ADS 503 Final Project

Claire Phibbs, Christopher Richardson, Martin Zagari

2022-06-17

The Data

Our target variable "Risk Performance" along with the first 13 predictor columns

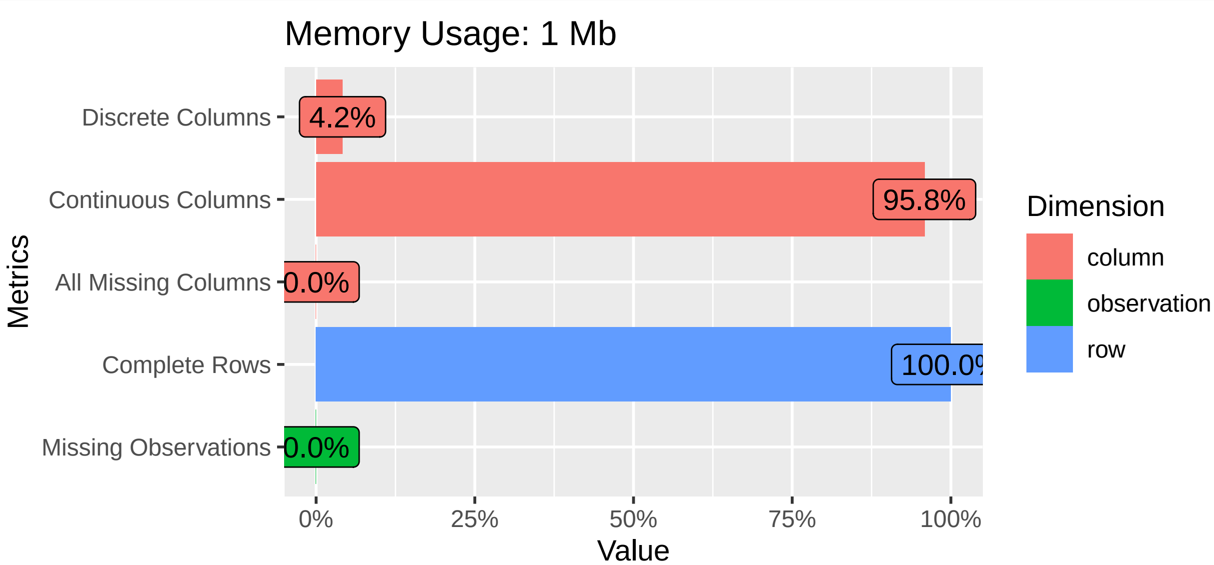
```
file_loc <- '/Volumes/GoogleDrive/My Drive/503/Project 503/Fico Data/heloc_dataset_v1.csv'
heloc <- read.csv(file_loc)

# sort col names for readability purposes
heloc <- heloc[ , order(names(heloc))]

knitr::kable(heloc[1:4,c(24,1:13)]) %>%
  kableExtra::kable_styling("striped", full_width = F) %>%
  kableExtra::row_spec(0, angle = -90)
```

Predictor columns 14 to 23

```
knitr::kable(heloc[1:4,c(24,14:23)]) %>%
  kableExtra::kable_styling("striped", full_width = F) %>%
  kableExtra::row_spec(0, angle = -90)
```

Data Pre-Processing

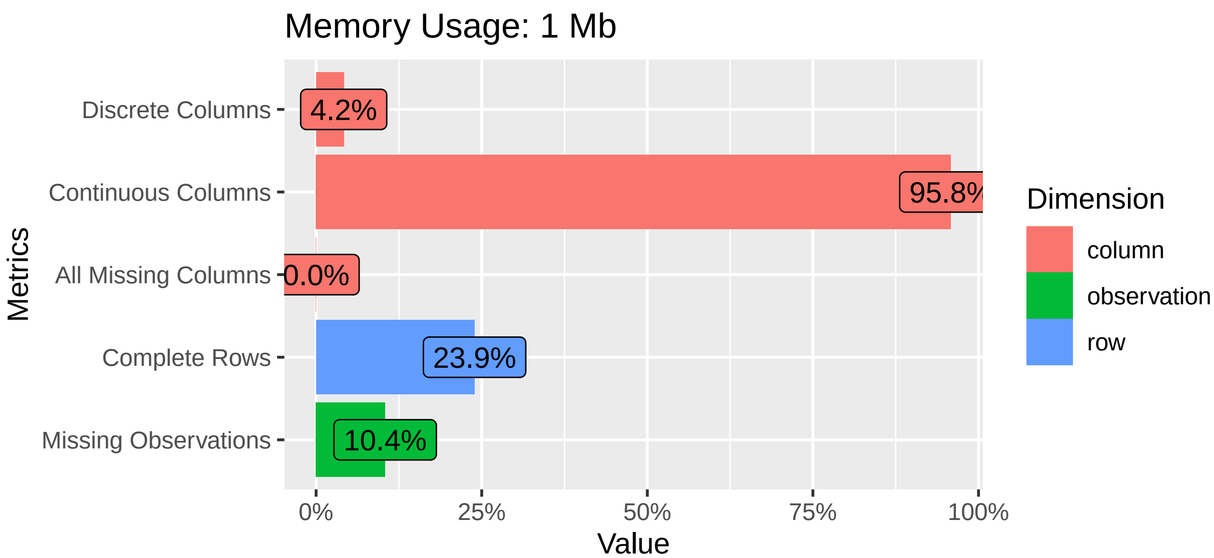
```
DataExplorer::plot_intro(heloc)
```



RiskPerformance	AverageMInFile	ExternalRiskEstimate	MaxDelq2PublicRecLast12M	MaxDelqEver	MSinceMostRecentDelq	MSinceMostRecentInqexcl7days	MSinceMostRecentTradeOpen	MSinceOldestTradeOpen	NetFractionInstallBurden	NetFractionRevolvingBurden	NumBank2NatlTradesWHighUtilization	NumInqLast6M	NumInqLast6Mexcl7days
Bad	84	55	3	5	2	0	4	144	-8	33	1	0	0
Bad	41	61	0	8	-7	0	15	58	-8	0	-8	0	0
Bad	24	67	7	8	-7	0	5	66	66	53	1	4	4
Bad	73	66	6	6	76	0	1	169	83	72	3	5	4

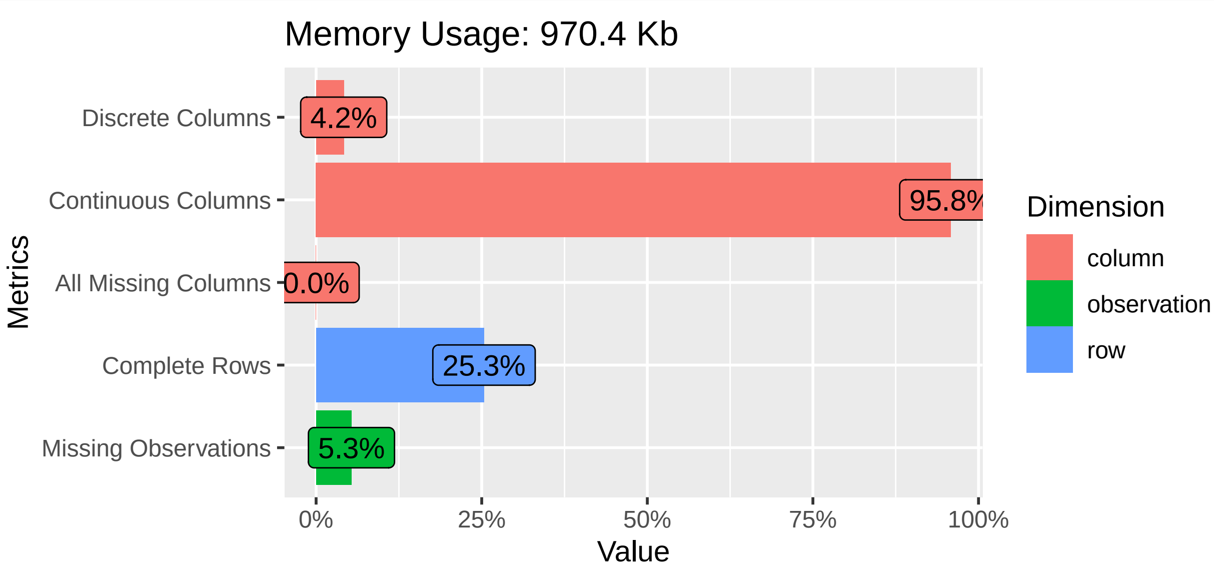
RiskPerformance	NumInstallTradesWBalance	NumRevolvingTradesWBalance	NumSatisfactoryTrades	NumTotalTrades	NumTrades60Ever2DerogPubRec	NumTrades90Ever2DerogPubRec	NumTradesOpeninLast12M	PercentInstallTrades	PercentTradesNeverDelq	PercentTradesWBalance
Bad	1	8	20	23	3	0	1	43	83	69
Bad	-8	0	2	7	4	4	0	67	100	0
Bad	2	4	9	9	0	0	4	44	100	86
Bad	4	6	28	30	1	1	3	57	93	91

```r
# -9 = No Credit History
# -8 and -7 = No recent activity
heloc[heloc == -9] <- NA
heloc[heloc == -8] <- NA
heloc[heloc == -7] <- NA
DataExplorer::plot_intro(heloc)
```

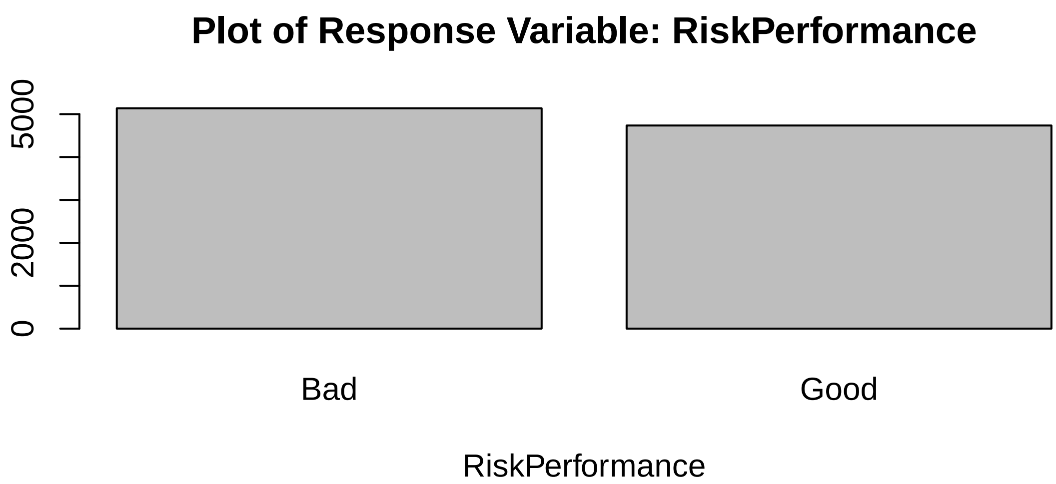


It appears that roughly 76.1% of our data had one or more predictors had a -9,-8,-7.

```r
# removing missing values from rows that span across all columns (588 values)
heloc_No_NA <- heloc %>% dplyr::filter_at(vars(-RiskPerformance),
                                          any_vars(!is.na(.)))
DataExplorer::plot_intro(heloc_No_NA)
```




```r
# bar plot of response variable; RiskPerformance
barplot(table(heloc_No_NA$RiskPerformance),
        main="Plot of Response Variable: RiskPerformance",
        xlab="RiskPerformance")
```




```
table(heloc_No_NA$RiskPerformance)
```

```
##
##  Bad Good
## 5136 4735
```

A 51:47 split! Nearly a 50:50 balance!

```
# create training indices
set.seed(3)
heloc_training <- caret::createDataPartition(heloc_No_NA$RiskPerformance,
                                             p=0.8,
                                             list=FALSE)

# training/set sets
heloc_train <- heloc_No_NA[heloc_training, ]
heloc_test <- heloc_No_NA[-heloc_training, ]

# knn imputation
heloc_impute <- caret::preProcess(heloc_train,
                                  method = 'knnImpute')

heloc_train <- stats::predict(heloc_impute,
                              newdata=heloc_train)

heloc_test <- stats::predict(heloc_impute,
                             newdata=heloc_test)

# remove highly correlated predictors
high_corr <- caret::findCorrelation(stats::cor(heloc_train[, -24]),
                                    0.85)

# removal of high cor predictors
heloc_train <- heloc_train[, -(high_corr)]
heloc_test <- heloc_test[, -(high_corr)]
names(heloc_test[high_corr])
```

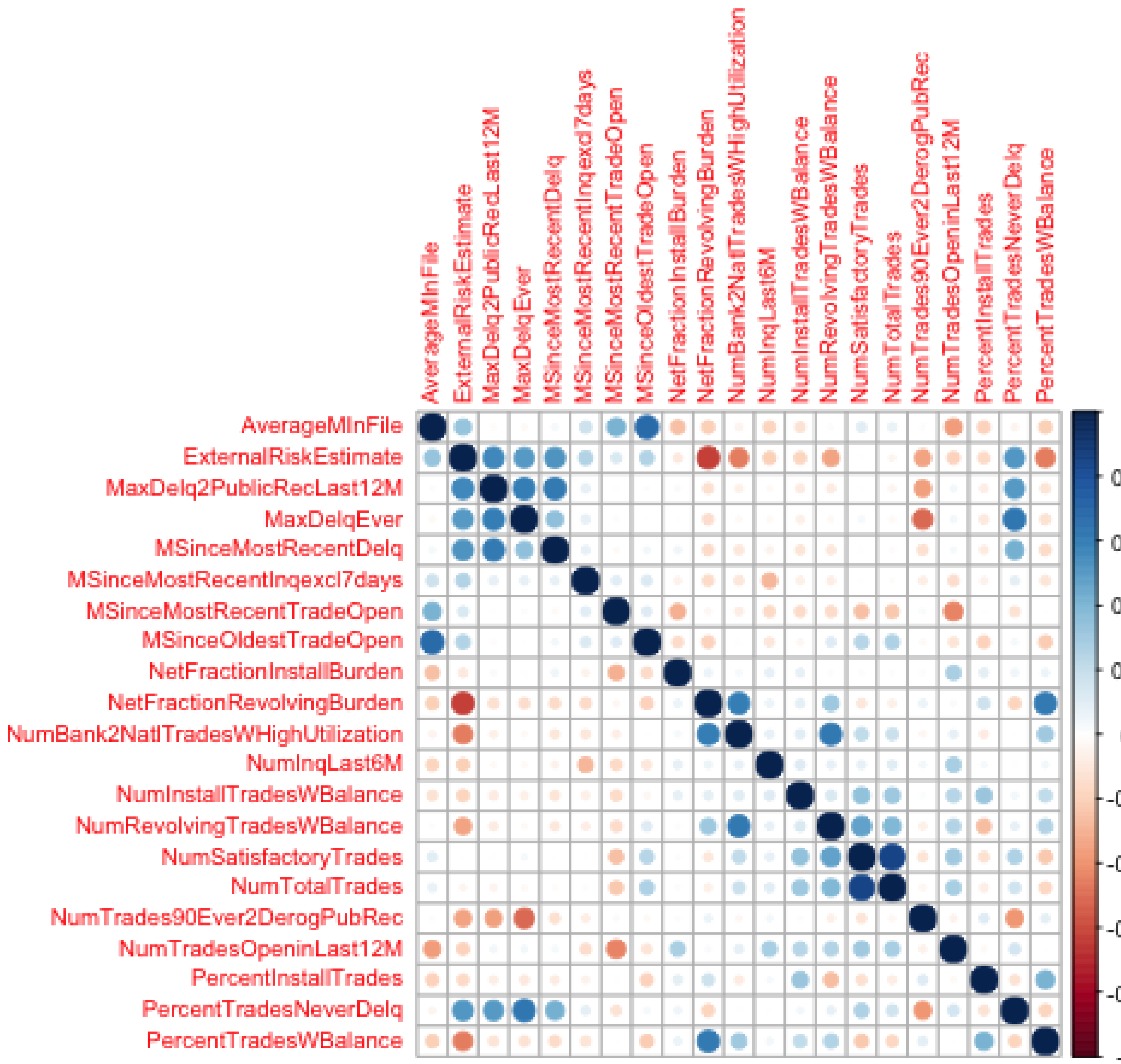
```
## [1] "NumTradesOpeninLast12M"   "NumInstallTradesWBalance"
```

We keep the heloc_train/test dataframes for formula based functions.

```r
no_risk_bool <- names(heloc_train) != 'RiskPerformance'

# x = predictors
heloc_train_x <- heloc_train[,no_risk_bool]
heloc_test_x <- heloc_test[,no_risk_bool]

# y = response/target
heloc_train_y <- heloc_train[,'RiskPerformance']
heloc_test_y <- heloc_test[,'RiskPerformance']
```

EDA

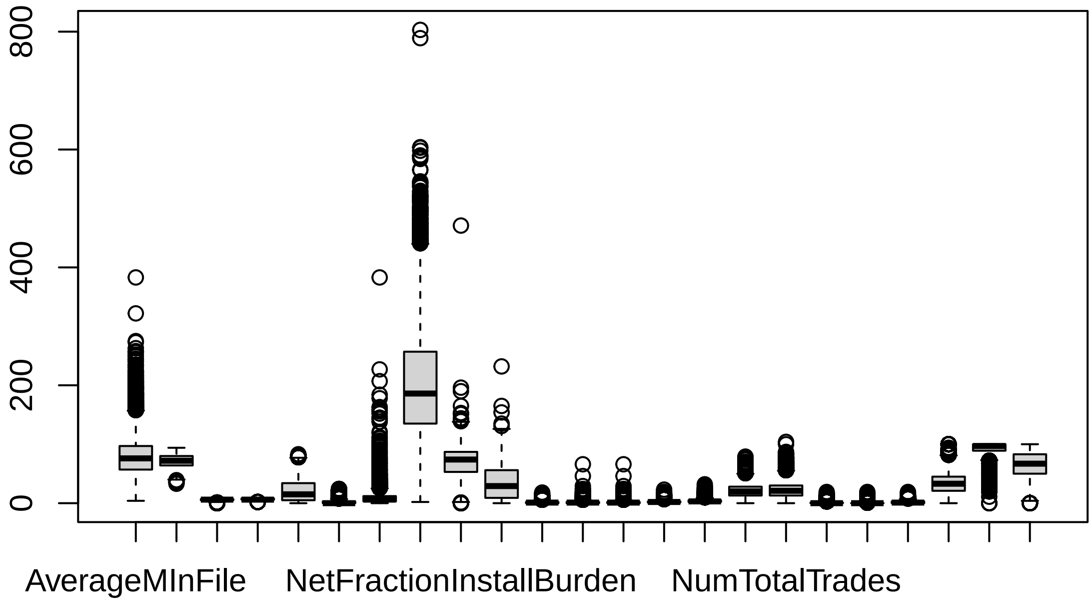
```r
# correlations
png(file="Corr_Plot.png")
# correlations
corrplot::corrplot(stats::cor(heloc_train),
                   number.cex = 0.5,
                   tl.cex = 0.8)
dev.off()
```



Correlation Plot

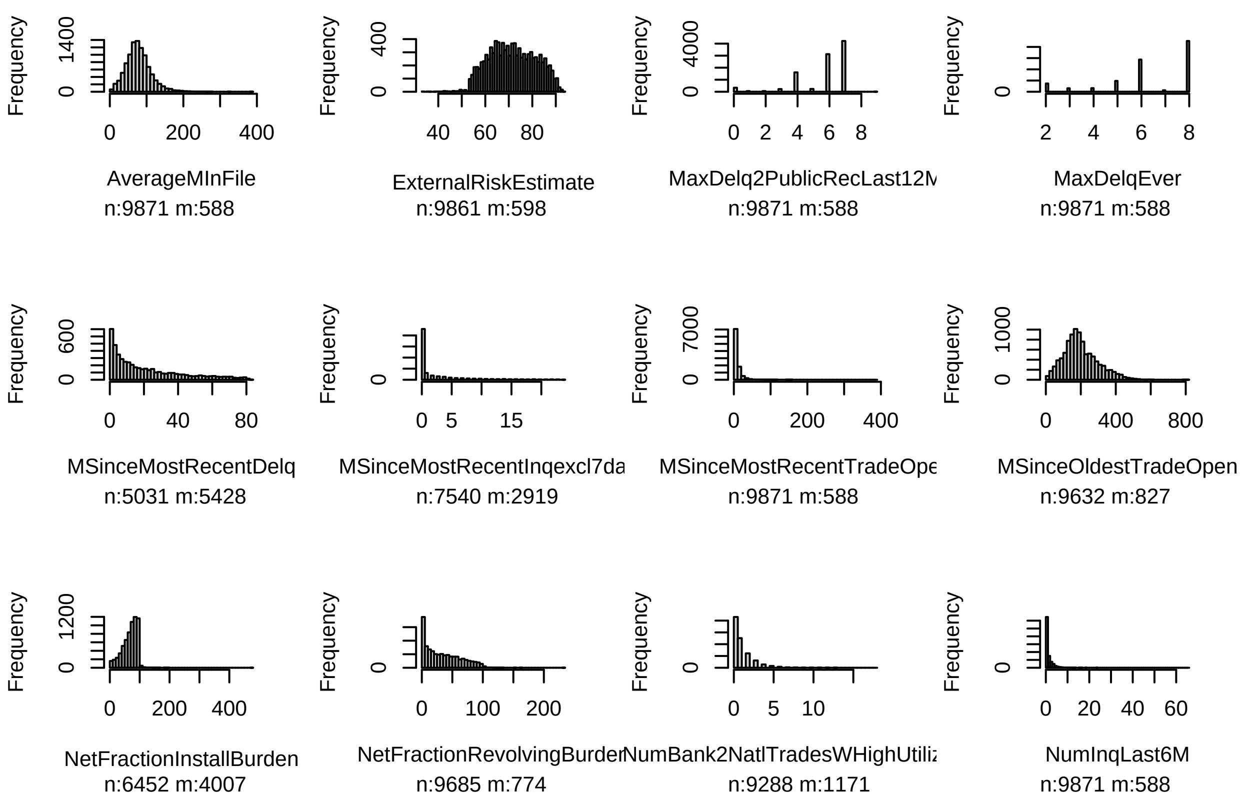
Histograms

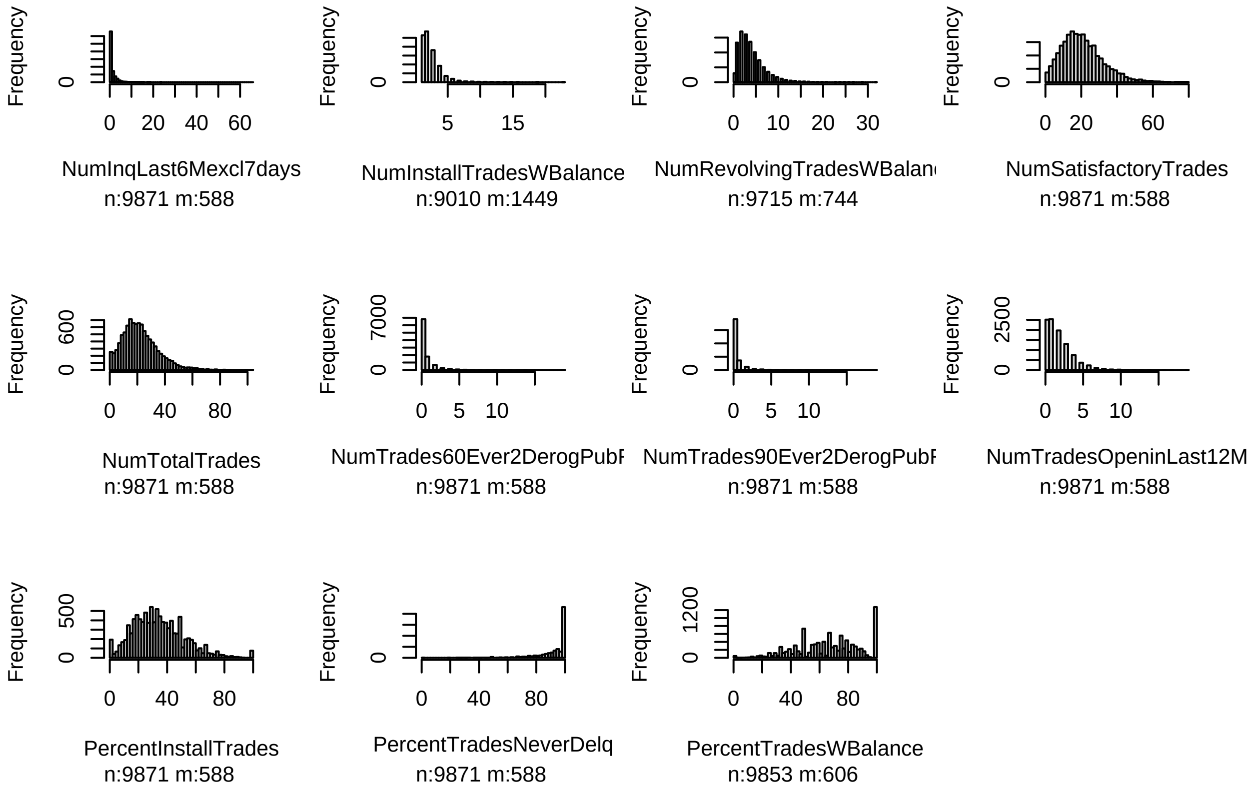
```
# boxplot to view outliers
boxplot(heloc[, 1:23])
```



Overall

```
# histograms to view predictor variable frequencies
par(mfrow=c(3,4))
Hmisc::hist.data.frame(heloc[,1:23])
```

