LabMeeting11-15

Jeremy Ash

November 15, 2016

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:

- PCR
  - How do I set the number of components to use for the homegrown PCR code?
  - Do we still want to use the homegrown PCR code?
  - Can tune `pls` and `pcr` functions in the `pls` package with caret
  - Can also use the `RMSEP` function in `pls`
- Nnet
  - weight decay tuned
  - regularizes the cost function
    - tunes the penalty for large weights and effectively limits the freedom of the model
  - An easy way to do that is by introducing a zero mean Gaussian prior over the weights, which is equivalent to changing the cost function to: $\hat{E}(w) = E(w) + \frac{\lambda}{2}w^2$. Larger weights are penalized more.
  - No regression
- Lars
  - May be good to implement lasso with both lars and glmnet. According to developer of scikit (and Dr. Bondell):

LARS is faster for small problems, very sparse problems, or very 'wide' problems (much much more features than samples). Its computational cost is limited by the number of features selected, if you don't compute the full regularization path. On the other hand, for big problems, glmnet (coordinate descent optimization) is faster.

- KNN
  - No regression
  - there is for discontinued KNNflex
- rpart
  - tune cp (complexity parameter):

  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.

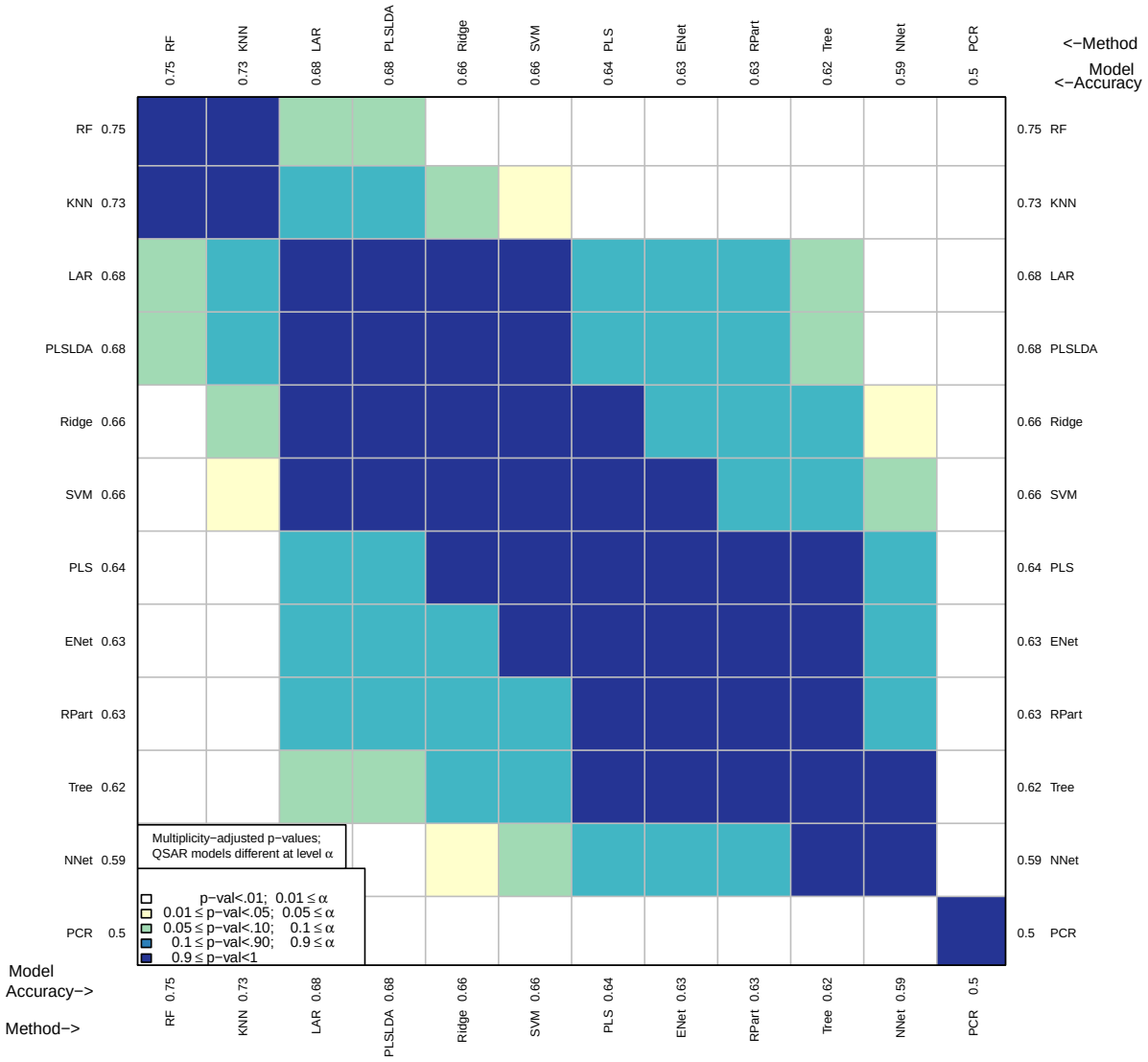
  - can tune the size of the tree this way
- tree
  - need to implement cost complexity pruning
- svm
  - cannot tune the radial basis kernel with caret, only the linear kernel (probably because it is so slow to fit radial basis kernel svm)
  - can tune with the `tune.svm` function in `e1071`
  - caret does support tuning `kvm` a redial basis kernel svm in `kernlab`

