Aquatic Toxicity Analysis

Analysis of the aquatic toxicity dataset

https://cran.r-project.org/web/packages/QSARdata/QSARdata.pdf

```
library(caret)
isida <- read.csv("aq_tox_train_isida.csv", header = F, row.names = 1)
colnames(isida) <- paste0("frag",1:401)
denis <- read.csv("aq_tox_train_des_denis_dragon.csv")
head(denis[, 1:6])
```

```
##    att_1 att_2 att_3 att_4 att_5 att_6
## 1  78.12     6  0.00     0     0 12.00
## 2 106.18     8  0.00     0     0  8.48
## 3  92.15     7  0.00     0     0 10.26
## 4 134.24    10  0.00     0     0 10.65
## 5 148.27    11  0.00     0     0 10.71
## 6 107.17     8 26.02     0     2  9.99
```

```
# response is log(IC50 -1)
response <- denis$class
denis$class <- NULL
dragon <- read.csv("aq_tox_train_des_3D_dragon.csv", row.names = 1)
dragon <- dragon[-1]

des_list <- list(isida, denis, dragon)
desc_lengths <- vector(length = 3)

d <- cbind(response)
for(i in 1:3){
  # remove low variance and highly correlated descriptors
  var_vec <- apply(des_list[[i]], 2, var)
