Monotonic Optimal Binning (MOB)

What is MOB:
The MOB package is a collection of R functions that generate the monotonic binning.

Why need MOB:
- Perform WoE (Weight of Evidence) transformation for numeric attributes in the scorecard development of consumer credit risk.
- Convert missing values and outliers to WoE transformation through the discretization.
- Derive IV (Information Value) to assess the variable importance through the calculation of WoE

How to install MOB:
- Download "mob_0.1.0.tar.gz" from https://github.com/statcompute/mob and then save the tar ball in the working folder
- Make sure to install dependent packages, e.g. base, stats, parallel, gbm, Hmisc
- Install MOB from the source by running

```
install.packages(" mob_0.1.0.tar.gz ", repos = NULL, type = "source")
```

MOB Functionalities

Binning functions:
- qtl_bin(): The binning algorithm based on the iterative discretization by quantiles for the whole development data sample
- iso_bin(): The binning algorithm based on the isotonic regression
- bad_bin(): The binning algorithm based on the iterative discretization by quantiles for the data sample with Y = 1
- gbm_bin(): The binning algorithm based on the generalized boosted modeling with the monotone restriction.
- arb_bin(): The binning algorithm based on the decision tree with the monotone restriction.
- batch_bin(): A wrapper for above binning algorithms to apply to all numeric variables in the data frame with the last column as Y.

Deployment functions:
- cal_woe(): Perform WoE transformation based on the spec file from the binning output.
- batch_woe(): A wrapper for cal_woe() to apply to all numeric variables in the data frame based on a list of spec from batch_bin().