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", "SVM",
                            "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     13   1.357e-01   1.044e-02   1.990e+01    <.0001
## Error     22   1.154e-02   5.245e-04
## Total     35   1.472e-01
##       R-Square   Coef Var   Root MSE       Mean
##         0.9216     3.5350     0.0229     0.6479
## Source       DF         SS         MS          F   p-value
## Split         2   1.31e-03   6.54e-04   1.25e+00    0.2649
## Method       11   1.34e-01   1.22e-02   2.33e+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
```

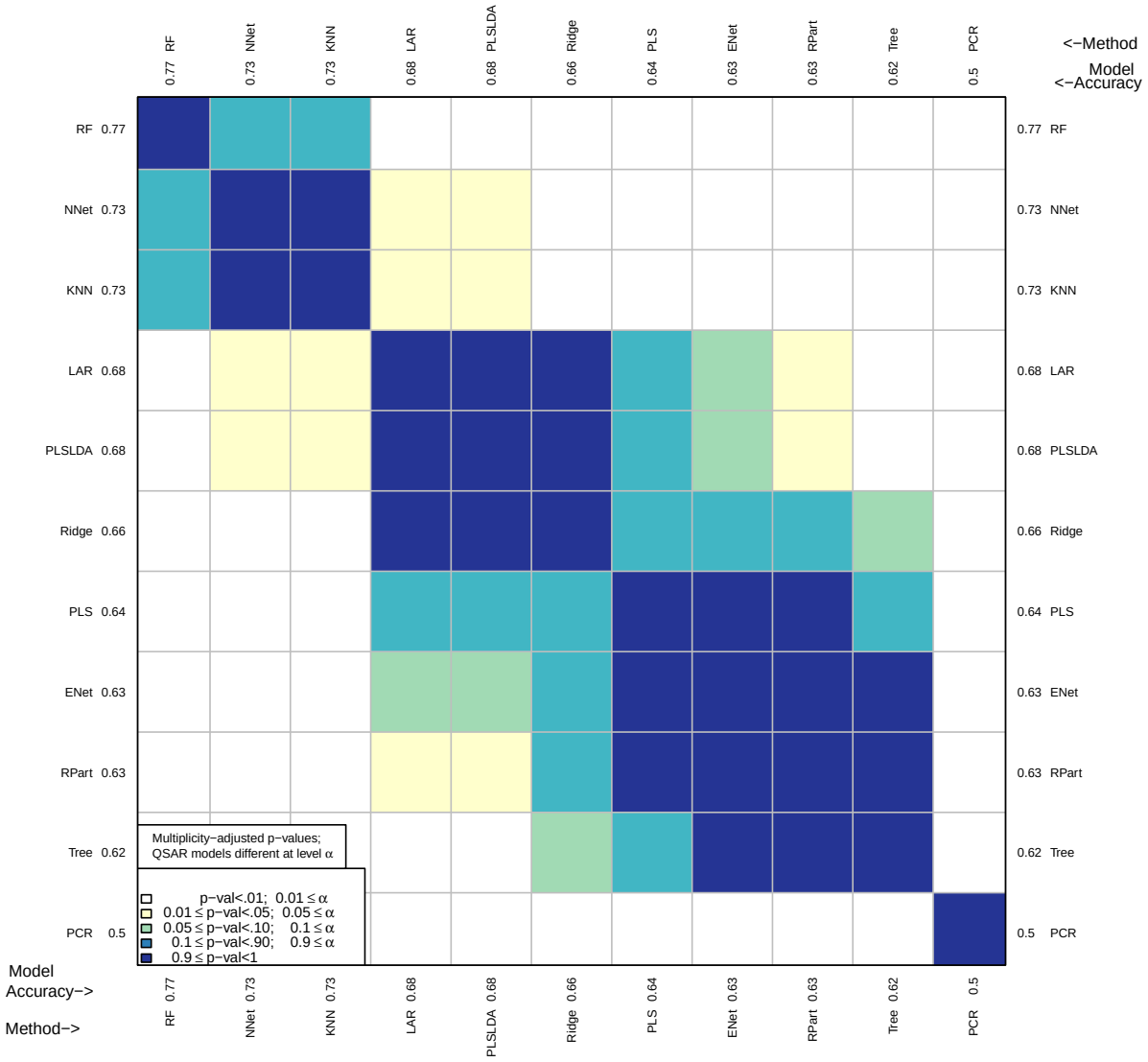
```
## Neural Network
##
## 3311 samples
##   24 predictor
##    2 classes: 'Active', 'Inactive'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2979, 2980, 2980, 2980, 2980, 2980, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        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)