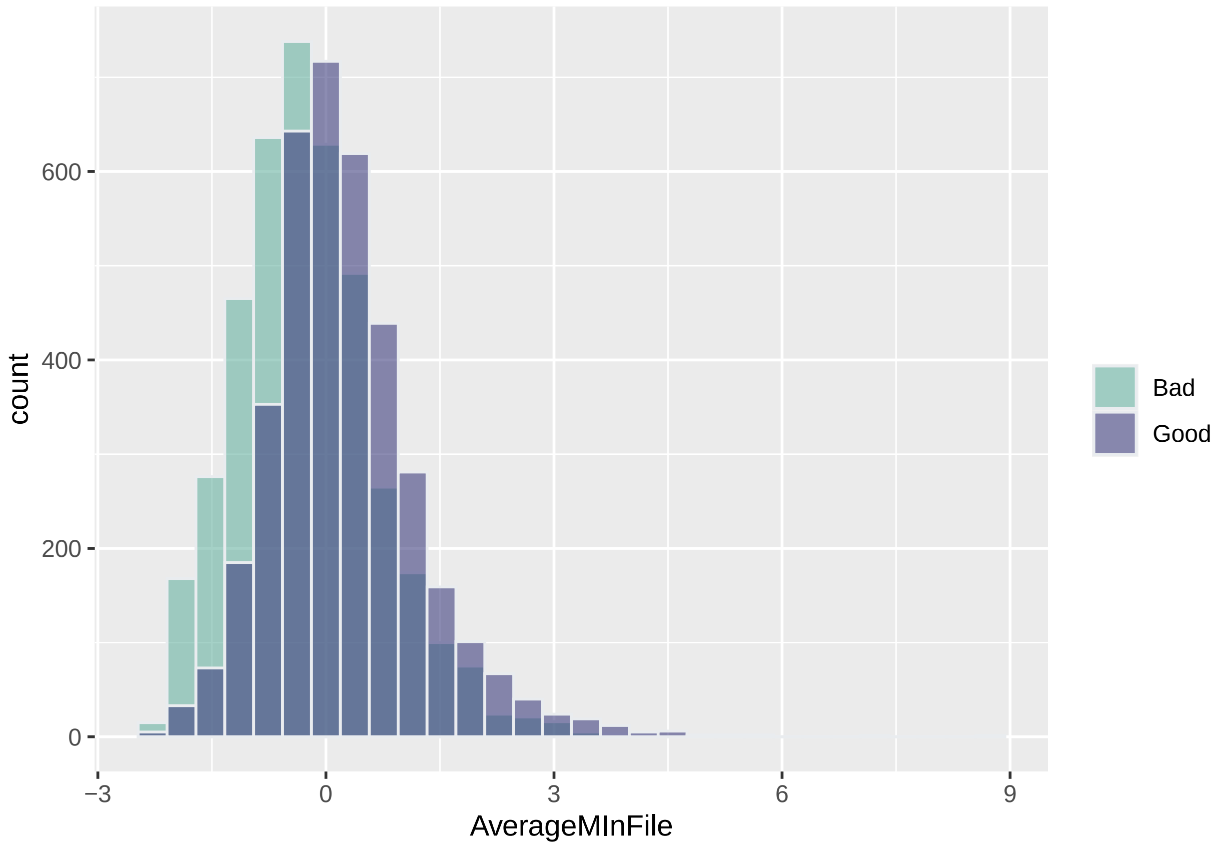




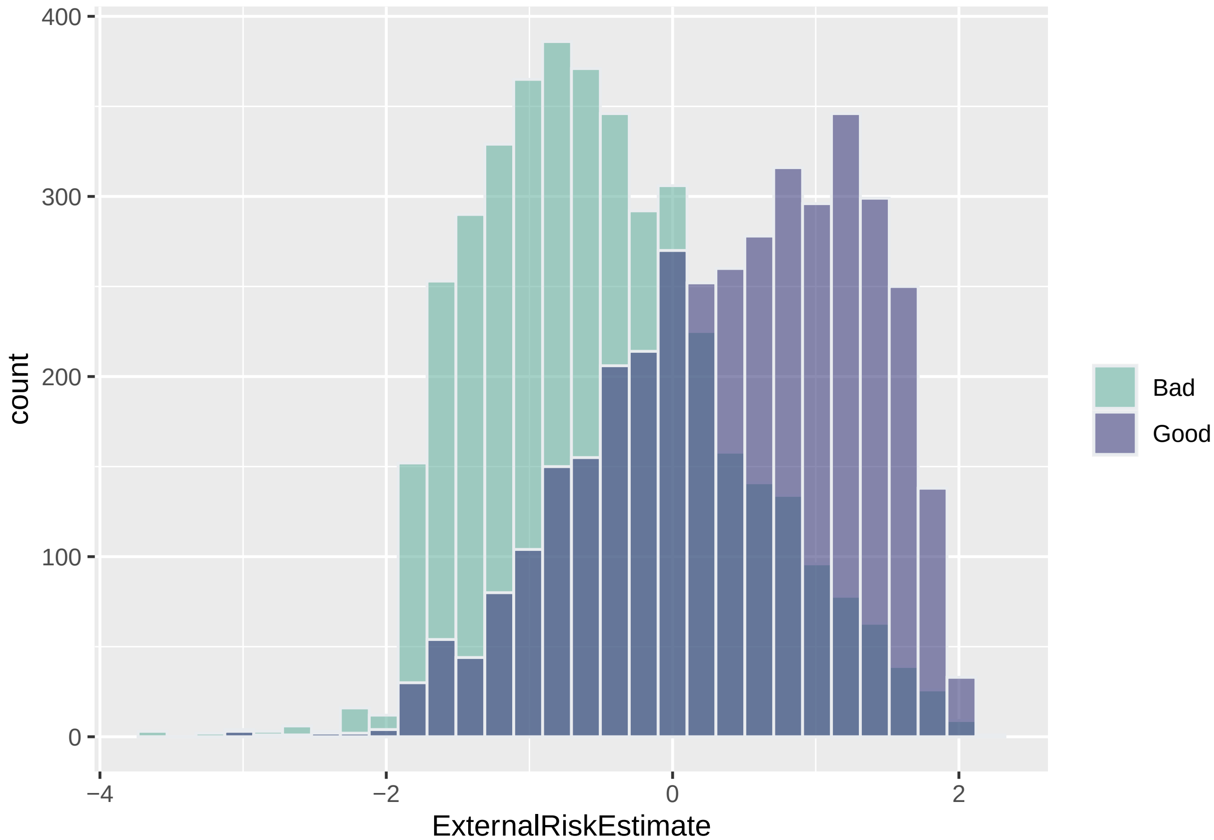
By Outcome

```
# Deselect Bool Outcome/Response/Target variable
no_risk_bool <- names(heloc_train) != 'RiskPerformance'

# heloc_imputed_full_set <- dplyr::as_tibble(heloc_imputed_full_set)
heloc_ifs_names <- colnames(heloc_train[,no_risk_bool])

# empty list to gather all
plot_list <- list()
for (name in heloc_ifs_names){
  p <- heloc_train %>%
    ggplot( aes(x=heloc_train[,name], fill=RiskPerformance)) +
      geom_histogram( color="#e9ecef", alpha=0.6, position = 'identity') +
      scale_fill_manual(values=c("#69b3a2", "#404080")) +
      xlab(name) +
      labs(fill="")
  print(p)
}
```

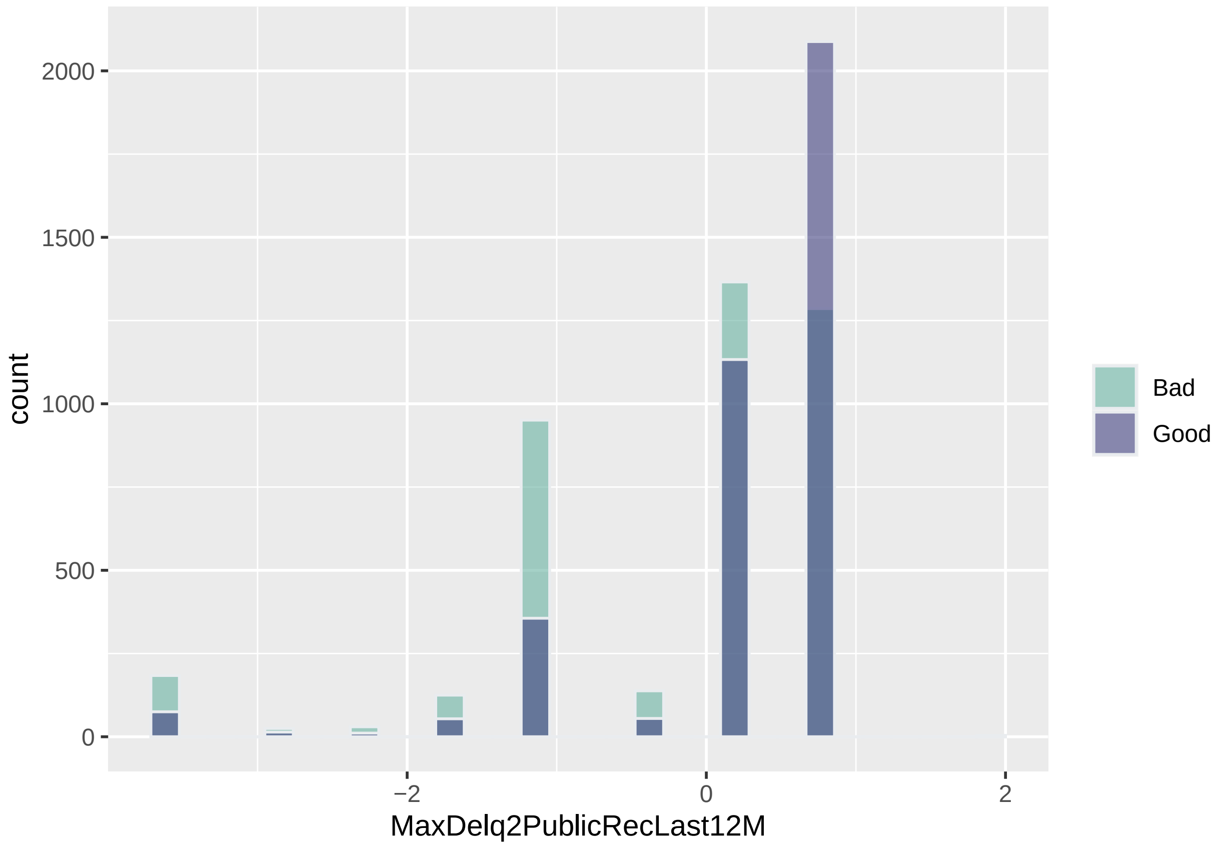
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



