Random_forest

Chi Ting Low (s3611774)

6/28/2018

```
library(mlr)
```

```
## Loading required package: ParamHelpers
```

```
library(Amelia)
```

```
## Loading required package: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.7.5, built: 2018-05-07)
## ## Copyright (C) 2005-2018 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
#load data
data <-  read.csv('dataR2.csv')
head(data)
```

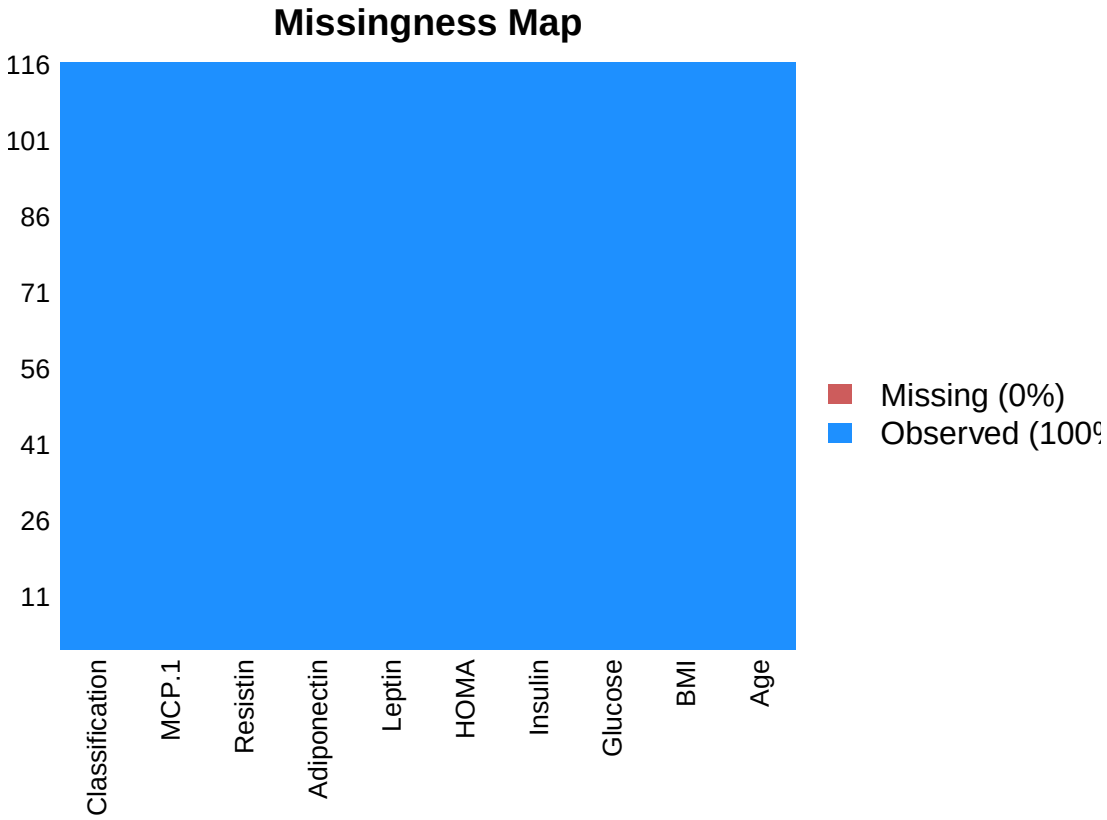
```
##   Age      BMI Glucose Insulin      HOMA  Leptin Adiponectin Resistin
## 1  48 23.50000      70   2.707 0.4674087  8.8071    9.702400  7.99585
## 2  83 20.69049      92   3.115 0.7068973  8.8438    5.429285  4.06405
## 3  82 23.12467      91   4.498 1.0096511 17.9393   22.432040  9.27715
## 4  68 21.36752      77   3.226 0.6127249  9.8827    7.169560 12.76600
## 5  86 21.11111      92   3.549 0.8053864  6.6994    4.819240 10.57635
## 6  49 22.85446      92   3.226 