user.params$SVM <- data.frame(gamma = 2^-5, cost = 10^-2)

stopImplicitCluster()
```

```
# 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)

bb <- ModelTrain(data, idcol=1,
                 models = c("NNet","PCR","ENet","PLS","Ridge",

              "SVM","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   1.517e-01   1.264e-02   4.639e+01    <.0001
## Error     20   5.451e-03   2.726e-04
## Total     32   1.572e-01
##       R-Square   Coef Var   Root MSE       Mean
##         0.9653     2.4943     0.0165     0.6619
## Source       DF          SS          MS           F   p-value
## Split         2   8.64e-05   4.32e-05   1.58e-01    0.8293
## Method       10   1.52e-01   1.52e-02   5.56e+01    <.0001
```


Multiple Comparisons Similarity (MCS) Plot
<−Method
Model
<−Accuracy
RF 0.77
NNet 0.73
KNN 0.73
LAR 0.68
PLSLDA 0.68
Ridge 0.66
PLS 0.64
ENet 0.63
RPart 0.63
Tree 0.62
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

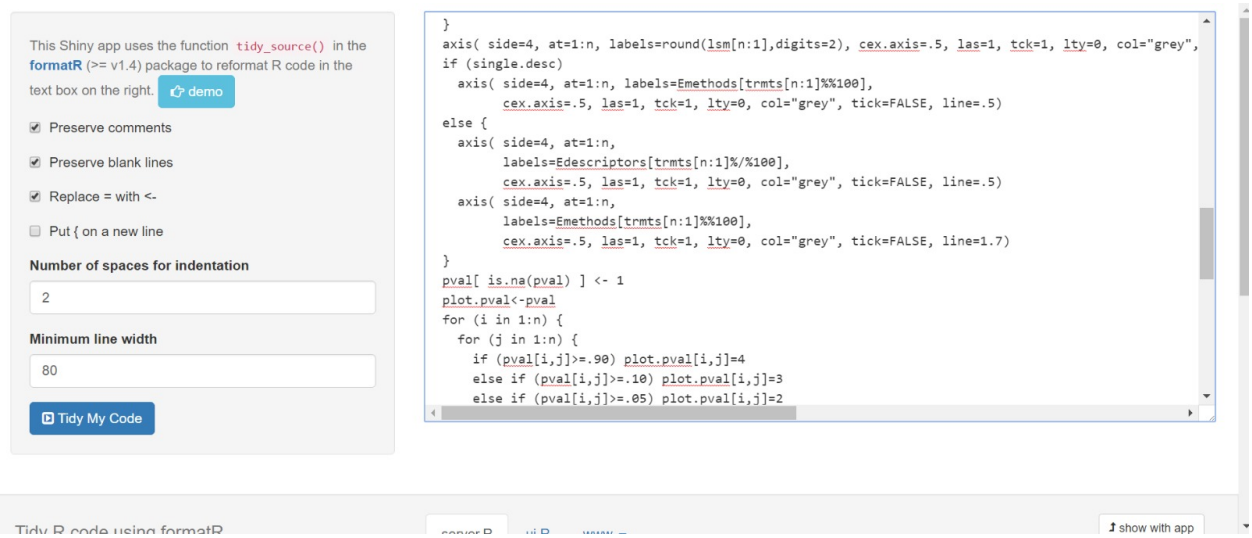


Figure 1: Before tidying

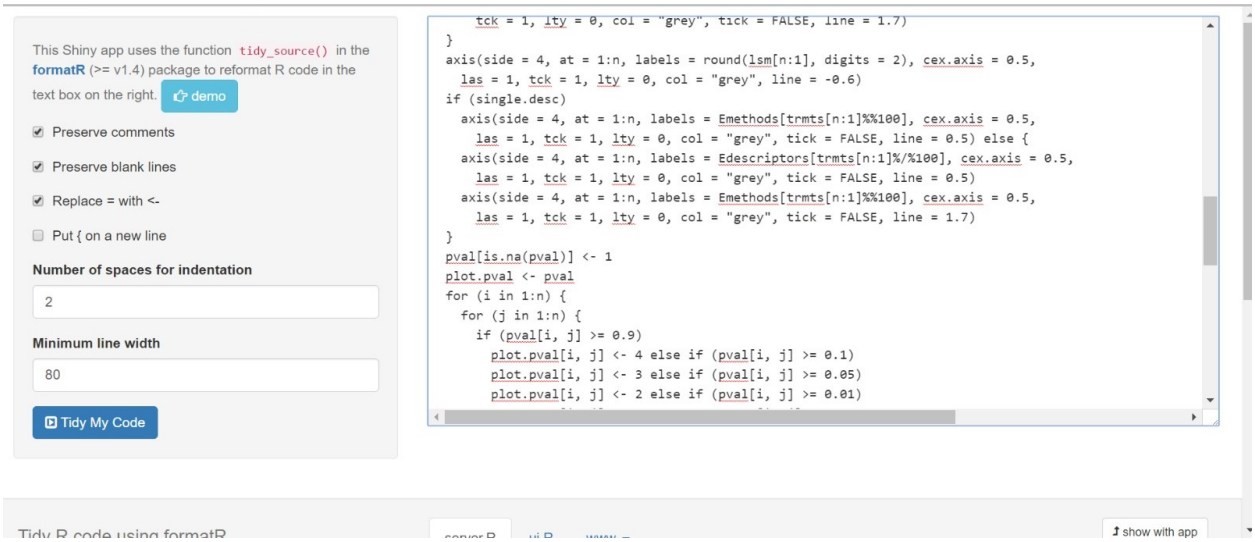


Figure 2: After tidying

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.
  - working on this
- 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:

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

    source("C:/Users/Vestige/Dropbox/ChemModLab/background_newparms.R")
```

```
source("background_newparms.R")

system.time(
bb_1proc <- ModelTrain(data, idcol=1,
  models = c("NNet","PCR","ENet","PLS","Ridge",

              "SVM","LARs","PLSLDA","RPart","Tree","KNN","Forest"),
  nfolds=10, nsplits = 3, user.params = user.params)
)
```

```
## Begining Analysis for Split:  1 and Descriptor Set:  Descriptor Set 1
## Number of Descriptors:  24
## Responses:  3311
## Starting Seed:  11111
## Number of CV folds:  10
##
## Tree ----------------------------------------
## Real time used:  2.03
##  CPU time used:  1.87
##
##
## RPart ---------------------------------------
## Real time used:  6.77
##  CPU time used:  6.63
##
##
## Forest --------------------------------------
## Real time used:  4.48
##  CPU time used:  4.36
##
##
## SVM -----------------------------------------
## Real time used:  4.06
##  CPU time used:  3.93
##
##
## NNet ----------------------------------------
## Real time used:  5.94
##  CPU time used:  5.83
##
##
## KNN -----------------------------------------
## Real time used:  0.44
##  CPU time used:  0.31
```