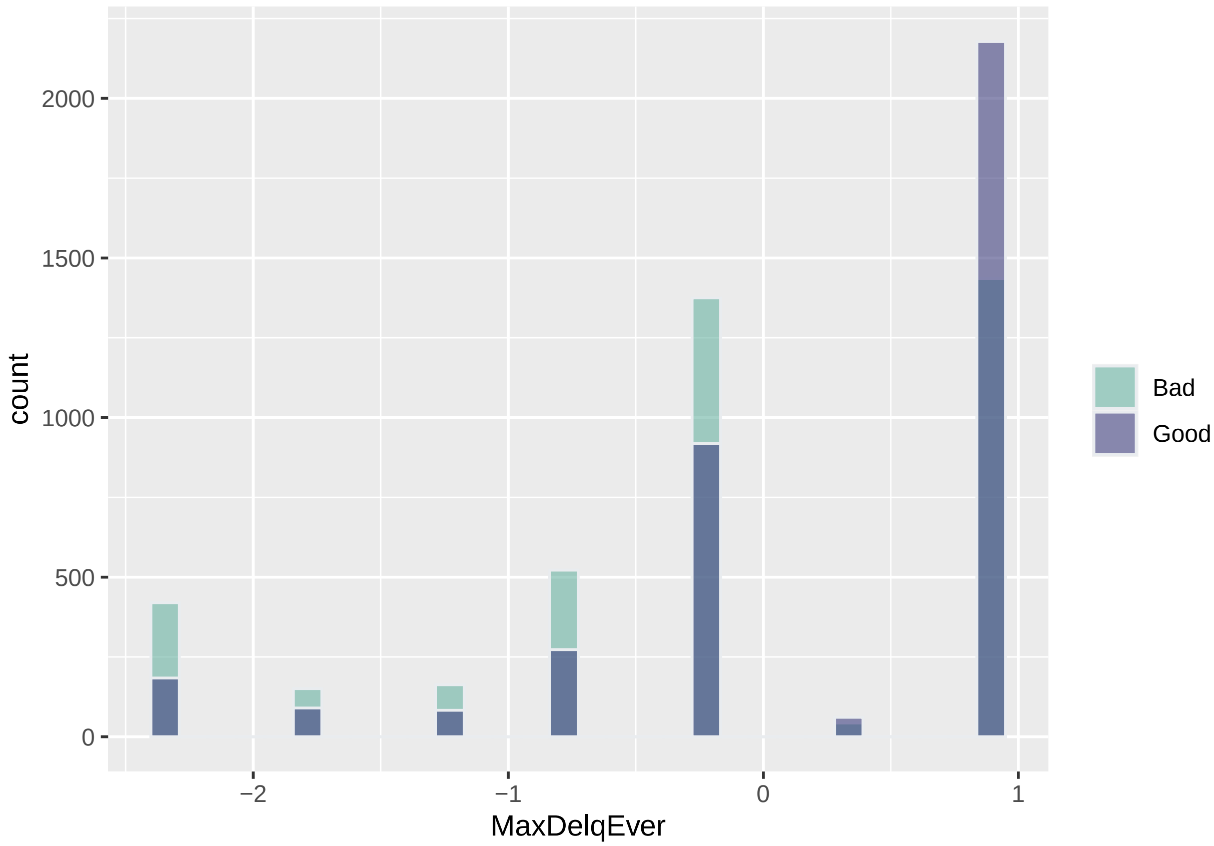

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



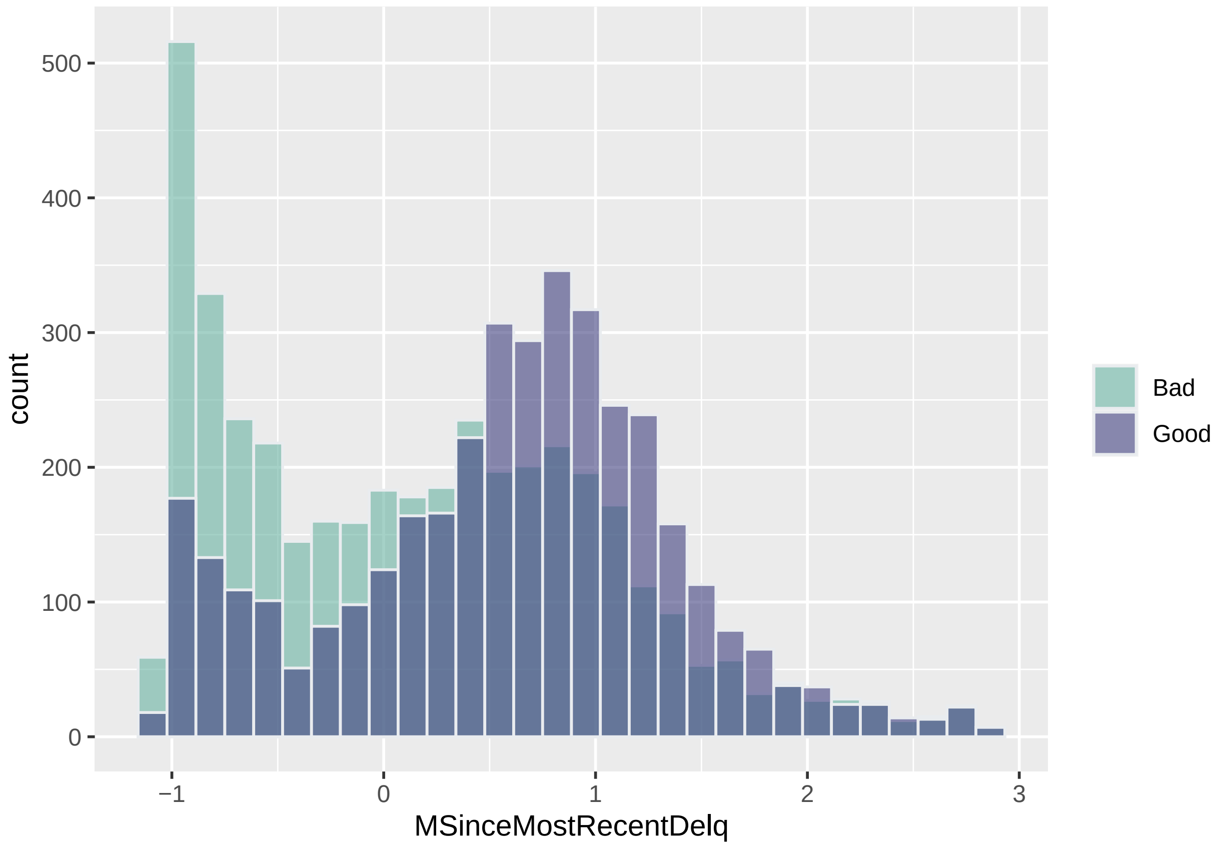

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



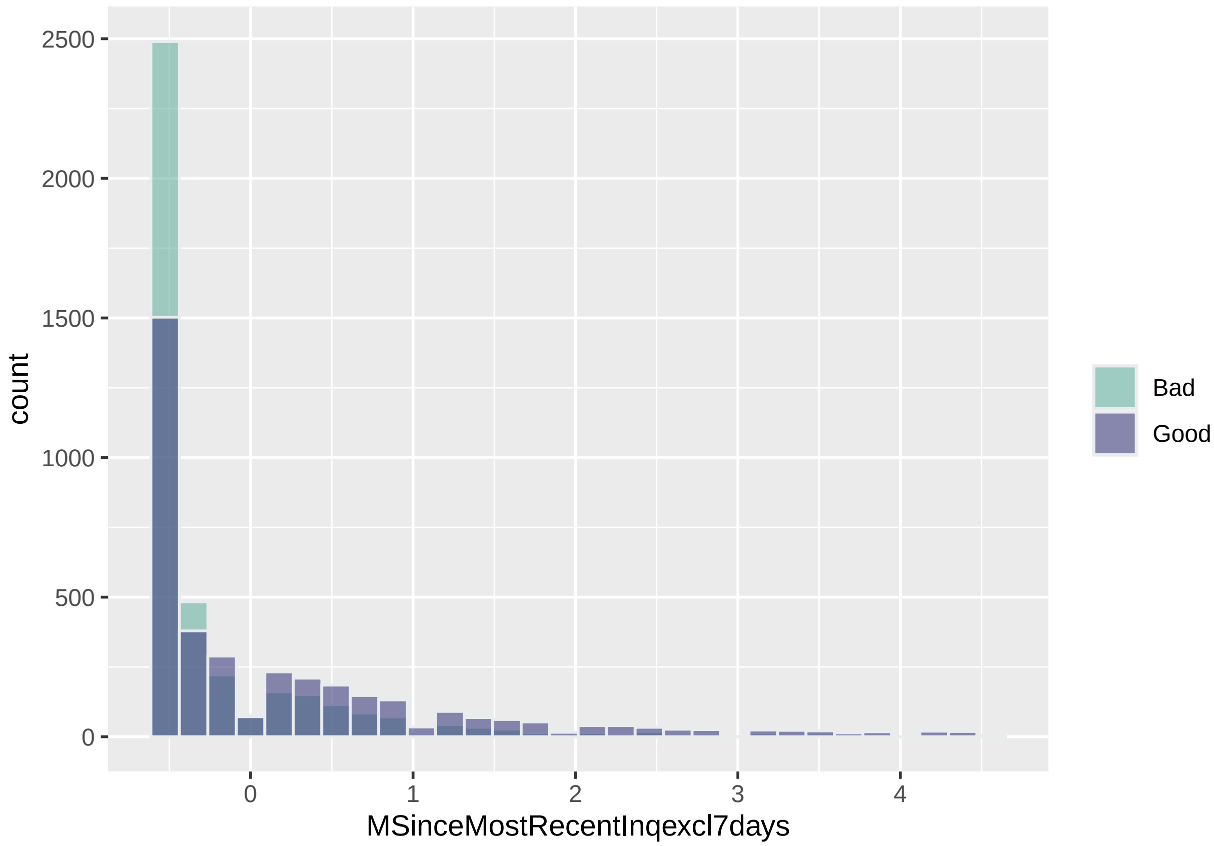

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



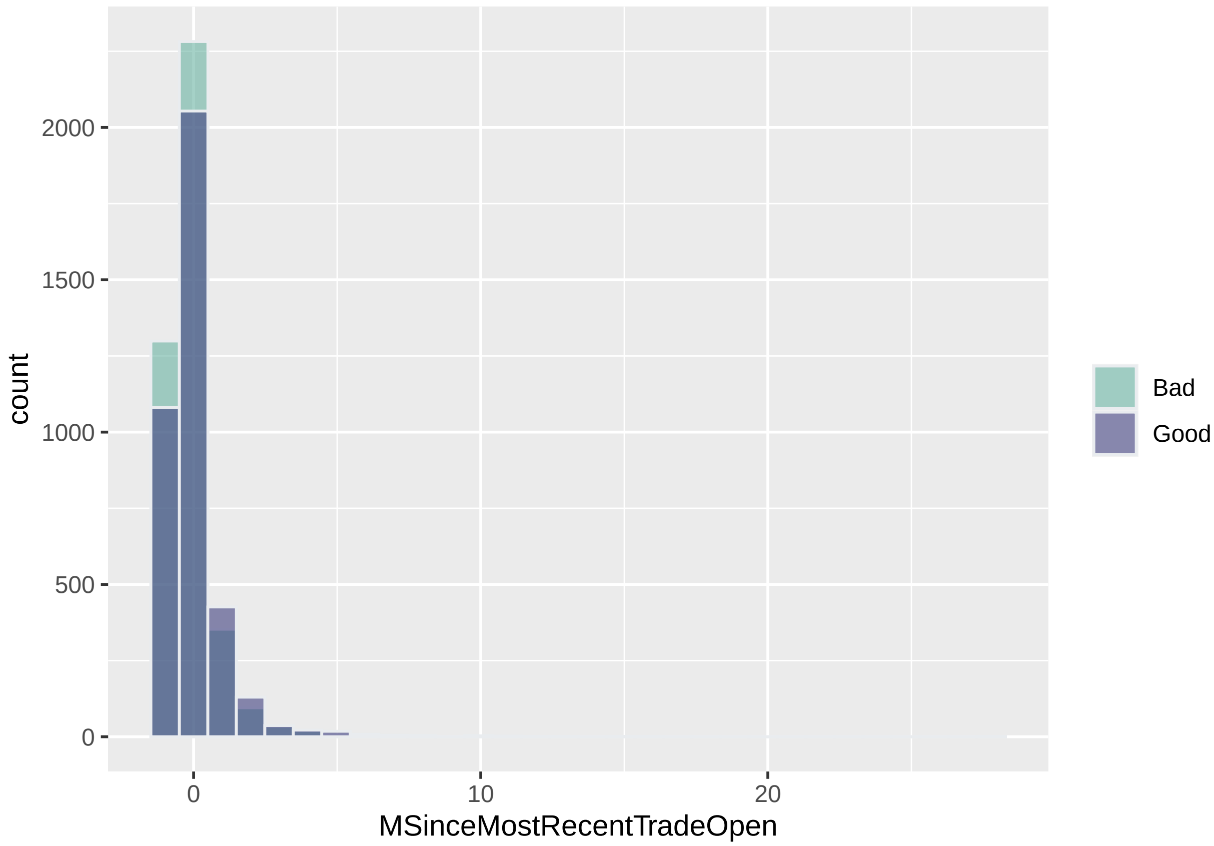

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



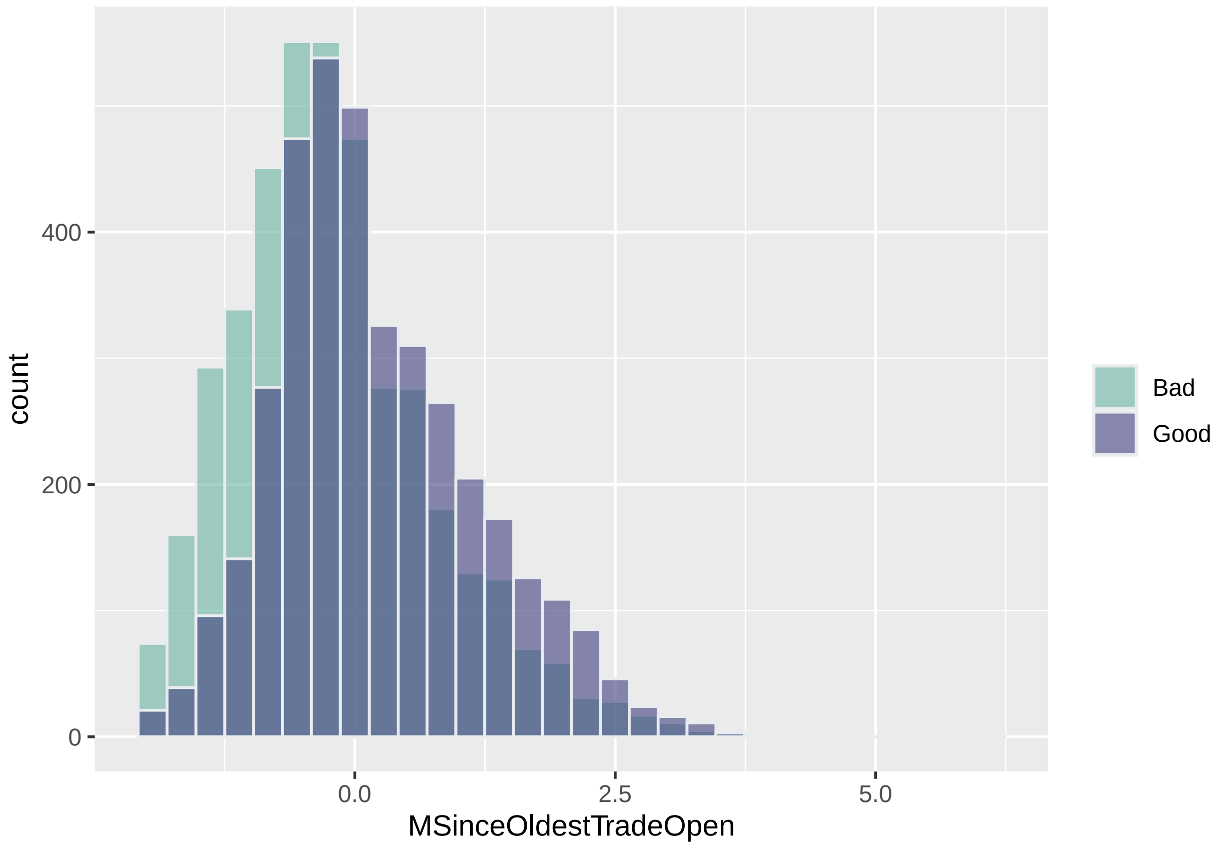

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



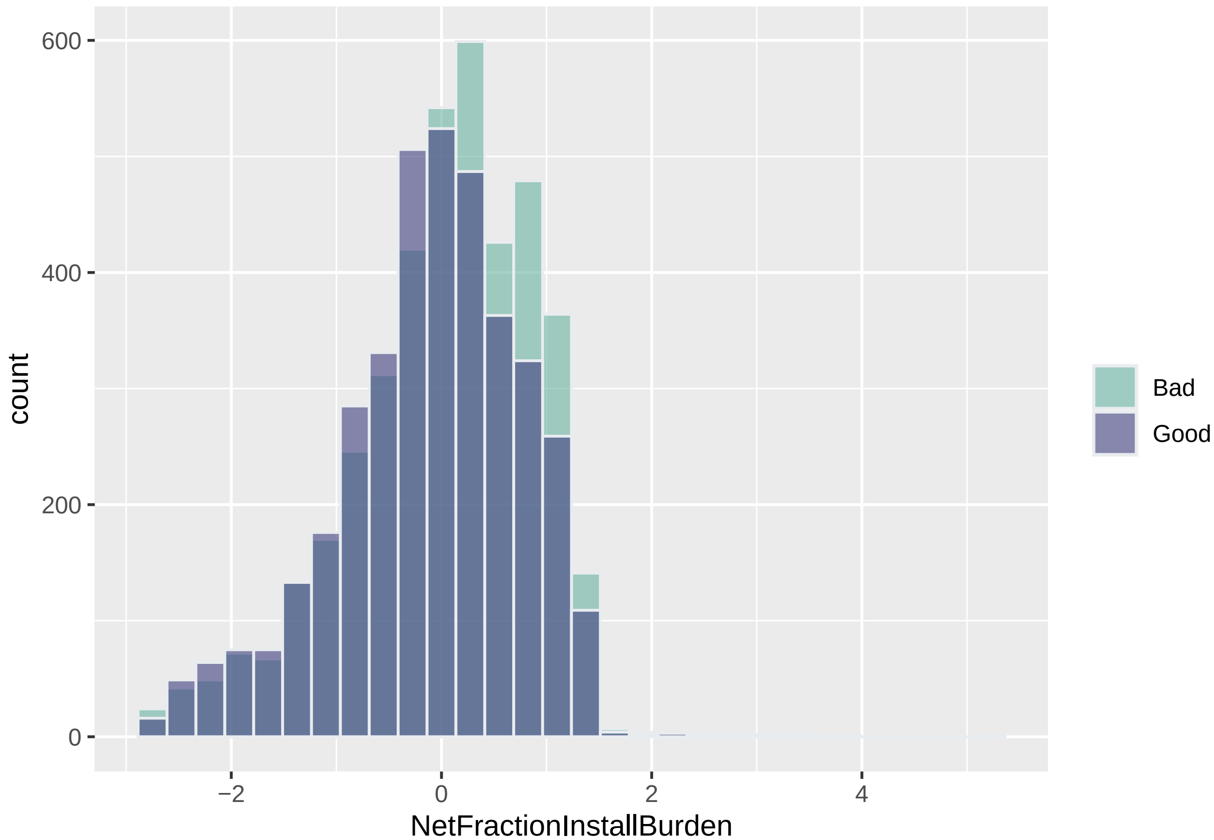

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



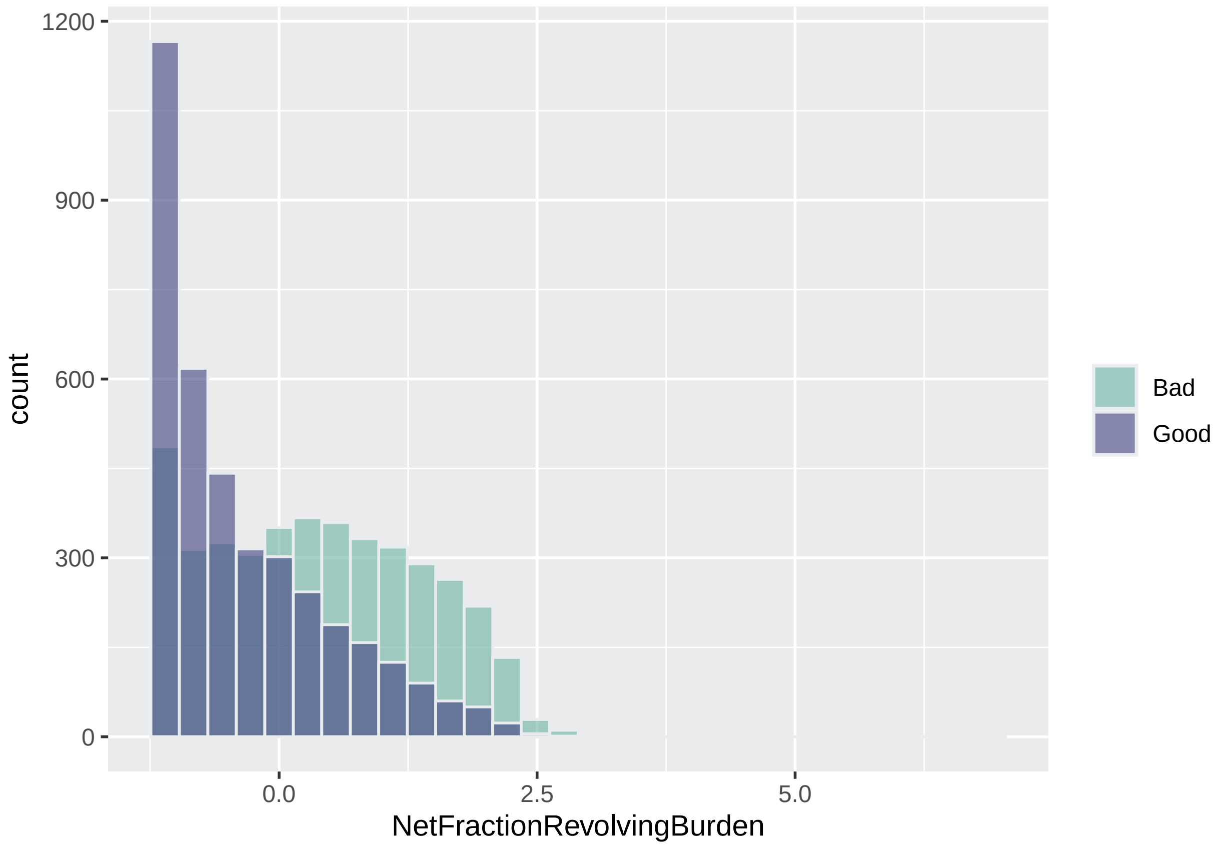

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



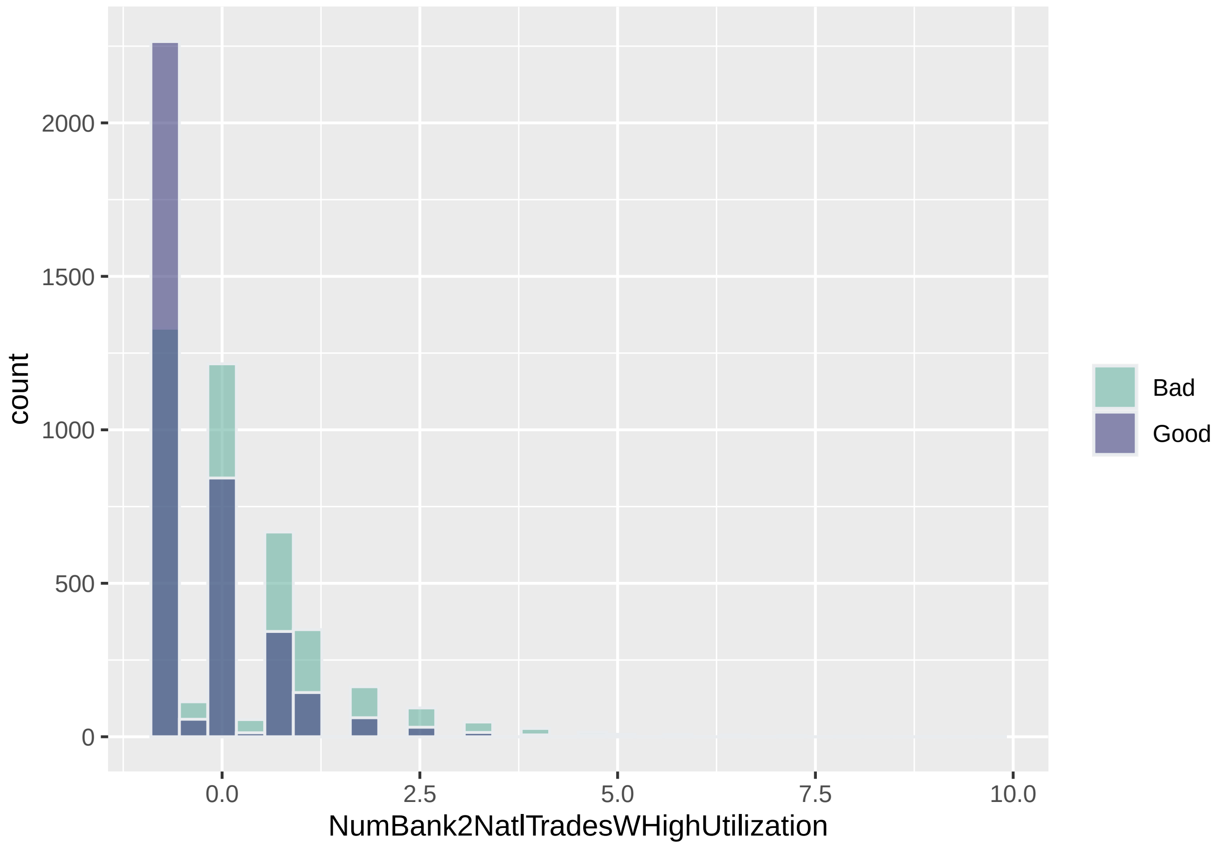

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



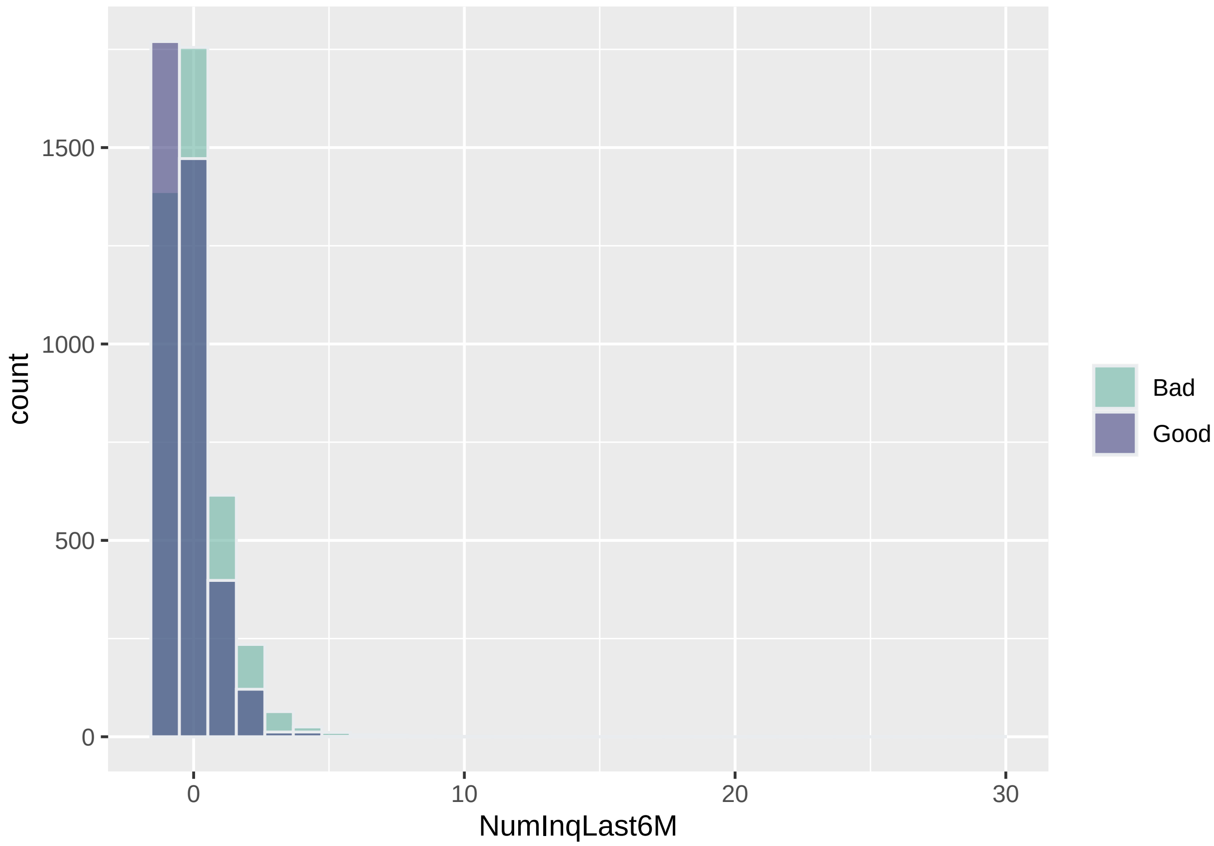

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



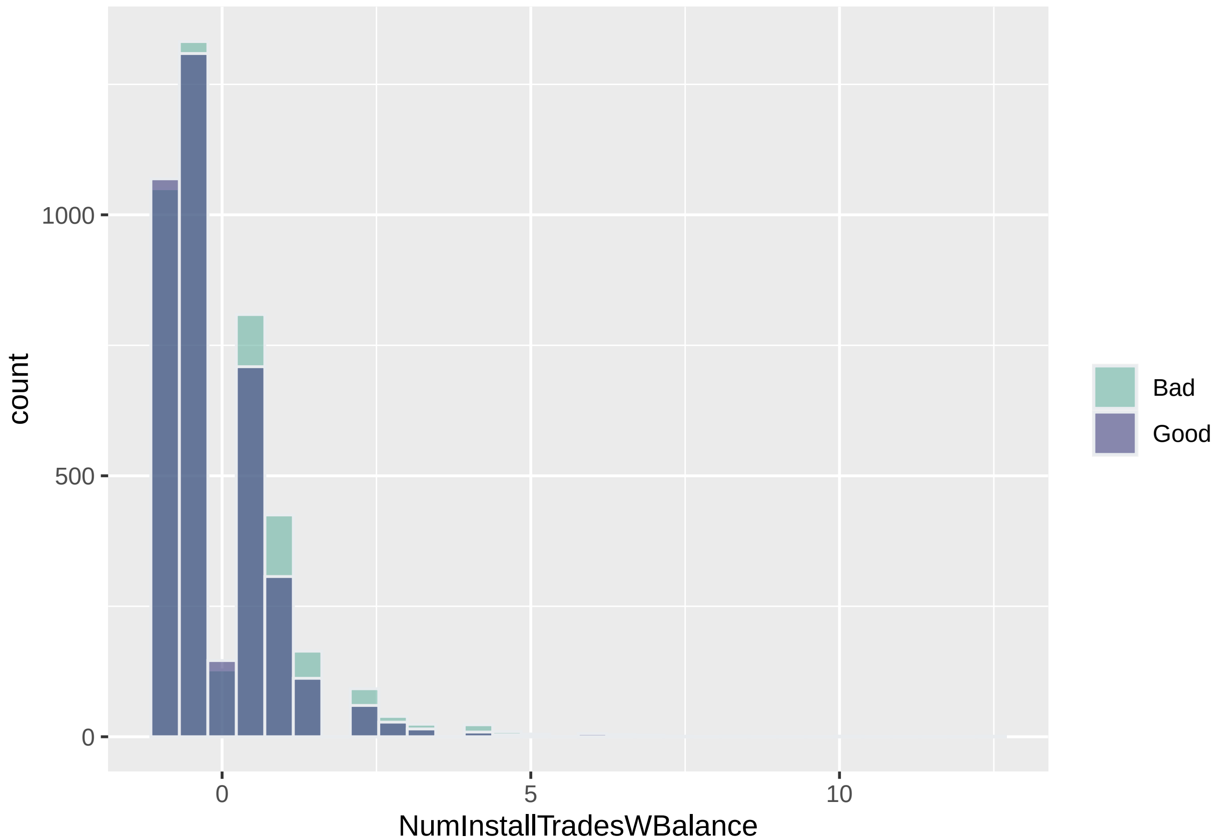

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



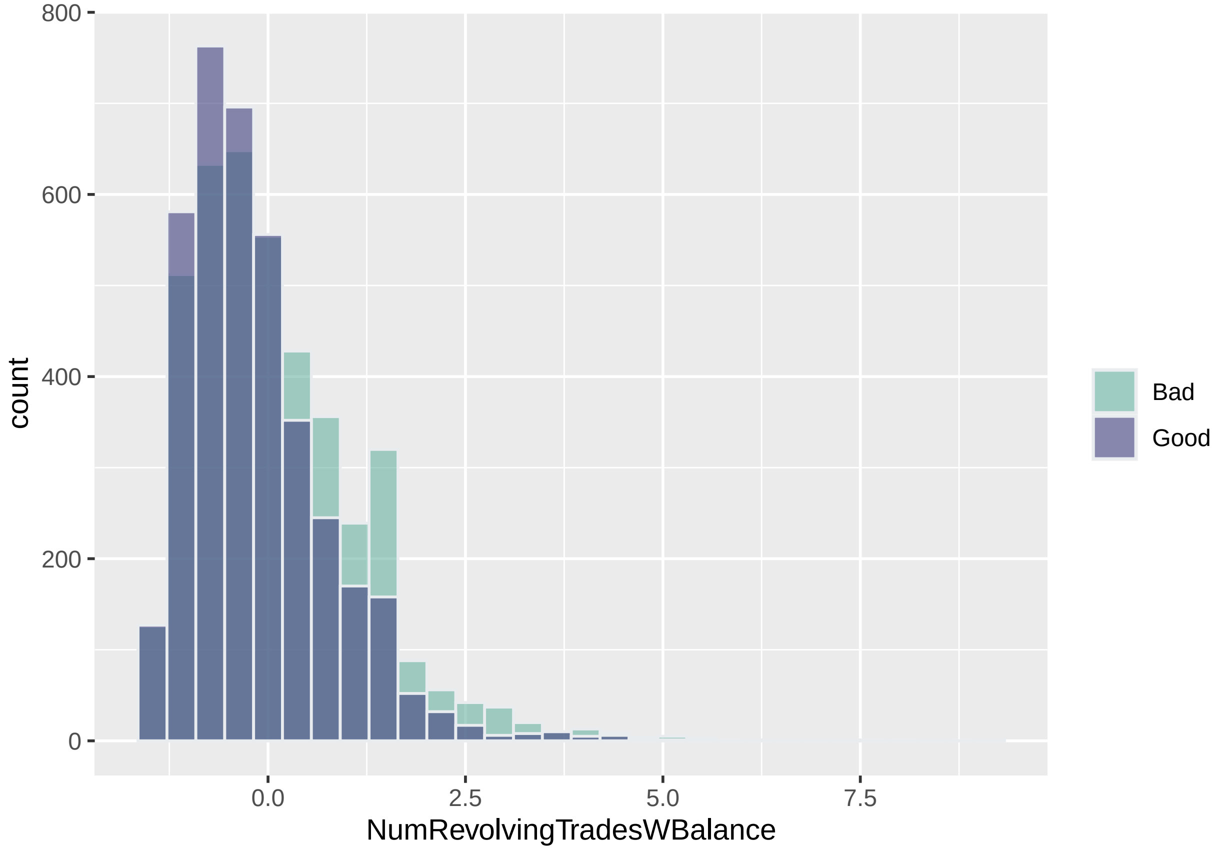

