Fitting a prediction rule ensemble using R package pre

Load package and data

First, we have to get the package from GitHub and install. Also, we have to get the example dataset, which is from a paper by Carillo et al. (2001) on predicting depression based on personality scales:

```
library(devtools)
install_github("marjoleinF/pre")
```

```
library(pre)
library(foreign)
car_data <- read.spss("https://github.com/marjoleinF/misc/raw/master/data Carillo et al.sav", to.data.frame = TRUE
)
```

```
## Warning in read.spss("https://github.com/marjoleinF/misc/raw/master/data
## Carillo et al.sav", : C:\Users\fokkemam\AppData\Local\Temp\RtmpQPP7Xs
## \file257460ff4264: Unrecognized record type 7, subtype 18 encountered in
## system file
```

```
names(car_data)
```

```
##  [1] "n1"      "n2"      "n3"      "n4"      "n5"      "n6"      "ntot"
##  [8] "e1"      "e2"      "e3"      "e4"      "e5"      "e6"      "etot"
## [15] "open1"   "open2"   "open3"   "open4"   "open6"   "opentot" "altot"
## [22] "contot"  "bdi"     "sexo"    "edad"    "open5"
```

Fit the ensemble

To fit the prediction rule ensemble, we have to regress depression (bdi) on all other variables.

```
set.seed(42)
car_pre <- pre(formula = bdi ~ ., data = car_data)
```

Note we have to set the seed to be able to reproduce our results later. Above, defaut settings are used. Alternatively, we can generate the initial ensemble like a bagged ensemble or random forest:

```
pre_bag <- pre(formula = bdi ~ ., data = car_data, maxdepth = Inf, learnrate = 0, mtry = Inf, sampfrac = 1)
pre_rf <- pre(formula = bdi ~ ., data = car_data, maxdepth = Inf, learnrate = 0, mtry = ncol(car_data)/3, sampfrac
= 1)
```

Inspect the ensemble

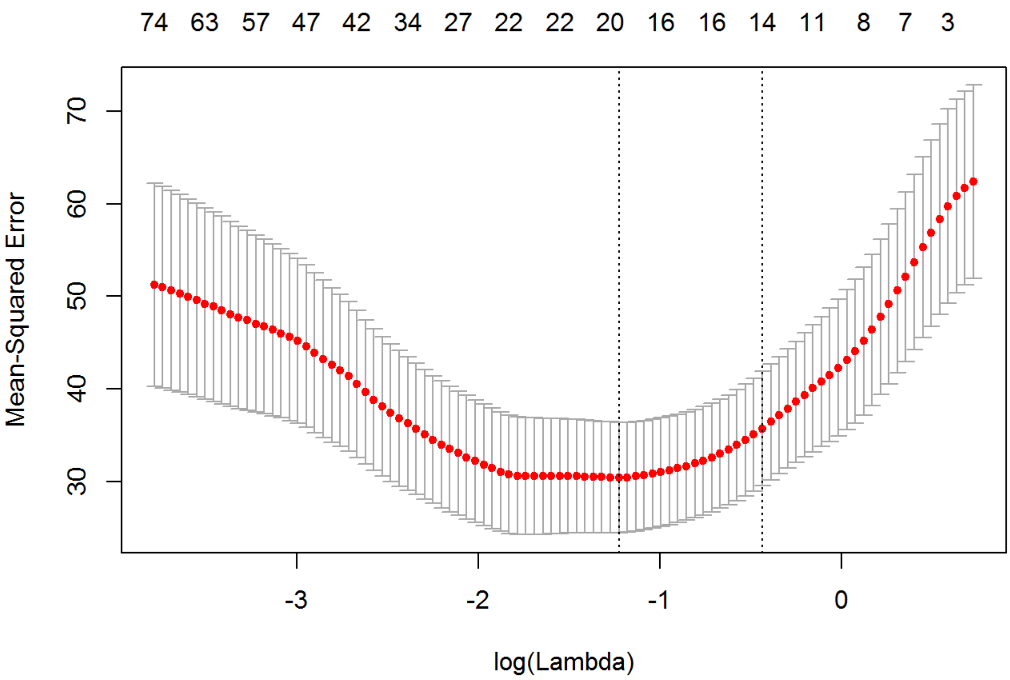
We can check out the resulting prediction rule ensemble using the function:

```
print(car_pre)
```

```
##
## Final ensemble with cv error within 1se of minimum:
##   lambda =  0.64585
##   number of terms = 14
##   mean cv error (se) = 35.71728 (6.164456)
##
##          rule  coefficient                description
##   (Intercept)   8.20331296                       <NA>
##       rule112   2.83654289     n4 > 15 & open4 <= 13
##       rule136  -1.37288781     n2 <= 16 & open4 > 10
##        rule24  -1.19069321  ntot <= 109 & etot > 101
##       rule123  -1.14486626     ntot <= 109 & e6 > 15
##        rule54   1.03135600                    n6 > 19
##        rule22  -1.00553016                   n3 <= 22
##         rule2   0.87187489                    n3 > 17
##       rule150  -0.40073882     n2 <= 16 & open5 > 11
##        rule88   0.37436067    open4 <= 13 & ntot > 82
##       rule121  -0.30189699        n2 <= 16 & e6 > 14
##        rule53  -0.27360995     n6 <= 19 & open4 > 12
##        rule41  -0.21459613    ntot <= 109 & n4 <= 14
##            n3   0.17546398          2 <= n3 <= 30.225
##        rule59   0.03072938                    n1 > 20
```