```
##
##
## PLSLDA ------------------------------------
## Real time used:  1.12
##  CPU time used:  1
##
##
## LARs --------------------------------------
## Real time used:  0.38
##  CPU time used:  0.25
##
##
## Ridge -------------------------------------
## Real time used:  0.45
##  CPU time used:  0.33
##
##
## ENet --------------------------------------
## Real time used:  0.92
##  CPU time used:  0.78
##
##
## PCR ---------------------------------------
## Real time used:  3.1
##  CPU time used:  2.99
##
##
## PLS ---------------------------------------
## Real time used:  0.78
##  CPU time used:  0.61
##
##
## Ending Analysis for Split:  1 and Descriptor Set:  Descriptor Set 1
##
##
## Begining Analysis for Split:  2 and Descriptor Set:  Descriptor Set 1
## Number of Descriptors:  24
## Responses:  3311
## Starting Seed:  22222
## Number of CV folds:  10
##
## Tree --------------------------------------
## Real time used:  1.71
##  CPU time used:  1.56
##
##
## RPart -------------------------------------
## Real time used:  6.78
##  CPU time used:  6.64
##
##
## Forest ------------------------------------
## Real time used:  5.05
##  CPU time used:  4.88
```

```
##
##
## SVM ---------------------------------------
## Real time used:  3.95
##  CPU time used:  3.84
##
##
## NNet --------------------------------------
## Real time used:  5.92
##  CPU time used:  5.81
##
##
## KNN ---------------------------------------
## Real time used:  0.49
##  CPU time used:  0.35
##
##
## PLSLDA ------------------------------------
## Real time used:  1.53
##  CPU time used:  1.39
##
##
## LARs --------------------------------------
## Real time used:  0.37
##  CPU time used:  0.23
##
##
## Ridge -------------------------------------
## Real time used:  0.44
##  CPU time used:  0.28
##
##
## ENet --------------------------------------
## Real time used:  1.2
##  CPU time used:  1.08
##
##
## PCR ---------------------------------------
## Real time used:  3.36
##  CPU time used:  3.19
##
##
## PLS ---------------------------------------
## Real time used:  0.8
##  CPU time used:  0.62
##
##
## Ending Analysis for Split:  2 and Descriptor Set:  Descriptor Set 1
##
##
## Begining Analysis for Split:  3 and Descriptor Set:  Descriptor Set 1
## Number of Descriptors:  24
## Responses:  3311
## Starting Seed:  33333
```

