Prediction with decision tree method in algae dataset.

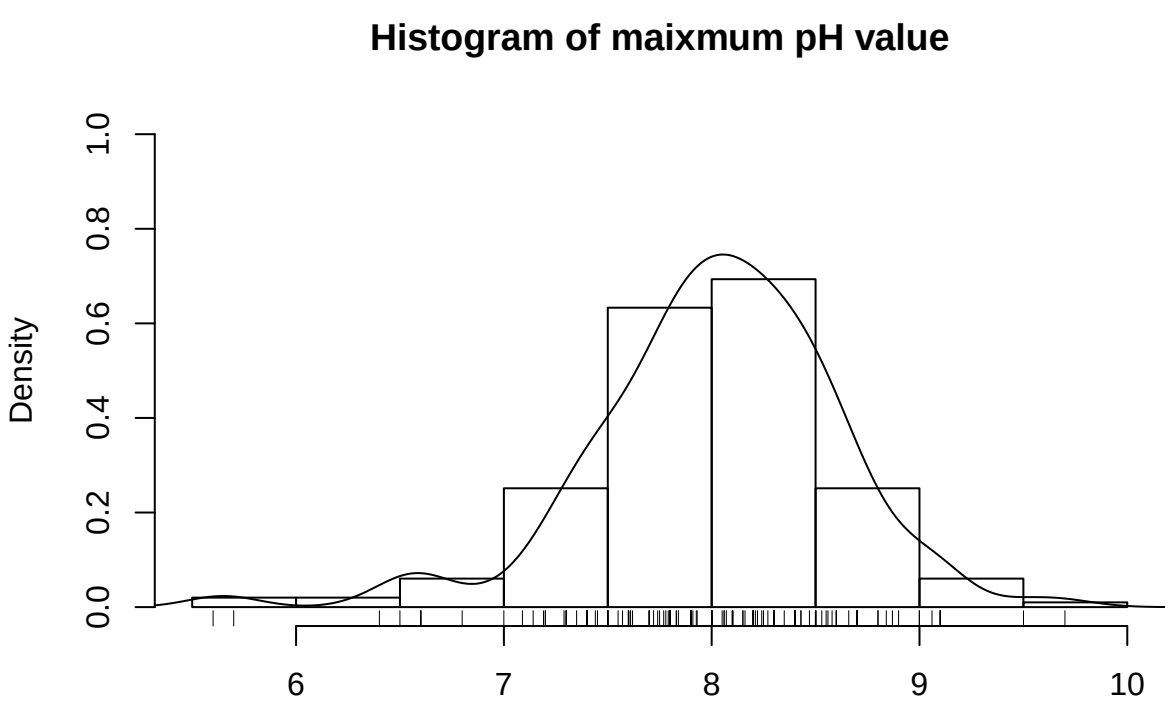
Yan Yan

February 22, 2015

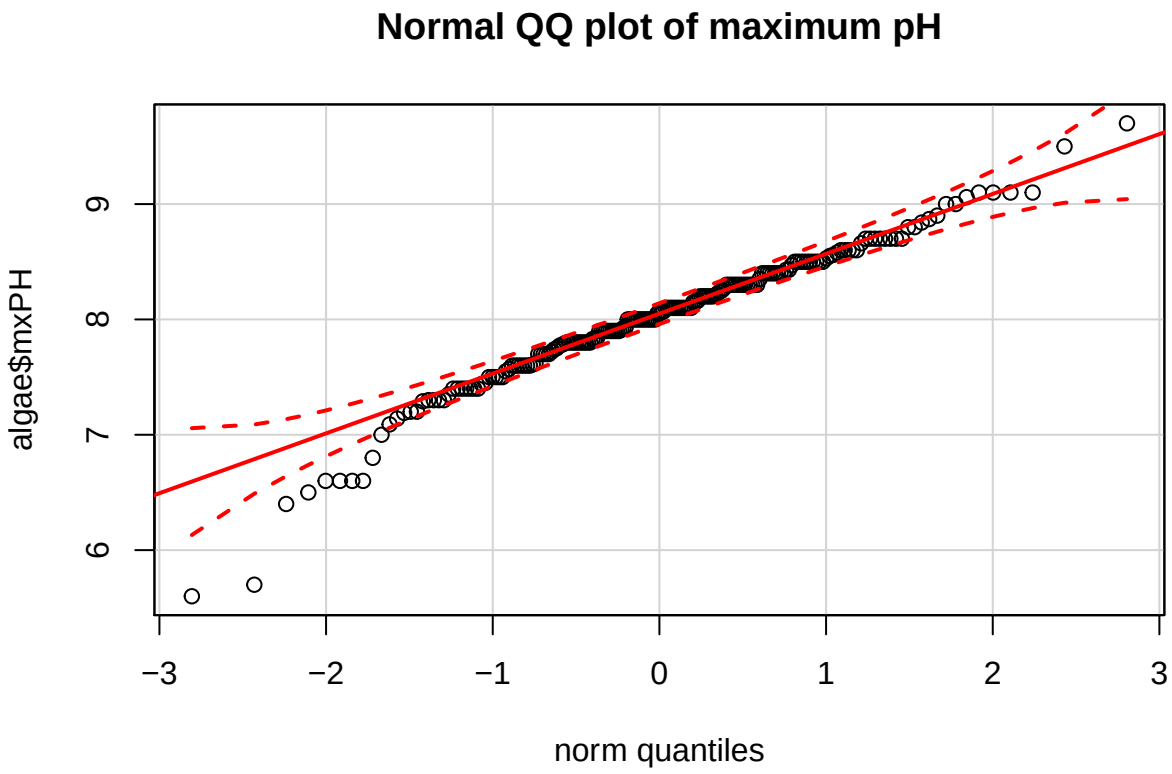
```
library(DMwR)
```

```
## Loading required package: lattice
## Loading required package: grid
```

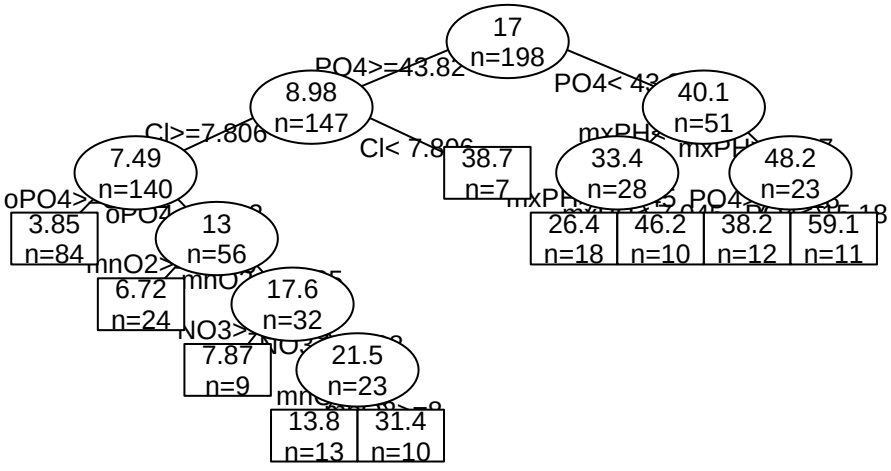
```
library(car)

#par(mfrow=c(1,2))
hist(algae$mxPH, prob=T, xlab='', main='Histogram of maixmum pH value', ylim=0:1)

# A smoothed version of hist graph
lines(density(algae$mxPH, na.rm=T))

#rug() used for plotting, jitter() randomized the original value to avoid overlap
rug(jitter(algae$mxPH))
```




```
#Q-Q graph, plot the scatterplot of value and Normal Distribution quantiles,
#and then the band chart of 95% confidence interval
qqPlot(algae$mxPH, main='Normal QQ plot of maximum pH')
```