We may be willing to trade some predictive accuracy in order to have a smaller ensemble, of only four rules, for example:

```
plot(car_pre$glmnet.fit)
```



Note that predictive accuracy will be much compromised in this case. To illustrate, we will use a very small ensemble here. Then we should get the value of the penalty parameter that gives us the desired number of terms:

```
head(data.frame(number_of_nonzero_terms = car_pre$glmnet.fit$nzero, lambda = car_pre$glmnet.fit$lambda))
```

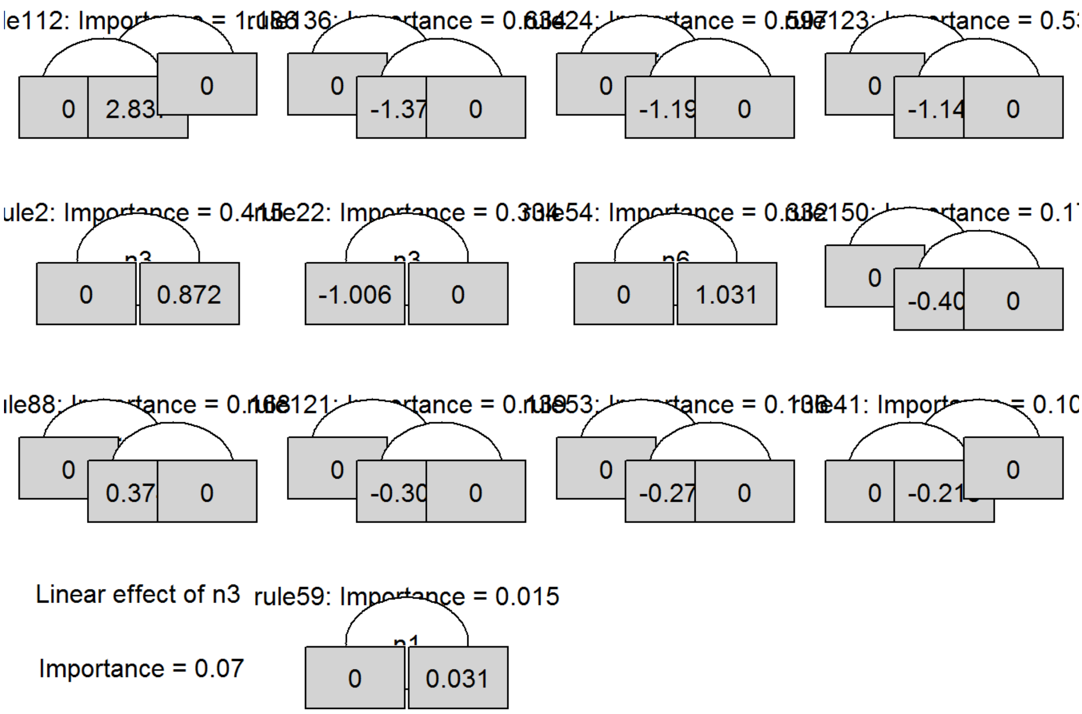
```
##    number_of_nonzero_terms   lambda
## s0                       0 2.066247
## s1                       1 1.972333
## s2                       3 1.882687
## s3                       3 1.797116
## s4                       4 1.715434
## s5                       5 1.637465
```

```
print(car_pre, penalty.par.val = 1.72)
```

```
## Final ensemble with lambda =  1.715434
##   number of terms = 4
##   mean cv error (se) = 58.34011 (10.3031)
##
##          rule  coefficient               description
##   (Intercept)   8.52339118                      <NA>
##        rule24  -0.74435122  ntot <= 109 & etot > 101
##         rule2   0.69482844                   n3 > 17
##       rule123  -0.57775956     ntot <= 109 & e6 > 15
##       rule112   0.01523659     n4 > 15 & open4 <= 13
```

