Model Evaluation

Contents

Overview

This section evaluates the fit of the model's using the car package MLR requires the residuals to be ~ IID N(0, sigma^2) the residuals will be standardized for the assessment Normality Assumptions will be accessed using: i. Normality tests from the nortest package ii. Visualizations such as histograms, QQ-plots, Residual Plots and Add Variable Plots Constant Variance will be accessed using: i. non-constant variance test Multi-collinearity will be accessed using: i. variance inflation factors Outliers will be accessed using: i. Cooks Distance

```
# load relevant libraries
library(rmarkdown)
library(knitr)
library(car)
```

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

```
library(MASS)
library(nortest)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following object is masked from 'package:car':
##
##     recode

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
#-- Data --#

# load in the historic voting data for deriving the voting blocs
processed_data <- read.csv(file = "./data/processed_data.csv", header = T)
# split the televote data
televote_data <- processed_data %>% filter(Voting_Method_J == 0)
# split out the jury vote data
jury_data <- processed_data %>% filter(Voting_Method_J == 1)
```

Overall Model

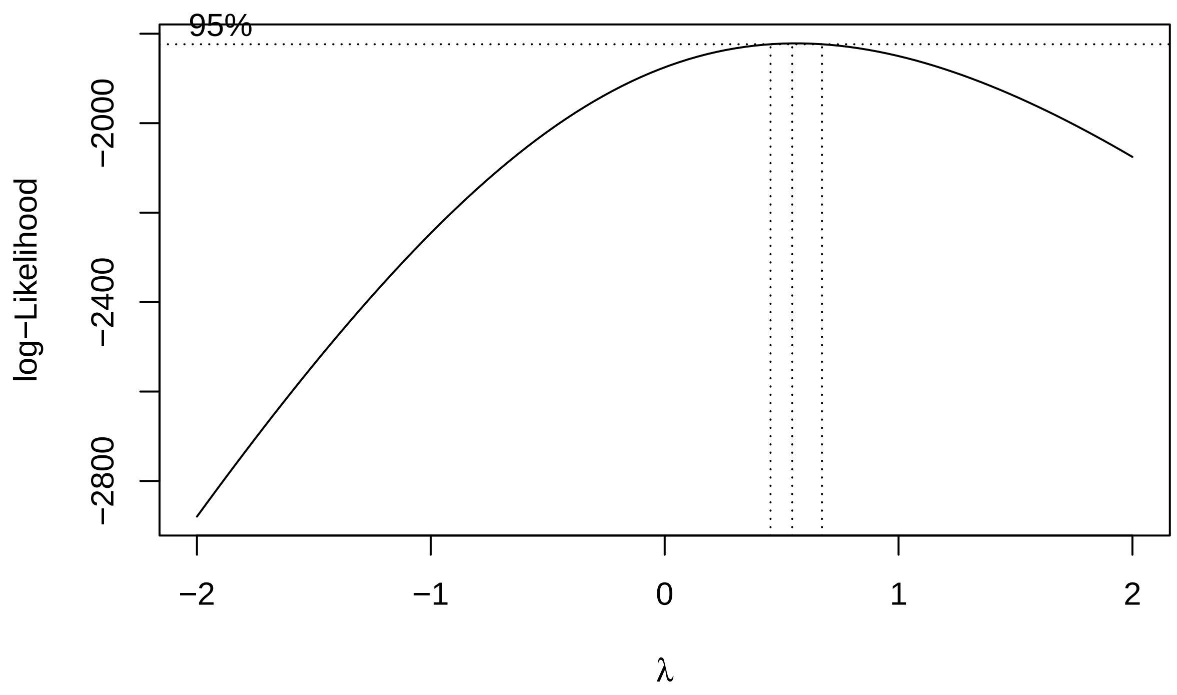
```r
# load final overall model
my_model_overall <- readRDS("./models/overall_final_model.RDS")
# extract out the model coefficients
overall_model_coeff <- names(my_model_overall$coefficients)[-1]
# recreate the model formula
overall_final_model_form<- as.formula(paste('Points ~', paste(overall_model_coeff, collapse = ' + ')))
# generate model summary
summary(my_model_overall)
```

```
##
## Call:
## lm(formula = overall_final_model_form, data = processed_data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5505 -2.3301 -0.2858  2.1846  7.8517
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.7244     0.6169   6.037 2.64e-09 ***
## Average_Points     0.4798     0.1253   3.830 0.000141 ***
## acousticness       0.6959     0.1302   5.344 1.26e-07 ***
## speechiness        0.6973     0.1362   5.119 4.05e-07 ***
## METRIC_Citizens    0.3251     0.1399   2.324 0.020438 *
## TC_PerfType_Solo   1.4412     0.5613   2.568 0.010457 *
## key_0              1.2923     0.4516   2.861 0.004353 **
## CAP_DIST_km        0.2956     0.1280   2.309 0.021260 *
## OOA                1.2837     0.4512   2.845 0.004579 **
## FC_NonCOB          0.3604     0.1391   2.592 0.009766 **
## ComSONGLAN         0.2760     0.1287   2.145 0.032338 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.074 on 647 degrees of freedom
## Multiple R-squared:  0.1762, Adjusted R-squared:  0.1635
## F-statistic: 13.84 on 10 and 647 DF,  p-value: < 2.2e-16
```

Transformation of Response Variable

```
# Apply box-cox transformation on the model to improve the normality assumptions
# box-cox transformation using car
bct <- MASS::boxcox(object = my_model_overall)
```

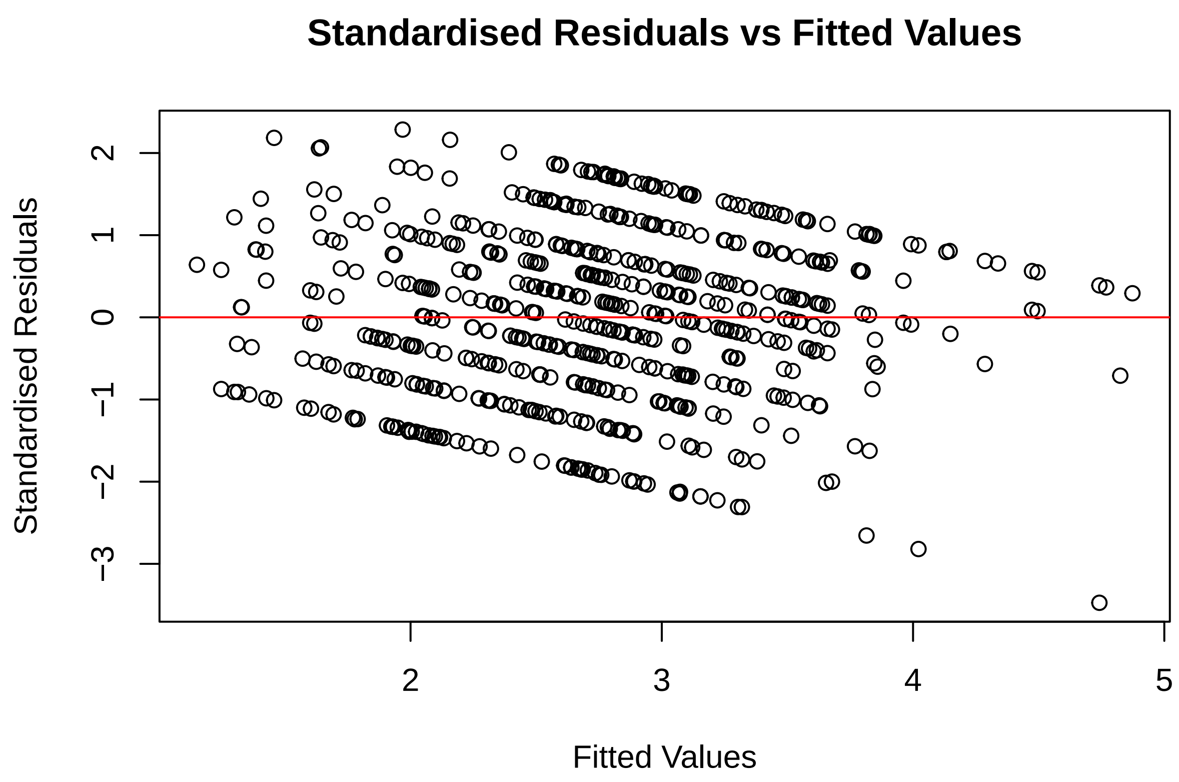


```
# find optimal box-cox power transformation power
p <- bct$x[which.max(x = bct$y)]
# transform points using the optimal power transformation
bctPoints <- (((processed_data$Points)^p) - 1)/(p)
# recreate the model formula
overall_final_model_bct_form<- as.formula(paste('bctPoints ~', paste(overall_model_coeff, collapse = ' +
# refit final model with with box-cox power transformation
my_model_overall <- lm(formula = overall_final_model_bct_form, data = processed_data)
# generate model summary
summary(my_model_overall)
```

```
##
## Call:
## lm(formula = overall_final_model_bct_form, data = processed_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7417 -1.0276  0.0449  1.1112  3.3083
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.71220    0.29239   5.856 7.55e-09 ***
## Average_Points    0.20209    0.05938   3.403 0.000706 ***
## acousticness      0.34061    0.06173   5.518 4.96e-08 ***
## speechiness       0.33666    0.06456   5.215 2.48e-07 ***
## METRIC_Citizens   0.12269    0.06630   1.851 0.064672 .
## TC_PerfType_Solo  0.69660    0.26601   2.619 0.009035 **
## key_0             0.65422    0.21405   3.056 0.002332 **
## CAP_DIST_km       0.12426    0.06069   2.048 0.041007 *
## OOA               0.62781    0.21385   2.936 0.003446 **
## FC_NonCOB         0.18387    0.06591   2.790 0.005428 **
## ComSONGLAN        0.14046    0.06099   2.303 0.021585 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.457 on 647 degrees of freedom
## Multiple R-squared:  0.169,  Adjusted R-squared:  0.1562
## F-statistic: 13.16 on 10 and 647 DF,  p-value: < 2.2e-16
```

Evaluate the Fit of the Model

```
# create standardize residuals
sresid <- studres(my_model_overall)
# Residual vs fits plot
plot(x = my_model_overall$fitted.values, y = sresid, main = "Standardised Residuals vs Fitted Values",
# add red horizontal line through y-axis 0
abline(h = 0, col = "red")
```



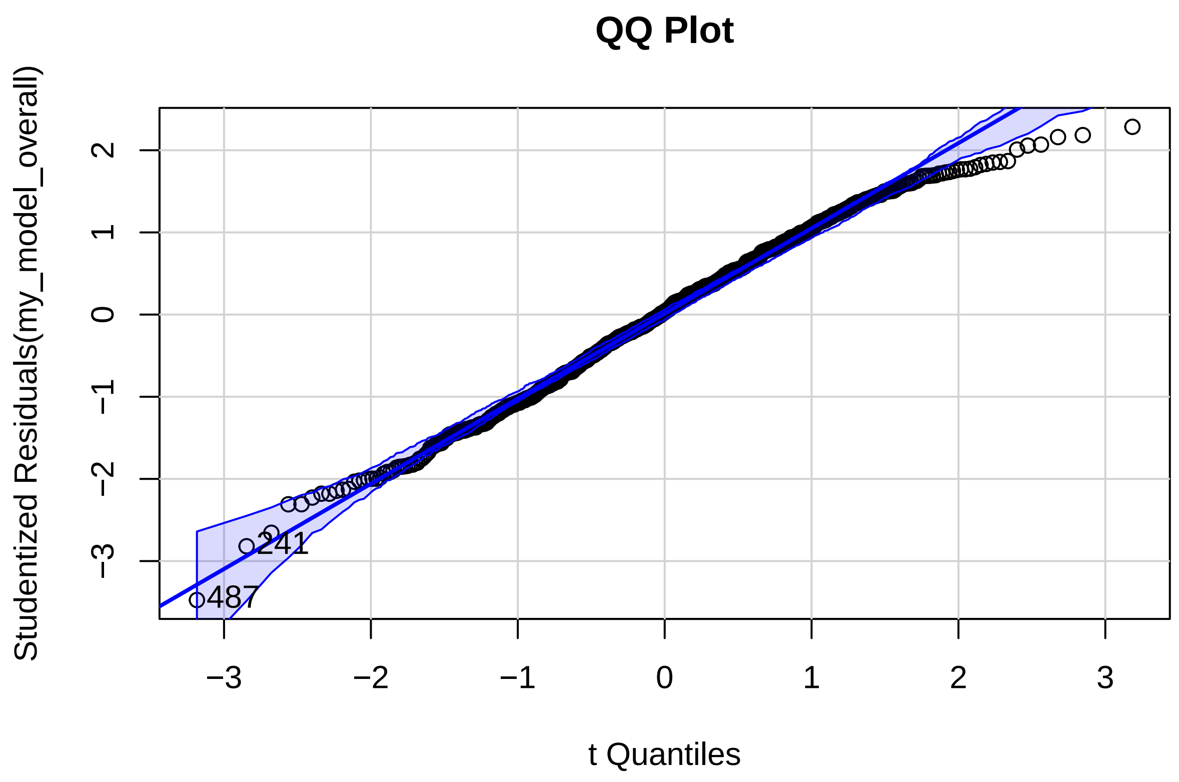
```r
# Assessing Outliers
# Bonferonni p-value for most extreme obs
outlierTest(my_model_overall)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##      rstudent unadjusted p-value Bonferroni p
## 487 -3.473914         0.00054721      0.36006
```

```r
# final outlier residuals
which(sresid < -2)
```

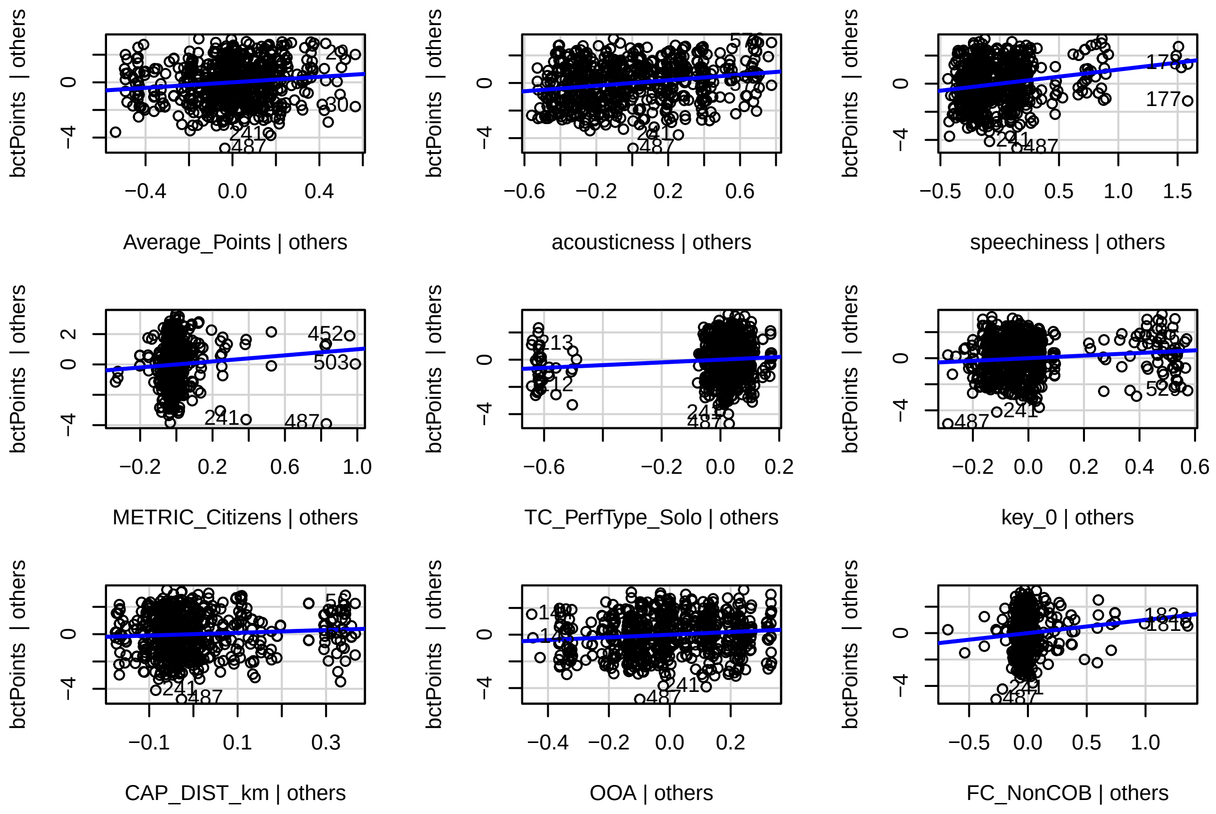
```
##  39  77 103 177 241 360 464 487 516 529 531 535 536 618
##  39  77 103 177 241 360 464 487 516 529 531 535 536 618
```

```r
# qq-plot for studentized residuals
qqPlot(my_model_overall, main = "QQ Plot")
```



