Group Assignment2: Algae Blooms

Aylin Kosar, Surya Aenuganti Ushasri, Salma Olmai, Nicolas Romero, Viraj Prasad Sapre

November 2, 2018

Understanding the data

```
library(DMwR)
```

```
## Loading required package: lattice

## Loading required package: grid
```

```
head(algae)
```

```
##   season  size  speed mxPH mnO2     Cl    NO3     NH4    oPO4     PO4 Chla
## 1 winter small medium 8.00  9.8 60.800  6.238 578.000 105.000 170.000 50.0
## 2 spring small medium 8.35  8.0 57.750  1.288 370.000 428.750 558.750  1.3
## 3 autumn small medium 8.10 11.4 40.020  5.330 346.667 125.667 187.057 15.6
## 4 spring small medium 8.07  4.8 77.364  2.302  98.182  61.182 138.700  1.4
## 5 autumn small medium 8.06  9.0 55.350 10.416 233.700  58.222  97.580 10.5
## 6 winter small   high 8.25 13.1 65.750  9.248 430.000  18.250  56.667 28.4
##     a1   a2   a3  a4   a5   a6  a7
## 1  0.0  0.0  0.0 0.0 34.2  8.3 0.0
## 2  1.4  7.6  4.8 1.9  6.7  0.0 2.1
## 3  3.3 53.6  1.9 0.0  0.0  0.0 9.7
## 4  3.1 41.0 18.9 0.0  1.4  0.0 1.4
## 5  9.2  2.9  7.5 0.0  7.5  4.1 1.0
## 6 15.1 14.6  1.4 0.0 22.5 12.6 2.9
```

```
#View(algae)
names(algae)
```

```
##  [1] "season" "size"   "speed"  "mxPH"   "mnO2"   "Cl"     "NO3"
##  [8] "NH4"    "oPO4"   "PO4"    "Chla"   "a1"     "a2"     "a3"
## [15] "a4"     "a5"     "a6"     "a7"
```

Algae data: The dataet contains 200 water samples. Each sample contains the aggregated data of chemicals collected in those particualr sample over a 3 month period.

First 3 columns are categorical. The next 8 are chemicals present in each of the sample. The next 7 columns are the harmful algae names which are named from a1 to a7.

Descriptive statistics

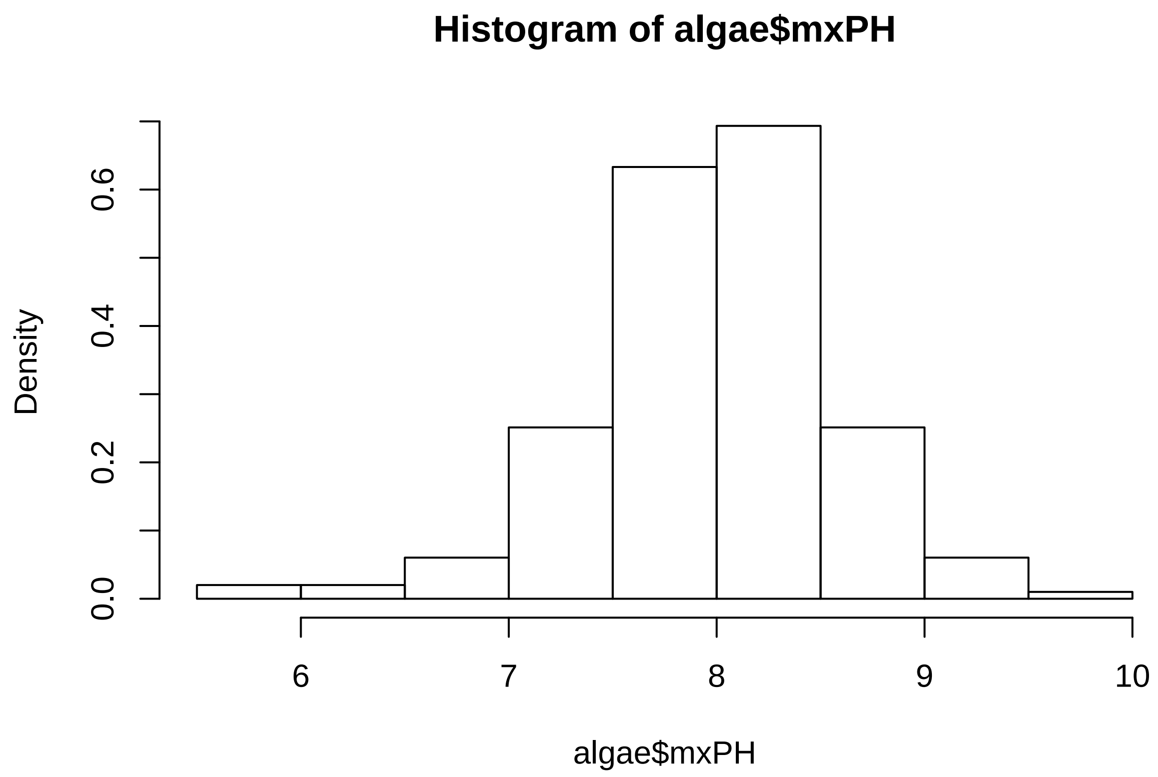
```
summary(algae)
```

```
##     season       size       speed         mxPH            mnO2
##  autumn:40   large :45   high  :84   Min.   :5.600   Min.   : 1.500
##  spring:53   medium:84   low   :33   1st Qu.:7.700   1st Qu.: 7.725
##  summer:45   small :71   medium:83   Median :8.060   Median : 9.800
##  winter:62                           Mean   :8.012   Mean   : 9.118
##                                      3rd Qu.:8.400   3rd Qu.:10.800
##                                      Max.   :9.700   Max.   :13.400
##                                      NA's   :1       NA's   :2
##        Cl               NO3               NH4               oPO4
##  Min.   :  0.222   Min.   : 0.050   Min.   :    5.00   Min.   :  1.00
##  1st Qu.: 10.981   1st Qu.: 1.296   1st Qu.:   38.33   1st Qu.: 15.70
##  Median : 32.730   Median : 2.675   Median :  103.17   Median : 40.15
##  Mean   : 43.636   Mean   : 3.282   Mean   :  501.30   Mean   : 73.59
##  3rd Qu.: 57.824   3rd Qu.: 4.446   3rd Qu.:  226.95   3rd Qu.: 99.33
##  Max.   :391.500   Max.   :45.650   Max.   :24064.00   Max.   :564.60
##  NA's   :10        NA's   :2        NA's   :2          NA's   :2
##       PO4               Chla                a1               a2
##  Min.   :  1.00   Min.   :  0.200   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 41.38   1st Qu.:  2.000   1st Qu.: 1.50   1st Qu.: 0.000
##  Median :103.29   Median :  5.475   Median : 6.95   Median : 3.000
##  Mean   :137.88   Mean   : 13.971   Mean   :16.92   Mean   : 7.458
##  3rd Qu.:213.75   3rd Qu.: 18.308   3rd Qu.:24.80   3rd Qu.:11.375
##  Max.   :771.60   Max.   :110.456   Max.   :89.80   Max.   :72.600
##  NA's   :2        NA's   :12
##        a3               a4               a5               a6
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 1.550   Median : 0.000   Median : 1.900   Median : 0.000
##  Mean   : 4.309   Mean   : 1.992   Mean   : 5.064   Mean   : 5.964
##  3rd Qu.: 4.925   3rd Qu.: 2.400   3rd Qu.: 7.500   3rd Qu.: 6.925
##  Max.   :42.800   Max.   :44.600   Max.   :44.400   Max.   :77.600
##
##        a7
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 1.000
##  Mean   : 2.495
##  3rd Qu.: 2.400
##  Max.   :31.600
##
```

There are more samples collected in winter than any other season

Checking the Normality

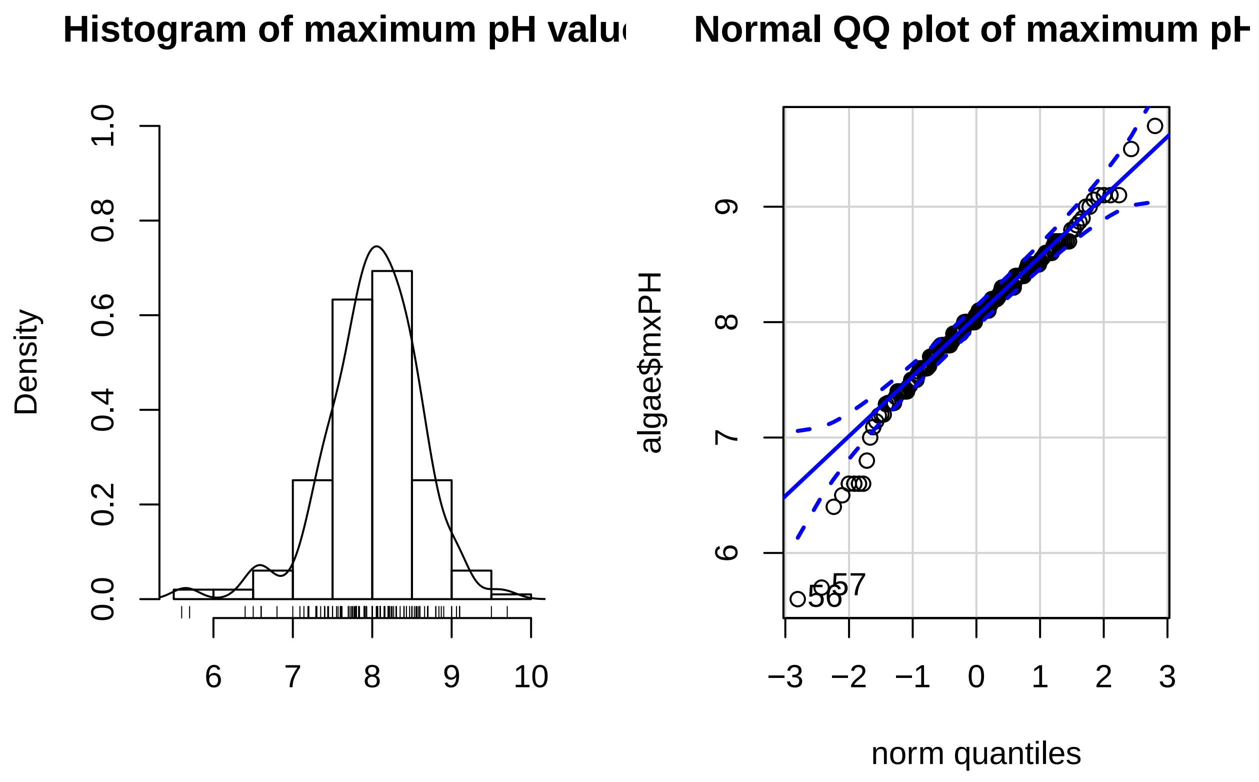
```
#Instead of frequency we get the probability densities
hist(algae$mxPH, prob = T)
```



```r
library(car)
```

```
## Loading required package: carData
```

```r
par(mfrow=c(1,2))
hist(algae$mxPH, prob=T, xlab='',
 main='Histogram of maximum pH value',ylim=0:1)
#adds a smooth kernel line over the distribution
lines(density(algae$mxPH,na.rm=T))
#rug() performs the plotting and jitter is used to randomly #perturb slightly the original values to pl
#so that we almost eliminate the possibility of two values being #equal, thus avoiding ticks
#over each other that would "hide" some values from the visual #inspection
rug(jitter(algae$mxPH))
qqPlot(algae$mxPH,main='Normal QQ plot of maximum pH')
```