  des_list[[i]] <- data.frame(des_list[[i]][which(var_vec > .0001)])
  des_list[[i]] <- des_list[[i]][-findCorrelation(cor(des_list[[i]]), cutoff = .9)]
  print(dim(des_list[[i]]))
  desc_lengths[i] <- ncol(des_list[[i]])
  d <- cbind(d, des_list[[i]])
}
```

```
## [1] 644 237
## [1] 644  49
## [1] 644 711
```

```
desc_lengths
```

```
## [1] 237  49 711
```

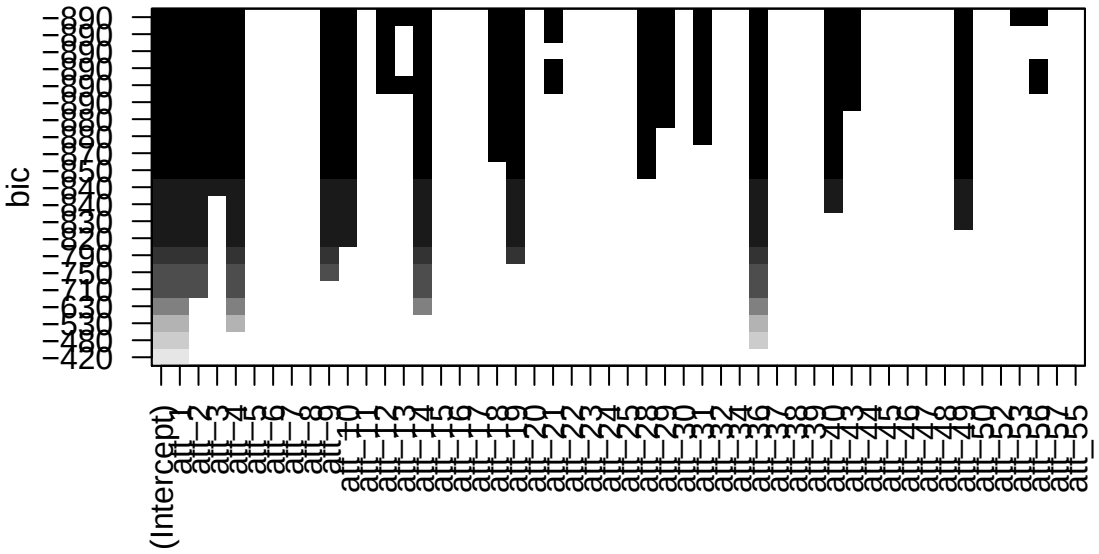
```
desc_names <- c("isida", "denis", "dragon")
```

```
codessa <- cbind(response, des_list[[2]])
regfit.fwd=regsubsets(response~.,data=codessa, method = "forward", nvmax = 20)
```

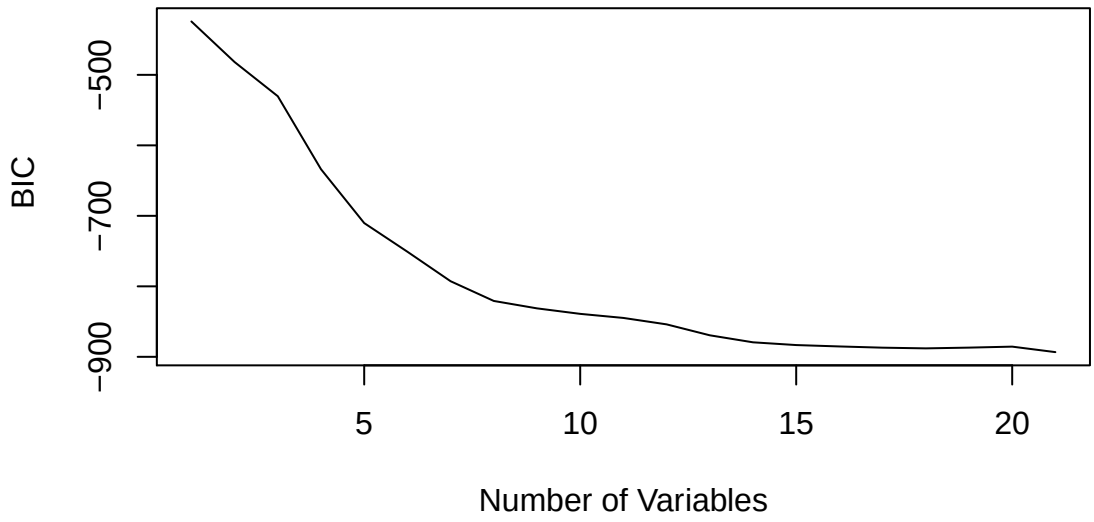
```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Reordering variables and trying again:
```

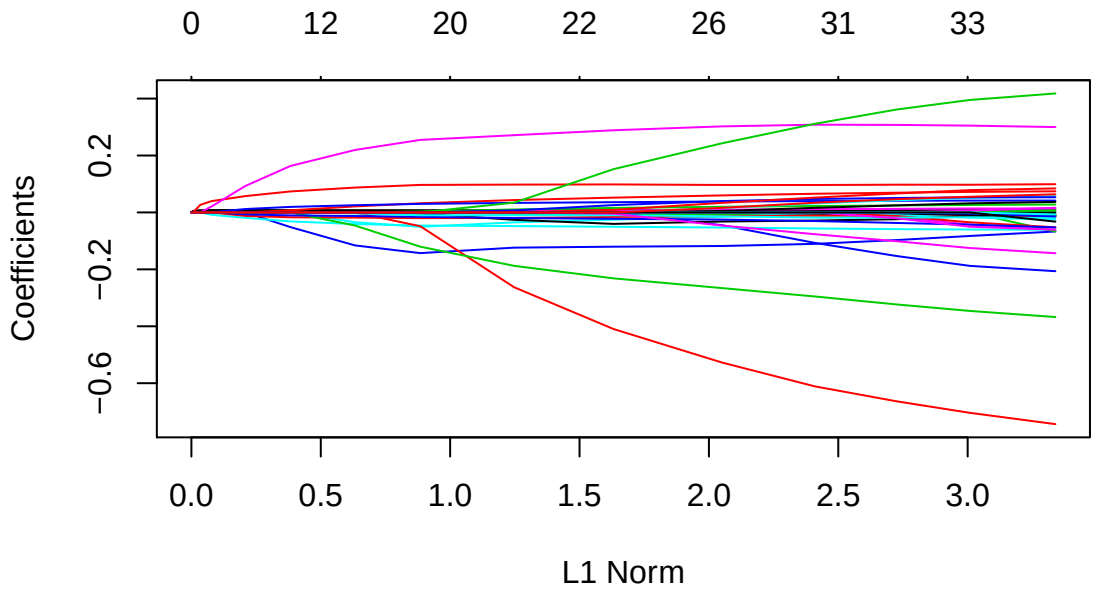
```
regfit.summary <- summary(regfit.fwd)
plot(regfit.fwd, scale ="bic")
```




```
plot(regfit.summary$bic, xlab=" Number of Variables ",ylab=" BIC",
type='l')
```



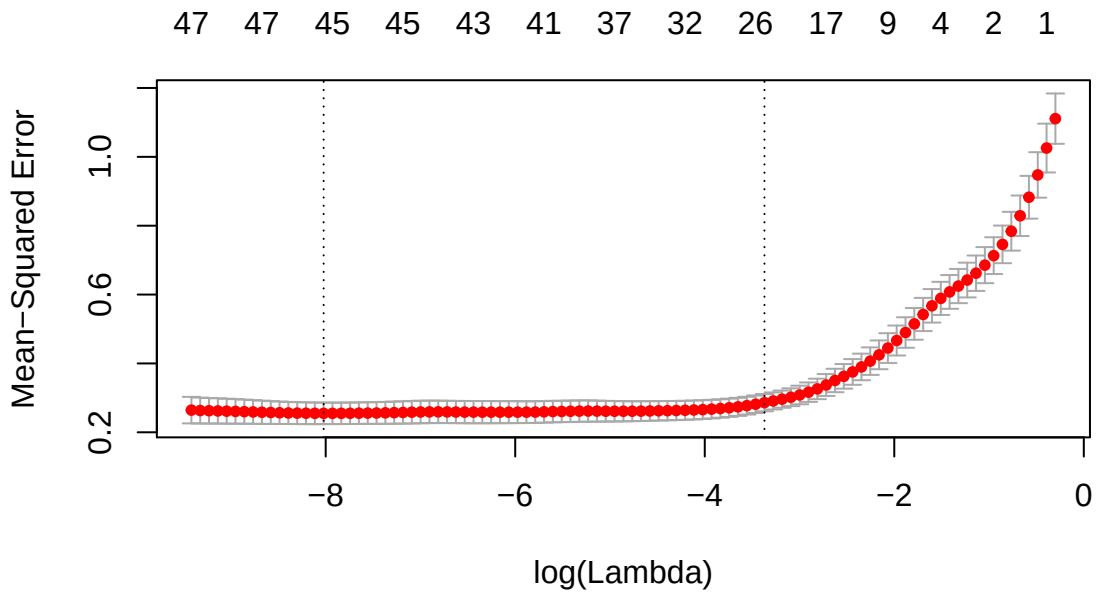