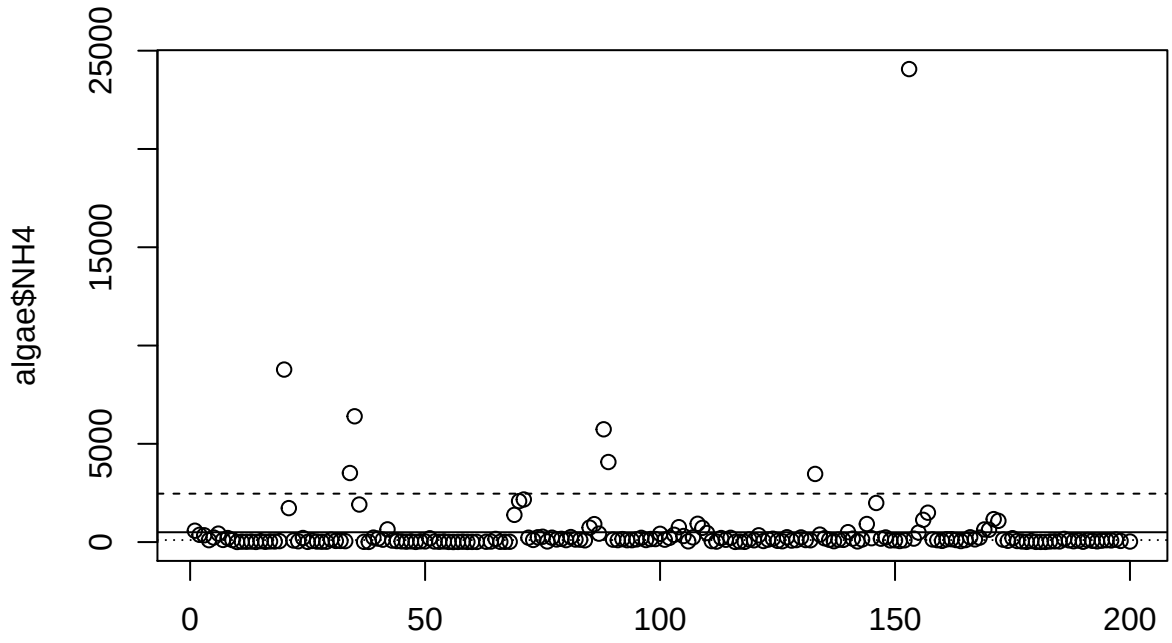

```
#par(mfrow=c(1,2))

#Boxplot of oPO4, the line in the middle of the box is the median, the upper
#boudary is the 3rd quantiles and the lower boundary is the 1st quantile.
boxplot(algae$oPO4, ylab = "Orthophosphate(oPO4)")

rug(jitter(algae$oPO4), side=2)
#Plot the average value line in the Boxplot
abline(h = mean(algae$oPO4, na.rm = T), lty = 2)
```



We can see the the distribution of oPO4 is concentrated in the lower values, so it is positive-skewed.

For the sperated values, we can handle with the following method.

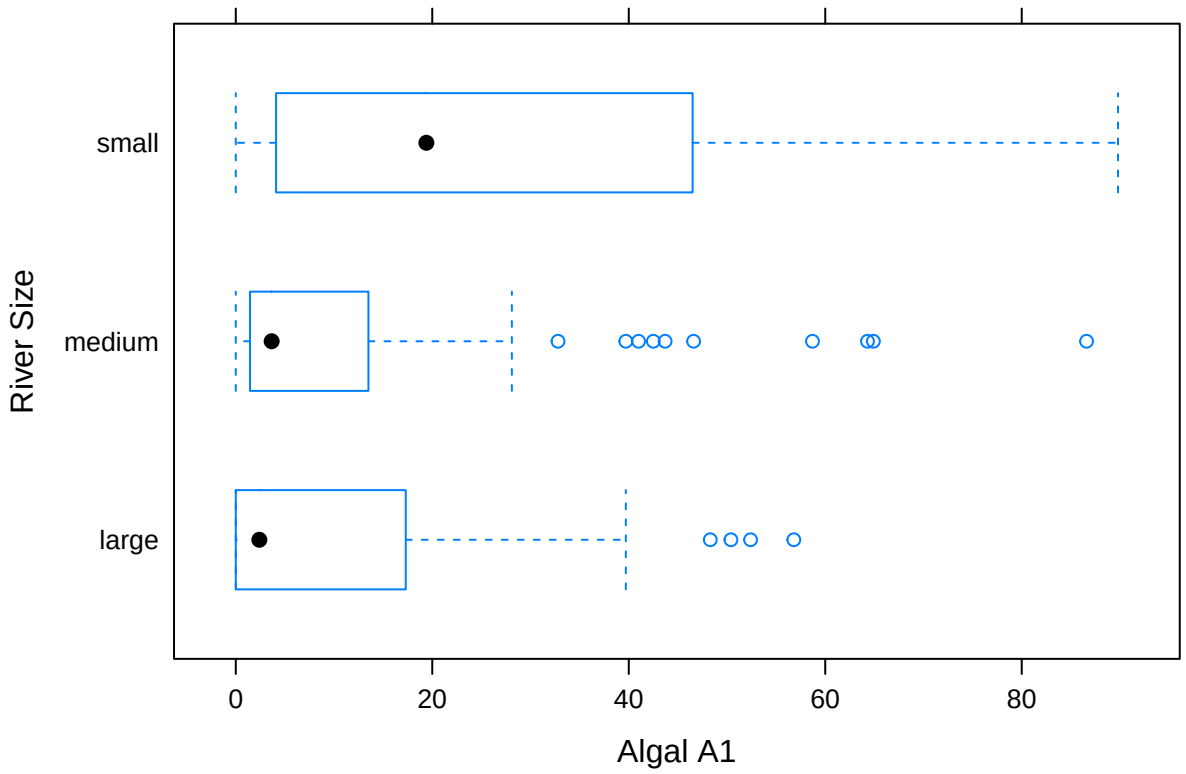
```
plot(algae$NH4, xlab = " ")
abline(h = mean(algae$NH4, na.rm = T), lty = 1) #avarage value
abline(h = mean(algae$NH4, na.rm = T) + sd(algae$NH4, na.rm = T), lty = 2)
abline(h = median(algae$NH4, na.rm = T), lty = 3) #median
```




```
#A interact method can be realized by using identify() function
#identify(algae$NH4)
#clicked.lines <- identify(algae$NH4)
#algae[clicked.lines]
```

To find how the distribution varies due to other variables.

```
library(lattice)
bwplot(size ~ a1, data=algae, ylab = 'River Size', xlab = 'Algal A1')
```