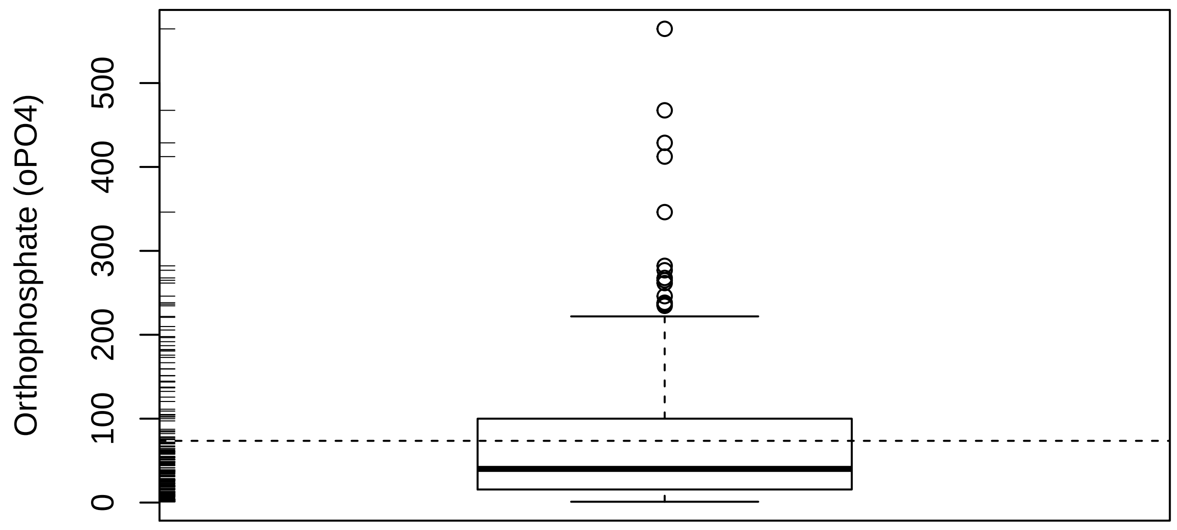
```
## [1] 56 57
```

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

First graph shows us that there are some 2 points lower than any other lying as outliers. By histogram we can assume that the data maximum pH is normal. But when we look at the same data in Normal QQ plot we see that within a 95% confidence interval we cannot say that the data is normal as there are many low values.

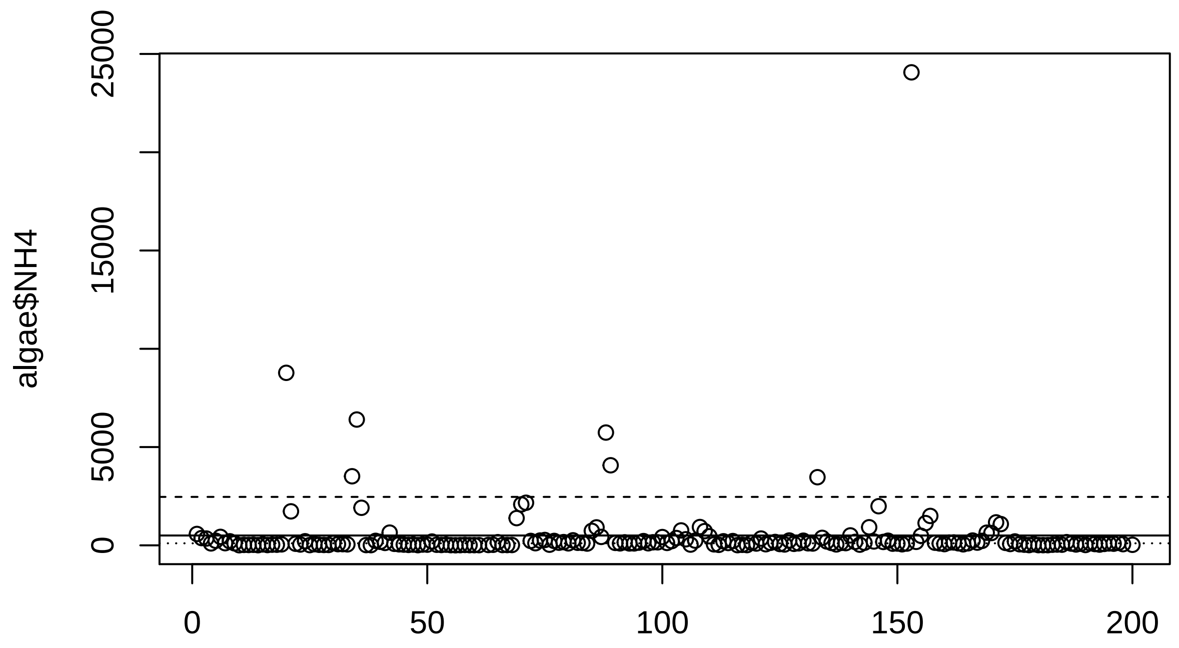
Checking the variable distribution

```
boxplot(algae$oPO4, ylab = "Orthophosphate (oPO4)")
rug(jitter(algae$oPO4), side = 2)
#draws the horizontal line at the mean
abline(h = mean(algae$oPO4, na.rm = T), lty = 2)
```



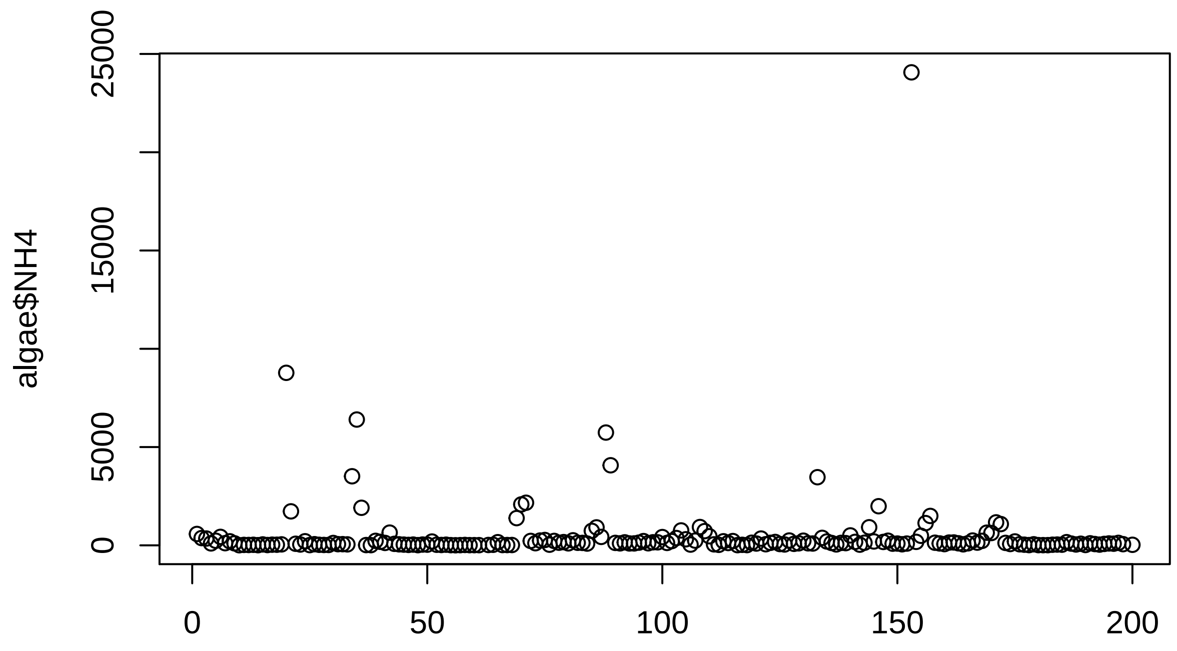
Box plot visualization orthophosphate(PO4) There are definitely outliers in the data which are of higher range. These outliers have influenced the mean and explains why is it bigger than median. But the distribution is concentrated on low values.

```
plot(algae$NH4, xlab = "")
#line at the mean
abline(h = mean(algae$NH4, na.rm = T), lty = 1)
#line at 1 std away from mean
abline(h = mean(algae$NH4, na.rm = T) + sd(algae$NH4, na.rm = T),lty=2)
#line at the median
abline(h = median(algae$NH4, na.rm = T), lty = 3)
#interactive function where clicking will diplay the data point
identify(algae$NH4)
```



```
## integer(0)
```

```
plot(algae$NH4, xlab = "")
clicked.lines <- identify(algae$NH4)
```



```
algae[clicked.lines, ]
```

