LabMeeting11-9

Jeremy Ash

November 9, 2016

New Model Accuracy Measures

I have implemented new performance measures for categorical data. I have also made changes to labels in the output to report which summary measures are used. I have also report informative errors when a summary measure is not requested properly:

The summary measures for categorical data are:

- Intial enhancement
  - The "at" parameter can now be customized
- Error Rate
  - Can set the probability threshold for considering compounds as active or inactive
- AUC
  - intuition behind AUC: Ideally would want high SEC or SPC no matter the threshold. May be underestimating or overestimating probabilities, still want to have high SEC or SPC even if threshold should be smaller, larger.
- ρ
  - I know this is less common
- Specificity
- Sensitivity

Additional ones that we might consider:

- Area under the sensitivity or specificity curve
- Kappa
- Area under the accumulation curve
- Matthews correlation coefficient
- Balanced accuracy

The summary measures for continuous data are:

- Intial enhancement
  - The "at" parameter can now be customized
- RMSE
- R^2
- ρ

Additional ones that we might consider:

- AAE

Snippit of code showing how I am computing some of the performance measures:

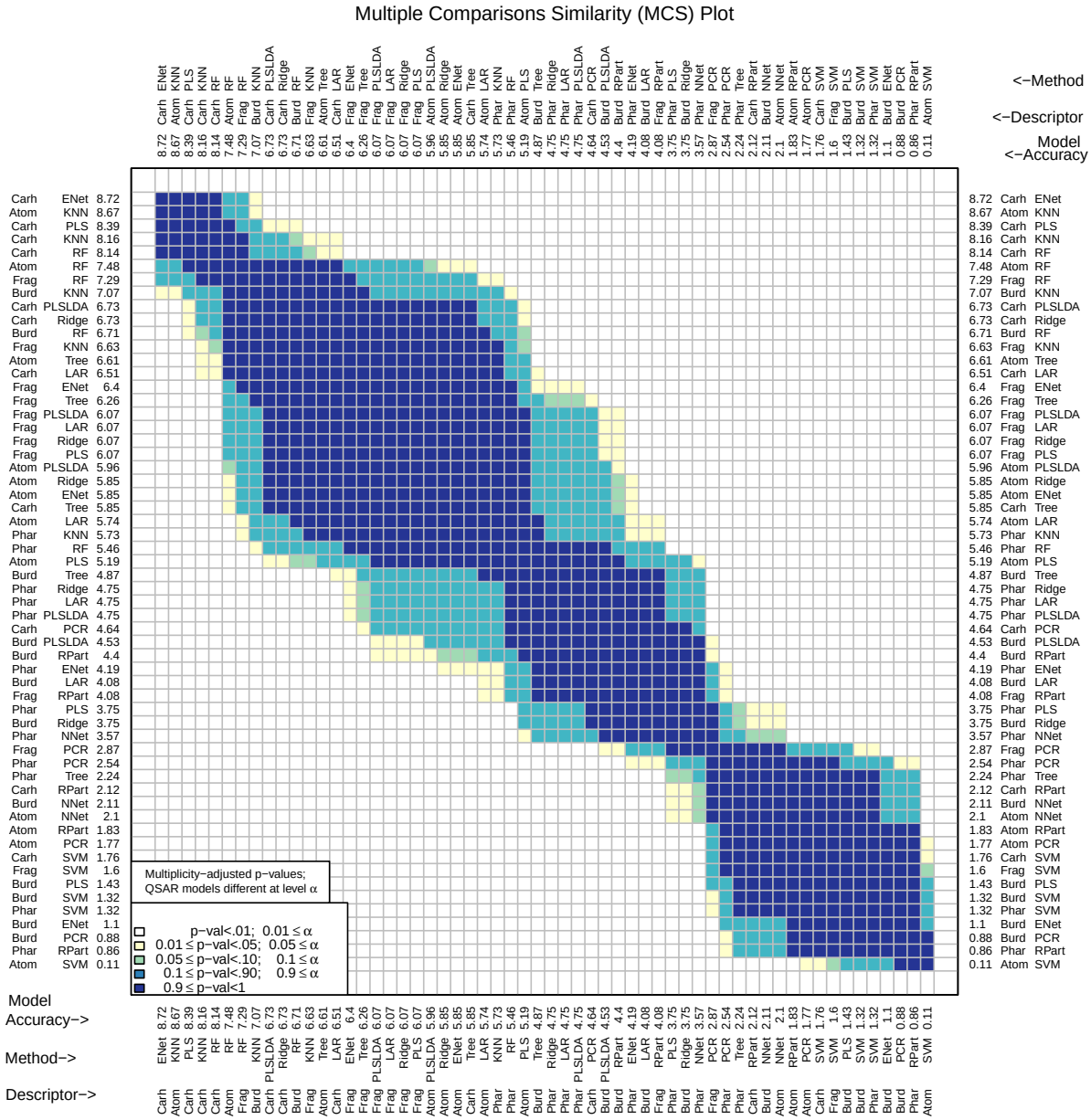
```
          yhat <- prob[[i]][, j] > thresh
          if (metric == "enhancement") {
            model.acc <- Enhancement(prob[[i]][, j], y, at)
          } else if (metric == "auc") {
            model.acc <- as.numeric(auc(y, prob[[i]][, j]))
          } else if (metric == "error rate") {
            model.acc <- mean(y != yhat)
          } else if (metric == "specificity") {
            idx <- y == 0
            model.acc <- mean(y[idx] == yhat[idx])
          } else if (metric == "sensitivity") {
            idx <- y == 1
            model.acc <- mean(y[idx] == yhat[idx])
          } else if (metric == "rho") {
            model.acc <- cor(y, prob[[i]][, j], method = "spearman")
          } else {
            stop("y is binary. 'metric' should be a model accuracy measure
              implemented for binary response in ChemModLab")
          }
```

Loading Previous Run

The run that I was given data for intially was recreated using the new code. I am reloading the results and analyzing again. 3 splits, all descriptor sets.

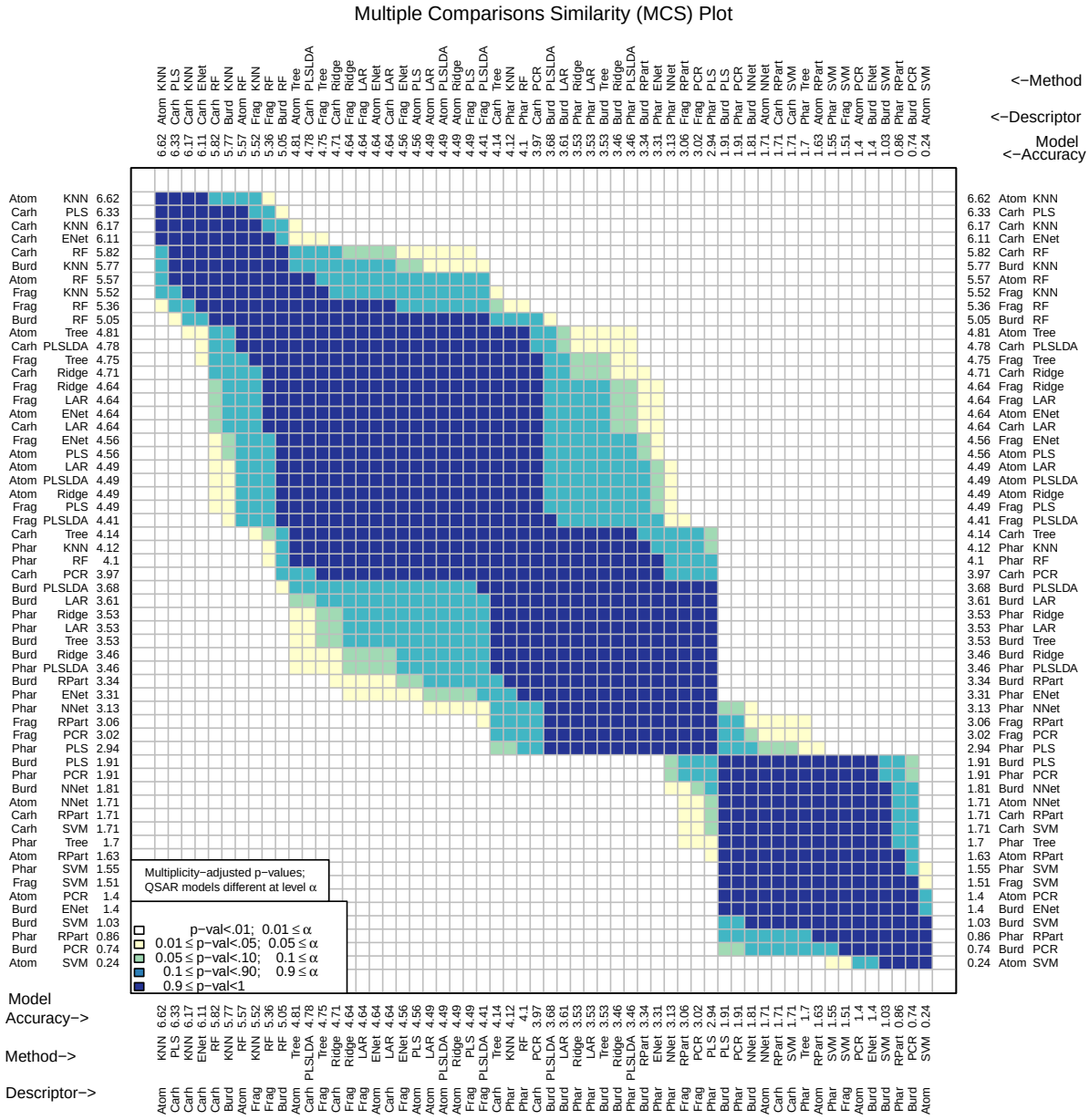
```
CombineSplits(bb, metric = "enhancement", at = 200)
```

```
##    Analysis of Variance on: 'enhancement'
##  Using factors: Split and Descriptor/Method combination
## Source     DF          SS          MS           F    p-value
## Model      59    915.7551     15.5213     77.9191     <.0001
## Error     114     22.7085      0.1992
## Total     173    938.4636
##        R-Square    Coef Var    Root MSE        Mean
##          0.9758      9.5701      0.4463      4.6636
## Source        DF          SS          MS           F    p-value
## Split          2       3.959       1.980       9.938      1e-04
## Desc/Meth     57     911.796      15.996      80.304     <.0001
```



