# Evaluate testing data (binary-class) - XGBoost

*EVE W.*

*2019-11-16*

## Contents

```r
## user input
project_home <- "~/EVE/examples"
project_name <- "xgboostR_binary_1"
```

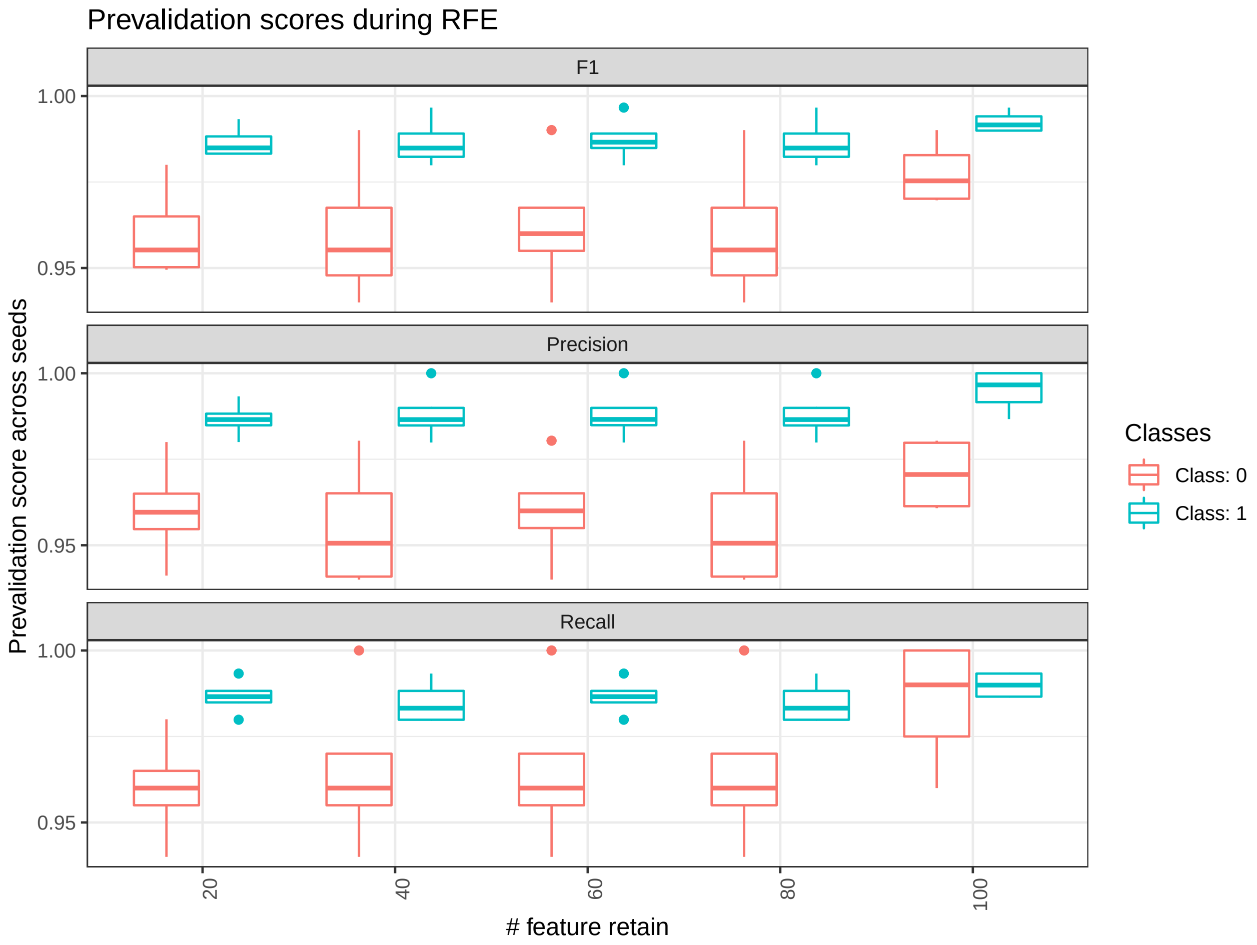
## 0. Load Data

```
## Warning: `cols` is now required.
## Please use `cols = c(df)`

## Warning: `cols` is now required.
## Please use `cols = c(df)`

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Patient_ID = col_character()
## )

## See spec(...) for full column specifications.

## 199 of samples were used

## 100 of full features

## 4 runs, each run contains 3 CVs.

## Labels:
```

run with XGBoost.r evaluation metric: .

# 1. Scores

## 1.1 Scores per Class



Confusion Matrix

```
## confusion matrix at feature size = 100
## sum across 4 seeds
```

```
##           Reference
## Prediction   0   1
##          0 197   6
##          1   3 590
```

### 1.2 Average score

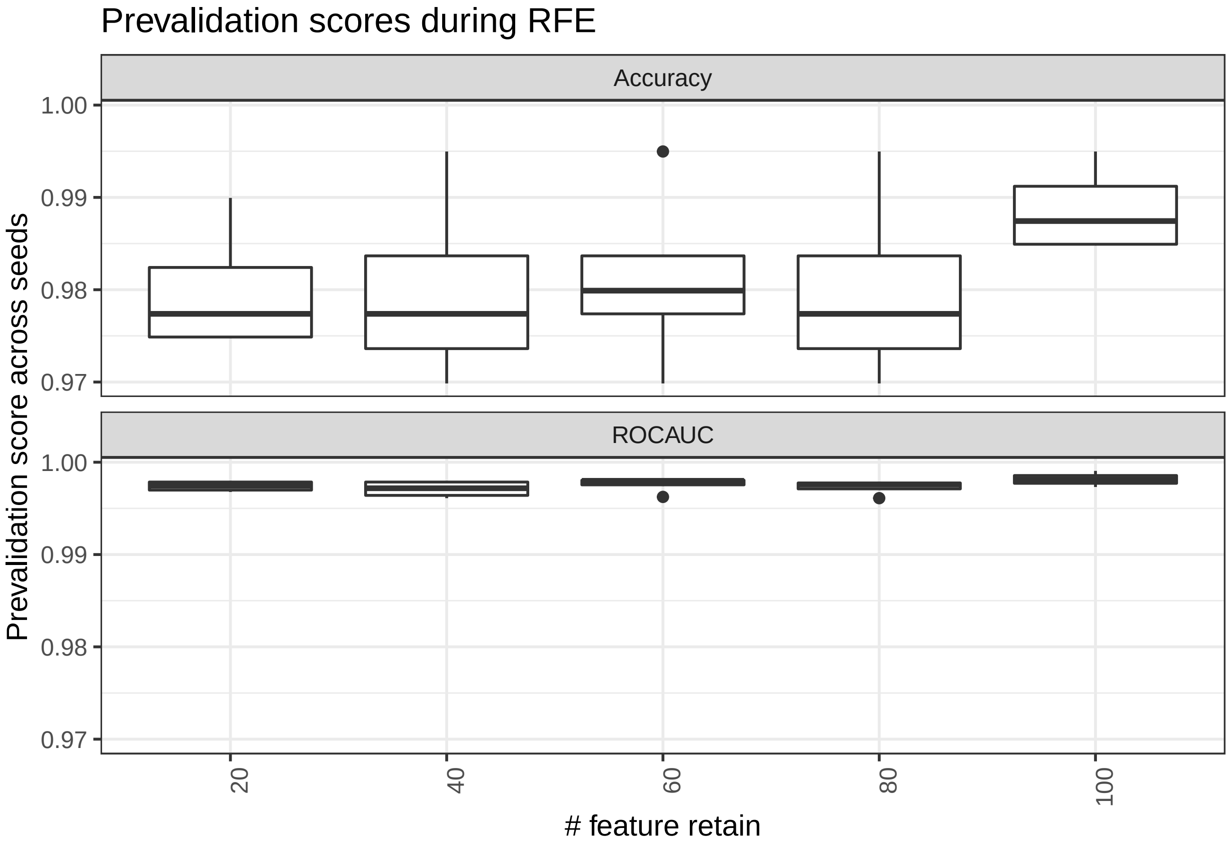
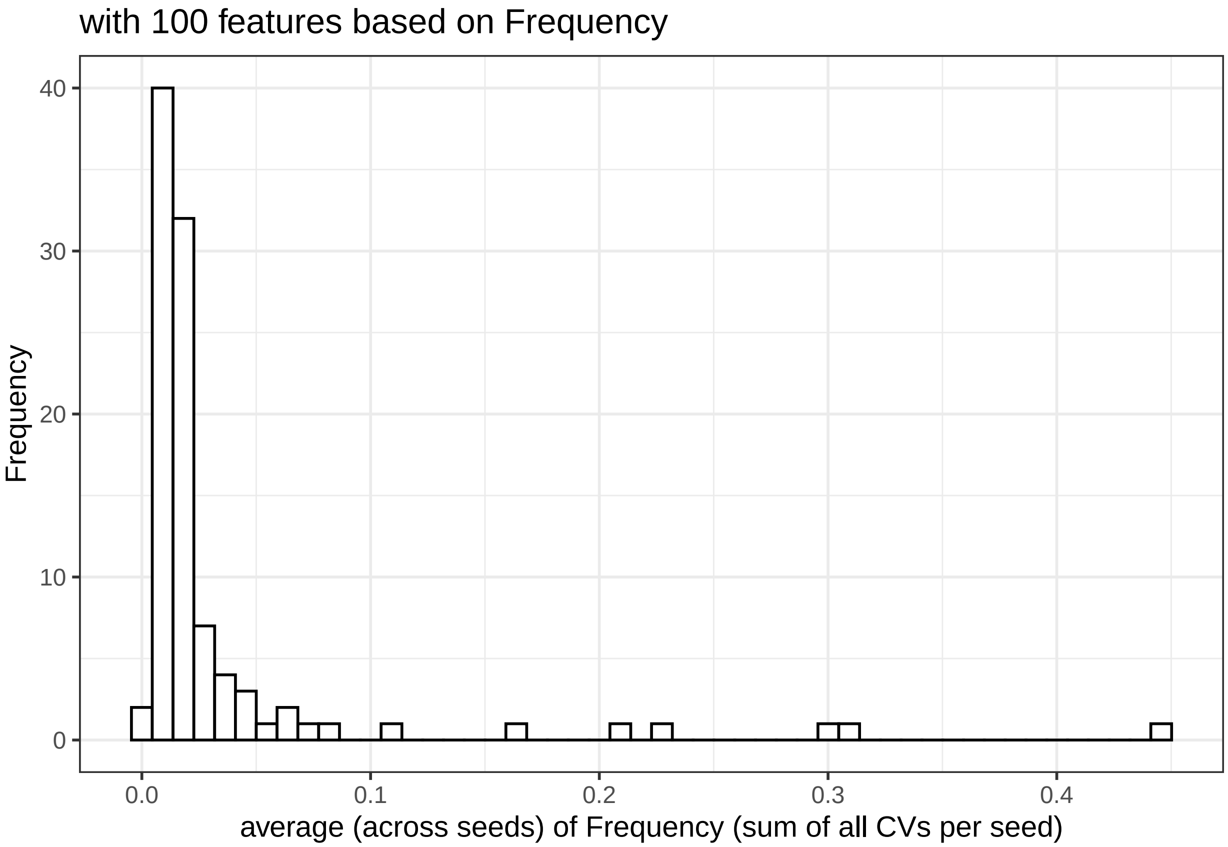
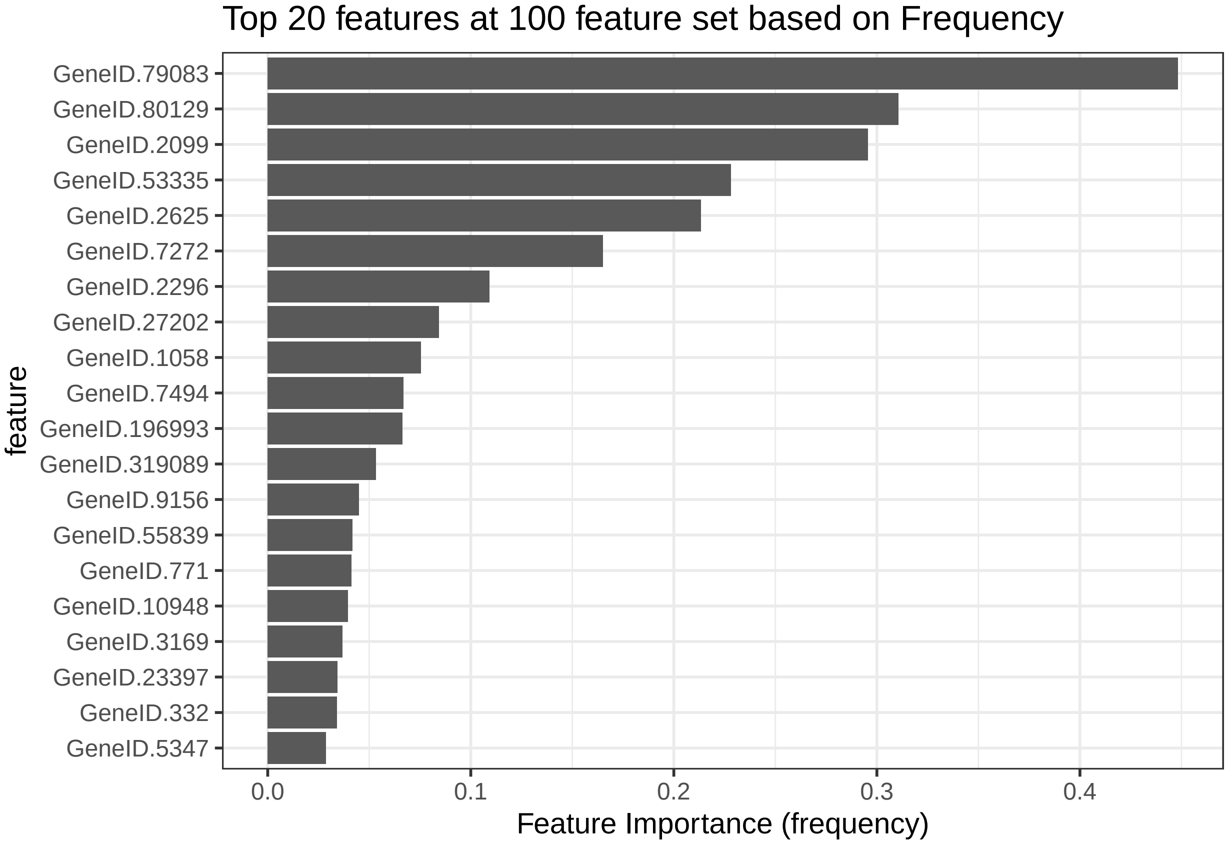