```
# locked up my computer, try running when you have more time?

isida <- cbind(response, des_list[[1]])
# head(codessa)
# regfit.fwd=regsubsets(response~.,data=isida, method = "seqrep", nvmax = 110)
# summary(regfit.seqrep)
# plot(regfit.fwd, scale ="bic")

# variable selection on codessa descriptors

x = model.matrix(response~.,data=codessa)[,-1]
y = codessa$response
grid = 10^seq(10, -2, length = 100)

lasso.mod <- glmnet(x, y, alpha = 1, lambda = grid)
plot(lasso.mod)
```




```
set.seed(835)
cv.out = cv.glmnet(x, y, alpha = 1, type.measure="mse", nfolds = 5)
bestlam =cv.out$lambda.1se
bestlam
```

```
## [1] 0.03439306
```

```
plot(cv.out)
```



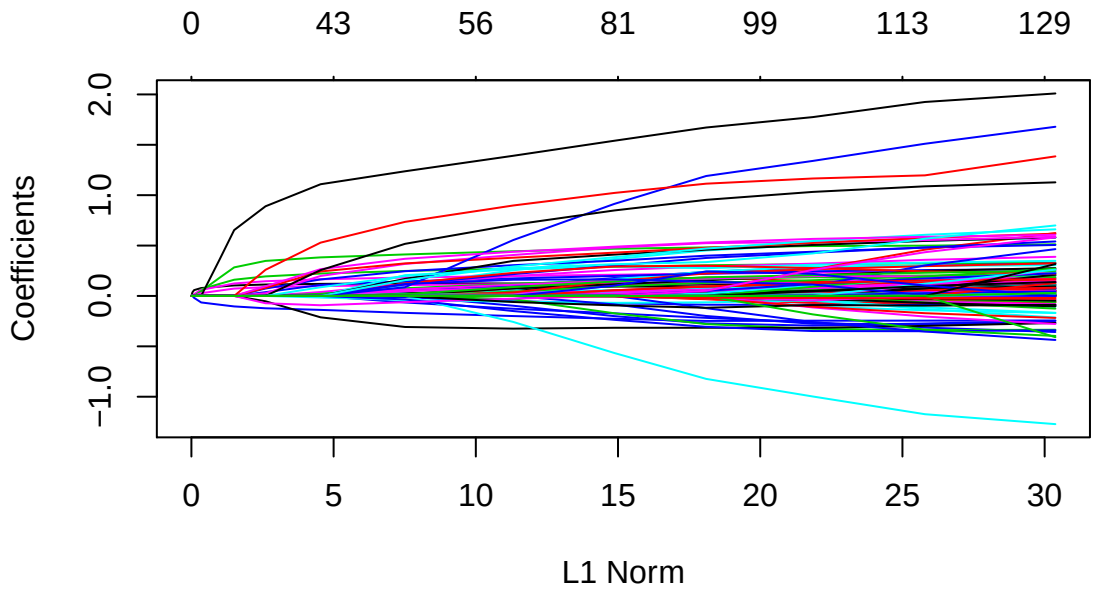

```
cv.coef <- coef(cv.out, s = bestlam)
cv.coef <- as.matrix(Matrix(cv.coef, sparse = F))
cv.coef[cv.coef > 0, ]
```

```
##       att_1       att_2       att_6       att_9      att_18      att_19
## 0.007627957 0.098596576 0.007999211 0.055907875 0.018190945 0.037208973
##      att_21      att_31      att_36      att_48      att_49      att_57
## 0.006757419 0.034719585 0.297116695 0.023341345 0.207870140 0.006460864
```

```
length(cv.coef[cv.coef > 0, ])
```

```
## [1] 12
```

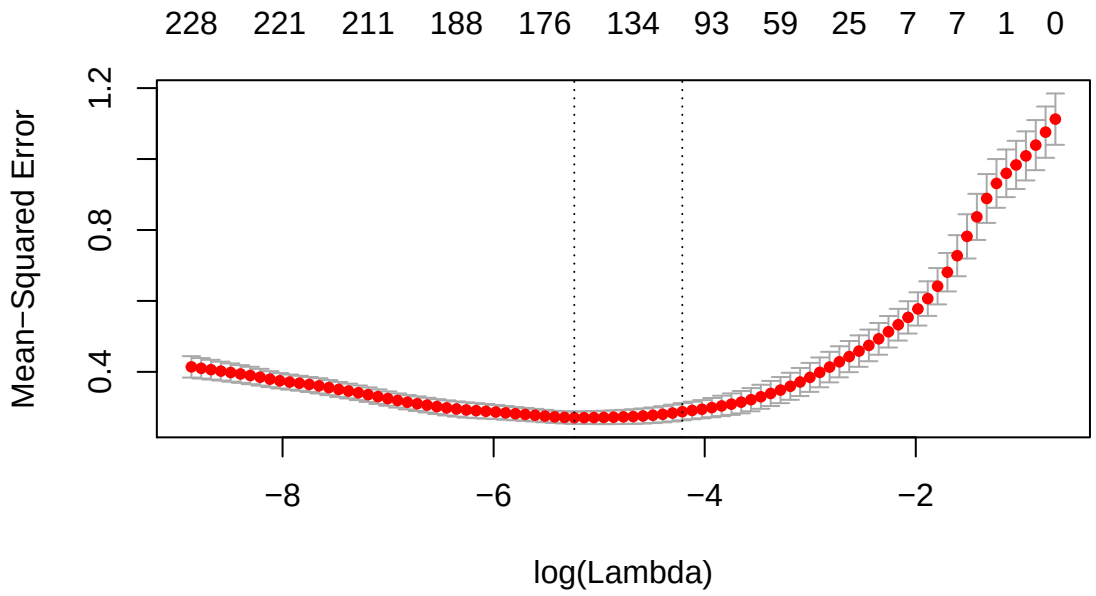
```
codessa <- codessa[, which(cv.coef > 0)]

x = model.matrix(response~.,data=isida)[,-1]
y = isida$response
grid = 10^seq(10, -2, length = 100)

lasso.mod <- glmnet(x, y, alpha = 1, lambda = grid)
plot(lasso.mod)
```




```
set.seed(835)
cv.out = cv.glmnet(x, y, alpha = 1, type.measure="mse", nfolds = 5)
bestlam =cv.out$lambda.1se
bestlam
```

```
## [1] 0.0148149
```

```
plot(cv.out)
```




```