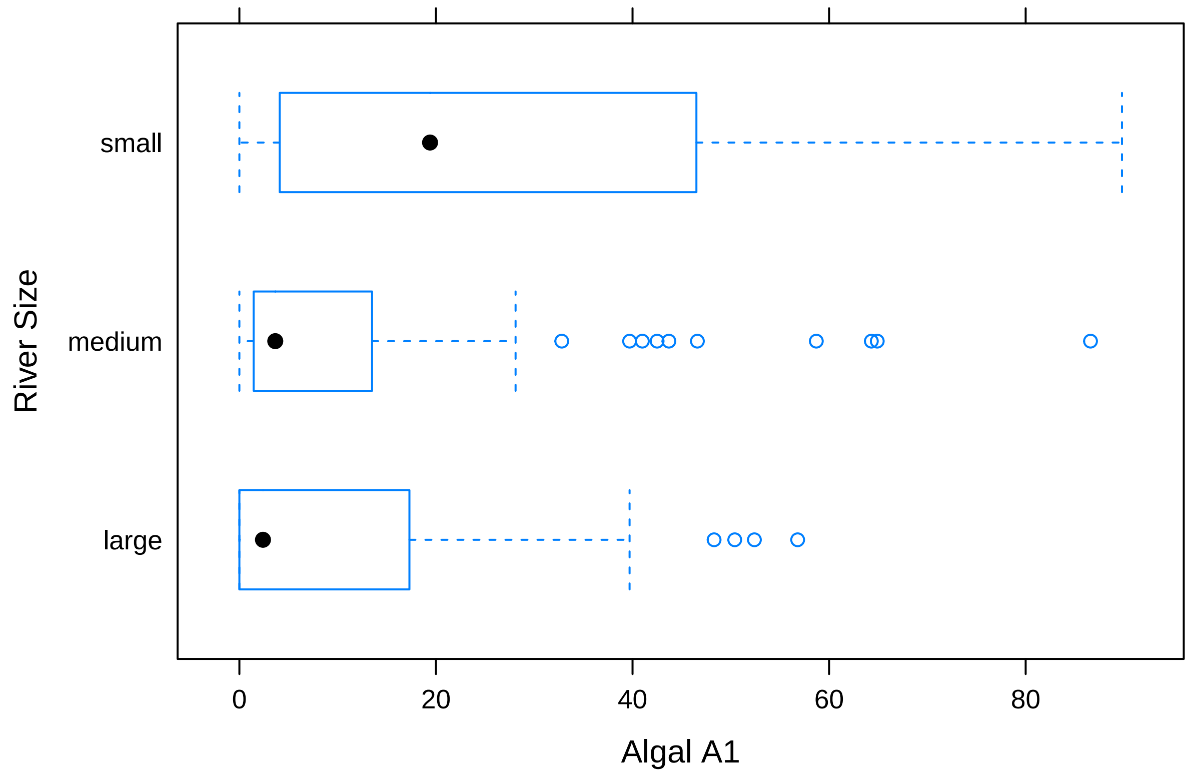
```
##  [1] season size   speed  mxPH   mnO2   Cl     NO3    NH4    oPO4   PO4
## [11] Chla   a1     a2     a3     a4     a5     a6     a7
## <0 rows> (or 0-length row.names)
```

```
algae[!is.na(algae$NH4) & algae$NH4 > 19000,]
```

```
##     season   size speed mxPH mnO2     Cl    NO3   NH4 oPO4 PO4 Chla  a1 a2
## 153 autumn medium  high  7.3 11.8 44.205 45.65 24064   44  34 53.1 2.2  0
##     a3  a4  a5   a6 a7
## 153  0 1.2 5.9 77.6  0
```

This gives out the rows which are known and greater than 19000

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

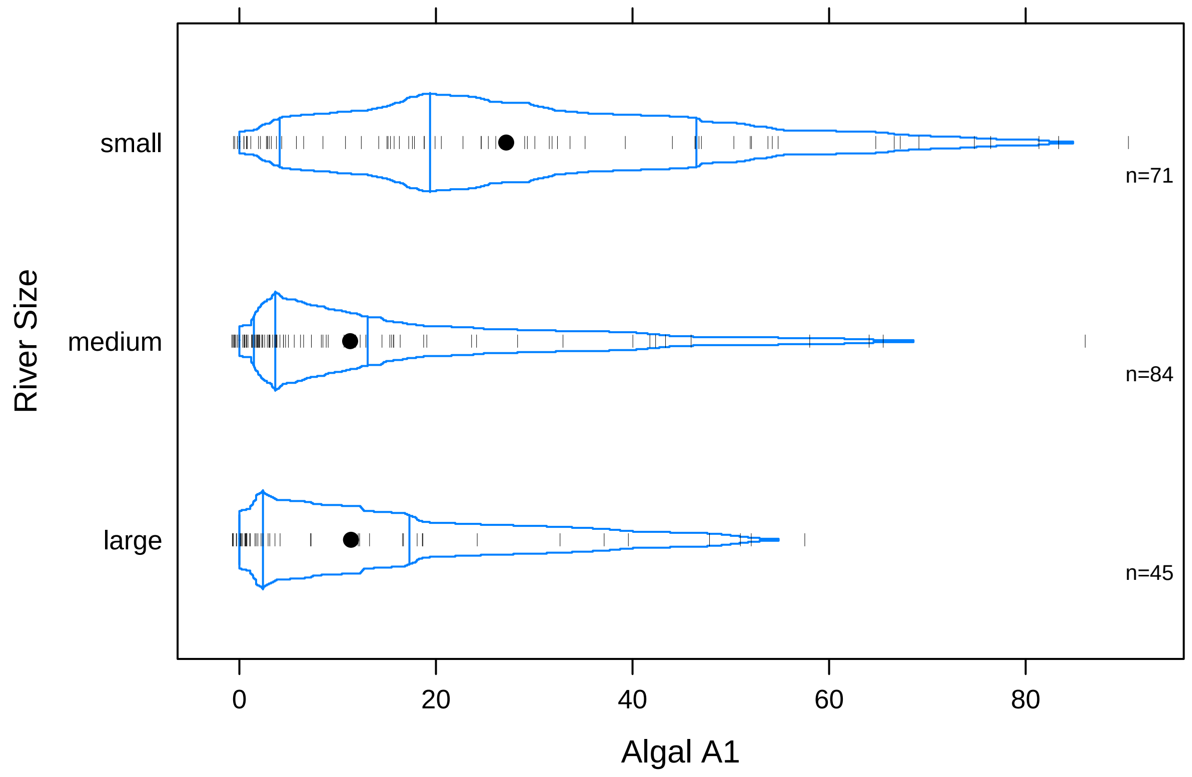


Higher frequencies of algal a1 are expected in smaller rivers.

```
#box percentile plots
library(Hmisc)
```

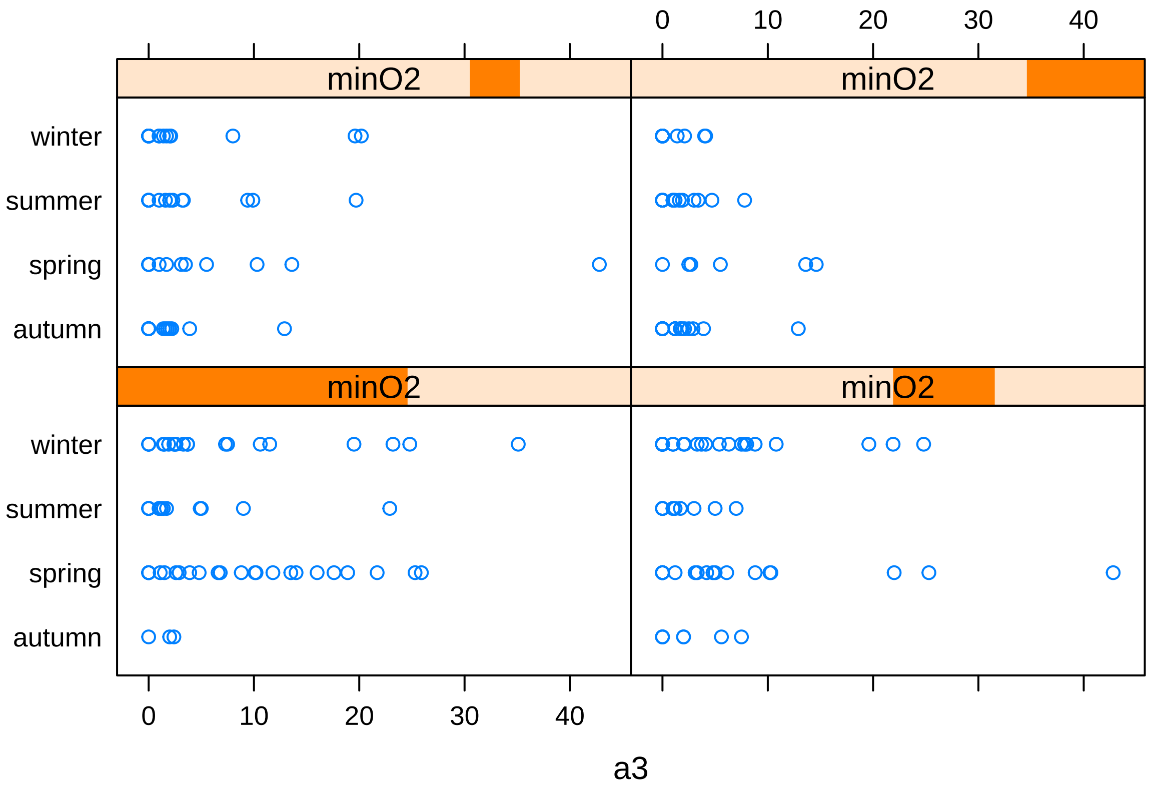
```
## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

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



The dots are the mean values. Here we can also see the spread of algae size according to the size of the rivers they are found in. Vertical lines are quartiles.

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



##Unknown values

1. Completely eliminating the rows consisting of NA's

```
data(algae)
#complete.cases() checks whether the complete observation is #clean of NA values and then outputing boo
nrow(algae[!complete.cases(algae),])
```

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

There are 16 rows which is not large enough to affect the model if we remove them

```
#Don't execute the below line because this is going to eliminate #the NA value present rows
#algae <- na.omit(algae)

#Don't execute the below code if you don't know which rows have #NA values in their columns
#algae <- algae[-c(62, 199), ]

#1 way to check the rows with NA values
#apply(algae, 1, function(x) sum(is.na(x)))

#preinstalled function which gives us the rows with 20% of #columns as NA
manyNAs(algae, 0.2)
```

```
## [1]  62 199
```

```
#eliminating the rows with default % of the columns as NA values
algae <- algae[-manyNAs(algae), ]
```

2. Filling in the Unknowns with the Most Frequent Values First check if the data is normally distributed

because if it is then we can use mean as a statistic to fill in the missing values. If the data is skewed or has outliers we can use median as a better statistic of centrality.

