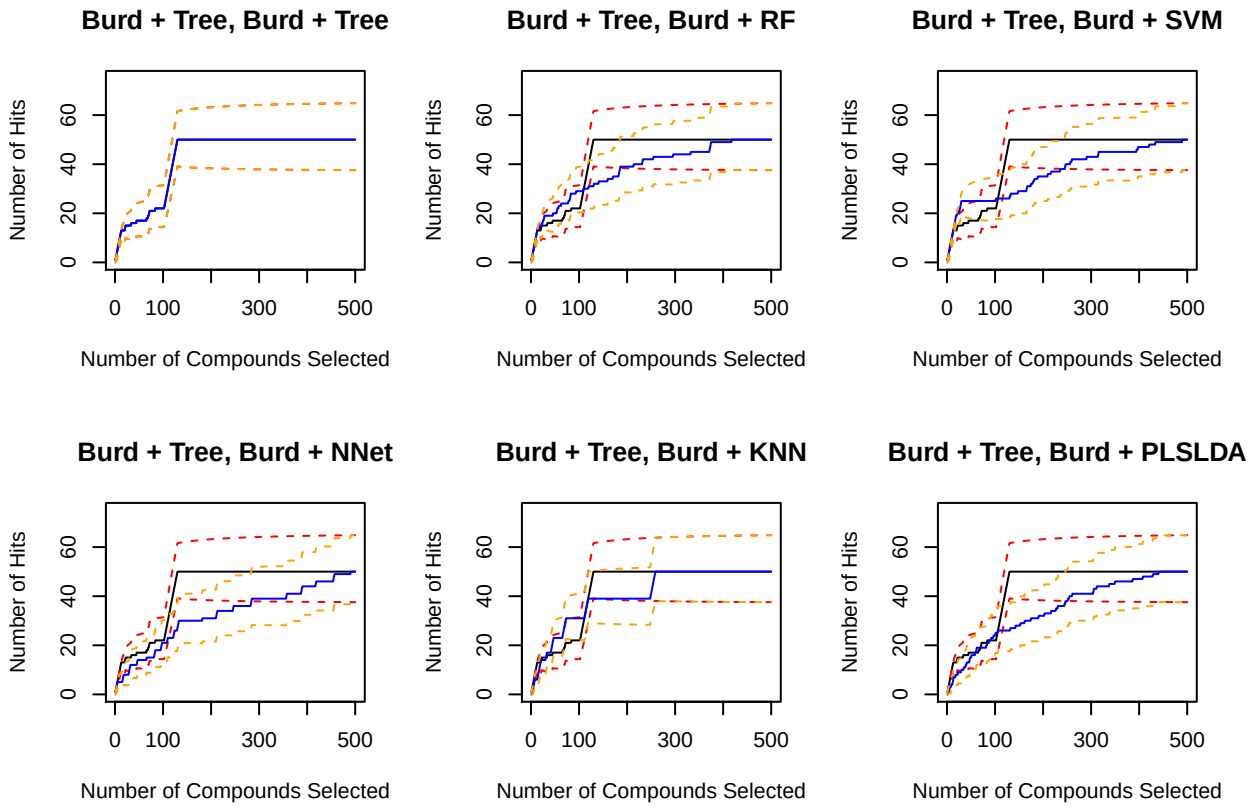
We can conclude that the Burden Numbers + Tree method performs signficantly better at identifying all of the actives early on than any of the Pharmacophores models. This also highlights that the difference in the performance of these models needs to be huge in order for us to have significance. This is a challenging problem where the number of actives in the data set is large, and the number of

```
i <- 1
par(mfrow = c(2, 3))
for (j in 1:6) {
  plot(int.list.burd[[i]][, 1], type = "l", ylim = c(0, 75),
       main = paste0("Burd + ", colnames(probs.Burd)[i+1],
                     ", Pharm + ", colnames(probs.Phar)[j+1]),
       ylab = "Number of Hits",
       xlab = "Number of Compounds Selected")
  lines(int.list.burd[[i]][, 2], type = "l", lty = "dashed", col = "red")
  lines(int.list.burd[[i]][, 3], type = "l", lty = "dashed", col = "red")

  lines(int.list.phar[[j]][, 1], type = "l", col = "blue")
  lines(int.list.phar[[j]][, 2], type = "l", lty = "dashed", col = "orange")
  lines(int.list.phar[[j]][, 3], type = "l", lty = "dashed", col = "orange")
}
```