```
## Number of CV folds:  10
##
## Tree ---------------------------------------
## Real time used:  1.58
##  CPU time used:  1.44
##
##
## RPart --------------------------------------
## Real time used:  8.28
##  CPU time used:  8.17
##
##
## Forest -------------------------------------
## Real time used:  4.69
##  CPU time used:  4.57
##
##
## SVM ----------------------------------------
## Real time used:  3.95
##  CPU time used:  3.83
##
##
## NNet ---------------------------------------
## Real time used:  5.8
##  CPU time used:  5.69
##
##
## KNN ----------------------------------------
## Real time used:  0.43
##  CPU time used:  0.31
##
##
## PLSLDA -------------------------------------
## Real time used:  1.08
##  CPU time used:  0.97
##
##
## LARs ---------------------------------------
## Real time used:  0.38
##  CPU time used:  0.24
##
##
## Ridge --------------------------------------
## Real time used:  0.48
##  CPU time used:  0.36
##
##
## ENet ---------------------------------------
## Real time used:  1.19
##  CPU time used:  1.06
##
##
## PCR ----------------------------------------
## Real time used:  3.17
```

```
##  CPU time used:  3.05
##
##
## PLS ---------------------------------------
## Real time used:  0.78
##  CPU time used:  0.62
##
##
## Ending Analysis for Split:  3 and Descriptor Set:  Descriptor Set 1
```

```
##    user  system elapsed
##   91.57    0.87   94.28
```

```
source("background_parallel.R")

cl <- makeCluster(3, outfile="")
registerDoSNOW(cl)

system.time(
  bb_3proc <- ModelTrain(data, idcol=1,
                  models = c("NNet","PCR","ENet","PLS","Ridge",

              "SVM","LARs","PLSLDA","RPart","Tree","KNN","Forest"),
                  nfolds=10, nsplits = 3, user.params = user.params)
)
```

```
## Warning: closing unused connection 8 (<-Vestige-laptop:11592)
```

```
## Warning: closing unused connection 7 (<-Vestige-laptop:11592)
```

```
## Warning: closing unused connection 6 (<-Vestige-laptop:11592)
```

```
## Warning: closing unused connection 5 (<-Vestige-laptop:11592)
```

```
##    user  system elapsed
##    0.51    0.08   50.64
```

```
# Are all the predictions equal?
for(i in 1:3){
  print(all.equal(bb_1proc$allpreds[[i]][[1]], bb_3proc$allpreds[[i]][[1]]))
}
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