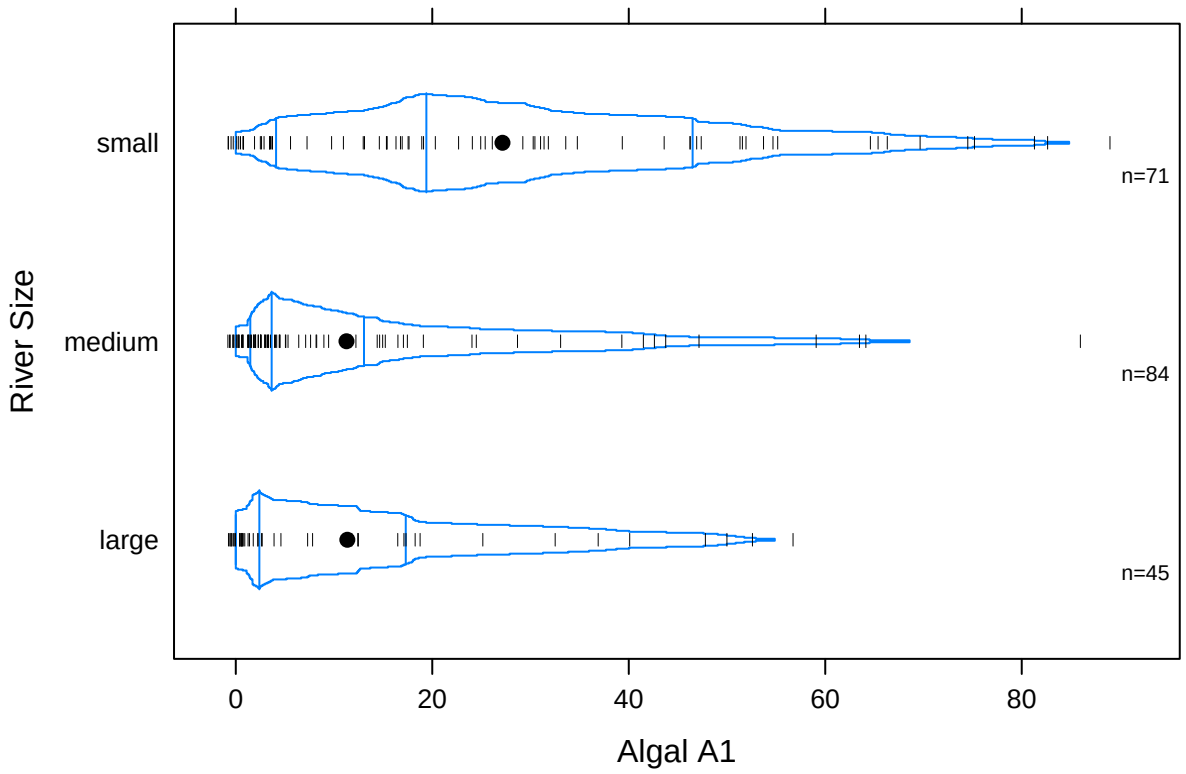


Also we can use the quantile box plot

```
library(Hmisc)
```

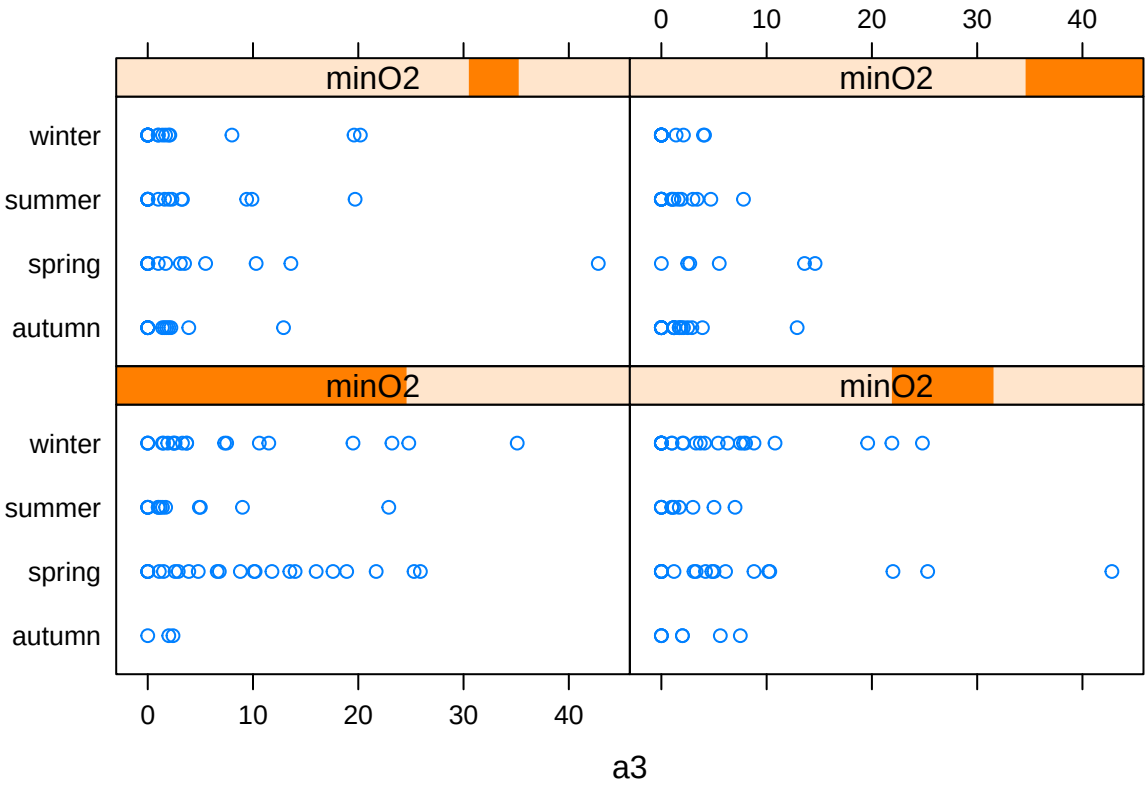
```
## Loading required package: survival
## Loading required package: splines
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
bwplot(size ~ a1, data = algae, panel = panel.bpplot,
       probs = seq(.01, .49, by = .01), datadensity = TRUE,
       ylab = 'River Size', xlab = 'Algal A1')
```



We can also discretize the data into several intervals, which means transfer the continus numerical data into factor data, for example:

```
minO2 <- equal.count(na.omit(algae$mnO2), number = 4, overlap = 1/5)
stripplot(season ~ a3|minO2, data=algae[!is.na(algae$mnO2),])
```



Then we go to the missing value process.

```
#caculate the numbers of lines with missing value
algae[!complete.cases(algae),]
```

```
##     season   size  speed mxPH mnO2    Cl   NO3 NH4    oPO4     PO4  Chla
## 28  autumn  small   high 6.80 11.1 9.000 0.630  20   4.000      NA  2.70
## 38  spring  small   high 8.00   NA 1.450 0.810  10   2.500   3.000  0.30
## 48  winter  small    low   NA 12.6 9.000 0.230  10   5.000   6.000  1.10
## 55  winter  small   high 6.60 10.8    NA 3.245  10   1.000   6.500    NA
## 56  spring  small medium 5.60 11.8    NA 2.220   5   1.000   1.000    NA
## 57  autumn  small medium 5.70 10.8    NA 2.550  10   1.000   4.000    NA
## 58  spring  small   high 6.60  9.5    NA 1.320  20   1.000   6.000    NA
## 59  summer  small   high 6.60 10.8    NA 2.640  10   2.000  11.000    NA
## 60  autumn  small medium 6.60 11.3    NA 4.170  10   1.000   6.000    NA
## 61  spring  small medium 6.50 10.4    NA 5.970  10   2.000  14.000    NA
## 62  summer  small medium 6.40   NA    NA    NA  NA      NA  14.000    NA
## 63  autumn  small   high 7.83 11.7 4.083 1.328  18   3.333   6.667    NA
## 116 winter medium   high 9.70 10.8 0.222 0.406  10  22.444  10.111    NA
## 161 spring  large    low 9.00  5.8    NA 0.900 142 102.000 186.000 68.05
## 184 winter  large   high 8.00 10.9 9.055 0.825  40  21.083  56.091    NA
## 199 winter  large medium 8.00  7.6    NA    NA  NA      NA      NA    NA
##       a1   a2  a3   a4  a5  a6  a7
## 28  30.3  1.9 0.0  0.0 2.1 1.4 2.1
## 38  75.8  0.0 0.0  0.0 0.0 0.0 0.0
## 48  35.5  0.0 0.0  0.0 0.0 0.0 0.0
## 55  24.3  0.0 0.0  0.0 0.0 0.0 0.0
## 56  82.7  0.0 0.0  0.0 0.0 0.0 0.0
## 57  16.8  4.6 3.9 11.5 0.0 0.0 0.0
```

```
## 58  46.8  0.0 0.0 28.8 0.0 0.0 0.0
## 59  46.9  0.0 0.0 13.4 0.0 0.0 0.0
## 60  47.1  0.0 0.0  0.0 0.0 1.2 0.0
## 61  66.9  0.0 0.0  0.0 0.0 0.0 0.0
## 62  19.4  0.0 0.0  2.0 0.0 3.9 1.7
## 63  14.4  0.0 0.0  0.0 0.0 0.0 0.0
## 116 41.0  1.5 0.0  0.0 0.0 0.0 0.0
## 161  1.7 20.6 1.5  2.2 0.0 0.0 0.0
## 184 16.8 19.6 4.0  0.0 0.0 0.0 0.0
## 199  0.0 12.5 3.7  1.0 0.0 0.0 4.9
```

```
nrow(algae[!complete.cases(algae),])
```

```
## [1] 16
```

```
#delete the lines with missing values
#algae <- na.omit(algae)

data(algae)
algae <- algae[-manyNAs(algae),]
algae <- centralImputation(algae)

#cor(algae[, 4:18], use = "complete.obs")
symnum(cor(algae[, 4:18], use = "complete.obs"))
```

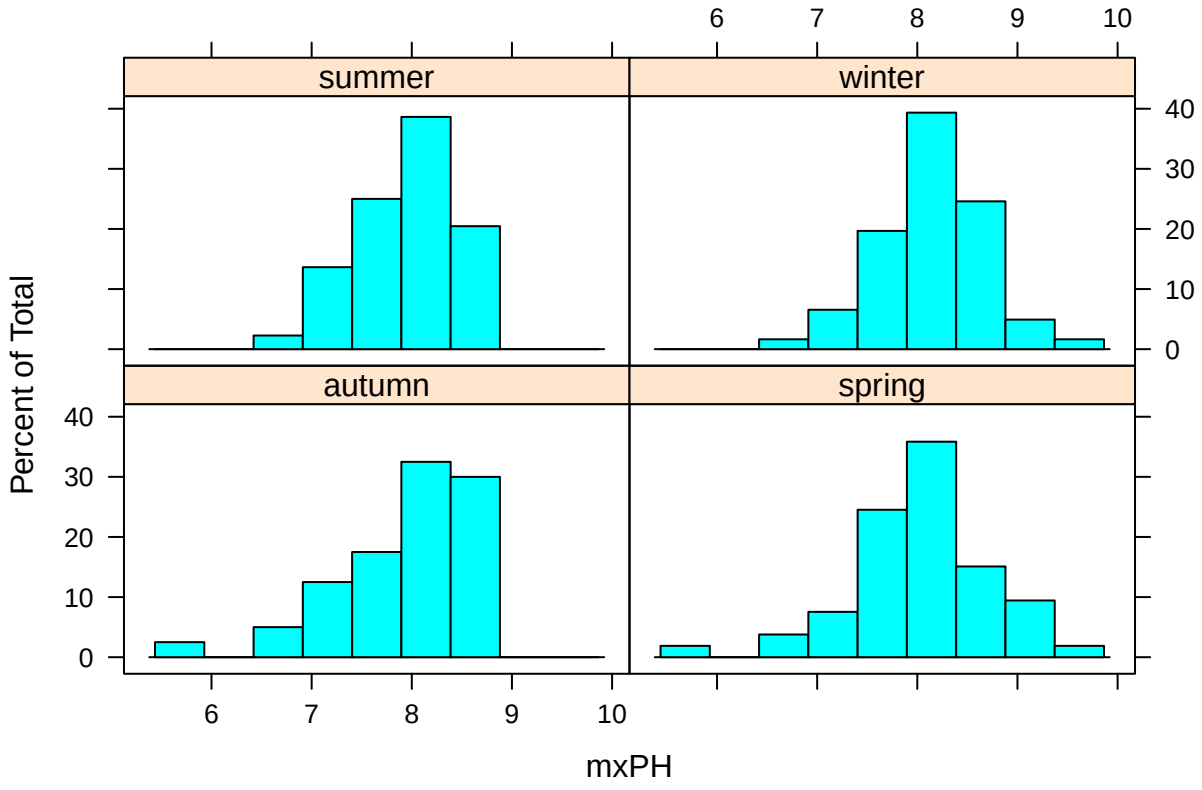
```
##      mP mO Cl NO NH o P Ch a1 a2 a3 a4 a5 a6 a7
## mxPH 1
## mnO2    1
## Cl         1
## NO3           1
## NH4           ,  1
## oPO4    .  .        1
## PO4     .  .        * 1
## Chla .                  1
## a1          .        . .    1
## a2   .                    .     1
## a3                               1
## a4      .             .             1
## a5                                   1
## a6            .  .                   .  1
## a7                                        1
## attr(,"legend")
## [1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