cv.coef <- coef(cv.out, s = bestlam)
cv.coef <- as.matrix(Matrix(cv.coef, sparse = F))
cv.coef[cv.coef > 0, ]
```

```
##       frag3       frag7      frag10      frag11      frag13      frag18
## 0.113002201 0.037558153 0.320895738 0.061492508 0.077602993 0.047942899
##      frag24      frag33      frag35      frag39      frag45      frag48
## 0.054421292 0.068717447 0.293645930 0.099037789 0.067512726 0.199758853
##      frag50      frag59      frag62      frag69      frag71      frag78
## 0.138654786 0.533082000 0.235100718 0.498067501 0.014146988 0.131639174
##      frag84      frag90      frag91      frag94     frag103     frag110
## 0.116293043 0.130392843 0.068634940 0.002858951 0.055325267 0.154553477
##     frag125     frag126     frag128     frag130     frag138     frag148
## 0.128247972 0.020706476 0.040101733 0.299317330 1.860173182 0.088535035
##     frag149     frag150     frag151     frag154     frag155     frag198
## 0.252677425 0.047054685 0.031937126 0.063288837 0.122094551 0.342253734
##     frag205     frag206     frag209     frag210     frag212     frag225
## 0.018139630 0.177692703 0.464293713 0.325044808 0.092488155 0.383511068
##     frag235     frag250     frag255     frag261     frag272     frag276
## 0.107337950 0.233593870 0.283338966 0.221502167 0.045753837 0.191674873
##     frag277     frag278     frag281     frag282     frag288     frag291
## 0.454017379 0.137414889 0.203598381 0.017898378 0.153179461 0.024867158
##     frag295     frag298     frag310     frag312     frag321     frag330
## 0.258906388 0.149781242 0.530363130 0.144227236 0.005066588 0.035825241
##     frag339     frag341     frag342     frag353     frag355     frag357
## 0.220533979 0.019905376 0.055282712 0.055937604 0.543523477 0.082260580
```