Example 1: How to Use

```
df <- readRDS("df.rds")

head(df, 2)
# tot_derog tot_tr age_oldest_tr tot_open_tr tot_rev_tr tot_rev_debt ⋯
#         6      7            46         NaN        NaN           NaN ⋯
#         0     21           153           6          1            97 ⋯

qtl_bin(df, bad, tot_derog)
# $df
#  bin                          rule freq   dist mv_cnt bad_freq bad_rate     woe     iv      ks
#   00                   is.na($X)  213 0.0365    213       70   0.3286  0.6416 0.0178  2.7716
#   01                     $X <= 1 3741 0.6409      0      560   0.1497 -0.3811 0.0828 18.9469
#   02           $X > 1 & $X <= 2  478 0.0819      0      121   0.2531  0.2740 0.0066 16.5222
#   03           $X > 2 & $X <= 4  587 0.1006      0      176   0.2998  0.5078 0.0298 10.6623
#   04                     $X > 4  818 0.1401      0      269   0.3289  0.6426 0.0685  0.0000
# $cuts
# [1] 1 2 4
```

```
bad_bin(df, bad, tot_derog)
# $df
#  bin                          rule freq   dist mv_cnt bad_freq bad_rate     woe     iv      ks
#   00                    is.na($X)  213 0.0365    213       70   0.3286  0.6416 0.0178  2.7716
#   01                      $X <= 2 4219 0.7228      0      681   0.1614 -0.2918 0.0563 16.5222
#   02           $X > 2 & $X <= 4  587 0.1006      0      176   0.2998  0.5078 0.0298 10.6623
#   03                      $X > 4  818 0.1401      0      269   0.3289  0.6426 0.0685  0.0000
# $cuts
# [1] 2 4

iso_bin(df, bad, tot_derog)
# $df
#  bin                          rule freq   dist mv_cnt bad_freq bad_rate     woe     iv      ks
#   00                    is.na($X)  213 0.0365    213       70   0.3286  0.6416 0.0178  2.7716
#   01                      $X <= 0 2850 0.4883      0      367   0.1288 -0.5559 0.1268 20.0442
#   02           $X > 0 & $X <= 1  891 0.1526      0      193   0.2166  0.0704 0.0008 18.9469
#   03           $X > 1 & $X <= 2  478 0.0819      0      121   0.2531  0.2740 0.0066 16.5222
#   04           $X > 2 & $X <= 3  332 0.0569      0       86   0.2590  0.3050 0.0058 14.6321
#   05          $X > 3 & $X <= 23 1064 0.1823      0      353   0.3318  0.6557 0.0931  0.4370
#   06                     $X > 23    9 0.0015      0        6   0.6667  2.0491 0.0090  0.0000
# $cuts
# [1]  0  1  2  3 23
```

```
### Binning in batch with qtl_bin() as the back-end
batch_bin(df, 1)
# |var             |  nbin|  unique|  miss|  min|   median|      max|       ks|      iv|
# |:---------------|-----:|-------:|-----:|----:|--------:|--------:|--------:|-------:|
# |tot_derog       |     5|      29|   213|    0|      0.0|       32|  18.9469|  0.2055|
# |tot_tr          |     5|      67|   213|    0|     16.0|       77|  15.7052|  0.1302|
# |age_oldest_tr   |    10|     460|   216|    1|    137.0|      588|  19.9821|  0.2539|
# |tot_open_tr     |     3|      26|  1416|    0|      5.0|       26|   6.7157|  0.0240|
# |tot_rev_tr      |     3|      21|   636|    0|      3.0|       24|   9.0104|  0.0717|
# |tot_rev_debt    |     3|    3880|   477|    0|   3009.5|    96260|   8.5102|  0.0627|
# |tot_rev_line    |     9|    3617|   477|    0|  10573.0|   205395|  26.4924|  0.4077|
# |rev_util        |     2|     101|     0|    0|     30.0|      100|  15.1570|  0.0930|
# |bureau_score    |    12|     315|   315|  443|    692.5|      848|  34.8028|  0.7785|
# |ltv             |     7|     145|     1|    0|    100.0|      176|  15.6254|  0.1538|
# |tot_income      |     4|    1639|     5|    0|   3400.0|  8147167|   9.1526|  0.0500|

batch_bin(df, 1)$BinLst[["tot_income"]]
# $df
#  bin                          rule freq   dist mv_cnt bad_freq bad_rate     woe     iv     ks
#   00                    is.na($X)    5 0.0009      5        1   0.2000 -0.0303 0.0000 0.0026
#   01                   $X <= 2570 1947 0.3336      0      486   0.2496  0.2553 0.0234 9.1526
#   02         $X > 2570 & $X <= 4510 1995 0.3418      0      406   0.2035 -0.0086 0.0000 8.8608
#   03                    $X > 4510 1890 0.3238      0      303   0.1603 -0.2999 0.0266 0.0000
# $cuts
# [1] 2570 4510
```