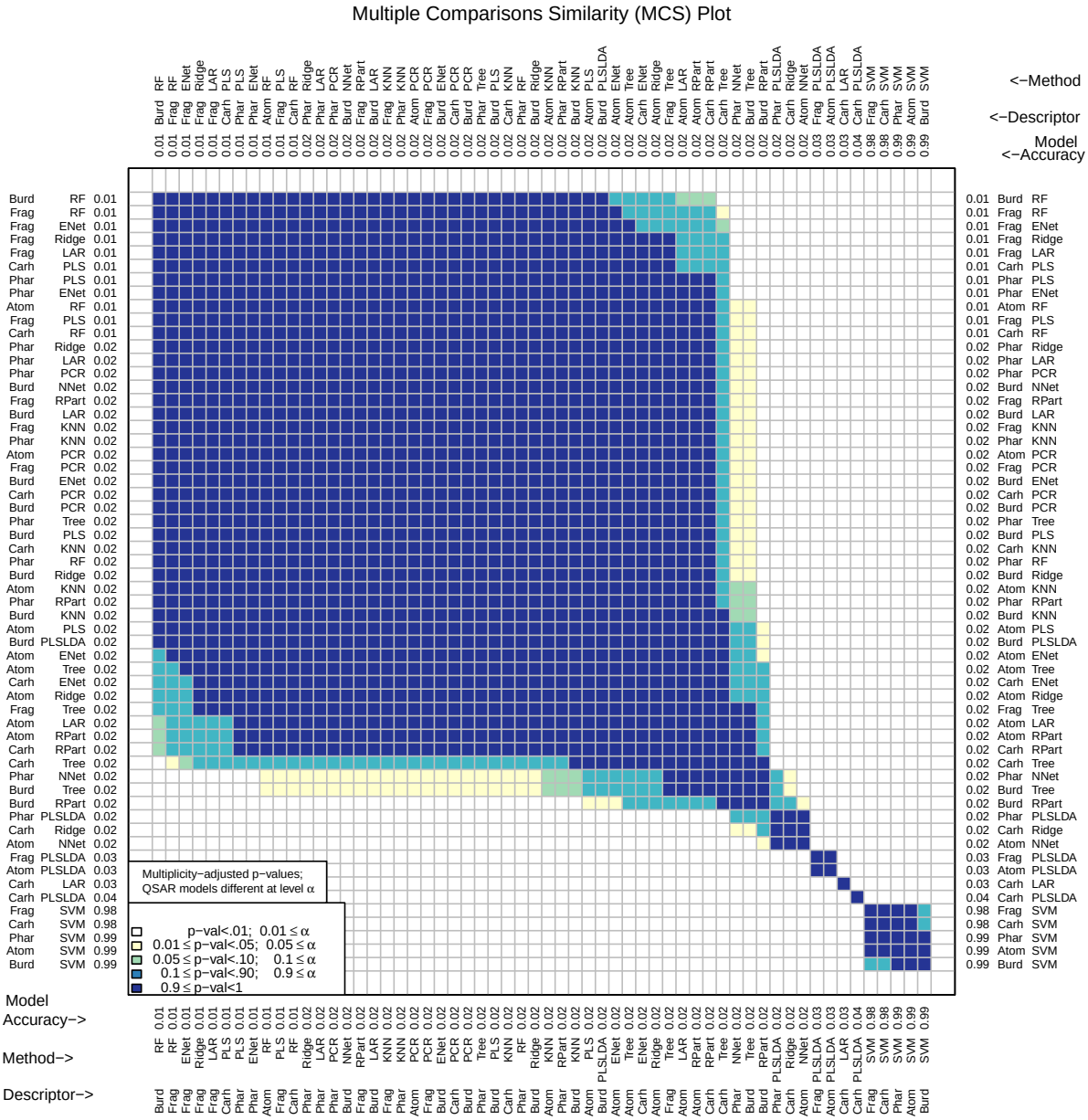
```
CombineSplits(bb, metric = "enhancement", at = 300)
```

```
##    Analysis of Variance on: 'enhancement'
##  Using factors: Split and Descriptor/Method combination
## Source     DF          SS          MS           F    p-value
## Model      59    448.3911      7.5998     57.3784     <.0001
## Error     114     15.0995      0.1325
## Total     173    463.4906
##        R-Square    Coef Var    Root MSE        Mean
##          0.9674     10.0431      0.3639      3.6238
## Source        DF          SS          MS           F    p-value
## Split          2       2.341       1.170       8.836      3e-04
## Desc/Meth     57     446.050       7.825      59.082     <.0001
```