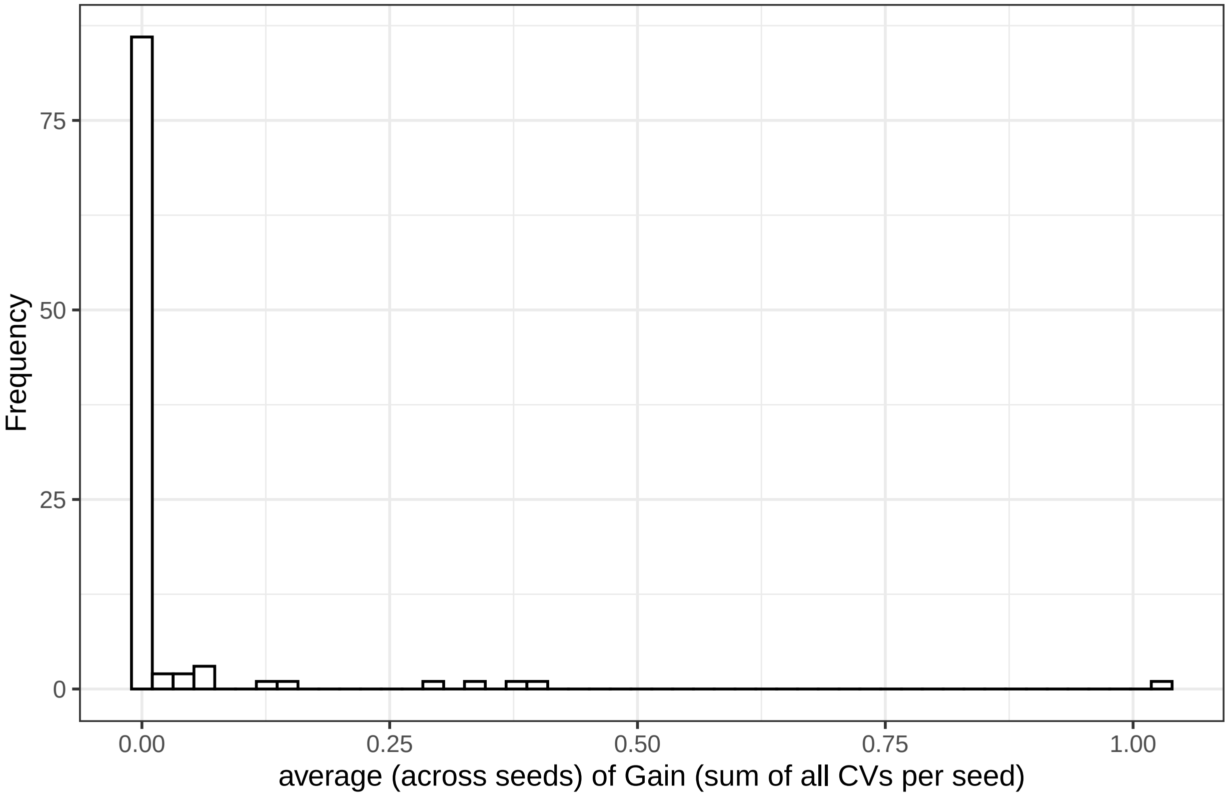
Table 1: best scores

| metrics | size.max | median.max | size.min | median.min |
|---|---:|---:|---:|---:|
| Accuracy | 100 | 0.987 | 20 | 0.977 |
| F1 | 100 | 0.983 | 20 | 0.970 |
| Precision | 100 | 0.982 | 40 | 0.969 |
| Recall | 100 | 0.988 | 20 | 0.972 |
| ROCAUC | 100 | 0.998 | 40 | 0.997 |