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



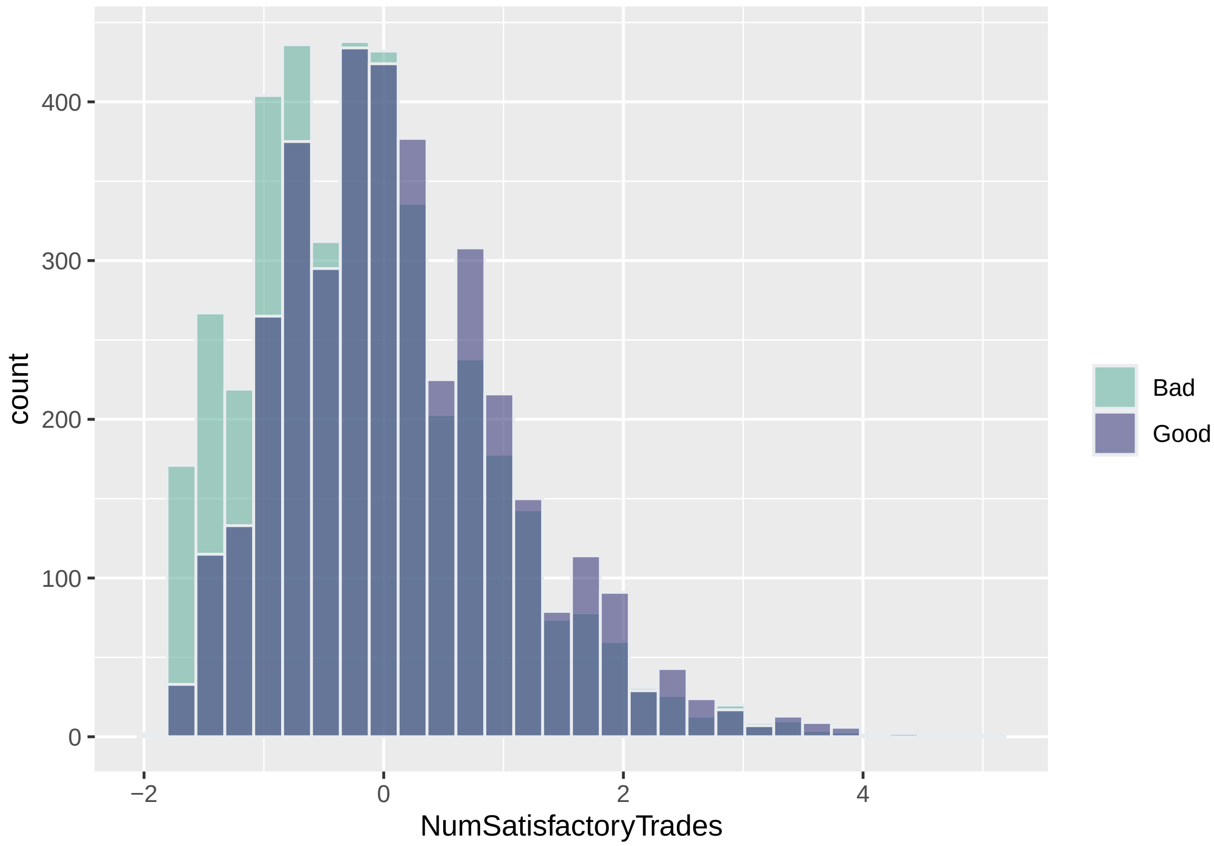

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



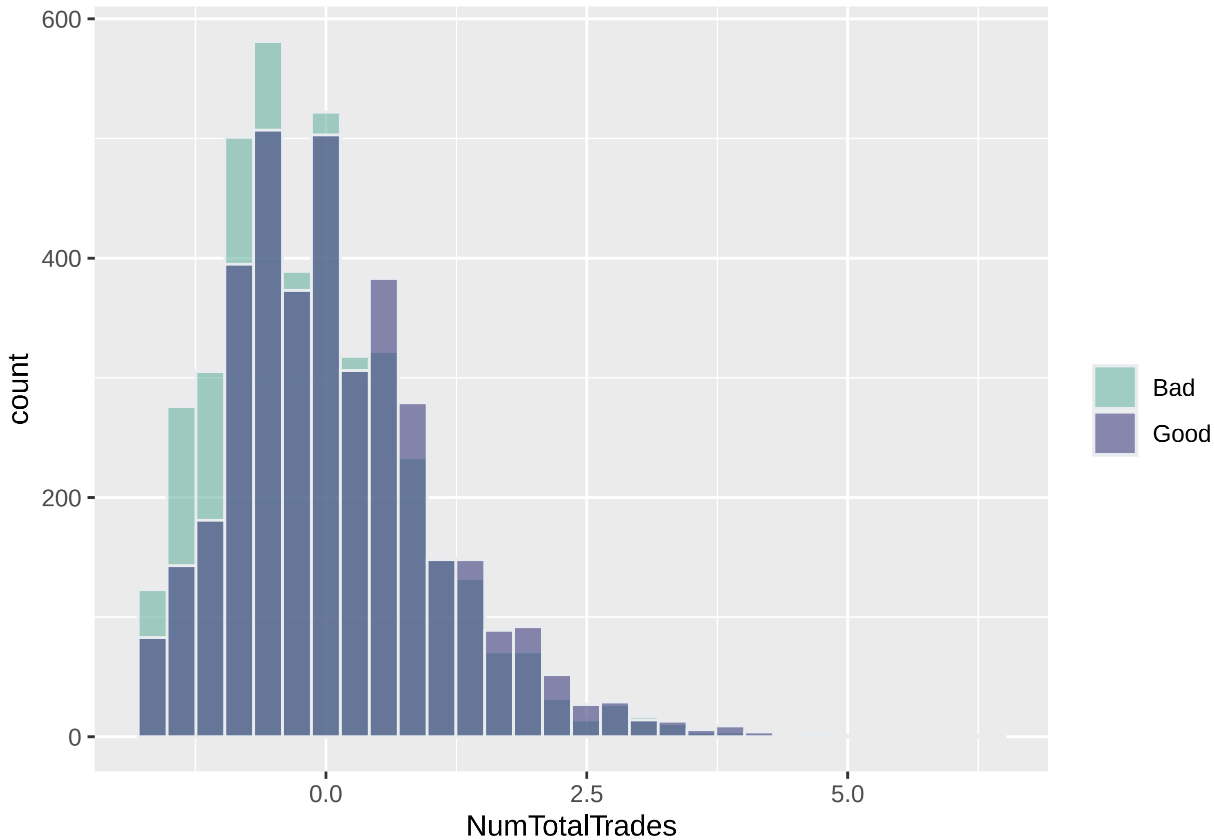

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



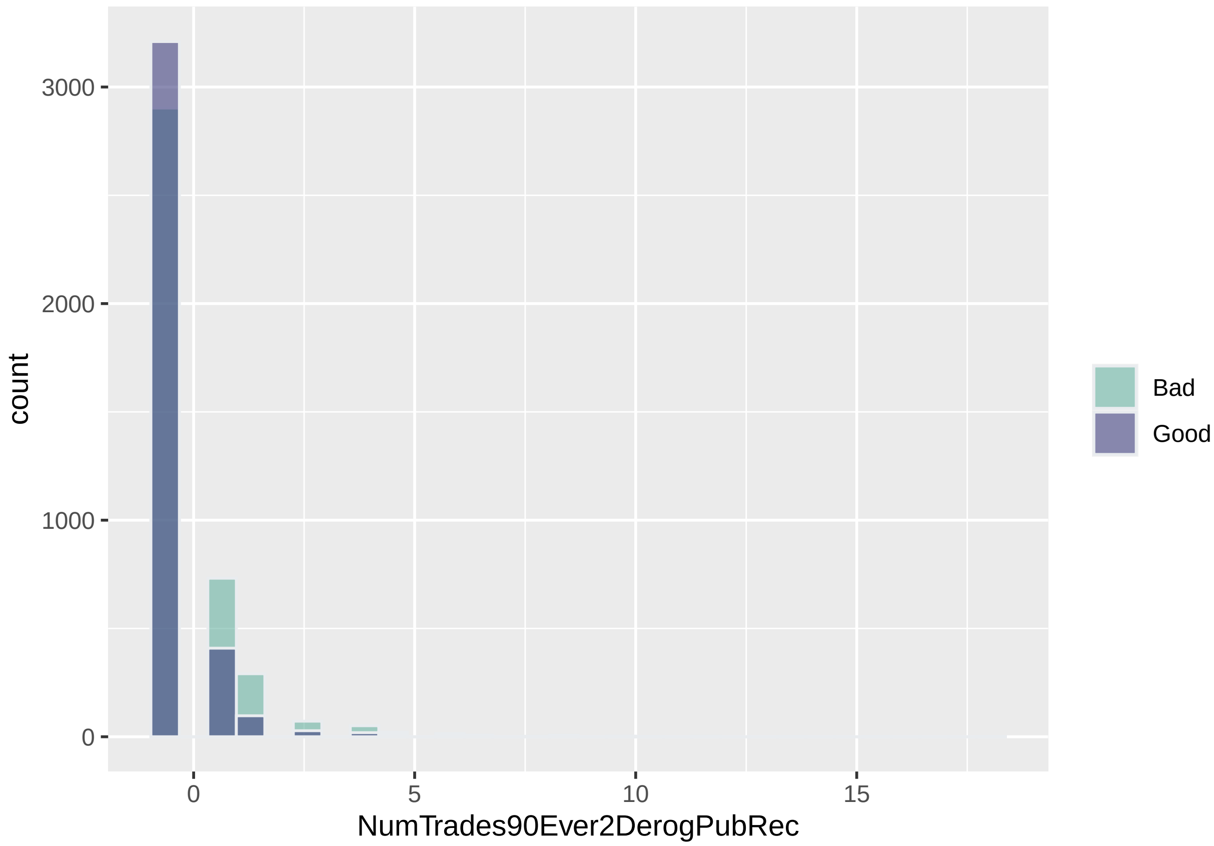

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



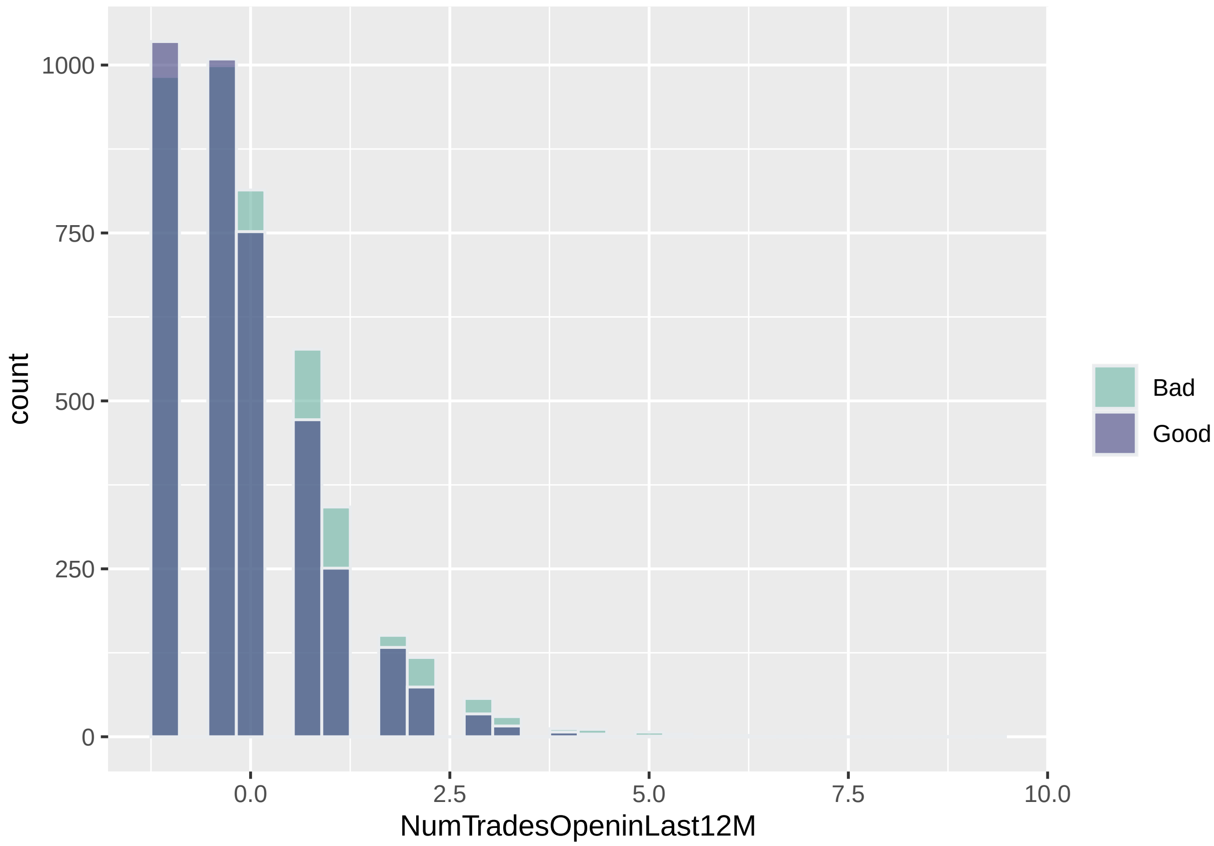

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



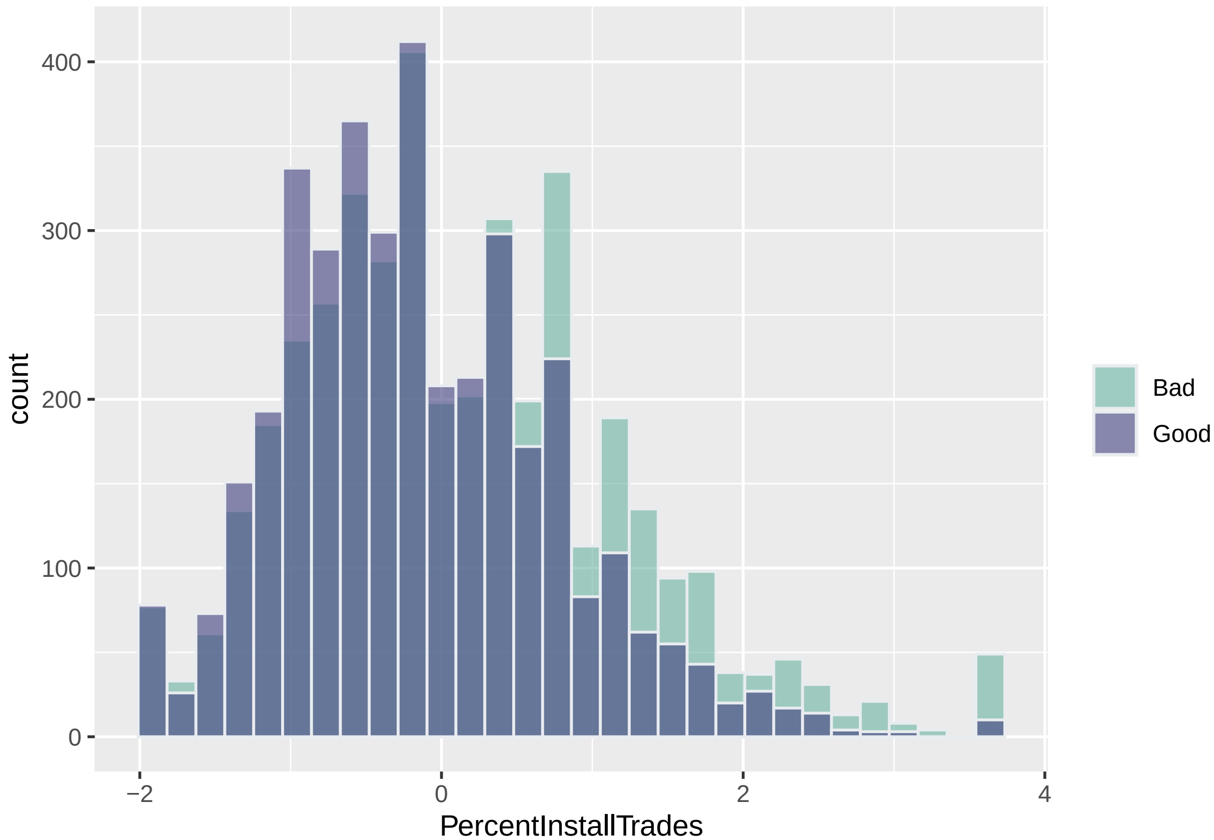

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



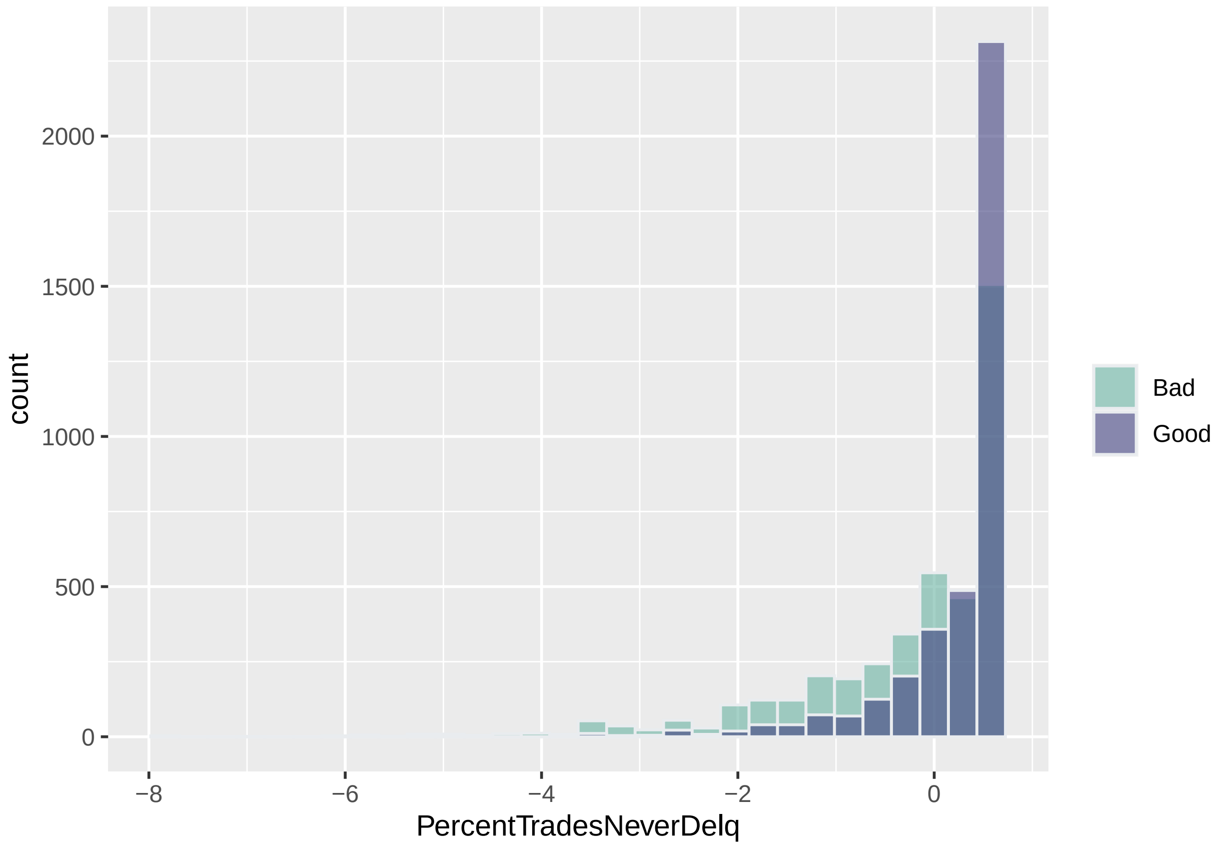

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



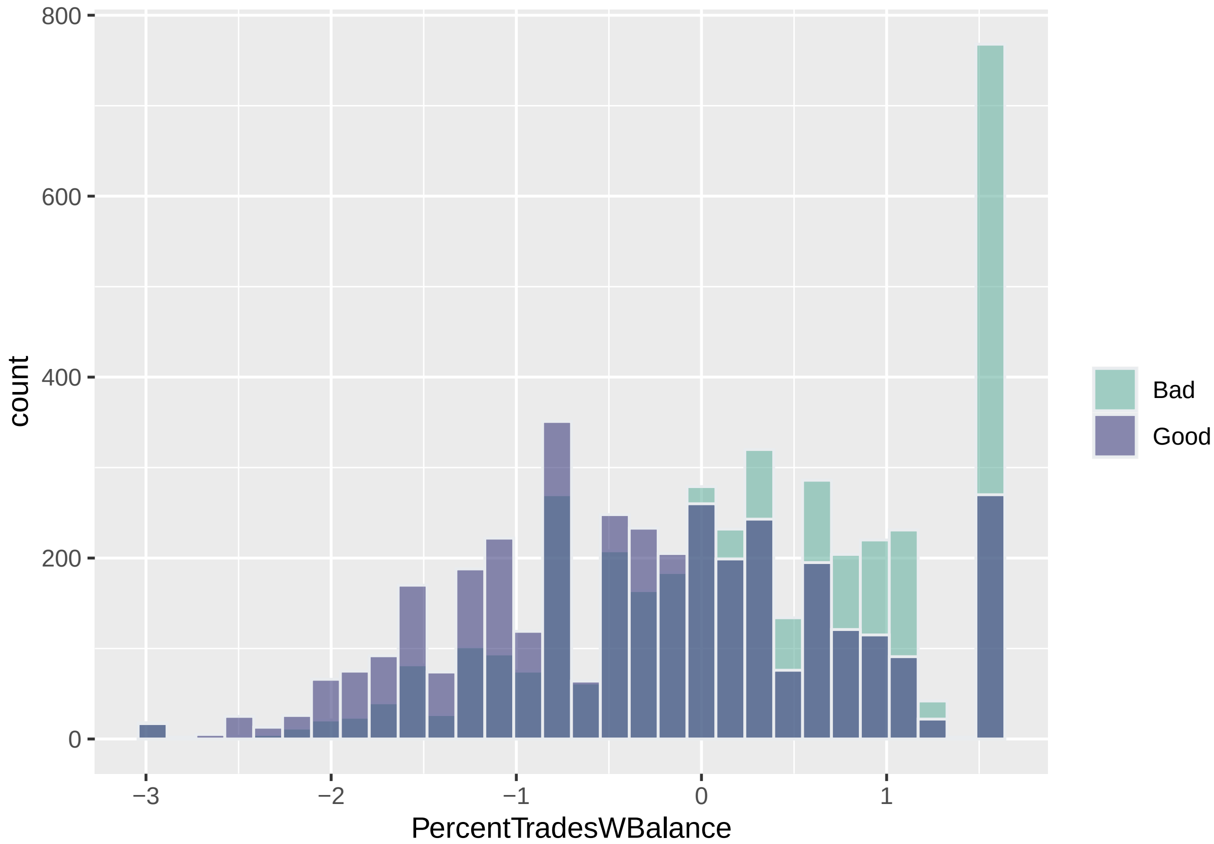

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

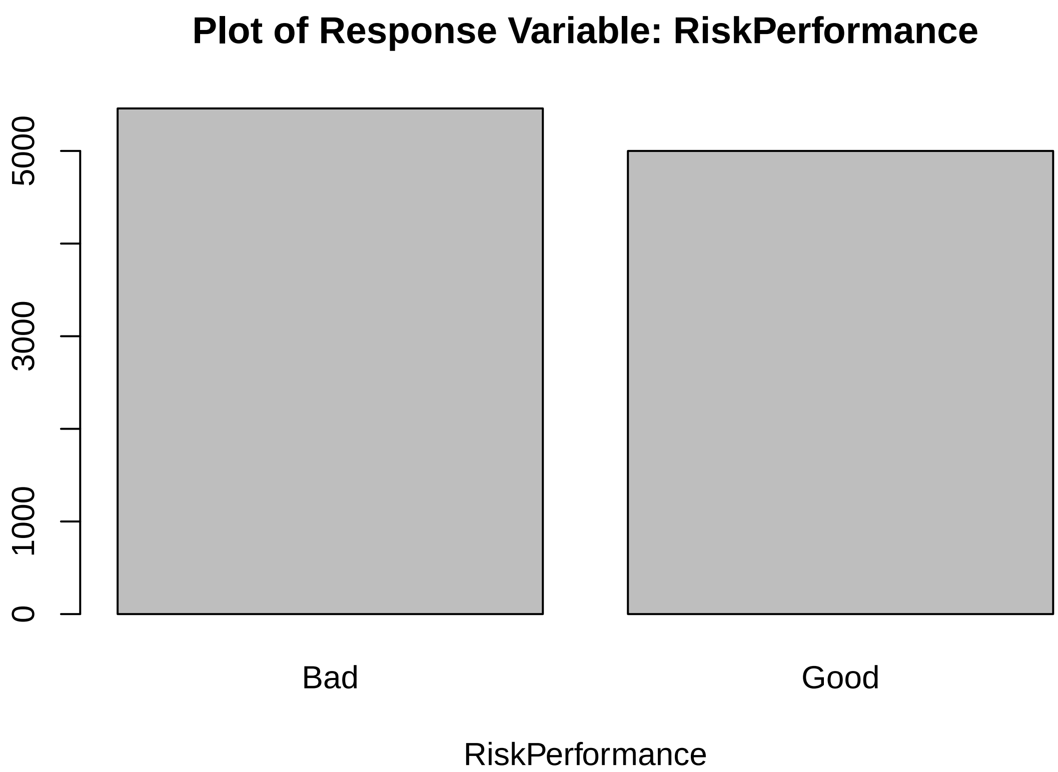



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



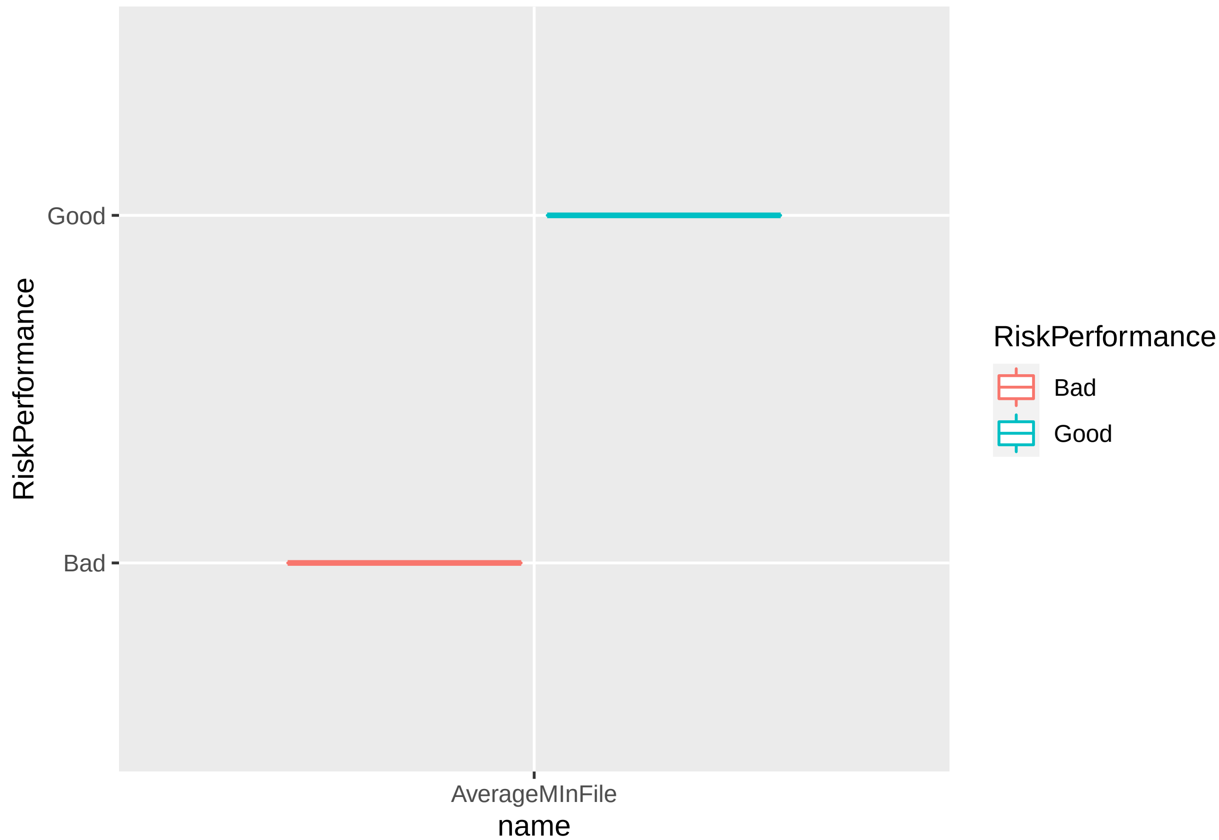
BoxPlots

```
# bar plot of response variable; RiskPerformance
barplot(table(heloc$RiskPerformance), main="Plot of Response Variable: RiskPerformance", xlab="RiskPerf
```