```
data(algae)
algae <- algae[-manyNAs(algae),]
lm(PO4 ~ oPO4, data = algae)
```

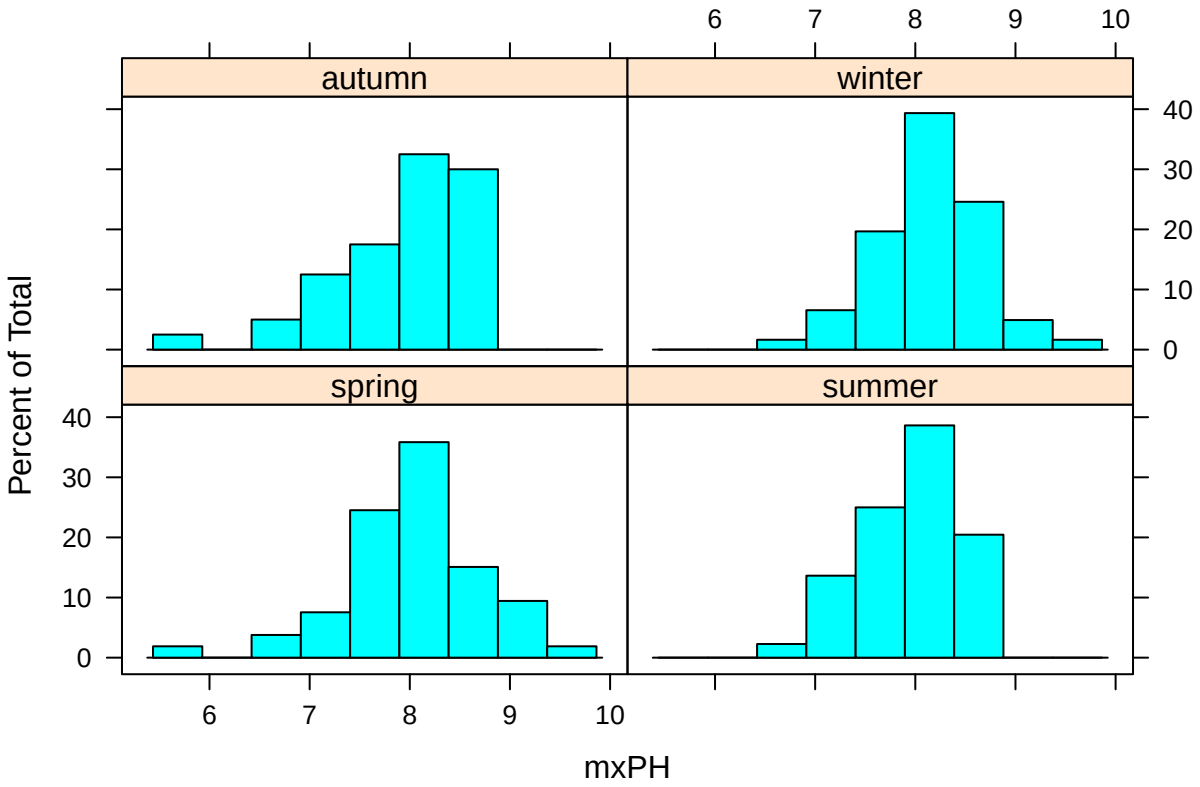
```
##
## Call:
## lm(formula = PO4 ~ oPO4, data = algae)
##
```

```
## Coefficients:
## (Intercept)         oPO4
##       42.90         1.29
```

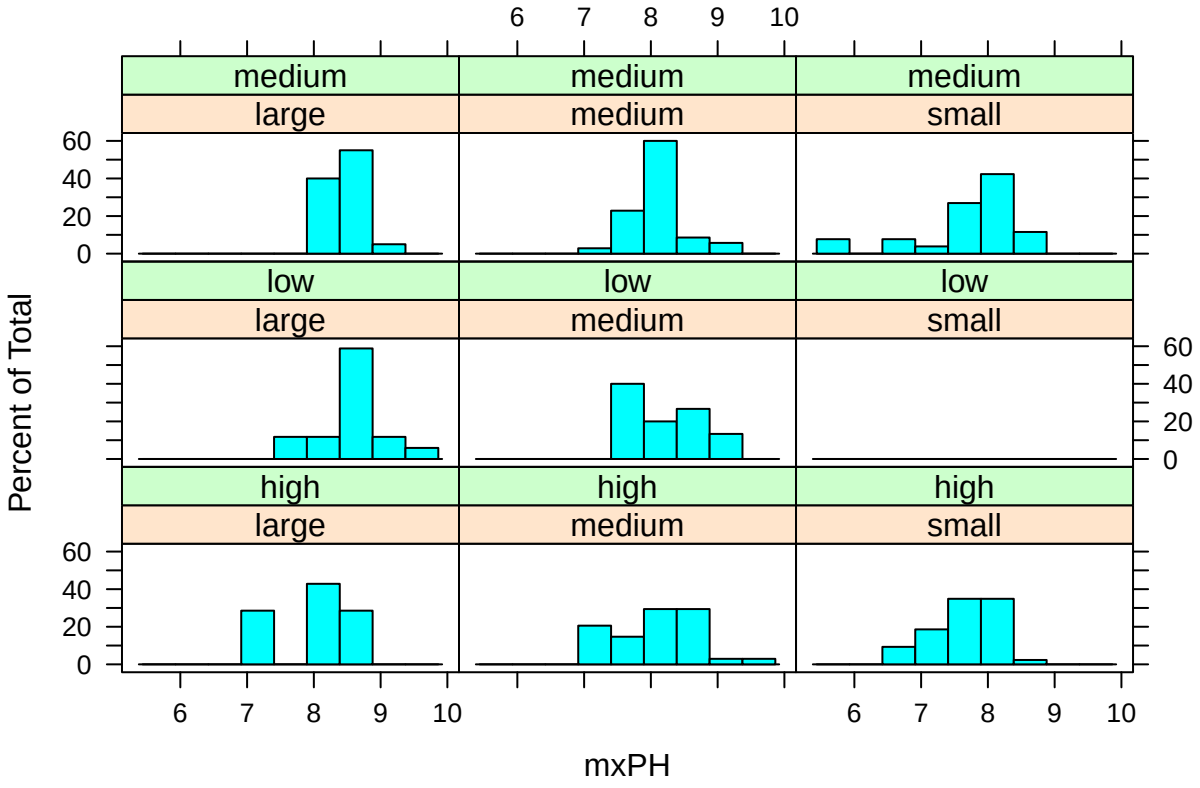
```
algae[28, "PO4"] <- 42.897 + 1.293 * algae[28, "oPO4"]

data(algae)
algae <- algae[-manyNAs(algae),]
fillPO4 <- function(oP){
        if(is.na(oP))
                return(NA)
        else return(42.897 + 1.293 * oP)
}

algae[is.na(algae$PO4), "PO4"] <- sapply(algae[is.na(algae$PO4), "oPO4"], fillPO4)