```
#Use the below if you know that row 48 has NA at mxPH
#We use mean as a centrality because we saw earlier that mxPH is #normally distributed
algae[48, "mxPH"] <- mean(algae$mxPH, na.rm = T)

#Varibale Chla is unknown for 12 samples
#distribution of Chla is skewed to lower values, and there are a #few extreme values that make the mean
algae[is.na(algae$Chla), "Chla"] <- median(algae$Chla, na.rm = T)
```

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

The above strategy is usually considered bad because it may create a large bias and can influence the analysis. But unbiased methods to fill in the unknowns are very complex.

3. Filling in the Unknown Values by Exploring Correlations

```
#use="complete.obs" ignores NA for calculating correlation
#Using from column 4 to 18 since 1-3 are nominal
#symnum visualizes the correlation matrix
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
```

From the above matrix we can see that PO4 and opO4 have correlation between 0.9 and 0.95. We can use this to fill in the unknown.

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

```
##
## Call:
## lm(formula = PO4 ~ oPO4, data = algae)
##
## Coefficients:
## (Intercept)         oPO4
##       42.897        1.293
```

The linear relation between these variables: PO4 = 42.897+1.293*oPO4

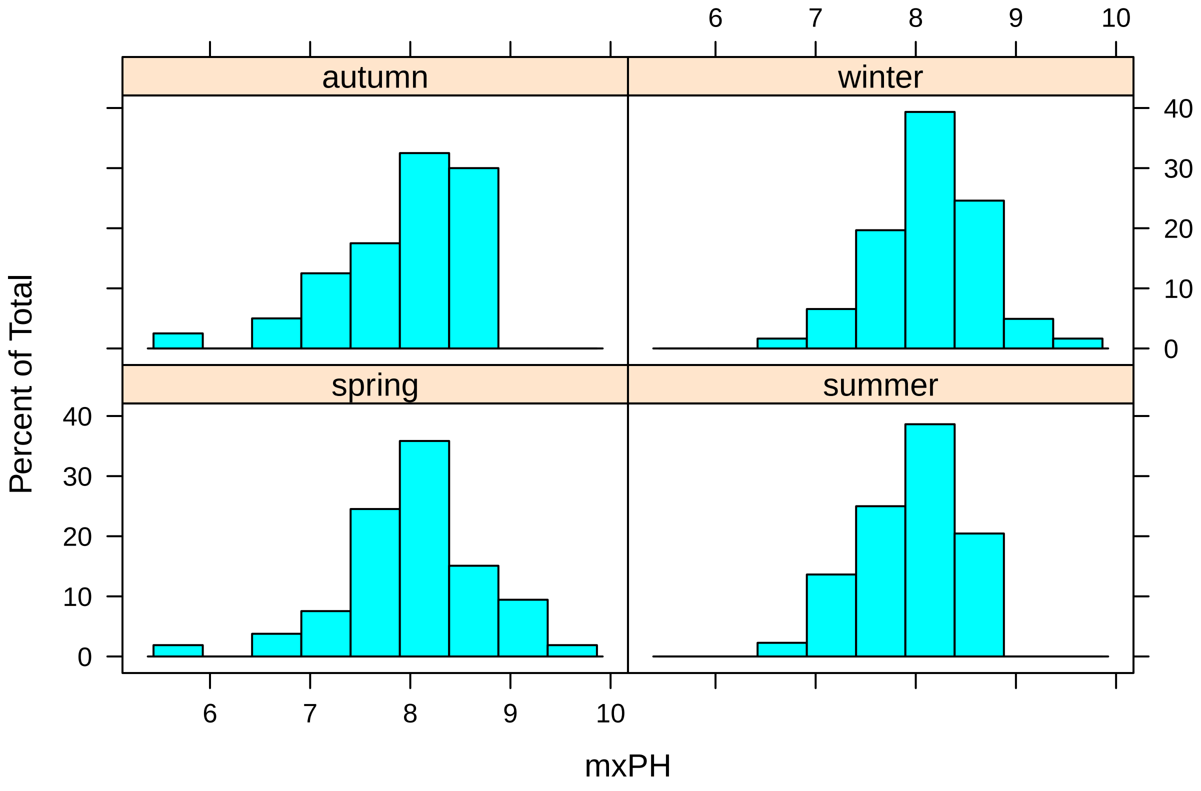
After removing the sample 62 and 199, we are left with a single observation with an unknown value on the variable PO4 (sample 28)

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

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

#Create the below function

#fillPO4 <- function(oP) {
# if(is.na(oP))
# return(NA)
# else return(42.897+1.293*oP)
# }

#Use sapply to apply this function

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

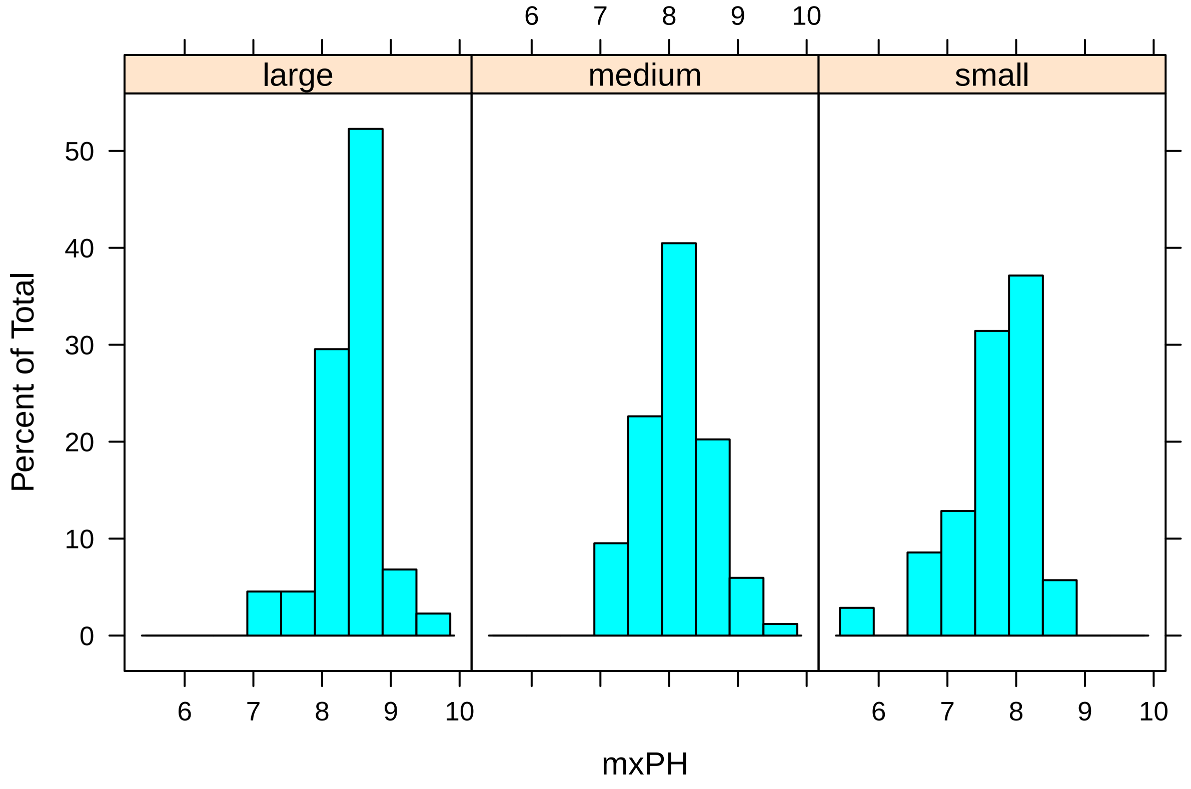
changing the order of factor levels on season according to their natural occurence

```
#By default when we factor the levels are assigned according to the alphabetical order
algae$season <- factor(algae$season, levels = c("spring",
"summer","autumn","winter"))
histogram(~mxPH | season, data = algae)
```

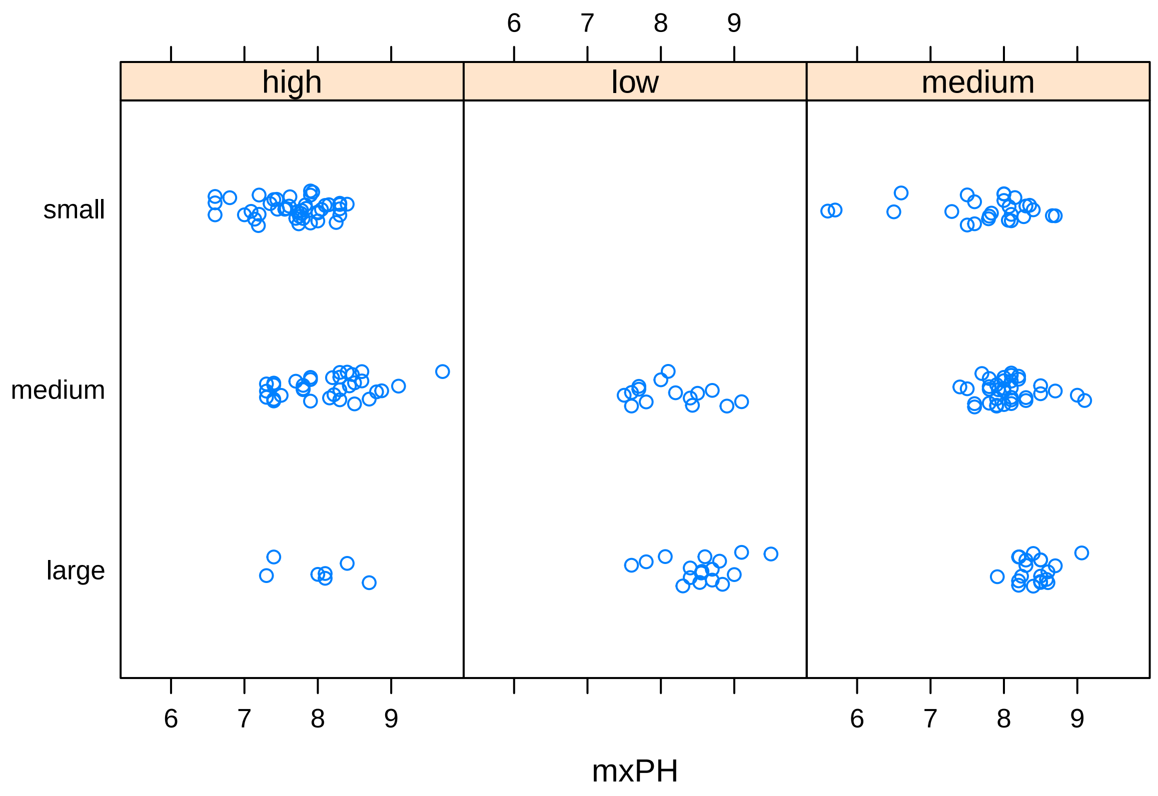


Looks like the mxPH are not influenced by the season they are collected in

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



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



This approach is too difficult if we have number of combinations to analyze and it makes sense to to use this method for data set with less variables.

4. Filling in the Unknown Values by Exploring Similarities between Cases

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

In this section assumes that if two water samples are similar, and one of them has an unknown value in some variable, there is a high probability that this value is similar to the value of the other sample.