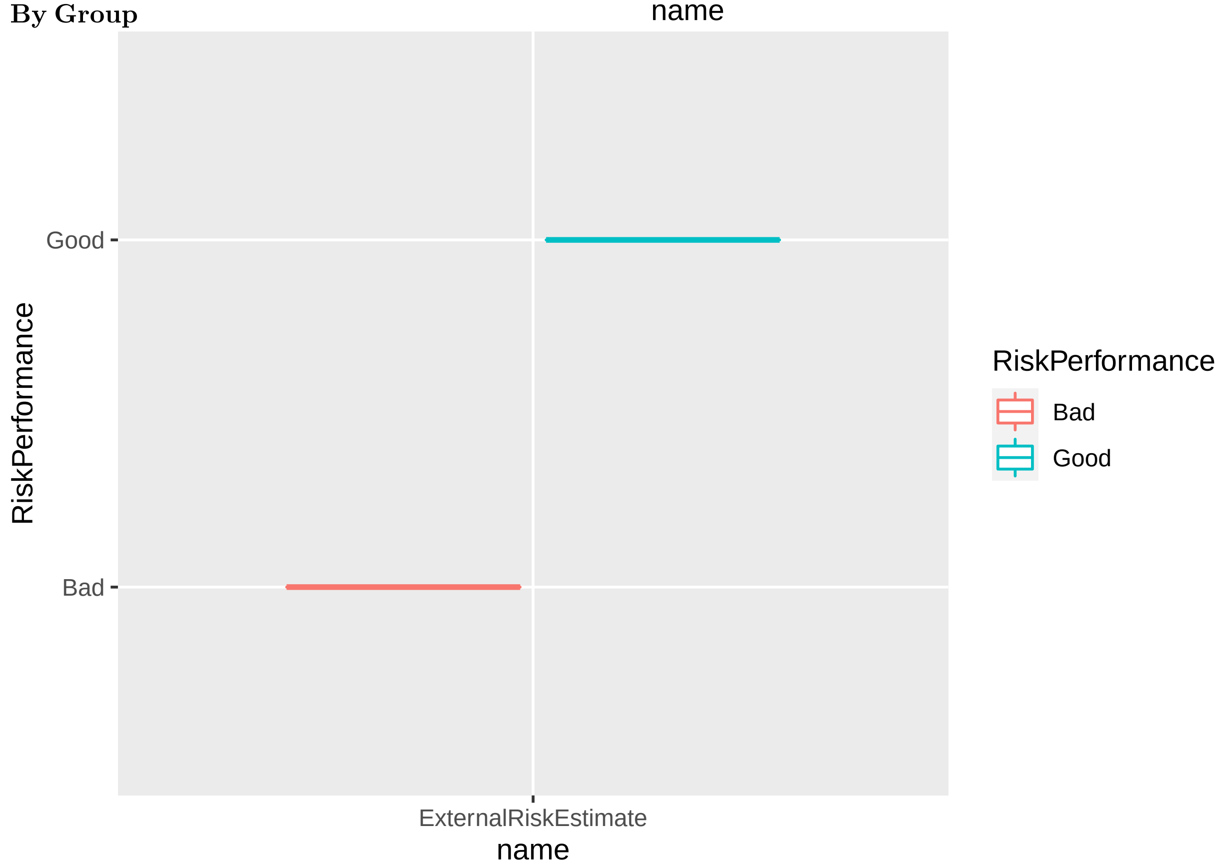


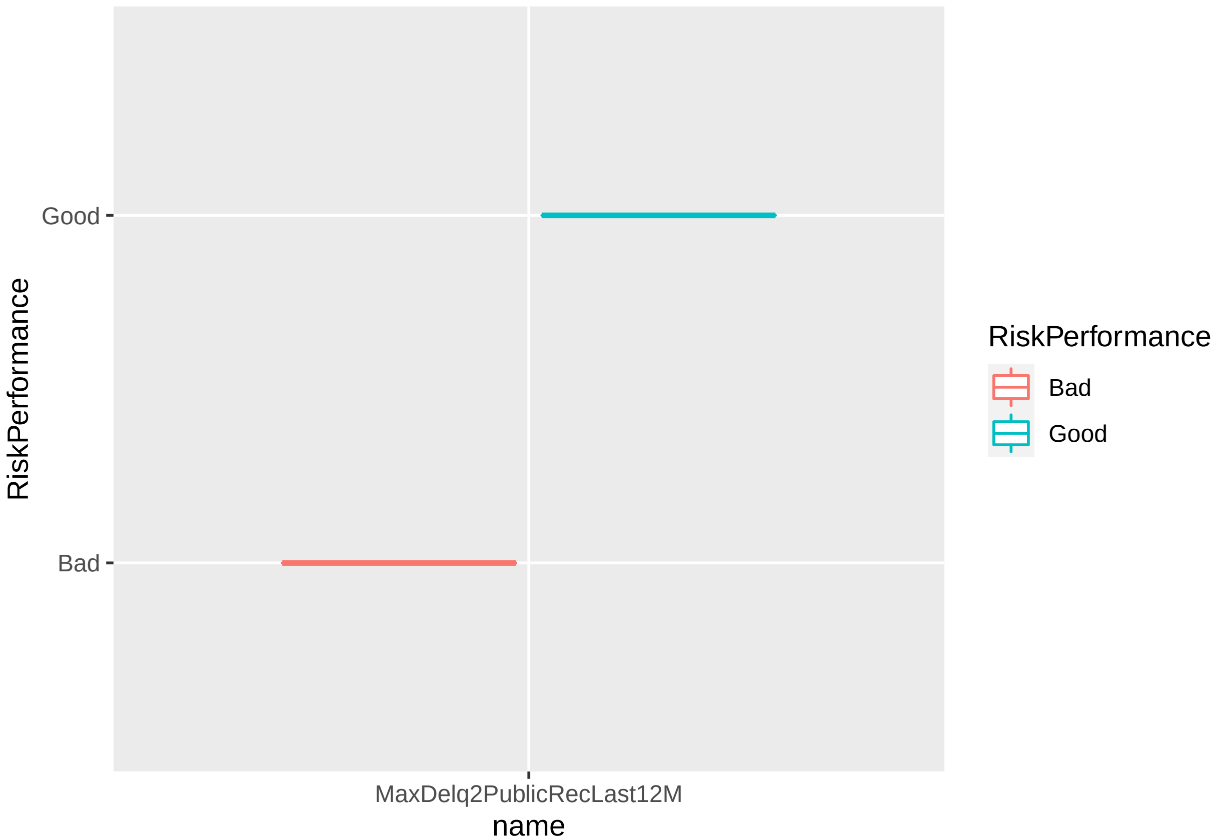
Overall

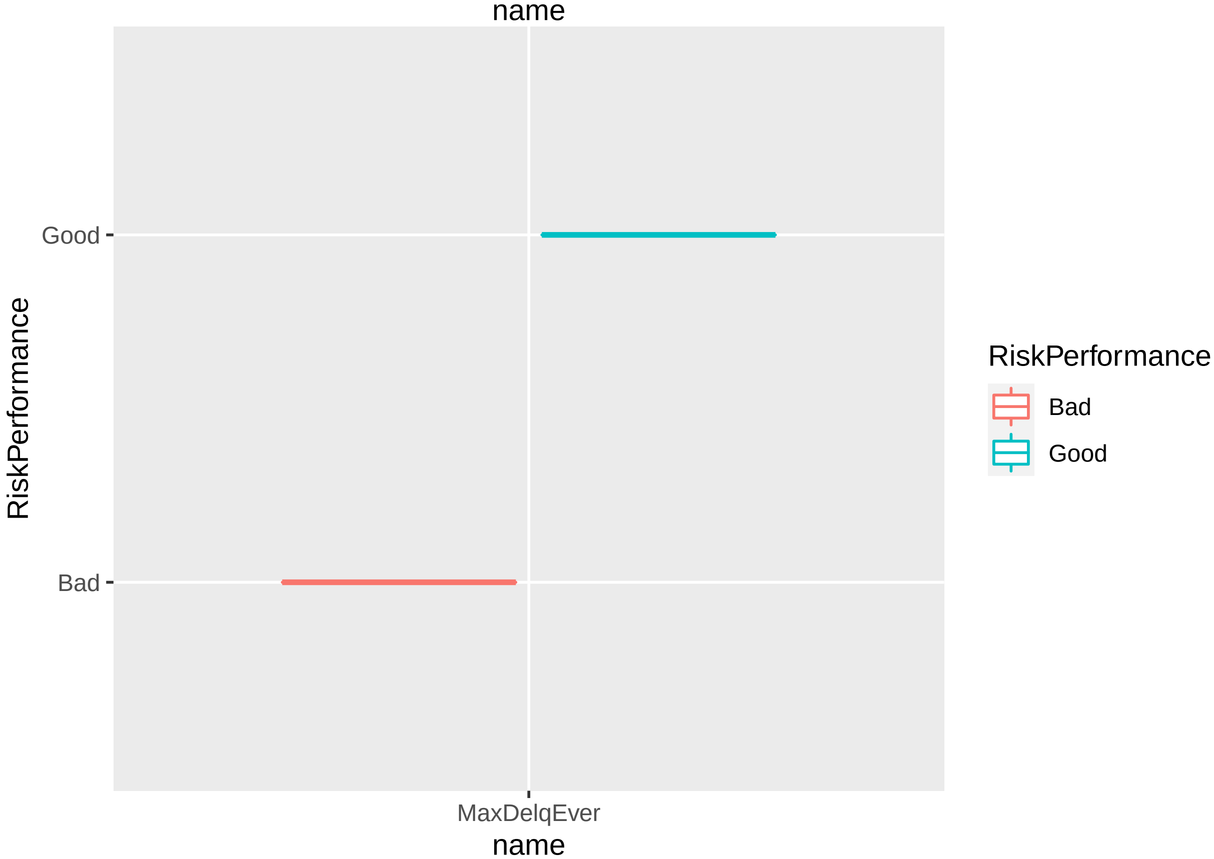
```
par(mfrow=c(4,6))
for (name in heloc_ifs_names){
  # boxplot to view outliers
  p <- ggplot(heloc_train,
              aes(x=name,
                  y=RiskPerformance,
                  color=RiskPerformance)) +
          geom_boxplot()
  print(p)
}
```