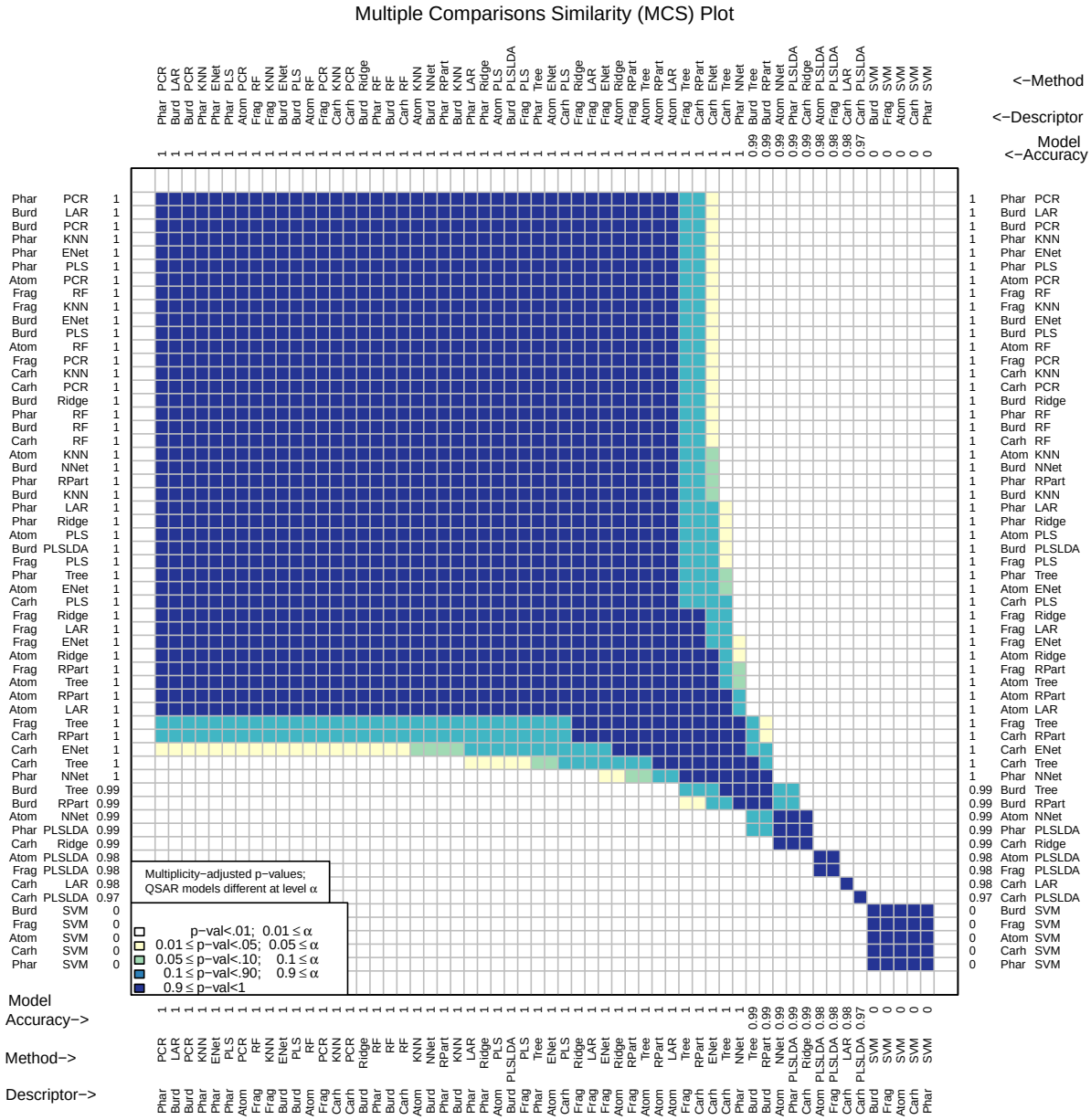
```
CombineSplits(bb, metric = "error rate")
```

```
##    Analysis of Variance on: 'error rate'
##  Using factors: Split and Descriptor/Method combination
## Source     DF          SS          MS           F   p-value
## Model      59   1.286e+01   2.180e-01   2.204e+05    <.0001
## Error     114   1.128e-04   9.891e-07
## Total     173   1.286e+01
##       R-Square   Coef Var   Root MSE        Mean
##       0.999991   0.988000   0.000995    0.100662
## Source       DF          SS          MS           F   p-value
## Split         2   7.77e-07   3.88e-07   3.93e-01    0.6678
## Desc/Meth    57   1.29e+01   2.26e-01   2.28e+05    <.0001
```



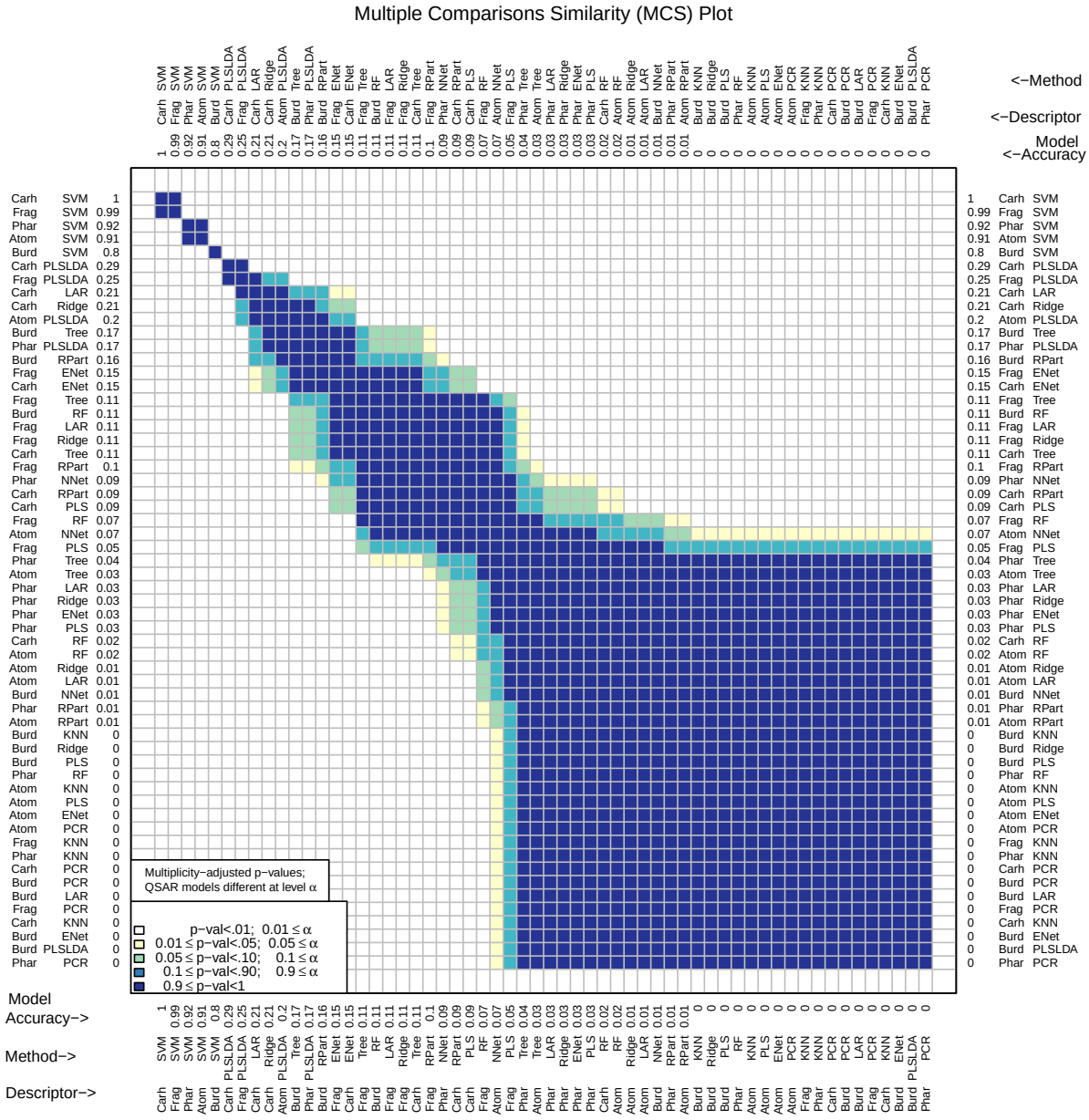
```
CombineSplits(bb, metric = "specificity")
```

```
##    Analysis of Variance on: 'specificity'
##  Using factors: Split and Descriptor/Method combination
## Source     DF           SS           MS            F    p-value
## Model      59    1.362e+01    2.309e-01    2.333e+05     <.0001
## Error     114    1.128e-04    9.898e-07
## Total     173    1.362e+01
##        R-Square    Coef Var    Root MSE        Mean
##        0.999992    0.109206    0.000995    0.911026
## Source       DF          SS          MS           F    p-value
## Split         2    5.07e-07    2.53e-07    2.56e-01     0.7681
## Desc/Meth    57    1.36e+01    2.39e-01    2.41e+05     <.0001
```