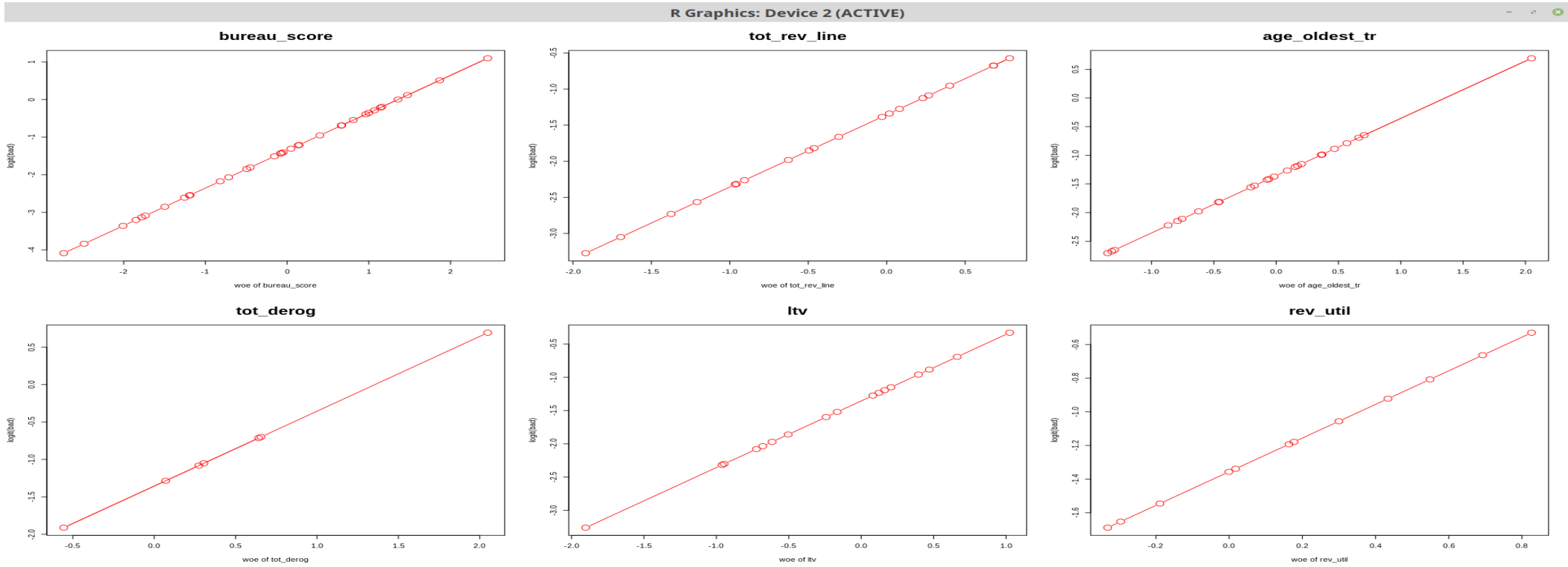
```
### How to deploy the binning outcome
ltv_bin <- qtl_bin(df, bad, ltv)

ltv_woe <- cal_woe(df[sample(seq(nrow(df)), 1000, replace = T), ], "ltv", ltv_bin$df)

str(ltv_woe, max.level = 1)
# List of 2
#  $ df :'data.frame':       1000 obs. of  13 variables:
#  $ psi:'data.frame':       7 obs. of  8 variables:
#  - attr(*, "class")= chr "psi"

ltv_woe$psi
#  bin                          rule   dist     woe cal_freq cal_dist cal_woe    psi
#   01                      $X <= 84 0.1638 -0.7690      177    0.177 -0.7690 0.0010
#   02            $X > 84 & $X <= 93 0.1645 -0.3951      143    0.143 -0.3951 0.0030
#   03            $X > 93 & $X <= 99 0.1501  0.0518      154    0.154  0.0518 0.0001
#   04           $X > 99 & $X <= 103 0.1407  0.0787      125    0.125  0.0787 0.0019
#   05          $X > 103 & $X <= 109 0.1324  0.1492      149    0.149  0.1492 0.0020
#   06          $X > 109 & $X <= 117 0.1237  0.3263      133    0.133  0.3263 0.0007
#   07          $X > 117 | is.na($X) 0.1249  0.5041      119    0.119  0.5041 0.0003

head(ltv_woe$df, 1)
# ... bureau_score ltv tot_income bad woe.ltv
# ...          667  83       2500   1  -0.769
```

```
### Generate a list of binning specifications
binout <- batch_bin(df, 1)

woeout <- batch_woe(df[sample(seq(nrow(df)), 2000, replace = T), ], binout$BinLst)

woeout
#     tot_derog tot_tr age_oldest_tr tot_open_tr tot_rev_tr tot_rev_debt ...
# psi    0.0027 0.0044        0.0144      0.0011      3e-04       0.0013 ...

str(woeout, max.level = 1)
# List of 2
#  $ psi:List of 11
#  $ df :'data.frame':       2000 obs. of  12 variables:
#  - attr(*, "class")= chr "psiSummary"

head(woeout$df, 1)
#  idx_ woe.tot_derog woe.tot_tr woe.age_oldest_tr woe.tot_open_tr woe.tot_rev_tr ...
#     1       -0.3811    -0.0215           -0.5356         -0.0722        -0.1012 ...
```