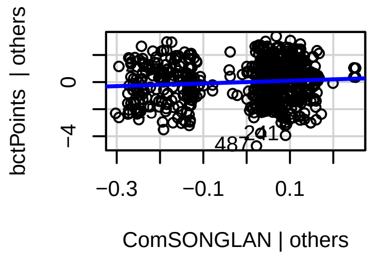
```
## [1] 241 487
```

```r
# leverage plots
leveragePlots(my_model_overall)
```

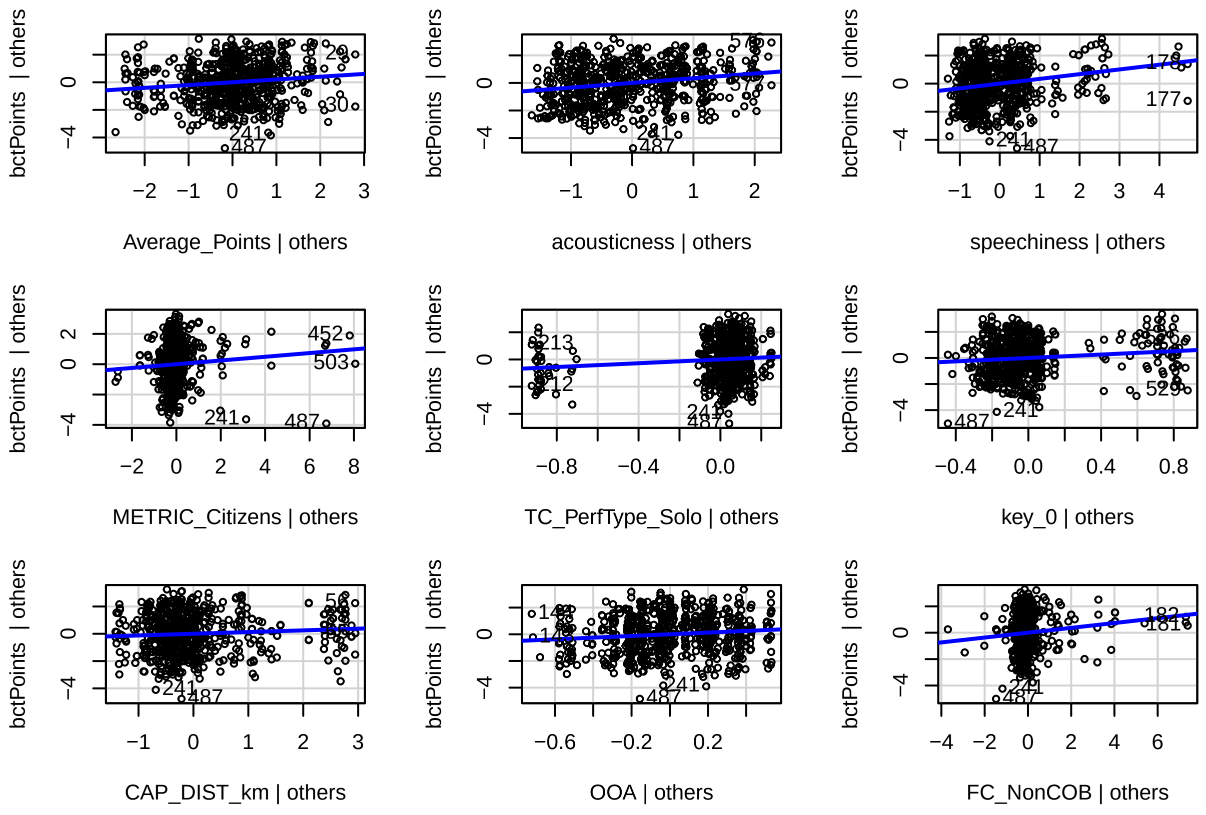


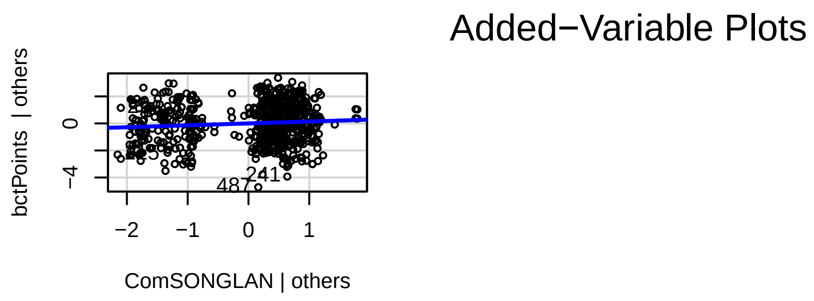
bctPoints | others
Average_Points | others
acousticness | others
speechiness | others
METRIC_Citizens | others
TC_PerfType_Solo | others
key_0 | others
CAP_DIST_km | others
OOA | others
FC_NonCOB | others
241
487
177
452
503
213
529
182
181

Leverage Plots

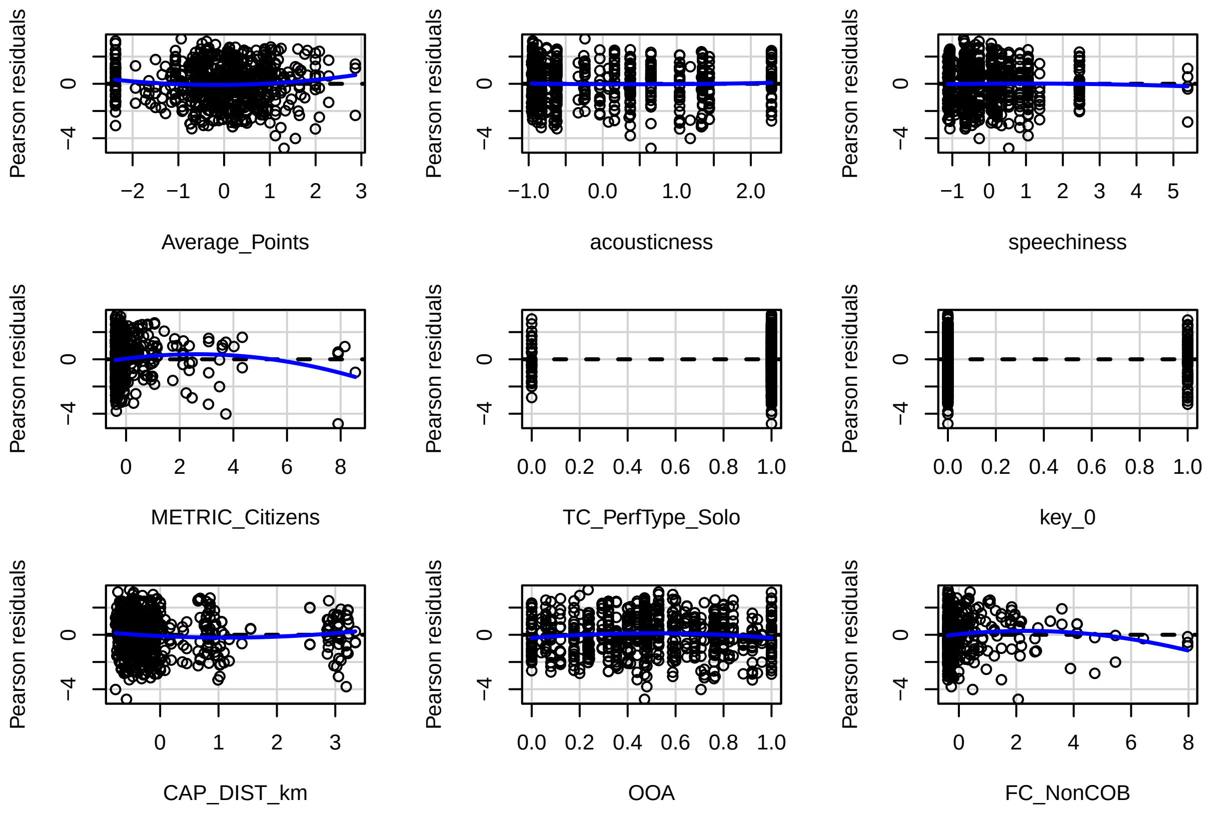


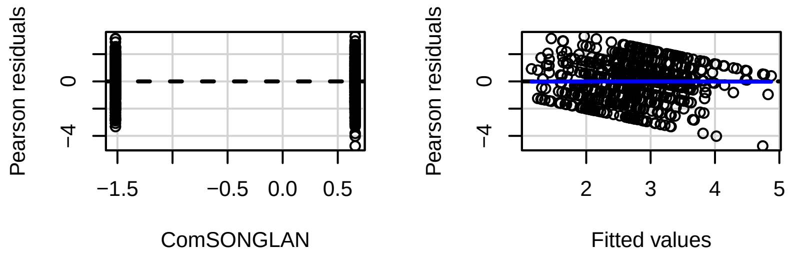
```
# Added variable Plots
avPlots(my_model_overall)
```


bctPoints | others
Average_Points | others
acousticness | others
speechiness | others
METRIC_Citizens | others
TC_PerfType_Solo | others
key_0 | others
CAP_DIST_km | others
OOA | others
FC_NonCOB | others



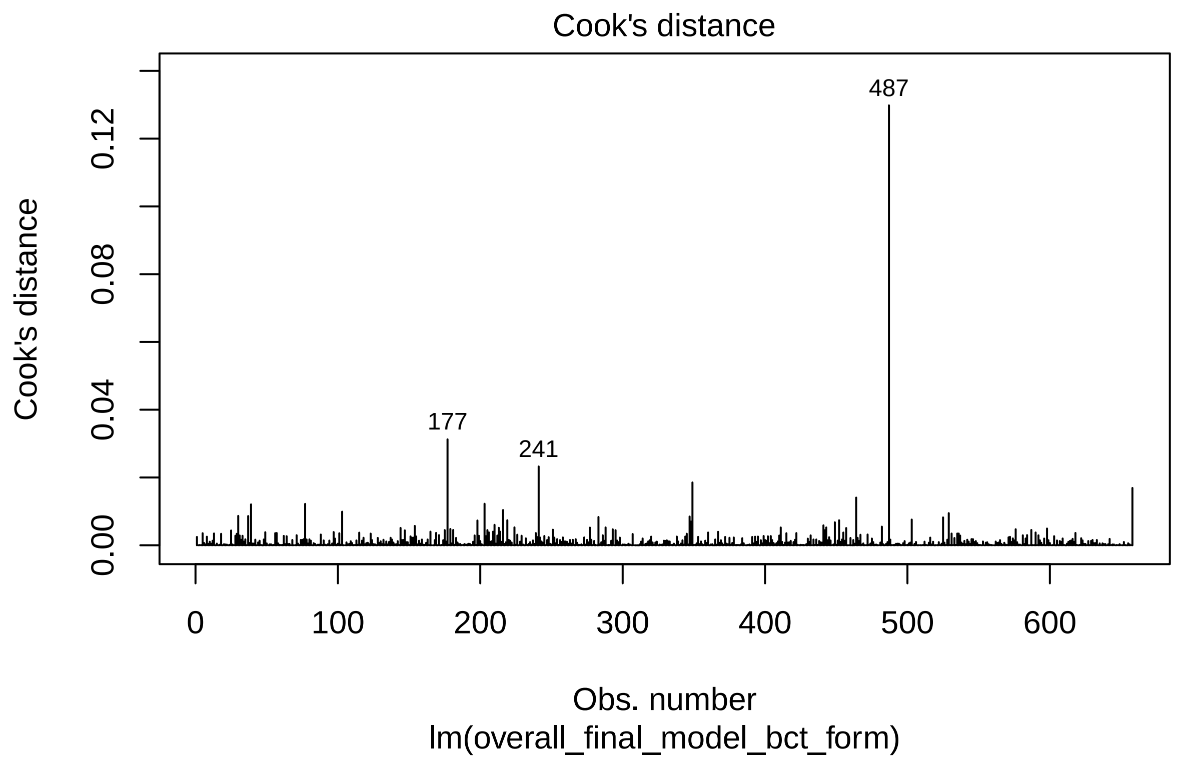
```
# Residual Plots
residualPlots(my_model_overall)
```


Pearson residuals
Average_Points
acousticness
speechiness
METRIC_Citizens
TC_PerfType_Solo
key_0
CAP_DIST_km
OOA
FC_NonCOB



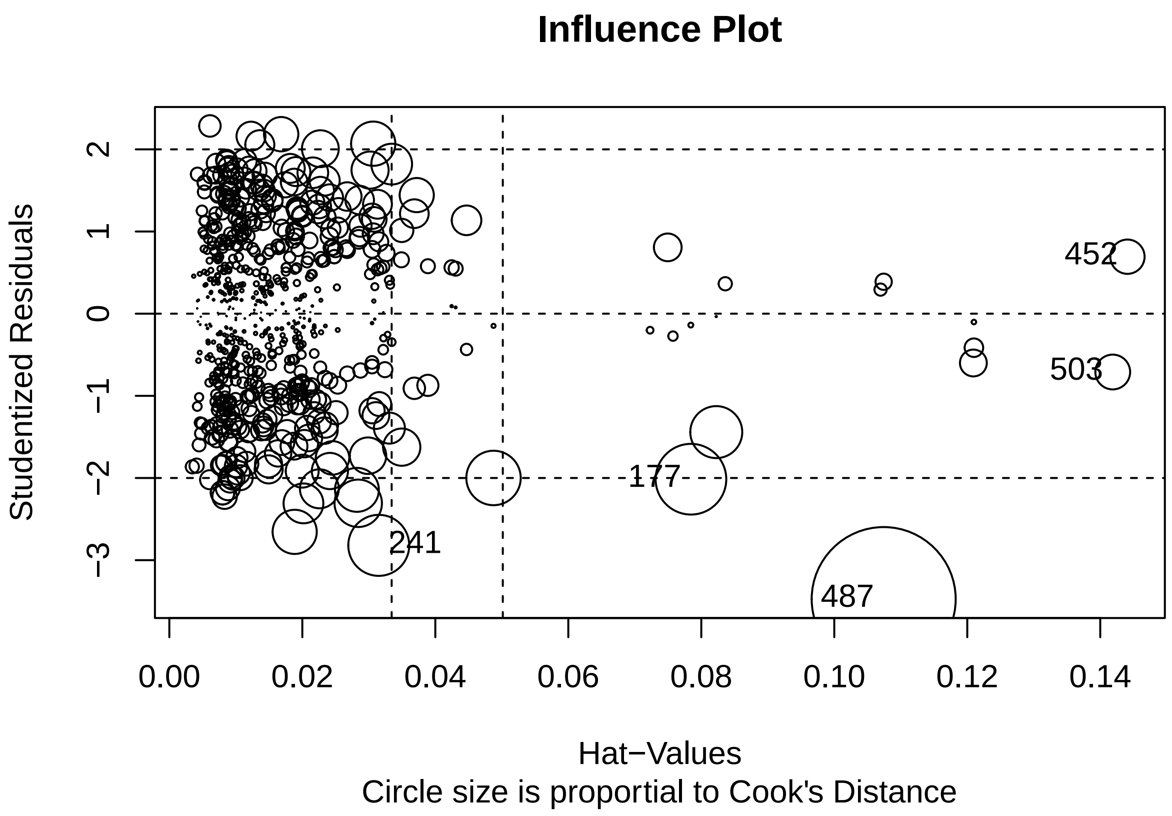
```
##                     Test stat Pr(>|Test stat|)
## Average_Points         2.2247         0.026443 *
## acousticness           0.5338         0.593682
## speechiness           -0.4246         0.671259
## METRIC_Citizens       -2.7460         0.006201 **
## TC_PerfType_Solo       0.7835         0.433605
## key_0                  0.5665         0.571238
## CAP_DIST_km            1.8086         0.070981 .
## OOA                   -2.4388         0.015004 *
## FC_NonCOB             -2.2184         0.026874 *
## ComSONGLAN             1.2837         0.199719
## Tukey test            -0.0599         0.952216
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Cook's D plot
# identify D values > 4/(n-k-1)
cutoff <- 4/((nrow(processed_data)-length(my_model_overall$coefficients)-2))
# Crooks Distance plot
plot(my_model_overall, which = 4, cook.levels = cutoff)
```



```
# Influence Plot
influencePlot(my_model_overall, id.method = "identify", main = "Influence Plot", sub = "Circle size is p
```

```
## Warning in plot.window(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter

## Warning in box(...): "id.method" is not a graphical parameter

## Warning in title(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.method" is not a
## graphical parameter
```



| | StudRes| Hat| CookD| |:—|———-:|———:|———:| |177 | -2.0146453| 0.0784447| 0.0312607| |241 | -2.8196641| 0.0314946| 0.0232539| |452 | 0.6934167| 0.1440912| 0.0073647| |487 | -3.4739137| 0.1074300| 0.1298262| |503 | -0.7100016| 0.1418771| 0.0075827|