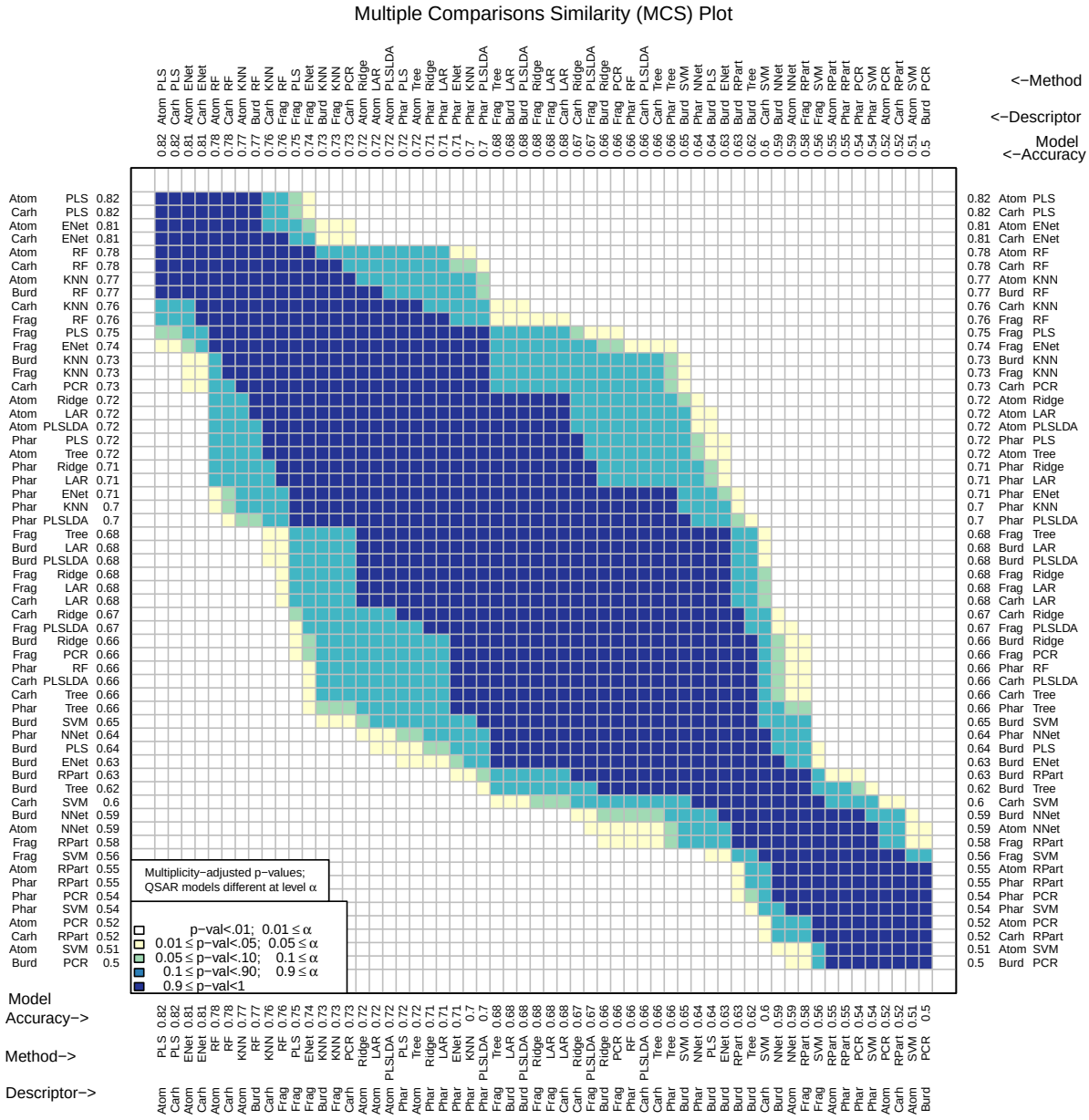
```
CombineSplits(bb, metric = "sensitivity")
```

```
##    Analysis of Variance on: 'sensitivity'
##  Using factors: Split and Descriptor/Method combination
## Source     DF           SS           MS            F    p-value
## Model      59    1.117e+01    1.894e-01    5.600e+02     <.0001
## Error     114    3.855e-02    3.382e-04
## Total     173    1.121e+01
##        R-Square    Coef Var    Root MSE        Mean
##          0.9966     13.4219      0.0184      0.1370
## Source        DF           SS           MS            F    p-value
## Split          2    1.05e-02    5.26e-03    1.55e+01     <.0001
## Desc/Meth     57    1.12e+01    1.96e-01    5.79e+02     <.0001
```



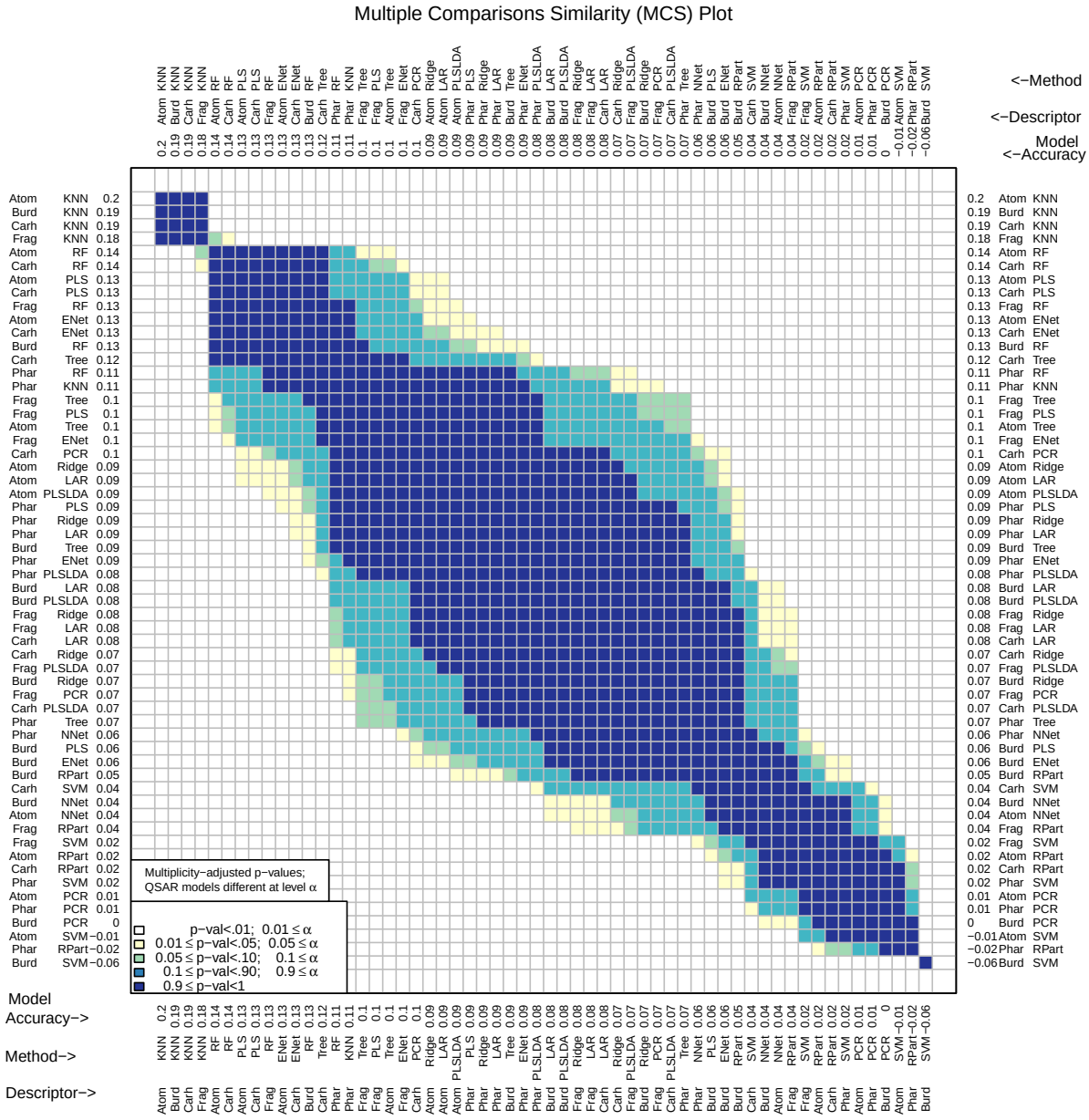
```
CombineSplits(bb, metric = "auc")
```

```
##    Analysis of Variance on: 'auc'
##  Using factors: Split and Descriptor/Method combination
## Source     DF           SS           MS            F    p-value
## Model      59    1.192e+00    2.020e-02    4.117e+01     <.0001
## Error     114    5.593e-02    4.906e-04
## Total     173    1.248e+00
##        R-Square    Coef Var    Root MSE        Mean
##          0.9552      3.2875      0.0221      0.6738
## Source        DF          SS          MS           F    p-value
## Split          2     0.00642     0.00321      6.5476     0.0022
## Desc/Meth     57     1.18525     0.02079     42.3826     <.0001
```



```
CombineSplits(bb, metric = "rho")
```

```
##    Analysis of Variance on: 'rho'
##  Using factors: Split and Descriptor/Method combination
## Source     DF          SS          MS           F   p-value
## Model      59   4.785e-01   8.111e-03   6.985e+01    <.0001
## Error     114   1.324e-02   1.161e-04
## Total     173   4.918e-01
##       R-Square   Coef Var   Root MSE        Mean
##         0.9731    13.3331     0.0108      0.0808
## Source       DF          SS          MS           F   p-value
## Split         2   1.14e-03   5.71e-04   4.92e+00    0.0091
## Desc/Meth    57   4.77e-01   8.38e-03   7.21e+01    <.0001
```



Many models have comparable specificity, but very few have high sensitivity. There is a slightly larger subset of models that have high auc and enhancement. The model with the best initial enhancement changes when I consider a different number of top ranked compounds.