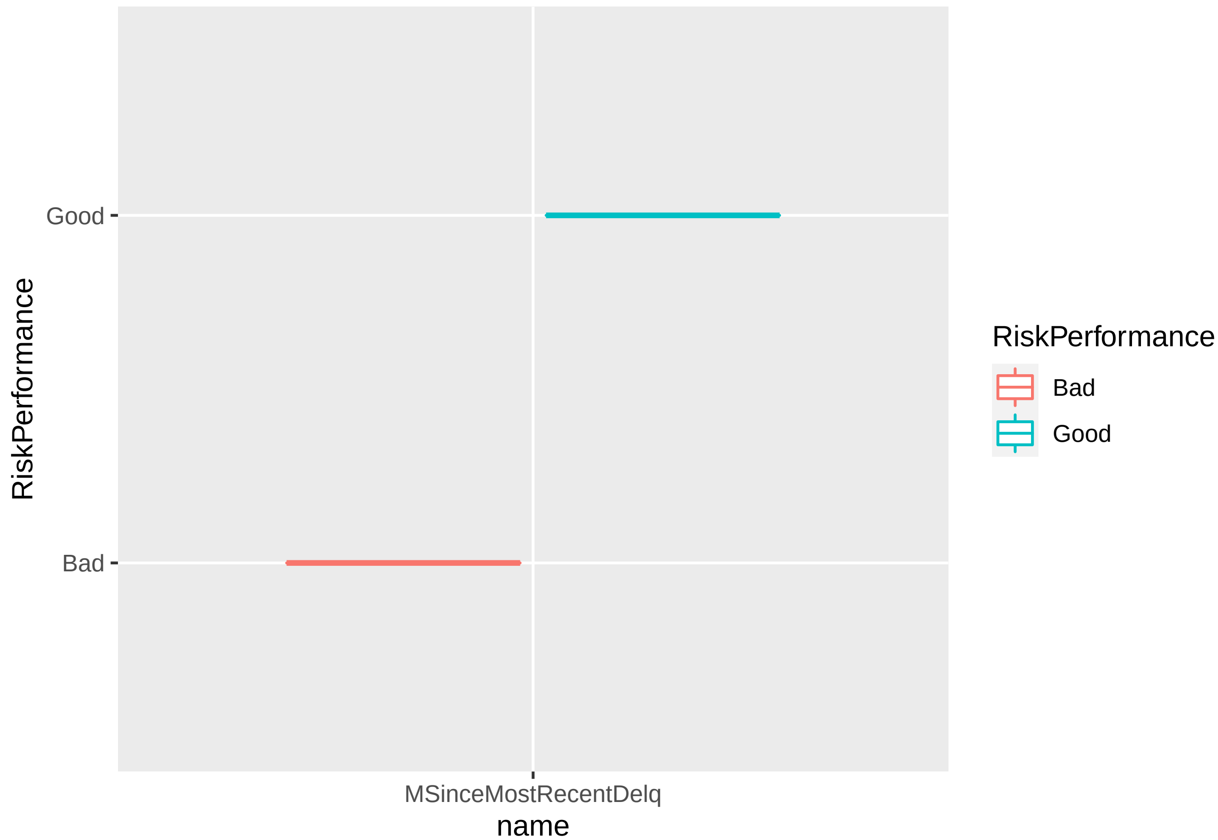

Good
Bad
RiskPerformance
AverageMInFile
name
RiskPerformance
Bad
Good

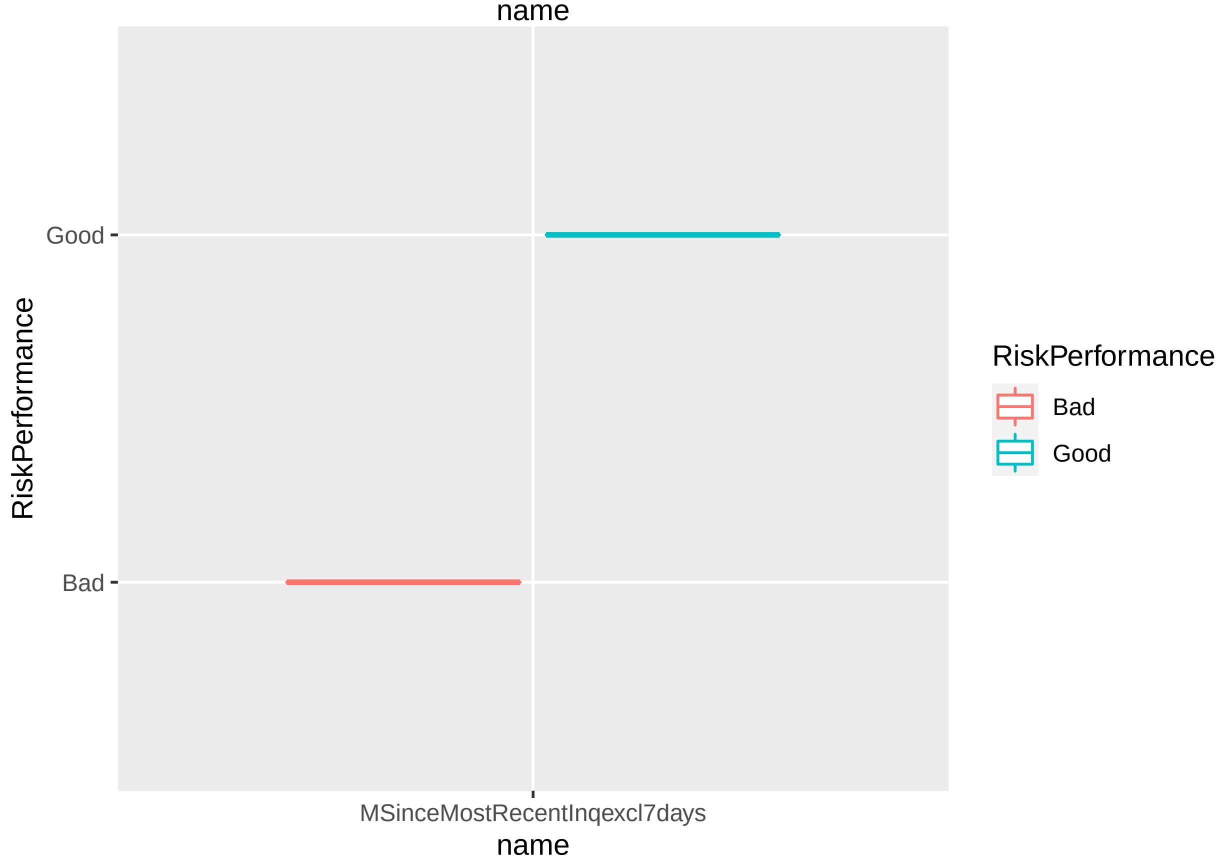
By Group

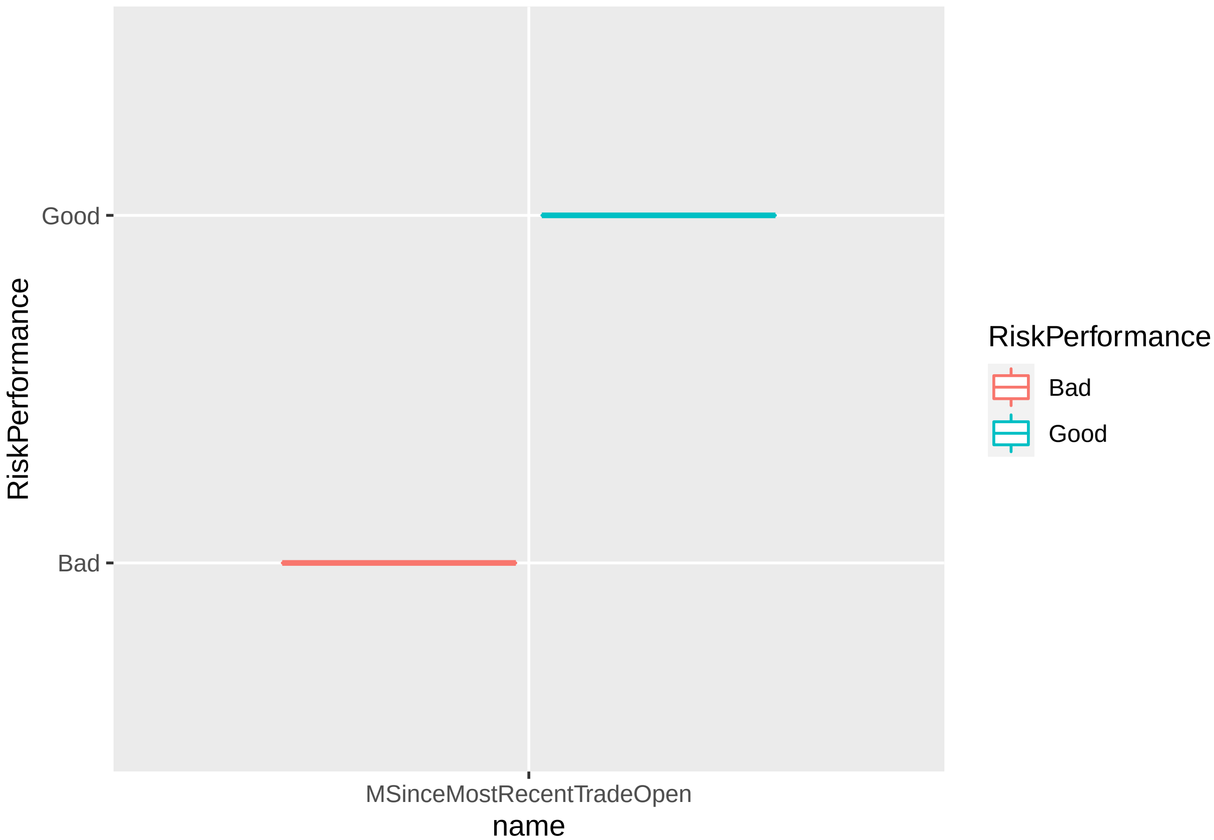

Good
Bad
RiskPerformance
ExternalRiskEstimate
name
RiskPerformance
Bad
Good

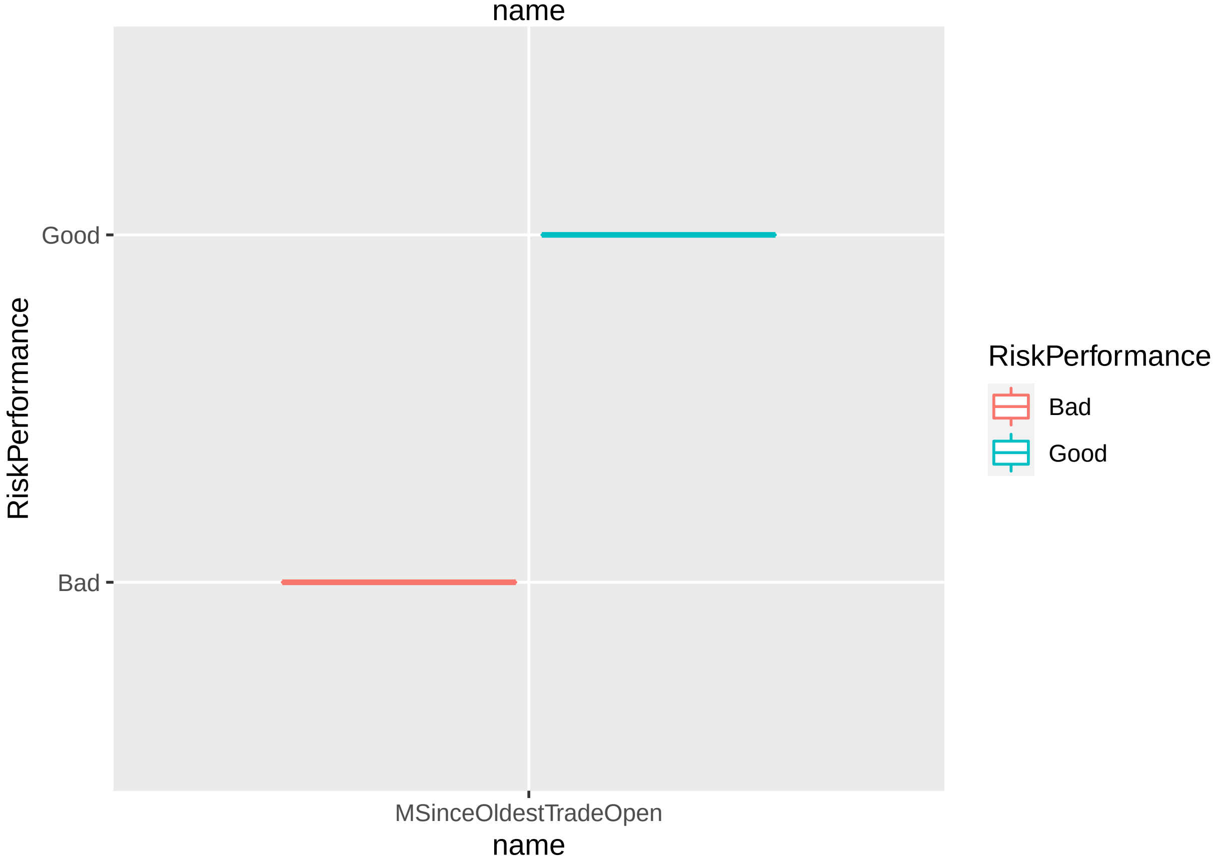

Good
Bad
RiskPerformance
MaxDelq2PublicRecLast12M
name
RiskPerformance
Bad
Good

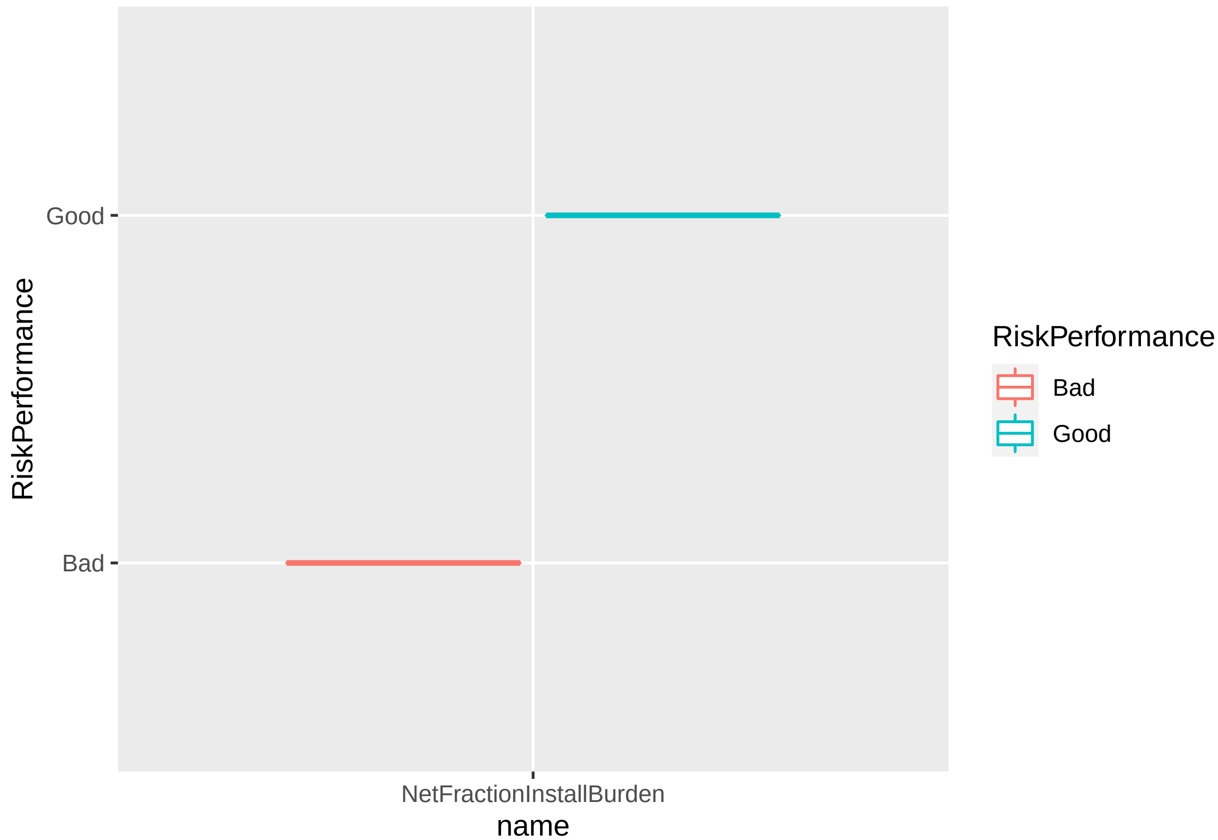

Good
Bad
RiskPerformance
MaxDelqEver
name
RiskPerformance
Bad
Good

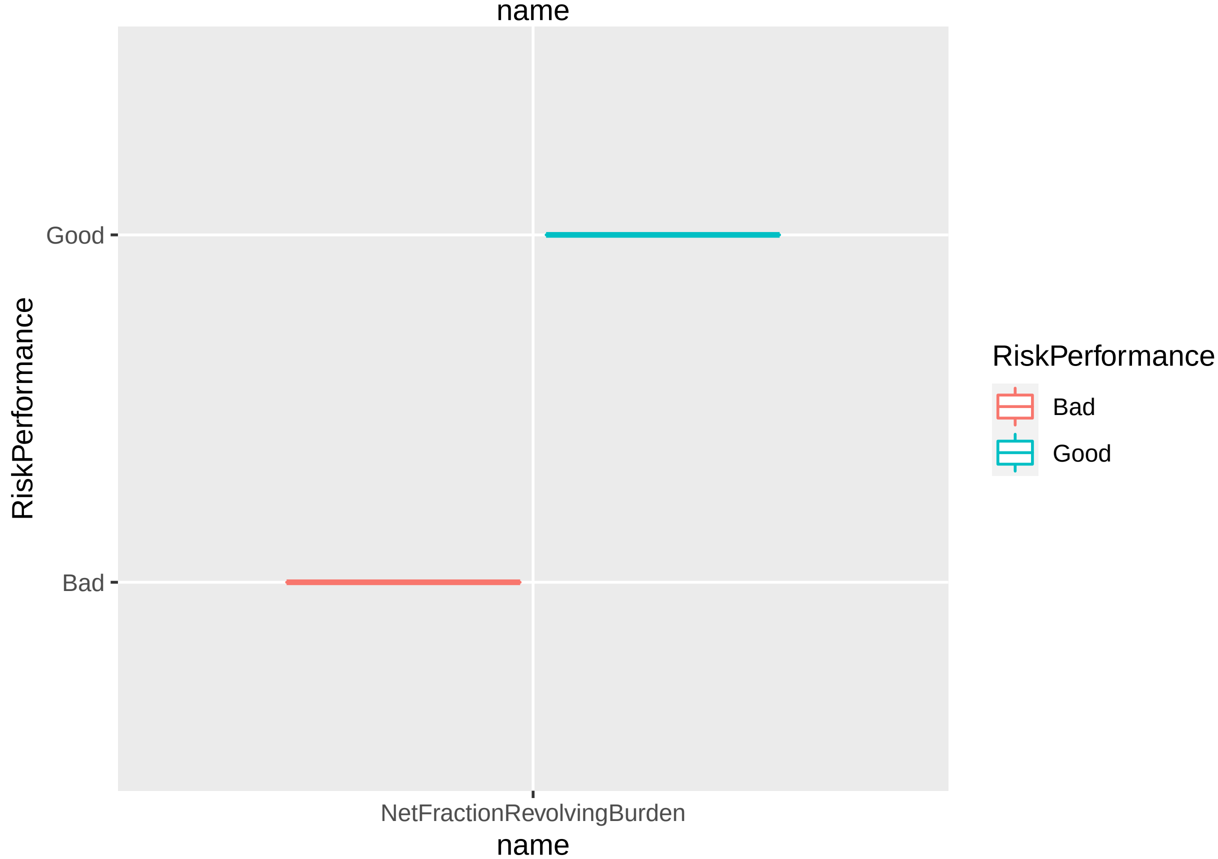

RiskPerformance
Good
Bad
MSinceMostRecentDelq
name
RiskPerformance
Bad
Good

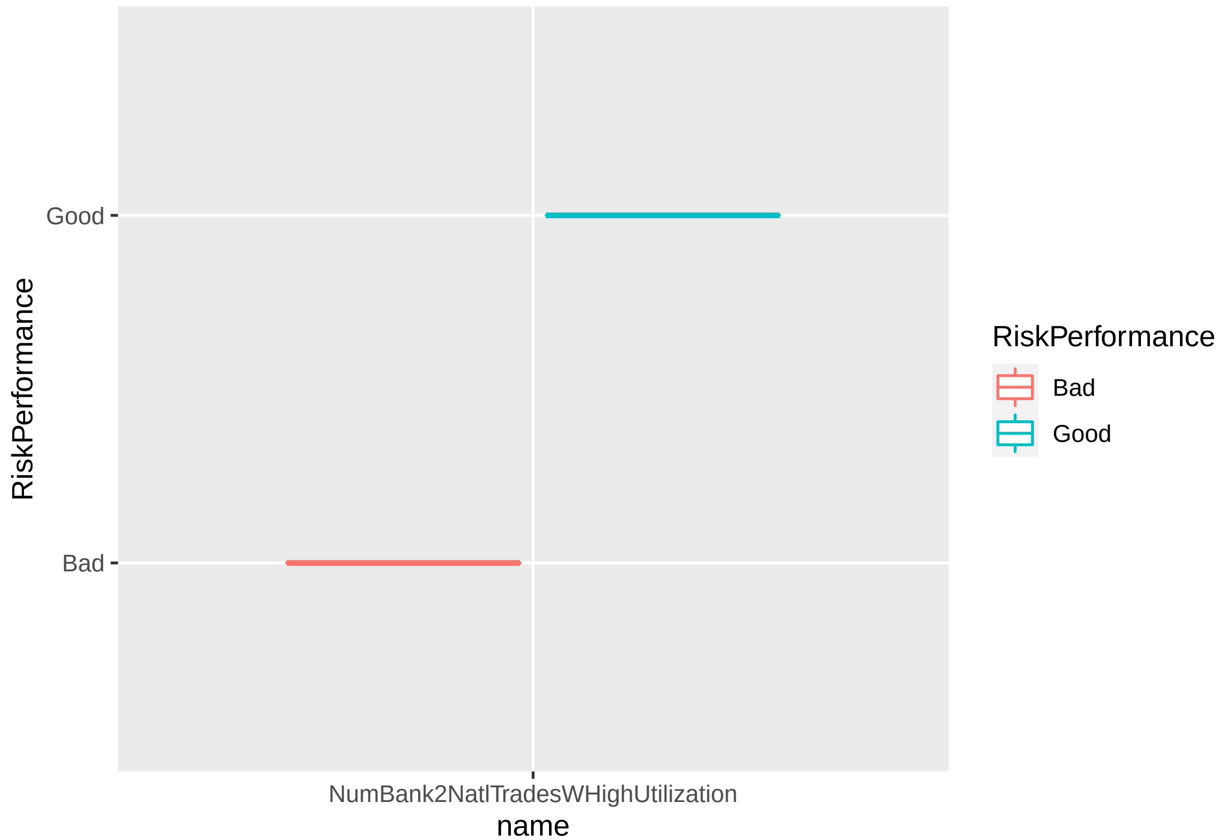

RiskPerformance
Good
Bad
MSinceMostRecentInqexcl7days
name
RiskPerformance
Bad
Good

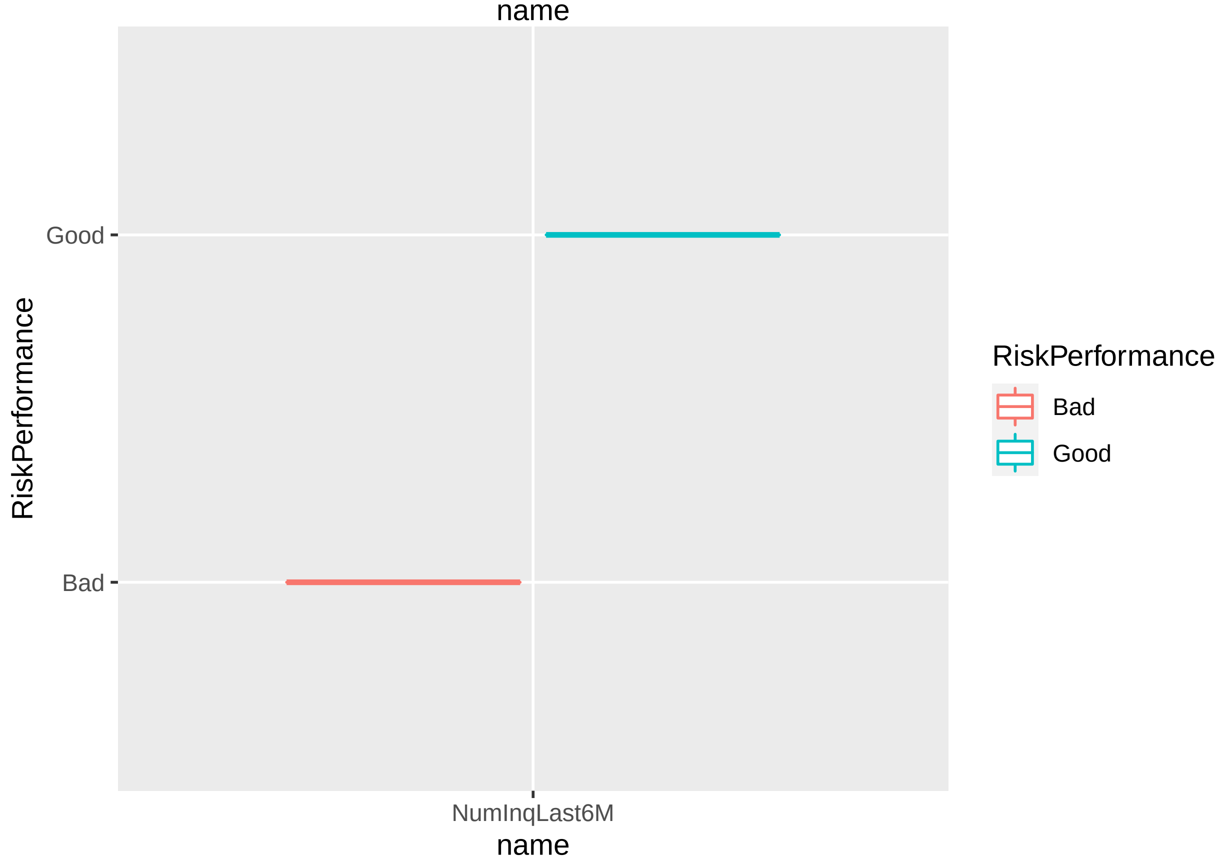

Good
Bad
RiskPerformance
MSinceMostRecentTradeOpen
name
RiskPerformance
Bad
Good

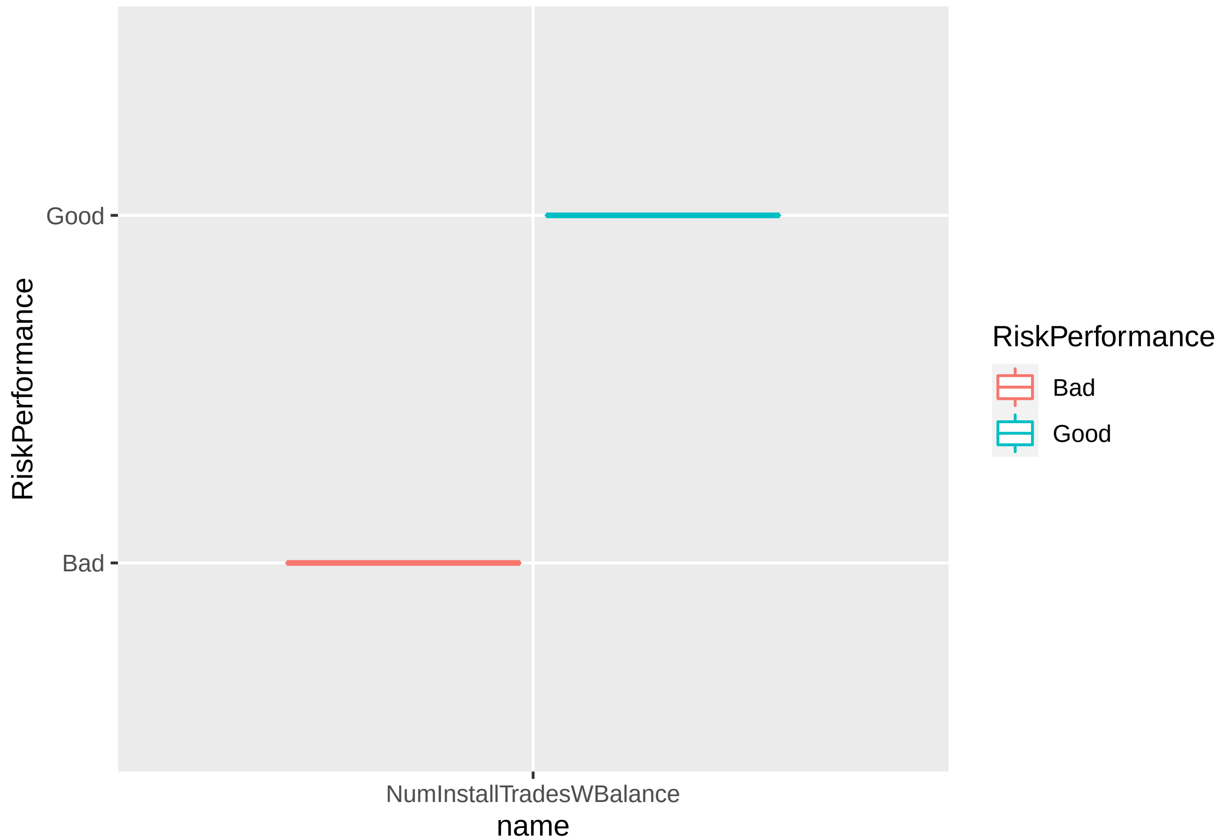

Good
Bad
RiskPerformance
MSinceOldestTradeOpen
name
RiskPerformance
Bad
Good

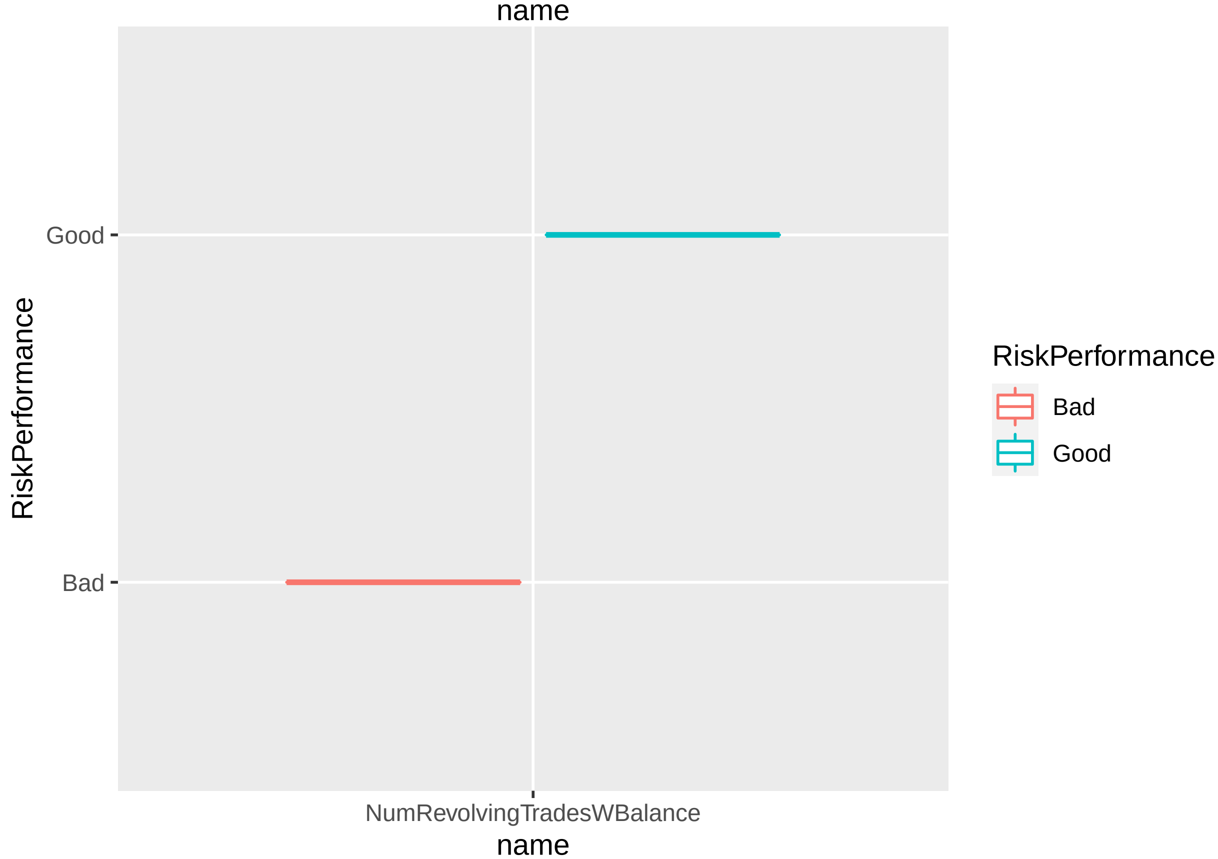

Good
Bad
RiskPerformance
NetFractionInstallBurden
name
RiskPerformance
Bad
Good

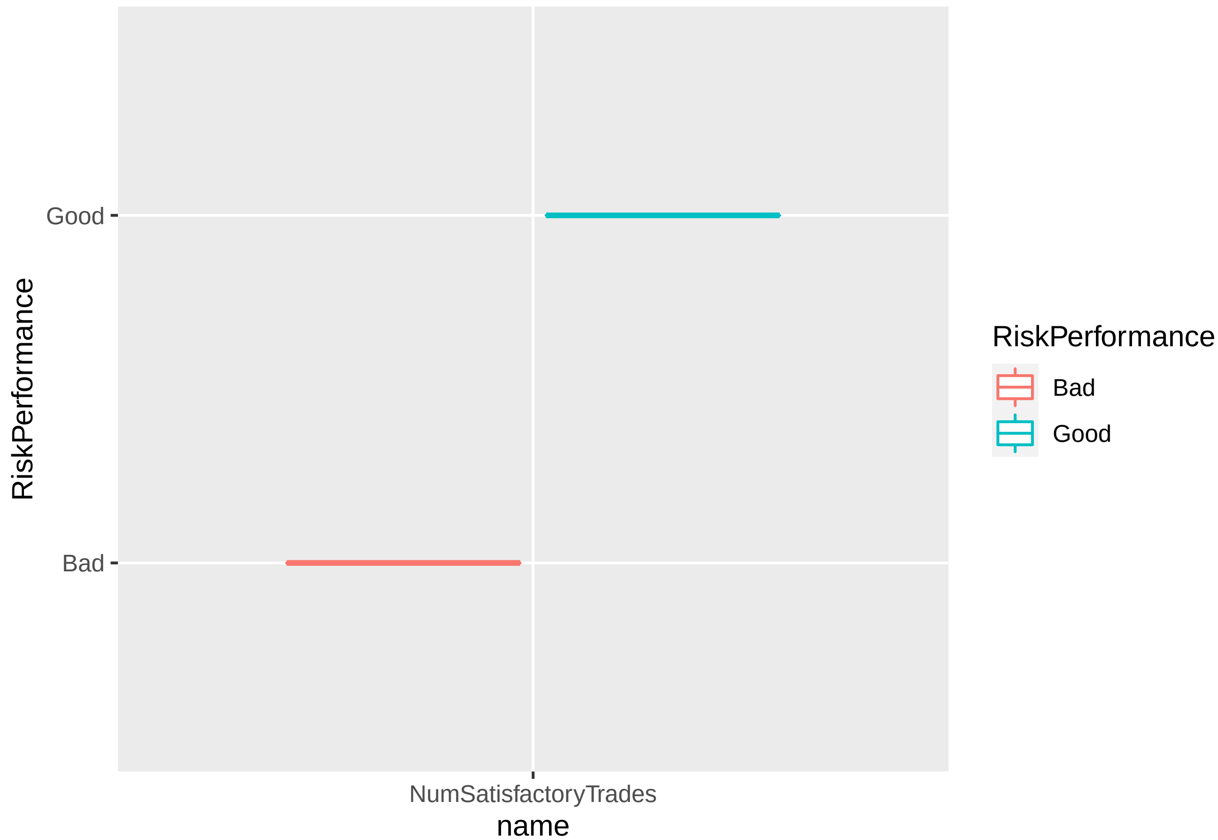

Good
Bad
RiskPerformance
NetFractionRevolvingBurden
name
RiskPerformance
Bad
Good

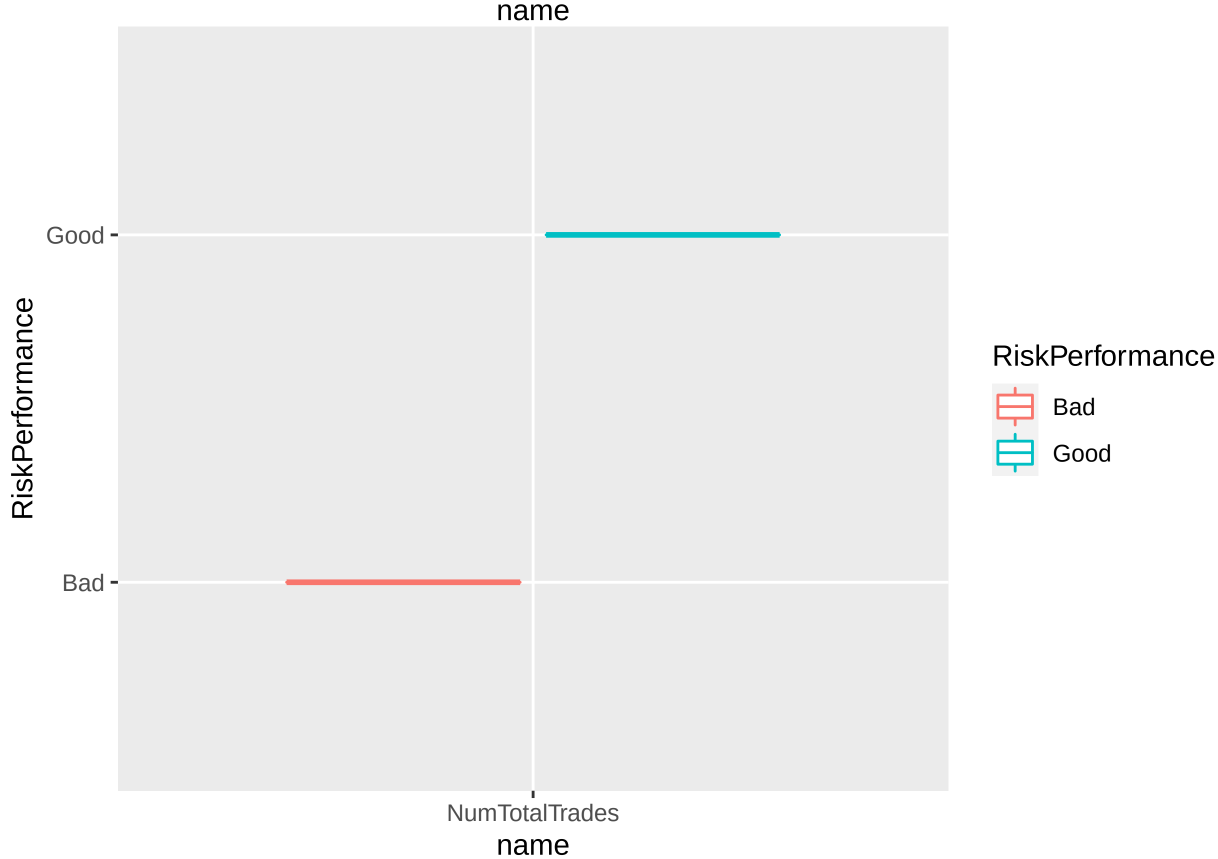

Good
Bad
RiskPerformance
NumBank2NatlTradesWHighUtilization
name
RiskPerformance
Bad
Good