The smaller ensemble will give lower predictive accuracy on new observations, as the plot of the cross-validated mean squared error above indicates. Let's continue with the ensemble selected by default, and plot it:

```
plot(car_pre)
```



We can generate predictions for new observations (though note that these observations are in fact not new, but were already used for training the ensemble):

```
head(coef(car_pre))
```

```
##             rule coefficient               description
## 1    (Intercept)    8.203313                      <NA>
## 34       rule112    2.836543     n4 > 15 & open4 <= 13
## 54       rule136   -1.372888     n2 <= 16 & open4 > 10
## 84        rule24   -1.190693 ntot <= 109 & etot > 101
## 42       rule123   -1.144866     ntot <= 109 & e6 > 15
## 107       rule54    1.031356                   n6 > 19
```

```
predict(car_pre, newdata = car_data[1:10,])
```

```
##         1         2         3         4         5         6         7
##  5.105917 14.348409  4.579525  4.267730  4.930453  8.416082 18.787560
##         8         9        10
##  7.519413  2.298494  4.298459
```

To obtain an estimate of future prediction error through k-fold cross validation:

```
set.seed(42)
cv_car1 <- cvpre(car_pre)
cv_car2 <- cvpre(car_pre, penalty.par.val = 1.72)
cv_car1$accuracy
```

```
## $MSE
## [1] 43.77261
##
## $MAE
## [1] 5.182656
```

```
cv_car2$accuracy
```

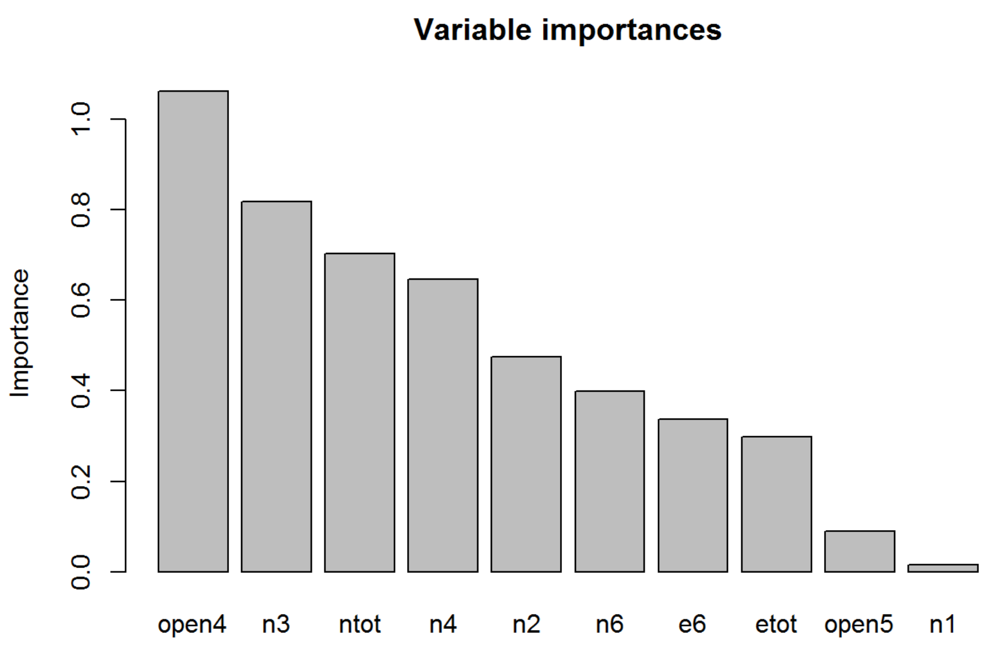
```
## $MSE
## [1] 56.26237
##
## $MAE
## [1] 5.753458
```

```
var(car_data$bdi)
```

```
## [1] 61.46774
```

We can get an estimate of the importance of variables and base learners using the function.

```
importance(car_pre, round = 4,)
```