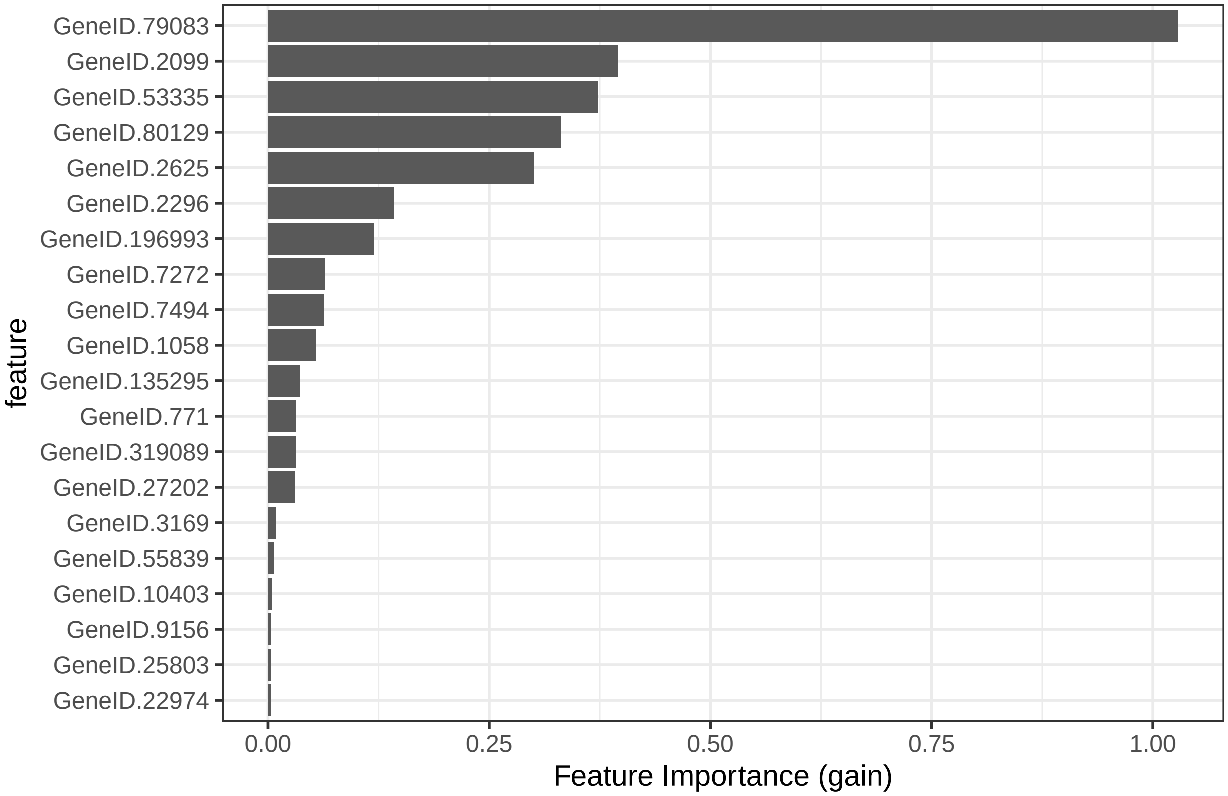
## 2. Important Features



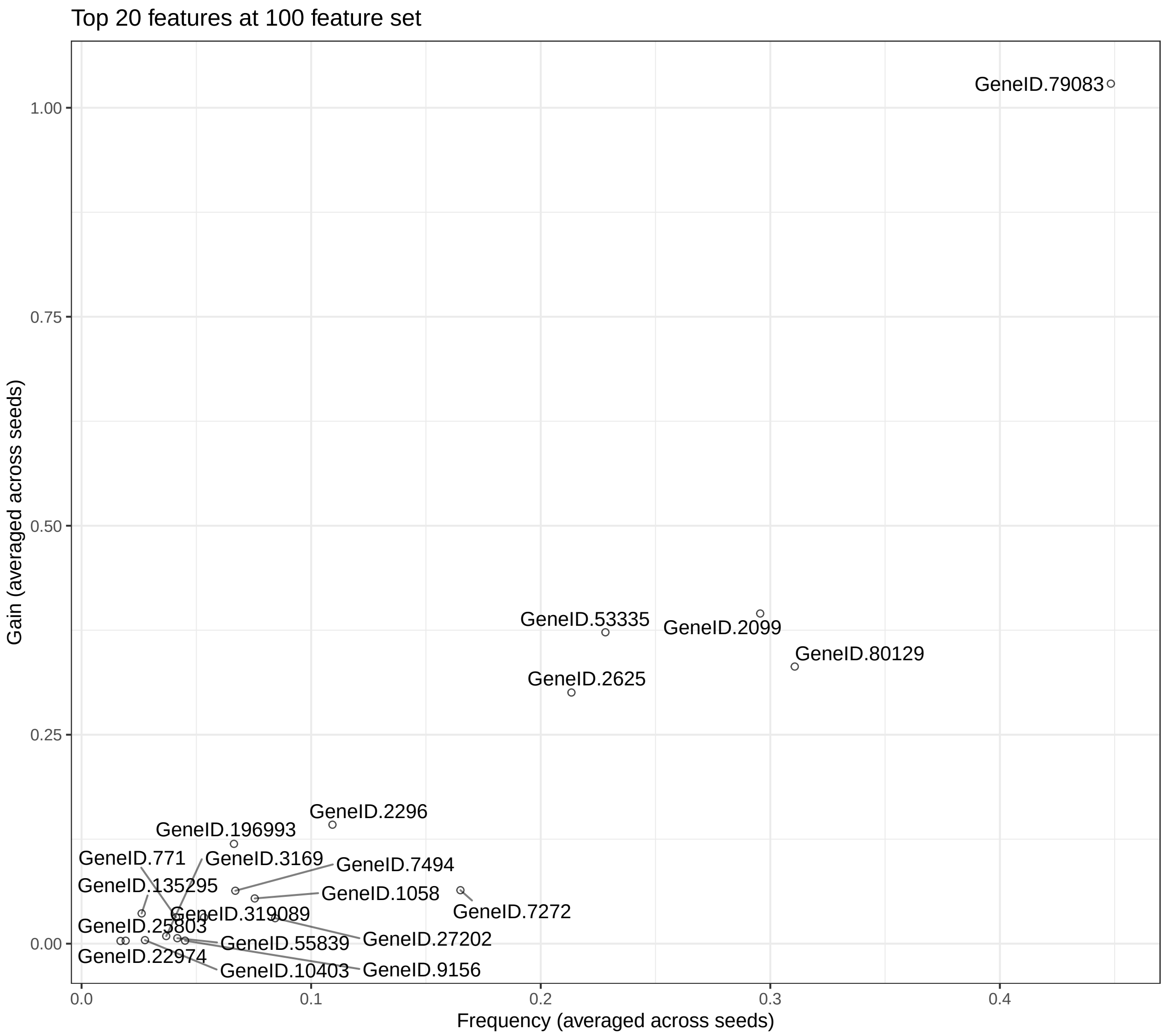


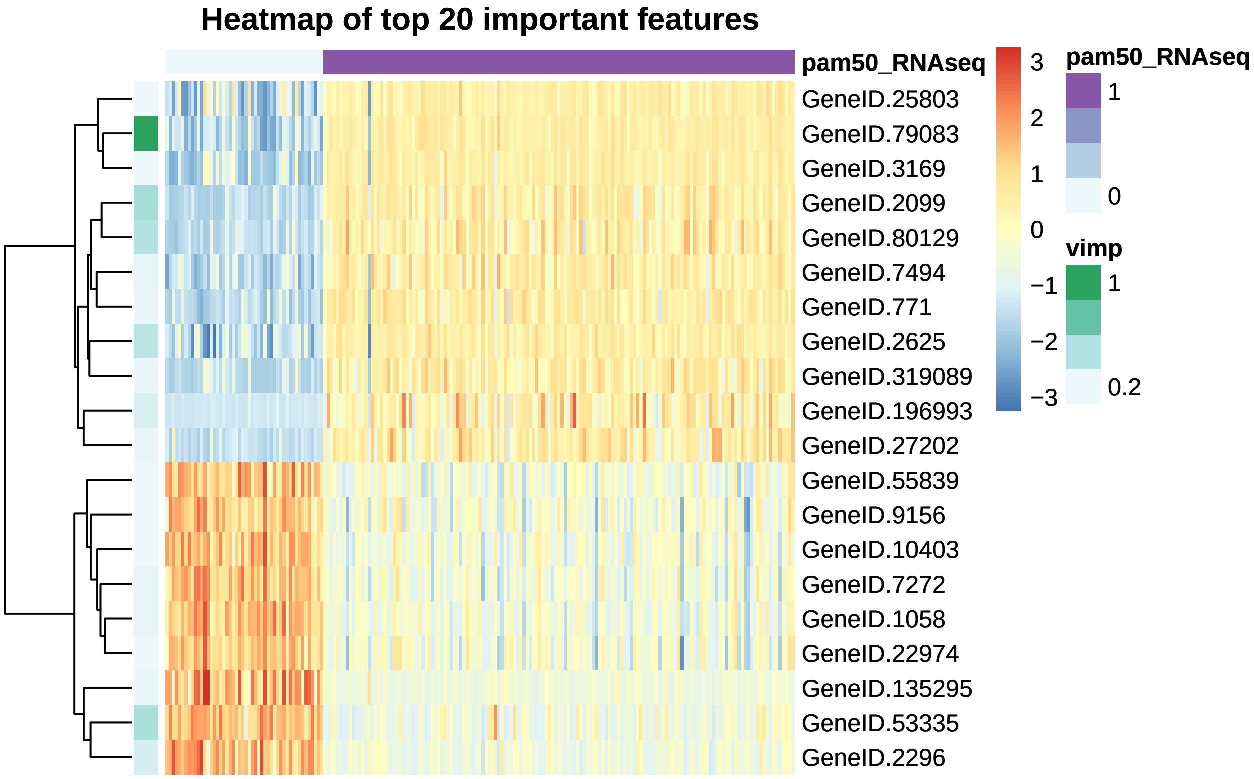


with 100 features based on Gain



Top 20 features at 100 feature set based on Gain


Top 20 features at 100 feature set
Gain (averaged across seeds)
1.00
0.75
0.50
0.25
0.00
Frequency (averaged across seeds)
0.0
0.1
0.2
0.3
0.4
GeneID.79083
GeneID.53335
GeneID.2099
GeneID.80129
GeneID.2625
GeneID.2296
GeneID.196993
GeneID.771
GeneID.3169
GeneID.7494
GeneID.135295
GeneID.1058
GeneID.7272
GeneID.319089
GeneID.25803
GeneID.27202
GeneID.55839
GeneID.22974
GeneID.10403
GeneID.9156

