LASSO Addendum

Francisco N. Ramos

Addendum: Using LASSO as Feature Selection for Psychologists in R

As an addendum, I will walk through the use of LASSO for feature selection on a psychological dataset in R. LASSO is a regression method that performs variable selection and regularization. LASSO is essentially an extension of ordinary least squares where a penalty term is added to the loss function, resulting in "weightings" of coefficients. The penalty term in LASSO (L1 regularization) uses absolute value, whereas the penalty term in the very-similar ridge regression uses squared values (L2 regularization, which we will discuss in the adaptive LASSO section below). LASSO can reduce a predictor's weight to 0, removing it from the model, which makes it more suitable for feature selection than ridge regression. The latter is best for handling highly collinear predictors, which we will also check soon since LASSO can handle multicollinearity, but only to an extent.

LASSO uses a least-squares "shrinkage operator", so we should make sure we can fit a linear model with our AMATUS data first. We'll check assumptions and conduct exploratory data analysis. It's more okay if the data seems to be less fitting of a linear model after LASSO calculation (more on this later), but we definitely want to make sure we could fit a linear model with our 10 variables as is.

As part of the AMATUS dataset, the original researchers already calculated a correlation matrix for the variables we're interested in. To save you time, I'll tell you right now that the variables that are correlated with each other to a potentially problematic extent are math self-efficacy and self-concept positively (duh) and math self-concept as well as math self-efficacy with math anxiety, negatively. This shouldn't cause too many issues because LASSO and ridge regression can handle multicollinearity, but it's likely that in some models, one of these variables will be picked and the other will be ignored since they cover much of the same ground. So don't be surprised if one LASSO model picks for example SDQ_M and another picks PISA_ME, as they explain much of the same variance.

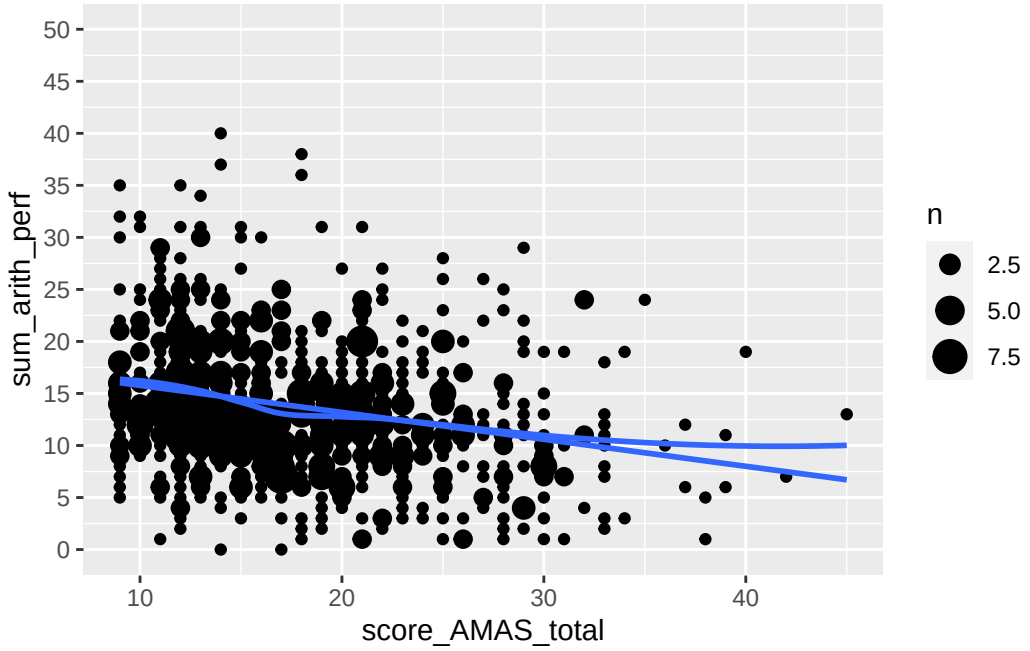
Real quick, let's load the data again. Same dataset as the random forest document, so use that as reference.

```r
#load packages
library(here)
## here() starts at /Users/frankie/Desktop/Dr. Lai Feature Selection Project 9.13.24/Main Sc

library(tidyverse)
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to b
library(caret)
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
library(readxl)
library(readr)
library(ggplot2)
library(randomForest)
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
library(stats)

# load dataset
```

```r
file_path <- normalizePath(here::here("OSF archive/AMATUS_dataset.csv"))
# adjust as needed.

amatus <- read_csv2(file_path, c("", "NA"), col_names = TRUE)
## i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
# it's already in the "working directory" of the main script folder,
# so we only need the subfolder OSF archive and the actual name of the dataset.

View(amatus)
# We can use this command to view our data.
table(amatus$sum_arith_perf)
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  4 15 12 18 15 19 24 40 40 53 62 68 71 64 57 45 42 37 18 29 36 25 23 16 18 16
## 26 27 28 29 30 31 32 34 35 36 37 38 40
##  6  8  4  3  8  6  2  1  2  1  2  1  1

#### Just some regular old data cleaning to start with.
amatus$sex <- as.factor(amatus$sex)
amatus$age_range <- as.factor(amatus$age_range)
amatus$breaks <- as.factor(amatus$breaks)
amatus$honesty <- as.factor(amatus$honesty)
amatus$native_speaker <- as.factor(amatus$native_speaker)
amatus$noise <- as.factor(amatus$noise)

amatusclean <- amatus[!is.na(amatus$sum_arith_perf), ] # removing the
# individuals who did not complete the performance test in order as instructed
amatusclean <- amatusclean[!(amatusclean$sample %in%
  c("german_teachers", "belgian_teachers")), ]

# removing the other two samples from the dataset
View(amatusclean)
```

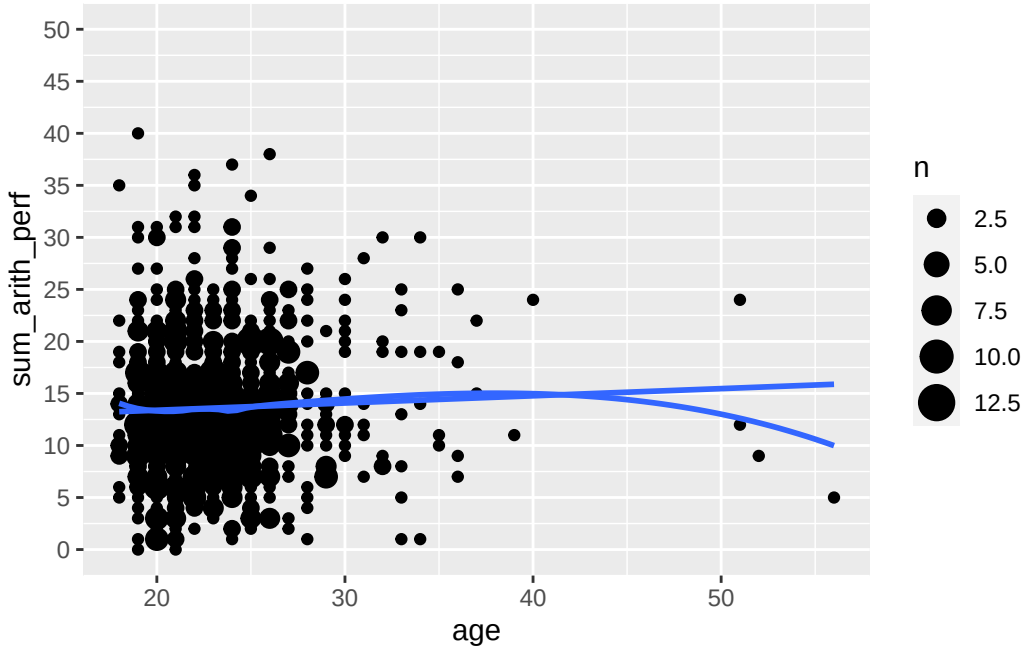
```r
library(ggplot2)
library(olsrr) # nice tools for linear regression
##
## Attaching package: 'olsrr'
## The following object is masked from 'package:datasets':
##
##     rivers
library(caret)
library(MASS)
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:olsrr':
##
##     cement
## The following object is masked from 'package:dplyr':
##
##     select
library(dplyr)
library(car)
## Loading required package: carData
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
## The following object is masked from 'package:purrr':
##
##     some
library(glmnet)
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
## Loaded glmnet 4.1-8

AMASplot <- ggplot(data = amatusclean, aes(x = score_AMAS_total, y = sum_arith_perf))
AMASplot <- AMASplot + geom_point(fill = "lightskyblue1", color = "black")
AMASplot <- AMASplot + geom_count(show.legend = TRUE)
AMASplot <- AMASplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
AMASplot <- AMASplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
AMASplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



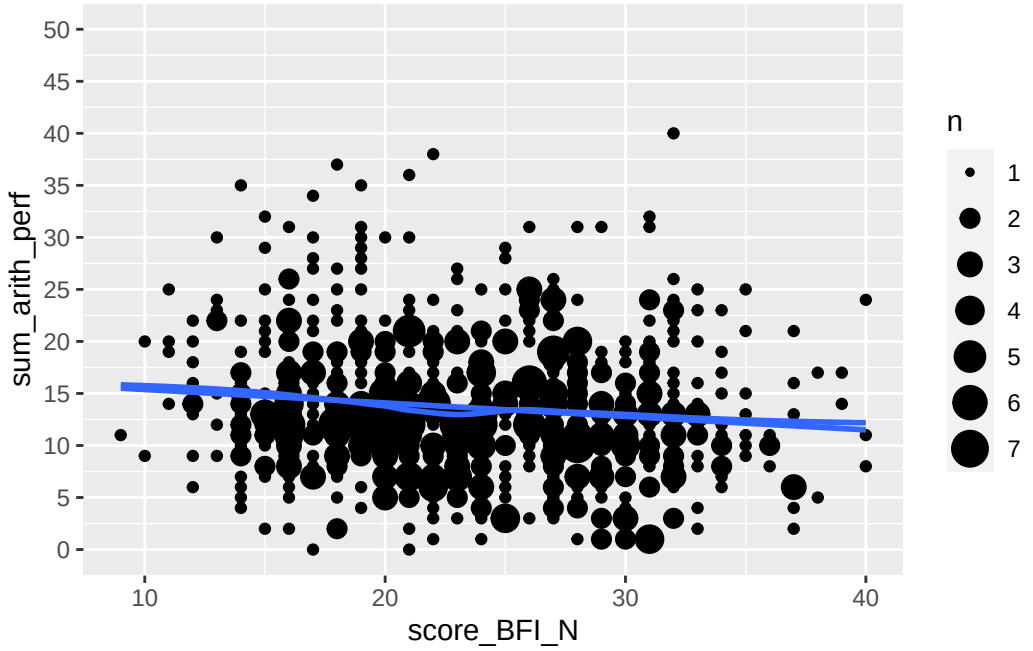