0.7320869  6.8317   13.679750 10.31760
##     MCP.1 Classification
## 1 417.114              1
## 2 468.786              1
## 3 554.697              1
## 4 928.220              1
## 5 773.920              1
## 6 530.410              1
```

```
#summary of the data
summarizeColumns(data)
```

```
##              name    type na       mean       disp     median        mad
## 1             Age integer  0  57.301724  16.112766  56.000000  18.532500
## 2             BMI numeric  0  27.582111   5.020136  27.662416   6.393859
## 3         Glucose integer  0  97.793103  22.525162  92.000000  11.860800
## 4         Insulin numeric  0  10.012086  10.067768   5.924500   3.652385
## 5            HOMA numeric  0   2.694988   3.642043   1.380939   0.961909
## 6          Leptin numeric  0  26.615080  19.183294  20.271000  15.547211
## 7     Adiponectin numeric  0  10.180874   6.843341   8.352692   4.387729
## 8        Resistin numeric  0  14.725966  12.390646  10.827740   7.753264
## 9           MCP.1 numeric  0 534.647000 345.912663 471.322500 304.612031
## 10 Classification integer  0   1.551724   0.499475   2.000000   0.000000
##           min        max nlevs
## 1  24.0000000   89.00000     0
```

```
## 2  18.3700000   38.57876     0
## 3  60.0000000  201.00000     0
## 4   2.4320000   58.46000     0
## 5   0.4674087   25.05034     0
## 6   4.3110000   90.28000     0
## 7   1.6560200   38.04000     0
## 8   3.2100000   82.10000     0
## 9  45.8430000 1698.44000     0
## 10  1.0000000    2.00000     0
```

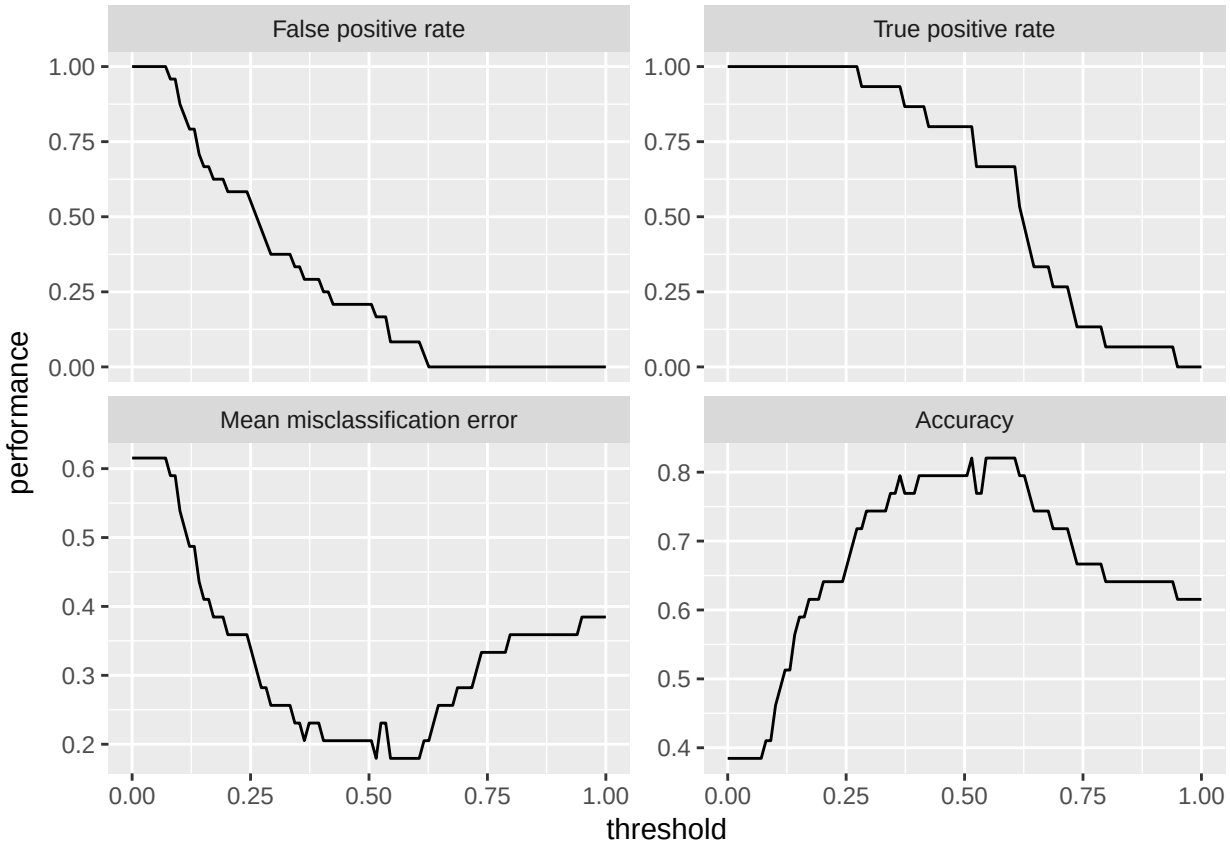
```
#checking missing value
missmap(data)
```



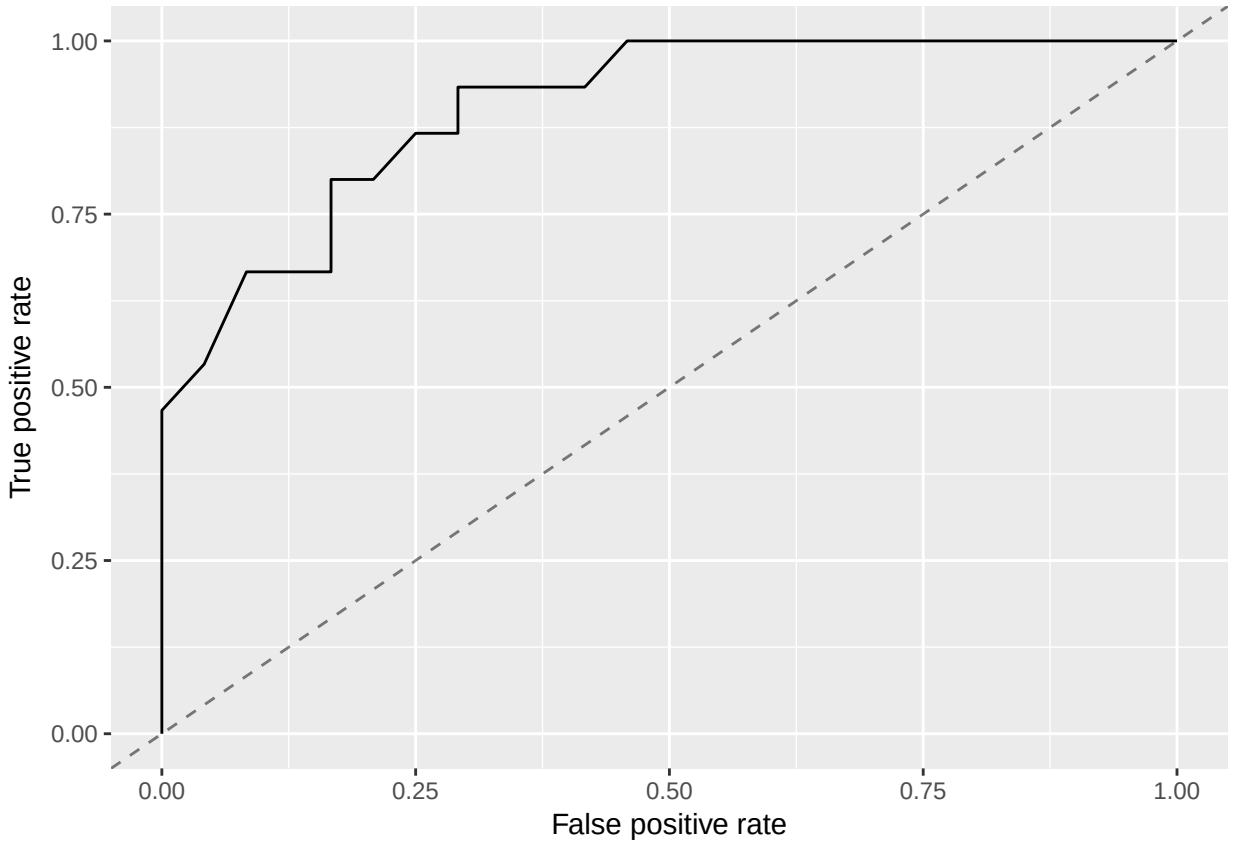

```
#reproducible research
set.seed(123)

#split data set
n = nrow(data)
train.set = sample(n, size = 2/3*n)
test.set = setdiff(1:n, train.set)

#making regression task