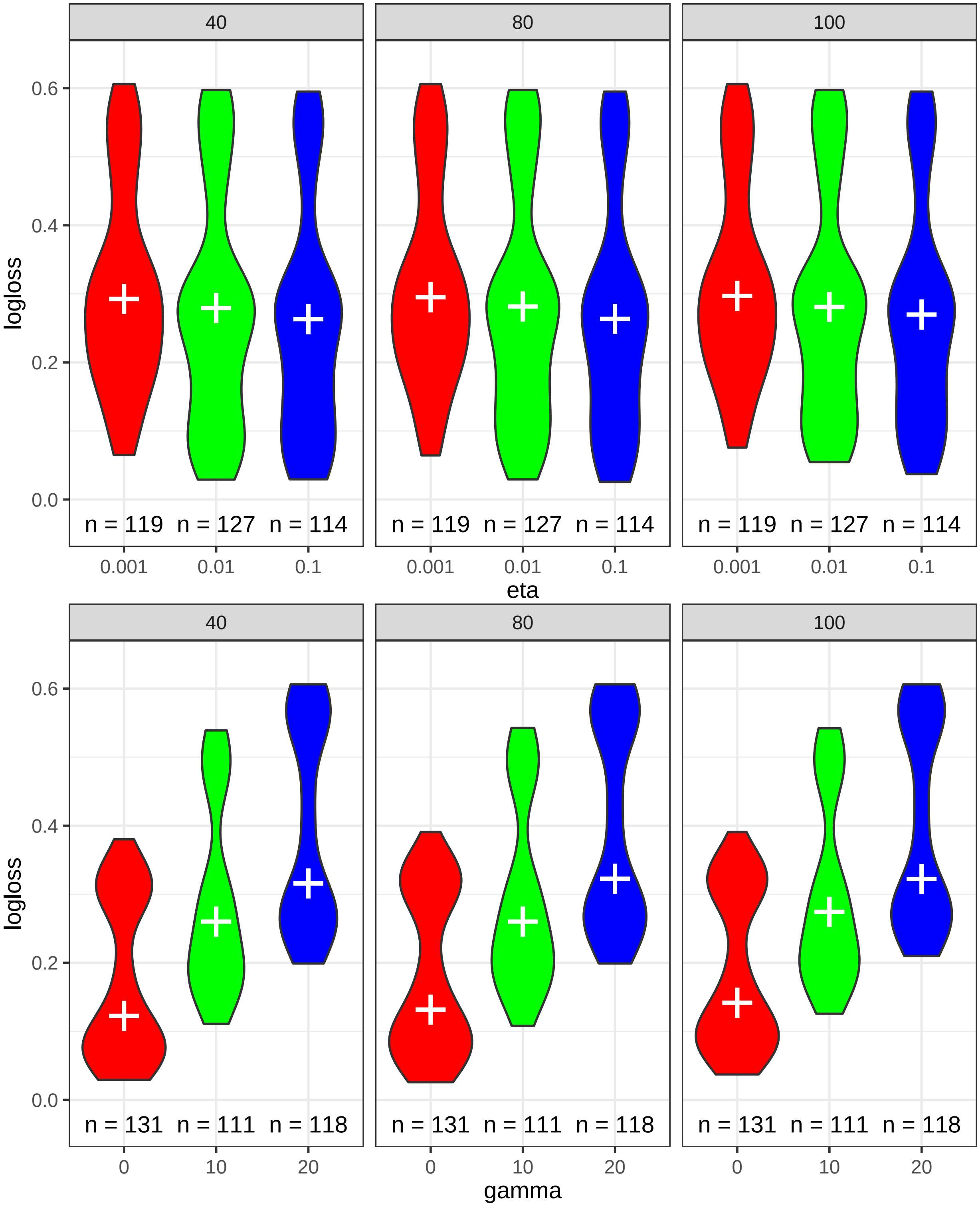


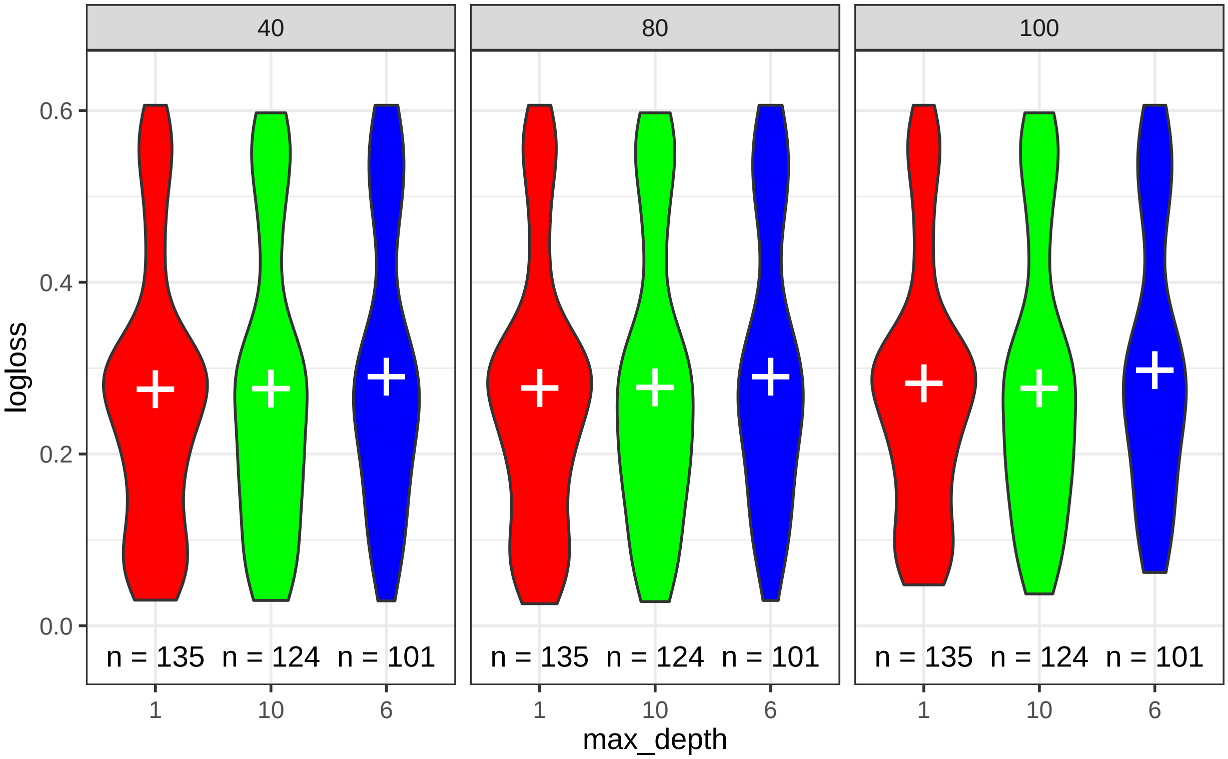
## 3. Hyper-parameters

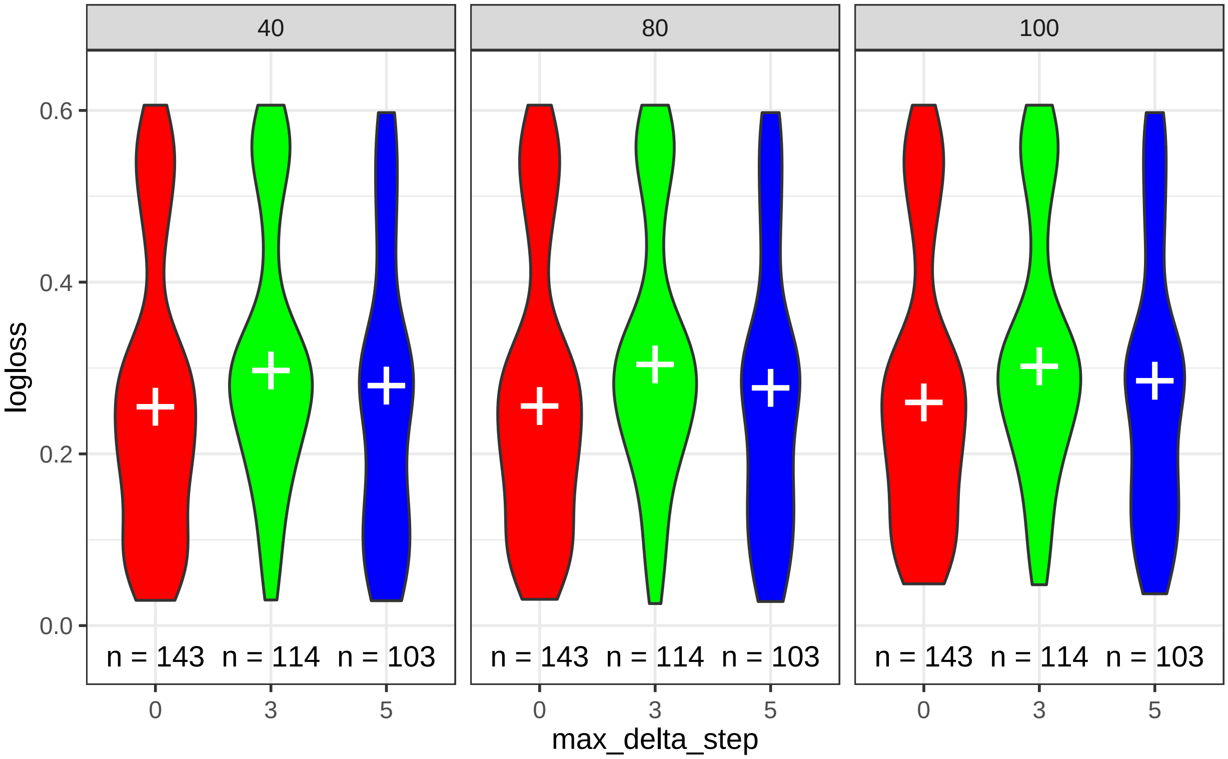
```
## Warning: `cols` is now required.
## Please use `cols = c(df)`
```

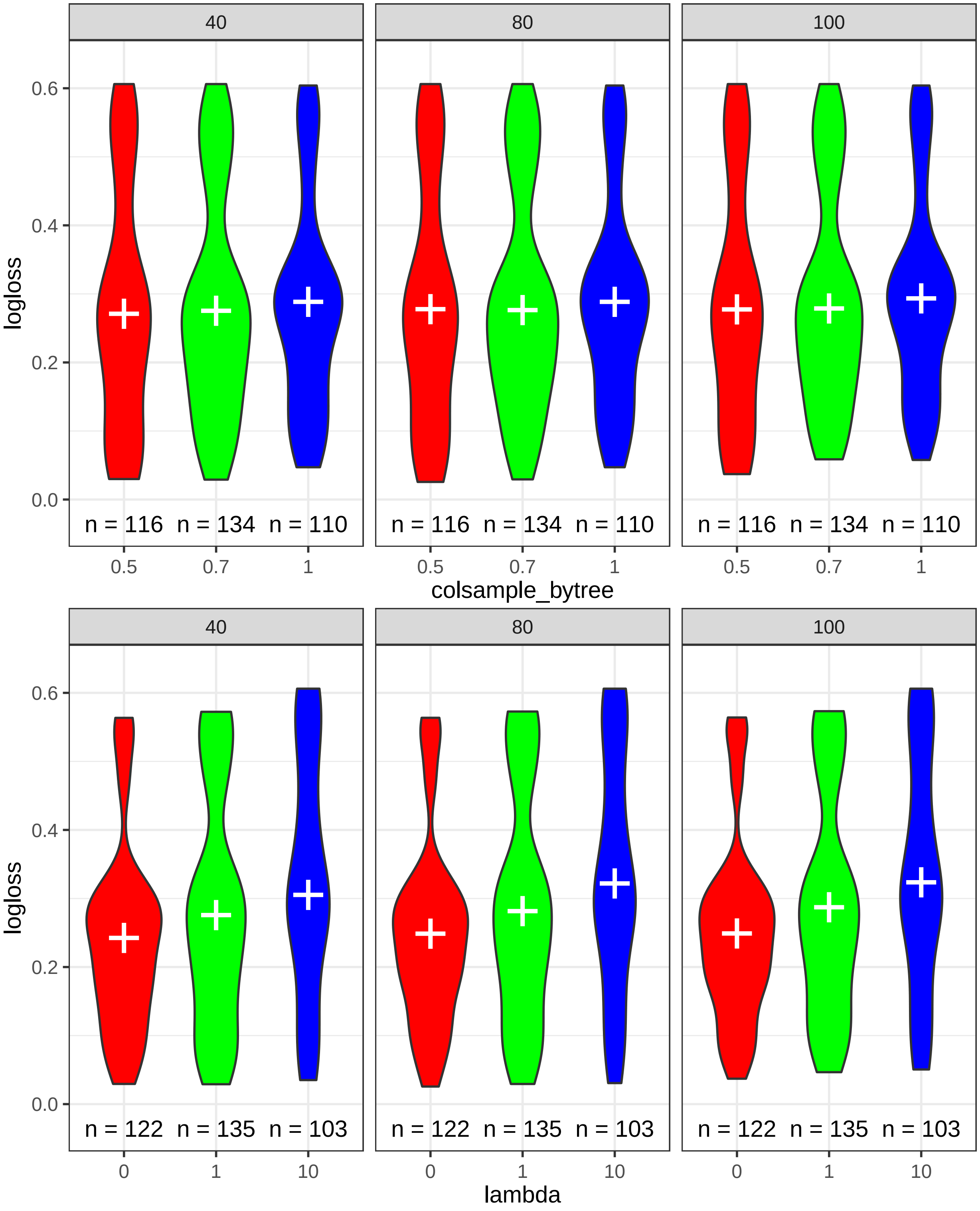
parameter optimization file (1080 records) includes 4 seeds. Each seed generates 3 cv splits. Within each cv split, there is a 3 step RFE (at 40, 80, 100). So 1080 / 4 / 3 / 3 = 30 parameter combinations tried in each cv split.