```
#using knnimputation
algae <- knnImputation(algae, k = 10, meth = "median")
summary(algae)
```

```
##     season        size        speed         mxPH            mnO2
##  autumn:40   large :44   high  :84   Min.   :5.600   Min.   : 1.500
##  spring:53   medium:84   low   :33   1st Qu.:7.705   1st Qu.: 7.825
##  summer:44   small :70   medium:81   Median :8.060   Median : 9.800
##  winter:61                           Mean   :8.019   Mean   : 9.135
##                                      3rd Qu.:8.400   3rd Qu.:10.800
##                                      Max.   :9.700   Max.   :13.400
##        Cl               NO3              NH4               oPO4
##  Min.   :  0.222   Min.   : 0.050   Min.   :    5.00   Min.   :  1.00
##  1st Qu.: 10.425   1st Qu.: 1.296   1st Qu.:   38.33   1st Qu.: 15.70
##  Median : 32.178   Median : 2.675   Median :  103.17   Median : 40.15
##  Mean   : 42.434   Mean   : 3.282   Mean   :  501.30   Mean   : 73.59
##  3rd Qu.: 57.492   3rd Qu.: 4.446   3rd Qu.:  226.95   3rd Qu.: 99.33
##  Max.   :391.500   Max.   :45.650   Max.   :24064.00   Max.   :564.60
##       PO4              Chla               a1               a2
##  Min.   :  1.00   Min.   :  0.200   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 41.38   1st Qu.:  2.000   1st Qu.: 1.525   1st Qu.: 0.000
##  Median :103.29   Median :  5.155   Median : 6.950   Median : 3.000
##  Mean   :137.88   Mean   : 13.355   Mean   :16.996   Mean   : 7.471
##  3rd Qu.:213.75   3rd Qu.: 17.200   3rd Qu.:24.800   3rd Qu.:11.275
##  Max.   :771.60   Max.   :110.456   Max.   :89.800   Max.   :72.600
##        a3               a4               a5               a6
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 1.550   Median : 0.000   Median : 2.000   Median : 0.000
##  Mean   : 4.334   Mean   : 1.997   Mean   : 5.116   Mean   : 6.005
##  3rd Qu.: 4.975   3rd Qu.: 2.400   3rd Qu.: 7.500   3rd Qu.: 6.975
##  Max.   :42.800   Max.   :44.600   Max.   :44.400   Max.   :77.600
##        a7
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 1.000
##  Mean   : 2.487
##  3rd Qu.: 2.400
##  Max.   :31.600
```

In this case we used the median of k=10 nearest similar variables to fill in the unknowns after removing the 2 samples whose many of the variables were unknown first.

The NAs are removed and are stored in a new data frame **clean.algae**.

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

A linear regression model is then created in order to predict the frequency of one of the alages. Within the model, all the variables from the data are the predictor values, hence the dot.

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

Below a summary of the linear model is obtained. We see the intercept is 42.94 which means there is an increase of 42.94 the seven alage samples. For the factor season, there are three extra variables created: seasonsummer, seasonspring,and seasonwinter. If there is a water sample with the value "autumn" stored within the variable "season", all the other extra variables will be set to zero.The Adjusted R square helps explain the variance of the model and has the numerical value 0.32.Since it is a small value it is not considered great for the model. Looking at the p value for the F test, it has the numerical value 7.22.Since the value is small we can reject the null hypothesis that the independent value, a1, does not not depend on the predictor values of the model.

```
summary(lm.a1)
```

```
##
## Call:
## lm(formula = a1 ~ ., data = clean.algae[, 1:12])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.679 -11.893  -2.567   7.410  62.190
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  42.942055  24.010879   1.788  0.07537 .
## seasonspring  3.726978   4.137741   0.901  0.36892
## seasonsummer  0.747597   4.020711   0.186  0.85270
## seasonwinter  3.692955   3.865391   0.955  0.34065
## sizemedium    3.263728   3.802051   0.858  0.39179
## sizesmall     9.682140   4.179971   2.316  0.02166 *
## speedlow      3.922084   4.706315   0.833  0.40573
## speedmedium   0.246764   3.241874   0.076  0.93941
## mxPH         -3.589118   2.703528  -1.328  0.18598
## mnO2          1.052636   0.705018   1.493  0.13715
## Cl           -0.040172   0.033661  -1.193  0.23426
## NO3          -1.511235   0.551339  -2.741  0.00674 **
## NH4           0.001634   0.001003   1.628  0.10516
## oPO4         -0.005435   0.039884  -0.136  0.89177
## PO4          -0.052241   0.030755  -1.699  0.09109 .
## Chla         -0.088022   0.079998  -1.100  0.27265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.65 on 182 degrees of freedom
## Multiple R-squared:  0.3731, Adjusted R-squared:  0.3215
## F-statistic: 7.223 on 15 and 182 DF,  p-value: 2.444e-12
```

Below the ANOVA table shows the variance of each variable within the model **lm.a1**. The variable **season** is the one that contributes the least in reducing the fitting error of the model compared to the other variables in the model.

```
anova(lm.a1)
```

```
## Analysis of Variance Table
##
## Response: a1
##            Df Sum Sq Mean Sq F value    Pr(>F)
## season      3     85    28.2  0.0905 0.9651944
## size        2  11401  5700.7 18.3088  5.69e-08 ***
## speed       2   3934  1967.2  6.3179 0.0022244 **
## mxPH        1   1329  1328.8  4.2677 0.0402613 *
## mnO2        1   2287  2286.8  7.3444 0.0073705 **
## Cl          1   4304  4304.3 13.8239 0.0002671 ***
## NO3         1   3418  3418.5 10.9789 0.0011118 **
## NH4         1    404   403.6  1.2963 0.2563847
## oPO4        1   4788  4788.0 15.3774 0.0001246 ***
## PO4         1   1406  1405.6  4.5142 0.0349635 *
## Chla        1    377   377.0  1.2107 0.2726544
## Residuals 182  56668   311.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The **update** function is used to update the model by removing season.

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

There is a bit of an increase of the intercept with the value of 44.95. The model's fit if we look at the Adjusted R square has imporved slightly with the value of 0.33.

```r
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
## -36.460 -11.953  -3.044   7.444  63.730
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 44.9532874 23.2378377   1.934  0.05458 .
## sizemedium   3.3092102  3.7825221   0.875  0.38278
## sizesmall   10.2730961  4.1223163   2.492  0.01358 *
## speedlow     3.0546270  4.6108069   0.662  0.50848
## speedmedium -0.2976867  3.1818585  -0.094  0.92556
## mxPH        -3.2684281  2.6576592  -1.230  0.22033
## mnO2         0.8011759  0.6589644   1.216  0.22561
## Cl          -0.0381881  0.0333791  -1.144  0.25407
## NO3         -1.5334300  0.5476550  -2.800  0.00565 **
## NH4          0.0015777  0.0009951   1.586  0.11456
## oPO4        -0.0062392  0.0395086  -0.158  0.87469
## PO4         -0.0509543  0.0305189  -1.670  0.09669 .
## Chla        -0.0841371  0.0794459  -1.059  0.29096
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.57 on 185 degrees of freedom
## Multiple R-squared:  0.3682, Adjusted R-squared:  0.3272
## F-statistic: 8.984 on 12 and 185 DF,  p-value: 1.762e-13
```

ANOVA is used again however this time to compare the two models lm.a1 and lm2.a1. The sum of squares have decreased by -448 and are not significant.

```r
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   -447.62 0.4792 0.6971
```

The backward elimination is used for the following model. The function **step** is used for model search by the Akaike Information Criterion.

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

```
## Start:  AIC=1152.03
## a1 ~ season + size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 +
##     PO4 + Chla
```

```
##
##          Df Sum of Sq   RSS    AIC
## - season  3    447.62 57116 1147.6
## - speed   2    269.60 56938 1149.0
## - oPO4    1      5.78 56674 1150.0
## - Chla    1    376.96 57045 1151.3
## - Cl      1    443.46 57112 1151.6
## - mxPH    1    548.76 57217 1151.9
## <none>                56668 1152.0
## - mnO2    1    694.11 57363 1152.4
## - NH4     1    825.67 57494 1152.9
## - PO4     1    898.42 57567 1153.1
## - size    2   1857.16 58526 1154.4
## - NO3     1   2339.36 59008 1158.0
##
## Step:  AIC=1147.59
## a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 + PO4 +
##     Chla
##
##         Df Sum of Sq   RSS    AIC
## - speed  2    210.64 57327 1144.3
## - oPO4   1      7.70 57124 1145.6
## - Chla   1    346.27 57462 1146.8
## - Cl     1    404.10 57520 1147.0
## - mnO2   1    456.37 57572 1147.2
## - mxPH   1    466.95 57583 1147.2
## <none>               57116 1147.6
## - NH4    1    776.11 57892 1148.3
## - PO4    1    860.62 57977 1148.5
## - size   2   2175.59 59292 1151.0
## - NO3    1   2420.47 59537 1153.8
##
## Step:  AIC=1144.31
## a1 ~ size + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 + PO4 + Chla
##
##        Df Sum of Sq   RSS    AIC
## - oPO4  1     16.29 57343 1142.4
## - Chla  1    223.29 57550 1143.1
## - mnO2  1    413.77 57740 1143.7
## - Cl    1    472.70 57799 1143.9
## - mxPH  1    483.56 57810 1144.0
## <none>              57327 1144.3
## - NH4   1    720.19 58047 1144.8
## - PO4   1    809.30 58136 1145.1
## - size  2   2060.95 59388 1147.3
## - NO3   1   2379.75 59706 1150.4
##
## Step:  AIC=1142.37
## a1 ~ size + mxPH + mnO2 + Cl + NO3 + NH4 + PO4 + Chla
##
##        Df Sum of Sq   RSS    AIC
## - Chla  1     207.7 57551 1141.1
## - mnO2  1     402.6 57746 1141.8
## - Cl    1     470.7 57814 1142.0
## - mxPH  1     519.7 57863 1142.2
## <none>              57343 1142.4
## - NH4   1     704.4 58047 1142.8
## - size  2    2050.3 59393 1145.3
## - NO3   1    2370.4 59713 1148.4
## - PO4   1    5818.4 63161 1159.5
##
## Step:  AIC=1141.09
## a1 ~ size + mxPH + mnO2 + Cl + NO3 + NH4 + PO4
##
##        Df Sum of Sq   RSS    AIC
## - mnO2  1     435.3 57986 1140.6
## - Cl    1     438.1 57989 1140.6
## <none>              57551 1141.1
## - NH4   1     746.9 58298 1141.6
## - mxPH  1     833.1 58384 1141.9
## - size  2    2217.5 59768 1144.6
## - NO3   1    2667.1 60218 1148.1
## - PO4   1    6309.7 63860 1159.7
##
## Step:  AIC=1140.58
## a1 ~ size + mxPH + Cl + NO3 + NH4 + PO4
##
##        Df Sum of Sq   RSS    AIC
## - NH4   1     531.0 58517 1140.4
## - Cl    1     584.9 58571 1140.6
## <none>              57986 1140.6
## - mxPH  1     819.1 58805 1141.4
## - size  2    2478.2 60464 1144.9
## - NO3   1    2251.4 60237 1146.1
## - PO4   1    9097.9 67084 1167.4
##
## Step:  AIC=1140.38
## a1 ~ size + mxPH + Cl + NO3 + PO4
##
##        Df Sum of Sq   RSS    AIC
## <none>              58517 1140.4
## - mxPH  1     784.1 59301 1141.0
## - Cl    1     835.6 59353 1141.2
## - NO3   1    1987.9 60505 1145.0
## - size  2    2664.3 61181 1145.2
## - PO4   1    8575.8 67093 1165.5
```

The Adjusted R square still remains a small value.