Customizable Tuning Parameters

Now users can set tuning parameters manually. I have implemented a "MakeModelDefaults" function that allows the user to create a list of the default parameters, which they can then modify. I have also created a "PrintModelDefaults" function which will allow the user to see the model defaults in a prettier format. I have also implemented error handeling that throws error if parameters are set incorrectly (eg. multiple values for one parameter) and warnings if parameter values are not used.

Some issues I have encountered:

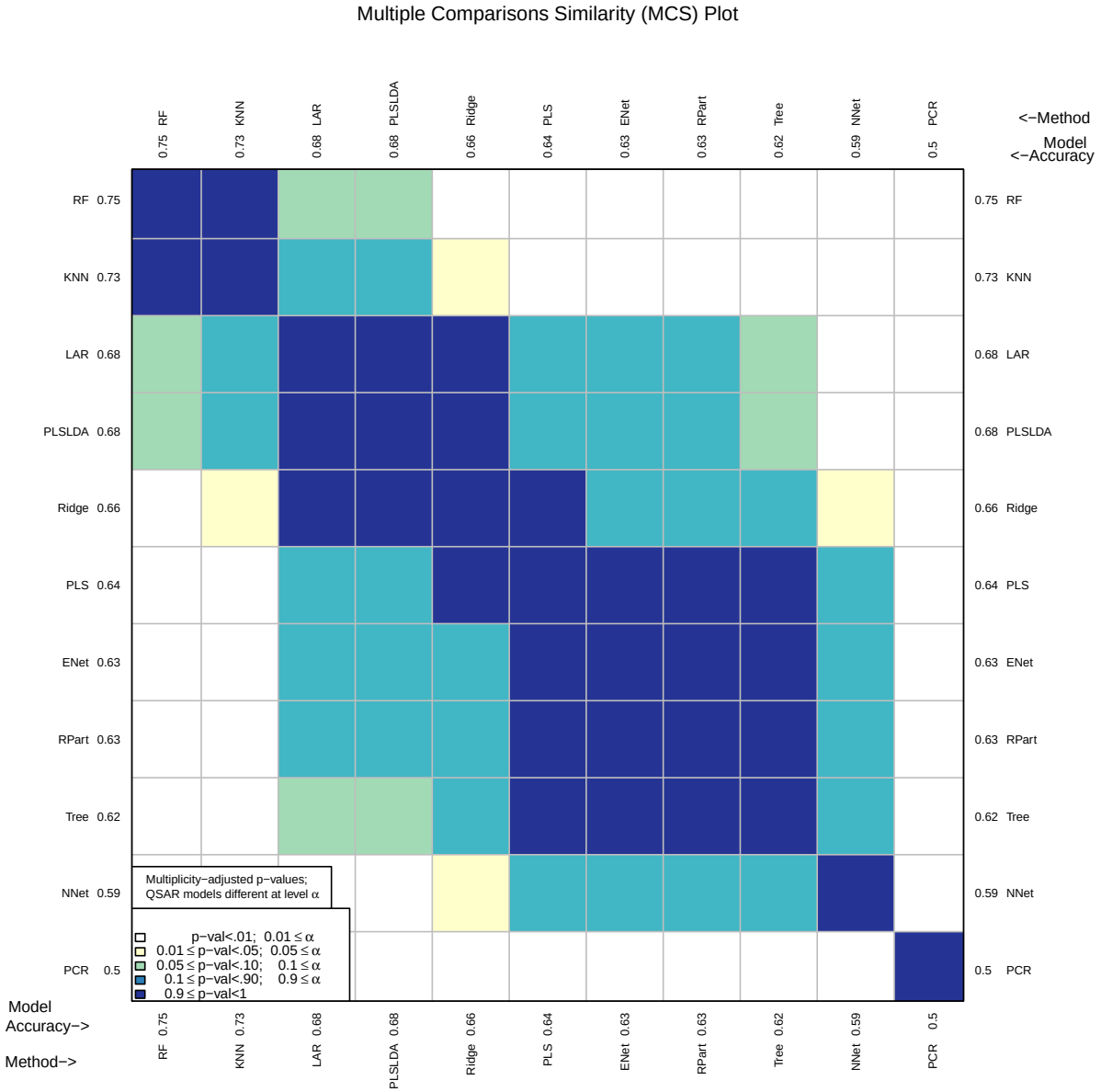
- How do I set the number of components to use for the homegrown PCR code?
- Nnet: weight decay tuned
  - regularizes the cost function, penalizes large weights and effectively limits the freedom of the model
  - For example, a simple way to regularize the cost function would be to add a zero mean gaussian prior to the weights: $\hat{E(w)} = E(w) + \frac{\lambda}{2}w^2$
  - not used for regression?
- Lar: tunning on max number of steps?
- KNN: no regression
  - only for KNNflex
- KNN: no regression
- rpart: tune cp
  - complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. For instance, with anova splitting, this means that the overall R-squared must increase by cp at each step. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially,the user informs the program that any split which does not improve the fit by cp will likely be pruned off by cross-validation, and that hence the program need not pursue it.
  - can tune the size of the tree this way