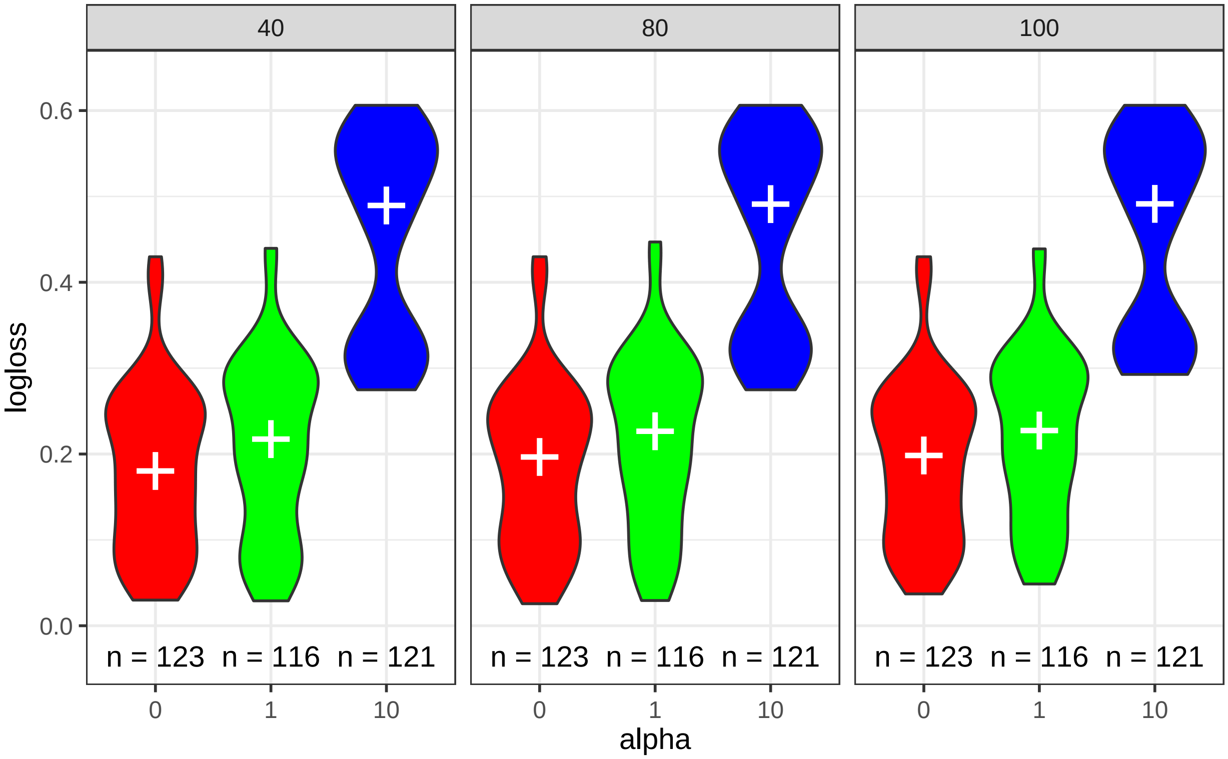
**all grid search results**

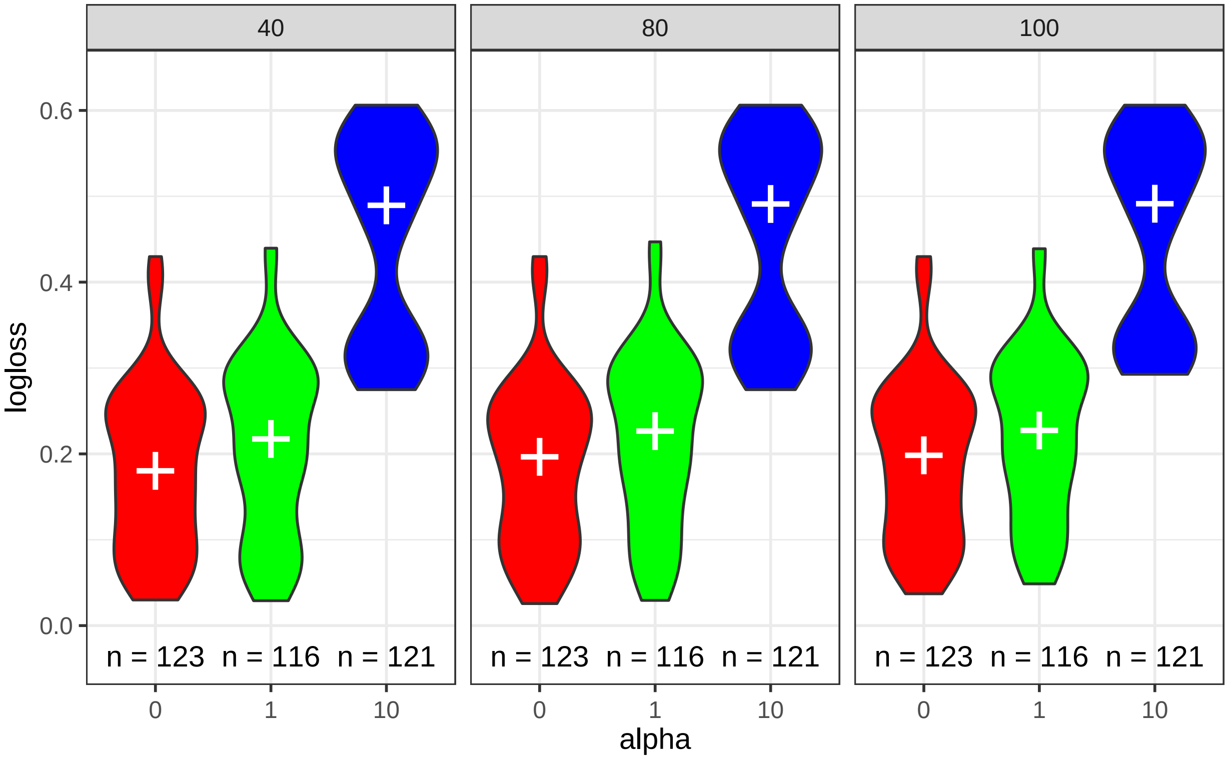


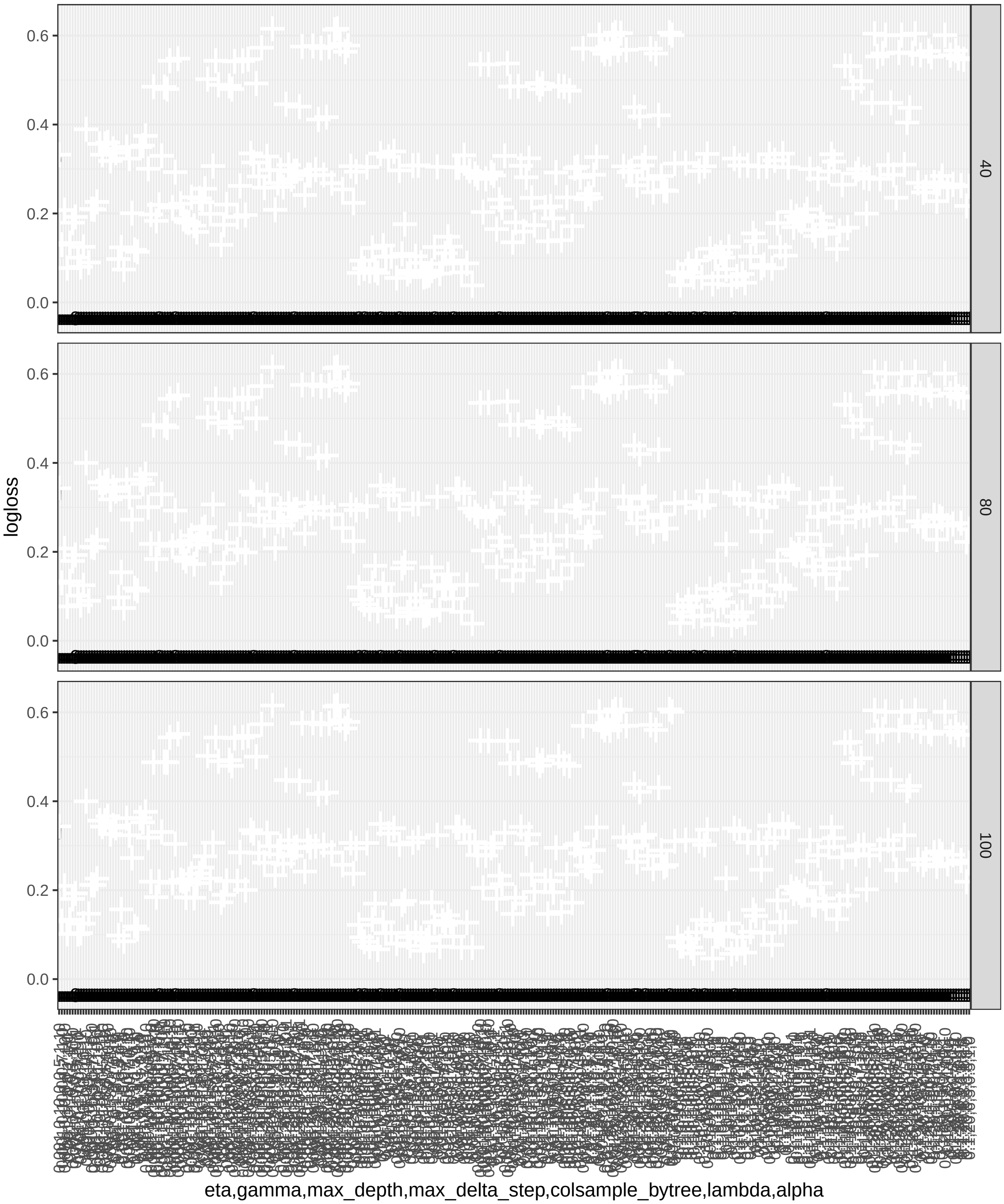

40
80
100
logloss
0.0
0.2
0.4
0.6
n = 135
n = 124
n = 101
1
10
6
max_depth


40
80
100
logloss
0.0
0.2
0.4
0.6
n = 143
n = 114
n = 103
0
3
5
max_delta_step


40
80
100
logloss
0.0
0.2
0.4
0.6
n = 116
n = 134
n = 110
0.5
0.7
1
colsample_bytree
n = 122
n = 135
n = 103
0
1
10
lambda


40
80
100
logloss
0.6
0.4
0.2
0.0