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

```
##
## Call:
## lm(formula = a1 ~ size + mxPH + Cl + NO3 + PO4, data = clean.algae[,
##     1:12])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.874 -12.732  -3.741   8.424  62.926
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 57.28555   20.96132   2.733  0.00687 **
## sizemedium   2.80050    3.40190   0.823  0.41141
## sizesmall   10.40636    3.82243   2.722  0.00708 **
## mxPH        -3.97076    2.48204  -1.600  0.11130
## Cl          -0.05227    0.03165  -1.651  0.10028
## NO3         -0.89529    0.35148  -2.547  0.01165 *
## PO4         -0.05911    0.01117  -5.291 3.32e-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:  0.3527, Adjusted R-squared:  0.3324
## F-statistic: 17.35 on 6 and 191 DF,  p-value: 5.554e-16
```

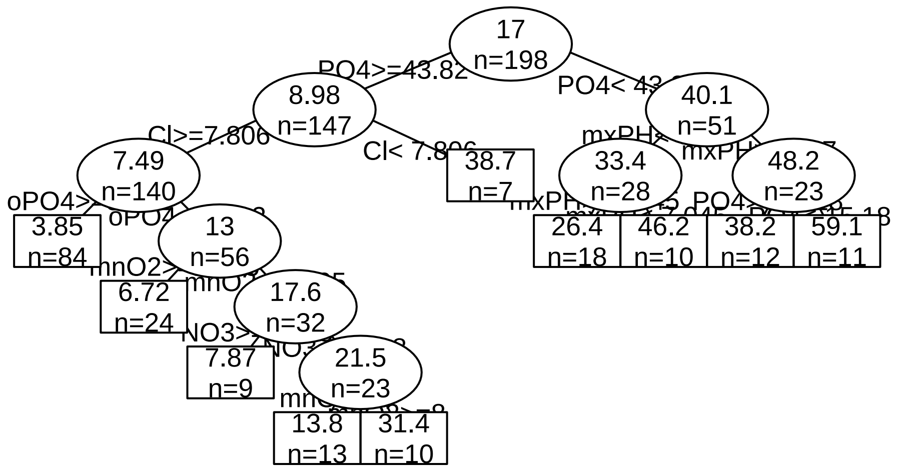
The following steps will help create a regression tree in order to predict the value of the frequencies of algae a1. Below the process of removing NAs and samples 62 and 199 are repeated again. The library **rpart** helps creates regression trees in R. There are 198 samples that were obtained to create the tree.

```r
library(rpart)

 data(algae)

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

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

 rt.a1
```

```
## n= 198
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 198 90401.290 16.996460
##    2) PO4>=43.818 147 31279.120  8.979592
##      4) Cl>=7.8065 140 21622.830  7.492857
##        8) oPO4>=51.118 84  3441.149  3.846429 *
##        9) oPO4< 51.118 56 15389.430 12.962500
##         18) mnO2>=10.05 24  1248.673  6.716667 *
##         19) mnO2< 10.05 32 12502.320 17.646870
##           38) NO3>=3.1875 9   257.080  7.866667 *
##           39) NO3< 3.1875 23 11047.500 21.473910
##             78) mnO2< 8 13  2919.549 13.807690 *
##             79) mnO2>=8 10  6370.704 31.440000 *
##      5) Cl< 7.8065 7  3157.769 38.714290 *
##    3) PO4< 43.818 51 22442.760 40.103920
##      6) mxPH< 7.87 28 11452.770 33.450000
##       12) mxPH>=7.045 18  5146.169 26.394440 *
##       13) mxPH< 7.045 10  3797.645 46.150000 *
##      7) mxPH>=7.87 23  8241.110 48.204350
##       14) PO4>=15.177 12  3047.517 38.183330 *
##       15) PO4< 15.177 11  2673.945 59.136360 *
```

Below a graphical image of the tree is created.

```r
prettyTree(rt.a1)
```



A subset of subtress for the tree are created using the function **printcp**.

```r
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 = 456.57
##
## n= 198
##
##          CP nsplit rel error  xerror    xstd
## 1 0.405740      0   1.00000 1.00851 0.13102
## 2 0.071885      1   0.59426 0.70735 0.12042
## 3 0.030887      2   0.52237 0.65556 0.11420
## 4 0.030408      3   0.49149 0.66801 0.11630
## 5 0.027872      4   0.46108 0.66510 0.11701
## 6 0.027754      5   0.43321 0.68438 0.11530
## 7 0.018124      6   0.40545 0.63819 0.11174
## 8 0.016344      7   0.38733 0.64161 0.10501
## 9 0.010000      9   0.35464 0.66702 0.11282
```

A tree is obtained by using th cp value of 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 90401.29 16.996460
##   2) PO4>=43.818 147 31279.12  8.979592 *
##   3) PO4< 43.818 51 22442.76 40.103920 *
```

Below the function **rpartXse** is used to split the tree and obtain a subtree.

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

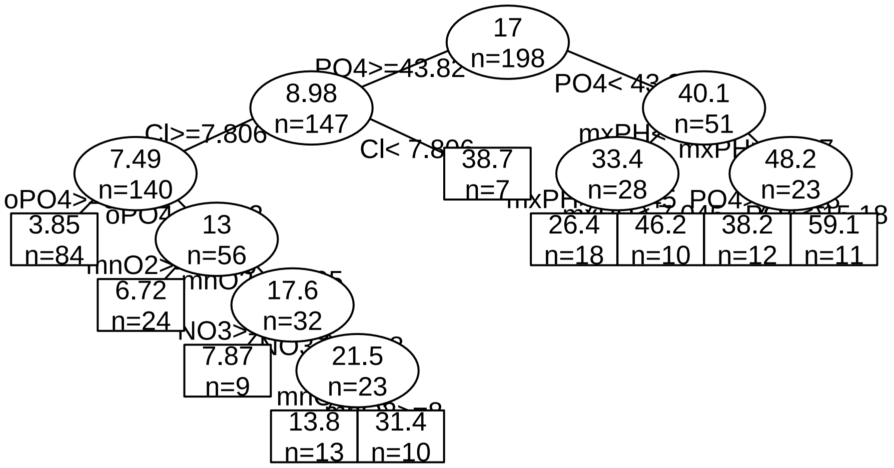
```
## n= 198
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 198 90401.290 16.996460
##   2) PO4>=43.818 147 31279.120  8.979592
##     4) Cl>=7.1665 142 21763.160  7.530282 *
##     5) Cl< 7.1665 5   746.792 50.140000 *
##   3) PO4< 43.818 51 22442.760 40.103920 *
```

Below a pruned tree is obtained by using the function **snip.part**. Using this function, a pruned tree can be generated by indicating the number of nodes you would like to prune the tree or use it in a graphical way by plotting the tree and plot it without calling the function with a second argument.

```
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 90401.290 16.996460
##    2) PO4>=43.818 147 31279.120  8.979592
##      4) Cl>=7.8065 140 21622.830  7.492857 *
##      5) Cl< 7.8065 7  3157.769 38.714290 *
##    3) PO4< 43.818 51 22442.760 40.103920
##      6) mxPH< 7.87 28 11452.770 33.450000
##       12) mxPH>=7.045 18  5146.169 26.394440 *
##       13) mxPH< 7.045 10  3797.645 46.150000 *
##      7) mxPH>=7.87 23  8241.110 48.204350 *
```

```
prettyTree(first.tree)

snip.rpart(first.tree)
```



```
## n= 198
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 198 90401.290 16.996460
##    2) PO4>=43.818 147 31279.120  8.979592
##      4) Cl>=7.8065 140 21622.830  7.492857
##        8) oPO4>=51.118 84  3441.149  3.846429 *
##        9) oPO4< 51.118 56 15389.430 12.962500
##         18) mnO2>=10.05 24  1248.673  6.716667 *
##         19) mnO2< 10.05 32 12502.320 17.646870
##           38) NO3>=3.1875 9   257.080  7.866667 *
##           39) NO3< 3.1875 23 11047.500 21.473910
##             78) mnO2< 8 13  2919.549 13.807690 *
##             79) mnO2>=8 10  6370.704 31.440000 *
##      5) Cl< 7.8065 7  3157.769 38.714290 *
##    3) PO4< 43.818 51 22442.760 40.103920
##      6) mxPH< 7.87 28 11452.770 33.450000
##       12) mxPH>=7.045 18  5146.169 26.394440 *
##       13) mxPH< 7.045 10  3797.645 46.150000 *
##      7) mxPH>=7.87 23  8241.110 48.204350
##       14) PO4>=15.177 12  3047.517 38.183330 *
##       15) PO4< 15.177 11  2673.945 59.136360 *
```

Below the predictions for the model are created for two seperate models in order to figure out their mean absolute error (MAE).

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

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

The mean absolute error is calcualted below for the models.

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

```
## [1] 13.10681
```

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

```
## [1] 10.36242
```

Then below we have the mean sqaured error (MSE) that gets calculated.

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

```
## [1] 295.5407
```

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

```
## [1] 227.0339
```

Below the Normalized Mean Squared Error is calculated in order see if the scores gathered from the models are good or bad.

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