```
# Are all the predicted probabilities equal?
for(i in 1:3){
  print(all.equal(bb_1proc$allprobs[[i]][[1]], bb_3proc$allprobs[[i]][[1]]))
}
```

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
```

```
stopImplicitCluster()
```

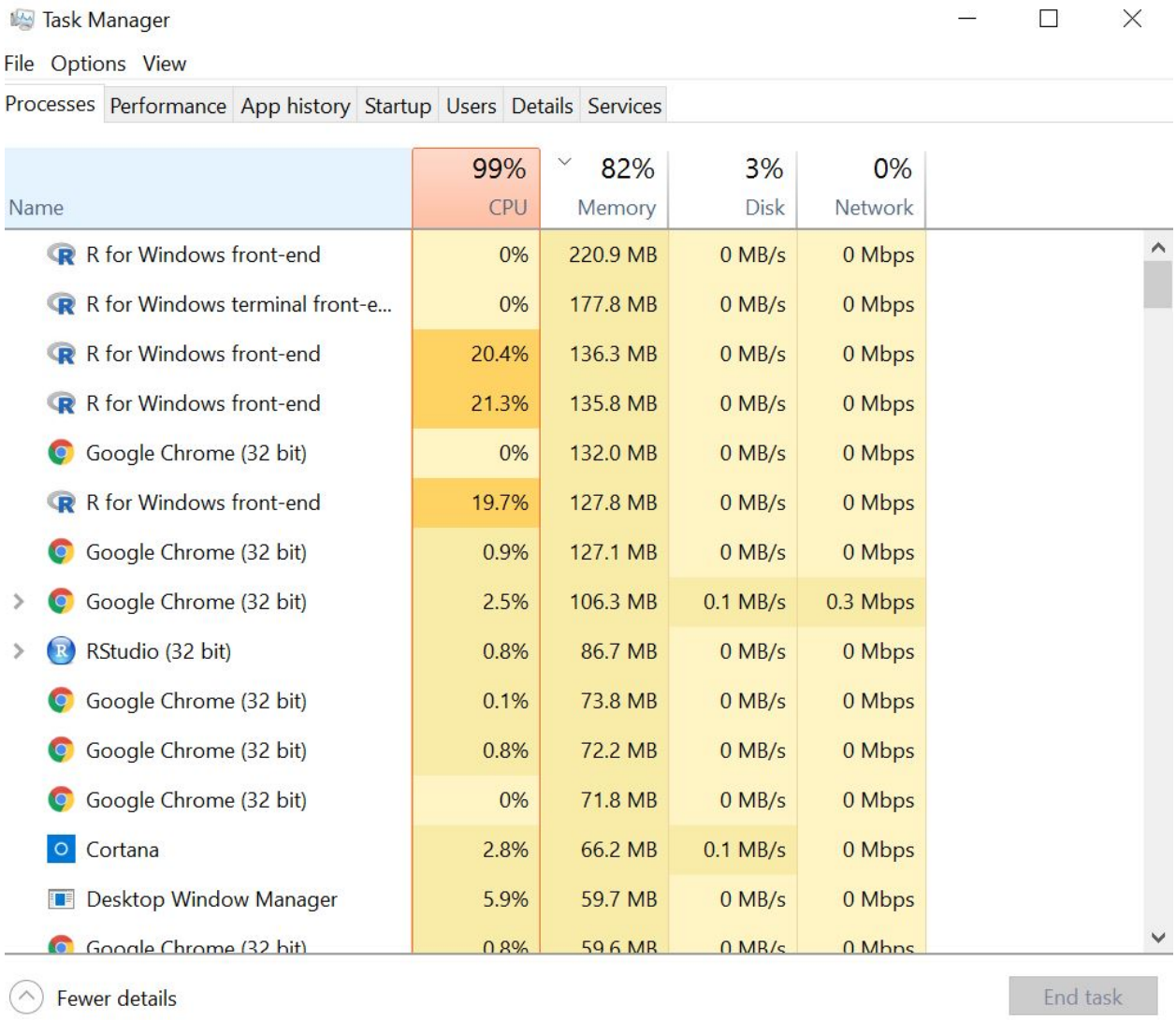


Figure 3: Using 3 processors