```
# modestly linear, right tail deviates.
```

```
lassoamatus <- amatusclean
ageplot <- ggplot(data = lassoamatus, aes(x = age, y = sum_arith_perf))
ageplot <- ageplot + geom_point(fill = "lightskyblue1", color = "black")
ageplot <- ageplot + geom_count(show.legend = TRUE)
ageplot <- ageplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
ageplot <- ageplot + geom_smooth(method = "lm", se = FALSE)
ageplot <- ageplot + geom_smooth(method = "loess", se = FALSE)
ageplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



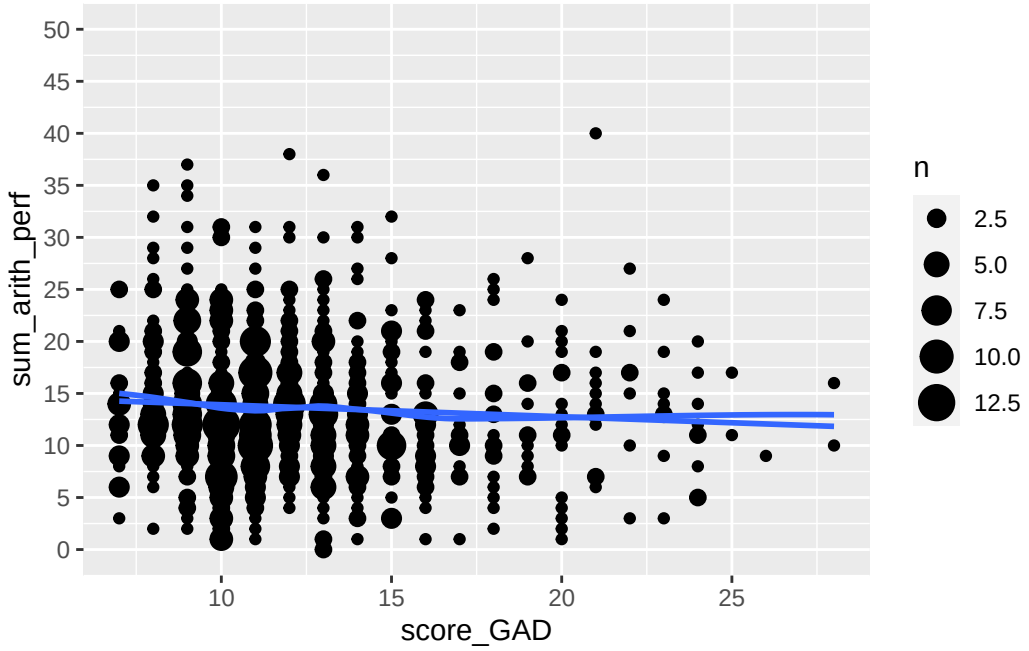

```
# notreallylinear. A log transformation makes it look a little better.
```

```
bfiplot <- ggplot(data = lassoamatus, aes(x = score_BFI_N, y = sum_arith_perf))
bfiplot <- bfiplot + geom_point(fill = "lightskyblue1", color = "black")
bfiplot <- bfiplot + geom_count(show.legend = TRUE)
bfiplot <- bfiplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
bfiplot <- bfiplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
bfiplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



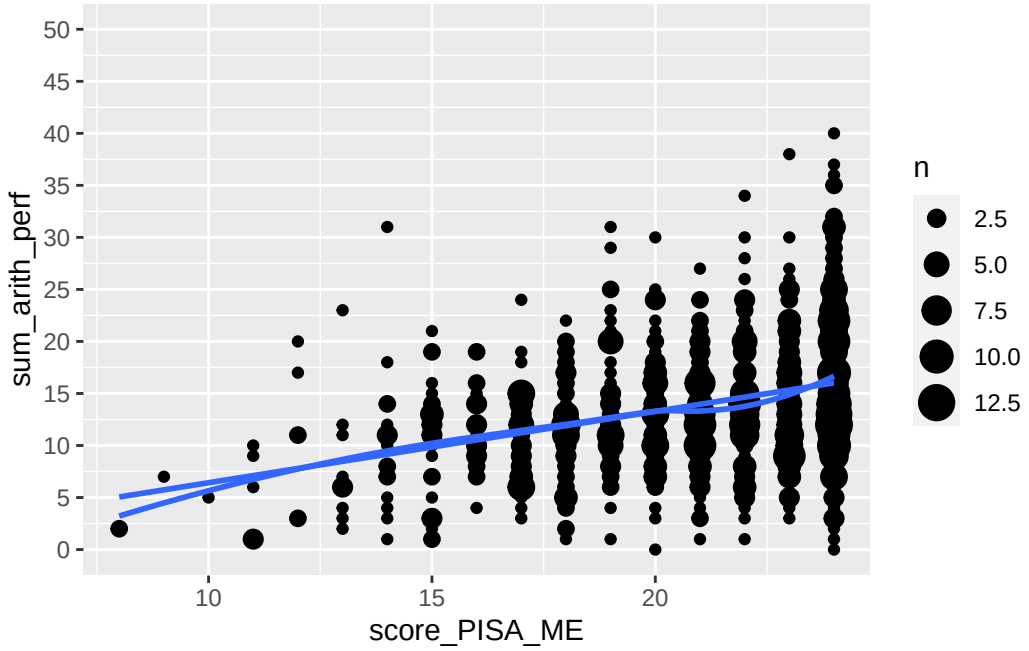

```
# Pretty damn linear.
```

```
gadplot <- ggplot(data = lassoamatus, aes(x = score_GAD, y = sum_arith_perf))
gadplot <- gadplot + geom_point(fill = "lightskyblue1", color = "black")
gadplot <- gadplot + geom_count(show.legend = TRUE)
gadplot <- gadplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
gadplot <- gadplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
gadplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



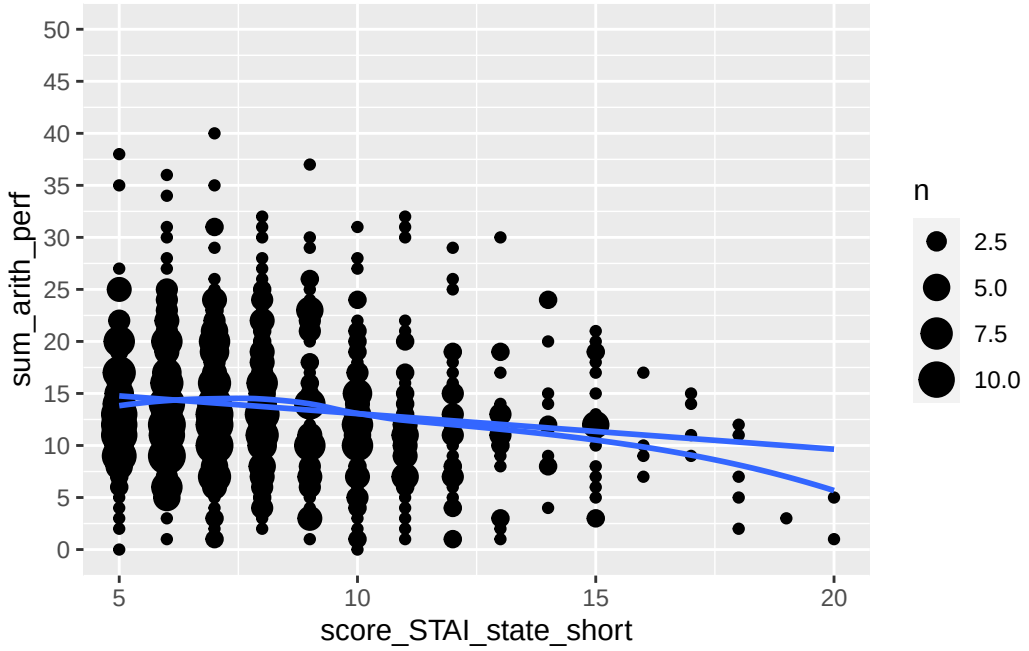

```
# modestly linear?
```

```
PISAplot <- ggplot(data = lassoamatus, aes(x = score_PISA_ME, y = sum_arith_perf))
PISAplot <- PISAplot + geom_point(fill = "lightskyblue1", color = "black")
PISAplot <- PISAplot + geom_count(show.legend = TRUE)
PISAplot <- PISAplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
PISAplot <- PISAplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
PISAplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



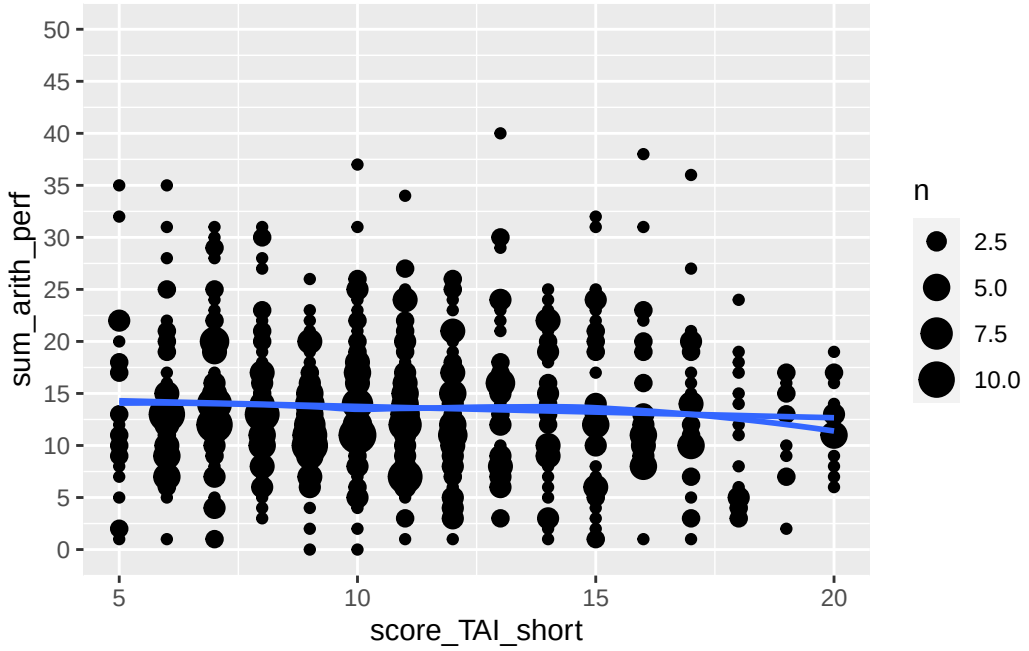