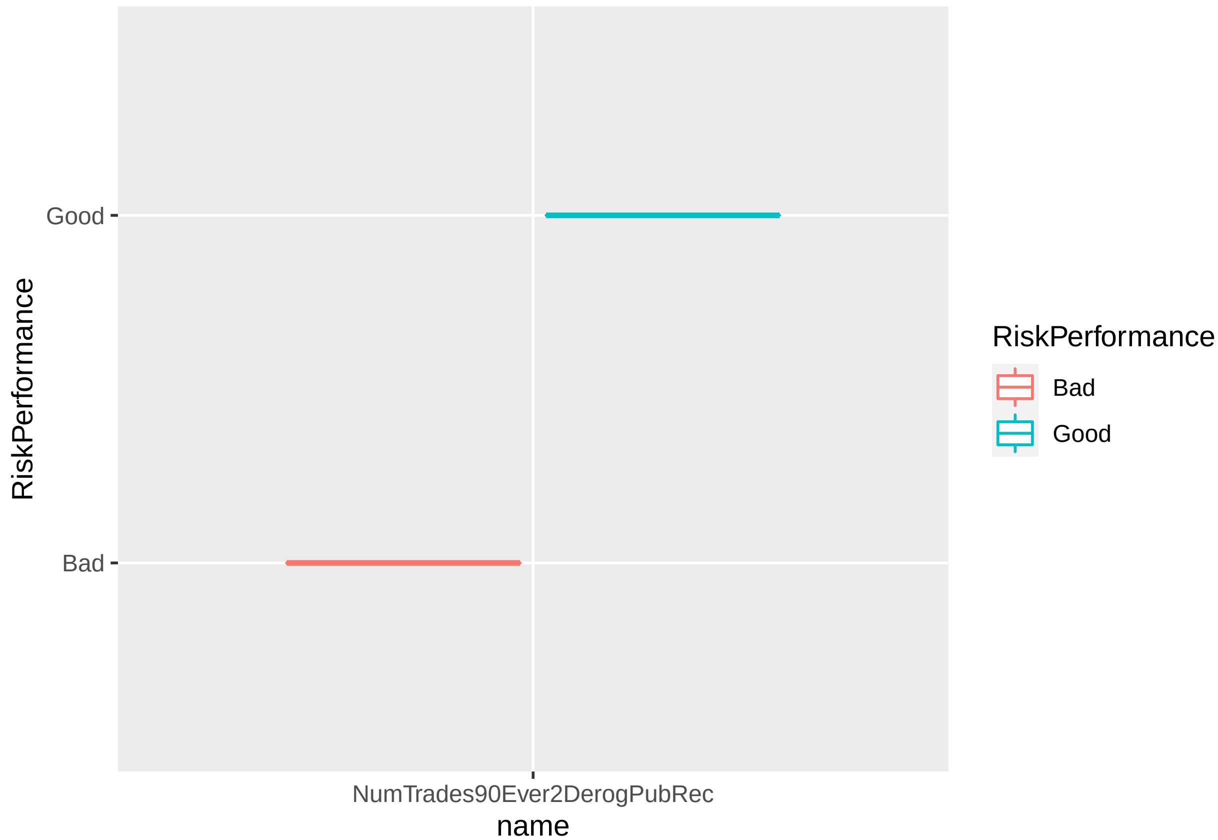

Good
Bad
RiskPerformance
NumInqLast6M
name
RiskPerformance
Bad
Good

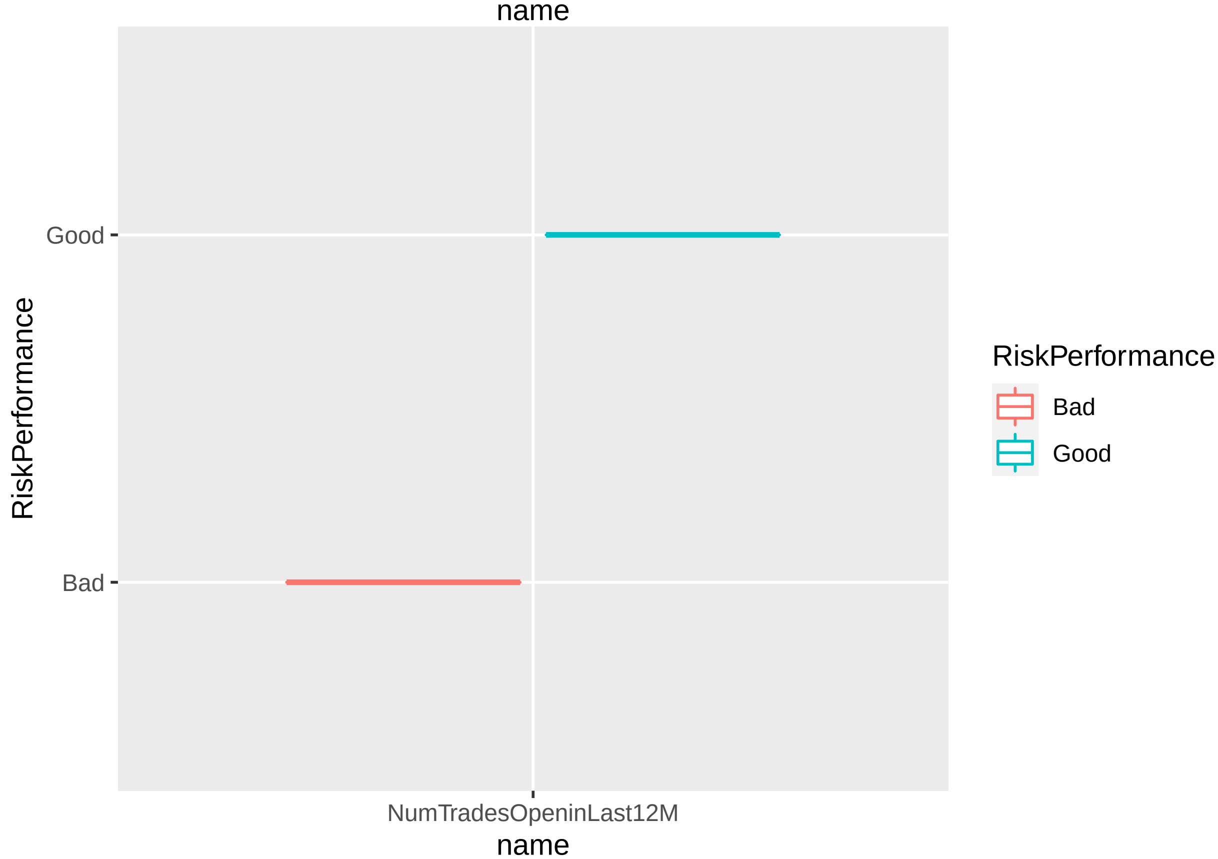

RiskPerformance
Good
Bad
NumInstallTradesWBalance
name
RiskPerformance
Bad
Good

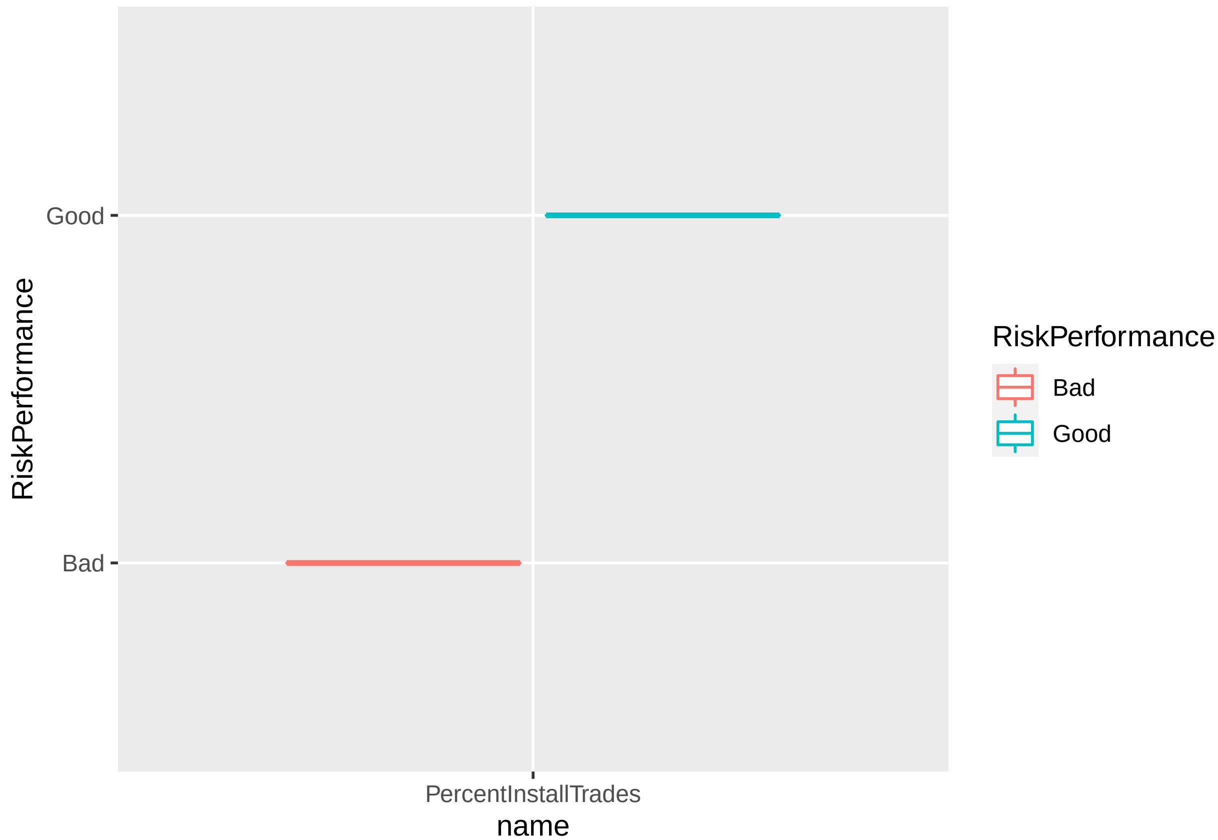

RiskPerformance
Good
Bad
NumRevolvingTradesWBalance
name
RiskPerformance
Bad
Good

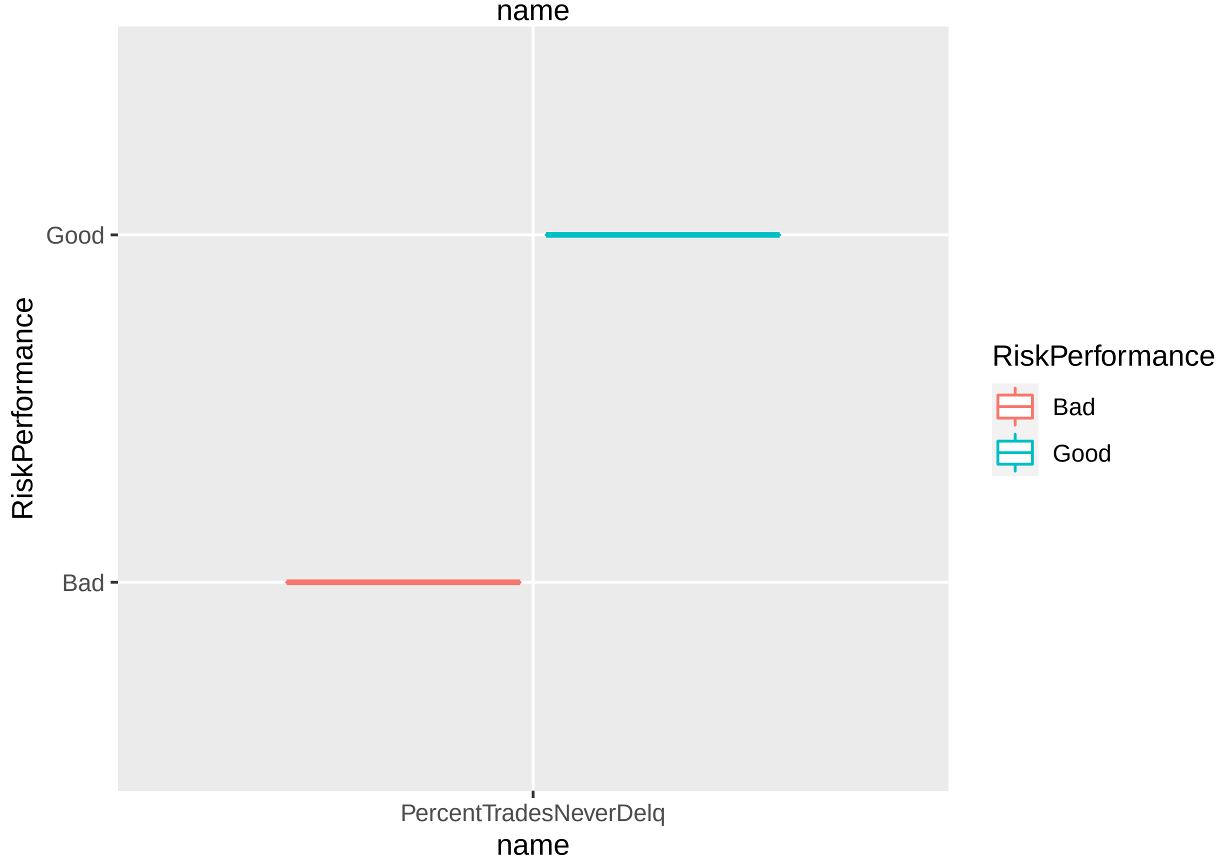

Good
Bad
RiskPerformance
NumSatisfactoryTrades
name
RiskPerformance
Bad
Good

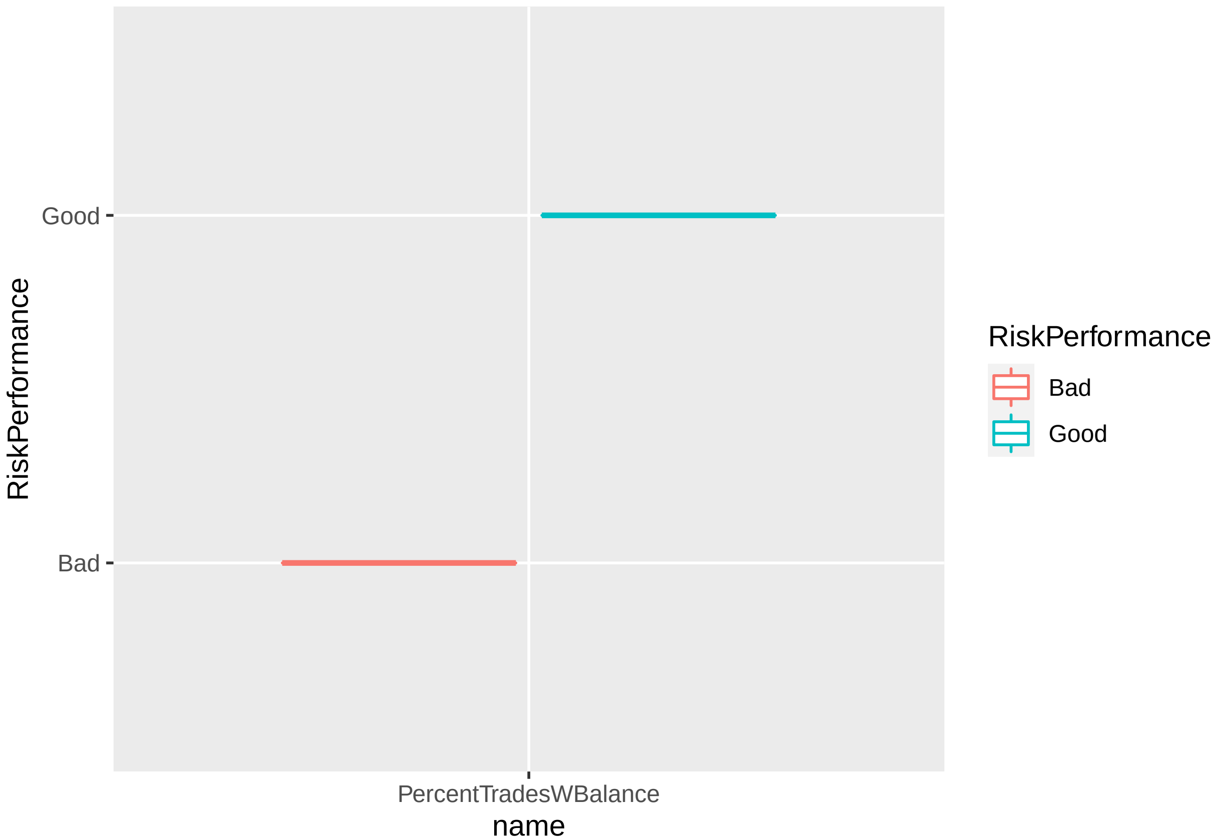

Good
Bad
RiskPerformance
NumTotalTrades
name
RiskPerformance
Bad
Good


RiskPerformance
Good
Bad
NumTrades90Ever2DerogPubRec
name
RiskPerformance
Bad
Good


RiskPerformance
Good
Bad
NumTradesOpeninLast12M
name
RiskPerformance
Bad
Good


Good
Bad
RiskPerformance
PercentInstallTrades
name
RiskPerformance
Bad
Good


Good
Bad
RiskPerformance
PercentTradesNeverDelq
name
RiskPerformance
Bad
Good




```
control <- caret::trainControl(method="cv",
                              classProbs=TRUE,
                              savePredictions=TRUE,
                              summaryFunction=twoClassSummary)
```

Models

With the following models, we are trying to maximize the SPECIFICITY of the models to prevent any non-qualified loanees being presented with a HELOC loan.

Specificity is defined as: "The specificity is defined as the rate that nonevent samples are predicted as nonevents" (Kuhn & Johnson, 2013)

```
bestIndex <- function(model){
  # returns top ROC value and surrounding indices from model
  highest_score <- max(model$results$ROC)

  # get row index and convert to type Int
  best_index <- rownames(model$results[model$results$ROC == highest_score,])
  best_index <- as.integer(best_index)

  return(best_index)
}

confusionMatrix <- function(testResults.model){
  caret::confusionMatrix(testResults.model,
                         as.factor(testResults$obs),
```

```r
                       positive="Good")
}

modelScoreBoard <- function(testResults){
  # feed in testResults dataframe and out comes a model scoreboard!
  scoreboard <- data.frame()

  bool <- names(testResults) != 'obs'
  col_names <- colnames(testResults[,bool])

  for (colname in col_names){

    testResults.model <- testResults[,colname]
    cf <- caret::confusionMatrix(testResults.model,
                                 as.factor(testResults$obs),
                                 positive="Good")

    # gather testResults
    acc <- data.frame(metric=cf$overall[1])
    # gather Precision, Sensitivity, Specificity, & F1
    metrics <- list(cf$byClass[c(5,1,2,7)])
    metrics <- data.frame(Metrics=metrics)
    names(metrics) <- 'metric'
    # gather all metrics in 1 df
    metrics <- rbind(acc,metrics)
    names(metrics) <- colname

    metrics <- t(metrics)
    scoreboard <- rbind(scoreboard, metrics)
  }
  return(scoreboard)
}

# helper function for roc
roc_build <- function(model) {
  THE_ROC <- roc(response = model$pred$obs,
                 predictor = model$pred$Bad,
                 levels = rev(levels(model$pred$obs)))
  return(THE_ROC)
}
```

Discriminant Classification Models

LDA

```r
set.seed(100)
lda_model <- caret::train(x=heloc_train_x,
                          y=heloc_train_y,
                          method="lda",
                          metric="ROC",
                          trControl=control)

lda_modelRoc <- roc_build(lda_model)
```

```
## Setting direction: controls < cases
```

```r
lda_model
```

```
## Linear Discriminant Analysis
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7998689  0.7527435  0.7114538
```

```r
lda_predictions <- stats::predict(lda_model, heloc_test_x)

# create dataframe to store
testResults <- data.frame(obs=heloc_test_y,
                          lda_model=lda_predictions)

# confusion matrix
confusionMatrix(testResults$lda_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  768  288
##       Good 259  659
##
##                Accuracy : 0.7229
##                  95% CI : (0.7026, 0.7426)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4442
##
##  Mcnemar's Test P-Value : 0.2312
##
##             Sensitivity : 0.6959
##             Specificity : 0.7478
##          Pos Pred Value : 0.7179
##          Neg Pred Value : 0.7273
##              Prevalence : 0.4797
##          Detection Rate : 0.3338
##    Detection Prevalence : 0.4650
##       Balanced Accuracy : 0.7218
##
##        'Positive' Class : Good
##
```

Logistic Regression

```
set.seed(100)
logreg_model <- caret::train(x=heloc_train_x,
                             y=heloc_train_y,
                             method="glm",
                             metric="ROC",
                             trControl=control)

testResults$log_reg_model <- stats::predict(logreg_model, heloc_test_x)
logreg_modelRoc <- roc_build(logreg_model)
```

```
## Setting direction: controls < cases
```

```
logreg_model
```

```
## Generalized Linear Model
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8000842  0.7554216  0.7130383
```

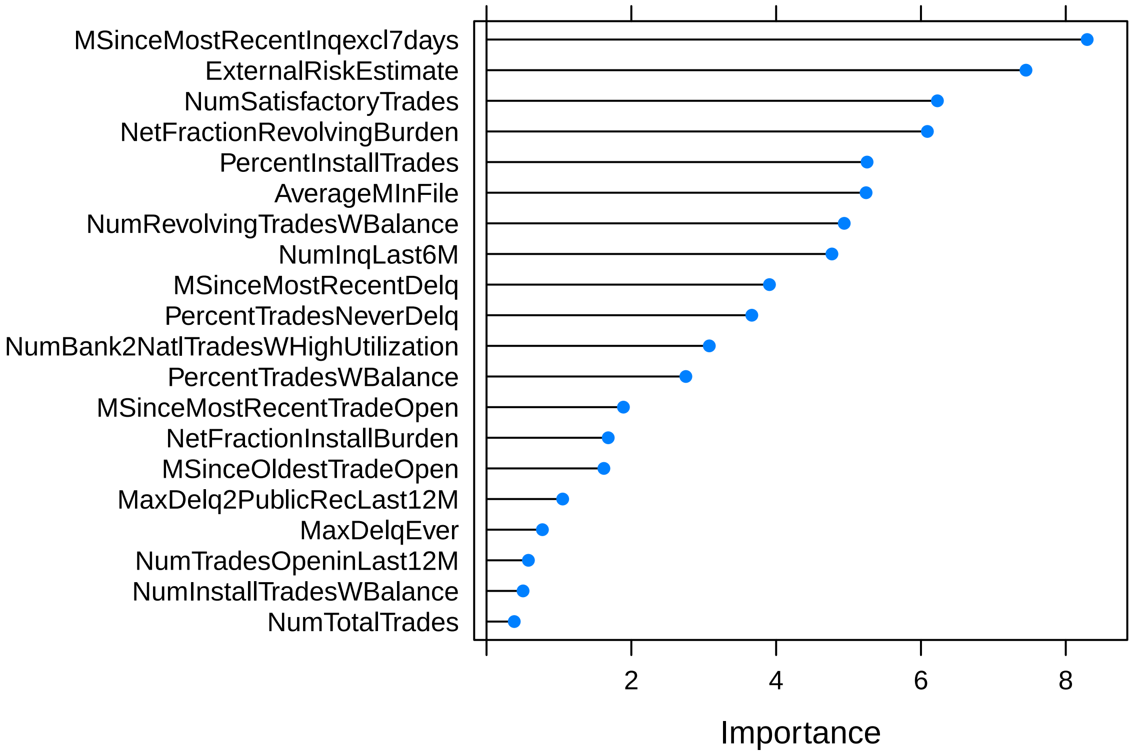
```
logreg_model$finalModel$coefficients
```

```
##                   (Intercept)                    AverageMInFile
##                 -0.1803777116                      0.2470129097
##          ExternalRiskEstimate            MaxDelq2PublicRecLast12M
##                  0.4763324385                      0.0454362796
##                   MaxDelqEver                MSinceMostRecentDelq
##                  0.0320452303                      0.1630765084
##      MSinceMostRecentInqexcl7days          MSinceMostRecentTradeOpen
##                  0.2726958674                     -0.0650335190
##           MSinceOldestTradeOpen            NetFractionInstallBurden
##                  0.0649987498                     -0.0543543815
##        NetFractionRevolvingBurden NumBank2NatlTradesWHighUtilization
##                 -0.2929249411                     -0.1376246387
##                  NumInqLast6M            NumInstallTradesWBalance
##                 -0.1638154810                      0.0167061783
##        NumRevolvingTradesWBalance               NumSatisfactoryTrades
##                 -0.2278121804                      0.3689635005
##                 NumTotalTrades         NumTrades90Ever2DerogPubRec
##                  0.0194119014                      0.0001783062
##          NumTradesOpeninLast12M               PercentInstallTrades
##                 -0.0196622137                     -0.1878716750
##          PercentTradesNeverDelq               PercentTradesWBalance
##                  0.1716749452                      0.1173618900
```

```r
confusionMatrix(testResults$log_reg_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  766  288
##       Good 261  659
##
##                Accuracy : 0.7219
##                  95% CI : (0.7015, 0.7416)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4422
##
##  Mcnemar's Test P-Value : 0.2671
##
##             Sensitivity : 0.6959
##             Specificity : 0.7459
##          Pos Pred Value : 0.7163
##          Neg Pred Value : 0.7268
##              Prevalence : 0.4797
##          Detection Rate : 0.3338
##    Detection Prevalence : 0.4661
##       Balanced Accuracy : 0.7209
##
##        'Positive' Class : Good
##
```

```r
lr_varImp <- caret::varImp(logreg_model, scale=FALSE)
plot(lr_varImp, top=20)
```