```
##     frag358     frag362     frag364     frag365     frag366     frag367
## 0.554852927 1.433936402 0.580215850 0.580181402 1.067840853 1.185603595
##     frag372     frag373     frag384
## 0.140931925 0.523888494 0.360920014
```

```
length(cv.coef[cv.coef > 0, ])
```

```
## [1] 75
```

```
isida <- isida[, c(1, which(cv.coef > 0))]

cl <- makeCluster(3)
registerDoParallel(cl)

knnGrid <-  expand.grid(k = seq(1, 50, by = 2))
fitControl <- trainControl(method = "CV", number = 5)

set.seed(825)

# also tried doing knn with whole isida set, nearly the same
# performance (a little better)

knnFit <- train(response ~ ., data = isida, method = "knn",
                trControl = fitControl, verbose = FALSE,
                tuneGrid = knnGrid)
knnFit
```

```
## k-Nearest Neighbors
##
## 644 samples
##  75 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 516, 514, 516, 515, 515
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared
##    1  0.6808711  0.6202316
##    3  0.7287194  0.5731865
##    5  0.7276660  0.5862222
##    7  0.7316995  0.5956980
##    9  0.7412017  0.5959132
##   11  0.7540661  0.5886098
##   13  0.7634189  0.5817398
##   15  0.7726208  0.5748479
##   17  0.7847354  0.5630718
##   19  0.7907726  0.5581371
##   21  0.8015754  0.5456206
##   23  0.8096574  0.5385403
##   25  0.8159433  0.5300440
##   27  0.8292331  0.5164319
##   29  0.8327421  0.5135717
##   31  0.8352895  0.5110600
##   33  0.8380501  0.5099875
##   35  0.8432086  0.5052678
```

```