I show how the user can first tune their parameters using caret and then set those model parameters in ChemModLab. I also demonstrate how parallel processing is used in caret.

```
source("background_newparms.R")

yfilein <- read.csv("AID_364.csv")
xfilein1 <- read.csv("BurdenNumbers.csv")
data <- cbind(yfilein, xfilein1[,-1])
head(data[, 1:6])

bb <- ModelTrain(data, idcol=1,
                 models = c("NNet","PCR","ENet","PLS","Ridge",
                            "LARs","PLSLDA","RPart","Tree","KNN","Forest"),
                 nsplits = 3, nfolds=10)
```

```
CombineSplits(bb, metric = "auc")
```

```
##    Analysis of Variance on: 'auc'
##             Using factors: Split and Method
## Source    DF          SS          MS           F   p-value
## Model     12   1.349e-01   1.124e-02   2.207e+01    <.0001
## Error     20   1.019e-02   5.093e-04
## Total     32   1.451e-01
##       R-Square   Coef Var   Root MSE       Mean
##         0.9298     3.4876     0.0226     0.6471
## Source       DF          SS          MS           F   p-value
## Split         2   7.74e-04   3.87e-04   7.60e-01    0.4279
## Method       10   1.34e-01   1.34e-02   2.63e+01    <.0001
```



```
print(MakeModelDefaults)
```

```
## function (n, p, classify, nfolds)
## {
##     params <- list(NNet = data.frame(size = 2, decay = 0), PCR = NULL,
##         ENet = data.frame(lambda = 1), PLS = data.frame(ncomp = min(floor(n/nfolds),
##             p, 100)), Ridge = data.frame(lambda = 0.1), LARs = NULL,
##         PLSLDA = data.frame(ncomp = min(floor(n/nfolds), p, 100)),
##         RPart = data.frame(cp = 0.01), Tree = NULL, SVM = data.frame(gamma = 1,
##             cost = 1, epsilon = 0.01), KNN = data.frame(k = 10),
##         Forest = data.frame(mtry = if (classify) max(floor(p/3),
##             1) else floor(sqrt(p))))
##     params
## }
```

```
user.params <- MakeModelDefaults(n = nrow(data[, -1]), p = ncol(data[, -1]), classify = T, nfolds = 10)

PrintModelDefaults(n = nrow(data[, -1]), p = ncol(data[, -1]), classify = T, nfolds = 10)
```

```
## $NNet
##   size decay
## 1    2     0
##
## $PCR
## NULL
##
## $ENet
##   lambda
## 1      1
##
## $PLS
##   ncomp
## 1    25
##
## $Ridge
##   lambda
## 1    0.1
##
## $LARs
## NULL
##
## $PLSLDA
##   ncomp
## 1    25
##
## $RPart
##     cp
## 1 0.01
##
## $Tree
## NULL
##
```

```
## $SVM
##   gamma cost epsilon
## 1     1    1    0.01
##
## $KNN
##    k
## 1 10
##
## $Forest
##   mtry
## 1    8
```

```
# tuning model parameters in caret

cl <- makeCluster(4)
registerDoParallel(cl)

data_caret <- data[, -1]

# need to make outcome a 2 class variable

data_caret$Outcome <- as.factor(ifelse(data_caret$Outcome == 1, "Active", "Inactive"))

fitControl <- trainControl(method = "CV", 10, classProbs = TRUE,
  summaryFunction = twoClassSummary)

set.seed(823)

rfFit <- train(Outcome ~ ., data = data_caret, method = "rf",
  trControl = fitControl, verbose = FALSE, tuneLength = 10, metric = "ROC")

rfFit
```

```
## Random Forest
##
## 3311 samples
##   24 predictor
##    2 classes: 'Active', 'Inactive'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2979, 2980, 2980, 2980, 2980, 2980, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens  Spec
##    2    0.8188811  0.20  0.9996933
##    4    0.8099307  0.18  0.9996933
##    6    0.8215059  0.20  0.9996933
##    9    0.8196923  0.22  0.9993865
##   11    0.7883483  0.22  0.9996933
##   14    0.8032703  0.24  0.9996933
##   16    0.8168605  0.26  0.9996933
##   19    0.8114583  0.26  0.9993865
##   21    0.8016336  0.24  0.9993865
```

```
##   24    0.8149313  0.24  0.9993865
##
## ROC was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 6.
```

```
user.params$Forest <- data.frame(mtry= 6)

set.seed(823)

rfFit <- train(Outcome ~ ., data = data_caret, method = "nnet",
  trControl = fitControl, verbose = FALSE, tuneLength = 5, metric = "ROC")
```

```
## # weights:  131
## initial  value 2564.888948
## iter  10 value 260.466431
## iter  20 value 250.107394
## iter  30 value 246.856900
## iter  40 value 244.122196
## iter  50 value 243.567615
## iter  60 value 242.285712
## iter  70 value 240.576793
## iter  80 value 234.047703
## iter  90 value 229.035550
## iter 100 value 225.577507
## final  value 225.577507
## stopped after 100 iterations
```

```
rfFit
```

```
## Neural Network
##
## 3311 samples
##   24 predictor
##    2 classes: 'Active', 'Inactive'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2979, 2980, 2980, 2980, 2980, 2980, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens  Spec
##   1     0e+00  0.5224228  0.00  0.9996933
##   1     1e-04  0.5081340  0.00  0.9996933
##   1     1e-03  0.6460402  0.02  0.9993865
##   1     1e-02  0.5368002  0.00  0.9996933
##   1     1e-01  0.5990533  0.00  1.0000000
##   3     0e+00  0.5794364  0.02  0.9990798
##   3     1e-04  0.5583906  0.00  0.9996933
##   3     1e-03  0.6679700  0.02  0.9987730
##   3     1e-02  0.7057618  0.02  0.9981595
##   3     1e-01  0.7418956  0.00  1.0000000
##   5     0e+00  0.6142001  0.04  0.9990798
##   5     1e-04  0.5748619  0.00  0.9996933
```

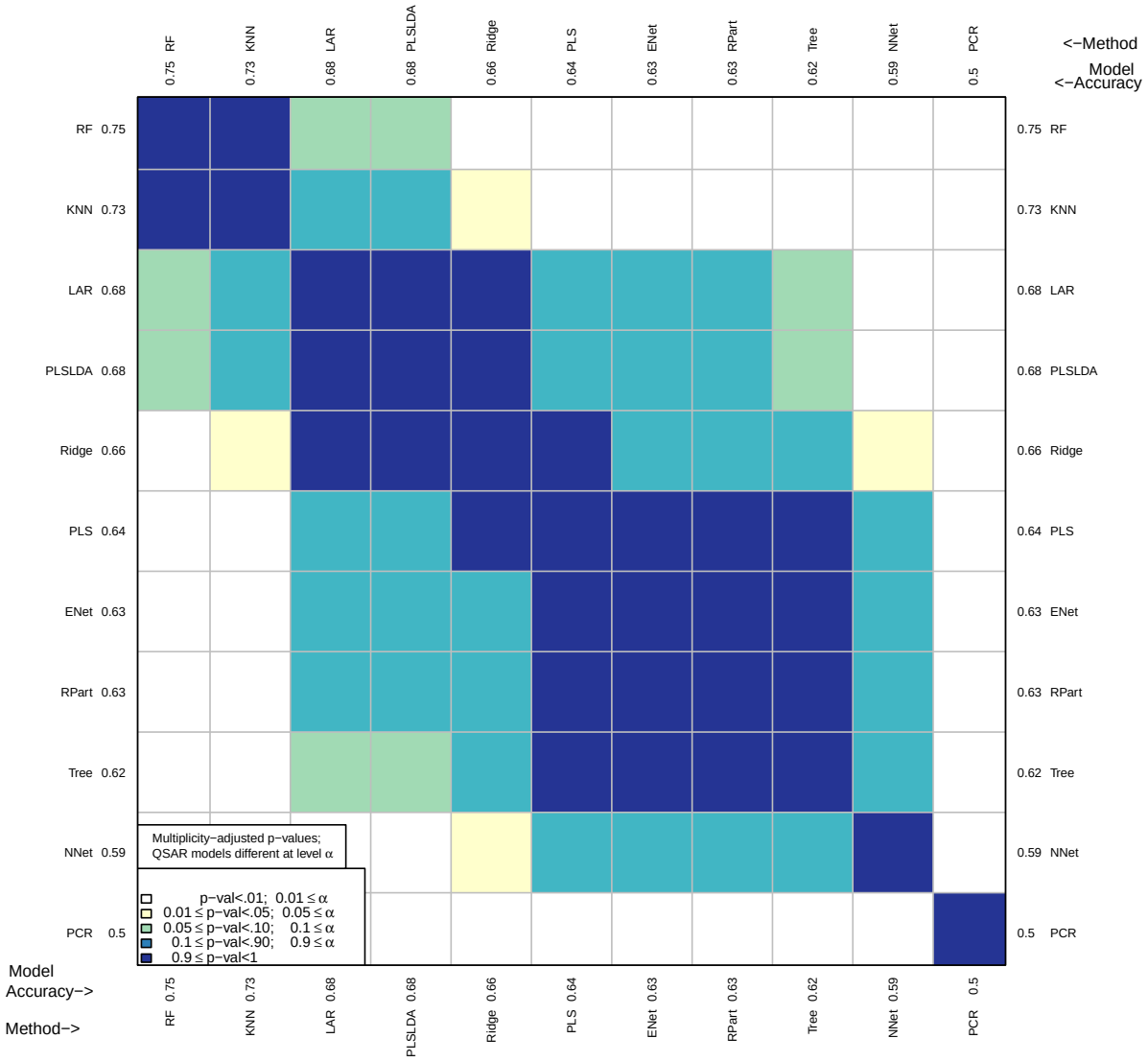
```
##   5     1e-03  0.6781888  0.02  0.9978528
##   5     1e-02  0.6836016  0.02  0.9993865
##   5     1e-01  0.7717970  0.00  1.0000000
##   7     0e+00  0.6714657  0.00  0.9984663
##   7     1e-04  0.6572855  0.02  0.9993865
##   7     1e-03  0.6802716  0.08  0.9947872
##   7     1e-02  0.7349761  0.06  0.9996933
##   7     1e-01  0.7592737  0.00  0.9996933
##   9     0e+00  0.6794133  0.08  0.9981595
##   9     1e-04  0.7160730  0.00  0.9975460
##   9     1e-03  0.6897445  0.10  0.9957074
##   9     1e-02  0.7167775  0.06  0.9987758
##   9     1e-01  0.7701940  0.00  0.9996933
##
## ROC was used to select the optimal model using  the largest value.
## The final values used for the model were size = 5 and decay = 0.1.
```

```
user.params$NNet <- data.frame(size = 5, decay = .1)

# #can tune svm parameters using tune.svm
#
# user.params$SVM <- data.frame(gamma = 2^-5, cost = 10^-2)
```

```
# Do not need to make a list with all the parameters specified, can omit some
# and then the defaults will be used for those parameters:

# user.params <- list(NNet = data.frame(size = 3, decay = 0)

source("background_newparms.R")

bb <- ModelTrain(data, idcol=1,
                 models = c("NNet","PCR","ENet","PLS","Ridge",
                            "LARs","PLSLDA","RPart","Tree","KNN","Forest"),
                 nsplits = 3, nfolds=10, user.params = user.params)
```

```
CombineSplits(bb, metric = "auc")
```

```
##    Analysis of Variance on: 'auc'
##             Using factors: Split and Method
## Source    DF          SS          MS           F   p-value
## Model     12   1.349e-01   1.124e-02   2.207e+01    <.0001
## Error     20   1.019e-02   5.093e-04
## Total     32   1.451e-01
##       R-Square   Coef Var   Root MSE       Mean
##         0.9298     3.4876     0.0226     0.6471
## Source       DF         SS          MS          F   p-value
## Split         2   7.74e-04   3.87e-04   7.60e-01    0.4279
## Method       10   1.34e-01   1.34e-02   2.63e+01    <.0001
```