####

Cost Matrix Threshold Analysis

```
#get raw probs from model
predictions <- predict(logreg_model, heloc_test_x, type = 'prob')
predictions$OBS <- as.factor(heloc_test$RiskPerformance)
predictions <- predictions %>%
  mutate(lr10 = as.factor(if_else(Bad > 0.1, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr20 = as.factor(if_else(Bad > 0.2, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr30 = as.factor(if_else(Bad > 0.3, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr40 = as.factor(if_else(Bad > 0.4, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr50 = as.factor(if_else(Bad > 0.5, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr60 = as.factor(if_else(Bad > 0.6, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr70 = as.factor(if_else(Bad > 0.7, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr80 = as.factor(if_else(Bad > 0.8, 'Bad', 'Good')))
predictions <- predictions %>%
  mutate(lr90 = as.factor(if_else(Bad > 0.9, 'Bad', 'Good')))

# cf function
cost_confusionMatrix <- function(prediction.rate){
  cm <- caret::confusionMatrix(prediction.rate,
                               predictions$OBS,
                               positive = "Bad")
  return(cm)
}
```

```r
CF10 <- cost_confusionMatrix(predictions$lr10)
CF20 <- cost_confusionMatrix(predictions$lr20)
CF30 <- cost_confusionMatrix(predictions$lr30)
CF40 <- cost_confusionMatrix(predictions$lr40)
CF50 <- cost_confusionMatrix(predictions$lr50)
CF60 <- cost_confusionMatrix(predictions$lr60)
CF70 <- cost_confusionMatrix(predictions$lr70)
CF80 <- cost_confusionMatrix(predictions$lr80)
CF90 <- cost_confusionMatrix(predictions$lr90)

Costs = matrix(c(0,-1000*.85,-60,60), ncol=2, nrow=2)

Prev = matrix(c(9.6/50,9.6/50,2,2), ncol=2, nrow=2)

CF10$table
```

```
##           Reference
## Prediction  Bad Good
##       Bad  1023  895
##       Good    4   52
```

```r
sum(CF10$table*Costs*Prev)
```

```
## [1] -101812.8
```

```r
CF20$table
```

```
##           Reference
## Prediction Bad Good
##       Bad  993  730
##       Good  34  217
```

```r
sum(CF20$table*Costs*Prev)
```

```
## [1] -67108.8
```

```r
CF30$table
```

```
##           Reference
## Prediction Bad Good
##       Bad  934  531
##       Good  93  416
```

```r
sum(CF30$table*Costs*Prev)
```

```
## [1] -28977.6
```

```r
CF40$table
```

```
##           Reference
## Prediction Bad Good
##       Bad  859  378
##       Good 168  569
```

```r
sum(CF40$table*Costs*Prev)
```

```
## [1] -4497.6
```

```r
CF50$table
```

```
##            Reference
## Prediction Bad Good
##       Bad  766  288
##       Good 261  659
```

```
sum(CF50$table*Costs*Prev)
```

```
## [1] 1924.8
```

```
CF60$table
```

```
##            Reference
## Prediction Bad Good
##       Bad  652  180
##       Good 375  767
```

```
sum(CF60$table*Costs*Prev)
```

```
## [1] 9240
```

```
CF70$table
```

```
##            Reference
## Prediction Bad Good
##       Bad  510  103
##       Good 517  844
```

```
sum(CF70$table*Costs*Prev)
```

```
## [1] 4545.6
```

```
CF80$table
```

```
##            Reference
## Prediction Bad Good
##       Bad  318   36
##       Good 709  911
```

```
sum(CF80$table*Costs*Prev)
```

```
## [1] -10708.8
```

```
CF90$table
```

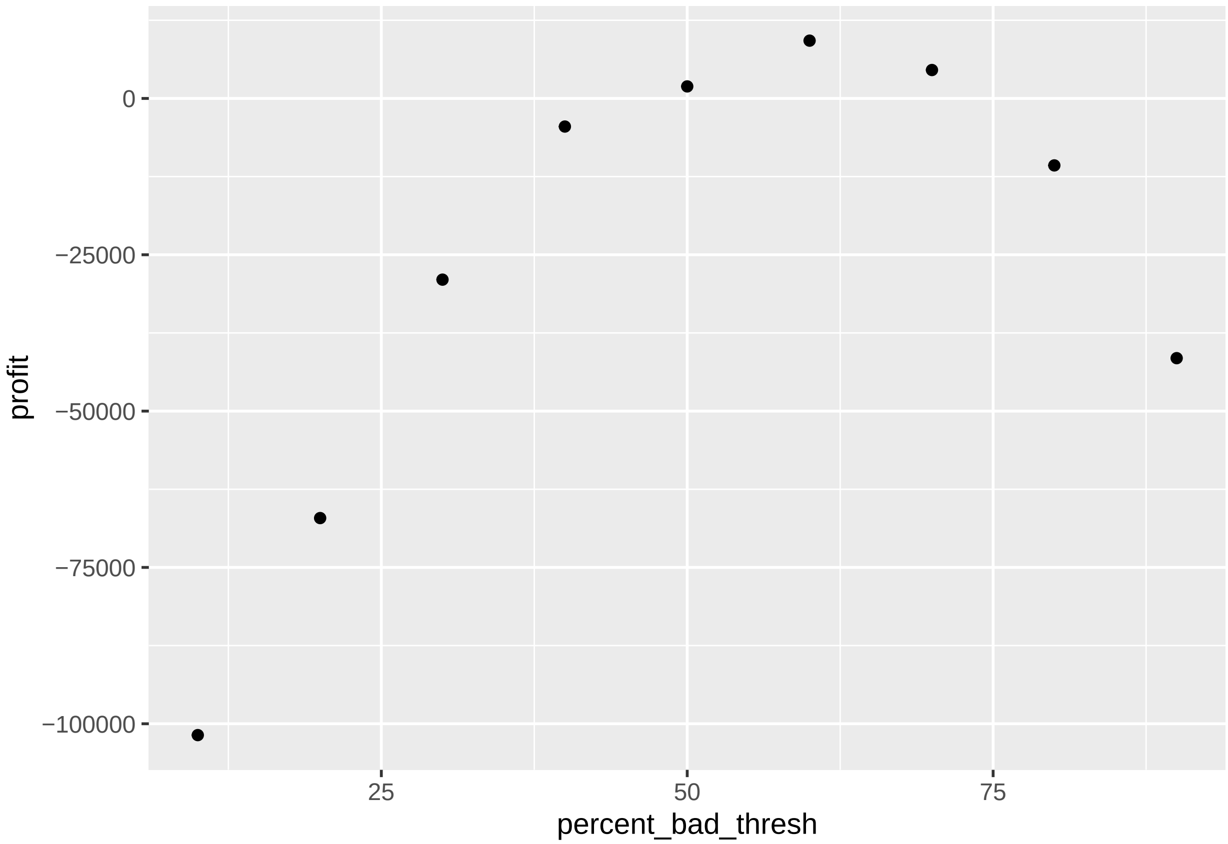
```
##            Reference
## Prediction Bad Good
##       Bad   85    6
##       Good 942  941
```

```
sum(CF90$table*Costs*Prev)
```

```
## [1] -41534.4
```

```
PlotProf<-data.frame(percent_bad_thresh = c(10,20,30,40,50,60,70,80,90),
                     profit = c(sum(CF10$table*Costs*Prev),
                                sum(CF20$table*Costs*Prev),
                                sum(CF30$table*Costs*Prev),
                                sum(CF40$table*Costs*Prev),
                                sum(CF50$table*Costs*Prev),
                                sum(CF60$table*Costs*Prev),
                                sum(CF70$table*Costs*Prev),
                                sum(CF80$table*Costs*Prev),
```

```
                                                sum(CF90$table*Costs*Prev)))

ggplot(PlotProf, aes(y=profit, x=percent_bad_thresh)) +
    geom_point()
```



Penalized Logistic Regression

```
set.seed(100)
glmnGrid <- expand.grid(alpha=c(0, 0.1, 0.2, 0.4, 0.6, 0.8, 1),
                        lambda=seq(0.01, 0.2, length=5))

logreg_penalized_model <- caret::train(x=heloc_train_x,
                                       y=heloc_train_y,
                                       method="glmnet",
                                       metric="ROC",
                                       tuneGrid=glmnGrid,
                                       trControl=control)

# bestIndex(logreg_penalized_model)
logreg_penalized_model$results[3:7,1:5]
```

```
##   alpha lambda       ROC      Sens      Spec
## 3   0.0 0.1050 0.7987157 0.7583402 0.7043291
## 4   0.0 0.1525 0.7981792 0.7607732 0.7024836
## 5   0.0 0.2000 0.7976588 0.7619904 0.7011636
## 6   0.1 0.0100 0.8001371 0.7551771 0.7114524
```

```
## 7   0.1 0.0575 0.7992612 0.7551777 0.7043277
```

```r
logreg_penalized_modelRoc <- roc_build(logreg_penalized_model)
```

```
## Setting direction: controls < cases
```

Based on the best ROC, has the best specificity, our main metric. This enables us to insure that we only accept best qualified candidates, thus reducing the risk of a loanee defaulting on a $100,000 loan.

```r
# Utilizing Best Model
testResults$logreg_penalized_model <- stats::predict(logreg_penalized_model,
                                                     heloc_test_x)
# confusion matrix
confusionMatrix(testResults$logreg_penalized_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  773  282
##       Good 254  665
##
##                Accuracy : 0.7285
##                  95% CI : (0.7083, 0.748)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4554
##
##  Mcnemar's Test P-Value : 0.2435
##
##             Sensitivity : 0.7022
##             Specificity : 0.7527
##          Pos Pred Value : 0.7236
##          Neg Pred Value : 0.7327
##              Prevalence : 0.4797
##          Detection Rate : 0.3369
##    Detection Prevalence : 0.4656
##       Balanced Accuracy : 0.7274
##
##        'Positive' Class : Good
##
```

Nonlinear Classification Models

Flexibble Discriminant Analysis

```r
# set.seed(100)
# fdaGrid <- expand.grid(degree=c(1,2),
#                        nprune=seq(14, 20, 1))
#
# set.seed(100)
# fdaModel <- caret::train(x = heloc_train_x,
#                          y = heloc_train_y,
#                          method = "fda",
#                          metric = "ROC",
```

```r
#                          trControl=control,
#                          tuneGrid=fdaGrid)
# # bestIndex(fdaModel)
# fdaModel$results[1:18,1:5]
```

As we see, the best is that of nprune 16 with a specificity of 72.15% (index = 15).

```r
fdaGrid <- expand.grid(degree=1,
                       nprune=16)

set.seed(100)
fdaModel <- caret::train(x = heloc_train_x,
                         y = heloc_train_y,
                         method = "fda",
                         metric = "ROC",
                         trControl=control,
                         tuneGrid=fdaGrid)
```

```
## Loading required package: earth
```

```
## Loading required package: Formula
```

```
## Loading required package: plotmo
```

```
## Loading required package: plotrix
```

```
## Loading required package: TeachingDemos
```

```r
fda_modelRoc <- roc_build(fdaModel)
```

```
## Setting direction: controls < cases
```

```r
testResults$fda_model <- predict(fdaModel,
                                 heloc_test_x)
# confusion matrix
confusionMatrix(testResults$fda_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  762  265
##       Good 265  682
##
##                Accuracy : 0.7315
##                  95% CI : (0.7114, 0.751)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4621
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7202
##             Specificity : 0.7420
##          Pos Pred Value : 0.7202
##          Neg Pred Value : 0.7420
##              Prevalence : 0.4797
```

```
##          Detection Rate : 0.3455
##    Detection Prevalence : 0.4797
##       Balanced Accuracy : 0.7311
##
##        'Positive' Class : Good
##
```

Neural Network

```r
# set.seed(100)
# nnetGrid <- expand.grid(size = 1:2,
#                         decay = c(0, 0.1, 0.25, 0.5, 0.75, 1))
#
# nnetModel <- caret::train(x = heloc_train_x,
#                           y = heloc_train_y,
#                           method = "nnet",
#                           tuneGrid = nnetGrid,
#                           metric = "ROC",
#                           trace = FALSE,
#                           maxit = 2000,
#                           trControl = control)
# nnetModel$bestTune$results[1:6,1:5]
```

It appears that the model starts to over fit once the decay goes to 0.1. The nnet model chose size=2 with decay of 2, with nearly identical ROC, Sensitivity, and Specificity scores and thus size 1 with decay 0 is our choice.

```r
set.seed(100)
nnetGrid <- expand.grid(size = 1,
                        decay = 0)

nnetModel <- caret::train(x = heloc_train_x,
                   y = heloc_train_y,
                   method = "nnet",
                   tuneGrid = nnetGrid,
                   metric = "ROC",
                   trace = FALSE,
                   maxit = 2000,
                   trControl = control)

nnet_modelRoc <- roc_build(nnetModel)
```

```
## Setting direction: controls < cases
```

```r
testResults$nnet_model <- predict(nnetModel,
                                  heloc_test_x)
# confusion matrix
confusionMatrix(testResults$nnet_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  769  291
##       Good 258  656
##
```

```
##                Accuracy : 0.7219
##                  95% CI : (0.7015, 0.7416)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4421
##
##  Mcnemar's Test P-Value : 0.172
##
##             Sensitivity : 0.6927
##             Specificity : 0.7488
##          Pos Pred Value : 0.7177
##          Neg Pred Value : 0.7255
##              Prevalence : 0.4797
##          Detection Rate : 0.3323
##    Detection Prevalence : 0.4630
##       Balanced Accuracy : 0.7207
##
##        'Positive' Class : Good
##
```

Classification Trees

Boosted Tree

```
# gbmGrid <- expand.grid(interaction.depth =  c(2,3),
#                         n.trees = c(1000,2000,3000,4000), #default val = 1000
#                         shrinkage = c(0.01, 0.1),
#                         n.minobsinnode = c(5,10)) # default val = 10
# set.seed(100)
# gbmModel <- caret::train(x = heloc_train_x,
#                           y = heloc_train_y,
#                           method = "gbm",
#                           tuneGrid = gbmGrid,
#                           verbose = FALSE,
#                           metric = "ROC",
#                           trControl= control)
# gbmModel$results
```

```
gbmGrid <- expand.grid(interaction.depth = 2,
                       n.trees = 1000,
                       shrinkage = 0.01,
                       n.minobsinnode = 5)
set.seed(100)
gbmModel <- caret::train(x = heloc_train_x,
                         y = heloc_train_y,
                         method = "gbm",
                         tuneGrid = gbmGrid,
                         verbose = FALSE,
                         metric = "ROC",
                         trControl= control)

gbm_modelRoc <- roc_build(gbmModel)
```

```
## Setting direction: controls < cases
```

```r
testResults$gbm_model <- predict(gbmModel,
                                 heloc_test_x)
# confusion matrix
confusionMatrix(testResults$gbm_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  775  285
##       Good 252  662
##
##                Accuracy : 0.728
##                  95% CI : (0.7078, 0.7475)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4543
##
##  Mcnemar's Test P-Value : 0.1673
##
##             Sensitivity : 0.6990
##             Specificity : 0.7546
##          Pos Pred Value : 0.7243
##          Neg Pred Value : 0.7311
##              Prevalence : 0.4797
##          Detection Rate : 0.3354
##    Detection Prevalence : 0.4630
##       Balanced Accuracy : 0.7268
##
##        'Positive' Class : Good
##
```

```r
# gbmGrid <- expand.grid(interaction.depth = c(2,3),
#                        n.trees = c(1000,2000),
#                        shrinkage = c(0.01, 0.1),
#                        n.minobsinnode = c(5,10))
# set.seed(100)
# gbmMono_model <- caret::train(x = heloc_train_x,
#                               y = heloc_train_y,
#                               method = "gbm",
#                               var.monotone = c(-1,-1,-1,-1,-1,-1,-1,-1,
#                                                1,1,1,1,0,0,-1,0,-1,1,
#                                                0,-1,1),
#                               tuneGrid = gbmGrid,
#                               verbose = FALSE,
#                               metric = "ROC",
#                               trControl= control)
# gbmMono_model$results
```

```r
gbmGrid <- expand.grid(interaction.depth = 2,
                       n.trees = 1000,
```

```r
                        shrinkage = 0.01,
                        n.minobsinnode = 5)
set.seed(100)
gbmMono_model <- caret::train(x = heloc_train_x,
                          y = heloc_train_y,
                          var.monotone = c(-1,-1,-1,-1,-1,-1,-1,-1,
                                           1,1,1,1,0,0,-1,0,-1,1,
                                           0,-1,1),
                          method = "gbm",
                          tuneGrid = gbmGrid,
                          verbose = FALSE,
                          metric = "ROC",
                          trControl= control)

gbmMono_modelRoc <- roc_build(gbmMono_model)
```

GBM with monotonic constraints

```
## Setting direction: controls < cases
```

```r
testResults$gbmMono_model <- predict(gbmMono_model,
                                     heloc_test_x)
confusionMatrix(testResults$gbmMono_model)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  772  281
##       Good 255  666
##
##                Accuracy : 0.7285
##                  95% CI : (0.7083, 0.748)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4555
##
##  Mcnemar's Test P-Value : 0.2802
##
##             Sensitivity : 0.7033
##             Specificity : 0.7517
##          Pos Pred Value : 0.7231
##          Neg Pred Value : 0.7331
##              Prevalence : 0.4797
##          Detection Rate : 0.3374
##    Detection Prevalence : 0.4666
##       Balanced Accuracy : 0.7275
##
##        'Positive' Class : Good
##
```

CART

```
# set.seed(100)
# rpart_grid <- expand.grid(cp=c(0.0005, 0.001250, 0.0015, 0.00175, 0.002))
#
# rpart_model <- caret::train(x=heloc_train_x,
#                              y=heloc_train_y,
#                              method="rpart",
#                              metric="ROC",
#                              trControl=control,
#                              tuneGrid = rpart_grid)
#
# testResults$rpart_model <- predict(rpart_model, heloc_test_x)
#
# rpart_model
```

As we see, specificity for this model is one of the worst out of all the models we have.

```
set.seed(100)
rpart_grid <- expand.grid(cp=0.00175)

rpart_model <- caret::train(x=heloc_train_x,
                            y=heloc_train_y,
                            method="rpart",
                            metric="ROC",
                            trControl=control,
                            tuneGrid = rpart_grid)

rpart_modelRoc <- roc_build(rpart_model)
```

```
## Setting direction: controls < cases
```

```
rpart_model
```

```
## CART
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7558931  0.7551742  0.671846
##
## Tuning parameter 'cp' was held constant at a value of 0.00175
```

```
testResults$rpart_model <- predict(rpart_model, heloc_test_x)

confusionMatrix(testResults$rpart_model)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction Bad Good
##       Bad  748  282
##       Good 279  665
##
##                Accuracy : 0.7158
##                  95% CI : (0.6953, 0.7356)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4306
##
##  Mcnemar's Test P-Value : 0.9327
##
##             Sensitivity : 0.7022
##             Specificity : 0.7283
##          Pos Pred Value : 0.7044
##          Neg Pred Value : 0.7262
##              Prevalence : 0.4797
##          Detection Rate : 0.3369
##    Detection Prevalence : 0.4782
##       Balanced Accuracy : 0.7153
##
##        'Positive' Class : Good
##
```

Random Forest

```
# set.seed(100)
# rf_grid <- expand.grid(mtry=c(5,10,15))
#
# randomForest_model <- caret::train(x=heloc_train_x,
#                             y=heloc_train_y,
#                             method="rf",
#                             metric="ROC",
#                             trControl=control,
#                             tuneGrid = rf_grid)
#
# randomForest_model
```

```
set.seed(100)
rf_grid <- expand.grid(mtry=5)

randomForest_model <- caret::train(x=heloc_train_x,
                                   y=heloc_train_y,
                                   method="rf",
                                   metric="ROC",
                                   trControl=control,
                                   tuneGrid = rf_grid)

randomForest_modelRoc <- roc_build(randomForest_model)
```

```
## Setting direction: controls < cases
```

```r
randomForest_model
```

```
## Random Forest
##
## 7897 samples
##   21 predictor
##    2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7107, 7107, 7107, 7108, 7107, 7108, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.7951729  0.7736746  0.6847978
##
## Tuning parameter 'mtry' was held constant at a value of 5
```

```r
testResults$randomForest_model <- predict(randomForest_model, heloc_test_x)

confusionMatrix(testResults$randomForest_model)
```

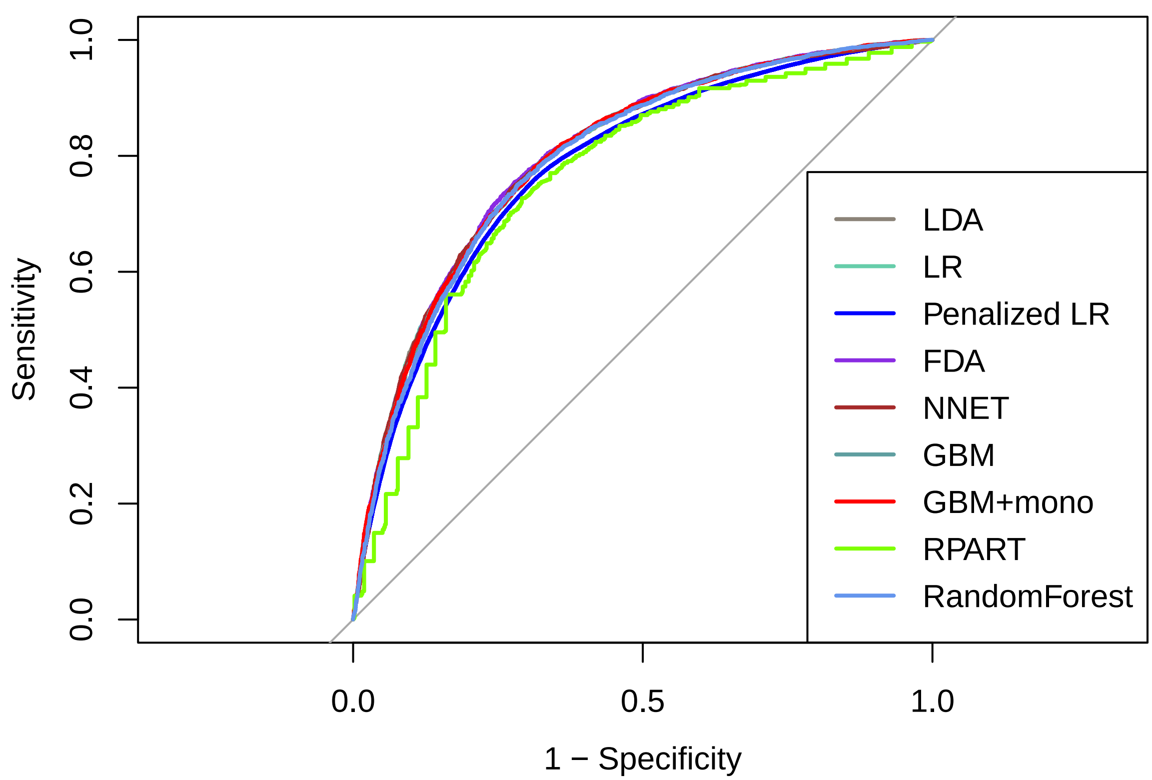
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad  790  291
##       Good 237  656
##
##                Accuracy : 0.7325
##                  95% CI : (0.7124, 0.7519)
##     No Information Rate : 0.5203
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.463
##
##  Mcnemar's Test P-Value : 0.02108
##
##             Sensitivity : 0.6927
##             Specificity : 0.7692
##          Pos Pred Value : 0.7346
##          Neg Pred Value : 0.7308
##              Prevalence : 0.4797
##          Detection Rate : 0.3323
##    Detection Prevalence : 0.4524
##       Balanced Accuracy : 0.7310
##
##        'Positive' Class : Good
##
```

Result Discussion

```r
plot(lda_modelRoc, type='s', col='antiquewhite4', legacy.axes=TRUE)
plot(logreg_modelRoc, type='s', col='aquamarine3', legacy.axes=TRUE, add=TRUE)
```

```
plot(logreg_penalized_modelRoc, type='s', col='blue', legacy.axes=TRUE, add=TRUE)
plot(fda_modelRoc, type='s', col='blueviolet', legacy.axes=TRUE, add=TRUE)
plot(nnet_modelRoc, type='s', col='brown', legacy.axes=TRUE, add=TRUE)
plot(gbm_modelRoc, type='s', col='cadetblue', legacy.axes=TRUE, add=TRUE)
plot(gbmMono_modelRoc, type='s', col='red', legacy.axes=TRUE, add=TRUE)
plot(rpart_modelRoc, type='s', col='chartreuse', legacy.axes=TRUE, add=TRUE)
plot(randomForest_modelRoc, type='s', col='cornflowerblue', legacy.axes=TRUE, add=TRUE)

legend_ <- c('LDA', 'LR', 'Penalized LR','FDA', 'NNET','GBM','GBM+mono', 'RPART','RandomForest')
colors_ <-c('antiquewhite4',
            'aquamarine3',
            'blue',
            'blueviolet',
            'brown',
            'cadetblue',
            'red',
            'chartreuse',
            'cornflowerblue')
legend('bottomright', legend=legend_,
       col=colors_, lwd=2)
title(main = 'Compare ROC curves from different models', outer = TRUE)
```

Compare ROC curves from different models




```
# gather all model AUCs in 1 list
aucs <- c(lda_modelRoc$auc,
          logreg_modelRoc$auc,
          logreg_penalized_modelRoc$auc,
          fda_modelRoc$auc,
```

```r
          nnet_modelRoc$auc,
          gbm_modelRoc$auc,
          gbmMono_modelRoc$auc,
          rpart_modelRoc$auc,
          randomForest_modelRoc$auc)

scoreboard <- modelScoreBoard(testResults)
# add AUC list as a column
scoreboard$AUC <- aucs
scoreboard
```

```
##                        Accuracy Precision Sensitivity Specificity        F1
## lda_model             0.7228977 0.7178649   0.6958817   0.7478092 0.7067024
## log_reg_model         0.7218845 0.7163043   0.6958817   0.7458617 0.7059454
## logreg_penalized_model 0.7284701 0.7236126   0.7022175   0.7526777 0.7127546
## fda_model             0.7315096 0.7201690   0.7201690   0.7419669 0.7201690
## nnet_model            0.7218845 0.7177243   0.6927138   0.7487829 0.7049973
## gbm_model             0.7279635 0.7242888   0.6990496   0.7546251 0.7114455
## gbmMono_model         0.7284701 0.7231270   0.7032735   0.7517040 0.7130621
## rpart_model           0.7158055 0.7044492   0.7022175   0.7283350 0.7033316
## randomForest_model    0.7325228 0.7346025   0.6927138   0.7692308 0.7130435
##                             AUC
## lda_model             0.7998810
## log_reg_model         0.8002390
## logreg_penalized_model 0.7820838
## fda_model             0.8032575
## nnet_model            0.8001313
## gbm_model             0.7998948
## gbmMono_model         0.8003082
## rpart_model           0.7554376
## randomForest_model    0.7950744
```

REFERENCES:

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. New York: Springer.