Normality Test Ho: The data is normally distributed Ha: the data is not normally distributed

```
# Normality Test
shapiro.test(sresid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sresid
## W = 0.99127, p-value = 0.0006293
```

```
ad.test(sresid)
```

```
##
##  Anderson-Darling normality test
##
## data:  sresid
## A = 1.173, p-value = 0.004584
```

```
cvm.test(sresid)
```

```
##
```

```
##  Cramer-von Mises normality test
##
## data:  sresid
## W = 0.16383, p-value = 0.01564
```

```
lillie.test(sresid)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  sresid
## D = 0.032825, p-value = 0.08898
```

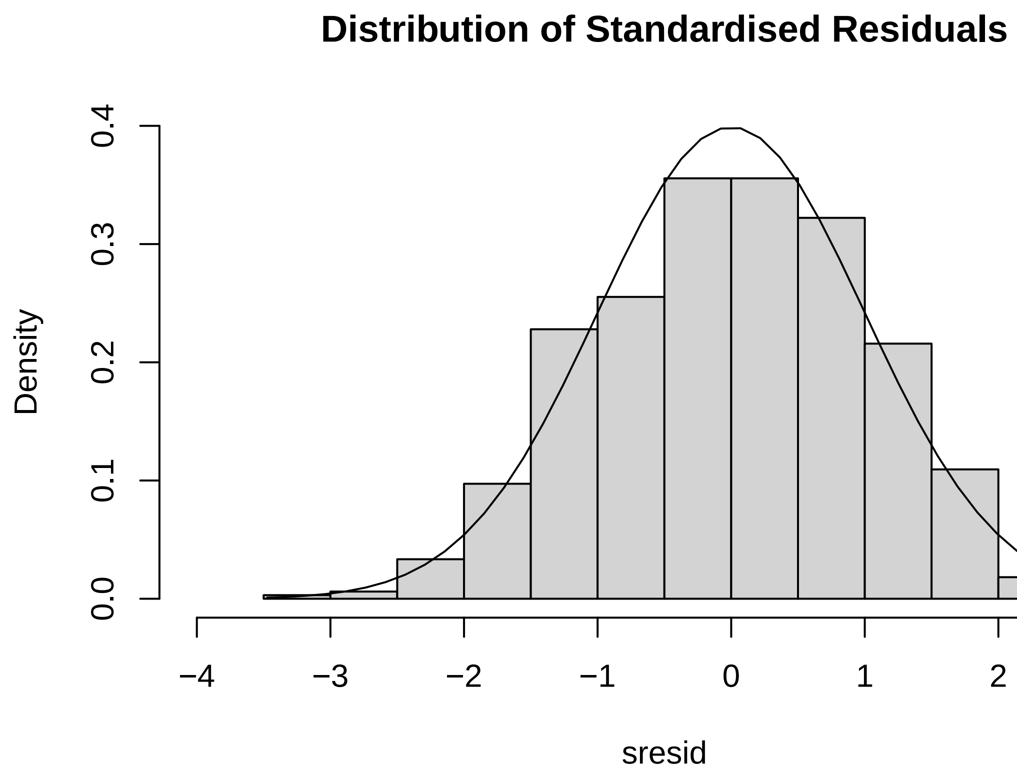
```
pearson.test(sresid)
```

```
##
##  Pearson chi-square normality test
##
## data:  sresid
## P = 37.024, p-value = 0.04352
```

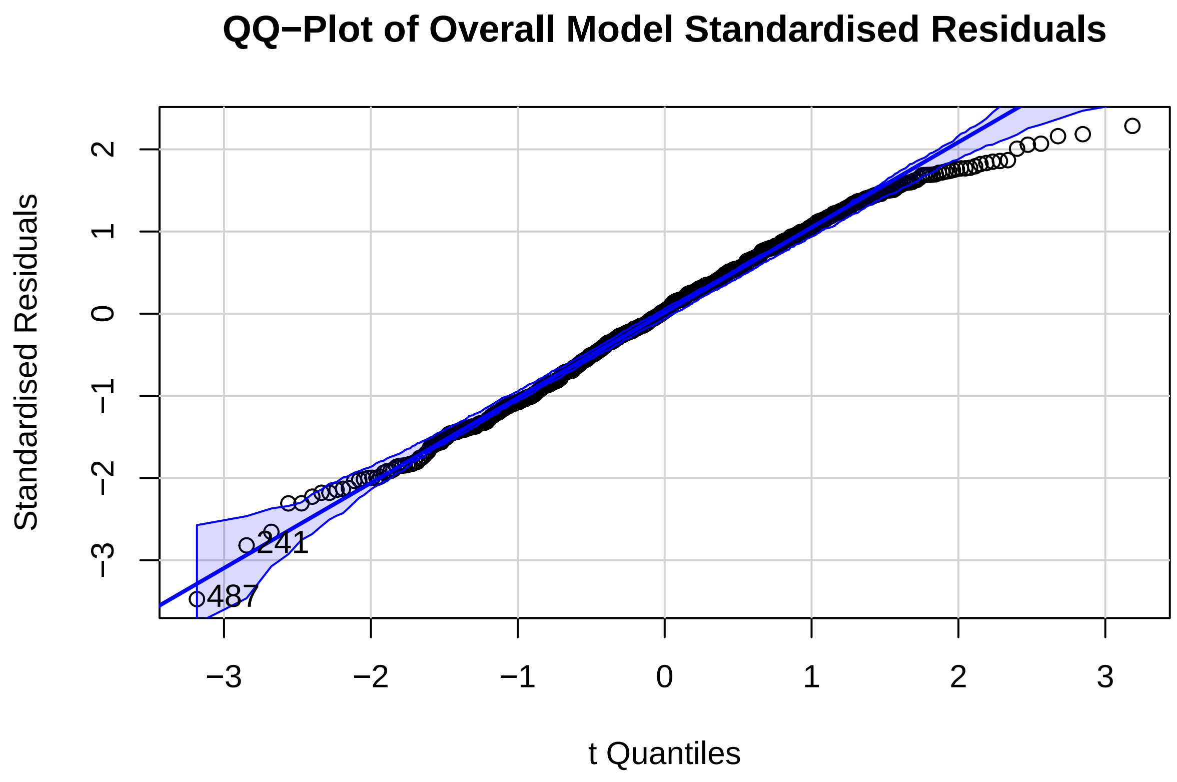
```
sf.test(sresid)
```

```
##
##  Shapiro-Francia normality test
##
## data:  sresid
## W = 0.99178, p-value = 0.001487
```

```
# the data is not normally distributed
# Histogram of residuals
hist(sresid, freq = FALSE, main = "Distribution of Standardised Residuals", ylim = c(0,0.4),  xlim = c(
xfit <- seq(min(sresid, na.rm = TRUE), max(sresid, na.rm = TRUE), length = 40)
yfit <- dnorm(xfit)
lines(xfit, yfit)
```



```
# QQ-plot of the data
qqPlot(my_model_overall, ylab = "Standardised Residuals", main = "QQ-Plot of Overall Model Standardised
```



```
## [1] 241 487
```

Non-Constant Error Variance Test Ho: constant error variance Ha: Non-constant error Variance

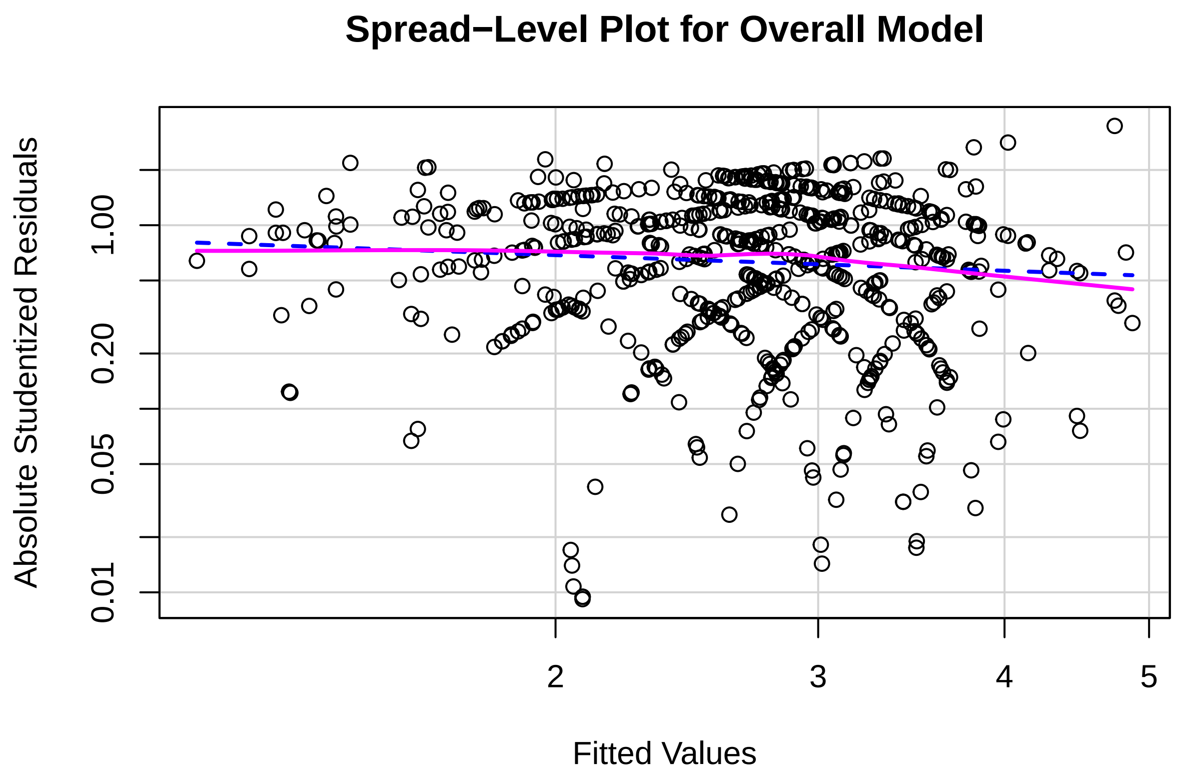
```
# Non-Constant Error Variance Test
ncvTest(my_model_overall)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.03663296, Df = 1, p = 0.84821
```

```
bptest(my_model_overall)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  my_model_overall
## BP = 12.715, df = 10, p-value = 0.2401
```

```
# plot studentized residuals vs. fitted values
spreadLevelPlot(my_model_overall, main = "Spread-Level Plot for Overall Model")
```



```
##
## Suggested power transformation:  1.282955
```

```
# Variance Inflation Factors
vif(my_model_overall)
```

```
##   Average_Points     acousticness      speechiness  METRIC_Citizens
##         1.091139         1.179141         1.289927         1.360161
## TC_PerfType_Solo            key_0      CAP_DIST_km              OOA
##         1.104542         1.246838         1.139690         1.188927
##        FC_NonCOB       ComSONGLAN
##         1.344230         1.150996
```

```
sqrt(vif(my_model_overall)) > 2
```

```
##   Average_Points     acousticness      speechiness  METRIC_Citizens
##            FALSE            FALSE            FALSE            FALSE
## TC_PerfType_Solo            key_0      CAP_DIST_km              OOA
##            FALSE            FALSE            FALSE            FALSE
##        FC_NonCOB       ComSONGLAN
##            FALSE            FALSE
```

```
# No signs of collinearity
```

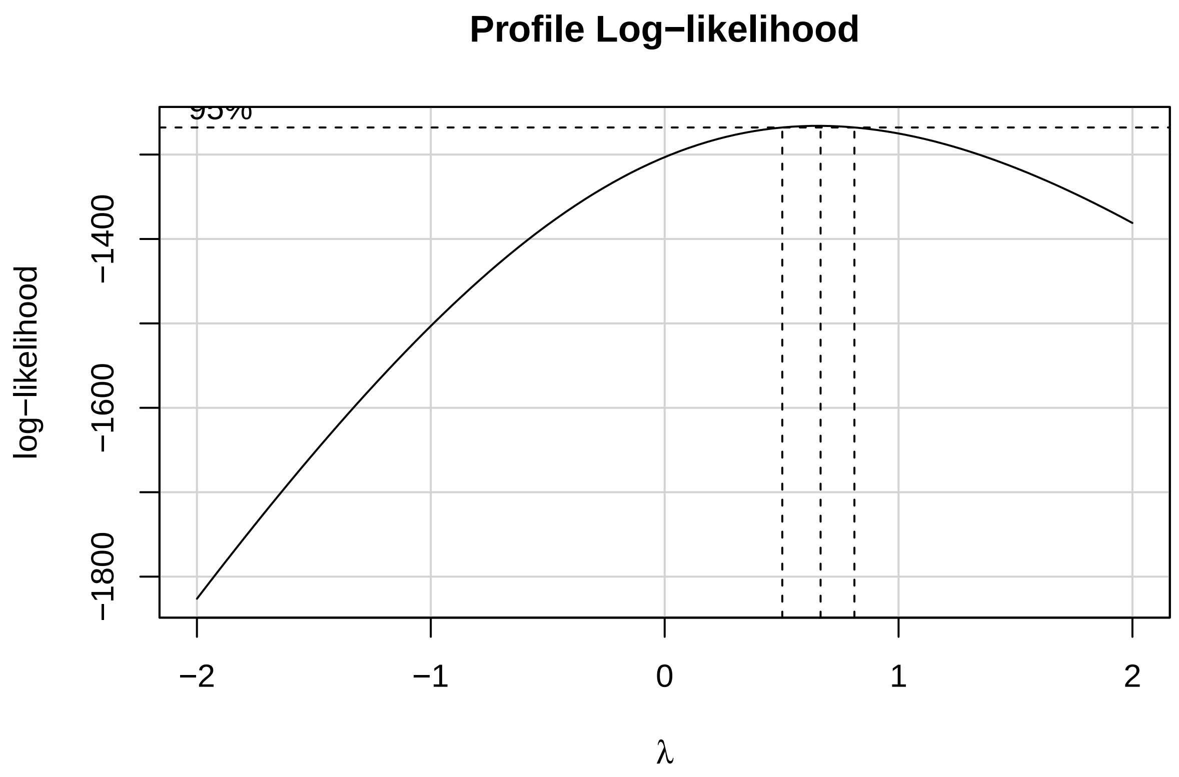
TELEVOTE MODEL

```
# load final televote model
my_model_tele <- readRDS("./models/televote_final_model.RDS")
# extract out the model coefficients
tele_model_coeff <- names(my_model_tele$coefficients)[-1]
# recreate the model formula
televote_final_model_form<- as.formula(paste('Points ~', paste(tele_model_coeff, collapse = ' + ')))
# generate model summary
summary(my_model_tele)
```

```
##
## Call:
## lm(formula = televote_final_model_form, data = televote_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3561 -1.9688 -0.0461  1.7443  6.7011
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.1314     0.3466  14.806  < 2e-16 ***
## METRIC_Citizens   0.5344     0.1555   3.436 0.000668 ***
## Average_Points    0.8126     0.1607   5.057 7.22e-07 ***
## TC_NumNeigh       0.7464     0.1742   4.286 2.42e-05 ***
## speechiness       0.5175     0.1656   3.125 0.001943 **
## acousticness      0.4804     0.1681   2.858 0.004550 **
## FC_NonCitzens     0.6452     0.1767   3.652 0.000304 ***
## VBlocs1_TC_13    -6.8165     2.1841  -3.121 0.001968 **
## OOA               0.8913     0.6028   1.479 0.140203
## CAP_DIST_km       0.3029     0.1726   1.755 0.080254 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.772 on 317 degrees of freedom
## Multiple R-squared:  0.3384, Adjusted R-squared:  0.3196
## F-statistic: 18.02 on 9 and 317 DF,  p-value: < 2.2e-16
```