Multiple Comparisons Similarity (MCS) Plot
<−Method
Model
<−Accuracy
RF 0.75
KNN 0.73
LAR 0.68
PLSLDA 0.68
Ridge 0.66
PLS 0.64
ENet 0.63
RPart 0.63
Tree 0.62
NNet 0.59
PCR 0.5
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
Accuracy−>
Method−>

Tidying Up R Code

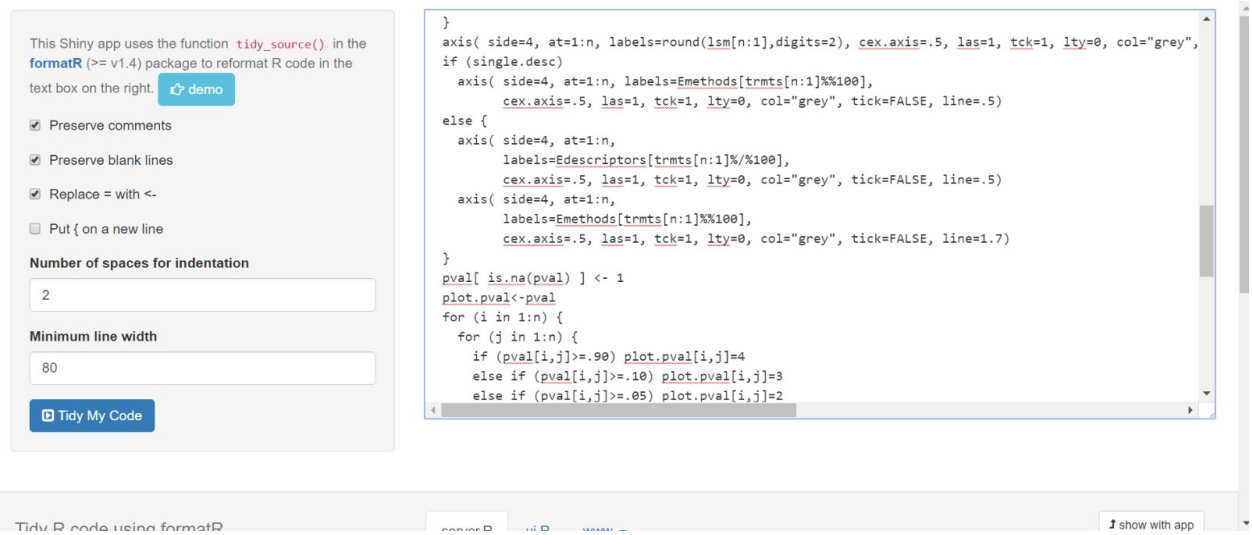


Figure 1: Before tidying

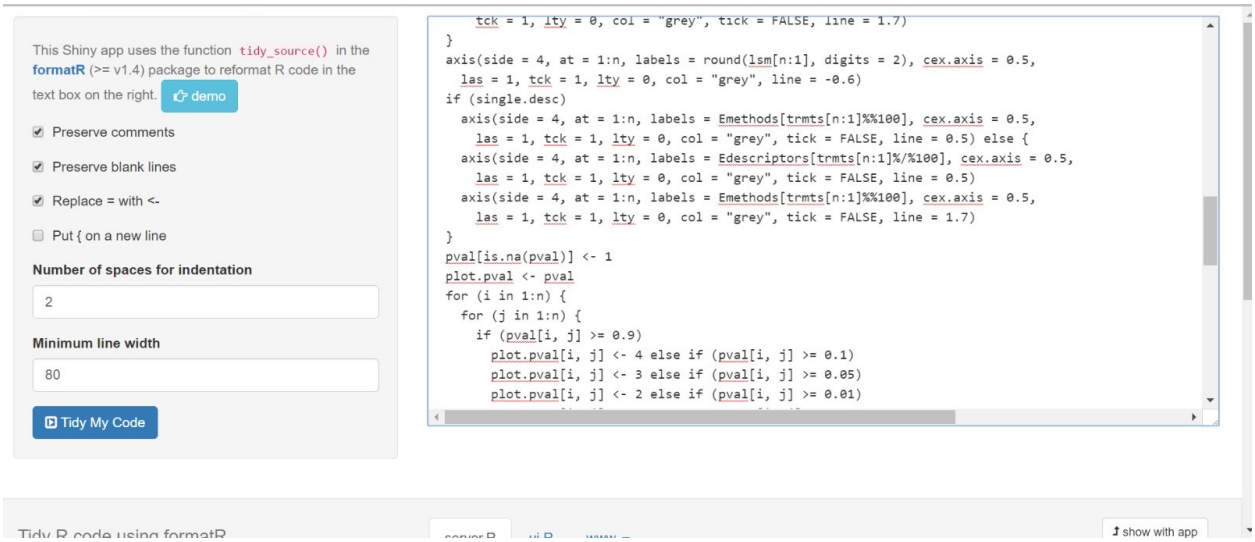


Figure 2: After tidying

Parallel Processing

Taking the same approach as caret, I have taken initial steps in parallel processing. There are still a few challenges that need to be dealt with:

- The output to the console is suppressed because now processes are being run simultaneously. A quick search suggests that writing to console will be possible
- The foreach command returns a list that needs to be reformatted afterwards. There is probably a more elegant way to have foreach return the right list format.

Looping over descriptor sets can be done in exactly the same way. Will do this soon.

Here is a sample of the foreach command:

```
 split.ls <- foreach (seed.idx = 1:nsplits, .packages = (.packages()),
           .export = as.vector(Funcs)) %dopar% {

           }
```

```
source("background_parallel.R")

cl <- makeCluster(1)
registerDoParallel(cl)

system.time(
bb <- ModelTrain(data, idcol=1,
  models = c("NNet","PCR","ENet","PLS","Ridge","LARs","SVM","PLSLDA","RPart","Tree"),
  nfolds=10, nsplits = 3, user.params = user.params)
)
```

```
##    user  system elapsed
##    0.40    0.03  231.29
```

```
CombineSplits(bb)
```

```
##    Analysis of Variance on :'enhancement'
##             Using factors: Split and Method
## Source    DF         SS         MS          F   p-value
## Model     11   41.54860    3.77715   65.48113    <.0001
## Error     18    1.03829    0.05768
## Total     29   42.58689
##       R-Square   Coef Var   Root MSE       Mean
##         0.9756     9.0084     0.2402     2.6661
## Source       DF       SS       MS        F   p-value
## Split         2    0.752    0.376    6.521    0.0073
## Method        9   40.796    4.533   78.583    <.0001
```

```
cl <- makeCluster(3)
registerDoParallel(cl)

system.time(
  bb <- ModelTrain(data, idcol=1,
                   models = c("NNet","PCR","ENet","PLS","Ridge","LARs","SVM","PLSLDA","RPart","Tree"),
                   nfolds=10, nsplits = 3, user.params = user.params)
)
```