task <- makeClassifTask(data = data, target = 'Classification', positive = 1)
lrn <- makeLearner('classif.randomForest', predict.type = 'prob')

mod <- train(lrn, task, subset = train.set)
pred <- predict(mod, task, subset = test.set)

performance(pred, measures = list(fpr, fnr, mmce, acc))
```

```
##       fpr       fnr      mmce       acc
## 0.2083333 0.2000000 0.2051282 0.7948718
```

```
df = generateThreshVsPerfData(pred, measures = list(fpr, tpr, mmce, acc))
plotThreshVsPerf(df)
```




```
plotROCCurves(df)
```




```
calculateConfusionMatrix(pred, relative = TRUE, sums = TRUE, set = 'both')
```

```
## Relative confusion matrix (normalized by row/column):
##          predicted
## true      1          2          -err.-    -n-
##   1       0.80/0.71  0.20/0.14  0.20      17
##   2       0.21/0.29  0.79/0.86  0.21      22
##   -err.-       0.29       0.14  0.21      <NA>
##   -n-     15         24         <NA>      77
##
##
## Absolute confusion matrix:
##          1  2 -err.- -n-
## 1       12  3      3  15
## 2        5 19      5  24
## -err.-  5  3      8  NA
## -n-     17 22     NA  77
```

```
#parameter tuning
getParamSet('classif.randomForest')
```

```
##                      Type  len   Def   Constr Req Tunable Trafo
## ntree             integer    -   500 1 to Inf   -    TRUE     -
## mtry              integer    -     - 1 to Inf   -    TRUE     -
## replace           logical    -  TRUE        -   -    TRUE     -
## classwt     numericvector <NA>     - 0 to Inf   -    TRUE     -
## cutoff      numericvector <NA>     -   0 to 1   -    TRUE     -
## strata            untyped    -     -        -   -   FALSE     -
```

```
## sampsize    integervector <NA>     - 1 to Inf   -    TRUE     -
## nodesize          integer    -     1 1 to Inf   -    TRUE     -
## maxnodes          integer    -     - 1 to