Transformation of Response Variable

```
# Apply box-cox transformation on the model to improve the normality assumptions
# box-cox transformation using car
bct <- boxCox(object = my_model_tele)
```

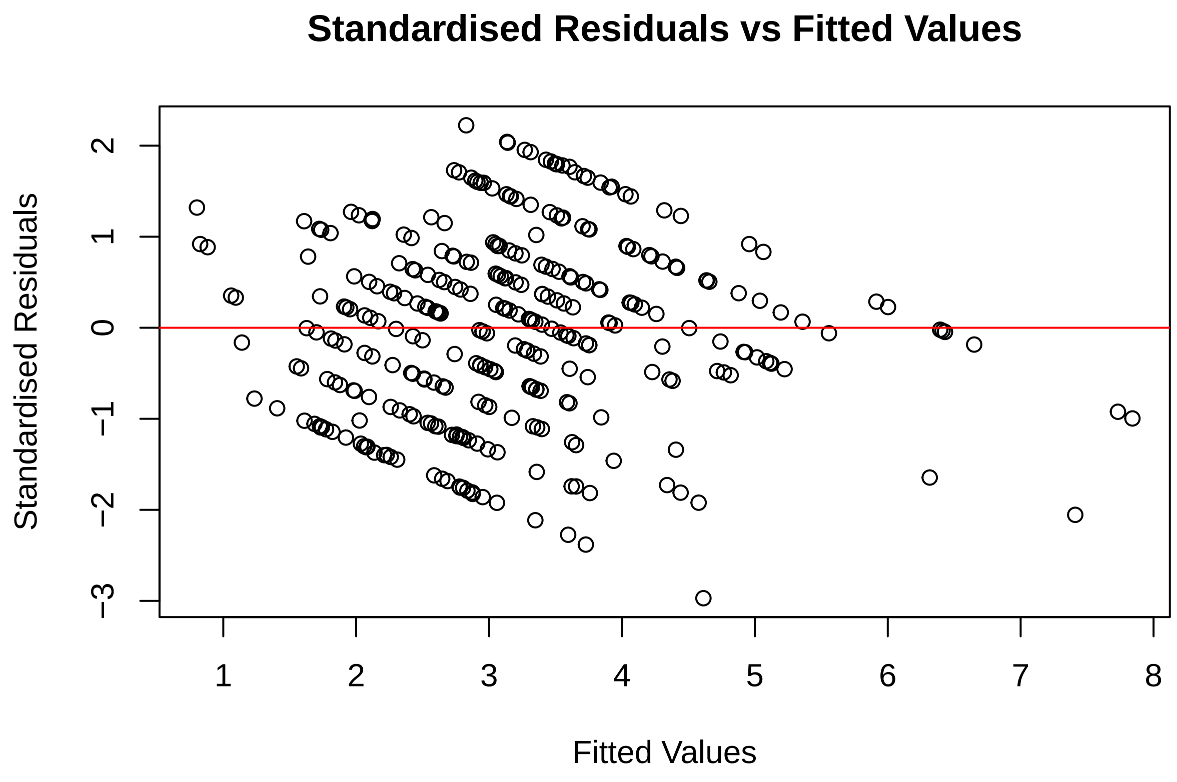


```
# find optimal box-cox power transformation power
p <- bct$x[which.max(x = bct$y)]
# transform points using the optimal power transformation
bctPoints <- (((televote_data$Points)^p) - 1)/(p)
# recreate the model formula
televote_final_model_bct_form <- as.formula(paste('bctPoints ~', paste(tele_model_coeff, collapse = ' +
# refit final model with with box-cox power transformation
my_model_tele <- lm(formula = televote_final_model_bct_form, data = televote_data)
# generate model summary
summary(my_model_tele)
```

```
##
## Call:
## lm(formula = televote_final_model_bct_form, data = televote_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6127 -1.0874  0.0907  1.1155  3.5345
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.75365    0.20088  13.708  < 2e-16 ***
## METRIC_Citizens   0.28085    0.09013   3.116 0.002000 **
## Average_Points    0.44003    0.09313   4.725 3.47e-06 ***
## TC_NumNeigh       0.44883    0.10095   4.446 1.21e-05 ***
## speechiness       0.30365    0.09599   3.163 0.001711 **
## acousticness      0.28053    0.09743   2.879 0.004257 **
## FC_NonCitzens     0.35982    0.10241   3.514 0.000506 ***
## VBlocs1_TC_13    -3.80137    1.26592  -3.003 0.002888 **
## OOA               0.52110    0.34938   1.492 0.136818
## CAP_DIST_km       0.19110    0.10006   1.910 0.057054 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.607 on 317 degrees of freedom
## Multiple R-squared:  0.3246, Adjusted R-squared:  0.3054
## F-statistic: 16.92 on 9 and 317 DF,  p-value: < 2.2e-16
```

Evaluate the Fit of the Model

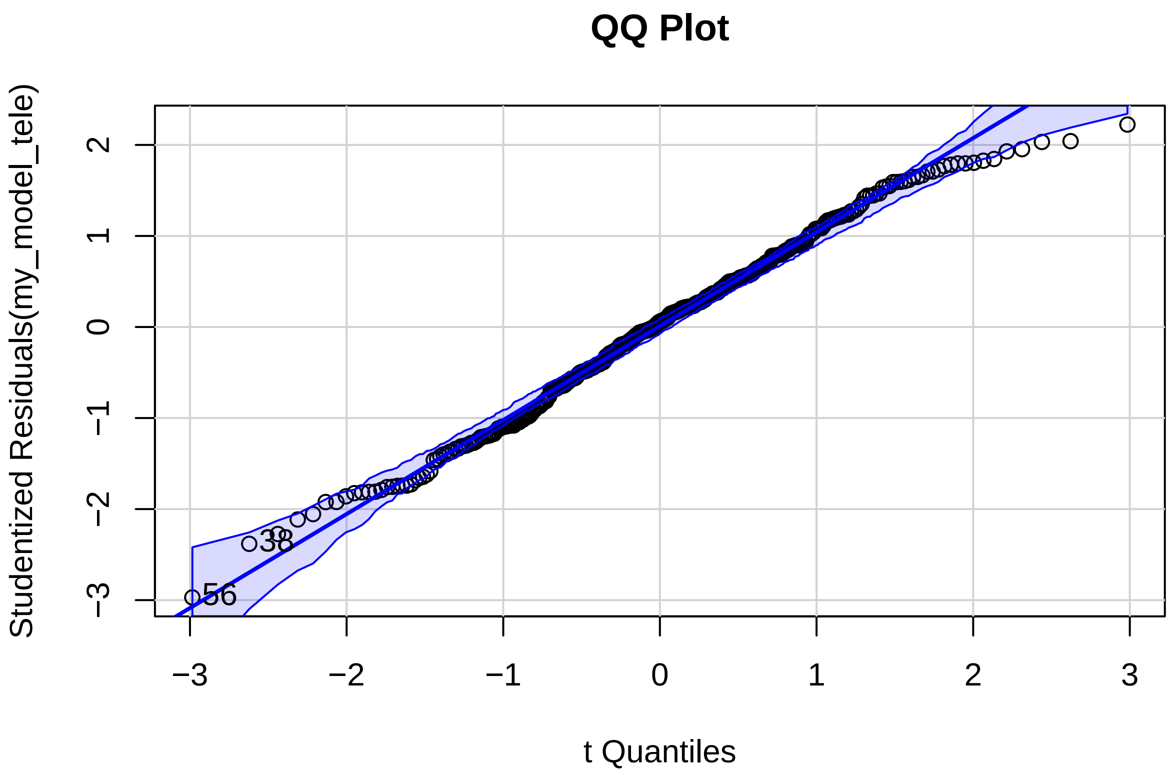
```
# create standardize residuals
sresid <- studres(my_model_tele)
# Residual vs fits plot
plot(x = my_model_tele$fitted.values, y = sresid, main = "Standardised Residuals vs Fitted Values", xlal
# add red horizontal line through y-axis 0
abline(h = 0, col = "red")
```



```
# Assessing Outliers
# Bonferonni p-value for most extreme obs
outlierTest(my_model_tele)
```

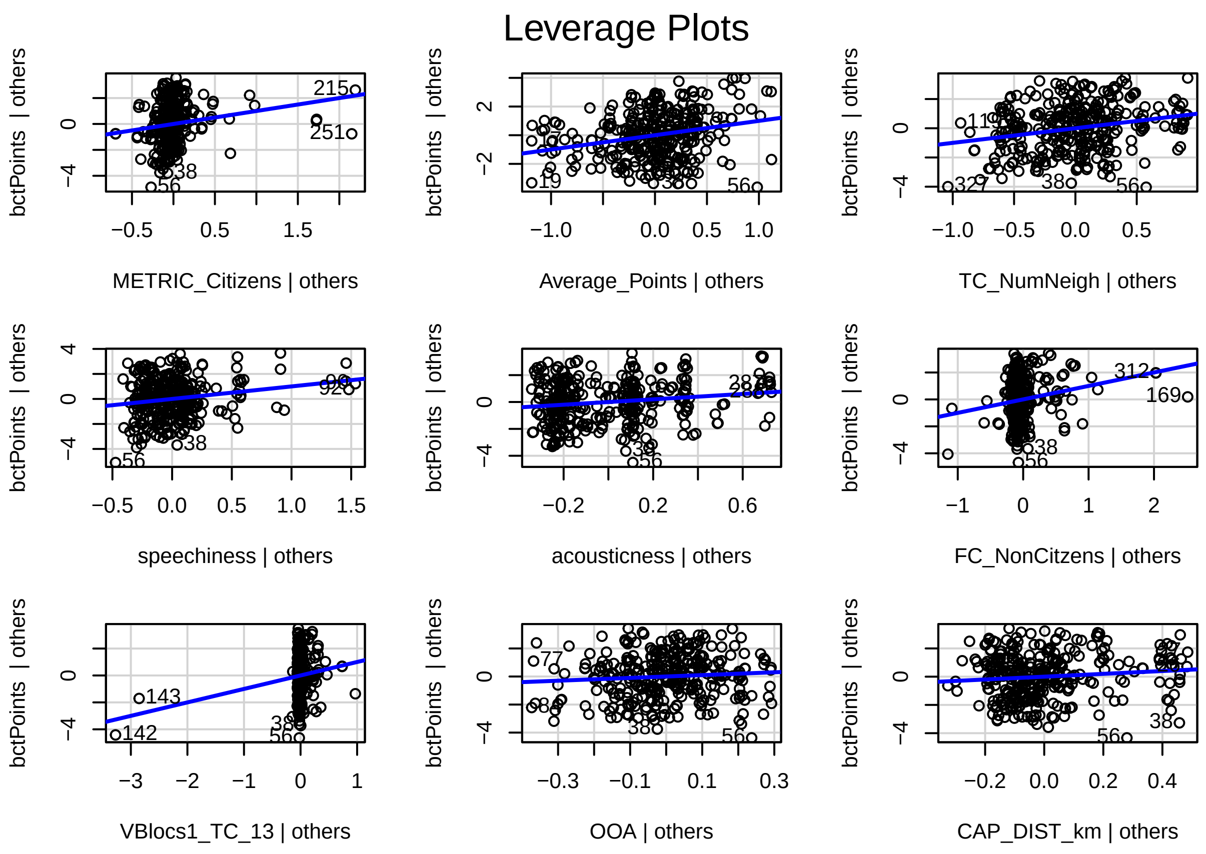
```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 56 -2.970711          0.0031989           NA
```

```
#qq plot for studentized residuals
qqPlot(my_model_tele, main = "QQ Plot")
```