```
# modestly linear.
```

```
STAIplot <- ggplot(data = lassoamatus, aes(x = score_STAI_state_short, y = sum_arith_perf))
STAIplot <- STAIplot + geom_point(fill = "lightskyblue1", color = "black")
STAIplot <- STAIplot + geom_count(show.legend = TRUE)
STAIplot <- STAIplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
STAIplot <- STAIplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
STAIplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



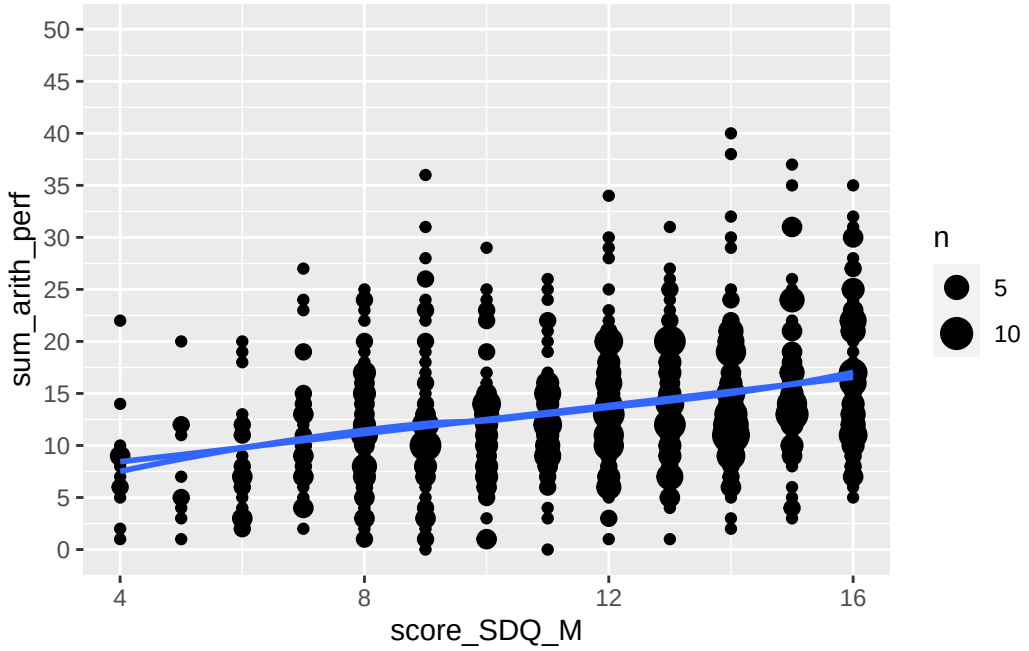

```
# modestly linear, big deviation at right tail.
```

```
TAIplot <- ggplot(data = lassoamatus, aes(x = score_TAI_short, y = sum_arith_perf))
TAIplot <- TAIplot + geom_point(fill = "lightskyblue1", color = "black")
TAIplot <- TAIplot + geom_count(show.legend = TRUE)
TAIplot <- TAIplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
TAIplot <- TAIplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
TAIplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



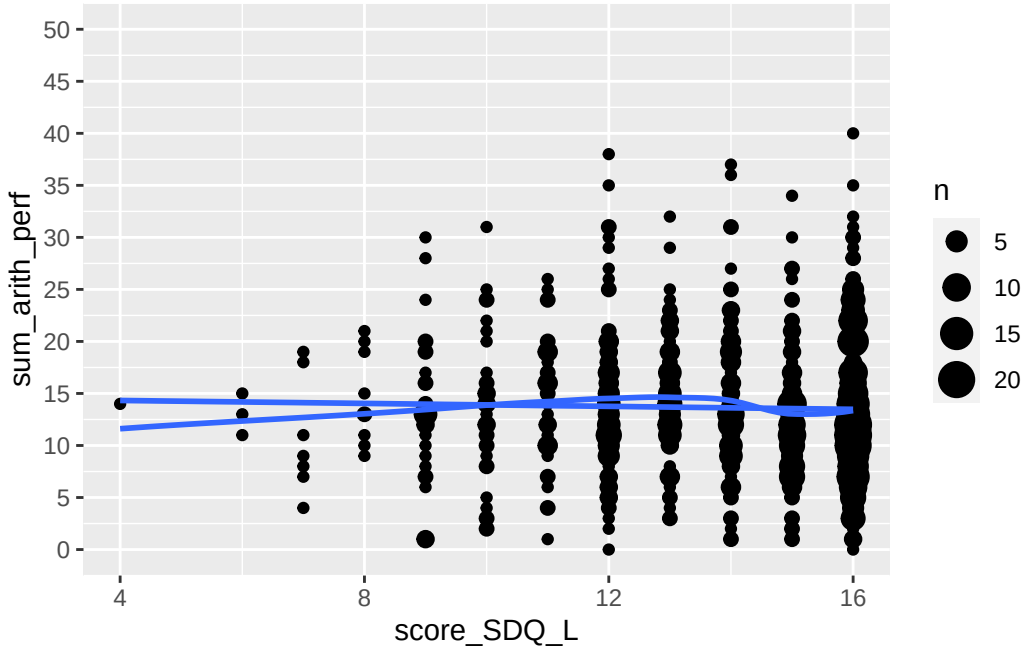

```
# Pretty linear, tiny deviation at right tail.
```

```
SDQ_Mplot <- ggplot(data = lassoamatus, aes(x = score_SDQ_M, y = sum_arith_perf))
SDQ_Mplot <- SDQ_Mplot + geom_point(fill = "lightskyblue1", color = "black")
SDQ_Mplot <- SDQ_Mplot + geom_count(show.legend = TRUE)
SDQ_Mplot <- SDQ_Mplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
SDQ_Mplot <- SDQ_Mplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
SDQ_Mplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




```
## Vert linear, small deviation at left tail.
```

```
SDQ_Lplot <- ggplot(data = lassoamatus, aes(x = score_SDQ_L, y = sum_arith_perf))
SDQ_Lplot <- SDQ_Lplot + geom_point(fill = "lightskyblue1", color = "black")
SDQ_Lplot <- SDQ_Lplot + geom_count(show.legend = TRUE)
SDQ_Lplot <- SDQ_Lplot + scale_y_continuous(breaks = seq(0, 50, 5), limits = c(0, 50))
SDQ_Lplot <- SDQ_Lplot + geom_smooth(method = "lm", se = FALSE) +
  geom_smooth(method = "loess", se = FALSE)
SDQ_Lplot
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```