histogram(~mxPH | season, data = algae)
```



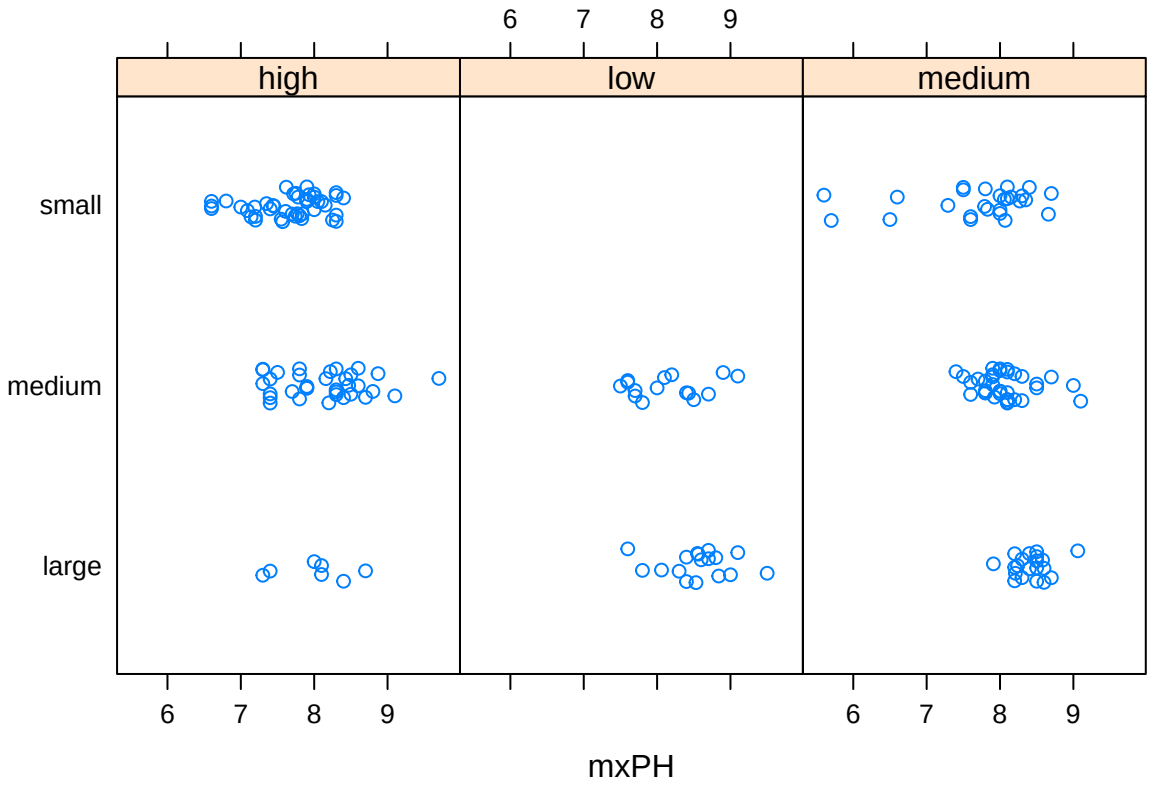

```
algae$season <- factor(algae$season, levels = c("spring", "summer", "autumn", "winter"))
histogram(~mxPH | season, data = algae)
```




```
histogram(~mxPH | size * speed, data =algae)
```



```
stripplot(size ~ mxPH | speed, data = algae, jitter = T)
```



If we assume the two kinds of water were similar, then in one sample there were some missing value, the missing value might be similar to the correspond one in the other kind of water.

```
data(algae)
algae <- algae[-manyNAs(algae),]

#algae <- knnImputation(algae, k = 10)
#or
algae <- knnImputation(algae, k = 10, meth = "median")
```

First we build a multivariate analysis for prediction

```
data(algae)
algae <- algae[-manyNAs(algae),]
clean.algae <- knnImputation(algae, k = 10)

lm.a1 <- lm(a1 ~., data = clean.algae[,1:12])
summary(lm.a1)
```

```
##
## Call:
## lm(formula = a1 ~ ., data = clean.algae[, 1:12])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -37.68 -11.89  -2.57   7.41  62.19
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.94206   24.01088    1.79   0.0754 .
## seasonspring  3.72698    4.13774    0.90   0.3689
## seasonsummer  0.74760    4.02071    0.19   0.8527
## seasonwinter  3.69295    3.86539    0.96   0.3406
## sizemedium    3.26373    3.80205    0.86   0.3918
## sizesmall     9.68214    4.17997    2.32   0.0217 *
## speedlow      3.92208    4.70631    0.83   0.4057
## speedmedium   0.24676    3.24187    0.08   0.9394
## mxPH         -3.58912    2.70353   -1.33   0.1860
## mnO2          1.05264    0.70502    1.49   0.1372
## Cl           -0.04017    0.03366   -1.19   0.2343
## NO3          -1.51124    0.55134   -2.74   0.0067 **
## NH4           0.00163    0.00100    1.63   0.1052
## oPO4         -0.00543    0.03988   -0.14   0.8918
## PO4          -0.05224    0.03075   -1.70   0.0911 .
## Chla         -0.08802    0.08000   -1.10   0.2727
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.6 on 182 degrees of freedom
## Multiple R-squared:  0.373,  Adjusted R-squared:  0.321
## F-statistic: 7.22 on 15 and 182 DF,  p-value: 2.44e-12
```

Use anova() to simplify the model, when anova() was adopted into the simple linear model, this function provide a Sequential analysis of variance.

```
library(stats)
anova(lm.a1)
```

```
## Analysis of Variance Table
##
## Response: a1
##            Df Sum Sq Mean Sq F value  Pr(>F)
## season      3     85      28    0.09 0.96519
## size        2  11401    5701   18.31 5.7e-08 ***
## speed       2   3934    1967    6.32 0.00222 **
## mxPH        1   1329    1329    4.27 0.04026 *
## mnO2        1   2287    2287    7.34 0.00737 **
## Cl          1   4304    4304   13.82 0.00027 ***
## NO3         1   3418    3418   10.98 0.00111 **
## NH4         1    404     404    1.30 0.25638
## oPO4        1   4788    4788   15.38 0.00012 ***
## PO4         1   1406    1406    4.51 0.03496 *
## Chla        1    377     377    1.21 0.27265
## Residuals 182  56668     311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the result we can see that season contibute least for the error of fitting, then we delete it from the model.

```
lm2.a1 <- update(lm.a1, .~ .-season)
summary(lm2.a1)
```

```
##
## Call:
## lm(formula = a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 +
##     oPO4 + PO4 + Chla, data = clean.algae[, 1:12])
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -36.46 -11.95  -3.04   7.44  63.73
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 44.953287  23.237838    1.93   0.0546 .
## sizemedium   3.309210   3.782522    0.87   0.3828
## sizesmall   10.273096   4.122316    2.49   0.0136 *
## speedlow     3.054627   4.610807    0.66   0.5085
## speedmedium -0.297687   3.181858   -0.09   0.9256
## mxPH        -3.268428   2.657659   -1.23   0.2203
## mnO2         0.801176   0.658964    1.22   0.2256
## Cl          -0.038188   0.033379   -1.14   0.2541
## NO3         -1.533430   0.547655   -2.80   0.0057 **
## NH4          0.001578   0.000995    1.59   0.1146
## oPO4        -0.006239   0.039509   -0.16   0.8747
## PO4         -0.050954   0.030519   -1.67   0.0967 .
## Chla        -0.084137   0.079446   -1.06   0.2910
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.6 on 185 degrees of freedom
## Multiple R-squared:  0.368,  Adjusted R-squared:  0.327
## F-statistic: 8.98 on 12 and 185 DF,  p-value: 1.76e-13
```

Then let's consider a comparison between model1 and model2.

```
anova(lm.a1,lm2.a1)
```

```
## Analysis of Variance Table
##
## Model 1: a1 ~ season + size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 +
##     PO4 + Chla
## Model 2: a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 + PO4 +
##     Chla
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1    182 56668
## 2    185 57116 -3      -448 0.48    0.7
```