##   37  0.8445319  0.5044990
##   39  0.8466899  0.5027914
##   41  0.8464409  0.5013048
##   43  0.8471499  0.5018562
##   45  0.8484795  0.4980820
##   47  0.8516829  0.4916168
##   49  0.8521571  0.4888527
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was k = 1.
```

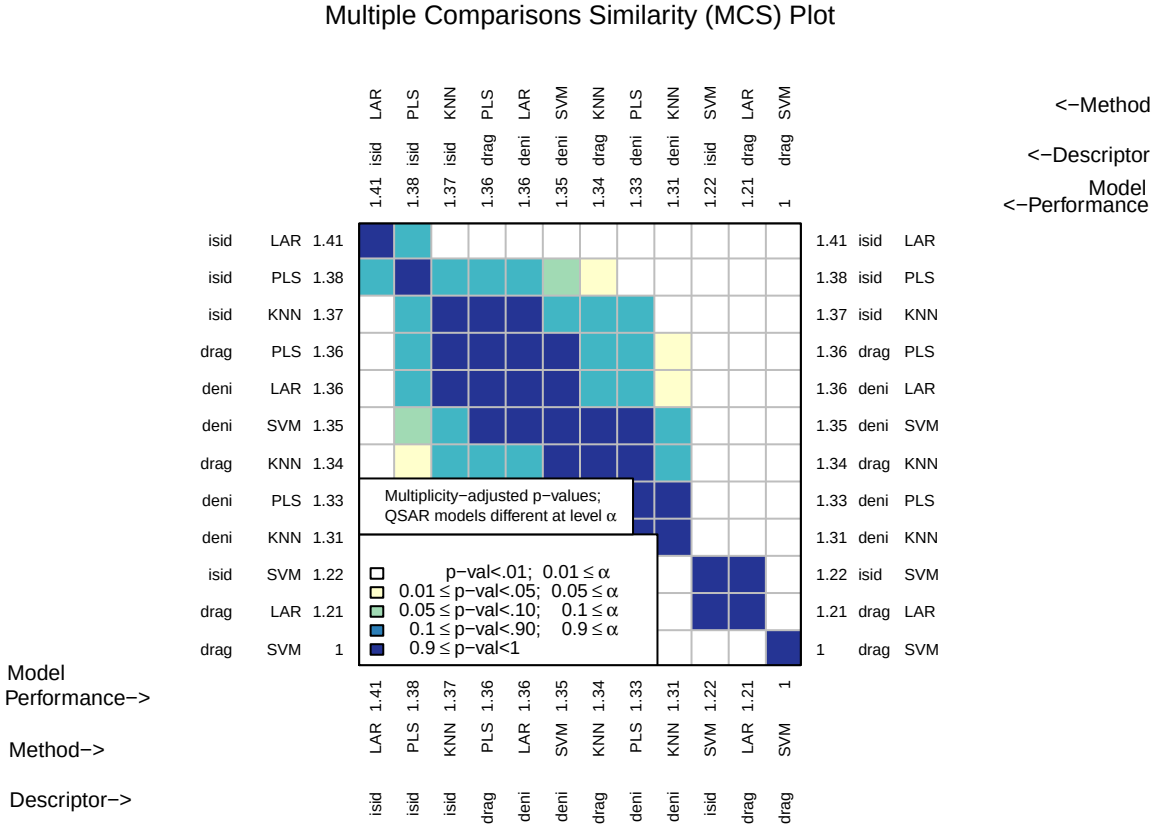
```
# fitControl <- trainControl(method = "CV", number = 5)
# set.seed(825)
#
# svmFit <- train(response ~ ., data = isida, method = "svmRadialSigma",
#                 trControl = fitControl, verbose = FALSE, tuneLength = 10)
# svmFit
#
# set.seed(1)
# tune.out=tune(svm , response ~ ., data=isida, kernel ="radial",
# ranges=list(epsilon=c(0, .34, .5), gamma=c(0.5,1,2,3,4), cost = c(.1, 1, 50, 128)), scale = F)
# summary(tune.out)

des_list[[1]] <- isida[, -1]
des_list[[2]] <- codessa[, -1]

d <- response
for(i in 1:3){
  desc_lengths[i] <- ncol(des_list[[i]])
  d <- cbind(d, des_list[[i]])
}
params <- MakeModelDefaults(nrow(d), ncol(d) - 1, classify = F, 5)

params$SVM$gamma <- .5
params$SVM$epsilon <- 0
params$SVM$cost <- 128

params$LAR$lambda  <- 0

params$KNN$k  <- 1
```

```
cml <- ModelTrain(d, ids = F,
                  xcol.lengths = desc_lengths, des.names = desc_names,
                  nfolds = 5, user.params = params, models = c("PLS", "LAR", "SVM", "KNN"))
```

```
CombineSplits(cml)
```

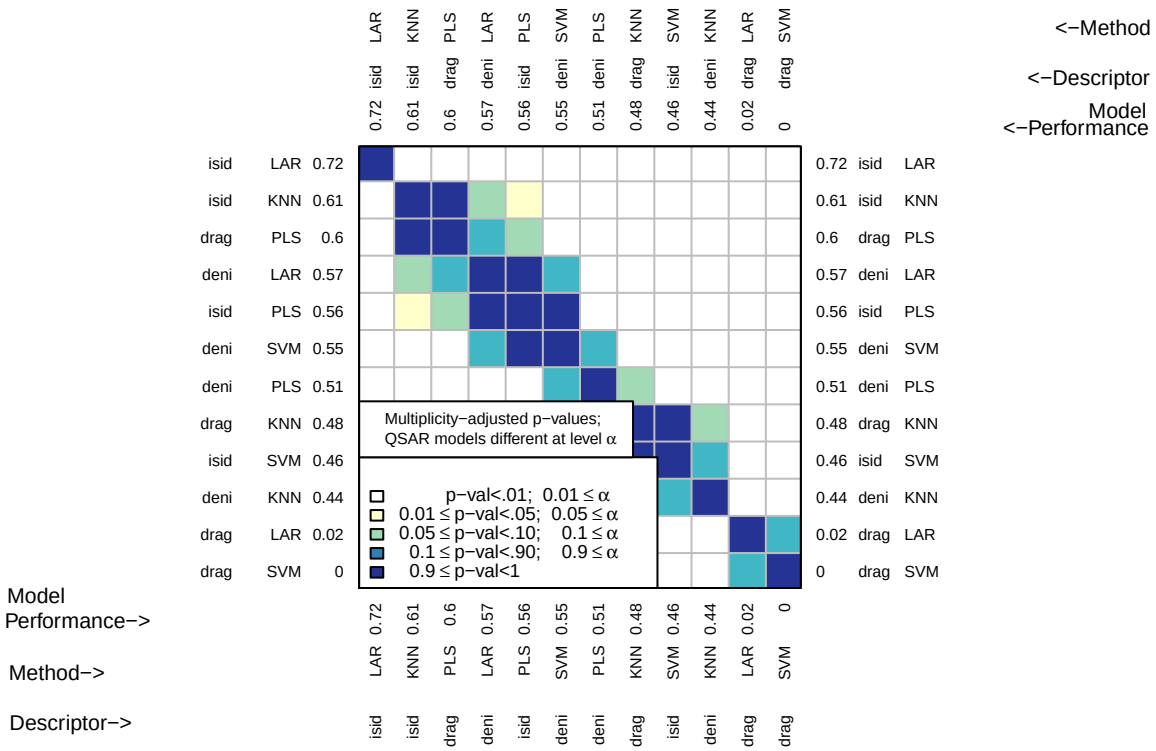
```
##    Analysis of Variance on: 'enhancement'
##  Using factors: Split and Descriptor/Method combination
## Source    DF          SS          MS           F   p-value
## Model     13   4.174e-01   3.211e-02   1.749e+02    <.0001
## Error     22   4.038e-03   1.836e-04
## Total     35   4.214e-01
##       R-Square   Coef Var   Root MSE       Mean
##         0.9904     1.0393     0.0135     1.3036
```