Inf   -    TRUE     -
## importance        logical    - FALSE        -   -    TRUE     -
## localImp          logical    - FALSE        -   -    TRUE     -
## proximity         logical    - FALSE        -   -   FALSE     -
## oob.prox          logical    -     -        -   Y   FALSE     -
## norm.votes        logical    -  TRUE        -   -   FALSE     -
## do.trace          logical    - FALSE        -   -   FALSE     -
## keep.forest       logical    -  TRUE        -   -   FALSE     -
## keep.inbag        logical    - FALSE        -   -   FALSE     -
```

```
ps = makeParamSet(
  makeIntegerParam('ntree', lower = 500, upper = 1500),
  makeIntegerParam('mtry', lower = 5, upper = 10)
)

#Stratified resampling using 10 fold
ctrl <- makeTuneControlGrid()
rdesc <- makeResampleDesc('CV', iters = 10L, stratify = T)

#Tune process
res <- tuneParams('classif.randomForest', task = task, resampling = rdesc, par.set = ps, control = ctrl
res
```

```
## Tune result:
## Op. pars: ntree=1167; mtry=9
## mmce.test.mean=0.2107226
```

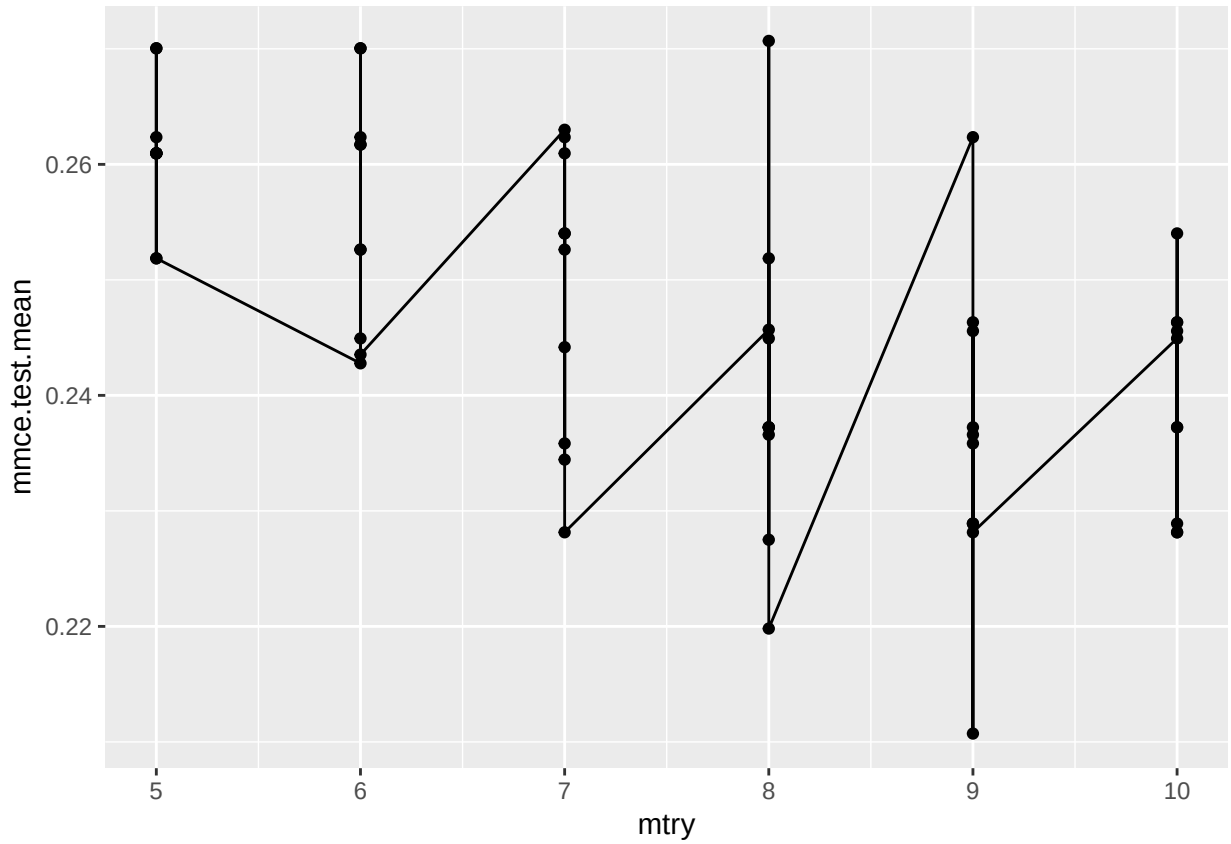
```
res$x
```

```
## $ntree
## [1] 1167
##
## $mtry
## [1] 9
```

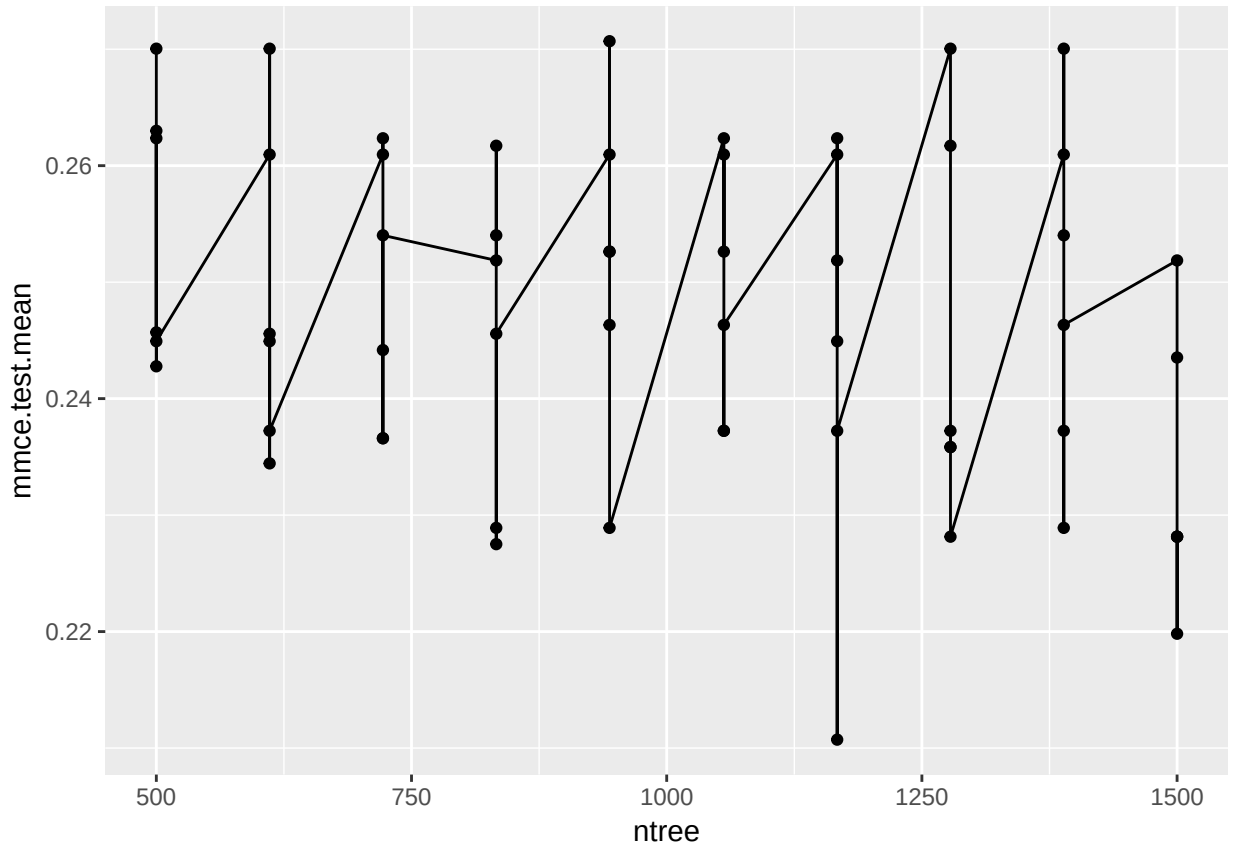
```
res$y
```

```
## mmce.test.mean
##      0.2107226
```

```
tunedata <- generateHyperParsEffectData(res)
plotHyperParsEffect(tunedata, x = 'mtry', y = 'mmce.test.mean', plot.type = 'line')
```



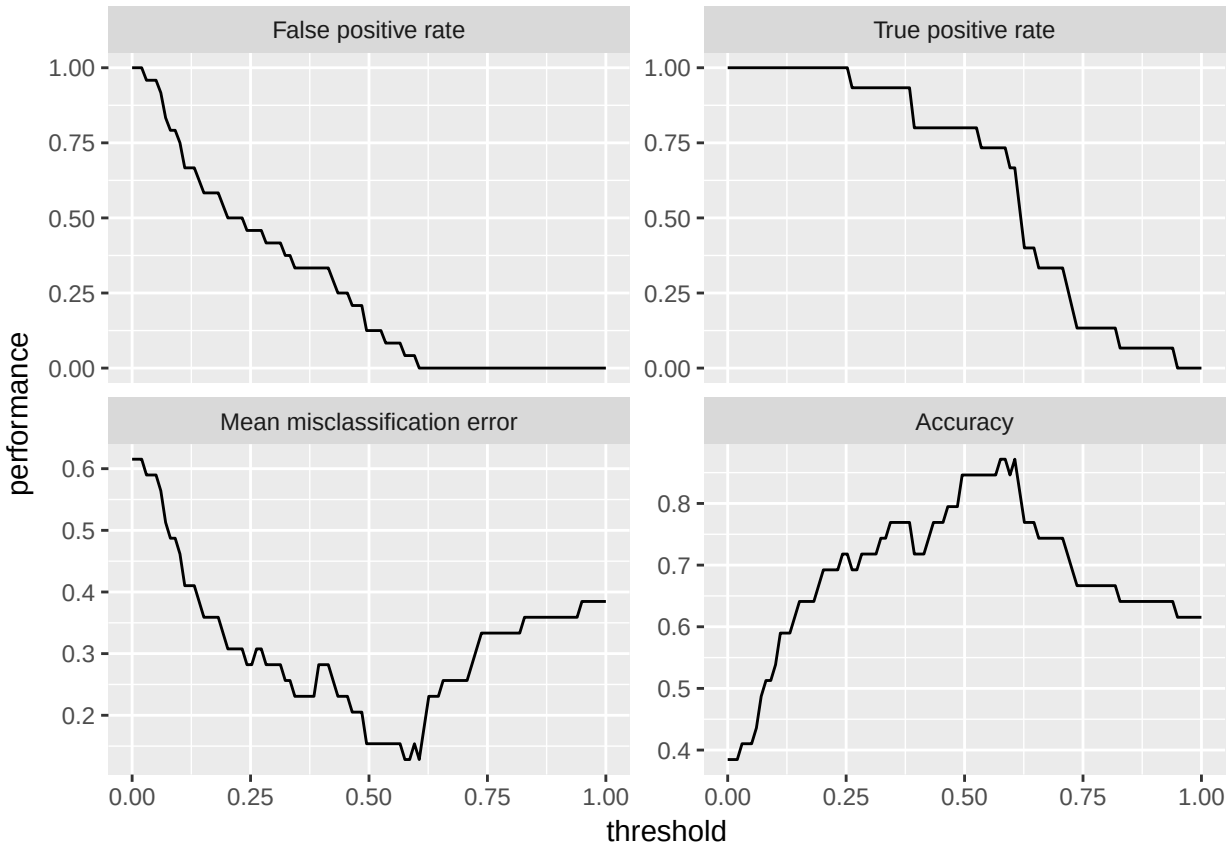

```
plotHyperParsEffect(tunedata, x = 'ntree', y = 'mmce.test.mean', plot.type = 'line')
```



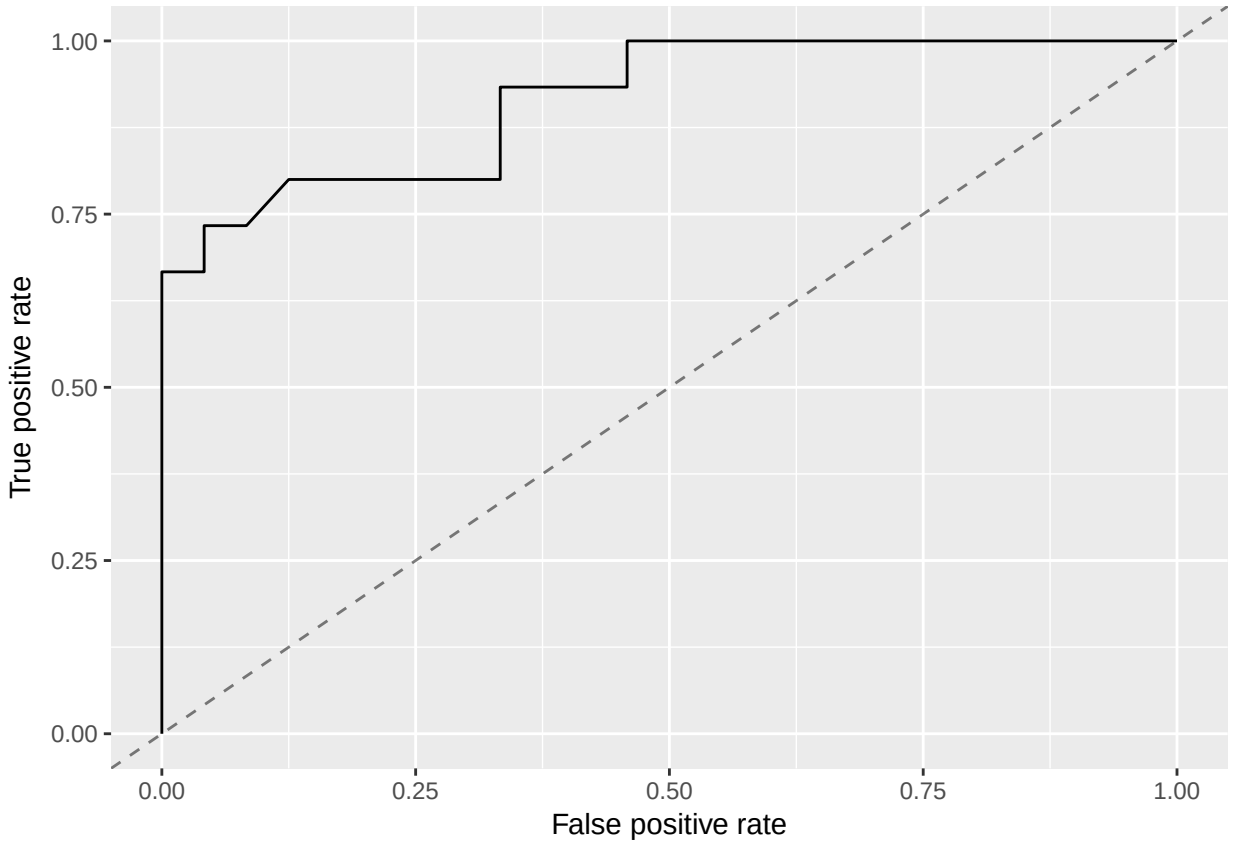

```
tunedlearners <- setHyperPars(makeLearner('classif.randomForest'), par.vals = res$x)


tunedlearners1 <- makeTuneWrapper(lrn, rdesc, mmce, ps, ctrl, show.info = F)
tunedmod <- train(tunedlearners1, task, subset = train.set)

tunedpred <- predict(tunedmod, task, subset = test.set)

performance(tunedpred, measures = list(fpr, fnr, mmce, acc))
```

```
##       fpr       fnr      mmce       acc
## 0.1250000 0.2000000 0.1538462 0.8461538
```

```
df = generateThreshVsPerfData(tunedpred, measures = list(fpr, tpr, mmce, acc))
plotThreshVsPerf(df)
```




```
plotROCCurves(df)
```




```
calculateConfusionMatrix(tunedpred, relative = TRUE, sums = TRUE, set = 'both')
```

```
## Relative confusion matrix (normalized by row/column):
##          predicted
## true      1          2          -err.-     -n-
##   1       0.80/0.80  0.20/0.12  0.20       15
##   2       0.12/0.20  0.88/0.88  0.12       24
##   -err.-        0.20       0.12  0.15       <NA>
##   -n-     15         24         <NA>       77
##
##
## Absolute confusion matrix:
##          1  2 -err.- -n-
## 1       12  3      3  15
## 2        3 21      3  24
## -err.-  3  3      6  NA
## -n-     15 24     NA  77
```