From the F-test to the anova analysis of the 2 models, the two models were significant different. Then we continue to do the variable remove process. In R we use

```
final.lm <- step(lm.a1)
```

```
## Start:  AIC=1152
## a1 ~ season + size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 +
##     PO4 + Chla
##
##          Df Sum of Sq   RSS  AIC
## - season  3       448 57116 1148
## - speed   2       270 56938 1149
## - oPO4    1         6 56674 1150
## - Chla    1       377 57045 1151
## - Cl      1       443 57112 1152
## - mxPH    1       549 57217 1152
## <none>                56668 1152
## - mnO2    1       694 57363 1152
## - NH4     1       826 57494 1153
## - PO4     1       898 57567 1153
## - size    2      1857 58526 1154
## - NO3     1      2339 59008 1158
##
## Step:  AIC=1148
## a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 + PO4 +
##     Chla
##
##         Df Sum of Sq   RSS  AIC
## - speed  2       211 57327 1144
## - oPO4   1         8 57124 1146
## - Chla   1       346 57462 1147
## - Cl     1       404 57520 1147
## - mnO2   1       456 57572 1147
## - mxPH   1       467 57583 1147
## <none>               57116 1148
## - NH4    1       776 57892 1148
## - PO4    1       861 57977 1149
## - size   2      2176 59292 1151
## - NO3    1      2420 59537 1154
##
## Step:  AIC=1144
## a1 ~ size + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 + PO4 + Chla
##
##        Df Sum of Sq   RSS  AIC
## - oPO4  1        16 57343 1142
## - Chla  1       223 57550 1143
## - mnO2  1       414 57740 1144
## - Cl    1       473 57799 1144
## - mxPH  1       484 57810 1144
## <none>              57327 1144
## - NH4   1       720 58047 1145
## - PO4   1       809 58136 1145
## - size  2      2061 59388 1147
## - NO3   1      2380 59706 1150
##
## Step:  AIC=1142
```

```
## a1 ~ size + mxPH + mnO2 + Cl + NO3 + NH4 + PO4 + Chla
##
##        Df Sum of Sq   RSS  AIC
## - Chla  1       208 57551 1141
## - mnO2  1       403 57746 1142
## - Cl    1       471 57814 1142
## - mxPH  1       520 57863 1142
## <none>              57343 1142
## - NH4   1       704 58047 1143
## - size  2      2050 59393 1145
## - NO3   1      2370 59713 1148
## - PO4   1      5818 63161 1160
##
## Step:  AIC=1141
## a1 ~ size + mxPH + mnO2 + Cl + NO3 + NH4 + PO4
##
##        Df Sum of Sq   RSS  AIC
## - mnO2  1       435 57986 1141
## - Cl    1       438 57989 1141
## <none>              57551 1141
## - NH4   1       747 58298 1142
## - mxPH  1       833 58384 1142
## - size  2      2218 59768 1145
## - NO3   1      2667 60218 1148
## - PO4   1      6310 63860 1160
##
## Step:  AIC=1141
## a1 ~ size + mxPH + Cl + NO3 + NH4 + PO4
##
##        Df Sum of Sq   RSS  AIC
## - NH4   1       531 58517 1140
## - Cl    1       585 58571 1141
## <none>              57986 1141
## - mxPH  1       819 58805 1141
## - size  2      2478 60464 1145
## - NO3   1      2251 60237 1146
## - PO4   1      9098 67084 1167
##
## Step:  AIC=1140
## a1 ~ size + mxPH + Cl + NO3 + PO4
##
##        Df Sum of Sq   RSS  AIC
## <none>              58517 1140
## - mxPH  1       784 59301 1141
## - Cl    1       836 59353 1141
## - NO3   1      1988 60505 1145
## - size  2      2664 61181 1145
## - PO4   1      8576 67093 1165
```

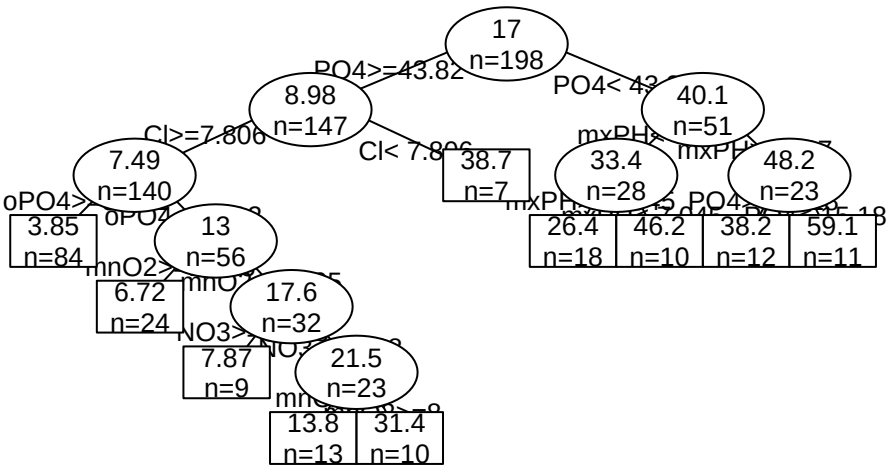
```
summary(final.lm)
```

```
##
## Call:
## lm(formula = a1 ~ size + mxPH + Cl + NO3 + PO4, data = clean.algae[,
```

```
##     1:12])
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -28.87 -12.73  -3.74   8.42  62.93
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  57.2855    20.9613    2.73   0.0069 **
## sizemedium    2.8005     3.4019    0.82   0.4114
## sizesmall    10.4064     3.8224    2.72   0.0071 **
## mxPH         -3.9708     2.4820   -1.60   0.1113
## Cl           -0.0523     0.0317   -1.65   0.1003
## NO3          -0.8953     0.3515   -2.55   0.0116 *
## PO4          -0.0591     0.0112   -5.29  3.3e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.5 on 191 degrees of freedom
## Multiple R-squared:  0.353,  Adjusted R-squared:  0.332
## F-statistic: 17.3 on 6 and 191 DF,  p-value: 5.55e-16
```

From which we can see that the R-square is still not high, which means it's not appropriate for the prediction using multivariate analysis.

```
library(rpart)
data(algae)
algae <- algae[-manyNAs(algae),]
rt.a1 <- rpart(a1 ~., data = algae[, 1:12])
prettyTree(rt.a1)
```




```
printcp(rt.a1)
```

```
##
## Regression tree:
## rpart(formula = a1 ~ ., data = algae[, 1:12])
##
```

```
## Variables actually used in tree construction:
## [1] Cl   mnO2 mxPH NO3  oPO4 PO4
##
## Root node error: 90401/198 = 457
##
## n= 198
##
##      CP nsplit rel error xerror xstd
## 1 0.406      0      1.00   1.01 0.13
## 2 0.072      1      0.59   0.69 0.11
## 3 0.031      2      0.52   0.66 0.11
## 4 0.030      3      0.49   0.70 0.11
## 5 0.028      4      0.46   0.71 0.12
## 6 0.028      5      0.43   0.71 0.12
## 7 0.018      6      0.41   0.72 0.11
## 8 0.016      7      0.39   0.70 0.11
## 9 0.010      9      0.35   0.73 0.11
```

Using *rpart()* to construct a tree would stop at some conditions were satisfied, like: the deviation less than a threshold; the samples number is smaller than a threshold; the deepth of the tree is larger than a threshold. In *rpart()* the parameters were *cp*, *minsplit* and *madepth*. Using *printcp()* to show the results.

Use the treshold of cp = 0.08

```
rt2.a1 <- prune(rt.a1, cp = 0.08)
rt2.a1
```

```
## n= 198
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 198 90400 17.00
##   2) PO4>=43.82 147 31280  8.98 *
##   3) PO4< 43.82 51 22440 40.10 *
```

```
(rt.a1 <- rpartXse(a1 ~., data = algae[, 1:12]))
```

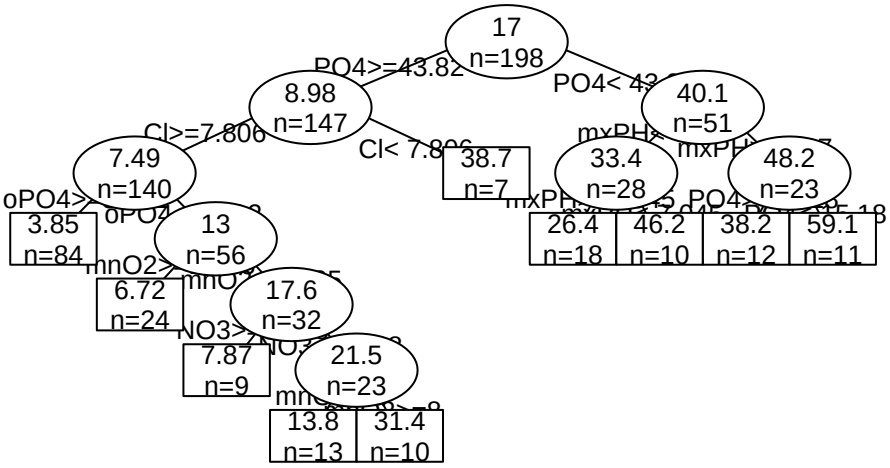
```
## n= 198
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 198 90400.0 17.00
##   2) PO4>=43.82 147 31280.0  8.98
##     4) Cl>=7.167 142 21760.0  7.53 *
##     5) Cl< 7.167 5   746.8 50.14 *
##   3) PO4< 43.82 51 22440.0 40.10
##     6) PO4>=3.5 47 17750.0 37.37 *
##     7) PO4< 3.5 4   225.6 72.18 *
```

Using snip.part() to pruning interactively

```
first.tree <- rpart(a1 ~., data = algae[,1:12])
snip.rpart(first.tree, c(4, 7))
```

```
## n= 198
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 198 90400 17.000
##    2) PO4>=43.82 147 31280  8.980
##      4) Cl>=7.806 140 21620  7.493 *
##      5) Cl< 7.806 7  3158 38.710 *
##    3) PO4< 43.82 51 22440 40.100
##      6) mxPH< 7.87 28 11450 33.450
##       12) mxPH>=7.045 18  5146 26.390 *
##       13) mxPH< 7.045 10  3798 46.150 *
##      7) mxPH>=7.87 23  8241 48.200 *
```

```
prettyTree(first.tree)
```




```
#snip.rpart(firt.tree) interactively pruning
```