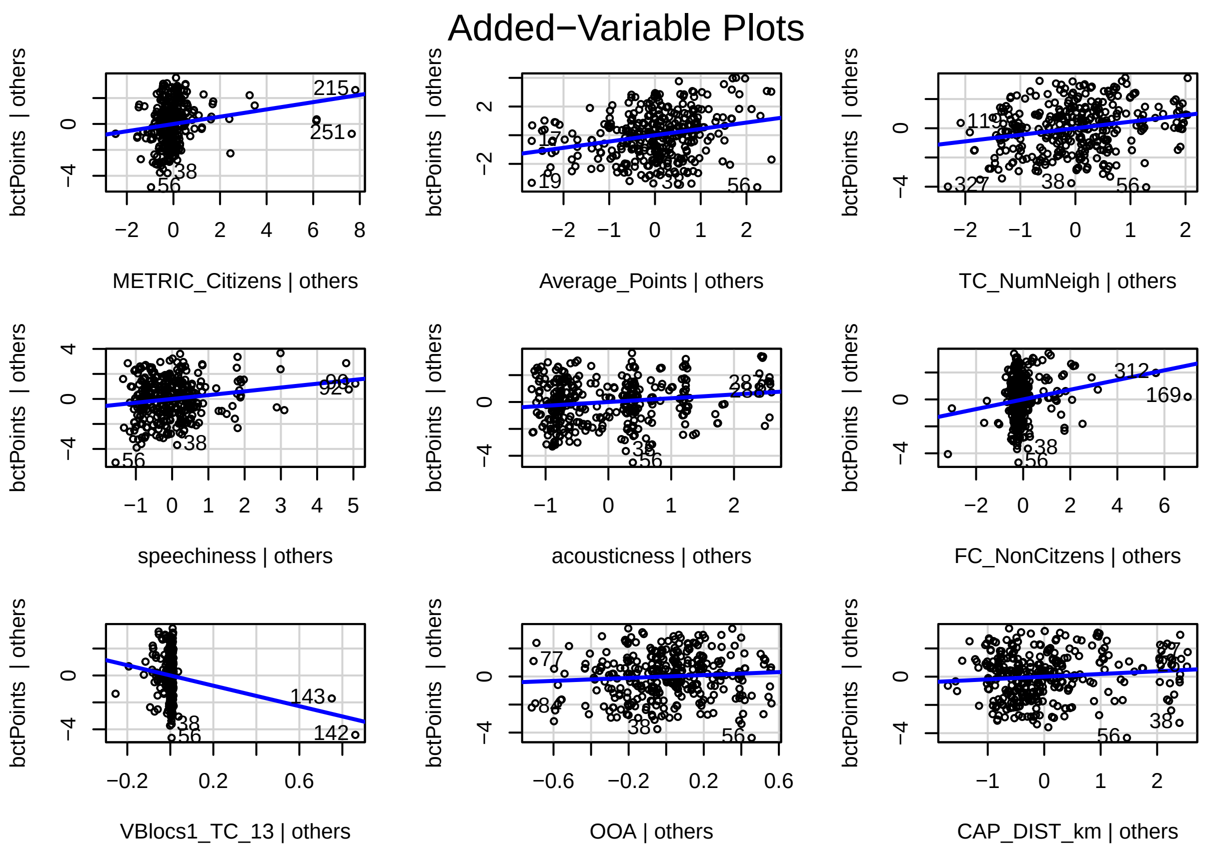


```
## [1] 38 56
```

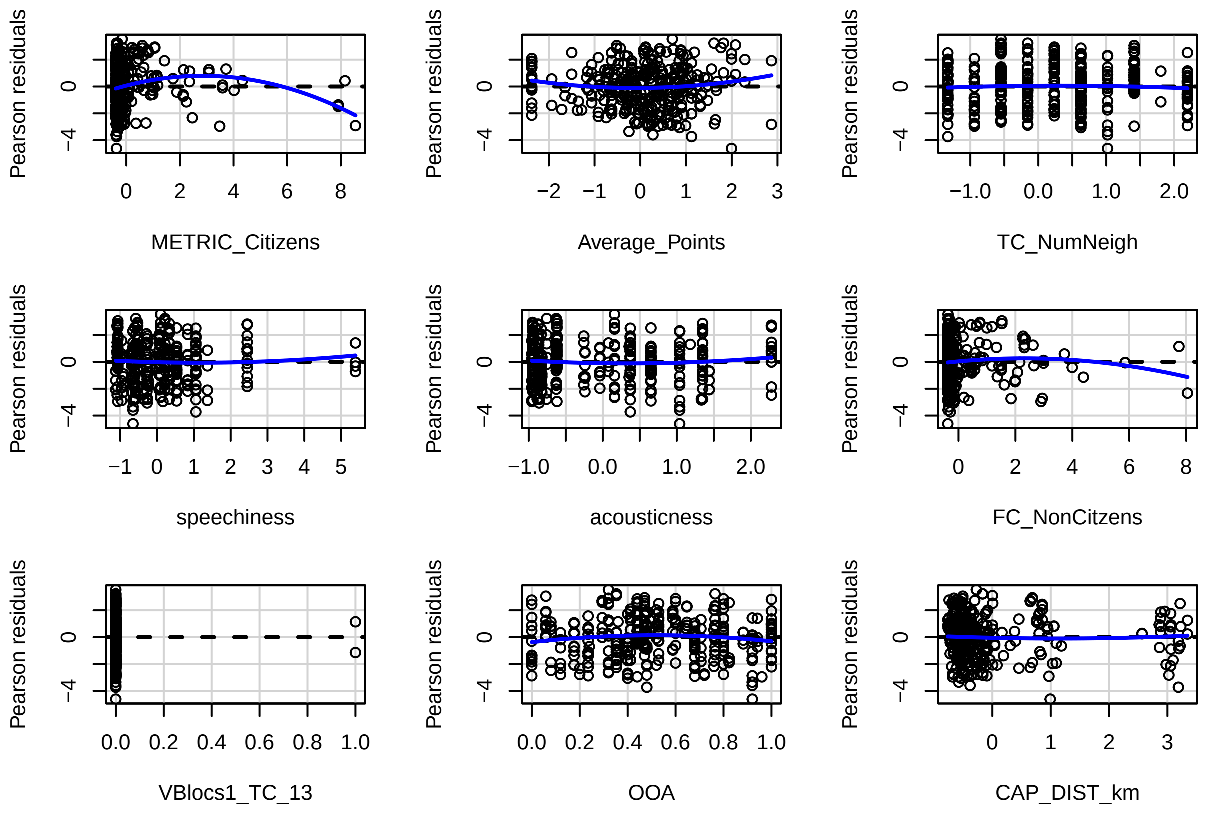
```
# leverage plots
leveragePlots(my_model_tele)
```

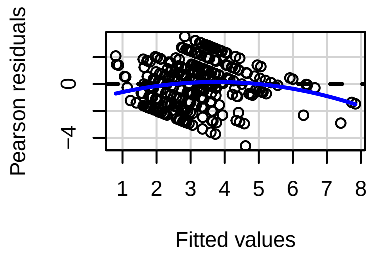


```r
# Added variable Plots
avPlots(my_model_tele)
```



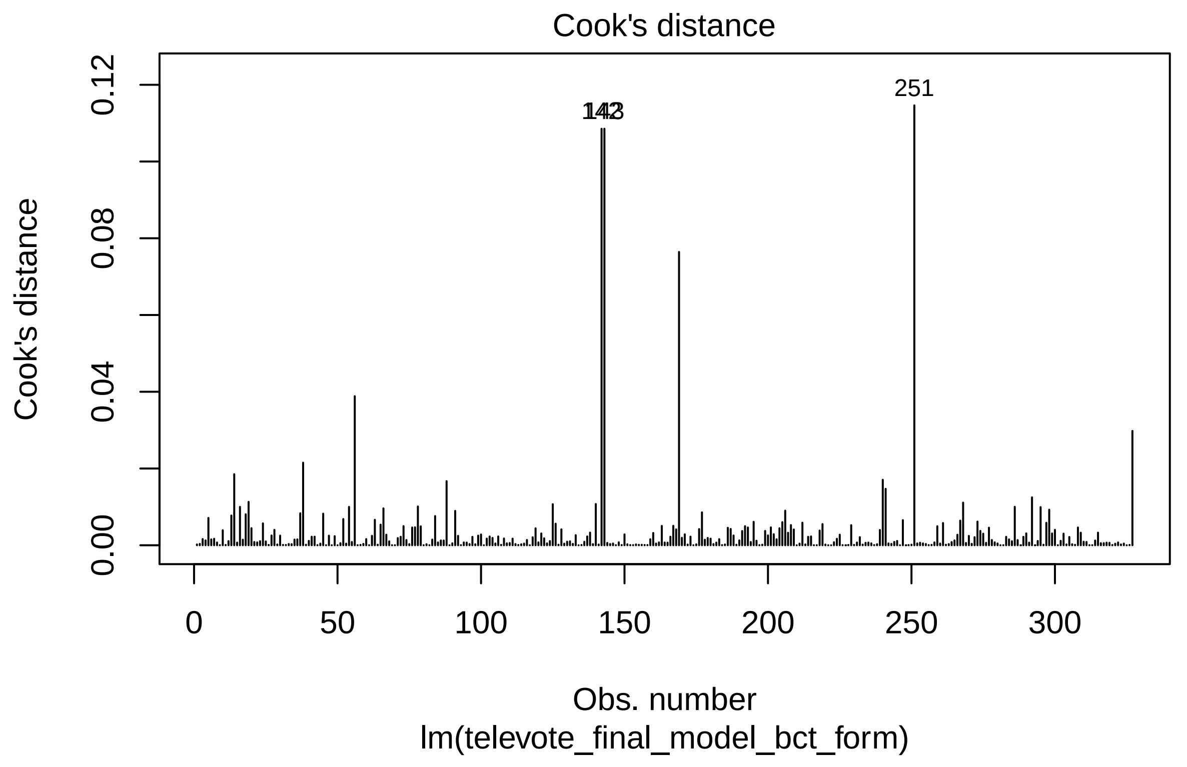
```
# Residual Plots
residualPlots(my_model_tele)
```


Pearson residuals
METRIC_Citizens
Average_Points
TC_NumNeigh
speechiness
acousticness
FC_NonCitzens
VBlocs1_TC_13
OOA
CAP_DIST_km



```
##                    Test stat Pr(>|Test stat|)
## METRIC_Citizens      -4.2900        2.378e-05 ***
## Average_Points        1.8891         0.059792 .
## TC_NumNeigh          -0.7737         0.439702
## speechiness           0.8468         0.397722
## acousticness          1.3501         0.177938
## FC_NonCitzens        -1.5501         0.122130
## VBlocs1_TC_13         0.8875         0.375504
## OOA                  -1.9812         0.048437 *
## CAP_DIST_km           0.5182         0.604654
## Tukey test           -3.2247         0.001261 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Cook's D plot
# identify D values > 4/(n-k-1)
cutoff <- 4/((nrow(processed_data) - length(my_model_tele$coefficients) - 2))
# Crooks Distance plot
plot(my_model_tele, which = 4, cook.levels = cutoff)
```



```
# Influence Plot
influencePlot(my_model_tele, id.method = "identify", main = "Influence Plot", sub = "Circle size is prop
```

```
## Warning in plot.window(...): "id.method" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "id.method" is not a graphical parameter
```

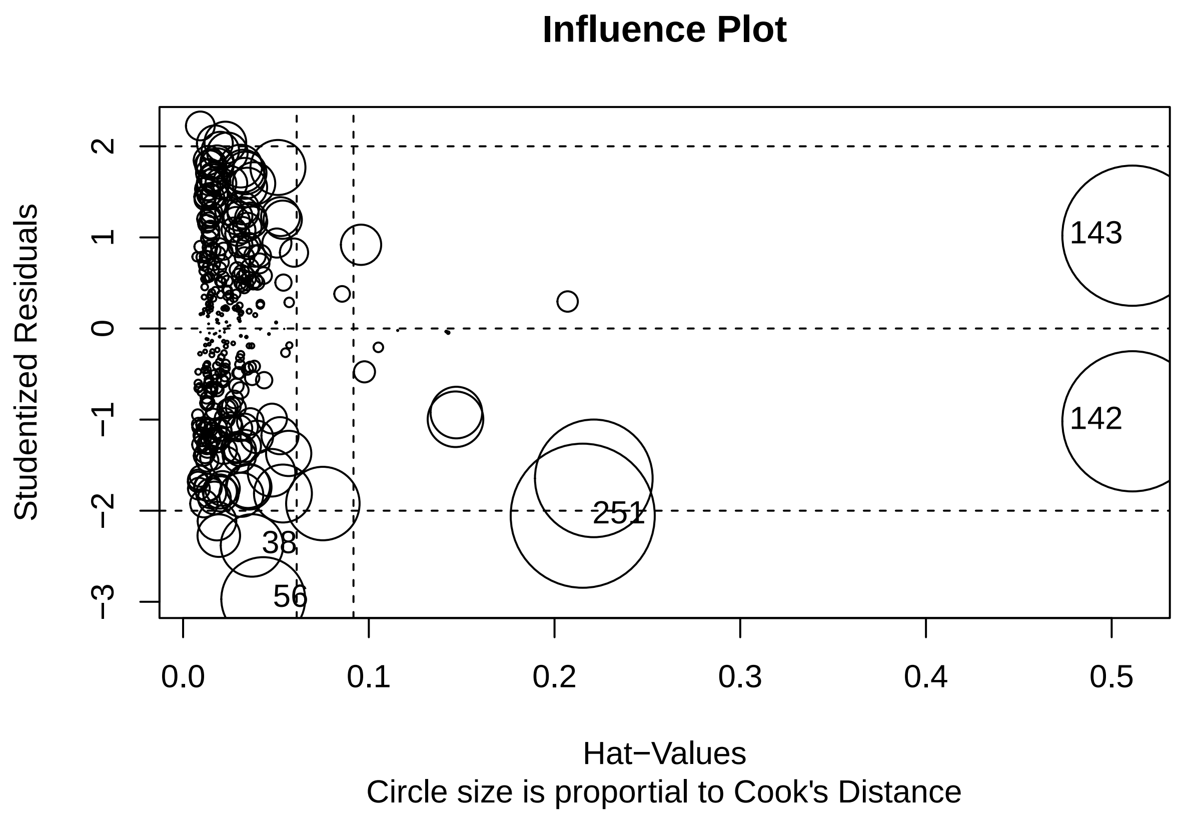
```
## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter
```

```
## Warning in box(...): "id.method" is not a graphical parameter
```

```
## Warning in title(...): "id.method" is not a graphical parameter
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.method" is not a
## graphical parameter
```



	StudRes	Hat	CookD
38	-2.381778	0.0370859	0.0215312
56	-2.970711	0.0431574	0.0388459
142	-1.018872	0.5111820	0.1085464
143	1.018872	0.5111820	0.1085464
251	-2.055209	0.2151787	0.1146427

Normality Test Ho: The data is normally distributed Ha: the data is not normally distributed

```
# Normality Test
shapiro.test(sresid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sresid
## W = 0.99075, p-value = 0.03758
```

```
ad.test(sresid)
```

```
##
##  Anderson-Darling normality test
##
## data:  sresid
## A = 0.59623, p-value = 0.1186
```

```
cvm.test(sresid)
```

```
##
```

```
##  Cramer-von Mises normality test
##
## data:  sresid
## W = 0.075916, p-value = 0.2334
```

```
lillie.test(sresid)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  sresid
## D = 0.038096, p-value = 0.2959
```

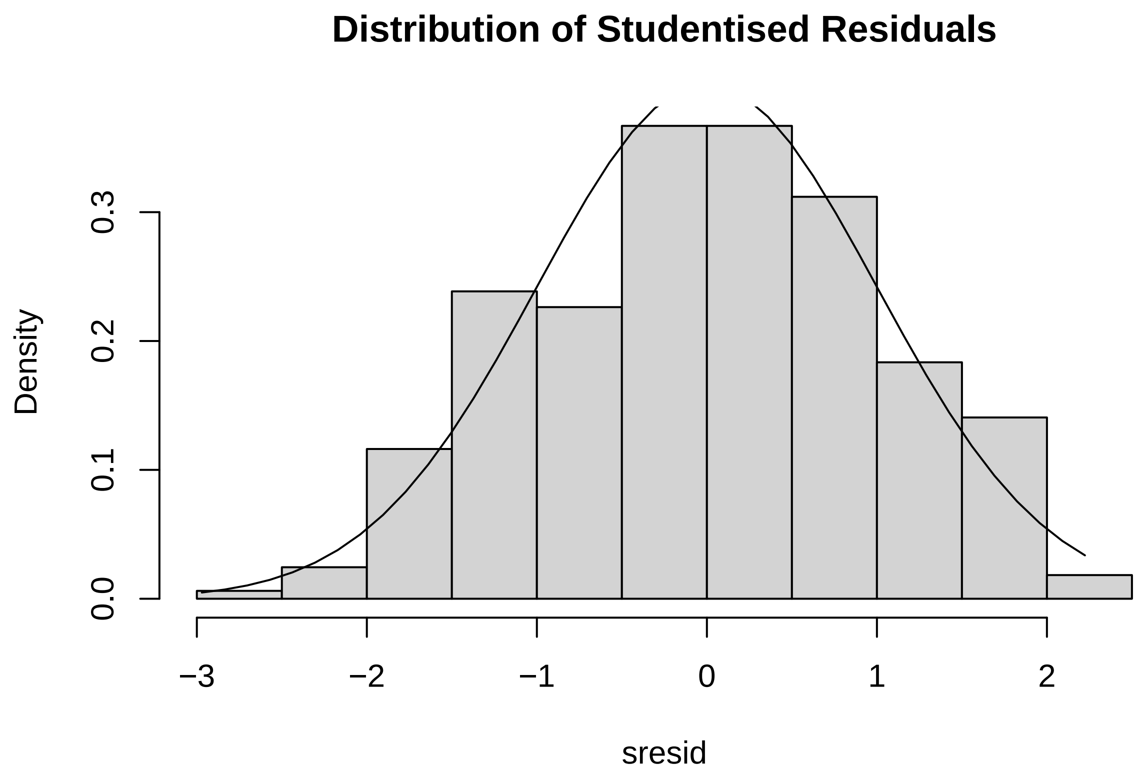
```
pearson.test(sresid)
```

```
##
##  Pearson chi-square normality test
##
## data:  sresid
## P = 21.138, p-value = 0.2725
```