n = 123
n = 116
n = 121
n = 123
n = 116
n = 121
n = 123
n = 116
n = 121
0
1
10
0
1
10
0
1
10
alpha

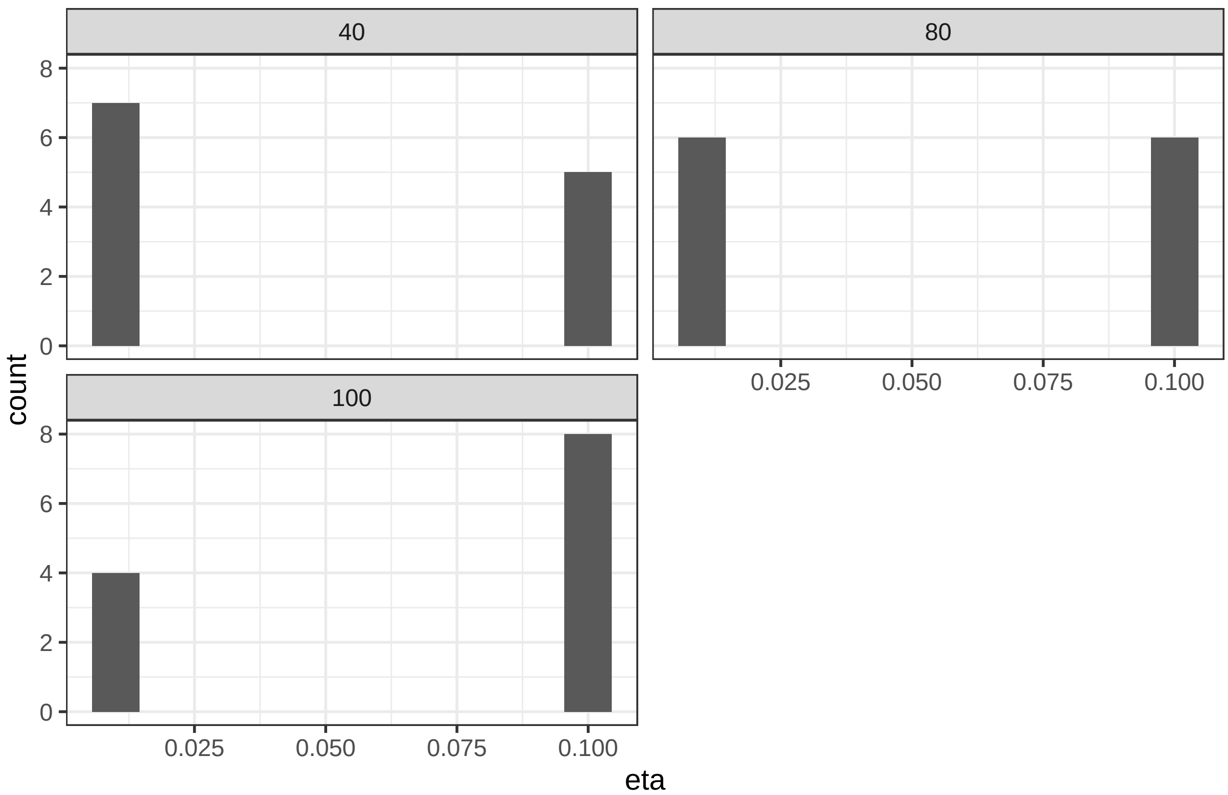

40
80
100
logloss
0.6
0.4
0.2
0.0
n = 123
n = 116
n = 121
n = 123
n = 116
n = 121
n = 123
n = 116
n = 121
0
1
10
0
1
10
0
1
10
alpha



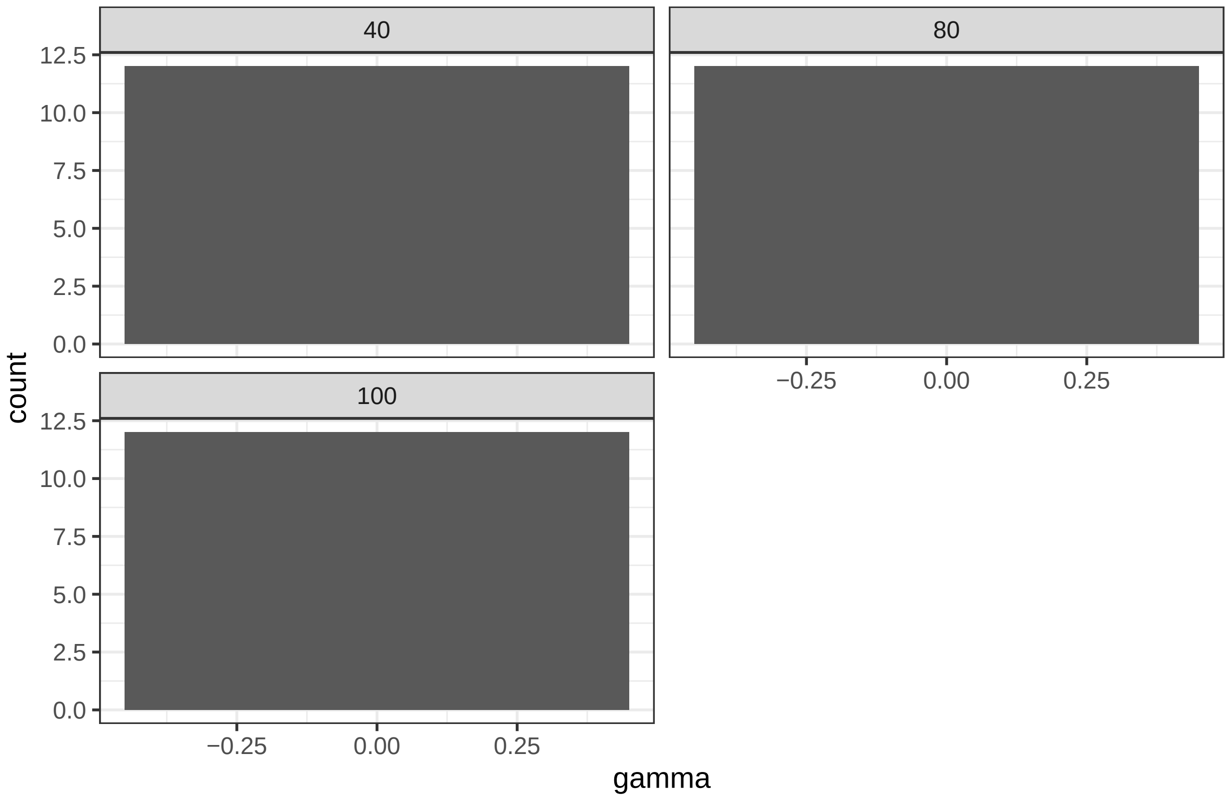
### over best parameter combo per cv

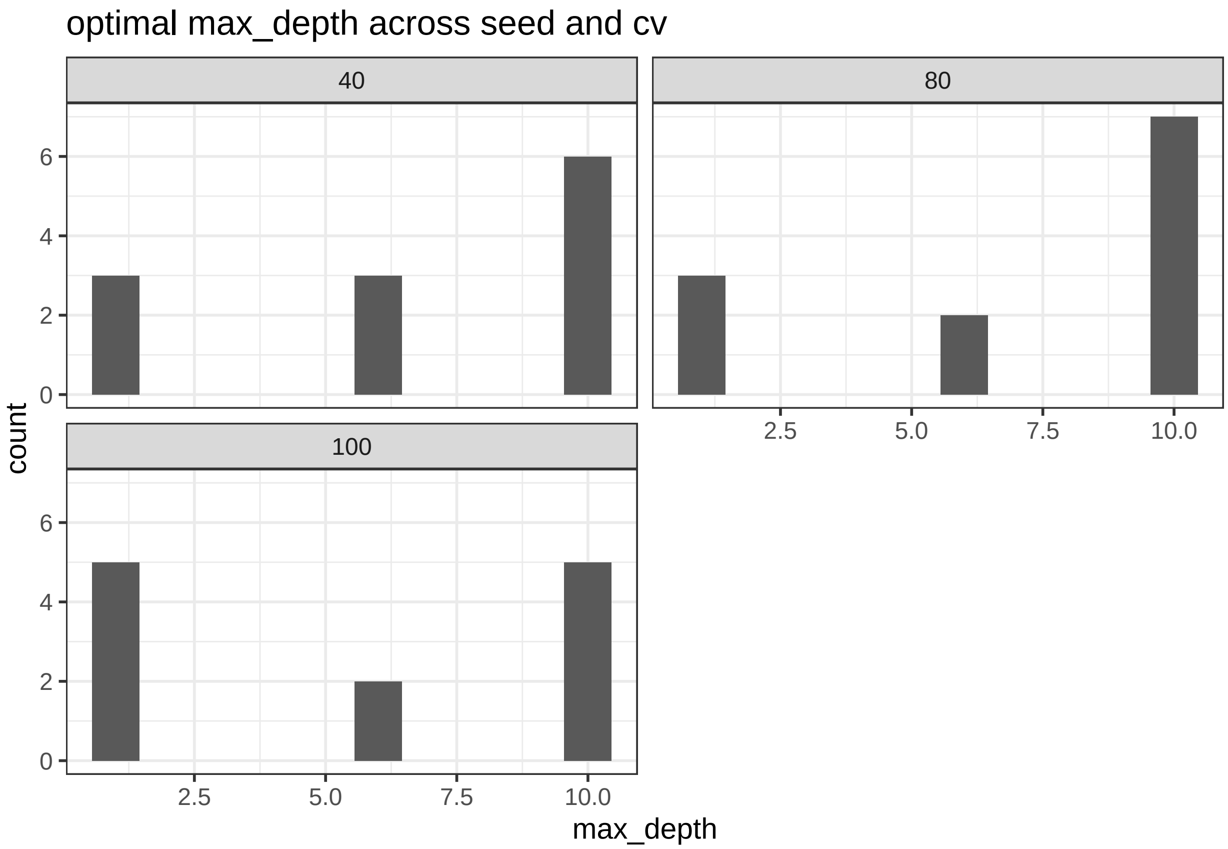
Note the 2nd /3rd best parameter combinations might not be too bad either.