Example 2: Improve Logit Model

```
bin_out <- batch_bin(df, 3) # binning in batch with iso_bin() as the back-end

df_woe <- batch_woe(df, bin_out$BinLst)

x1 <- paste("woe", bin_out$BinSum[bin_out$BinSum[["iv"]] > 0.1, ]$var, sep = ".")  # parse vars with IV > 0.1

fml1 <- as.formula(paste("bad", paste(x1, collapse = " + "), sep = " ~ "))

sum1 <- summary(glm(fml1, data = cbind(bad = df$bad, df_woe$df), family = "binomial"))

x2 <- paste(row.names(sum1$coefficients)[sum1$coefficients[, 4] < 0.05][-1]) # parse vars with p-value < 0.05

fml2 <- as.formula(paste("bad", paste(x2, collapse = " + "), sep = " ~ "))

mdl2 <- glm(fml2, data = cbind(bad = df$bad, df_woe$df), family = "binomial")
#                   Estimate Std. Error z value Pr(>|z|)
#(Intercept)       -1.38600    0.03801 -36.461  < 2e-16 ***
#woe.age_oldest_tr  0.30376    0.08176   3.715 0.000203 ***
#woe.tot_rev_line   0.42935    0.06793   6.321 2.61e-10 ***
#woe.rev_util       0.29150    0.08721   3.342 0.000831 ***
#woe.bureau_score   0.83568    0.04974  16.803  < 2e-16 ***
#woe.ltv            0.97789    0.09121  10.721  < 2e-16 ***
```

```
top <- paste(bin_out$BinSum[order(bin_out$BinSum[["iv"]], decreasing = T), ][1:6, "var"], sep = '')

par(mfrow = c(2, 3))

lapply(top,
  function(x)
    plot(bin_out$BinLst[[x]]$df[["woe"]],
         log(bin_out$BinLst[[x]]$df[["bad_rate"]] / (1 - bin_out$BinLst[[x]]$df[["bad_rate"]])),
         type = "b", main = x, cex.main = 2, xlab = paste("woe of", x), ylab = "logit(bad)", cex = 2, col = "red"))
```



Example 3: Improve General Regression Neural Network

```
df1 <- read.table("credit_count.txt", header = T, sep = ",")

df2 <- df1[which(df1$CARDHLDR == 1), ]

Y <- df2$DEFAULT

X <- scale(df2[, 3:ncol(df2)])

i <- sample(seq(length(Y)), length(Y) / 2)

# WITHOUT BINNING
Y1 <- Y[i]

Y2 <- Y[-i]

X1 <- X[i, ]

X2 <- X[-i, ]

net11 <- grnn.fit(x = X1, y = Y1)
```

```
test1 <- grnn.search_auc(net11, gen_latin(1, 3, 10), nfolds = 4)
# $best
#    sigma       auc
# 2.198381 0.6297201

net12 <- grnn.fit(x = X1, y = Y1, sigma = test1$best$sigma)

MLmetrics::AUC(grnn.parpred(net12, X1), Y1)
# 0.6855638

MLmetrics::AUC(grnn.parpred(net12, X2), Y2)
# 0.6555798

# WITH WOE Transformation
df3 <- data.frame(df2[, 3:ncol(df2)], Y)

bin_out <- batch_bin(df3, method = 3)

df_woe <- batch_woe(df3, bin_out$BinLst)

W <- scale(df_woe$df[, -1])

W1 <- W[i, ]

W2 <- W[-i, ]
```

```
net21 <- grnn.fit(x = W1, y = Y1)

test2 <- grnn.search_auc(net21, gen_latin(1, 3, 10), nfolds = 4)
# $best
#    sigma       auc
# 2.198381 0.6820317

net22 <- grnn.fit(x = W1, y = Y1, sigma = test2$best$sigma)

MLmetrics::AUC(grnn.parpred(net22, W1), Y1)
# 0.7150051

MLmetrics::AUC(grnn.parpred(net22, W2), Y2)
# 0.6884229
```