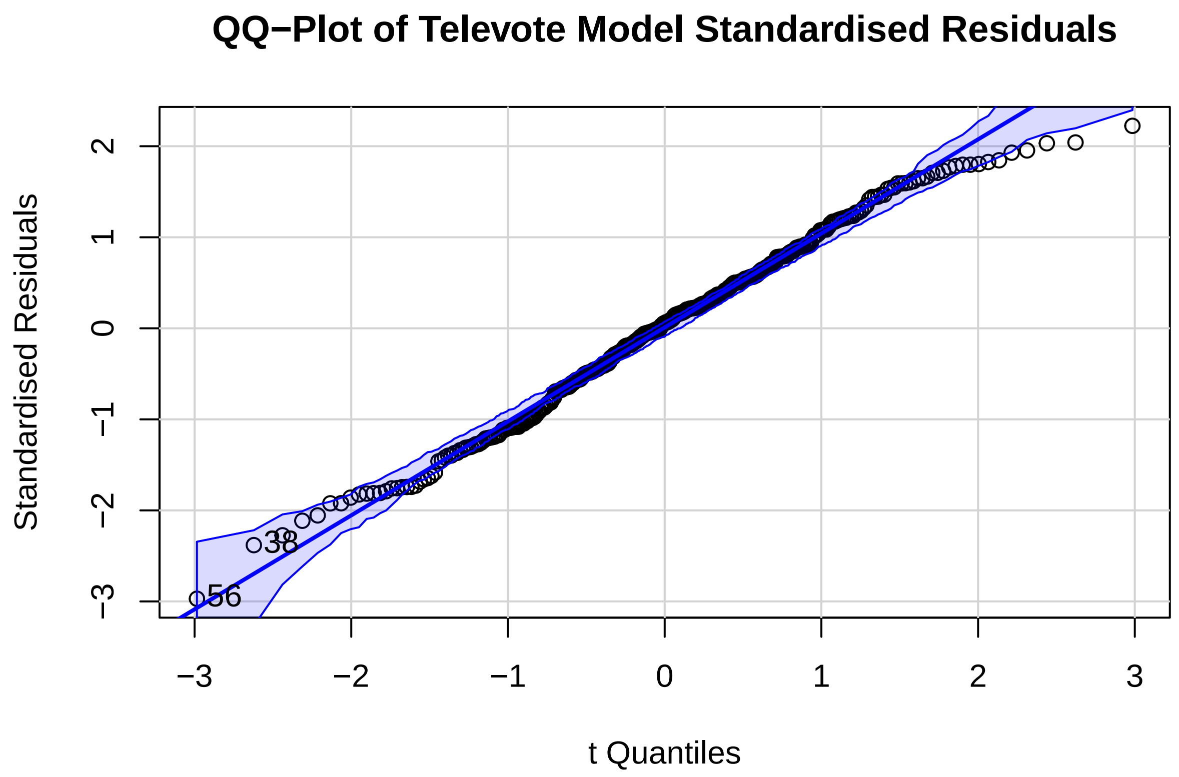
```
sf.test(sresid)
```

```
##
##  Shapiro-Francia normality test
##
## data:  sresid
## W = 0.99202, p-value = 0.07153
```

```
# the data is not normally distributed
# Histogram of residuals
hist(sresid, freq = FALSE, main = "Distribution of Studentised Residuals")
xfit <- seq(min(sresid, na.rm = TRUE), max(sresid, na.rm = TRUE), length = 40)
yfit <- dnorm(xfit)
lines(xfit, yfit)
```



```
# QQ-plot of the data
qqPlot(my_model_tele, ylab = "Standardised Residuals", main = "QQ-Plot of Televote Model Standardised Re
```



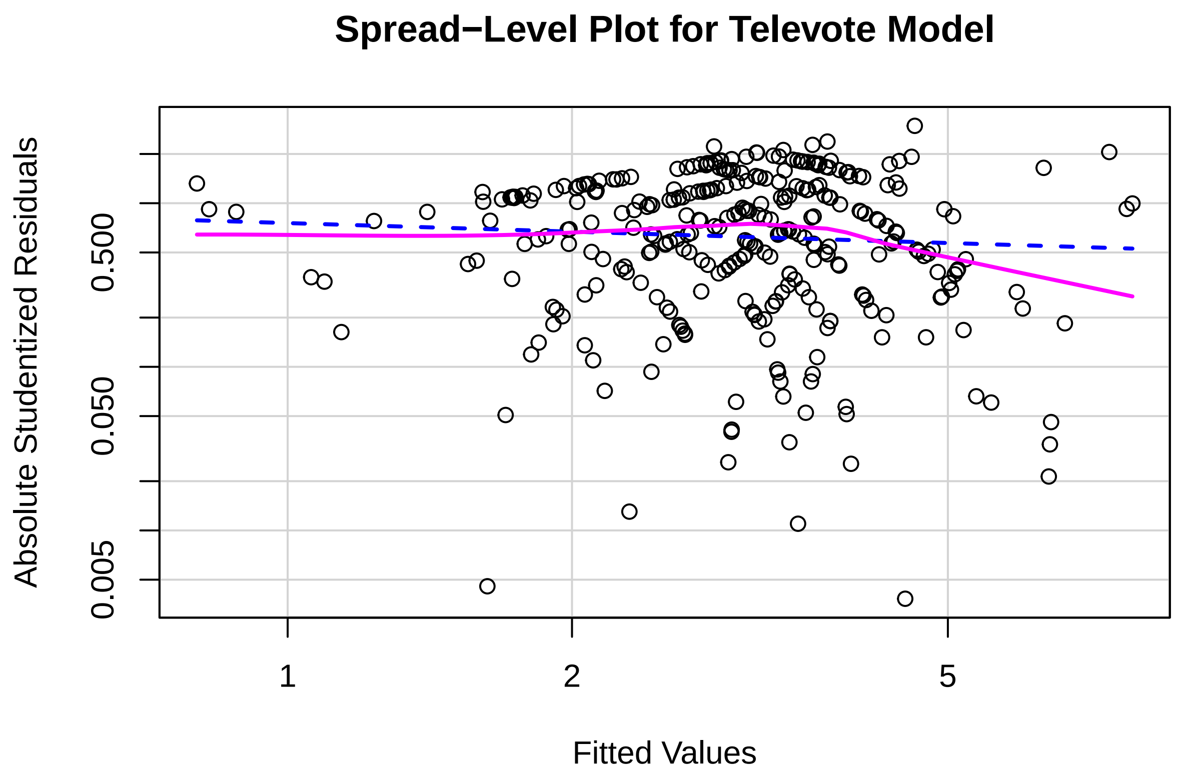
```
## [1] 38 56
```

Non-Constant Error Variance Test Ho: constant error variance Ha: Non-constant error Variance

```
# Non-Constant Error Variance Test
ncvTest(my_model_tele)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.05524184, Df = 1, p = 0.81418
```

```
# plot studentized residuals vs. fitted values
spreadLevelPlot(my_model_tele, main = "Spread-Level Plot for Televote Model")
```



```
##
## Suggested power transformation:  1.174411
```

```
# Variance Inflation Factors
vif(my_model_tele)
```

```
## METRIC_Citizens  Average_Points      TC_NumNeigh      speechiness     acousticness
##        1.440215        1.072688         1.440252         1.231842         1.059780
##    FC_NonCitzens    VBlocs1_TC_13              OOA      CAP_DIST_km
##        1.599009        1.233766         1.147115         1.291723
```

```
sqrt(vif(my_model_tele)) > 2
```

```
## METRIC_Citizens  Average_Points      TC_NumNeigh      speechiness     acousticness
##           FALSE           FALSE            FALSE            FALSE            FALSE
##    FC_NonCitzens    VBlocs1_TC_13              OOA      CAP_DIST_km
##           FALSE           FALSE            FALSE            FALSE
```

```
# No signs of collinearity
```

JURY MODEL

```r
# load final televote model
my_model_jury <- readRDS("./models/jury_final_model.RDS")
# extract out the model coefficients
jury_model_coeff <- names(my_model_jury$coefficients)[-1]
# recreate the model formula
jury_final_model_form<- as.formula(paste('Points ~', paste(jury_model_coeff, collapse = ' + ')))
# generate model summary
summary(my_model_jury)
```

```
##
## Call:
## lm(formula = jury_final_model_form, data = jury_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.136 -2.494 -0.291  2.024  8.297
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.0865     0.4637   8.812  < 2e-16 ***
## CAP_DIST_km            0.6617     0.1854   3.568 0.000414 ***
## acousticness           0.5032     0.1747   2.880 0.004247 **
## speechiness            0.8932     0.2004   4.457 1.15e-05 ***
## TC_PerfType_Mixed     -9.6005     3.2765  -2.930 0.003632 **
## TC_LANGFAM_Armenian   -3.1767     0.9880  -3.215 0.001435 **
## VBlocs1_TC_1           3.0611     0.6177   4.956 1.17e-06 ***
## ComVBlocs1_y          -2.2750     0.6857  -3.318 0.001011 **
## VBlocs1_FC_1           0.8442     0.4283   1.971 0.049601 *
## VBlocs2_TC_1           1.5367     0.4794   3.205 0.001484 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.027 on 321 degrees of freedom
## Multiple R-squared:  0.2132, Adjusted R-squared:  0.1912
## F-statistic: 9.667 on 9 and 321 DF,  p-value: 4.405e-13
```

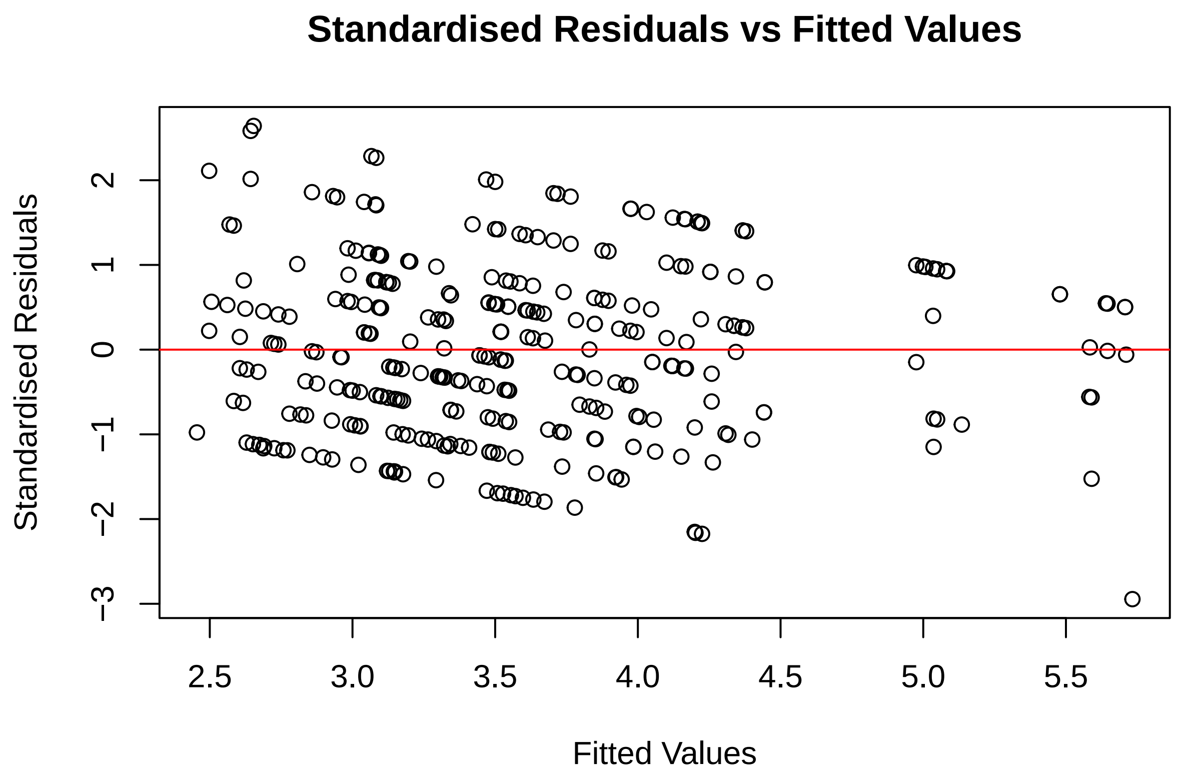
Transformation of Response Variable

```r
# transform points using the optimal power transformation
ptPoints <- jury_data$Points^(3/4)
# Note: weird bug occuring for row name 177 / row index 88 (possibly due to column with near all zero v
jury_coeff_data <- jury_data %>% subset(select = jury_model_coeff)
jury_coeff_zero_prop <- apply(X = jury_coeff_data, MARGIN = 2, FUN = function(x) sum(x == 0)/length(x)*
jury_model_coeff <- names(which(jury_coeff_zero_prop < 99))
# recreate the model formula
jury_final_model_pt_form <- as.formula(paste('ptPoints ~', paste(jury_model_coeff, collapse = ' + ')))
# refit final model with power transformation of 3/4
# NOTE: a box cox transformation resulted in normality but also non-constant variance
my_model_jury <- lm(formula = jury_final_model_pt_form, data = jury_data)
# generate model summary
summary(my_model_jury)
```

```
##
## Call:
## lm(formula = jury_final_model_pt_form, data = jury_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.051 -1.185 -0.045  1.071  3.804
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          2.73582    0.23066  11.861  < 2e-16 ***
## CAP_DIST_km          0.34953    0.09104   3.840 0.000148 ***
## acousticness         0.22920    0.08669   2.644 0.008598 **
## speechiness          0.32980    0.09305   3.544 0.000452 ***
## TC_LANGFAM_Armenian -1.49410    0.49134  -3.041 0.002553 **
## VBlocs1_TC_1         1.41307    0.30606   4.617 5.63e-06 ***
## ComVBlocs1_y        -1.03563    0.34079  -3.039 0.002569 **
## VBlocs1_FC_1         0.41954    0.21305   1.969 0.049787 *
## VBlocs2_TC_1         0.81313    0.23839   3.411 0.000730 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.506 on 322 degrees of freedom
## Multiple R-squared:  0.1852, Adjusted R-squared:  0.165
## F-statistic: 9.149 on 8 and 322 DF,  p-value: 2.378e-11
```