Prediction:

```
lm.predictions.a1 <- predict(final.lm, clean.algae)
rt.predictions.a1 <- predict(rt.a1, algae)
```

The average error

```
(mae.a1.lm <- mean(abs(lm.predictions.a1 - algae[, "a1"])))
```

```
## [1] 13.11
```

```
(mae.a1.rt <- mean(abs(rt.predictions.a1 - algae[,"a1"])))
```

```
## [1] 9.829
```

or the MSE

```
(mse.a1.lm <- mean((lm.predictions.a1 - algae[, "a1"])^2))
```

```
## [1] 295.5
```

```
(mse.a1.rt <- mean((rt.predictions.a1 - algae[, "a1"])^2))
```

```
## [1] 204.5
```

The problems with the mse were that the not unified units so that it's hard to explain. So the NMSE normorlized MSE was involved in:

```
(nmse.a1.lm <- mean((lm.predictions.a1 - algae[,'a1'])^2) / mean((mean(algae[,'a1']) - algae[,'a1'])^2))
```

```
## [1] 0.6473
```

```
(nmse.a1.rt <- mean((rt.predictions.a1 - algae[,'a1'])^2) / mean((mean(algae[,'a1']) - algae[,'a1'])^2))
```

```
## [1] 0.4479
```

The NMSE is smaller, the better the model is, if NMSE is larger than 1, the model is shit.

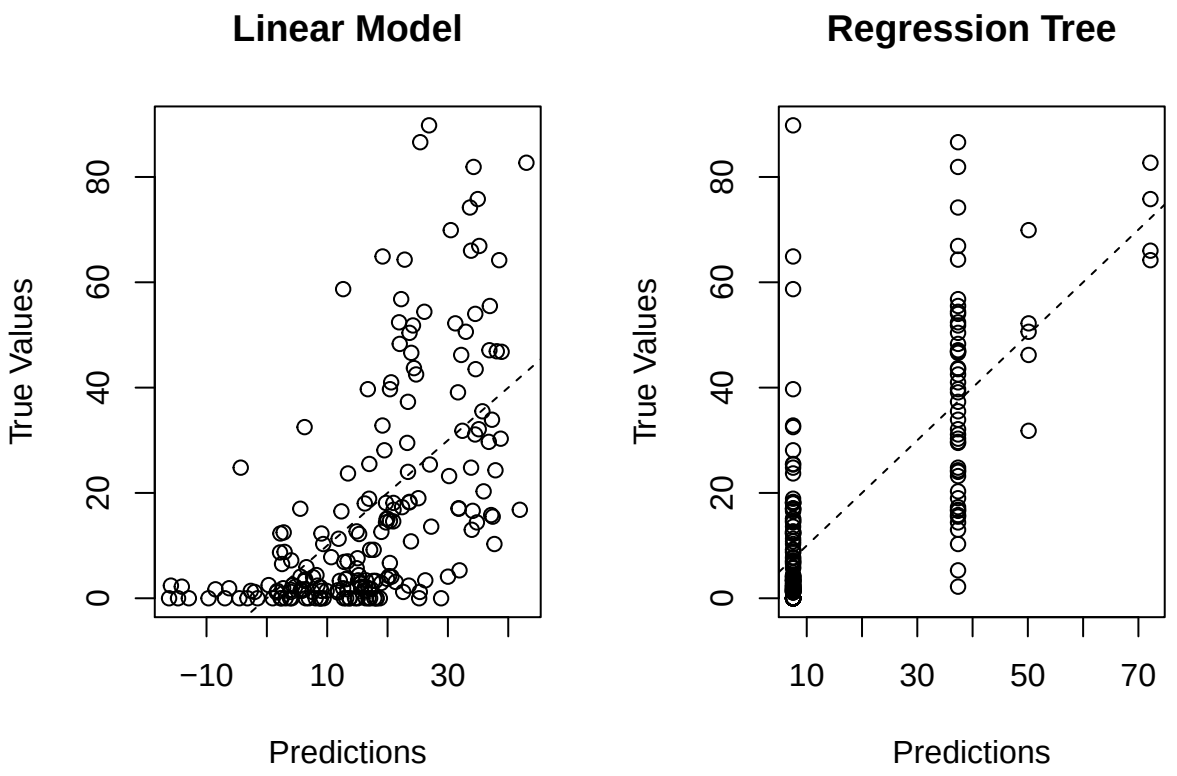
Use *regr.eval()* to calculate the linear regression model performance parameters. Sample:

```
regr.eval(algae[,"a1"], rt.predictions.a1, train.y = algae[,"a1"])
```

```
##      mae      mse     rmse     mape     nmse     nmae
##   9.8294 204.4866  14.2999      Inf   0.4479   0.5884
```

The visualization of the model and predictions in scatterplot:

```
old.par <- par(mfrow = c(1, 2))
plot(lm.predictions.a1, algae[,"a1"], main = "Linear Model", xlab = "Predictions", ylab = "True Values")
abline(0, 1, lty = 2)
plot(rt.predictions.a1, algae[,"a1"], main = "Regression Tree", xlab = "Predictions", ylab = "True Valu
abline(0, 1, lty=2)
```




```
par(old.par)
```

Use *ifelse()* to improve model prediction results. Three parameters include logic condition, the value of the function when logic is true, and the value when false.

```
sensible.lm.predictions.a1 <- ifelse(lm.predictions.a1 < 0, 0, lm.predictions.a1)
regr.eval(algae[,"a1"], lm.predictions.a1, stats = c("mae", "mse"))
```

```
##    mae    mse
##  13.11 295.54
```

```
regr.eval(algae[,"a1"], sensible.lm.predictions.a1, stats = c("mae", "mse"))
```

```
##    mae    mse
##  12.48 286.29
```

Use *experimentComparison()* to do model selection and comparison, three parameters include dataset, models and the parameters in the experiment.

```
cv.rpart <- function(form, train, test, ...){
        m <- rpartXse(form, train, ...)
        p <- predict(m, test)
        mse <- mean((p - resp(form, test))^2)
        c(nmse = mse/mean((mean(resp(form, train)) - resp(form, test))^2))
}

cv.lm <- function(form, train, test, ...){
```

```
        m <- lm(form, train, ...)
        p <- predict(m, test)
        p <- ifelse(p < 0, 0, p)
        mse <- mean((p - resp(form, test))^2)
        c(nmse = mse/mean((mean(resp(form, train)) - resp(form, test))^2))
}

res <- experimentalComparison(
        c(dataset(a1~., clean.algae[, 1:12], 'a1')),
        c(variants('cv.lm'), variants('cv.rpart', se = c(0, 0.5, 1))), cvSettings(3,10,1234))
```