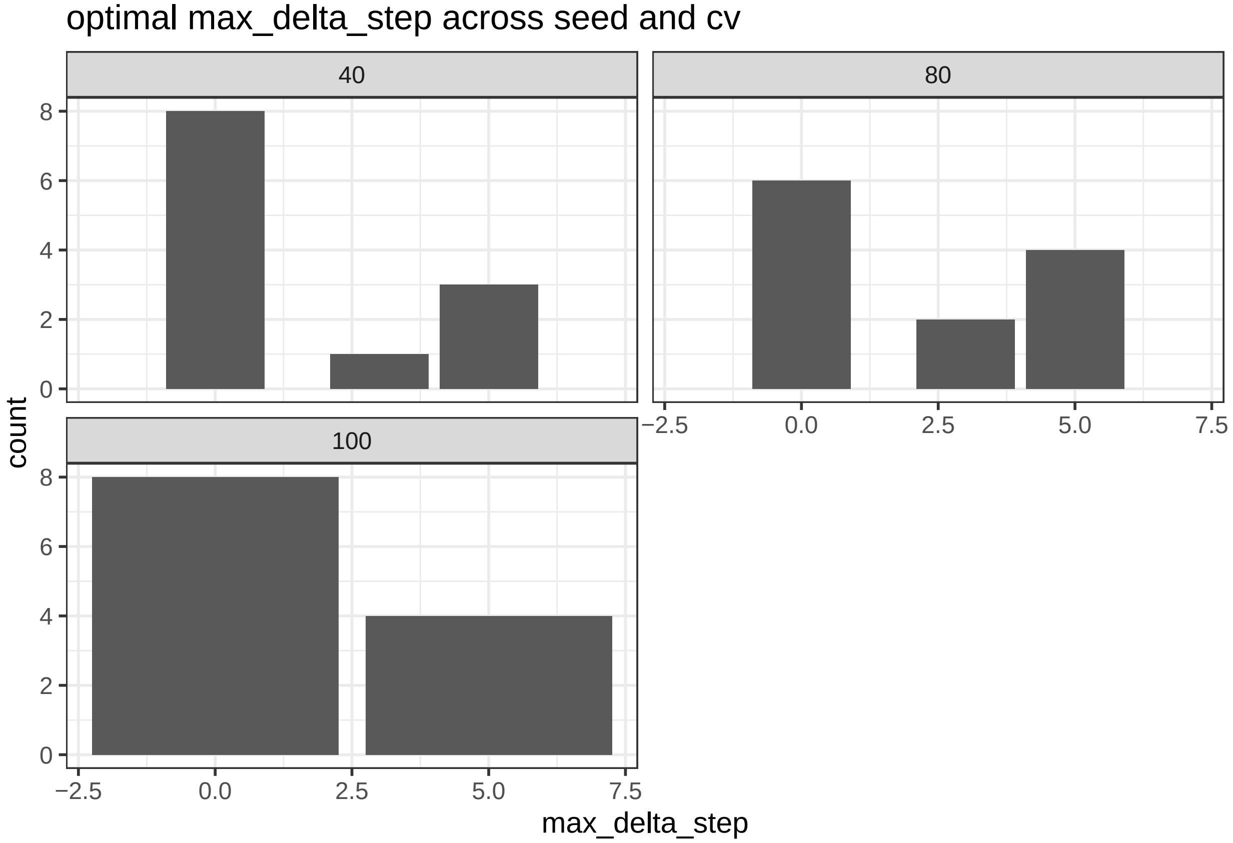
optimal eta across seed and cv



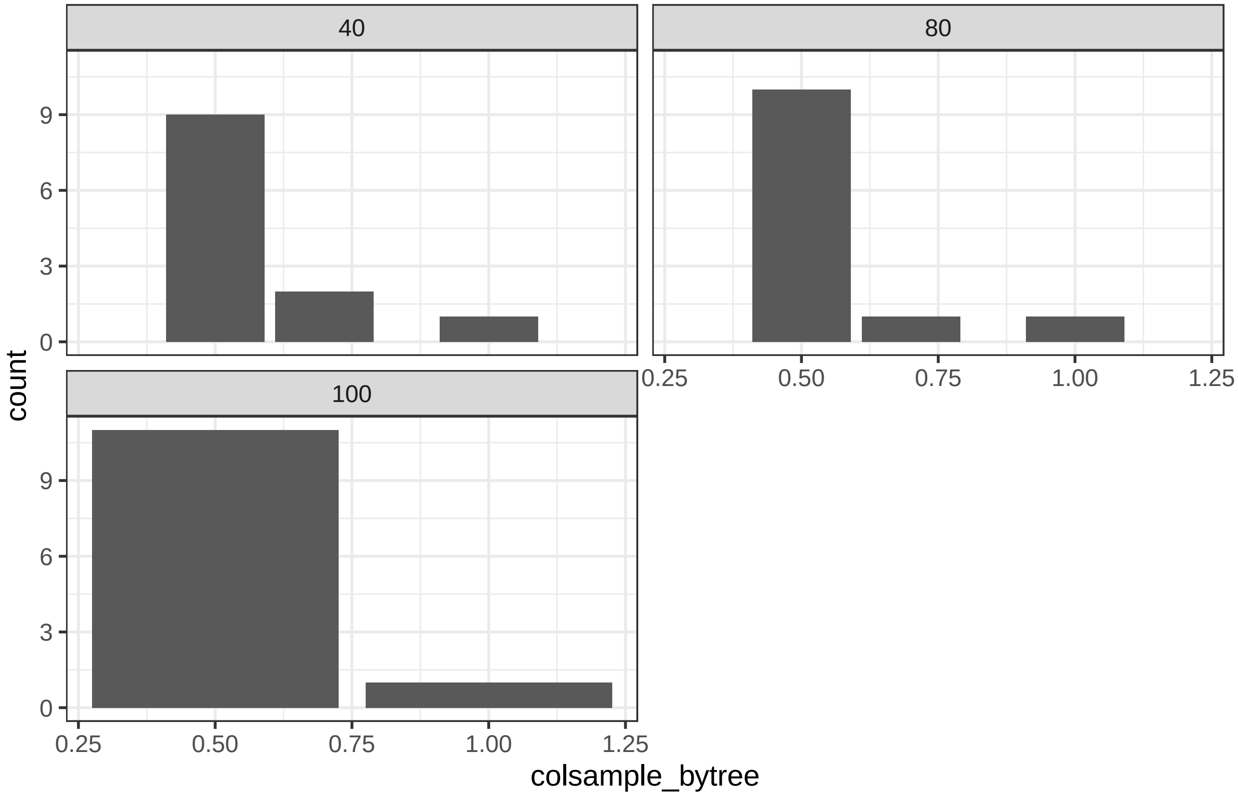
optimal gamma across seed and cv



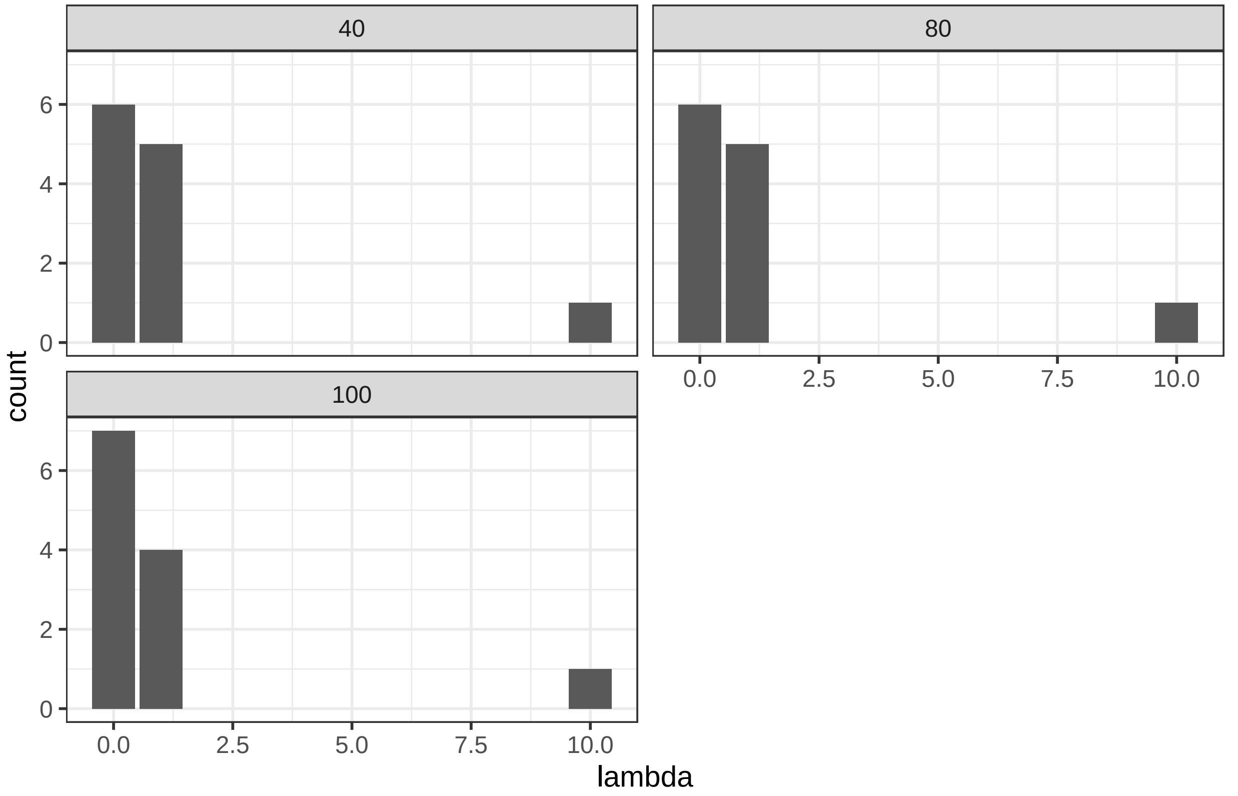

optimal max_depth across seed and cv
40
80
100
count
0
2
4
6
2.5
5.0
7.5
10.0
max_depth