Evaluate the Fit of the Model

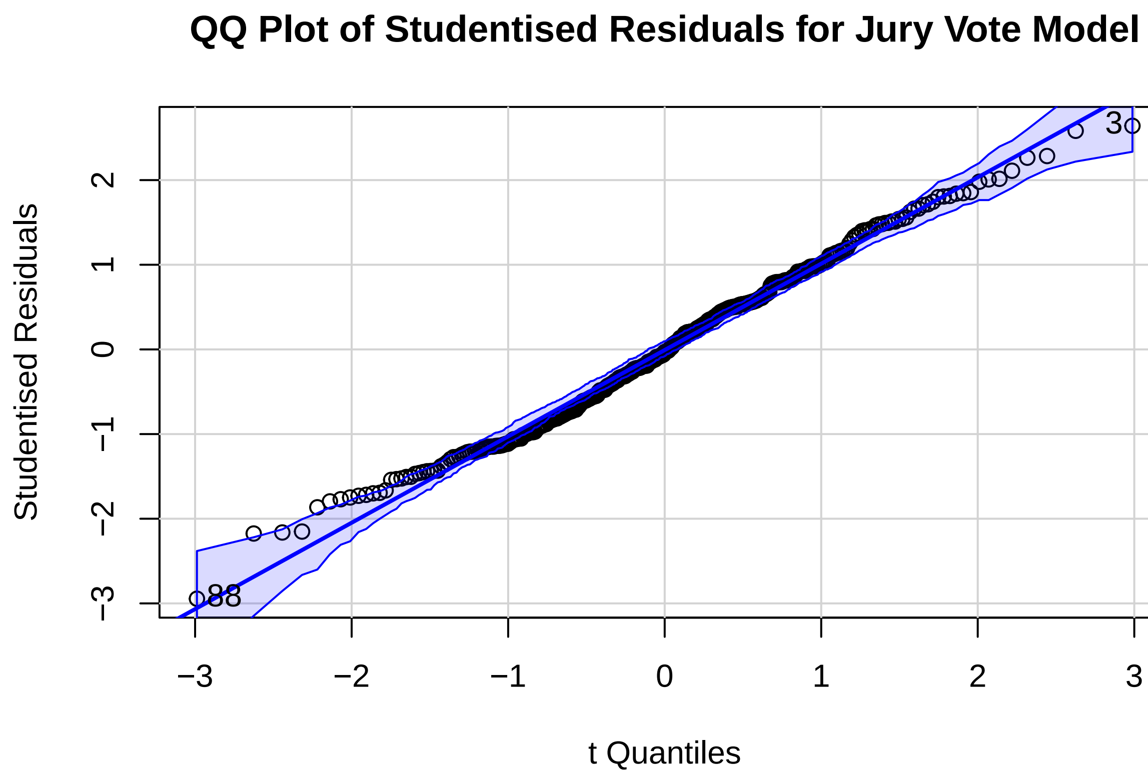
```
# create standardize residuals
sresid <- studres(my_model_jury)
# Residual vs fits plot
plot(x = my_model_jury$fitted.values, y = sresid,  main = "Standardised Residuals vs Fitted Values", xla
# add red horizontal line through y-axis 0
abline(h = 0, col = "red")
```



```
# Assessing Outliers
# Bonferonni p-value for most extreme obs
outlierTest(my_model_jury)
```

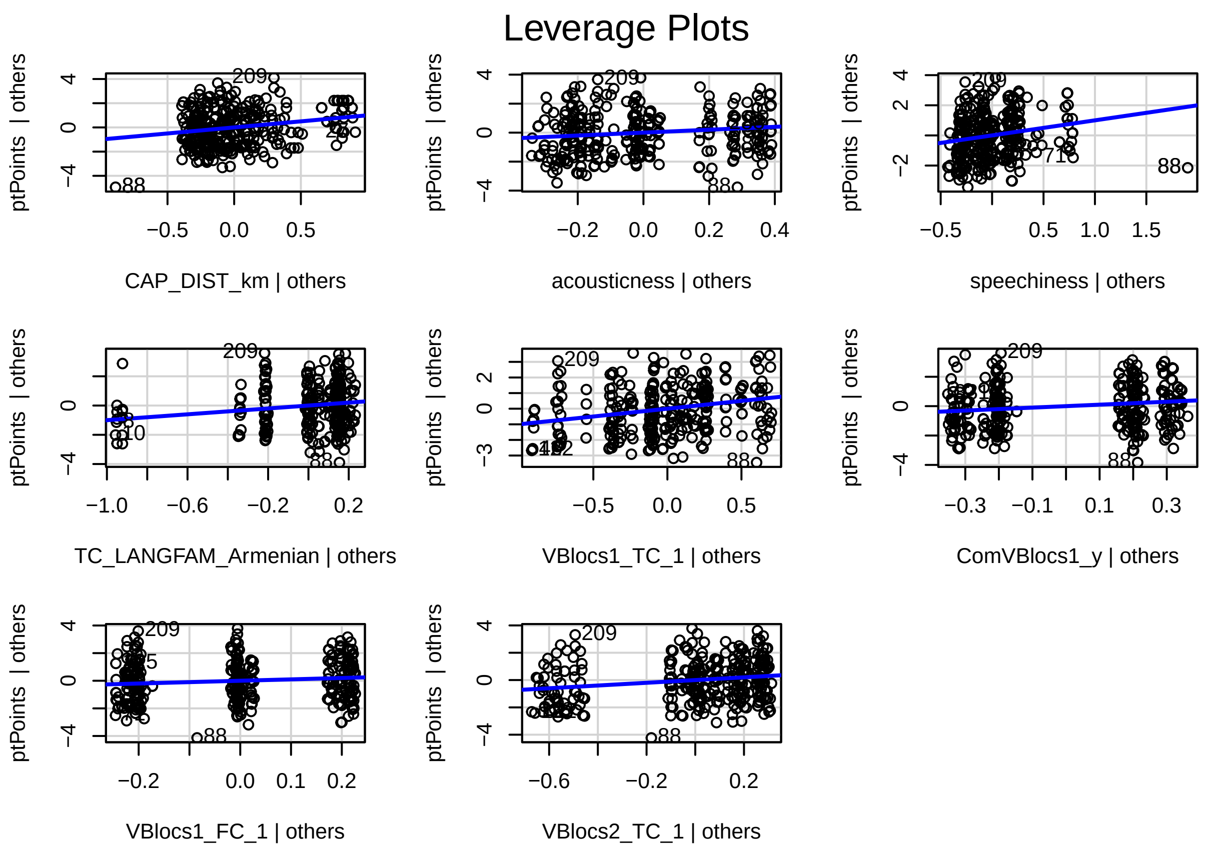
```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 88 -2.945099          0.0034643           NA
```

```
# qq plot for studentized residuals
qqPlot(my_model_jury, main = "QQ Plot of Studentised Residuals for Jury Vote Model", ylab = "Studentise
```

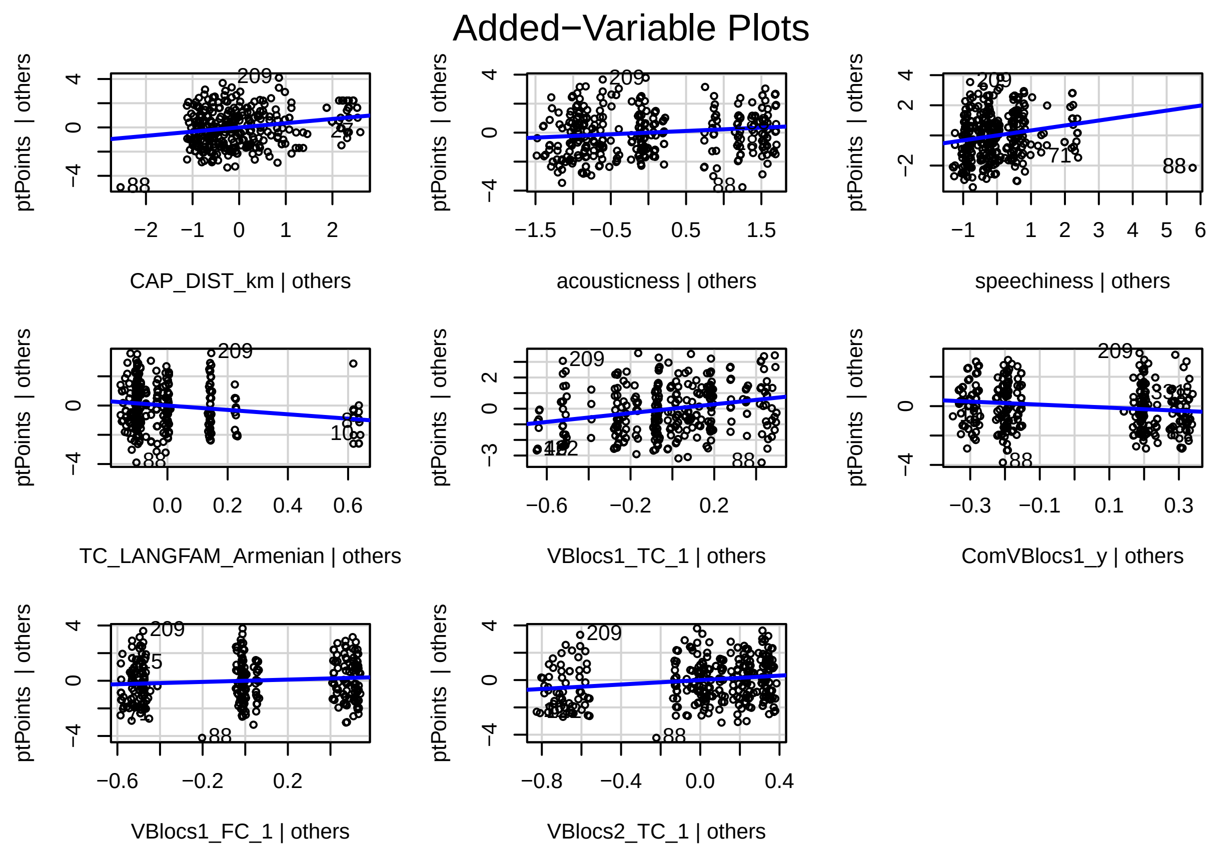


```
## [1]  3 88
```

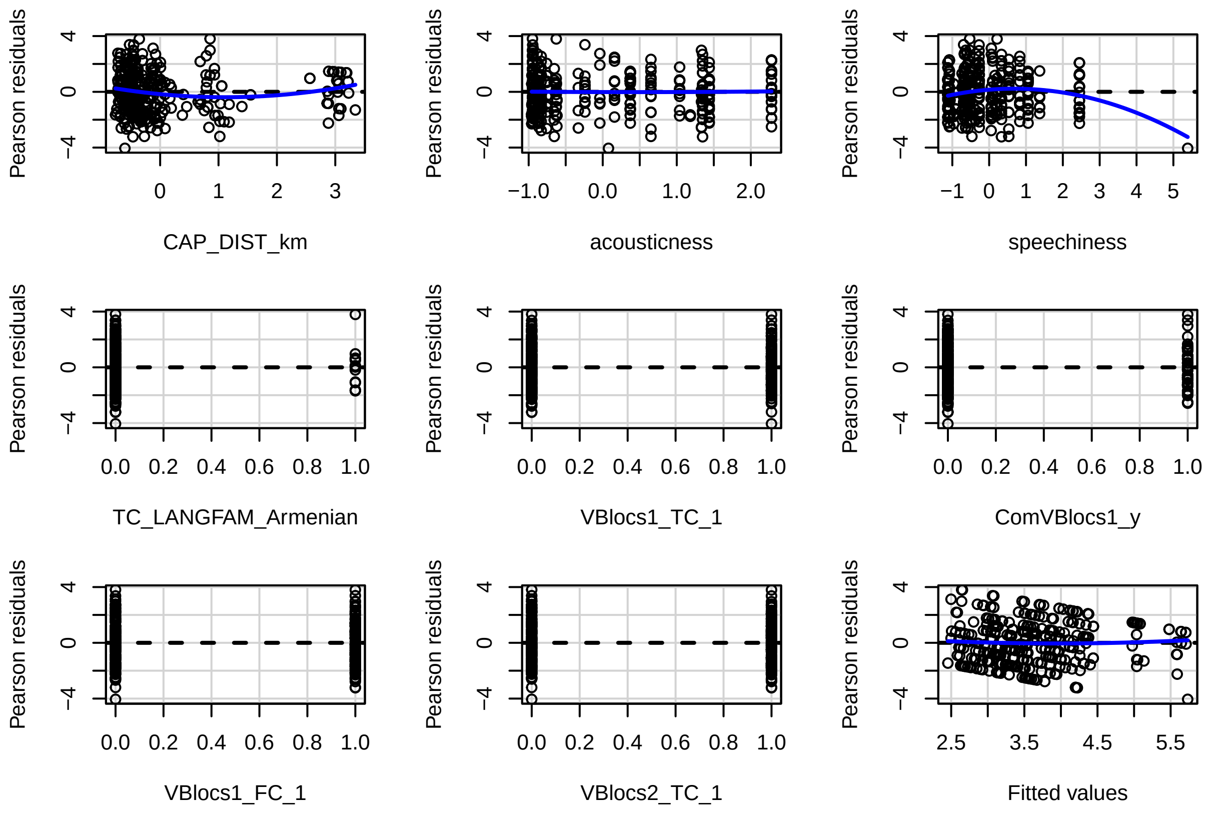
```
# leverage plots
leveragePlots(my_model_jury)
```



```
# Added variable Plots
avPlots(my_model_jury)
```

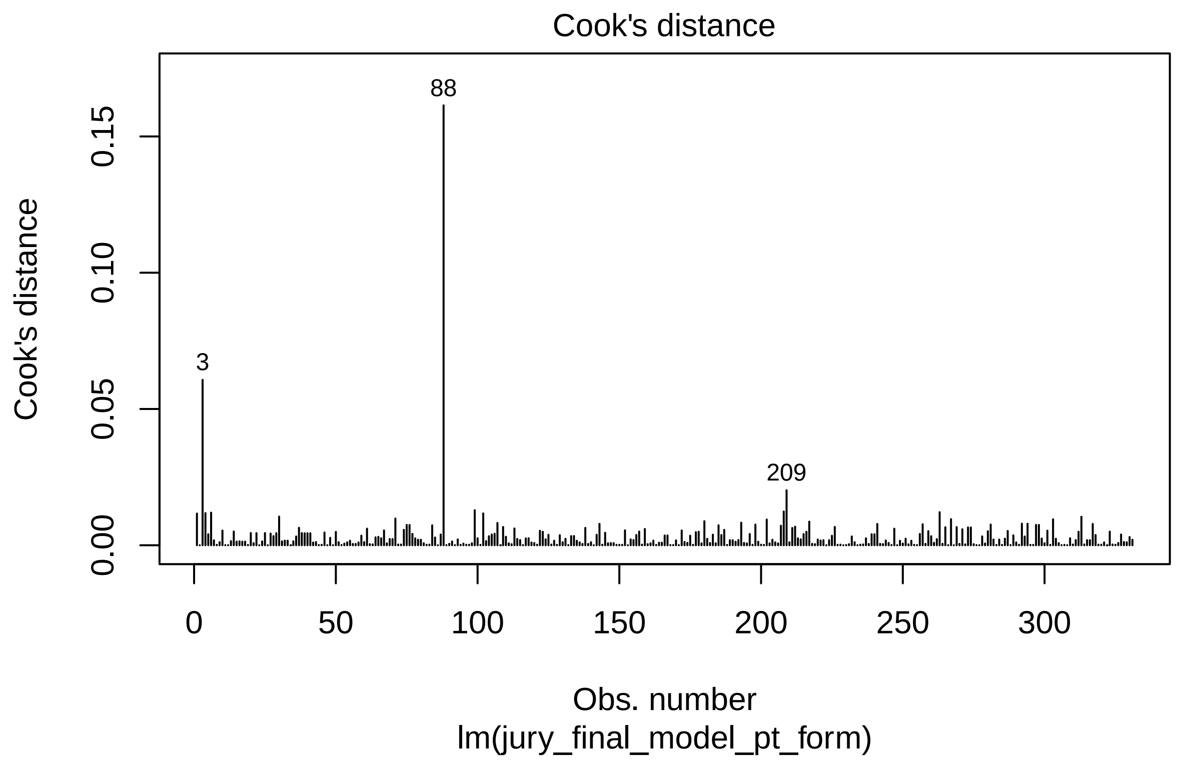


```
# Residual Plots
residualPlots(my_model_jury)
```