```
# doesn't seem really linear, to be honest.
```

Honestly, most of these seem pretty linear, albeit with slight deviations at the ends for some. SDQ_L is pretty clearly not linear. We would probably want to use some nonlinear modeling techniques in a real analysis, something like splines or piecewise linear regression. But that is outside the scope of the paper, so we'll stick with it for now.

#can maybe try splines. B-spline would have a penalty already. Is it used with LASSO or as a substitute for LASSO?

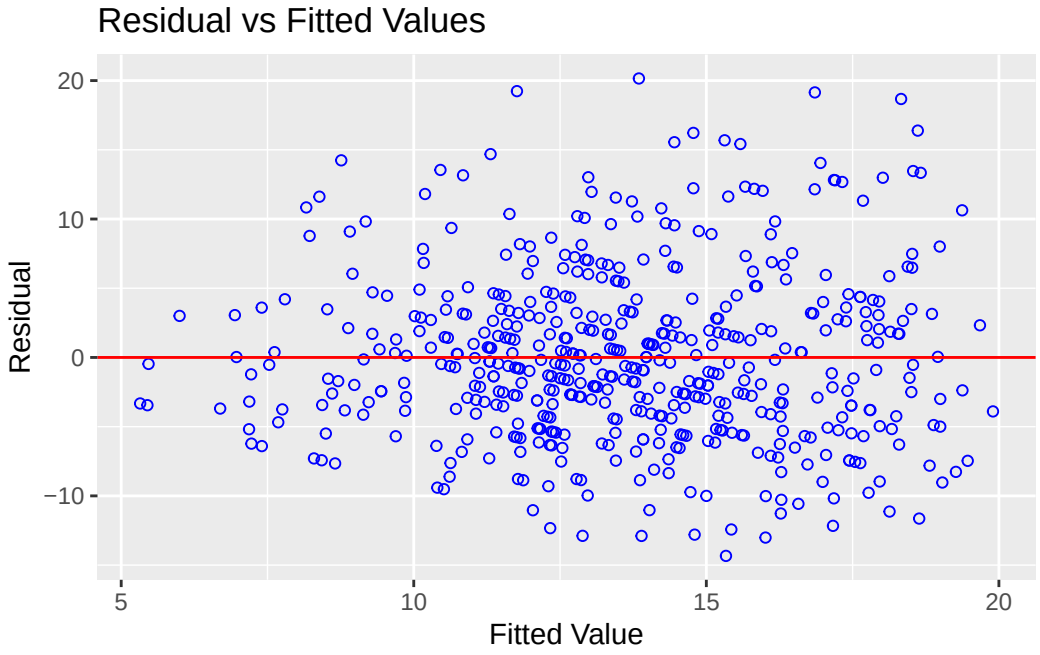
Running the Model

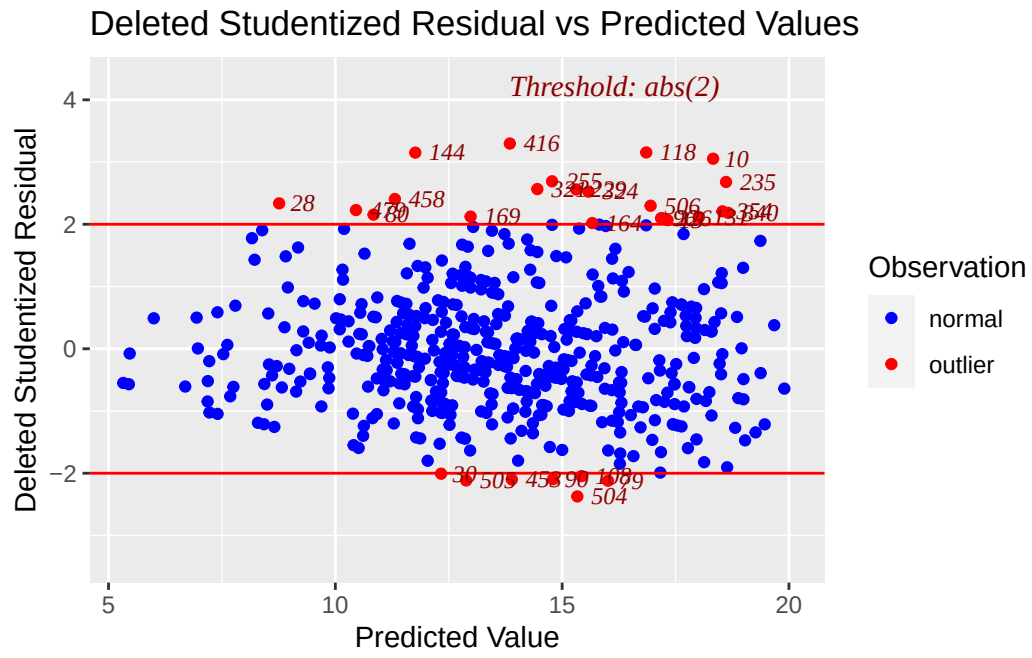
Normally, you should naturally transform your variables to the desired distributions then run the model. In order to show you why you should do that, I will first run the model with no transformations.

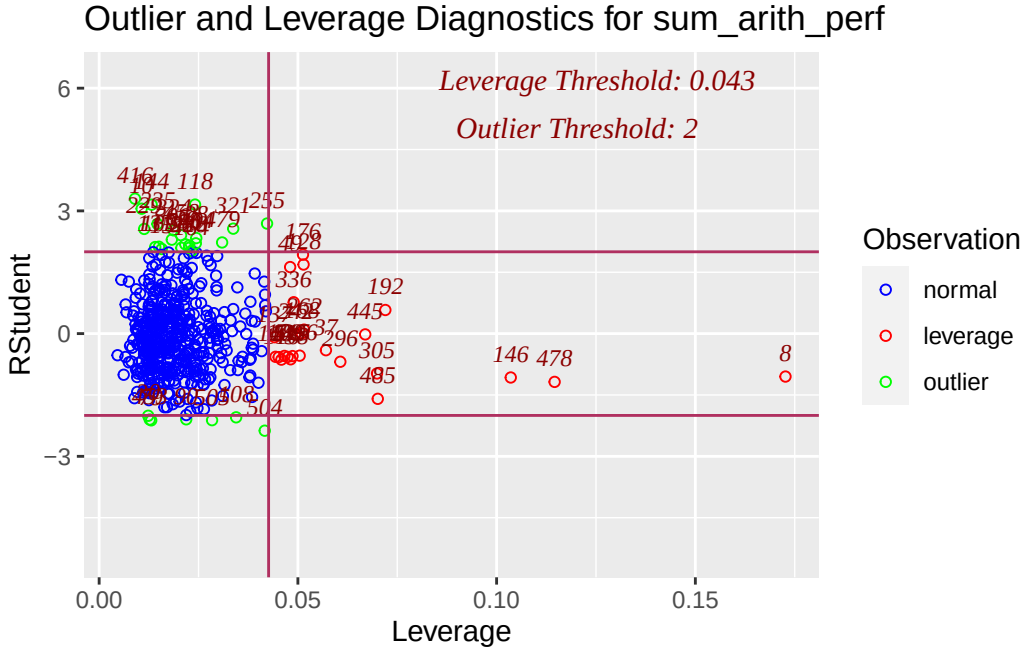
```
set.seed(150) # I'm going to put this before every random generation just for clarity.
inTrain <- createDataPartition(amatusclean$sum_arith_perf, p = 0.7, list = FALSE) # 0.7 sele

# create training vs test data.
training2 <- amatusclean[inTrain, ]
test2 <- amatusclean[-inTrain, ]
View(training2)
```

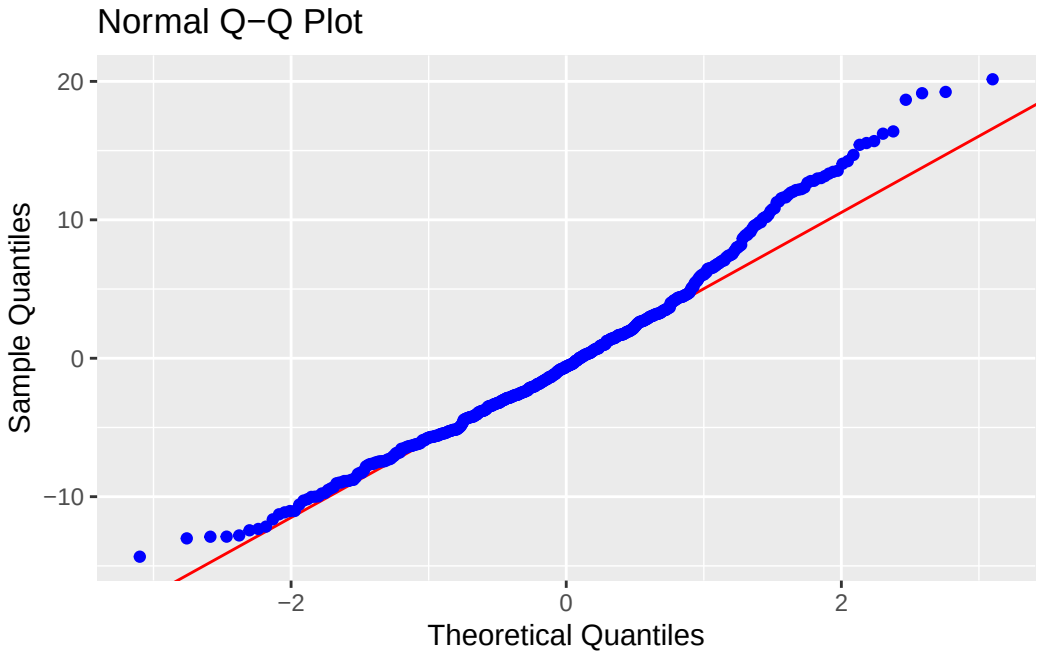
```r
View(test2)


# Extracting the dataframe to keep things simple.
# Define the variable names
numeric_predictors <- as.character(c(
  "score_BFI_N", "score_AMAS_total", "age", "score_GAD",
  "score_PISA_ME", "score_STAI_state_short",
  "score_TAI_short", "score_SDQ_L", "score_SDQ_M"
))
categorical_variable <- as.character("sex")
response_variable <- as.character("sum_arith_perf")

# Create a new dataframe by scaling and centering numeric predictors
scaled_training2 <- training2 %>%
  # Select only specified columns using all_of
  dplyr::select(all_of(c(numeric_predictors, categorical_variable, response_variable))) %>%
  # Identify numeric columns
  dplyr::mutate(across( # notice we put dplyr:: in front so it doesn't think it is using MASS
    # Specify numeric columns for scaling
    .cols = all_of(numeric_predictors), # Use all_of to ensure these are the numeric predicto
    .fns = ~ scale(.) %>% as.vector() # Scale and convert to vector
  ))