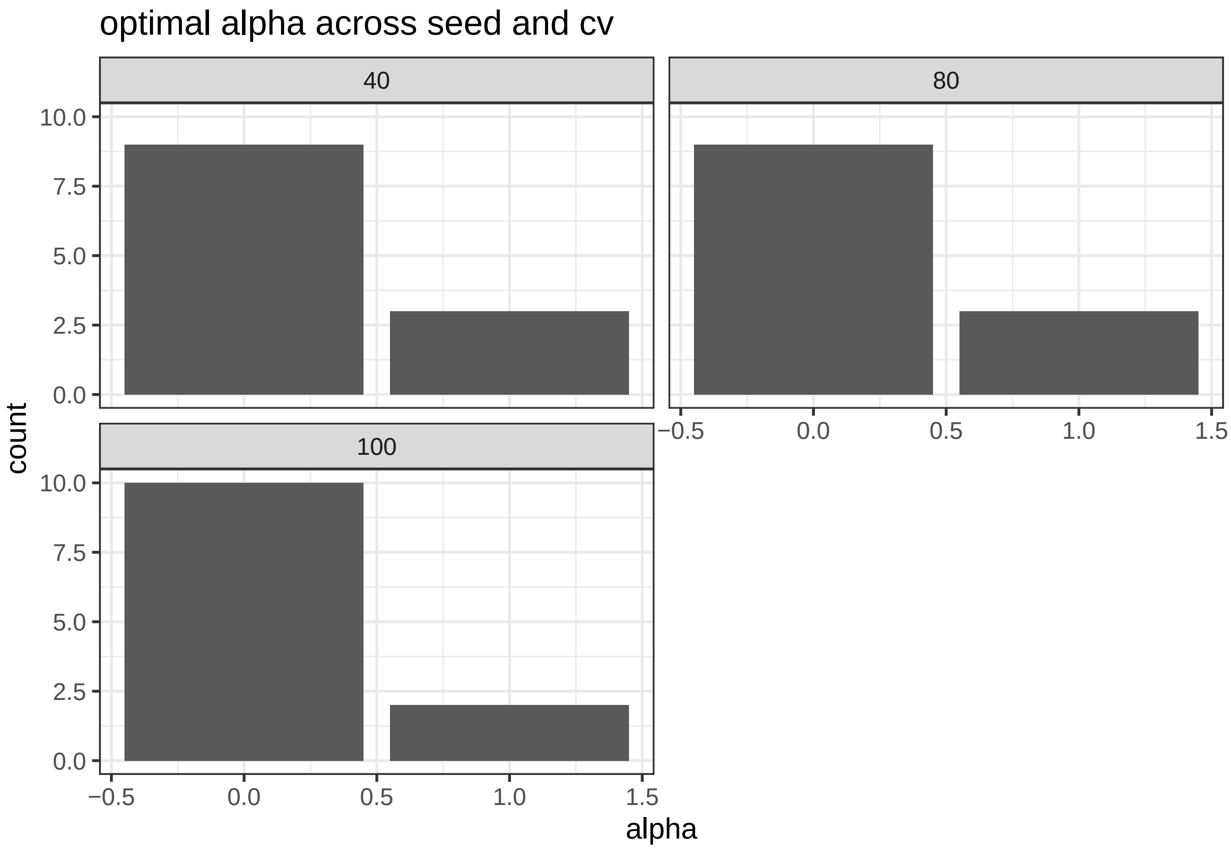

optimal max_delta_step across seed and cv
40
80
100
count
0
2
4
6
8
−2.5
0.0
2.5
5.0
7.5
max_delta_step

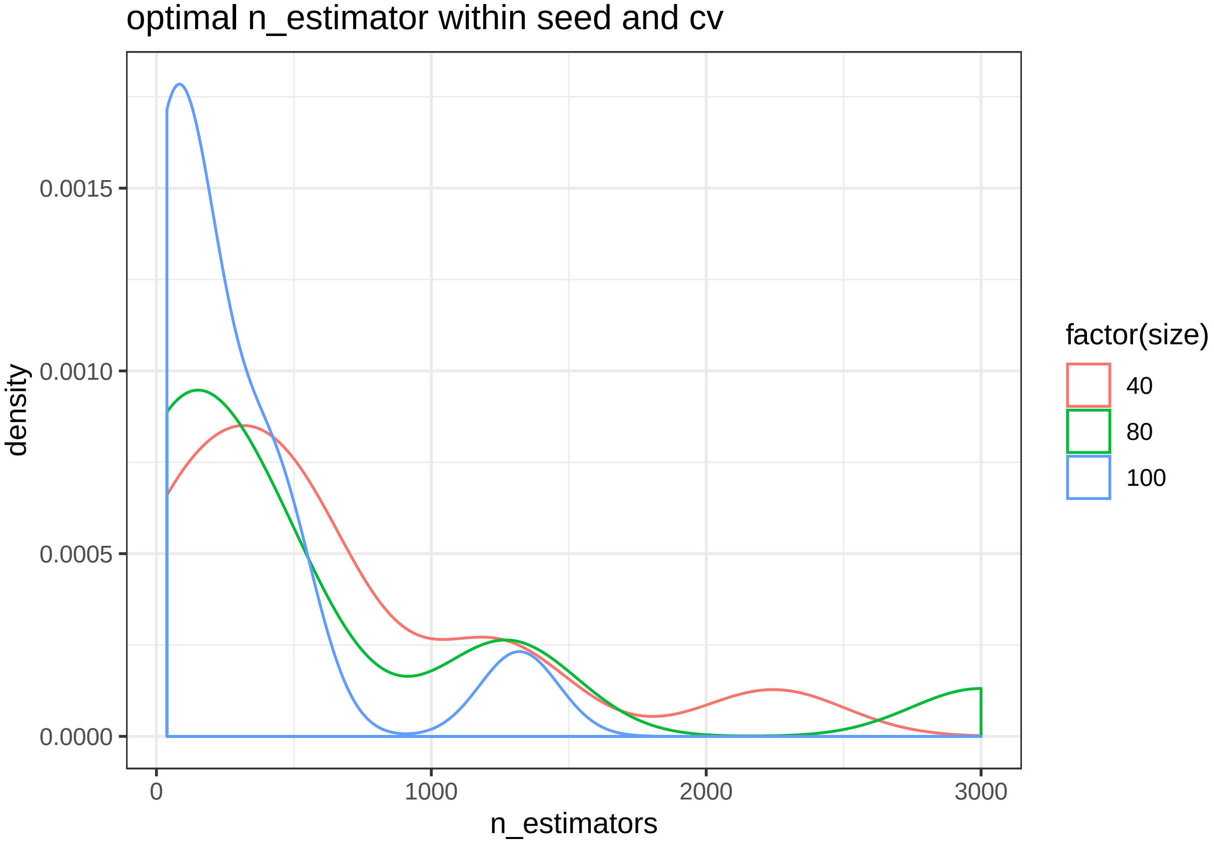
optimal colsample_bytree across seed and cv



optimal lambda across seed and cv




optimal alpha across seed and cv


optimal n_estimator within seed and cv

### more about the best parameter combination selection

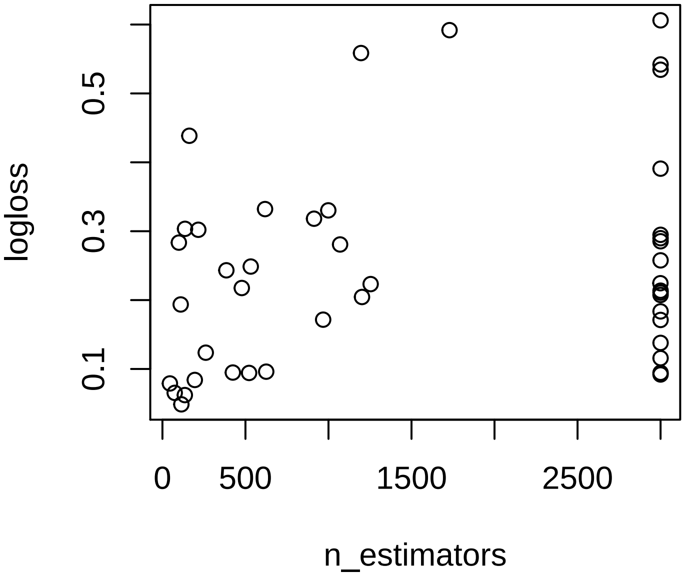
```
select_ft_step <- 100

df1 <- subset(grid_best, size==select_ft_step & max_depth==1 &  max_delta_step == 0 )
print( paste('summary of n estimator at',select_ft_step, 'feature step'))
```

```
## [1] "summary of n estimator at 100 feature step"
```

```
print(summary(df1$n_estimators))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   45.00   66.75  104.50  112.25  150.00  195.00
```

```
df2 <- subset(df.grid, size==select_ft_step & max_depth==1 &  max_delta_step == 0 )

with(df2, plot(x = n_estimators, y=score, ylab=score_label))
```


logloss
n_estimators