```
## Source       DF          SS          MS           F    p-value
## Split         2   1.02e-04    5.10e-05    2.78e-01     0.7257
## Desc/Meth    11   4.17e-01    3.79e-02    2.07e+02     <.0001
```



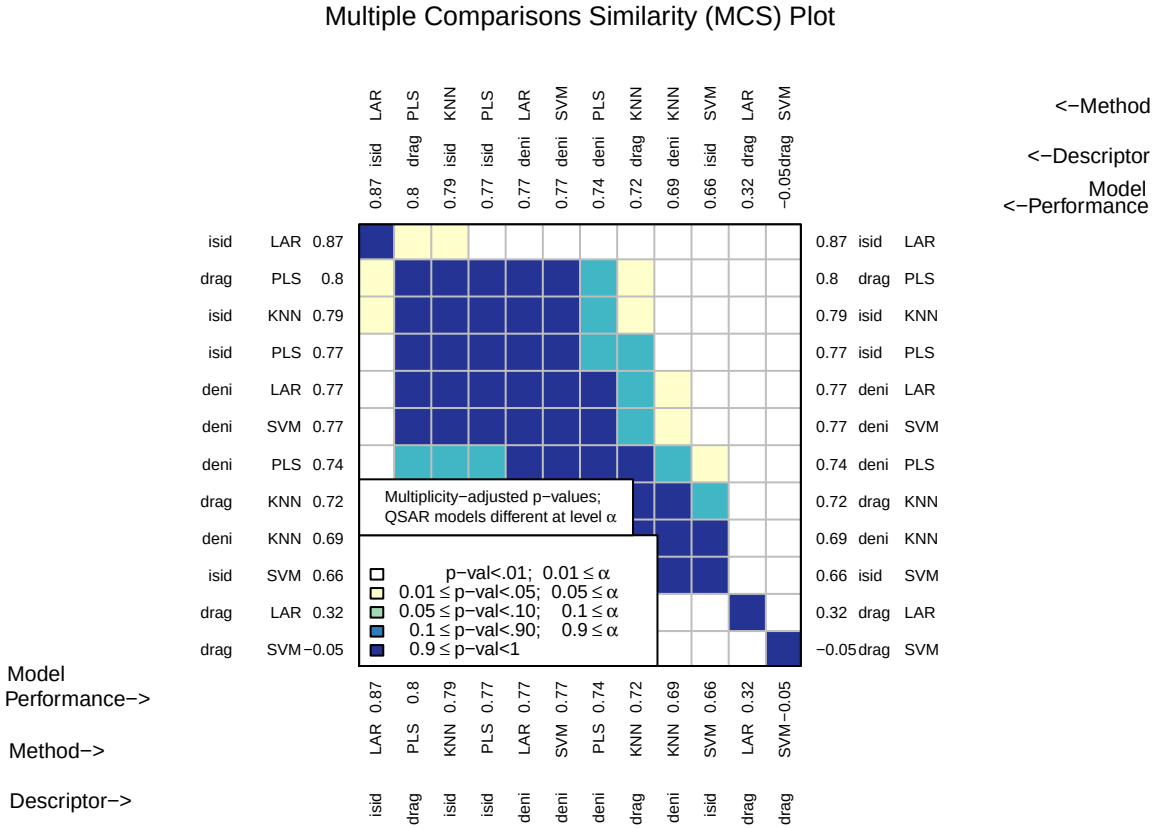

```
CombineSplits(cml, "R2")
```

```
##    Analysis of Variance on: 'R2'
##  Using factors: Split and Descriptor/Method combination
## Source     DF           SS           MS            F    p-value
## Model      13   1.623e+00    1.248e-01    7.409e+02     <.0001
## Error      22   3.707e-03    1.685e-04
## Total      35   1.626e+00
##        R-Square    Coef Var    Root MSE        Mean
##          0.9977      2.8199      0.0130      0.4603
## Source       DF          SS          MS           F    p-value
## Split         2   1.81e-04    9.06e-05    5.38e-01     0.5454
## Desc/Meth    11   1.62e+00    1.48e-01    8.75e+02     <.0001
```



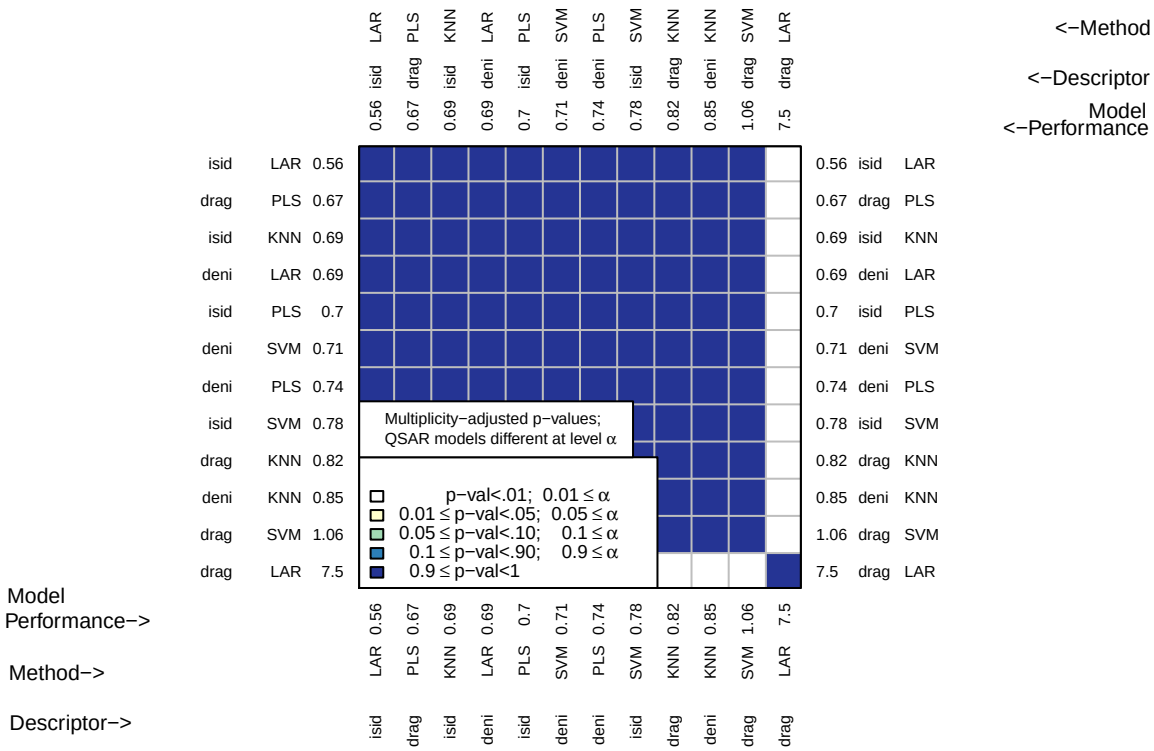

```
CombineSplits(cml, "rho")
```

```
##    Analysis of Variance on: 'rho'
##  Using factors: Split and Descriptor/Method combination
## Source    DF          SS          MS           F   p-value
## Model     13   2.258e+00   1.737e-01   2.986e+02    <.0001
## Error     22   1.280e-02   5.817e-04
## Total     35   2.271e+00
##       R-Square   Coef Var   Root MSE       Mean
##         0.9944     3.6970     0.0241     0.6524
## Source       DF          SS          MS           F   p-value
## Split         2   7.78e-04   3.89e-04   6.68e-01    0.4745
## Desc/Meth    11   2.26e+00   2.05e-01   3.53e+02    <.0001
```