```
## [1] 0.6473034
```

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

```
## [1] 0.4972574
```

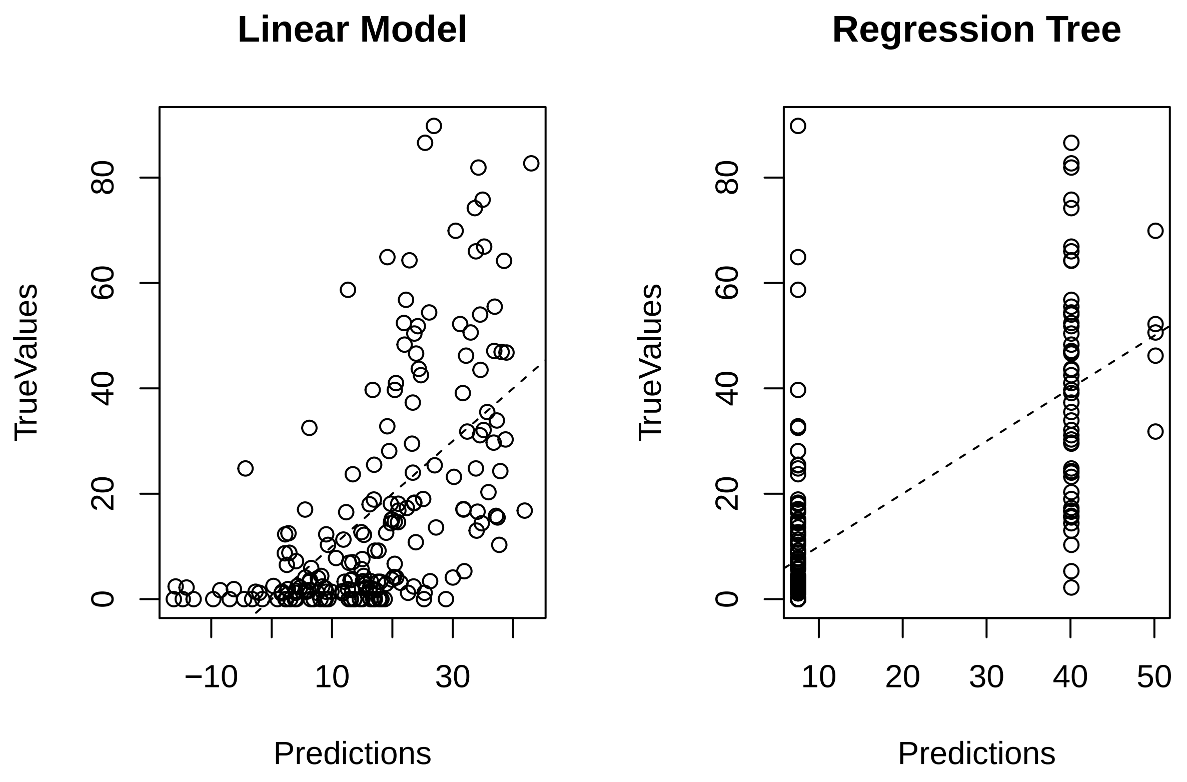
Below the function **regr.eval** is used to calculate the value of a set of regression evaluation metrics.

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

```
##         mae         mse        rmse        mape        nmse        nmae
##  10.3624227 227.0338940  15.0676439         Inf   0.4972574   0.6202654
```

Below the predictions are plotted for the mdoels using a scatterplot of the errors.

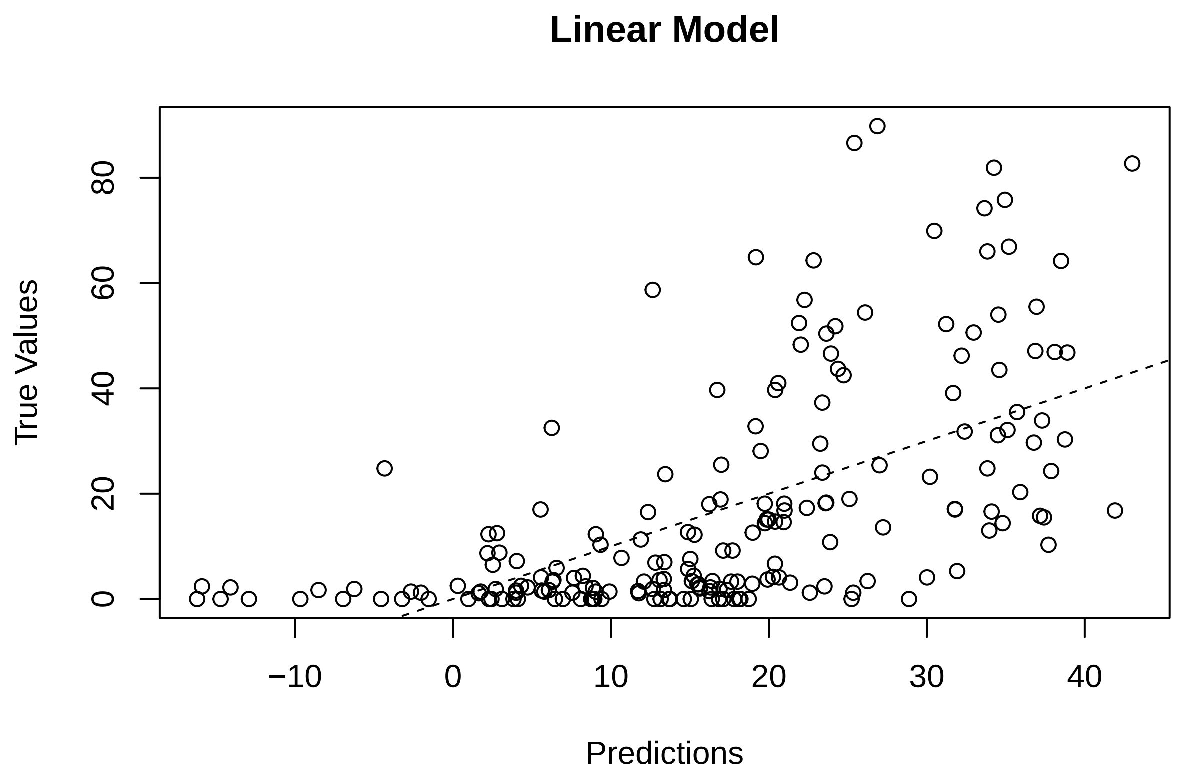
```r
old.par <- par(mfrow = c(1, 2))

plot(lm.predictions.a1, algae[, "a1"], main = "Linear Model",
xlab="Predictions",ylab="TrueValues")
abline(0, 1, lty = 2)

plot(rt.predictions.a1, algae[, "a1"], main = "Regression Tree",
xlab="Predictions",ylab="TrueValues")
abline(0, 1, lty = 2)
```



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

Below the sample number that provides the bad prediction is obtained by using the function **identify**.

```
plot(lm.predictions.a1,algae[,'a1'],main="Linear Model",
xlab="Predictions",ylab="True Values")
abline(0,1,lty=2)

algae[identify(lm.predictions.a1,algae[,'a1']),]
```



```
##  [1] season size   speed  mxPH   mnO2   Cl     NO3    NH4    oPO4   PO4
## [11] Chla   a1     a2     a3     a4     a5     a6     a7
## <0 rows> (or 0-length row.names)
```

The performance of the model gets improved by add an if else statement.

```
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
##  13.10681 295.54069
```

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

```
##       mae       mse
##  12.48276 286.28541
```

Below the two models are being prepared for cross validation.

```
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))

 }
```

The cross validation comparison of the two models are created below.

```
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
```

Below a summary of the cross validation results are created.

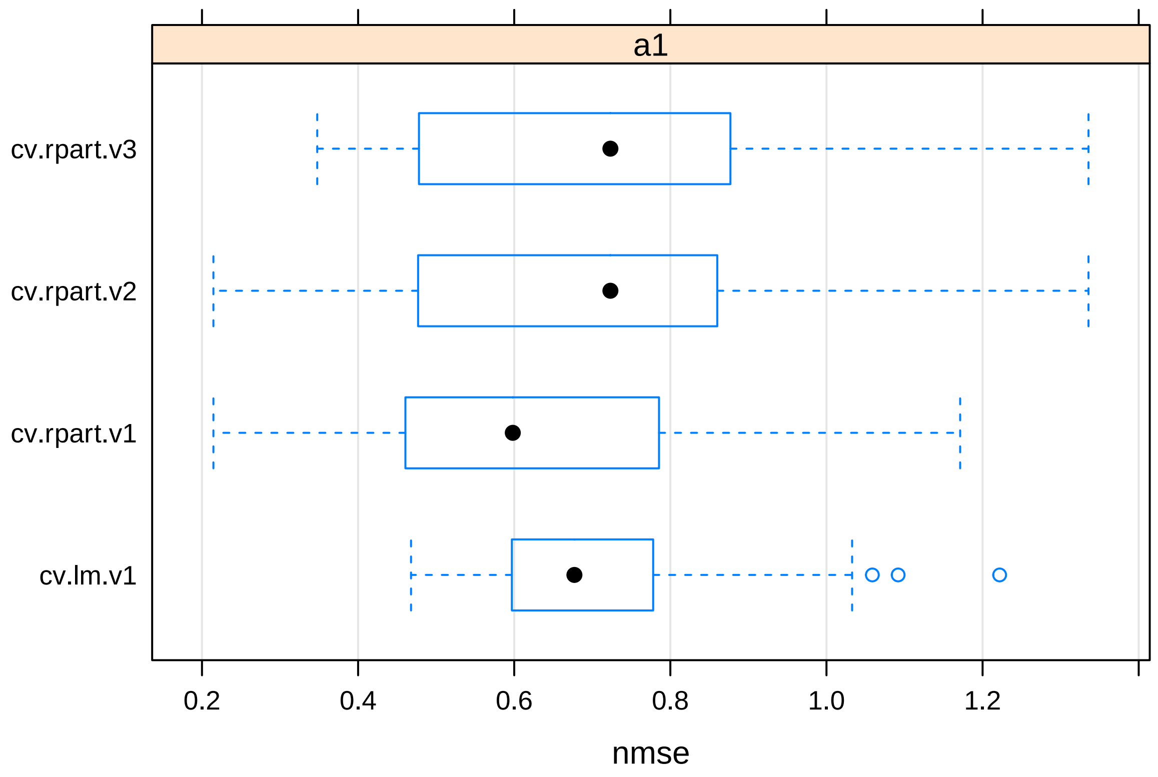
```
summary(res)
```

```
##
## == 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     0.7196105
## std     0.1833064
## min     0.4678248
## max     1.2218455
## invalid 0.0000000
##
##  *Learner: cv.rpart.v1
##              nmse
## avg     0.6440843
## std     0.2521952
## min     0.2146359
## max     1.1712674
## invalid 0.0000000
```

```
##
##  *Learner: cv.rpart.v2
##                nmse
## avg     0.6873747
## std     0.2669942
## min     0.2146359
## max     1.3356744
## invalid 0.0000000
##
##  *Learner: cv.rpart.v3
##                nmse
## avg     0.7167122
## std     0.2579089
## min     0.3476446
## max     1.3356744
## invalid 0.0000000
```

The plot of the cross validation results is created. cv.lm.v1 contains residuals. All plots look more right skewed.