# View the scaled data
View(scaled_training2)


lambdamodel <- lm(sum_arith_perf ~ score_AMAS_total + sex + age + score_BFI_N + score_GAD + s
summary(lambdamodel)
##
## Call:
## lm(formula = sum_arith_perf ~ score_AMAS_total + sex + age +
##     score_BFI_N + score_GAD + score_PISA_ME + score_STAI_state_short +
##     score_TAI_short + score_SDQ_M + score_SDQ_L, data = scaled_training2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3354  -4.2040  -0.6142   3.2286  20.1525
##
## Coefficients:
```

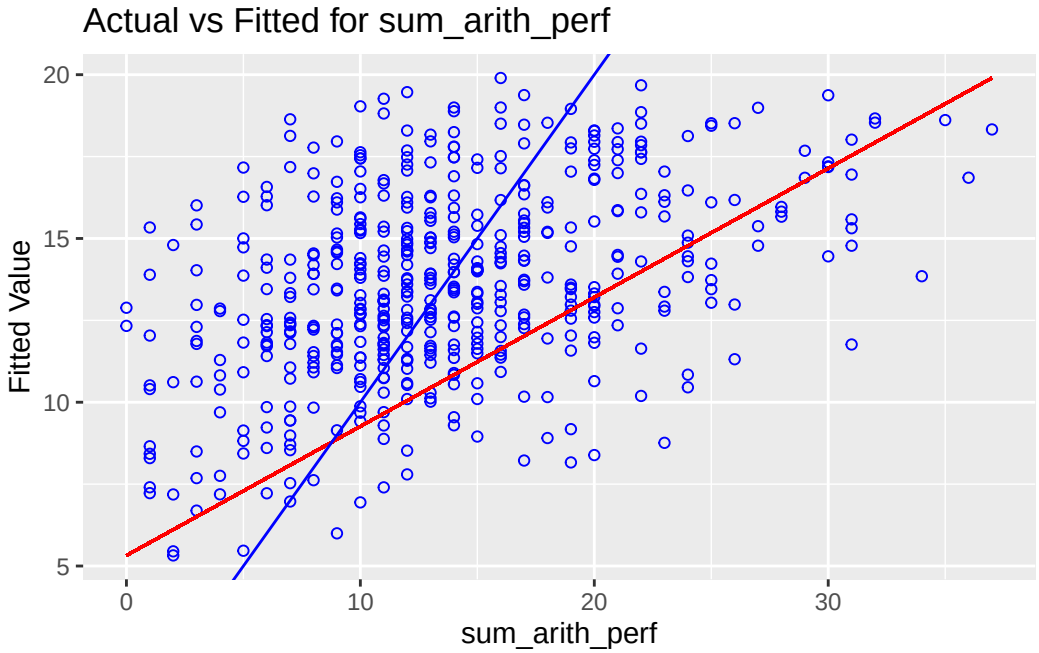
```
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              12.5812569  0.3399185  37.013  < 2e-16 ***
## score_AMAS_total         -0.3091363  0.4247484  -0.728  0.46707
## sexm                      3.3317267  0.6417090   5.192 3.02e-07 ***
## age                      -0.0077477  0.2836661  -0.027  0.97822
## score_BFI_N               0.0002823  0.3674932   0.001  0.99939
## score_GAD                 0.0701379  0.3711814   0.189  0.85020
## score_PISA_ME             1.0539930  0.3604607   2.924  0.00361 **
## score_STAI_state_short -0.3361031  0.3257345  -1.032  0.30264
## score_TAI_short           0.2983691  0.3150757   0.947  0.34410
## score_SDQ_M               1.1732538  0.4028332   2.913  0.00374 **
## score_SDQ_L               0.3443142  0.2845448   1.210  0.22682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.198 on 505 degrees of freedom
## Multiple R-squared:  0.1841, Adjusted R-squared:  0.168
## F-statistic:  11.4 on 10 and 505 DF,  p-value: < 2.2e-16
## see residual/fitted plots, qq plots, and much more
ols_plot_diagnostics(lambdamodel)