```
CombineSplits(cml, "RMSE")
```

```
##    Analysis of Variance on: 'RMSE'
##  Using factors: Split and Descriptor/Method combination
## Source    DF         SS         MS          F   p-value
## Model     13   126.2614     9.7124    38.2753    <.0001
## Error     22     5.5825     0.2538
## Total     35   131.8439
##       R-Square   Coef Var   Root MSE       Mean
##         0.9577    38.3577     0.5037     1.3133
## Source       DF        SS        MS         F   p-value
## Split         2     0.483     0.241     0.951    0.3547
## Desc/Meth    11   125.779    11.434    45.062    <.0001
```


Multiple Comparisons Similarity (MCS) Plot
<−Method
<−Descriptor
Model
<−Performance
LAR PLS KNN LAR PLS SVM PLS SVM KNN KNN SVM LAR
isid drag isid deni isid deni deni isid drag deni drag drag
0.56 0.67 0.69 0.69 0.7 0.71 0.74 0.78 0.82 0.85 1.06 7.5
isid LAR 0.56
drag PLS 0.67
isid KNN 0.69
deni LAR 0.69
isid PLS 0.7
deni SVM 0.71
deni PLS 0.74
isid SVM 0.78
drag KNN 0.82
deni KNN 0.85
drag SVM 1.06
drag LAR 7.5
0.56 isid LAR
0.67 drag PLS
0.69 isid KNN
0.69 deni LAR
0.7 isid PLS
0.71 deni SVM
0.74 deni PLS
0.78 isid SVM
0.82 drag KNN
0.85 deni KNN
1.06 drag SVM
7.5 drag LAR
Multiplicity−adjusted p−values;
QSAR models different at level α
p−val<.01; 0.01 ≤ α
0.01 ≤ p−val<.05; 0.05 ≤ α
0.05 ≤ p−val<.10; 0.1 ≤ α
0.1 ≤ p−val<.90; 0.9 ≤ α
0.9 ≤ p−val<1
Model
Performance−>
Method−>
Descriptor−>
LAR PLS KNN LAR PLS SVM PLS SVM KNN KNN SVM LAR
isid drag isid deni isid deni deni isid drag deni drag drag
0.56 0.67 0.69 0.69 0.7 0.71 0.74 0.78 0.82 0.85 1.06 7.5