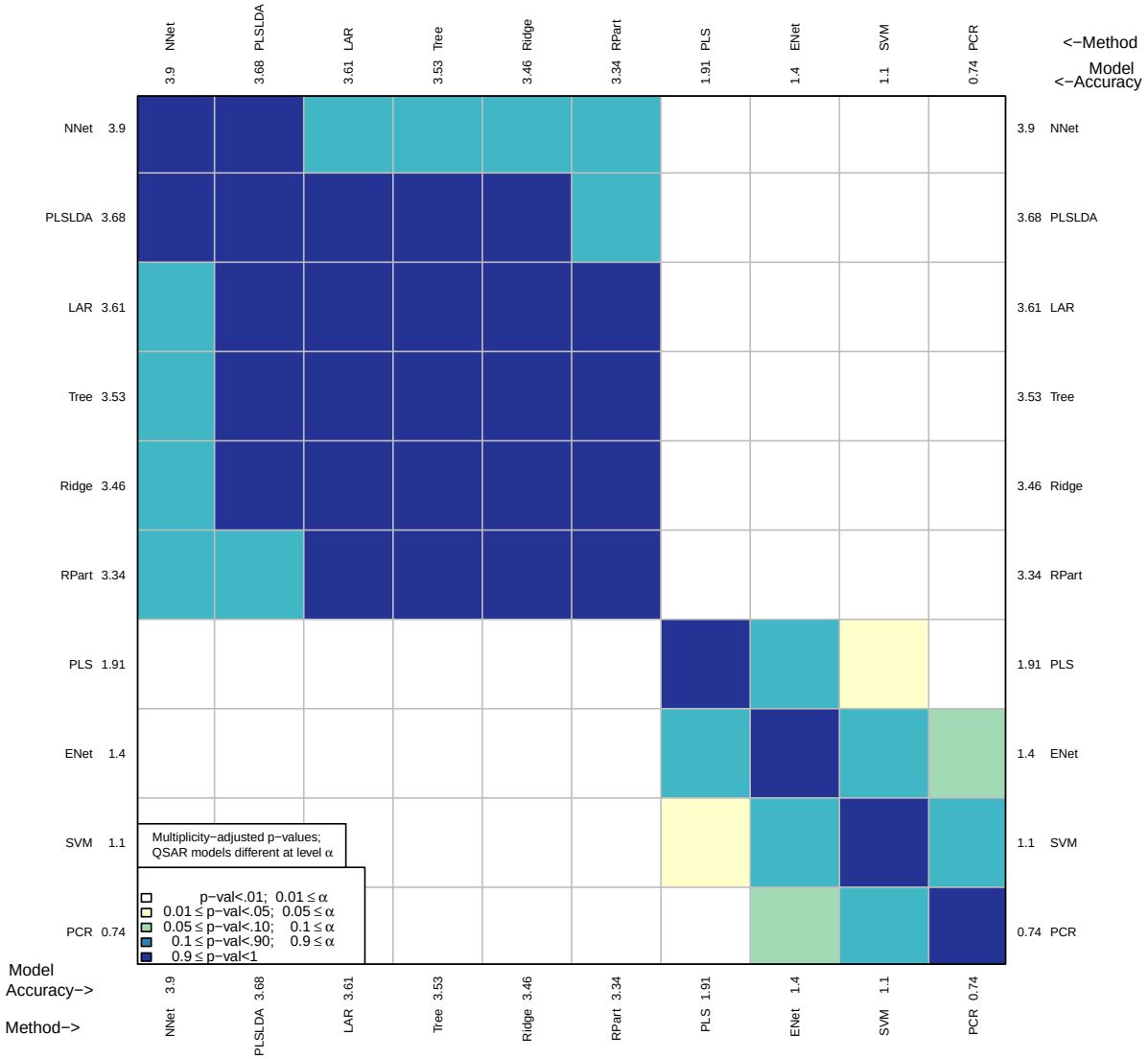
```
## Warning: closing unused connection 5 (<-Vestige-laptop:11056)
```

```
##    user  system elapsed
##    0.60    0.08  141.43
```

```
CombineSplits(bb)
```

```
##    Analysis of Variance on :'enhancement'
##             Using factors: Split and Method
## Source    DF         SS         MS          F   p-value
## Model     11   41.54860    3.77715   65.48113    <.0001
## Error     18    1.03829    0.05768
## Total     29   42.58689
##       R-Square   Coef Var   Root MSE       Mean
##         0.9756     9.0084     0.2402     2.6661
## Source       DF       SS       MS        F   p-value
## Split         2    0.752    0.376    6.521    0.0073
## Method        9   40.796    4.533   78.583    <.0001
```

Multiple Comparisons Similarity (MCS) Plot



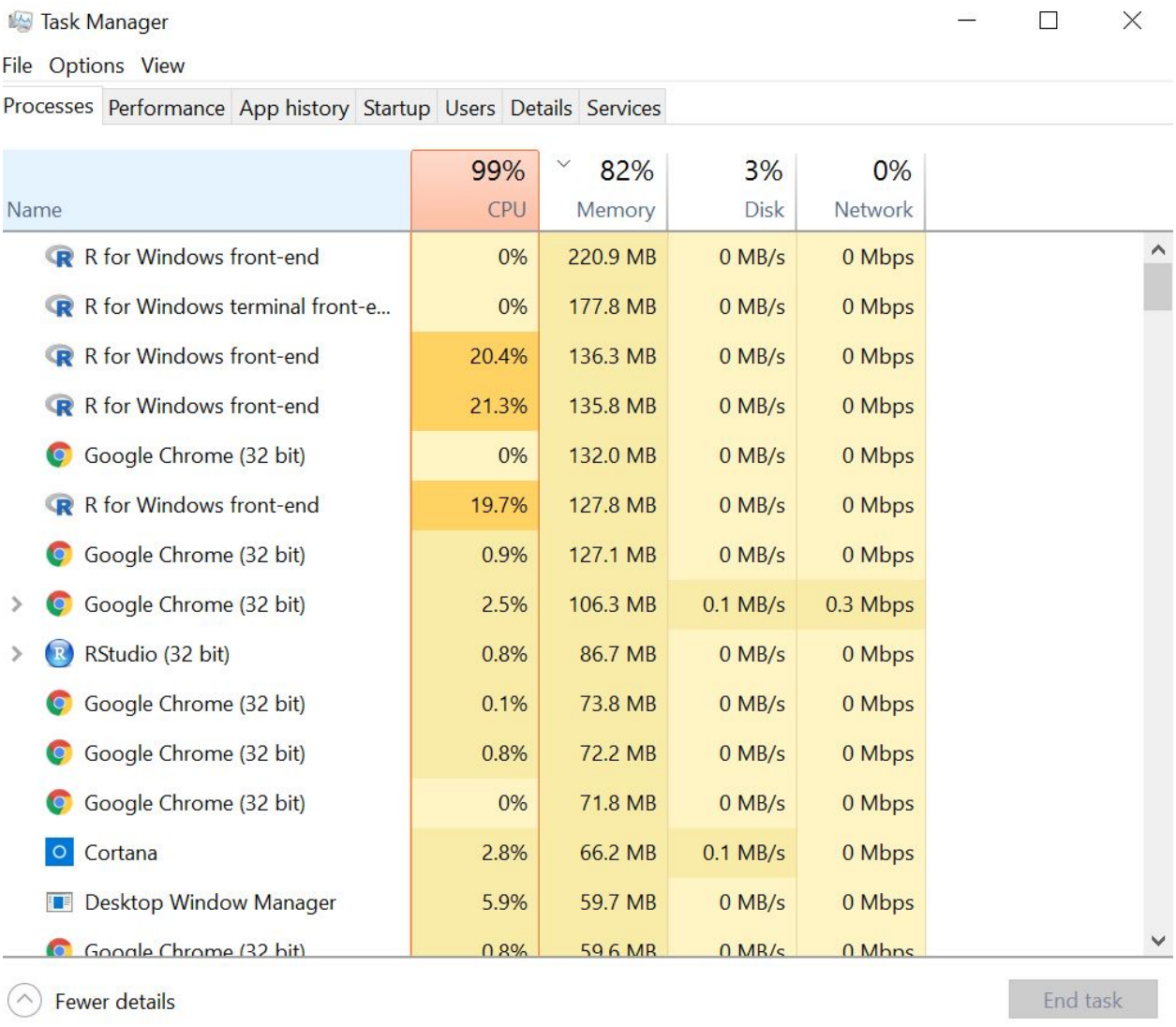


Figure 3: Using 3 processors