```

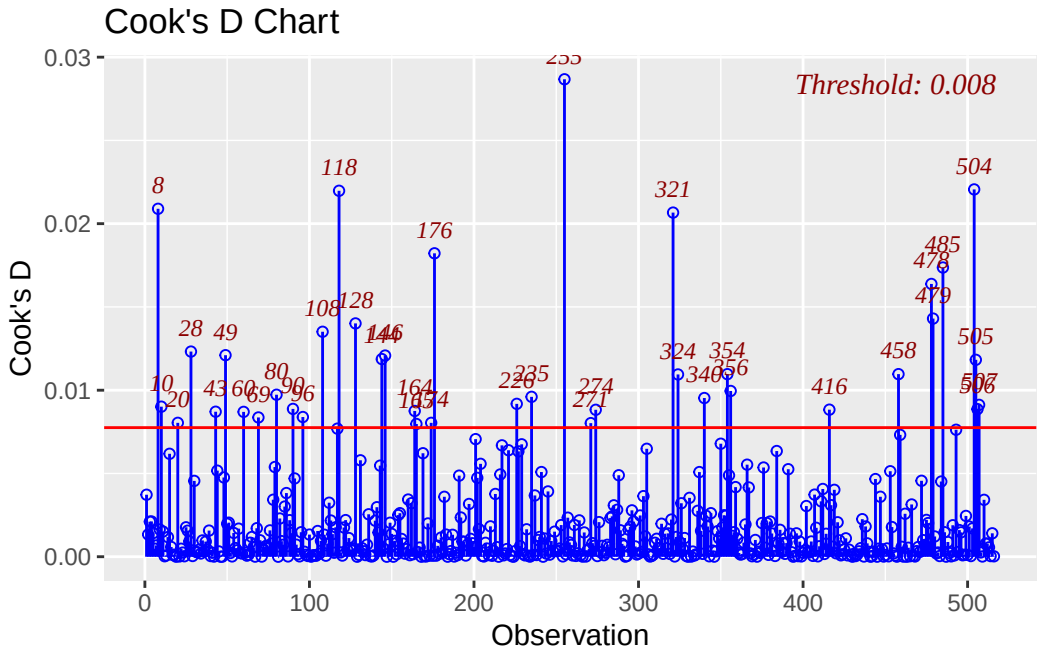


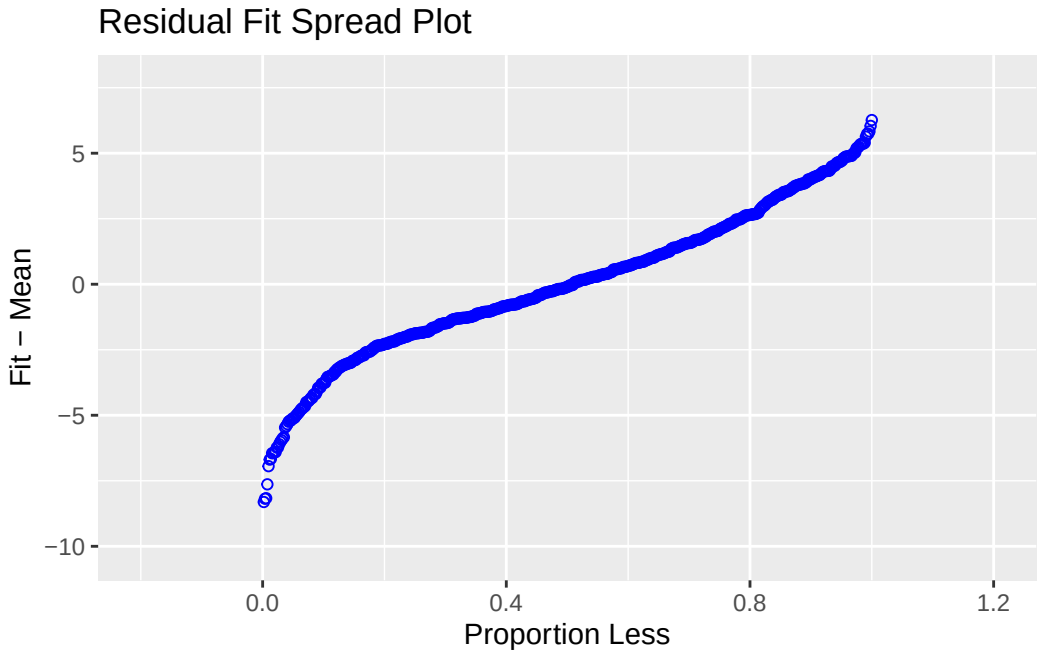

Deleted Studentized Residual vs Predicted Values
Threshold: abs(2)
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier

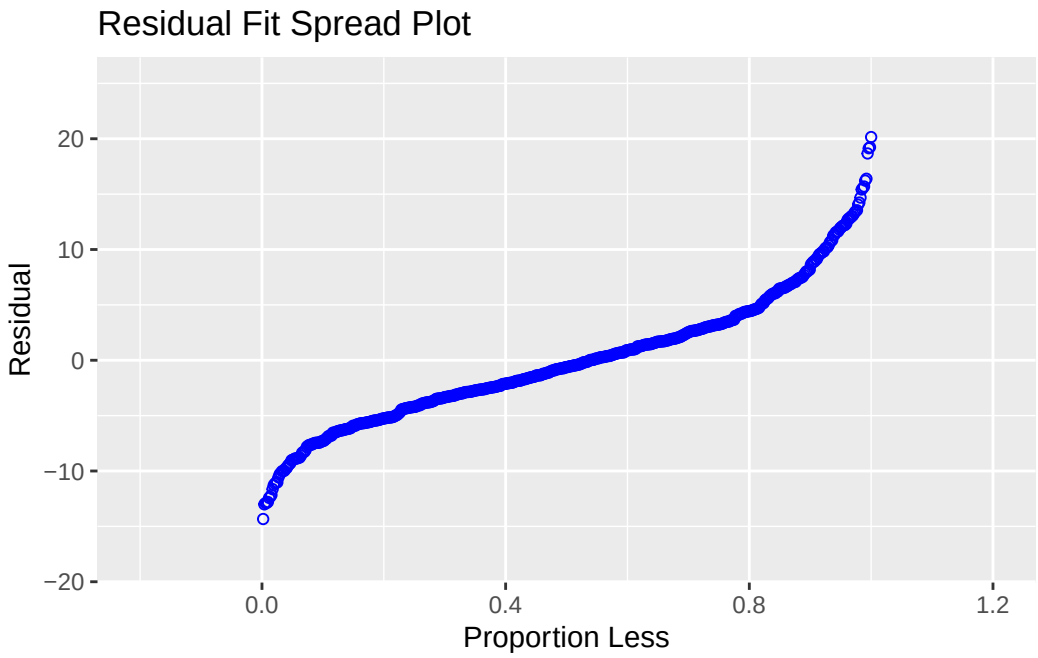

Outlier and Leverage Diagnostics for sum_arith_perf
Leverage Threshold: 0.043
Outlier Threshold: 2
RStudent
Leverage
Observation
normal
leverage
outlier

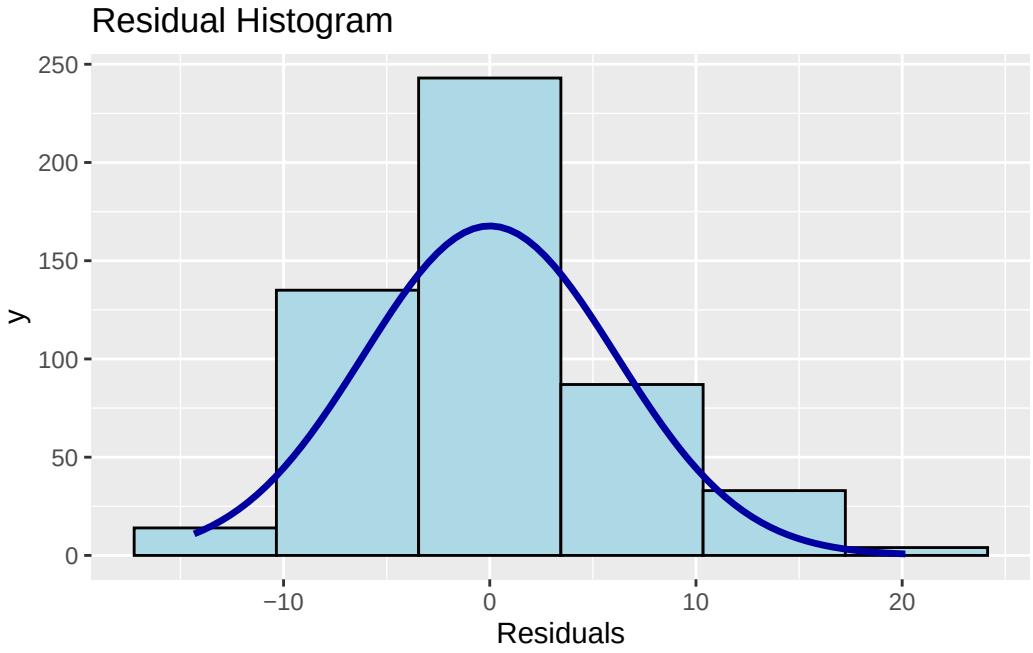

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
20
10
0
−10
−2
0
2

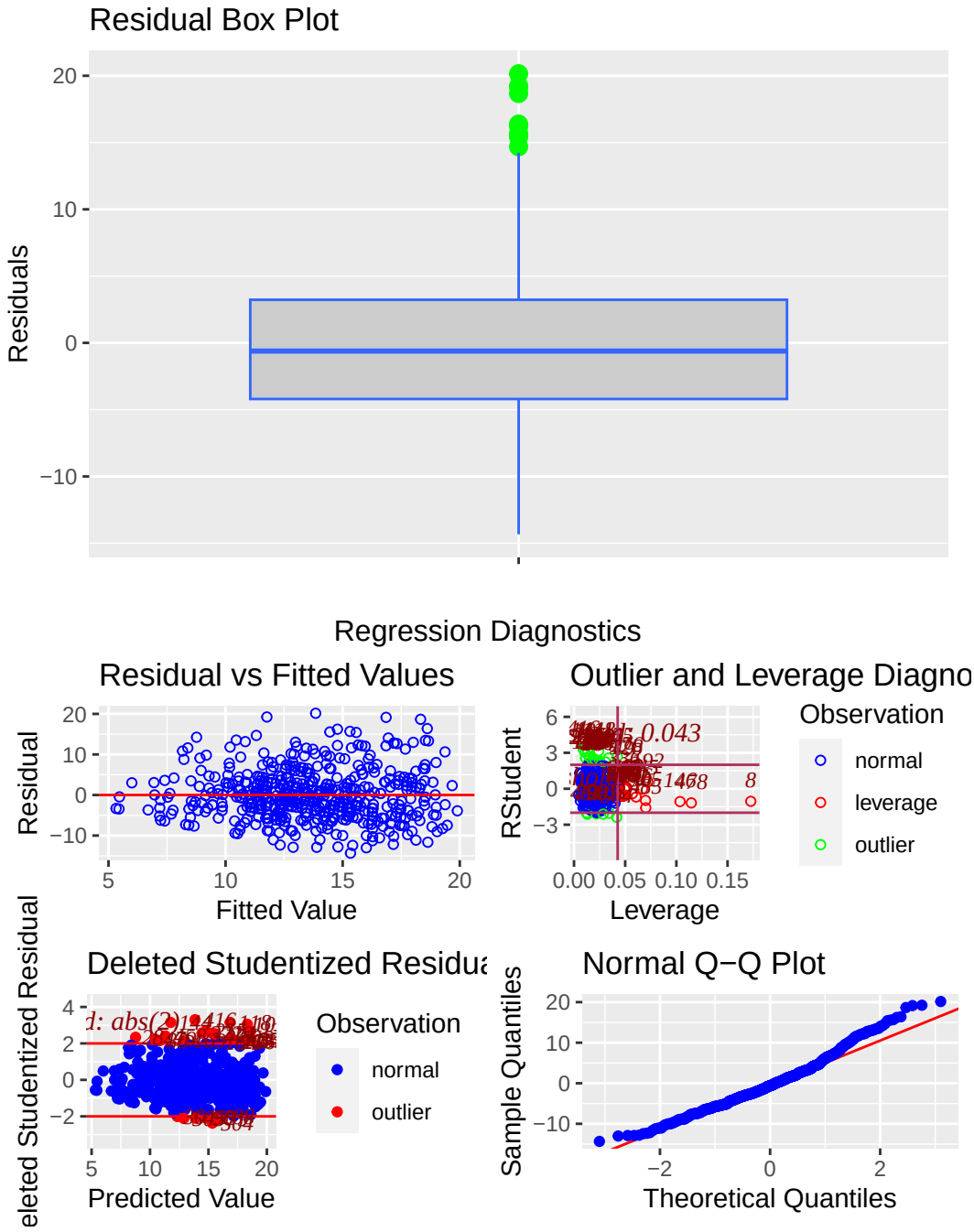

Actual vs Fitted for sum_arith_perf
Fitted Value
sum_arith_perf
20
15
10
5
0
10
20
30

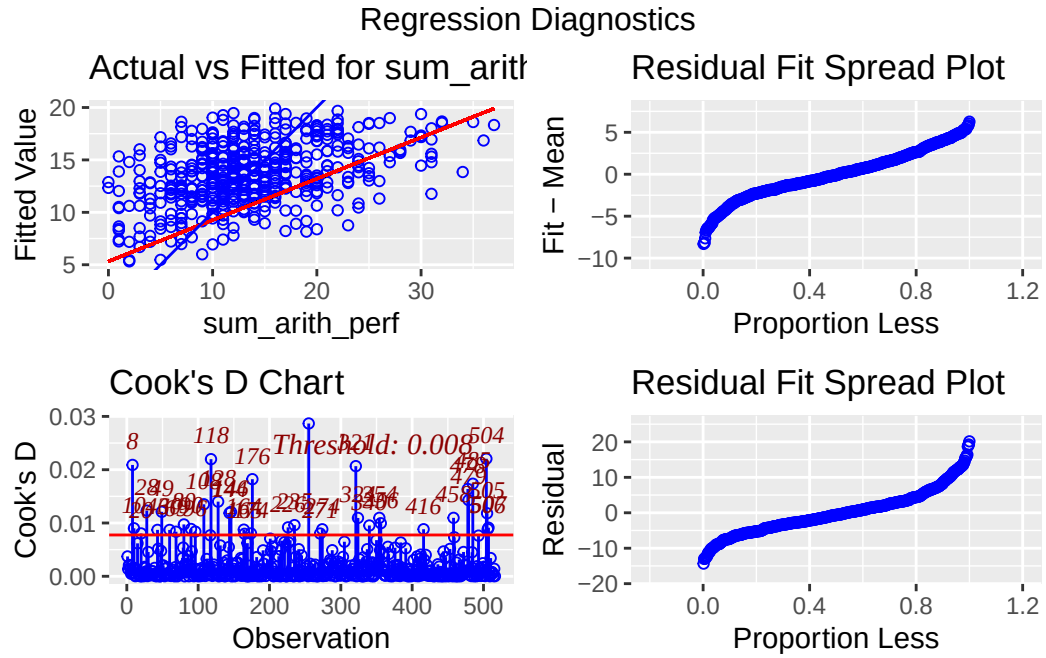

Cook's D Chart
Threshold: 0.008
Cook's D
Observation
0.03
0.02
0.01
0.00
0
100
200
300
400
500
8
118
176
321
504
485
478
479
505
458
28
49
108
128
324
354
10
20
43
60
80
90
96
164
416

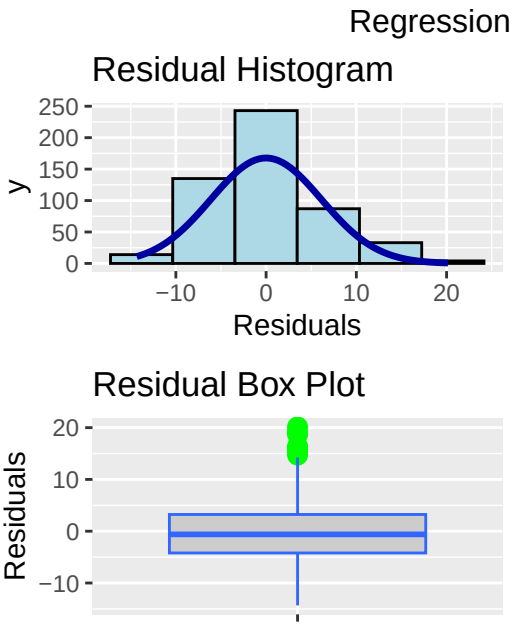

Residual Fit Spread Plot
Fit − Mean
Proportion Less
5
0
−5
−10
0.0
0.4
0.8
1.2


Residual Fit Spread Plot
Residual
20
10
0
−10
−20
0.0
0.4
0.8
1.2
Proportion Less


Residual Histogram
y
250
200
150
100
50
0
−10
0
10
20
Residuals

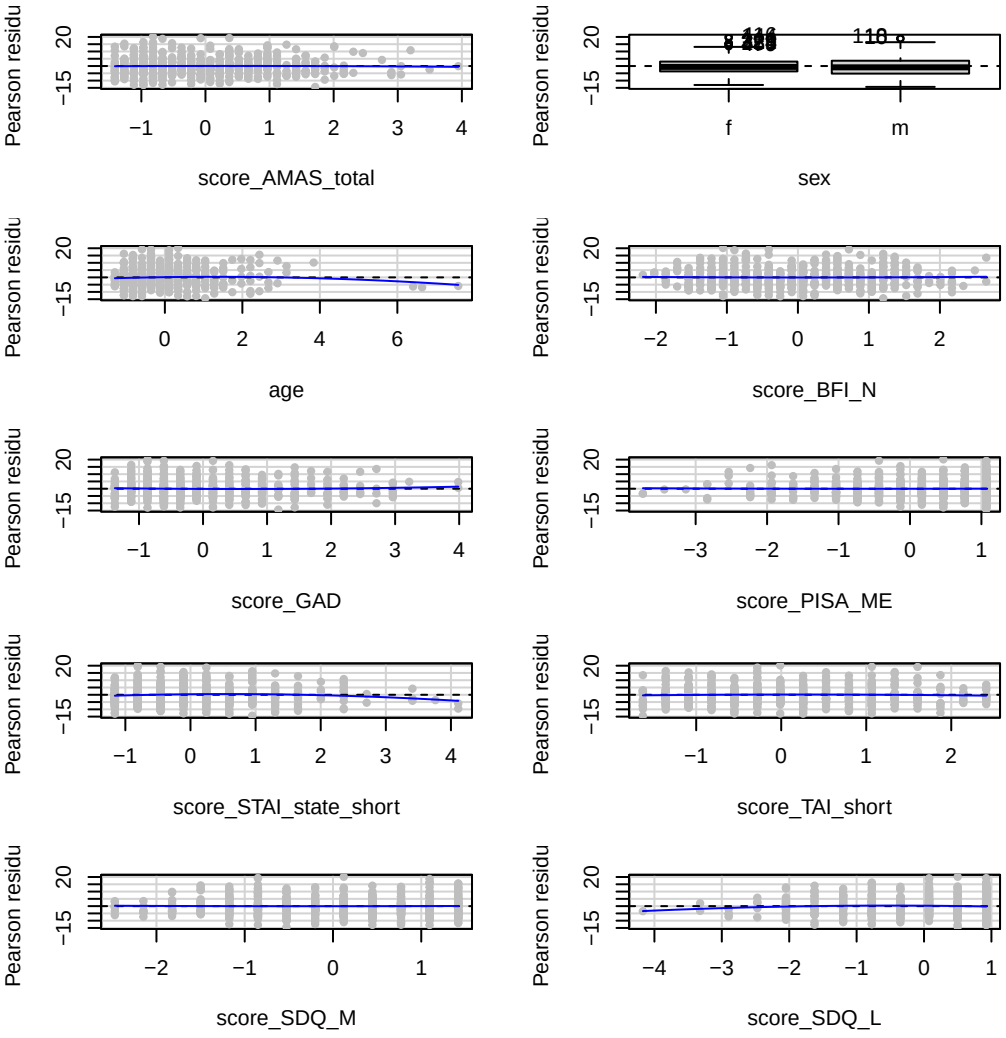

Residual Box Plot
Residuals
20
10
0
−10
Regression Diagnostics
Residual vs Fitted Values
Residual
Fitted Value
Outlier and Leverage Diagno
RStudent
Leverage
Observation
normal
leverage
outlier
Deleted Studentized Residua
eleted Studentized Residual
Predicted Value
Observation
normal
outlier
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles



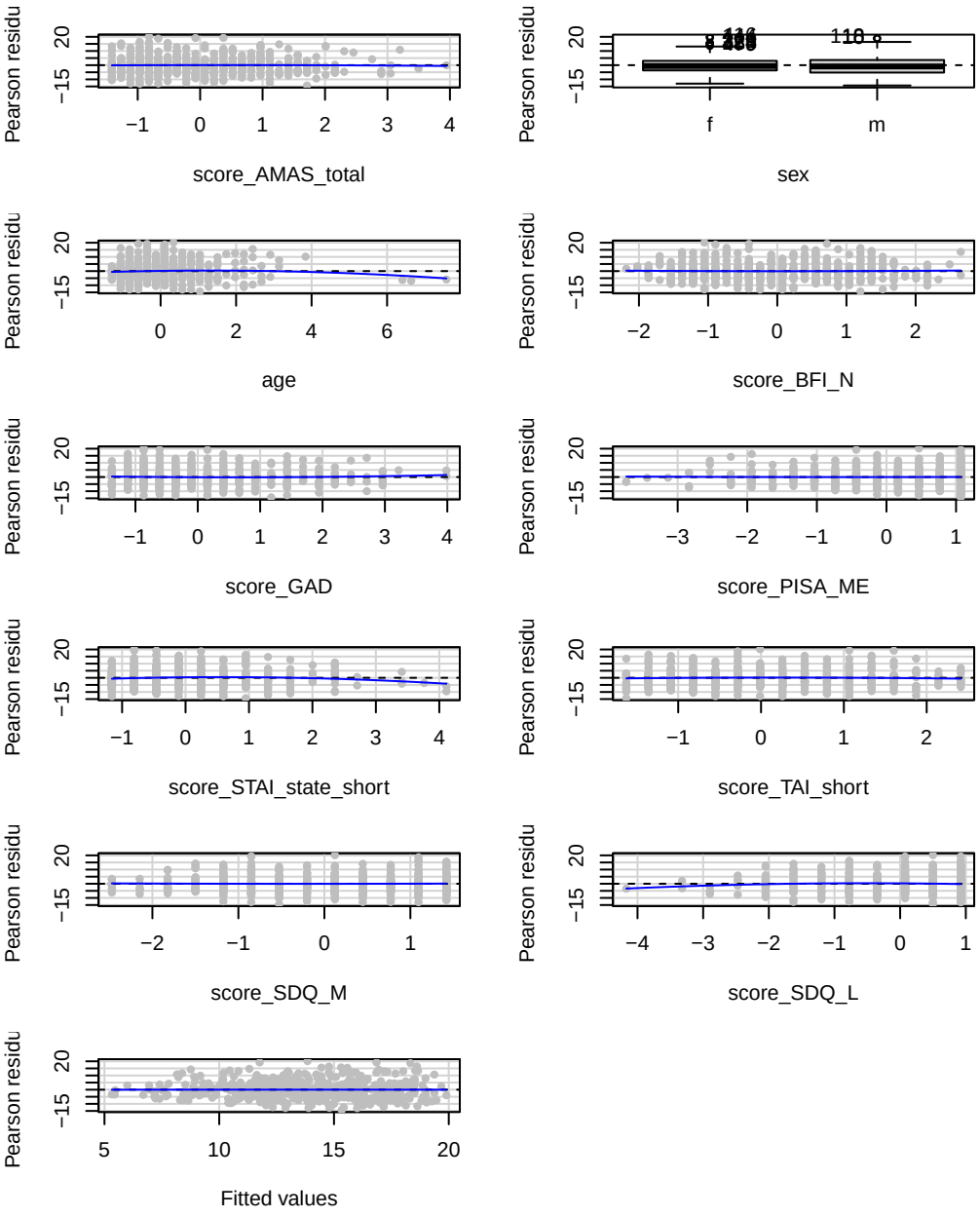