```r
plot(res)
```



Below the specific parameter settings corresponding to a specifc label is checked.

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

```
##
## Learner::  "cv.rpart"
```

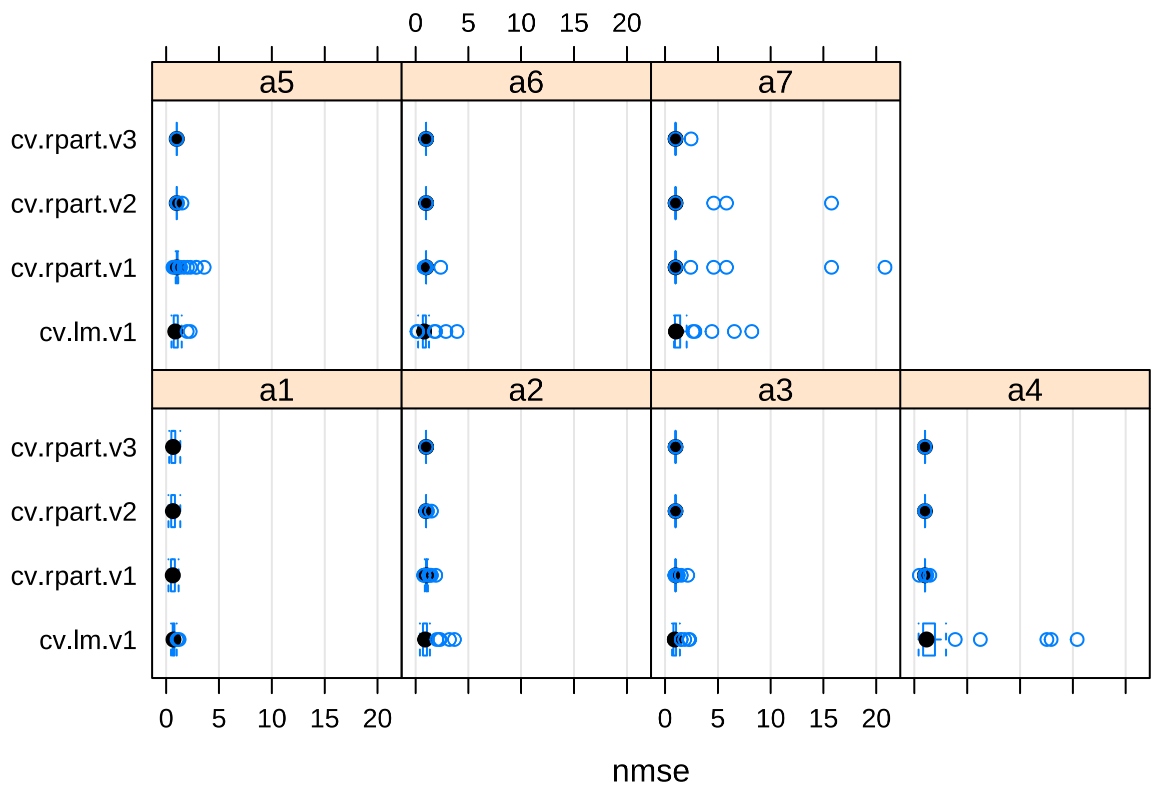
```
##
## Parameter values
##   se  =  0
```

Below we creater vectors of datasets which are useful for comparison of 7 different predicitive tasks. We create a formula for the comparison to be carried out by using as.formula function and we include all the attributes for the comparison. Then, we use the experimentalComparison function where we can compare the two models and try to find out the best results from them.

```
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
```

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



From this plot, we can observe that algae a1 has the least NMSE. So, We can say that the highest predictive accuracy is seen in predicting algae a1.

```
bestScores(res.all) #this function is used to display the best results obtained from comparing the norm
```

```
## $a1
##           system   score
## nmse cv.rpart.v1 0.64231
##
## $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 0.9316803
##
## $a6
##        system     score
## nmse cv.lm.v1 0.9359697
##
## $a7
##           system    score
## nmse cv.rpart.v3 1.029505
```

Random Forests are similar to a famous Ensemble technique called Bagging but have a different tweak in it. In Random Forests the idea is to decorrelate the several trees which are generated by the different bootstrapped samples from training Data.

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
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))
}
# As above, we applied random forest cross-validation and create training and test sets for the dataset
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)) #This class of objects contains the information describing a cross validatio
```

```
##
##
## #####  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
##
##
## ++ LEARNER :: cv.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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
##
##
## ++ LEARNER :: cv.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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
##
##
## ++ LEARNER :: cv.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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
##
##
## ++ LEARNER :: cv.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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
##
##
## ++ LEARNER :: cv.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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
##
##
## ++ LEARNER :: cv.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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
##
##
## ++ LEARNER :: cv.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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.rf  variant ->  cv.rf.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
```

```
# After implementing random forest cross validation we use the experimentalComparison function to coarr
bestScores(res.all)
```

```
## $a1
##        system     score
## nmse cv.rf.v3 0.5467636
##
## $a2
##        system     score
## nmse cv.rf.v3 0.7695782
##
## $a3
##           system score
## nmse cv.rpart.v2     1
##
## $a4
##        system     score
## nmse cv.rf.v1 0.9728596
##
## $a5
##        system     score
## nmse cv.rf.v2 0.7916332
##
## $a6
##        system    score
## nmse cv.rf.v2 0.911758
##
## $a7
##           system    score
## nmse cv.rpart.v3 1.029505
```

So from the above bestScores function we can have the bestScores from the "cv.rf.v3","cv.rf.v2","cv.rf.v1" and "cv.rpart.v3" tasks.

compAnalysis() function: When you run the experimentalComparison() function to compare a set of learners over a set of problems you obtain estimates of their performances across these problems. This function allows you to test whether the observed differences in these estimated performances are statistically significant with a certain confidence level.

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

```
##
## == Statistical Significance Analysis of Comparison Results ==
##
## Baseline Learner::    cv.rf.v3  (Learn.1)
##
## ** Evaluation Metric::    nmse
##
## - Dataset: a1
##       Learn.1   Learn.2 sig.2   Learn.3 sig.3   Learn.4 sig.4   Learn.5
## AVG 0.5467636 0.7077282    ++ 0.6423100    +  0.6569726    ++ 0.6875212
## STD 0.1727235 0.1639373       0.2399321       0.2397636       0.2348946
##     sig.5   Learn.6 sig.6   Learn.7 sig.7
## AVG    ++ 0.5505008       0.5473338
## STD       0.1783960       0.1724374
##
## - Dataset: a2
##       Learn.1   Learn.2 sig.2   Learn.3 sig.3    Learn.4 sig.4
## AVG 0.7695782 1.0449317    ++ 1.0426327    ++ 1.01626123    ++
## STD 0.1431761 0.6276144       0.2005522       0.07435826
##          Learn.5 sig.5   Learn.6 sig.6   Learn.7 sig.7
## AVG 1.000000e+00    ++ 0.7775628       0.7744307
## STD 2.389599e-16       0.1473327       0.1462083
##
## - Dataset: a4
##       Learn.1  Learn.2 sig.2   Learn.3 sig.3      Learn.4 sig.4
## AVG 0.9746980 2.111976       1.0073953    +  1.000000e+00    +
## STD 0.3823094 3.118196       0.1065607       2.774424e-16
##          Learn.5 sig.5   Learn.6 sig.6   Learn.7 sig.7
## AVG 1.000000e+00    +  0.9728596       0.9833417
## STD 2.774424e-16       0.3515190       0.3829643
##
## - Dataset: a6
##       Learn.1   Learn.2 sig.2   Learn.3 sig.3      Learn.4 sig.4
## AVG 0.9133912 0.9359697    ++ 1.0191041       1.000000e+00
## STD 0.3573499 0.6045963       0.1991436       2.451947e-16
##          Learn.5 sig.5   Learn.6 sig.6   Learn.7 sig.7
## AVG 1.000000e+00       0.9275673       0.9117580
## STD 2.451947e-16       0.3793325       0.3757454
##
```

```
## Legends:
## Learners -> Learn.1 = cv.rf.v3 ; Learn.2 = cv.lm.v1 ; Learn.3 = cv.rpart.v1 ; Learn.4 = cv.rpart.v2
## Signif. Codes -> 0 '++' or '--' 0.001 '+' or '-' 0.05 ' ' 1
```

```
# we try to test which of the performances of the four dataframes are statistically significant with a
```

In the below code, we compare all the best scores of the models and try to figure out which model or technique gives us the most accurate result.

```
bestModelsNames <- sapply(bestScores(res.all),
                           function(x) x[ 'nmse','system'])
learners <- c(rf= 'randomForest',rpart='rpartXse') # we use two learners for comparing the models
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]]))
}
```

In the below code, we use knnImputation to fill in the values with the most relevant data by mapping it to its closest neighbour. We use k= 10 here. Then we create a preds matrix where we predict the values of all the seven algae by using the best model. Here we compare these predictions with the real values to obtain some feedback on the quality of our approach to this prediction problem. The true values of the test set are contained in the algae.sols data frame, available in our package.

```
clean.test.algae <- knnImputation(test.algae, k = 10, distData = algae[, 1:11])
preds <- matrix(ncol=7,nrow=140)
for(i in 1:nrow(clean.test.algae))
  preds[i,] <- sapply(1:7,
                      function(x)
    predict(bestModels[[x]],clean.test.algae[i,])
    )
```

```
avg.preds <- apply(algae[,12:18],2,mean)
apply( ((algae.sols-preds)^2), 2,mean) / apply( (scale(algae.sols,avg.preds,F)^2),2,mean)
```

```
##        a1        a2        a3        a4        a5        a6        a7
## 0.4694850 0.8639454 1.0000000 0.7085136 0.7242240 0.8318299 1.0000000
```

We thus calculate the Normalised Mean Square Error of all the seven types of algae prediction and successfully implemented the regression models for prediction of the algae in the water Thus, from the obtained results we can say that algae a1 has the least error in prediction whereas a3 and a7 have the worst prediction perfomance since they have the maximum error. So we can say that from the water.