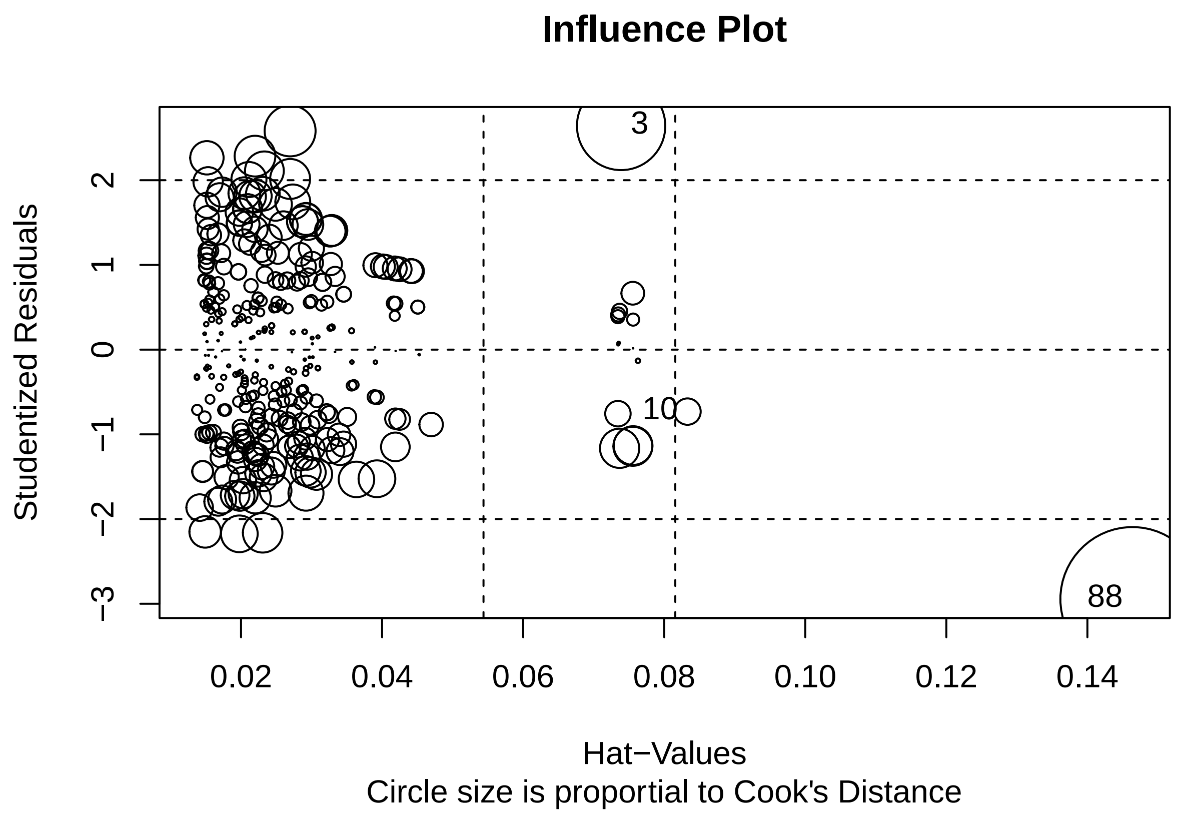
```
##                          Test stat Pr(>|Test stat|)
## CAP_DIST_km                2.8141        0.0051936 **
## acousticness               0.1809        0.8565345
## speechiness               -3.3384        0.0009418 ***
## TC_LANGFAM_Armenian        0.6549        0.5129983
## VBlocs1_TC_1               1.4162        0.1576780
## ComVBlocs1_y              -0.1439        0.8856911
## VBlocs1_FC_1               0.2581        0.7964978
## VBlocs2_TC_1              -0.3352        0.7376926
## Tukey test                 0.8966        0.3699360
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Cook's D plot
# identify D values > 4/(n-k-1)
cutoff <- 4/((nrow(processed_data) - length(my_model_jury$coefficients) - 2))
# Crooks Distance plot
plot(my_model_jury, which = 4, cook.levels = cutoff)
```



```
# Influence Plot
influencePlot(my_model_jury, id.method = "identify", main = "Influence Plot", sub = "Circle size is prop
```

```
## Warning in plot.window(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "id.method" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is not
## a graphical parameter

## Warning in box(...): "id.method" is not a graphical parameter

## Warning in title(...): "id.method" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.method" is not a
## graphical parameter
```



| | StudRes| Hat| CookD| |:--|---------:|---------:|---------:| |3 | 2.6408588| 0.0738904| 0.0607002| |10 | -0.7314985| 0.0833014| 0.0054105| |88 | -2.9450988| 0.1463823| 0.1614188|

Normality Test Ho: The data is normally distributed Ha: the data is not normally distributed

```
# Normality Test
shapiro.test(sresid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sresid
## W = 0.99153, p-value = 0.05511
```

```
ad.test(sresid)
```

```
##
##  Anderson-Darling normality test
##
## data:  sresid
## A = 0.89016, p-value = 0.02271
```

```
cvm.test(sresid)
```

```
##
```

```
##  Cramer-von Mises normality test
##
## data:  sresid
## W = 0.13742, p-value = 0.03478
```

```
lillie.test(sresid)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  sresid
## D = 0.047301, p-value = 0.0716
```

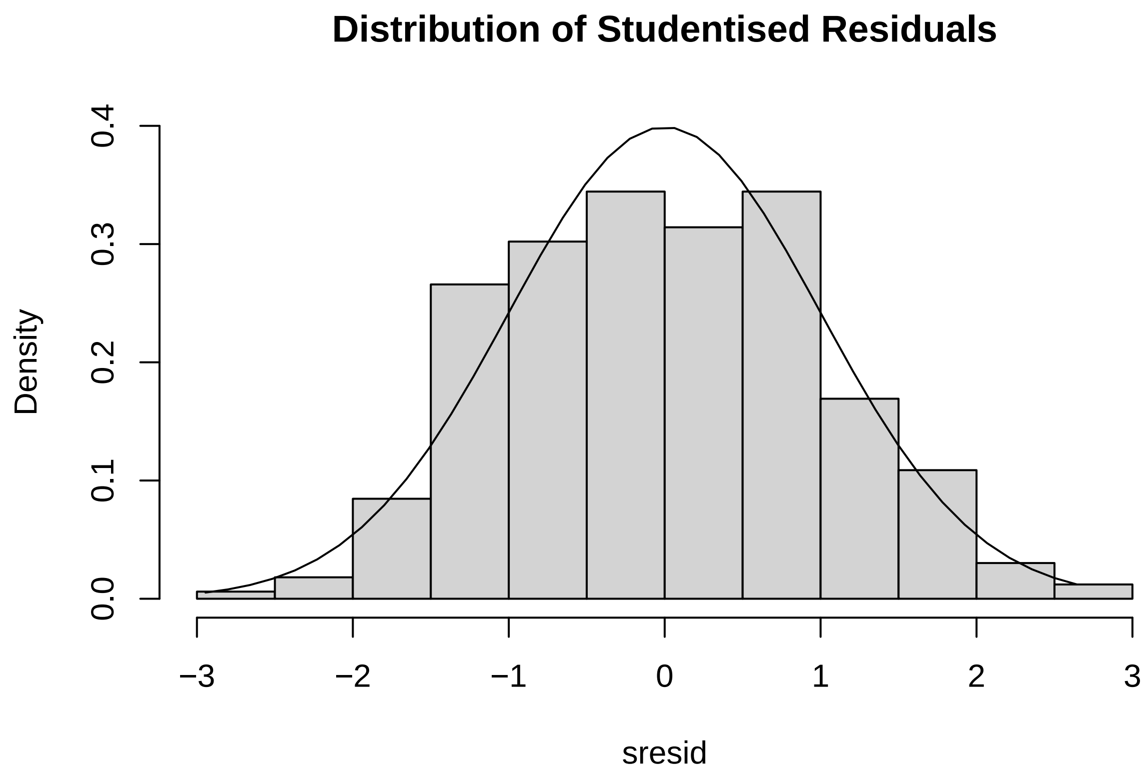
```
pearson.test(sresid)
```

```
##
##  Pearson chi-square normality test
##
## data:  sresid
## P = 30.568, p-value = 0.03228
```

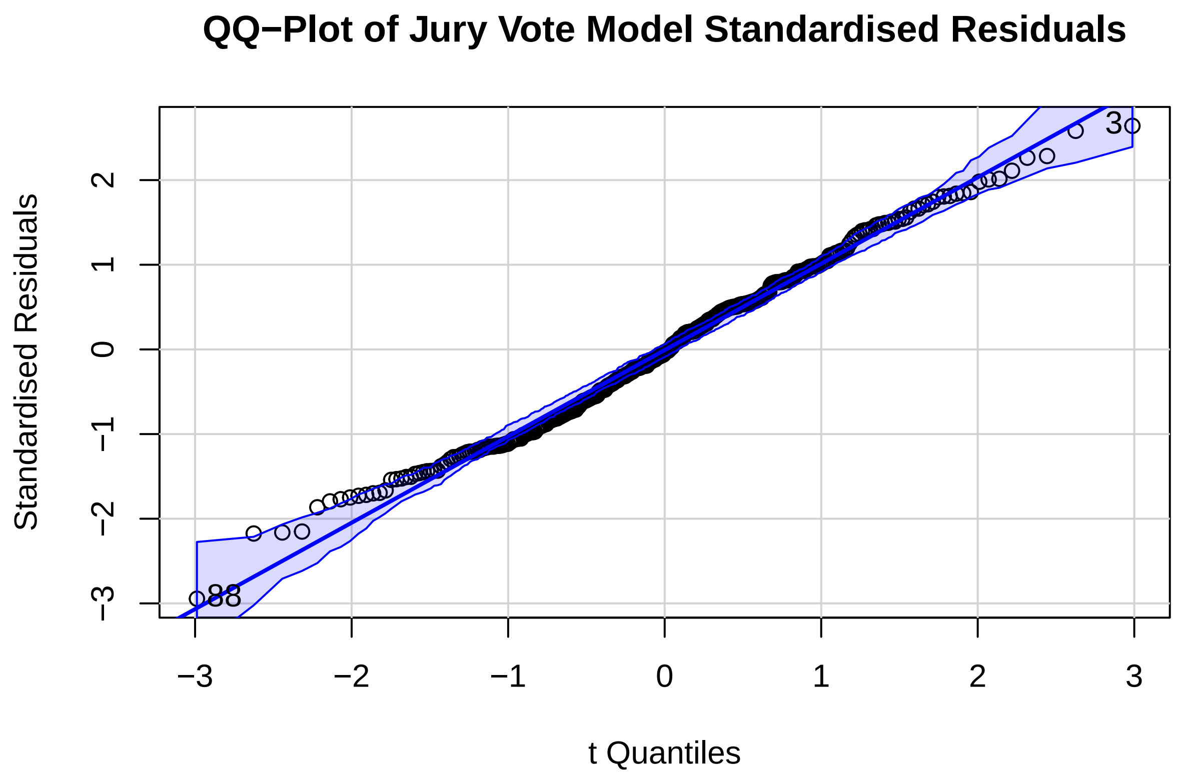
```
sf.test(sresid)
```

```
##
##  Shapiro-Francia normality test
##
## data:  sresid
## W = 0.99221, p-value = 0.07562
```

```
# the data is not normally distributed
# Histogram of residuals
hist(sresid, freq = FALSE, main = "Distribution of Studentised Residuals", ylim = c(0, 0.4))
xfit <- seq(min(sresid, na.rm = TRUE), max(sresid, na.rm = TRUE), length = 40)
yfit <- dnorm(xfit)
lines(xfit, yfit)
```



```
# QQ-plot of the data
qqPlot(my_model_jury, ylab = "Standardised Residuals", main = "QQ-Plot of Jury Vote Model Standardised R
```



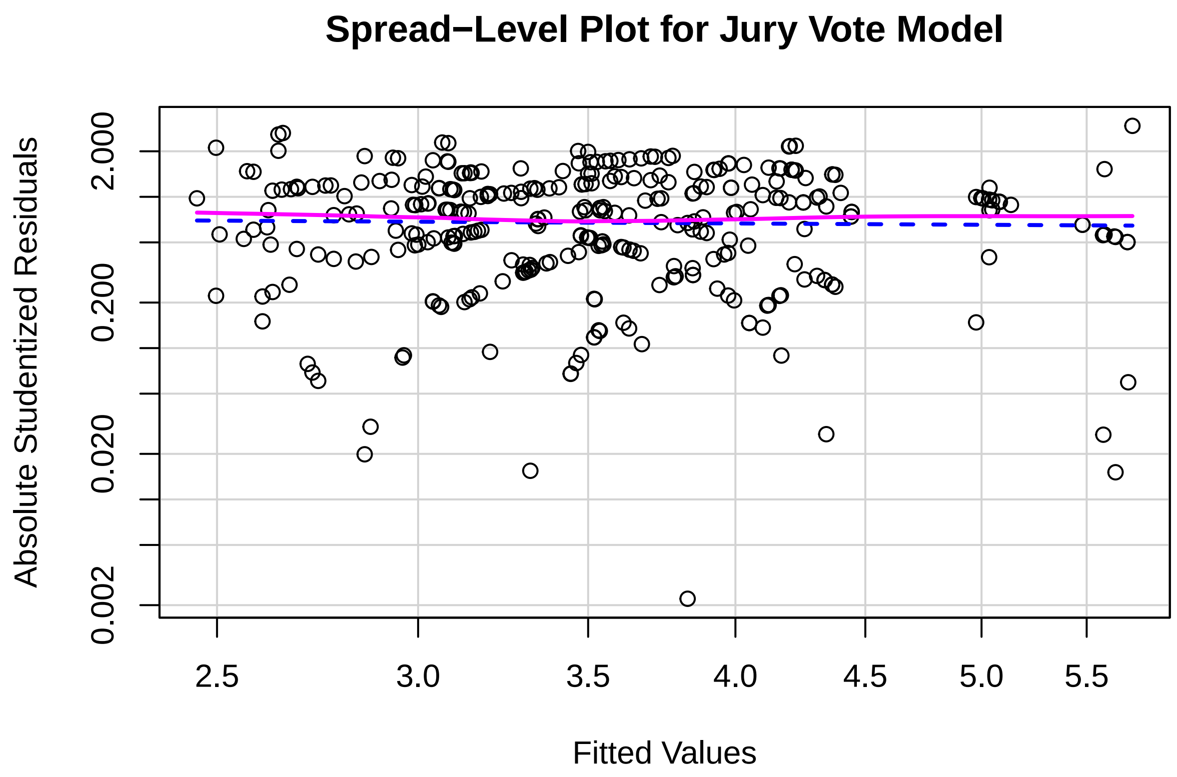
```
## [1]  3 88
```

Non-Constant Error Variance Test Ho: constant error variance Ha: Non-constant error Variance

```
# Non-Constant Error Variance Test
ncvTest(my_model_jury)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.2153073, Df = 1, p = 0.64264
```

```
# plot studentized residuals vs. fitted values
spreadLevelPlot(my_model_jury, main = "Spread-Level Plot for Jury Vote Model")
```



```
##
## Suggested power transformation:  1.091406
```

```
# Variance Inflation Factors
vif(my_model_jury)
```

```
##         CAP_DIST_km        acousticness         speechiness TC_LANGFAM_Armenian
##            1.181514            1.220899            1.187223            1.523284
##        VBlocs1_TC_1        ComVBlocs1_y        VBlocs1_FC_1        VBlocs2_TC_1
##            3.249223            2.641876            1.654895            2.067919
```

```
sqrt(vif(my_model_jury)) > 2
```

```
##         CAP_DIST_km        acousticness         speechiness TC_LANGFAM_Armenian
##               FALSE               FALSE               FALSE               FALSE
##        VBlocs1_TC_1        ComVBlocs1_y        VBlocs1_FC_1        VBlocs2_TC_1
##               FALSE               FALSE               FALSE               FALSE
```

```
# No signs of collinearity
```