```
## see residuals by variable, mostly for linearity.
car::residualPlots(lambdamodel,
  pch = 20, col = "gray",
  fitted = F,
  ask = F, layout = c(3, 2),
```

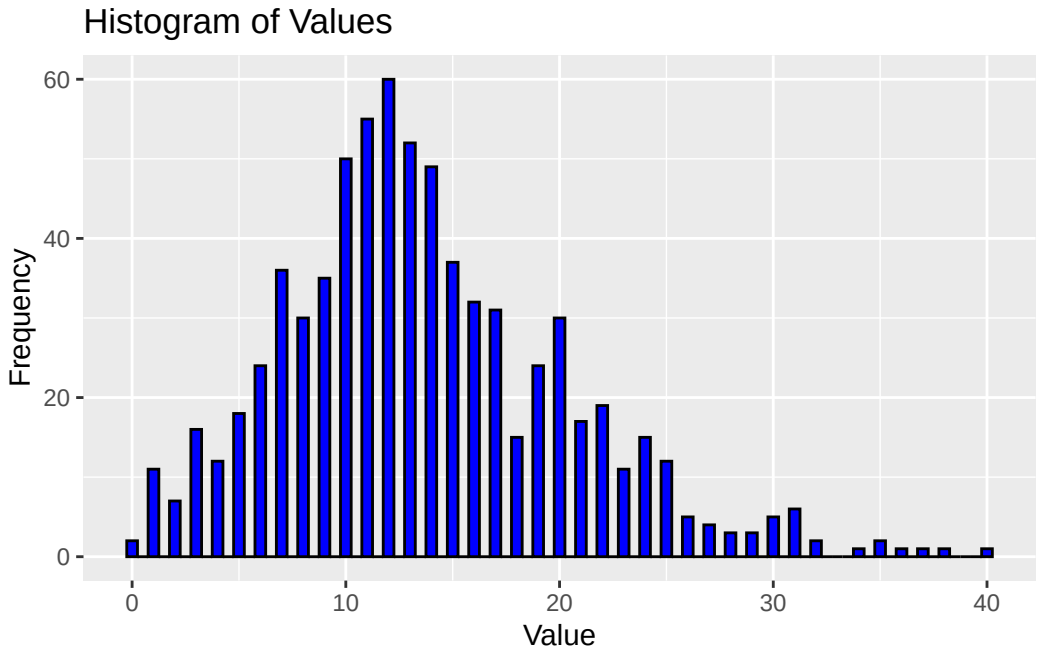
```
  tests = F
)
```




```
## see residuals by variable, mostly for heteroskedasticity, notice the fitted argument
car::residualPlots(lambdamodel,
  pch = 20, col = "gray",
  ask = F, layout = c(3, 2),
  fitted = T,
  tests = F
)
```

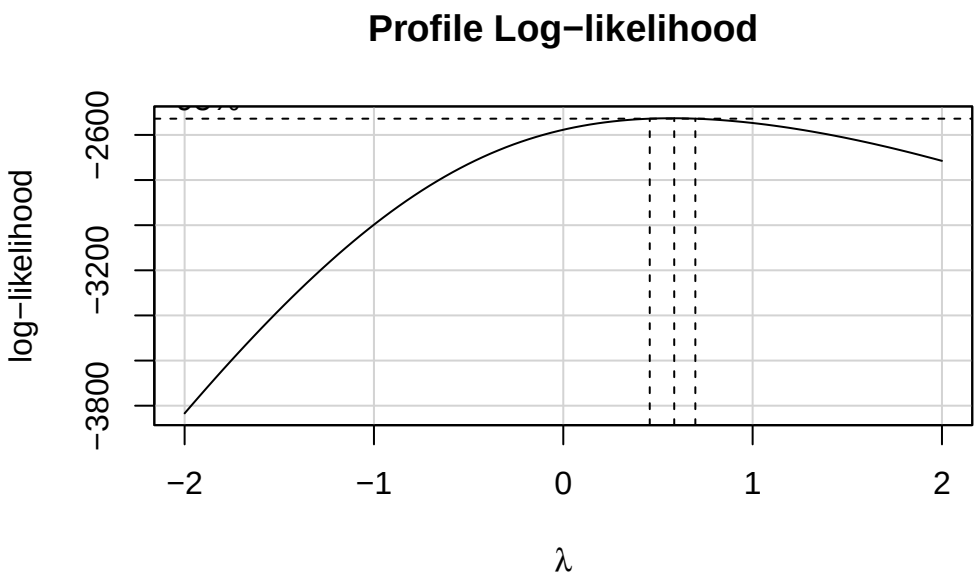