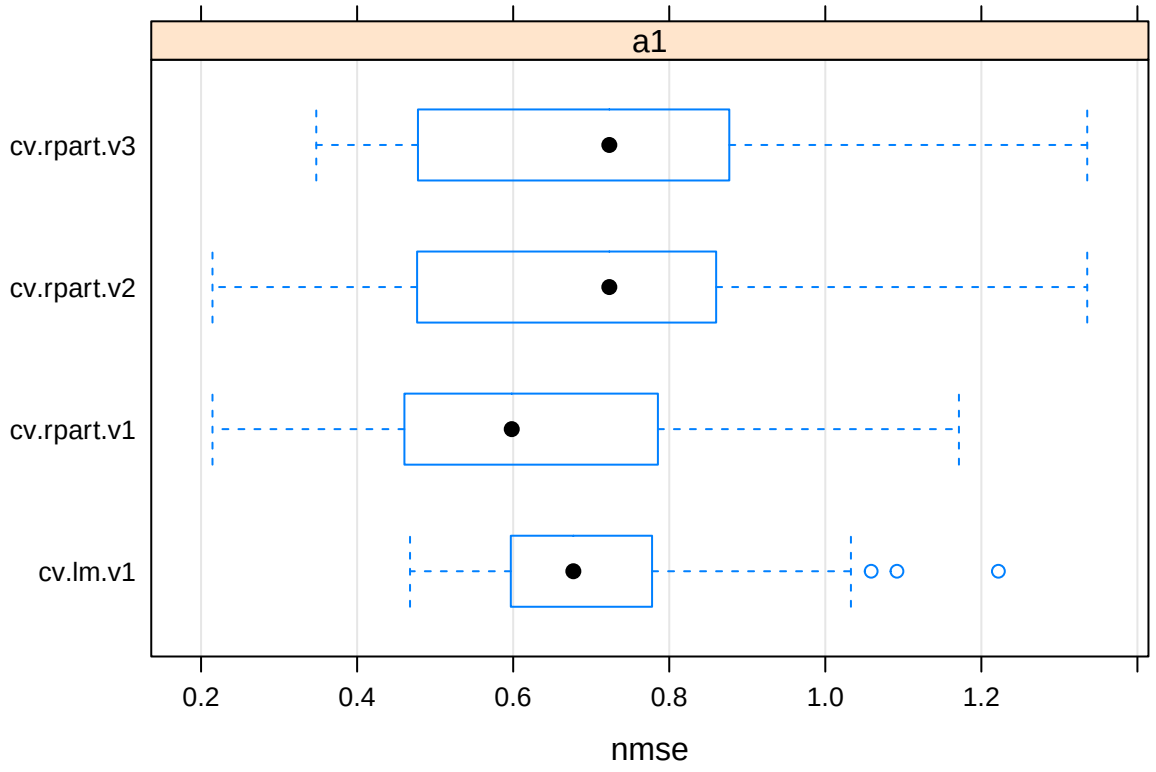
```
##
##
## #####  CROSS VALIDATION  EXPERIMENTAL COMPARISON #####
##
## ** DATASET :: a1
##
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  3 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
##
##  3 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
##  3 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  3 x 10 - Fold Cross Validation run with seed =  1234
```

```
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
```

```
summary(res)
```

```
##
## == Summary of a  Cross Validation  Experiment ==
##
##  3 x 10 - Fold Cross Validation run with seed =  1234
##
## * Data sets ::  a1
## * Learners  ::  cv.lm.v1, cv.rpart.v1, cv.rpart.v2, cv.rpart.v3
##
## * Summary of Experiment Results:
##
##
## -> Datataset:  a1
##
##  *Learner: cv.lm.v1
##           nmse
## avg     0.7196
## std     0.1833
## min     0.4678
## max     1.2218
## invalid 0.0000
##
##  *Learner: cv.rpart.v1
##           nmse
## avg     0.6441
## std     0.2522
## min     0.2146
## max     1.1713
## invalid 0.0000
##
##  *Learner: cv.rpart.v2
##           nmse
## avg     0.6874
## std     0.2670
## min     0.2146
## max     1.3357
## invalid 0.0000
##
##  *Learner: cv.rpart.v3
##           nmse
## avg     0.7167
## std     0.2579
## min     0.3476
## max     1.3357
## invalid 0.0000
```

```
plot(res)
```



If you want to get any model's parameter, use

```
getVariant("cv.rpart.v1", res)
```

```
##
## Learner::  "cv.rpart"
##
## Parameter values
##   se  =  0
```

For all comparisons within the seven prediction tasks, use

```
DSs <- sapply(names(clean.algae)[12:18],
              function(x,names.attrs){
                      f <- as.formula(paste(x, ' ~.'))
                      dataset(f, clean.algae[,c(names.attrs, x)], x)
              },
              names(clean.algae[1:11]))


res.all <- experimentalComparison(DSs,
                                  c(variants('cv.lm'),
                                    variants('cv.rpart', se=c(0,0.5,1))
                                    ),
                                  cvSettings(5, 10, 1234))
```

```
##
##
## #####  CROSS VALIDATION  EXPERIMENTAL COMPARISON #####
##
## ** DATASET :: a1
##
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
```

```
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ** DATASET :: a2
##
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
```

```
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ** DATASET :: a3
##
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
```

```
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ** DATASET :: a4
##
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
```

```
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ** DATASET :: a5
##
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
```

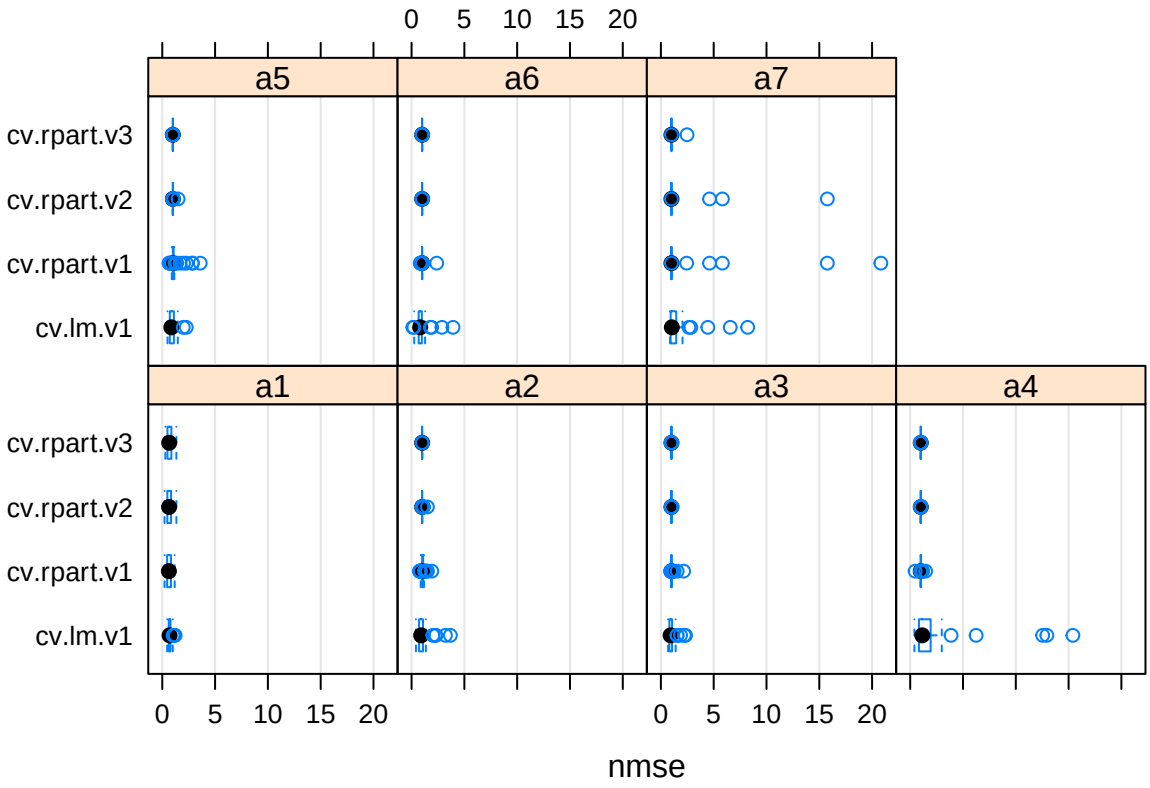
```
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ** DATASET :: a6
##
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
```

```
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ** DATASET :: a7
##
```

```
## ++ LEARNER :: cv.lm  variant ->  cv.lm.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
##
##
## ++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
##
##  5 x 10 - Fold Cross Validation run with seed =  1234
## Repetition  1
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  2
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  3
## Fold:  1  2  3  4  5  6  7  8  9  10
```

```
## Repetition  4
## Fold:  1  2  3  4  5  6  7  8  9  10
## Repetition  5
## Fold:  1  2  3  4  5  6  7  8  9  10
```

```
plot(res.all)
```




```
bestScores(res.all)
```

```
## $a1
##           system  score
## nmse cv.rpart.v1 0.6423
##
## $a2
##           system score
## nmse cv.rpart.v3     1
##
## $a3
##           system score
## nmse cv.rpart.v2     1
##
## $a4
##           system score
## nmse cv.rpart.v2     1
##
## $a5
##        system  score
## nmse cv.lm.v1 0.9317
```

```
##
## $a6
##        system score
## nmse cv.lm.v1 0.936
##
## $a7
##           system score
## nmse cv.rpart.v3  1.03
```

Use boosting to choose the best model:

```
library(randomForest)
cv.rf <- function(form, train, test, ...){
        m <- randomForest(form, train, ...)
        p <- predict(m, test)
        mse <- mean((p - resp(form, test))^2)
        c(nmse = mse/mean((mean(resp(form, train)) - resp(form, test))^2))
}

res.all <- experimentalComparison(
        DSs,
        c(variants('cv.lm'),
          variants('cv.rpart', se = c(0, 0.5, 1)),
          variants('cv.rf', ntree=c(200, 500, 700))
          ),
        cvSettings(5,10,1234))

bestScores(res.all)

compAnalysis(res.all, against='cv.rf.v3', datasets=c('a1','a2','a4','a6'))
```

To acquire all 7 models:

```
bestModelsNames <- sapply(bestScores(res.all),
                          function(x) x['nmse', 'system'])

learners <- c(rf = 'randomForest', rpart='rpartXse')

funcs <- learners[sapply(strsplit(bestModelsNames, '\\.'), function(x) x[2])]

parSetts <- lapply(bestModelsNames, function(x) getVariant(x, res.all) @pars)

bestModels <- list()

#for(a in 1:7){
#        form <- as.formula(paste(names(clean.algae)[11+a],'~.'))
#        bestModels[[a]] <- do.call(funcs[a],
#                                   c(list(form, clean.algae[, c(1:11,11+a)]), parSetts[[a]]))
#}
```