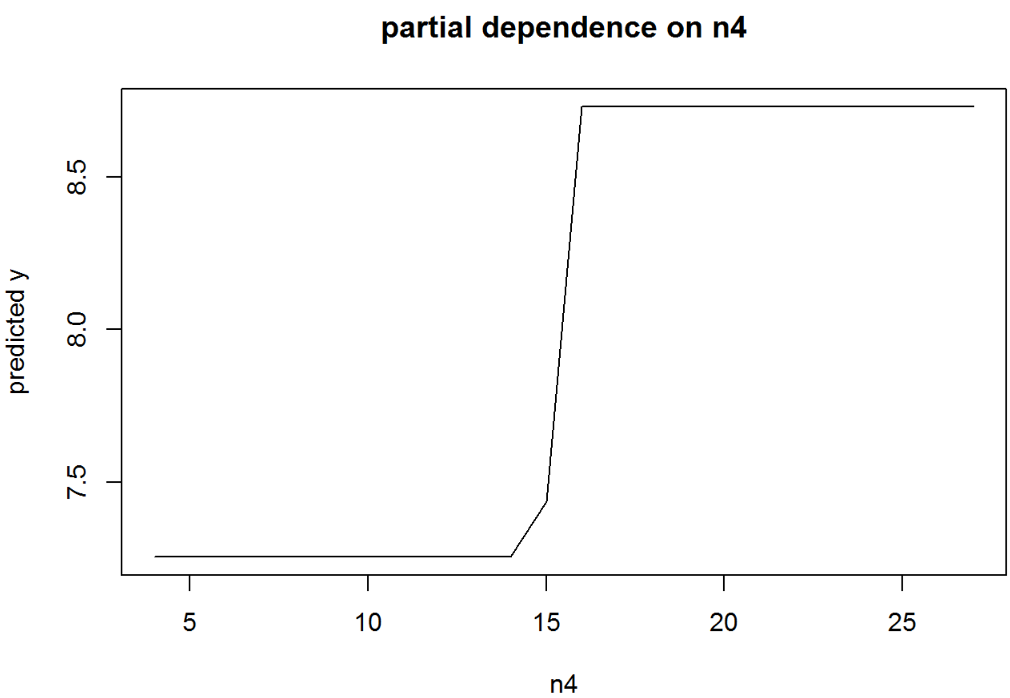


```
## $varimps
##    varname    imp
## 18   open4 1.0622
## 3       n3 0.8189
## 7     ntot 0.7029
## 4       n4 0.6469
## 2       n2 0.4759
## 6       n6 0.3997
## 13      e6 0.3362
## 14    etot 0.2986
## 25   open5 0.0891
## 1       n1 0.0147
##
## $baseimps
##        rule                description    imp coefficient     sd
## 35  rule112    n4 > 15 & open4 <= 13 1.1864      2.8365 0.4183
## 55  rule136    n2 <= 16 & open4 > 10 0.6341     -1.3729 0.4619
## 85   rule24 ntot <= 109 & etot > 101 0.5972     -1.1907 0.5015
## 43  rule123    ntot <= 109 & e6 > 15 0.5330     -1.1449 0.4656
## 81    rule2                  n3 > 17 0.4147      0.8719 0.4756
## 83   rule22                 n3 <= 22 0.3340     -1.0055 0.3322
## 108  rule54                  n6 > 19 0.3318      1.0314 0.3218
## 65  rule150    n2 <= 16 & open5 > 11 0.1783     -0.4007 0.4448
## 131  rule88  open4 <= 13 & ntot > 82 0.1682      0.3744 0.4494
## 41  rule121       n2 <= 16 & e6 > 14 0.1394     -0.3019 0.4619
## 107  rule53    n6 <= 19 & open4 > 12 0.1356     -0.2736 0.4957
## 97   rule41   ntot <= 109 & n4 <= 14 0.1073     -0.2146 0.5002
## 13       n3        2 <= n3 <= 30.225 0.0702      0.1755 0.4000
## 111  rule59                  n1 > 20 0.0147      0.0307 0.4785
```

Note that predictive accuracy would have been much better if we would have gone with the default .

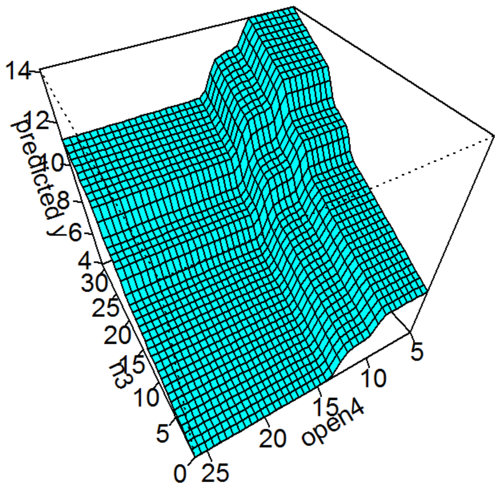
We can assess the effect of a single variable on the predictions of the ensemble using the function:

```
singleplot(car_pre, "n4")
```



We can assess the effect of pairs of variables on the predictions of the ensemble using the function:

```
pairplot(car_pre, c("n3", "open4"), nticks = 6, theta = 240)
```




```
## NOTE: function pairplot uses package 'akima', which has an ACM license.
##     See also https://www.acm.org/publications/policies/software-copyright-notice.
```