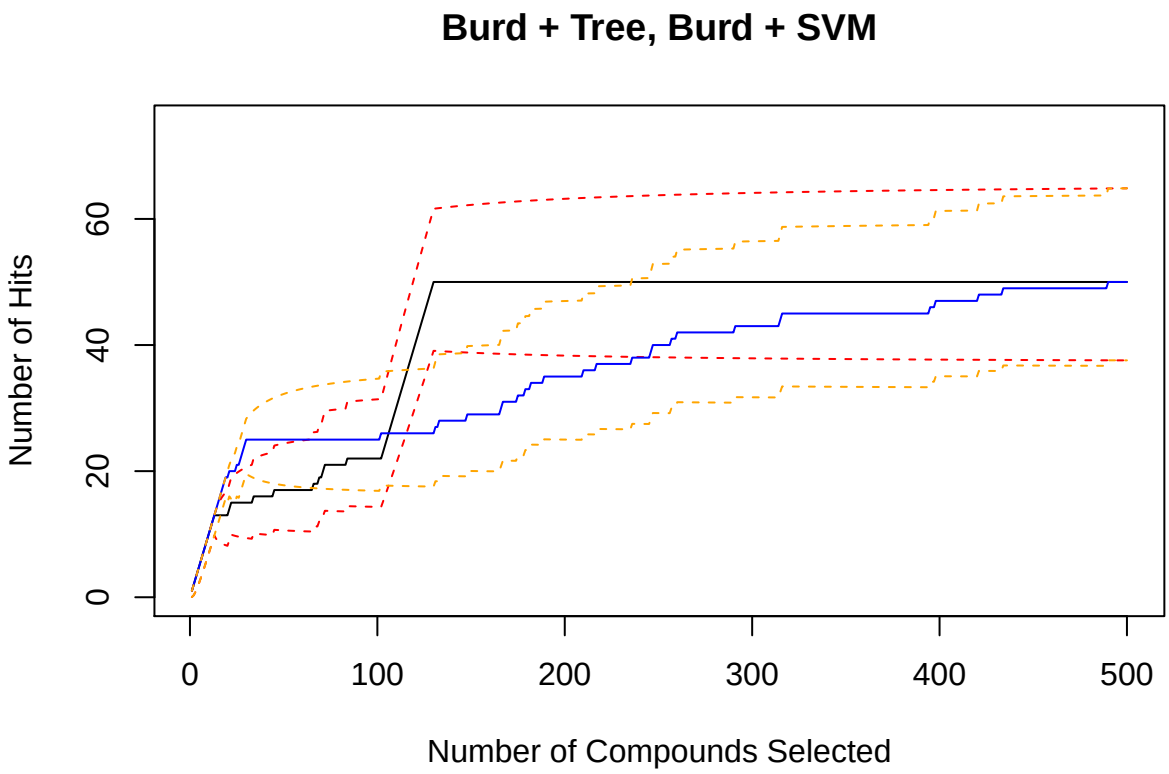


But the Burden Number + Tree method does not show significant improvement over other Burden Number modeling method combintations.

```
i <- 1
par(mfrow = c(2, 3))
for (j in 1:6) {
  plot(int.list.burd[[i]][, 1], type = "l", ylim = c(0, 75),
       main = paste0("Burd + ", colnames(probs.Burd)[i+1],
                     ", Burd + ", colnames(probs.Burd)[j+1]),
       ylab = "Number of Hits",
       xlab = "Number of Compounds Selected")
  lines(int.list.burd[[i]][, 2], type = "l", lty = "dashed", col = "red")
  lines(int.list.burd[[i]][, 3], type = "l", lty = "dashed", col = "red")

  lines(int.list.burd[[j]][, 1], type = "l", col = "blue")
  lines(int.list.burd[[j]][, 2], type = "l", lty = "dashed", col = "orange")
  lines(int.list.burd[[j]][, 3], type = "l", lty = "dashed", col = "orange")
}
```




```
i <- 1
j <- 3
plot(int.list.burd[[i]][, 1], type = "l", ylim = c(0, 75),
     main = paste0("Burd + ", colnames(probs.Burd)[i+1],
                   ", Burd + ", colnames(probs.Burd)[j+1]),
     ylab = "Number of Hits",
     xlab = "Number of Compounds Selected")
lines(int.list.burd[[i]][, 2], type = "l", lty = "dashed", col = "red")
lines(int.list.burd[[i]][, 3], type = "l", lty = "dashed", col = "red")

lines(int.list.burd[[j]][, 1], type = "l", col = "blue")
lines(int.list.burd[[j]][, 2], type = "l", lty = "dashed", col = "orange")
lines(int.list.burd[[j]][, 3], type = "l", lty = "dashed", col = "orange")
```


Burd + Tree, Burd + SVM
Number of Hits
0
20
40
60
0
100
200
300
400
500
Number of Compounds Selected