```
ggplot(lassoamatus, aes(x = sum_arith_perf)) +
  geom_histogram(binwidth = 0.5, fill = "blue", color = "black") +
  labs(title = "Histogram of Values", x = "Value", y = "Frequency")
```

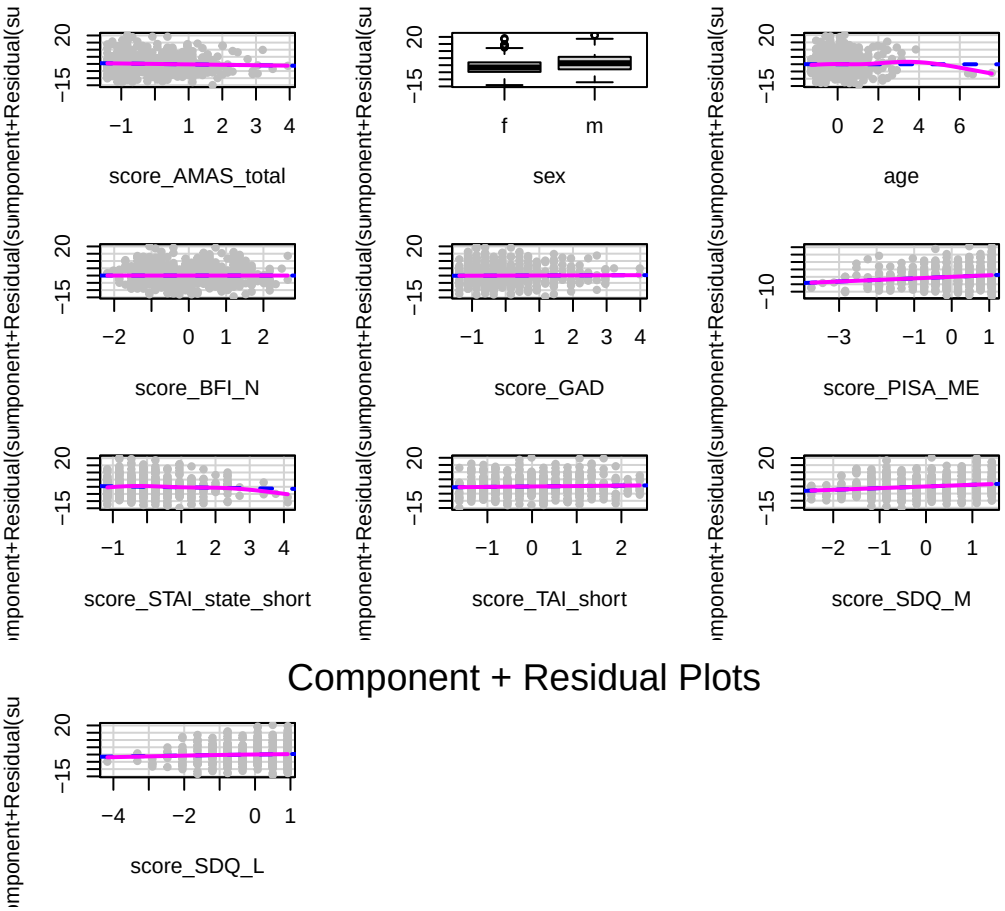


Like we mentioned in the first document, the dependent variable was skewed right. So is our distribution of residuals, unsurprisingly. We're gonna need to try a transformation of the DV. Let's try a Box Cox transformation to center the varia- wait, we have two observations that are 0 and Box Cox doesn't work with those! We probably wouldn't lose variability if we just cut the two 0s. But that's lazy. Let's try an extremely similar transformation, a Yeo-Johnson transformation that can indeed handle values of 0.

```
library(bestNormalize)
##
## Attaching package: 'bestNormalize'
## The following object is masked from 'package:MASS':
##
##     boxcox
library(car)
# optimal lambda for Yeo-Johnson transformation? Let's find it.
a <- boxCox(lambdamodel, family = "yjPower", plotit = TRUE)
```

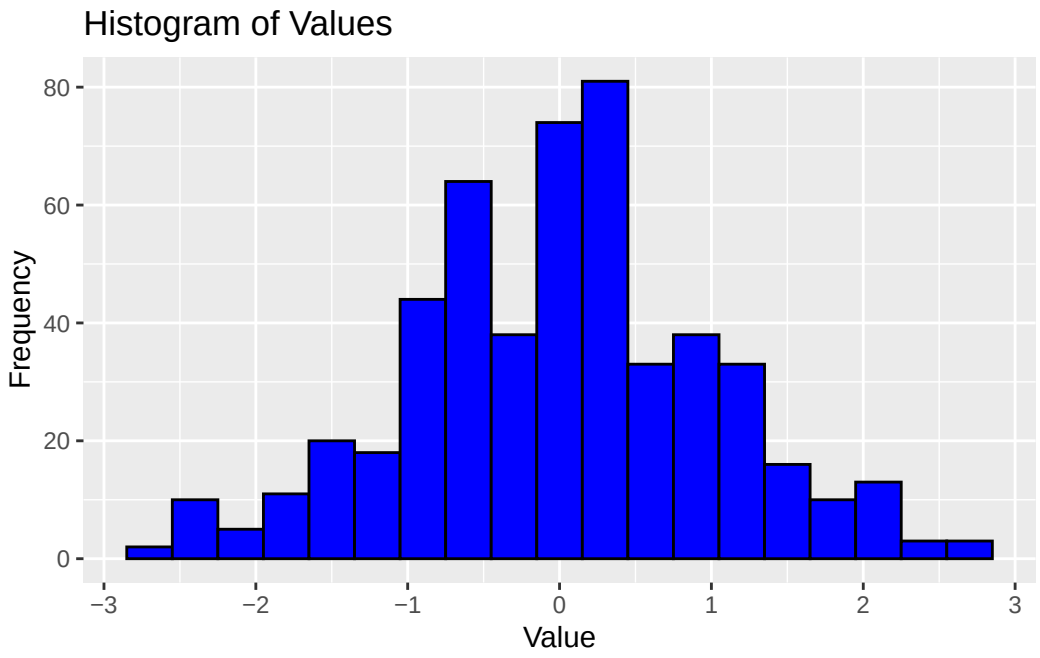



```
optimal_lambda2 <- a$x[which.max(a$y)]
optimal_lambda2
## [1] 0.5858586
# .58585856

# Let's visualize it too.
crPlots(lambdamodel,
  pch = 20, col = "gray",
  smooth = list(smoother = car::gamLine)
)
```



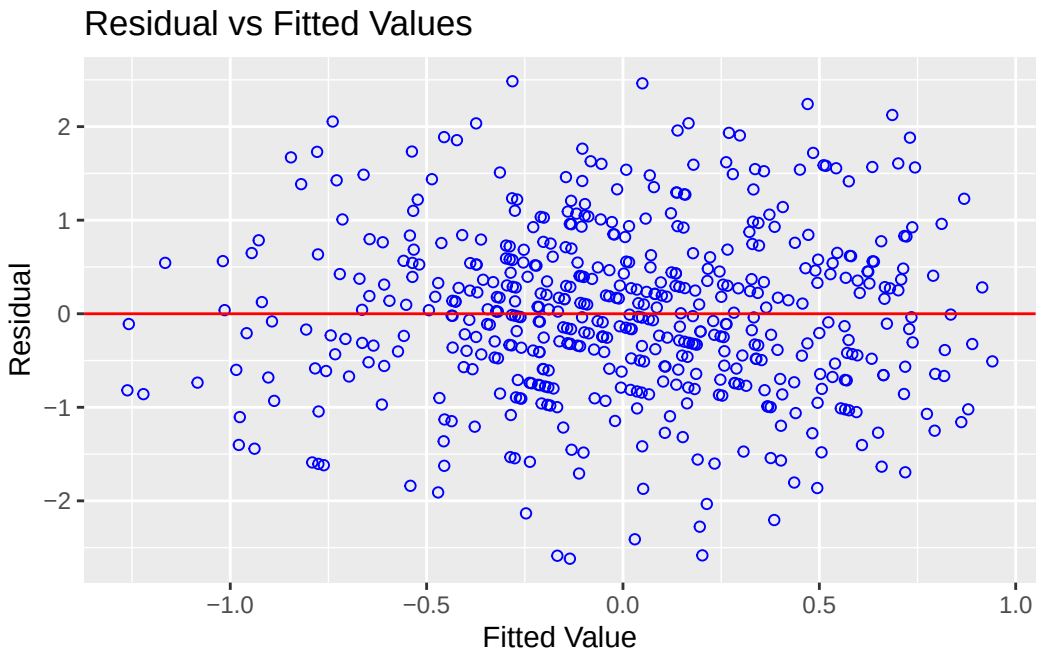
Component + Residual Plots

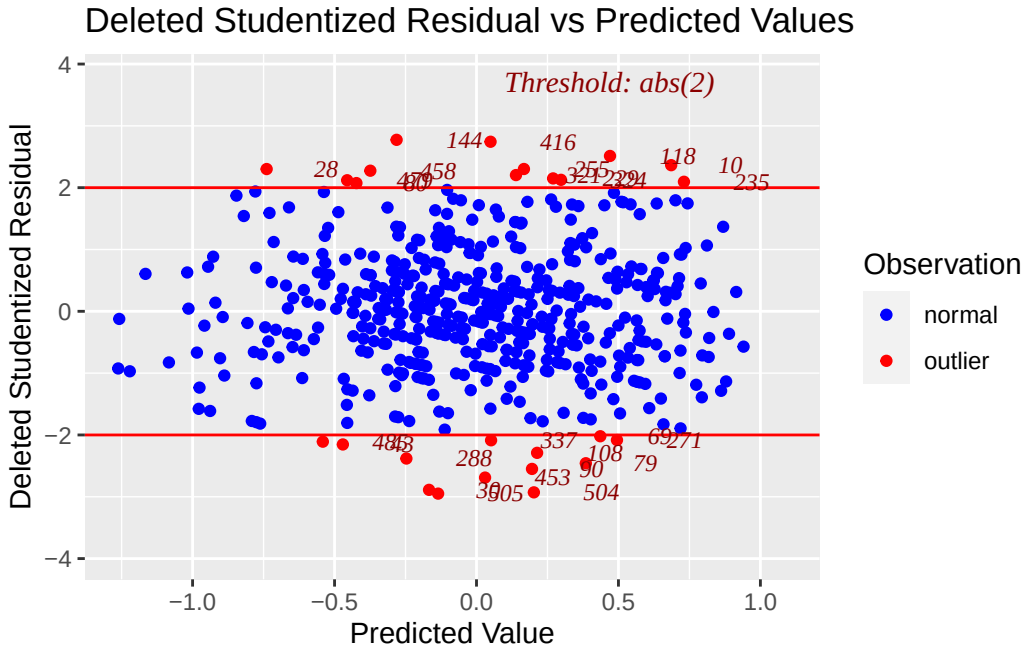
```
# Apply the Yeo-Johnson transformation manually
yj_transform <- yeojohnson(scaled_training2$sum_arith_perf, lambda = optimal_lambda)

scaled_training2$sum_arith_yj <- yj_transform$x.t
# View the transformed data
View(scaled_training2)

ggplot(scaled_training2, aes(x = sum_arith_yj)) +
  geom_histogram(binwidth = 0.3, fill = "blue", color = "black") +
  labs(title = "Histogram of Values", x = "Value", y = "Frequency")
```

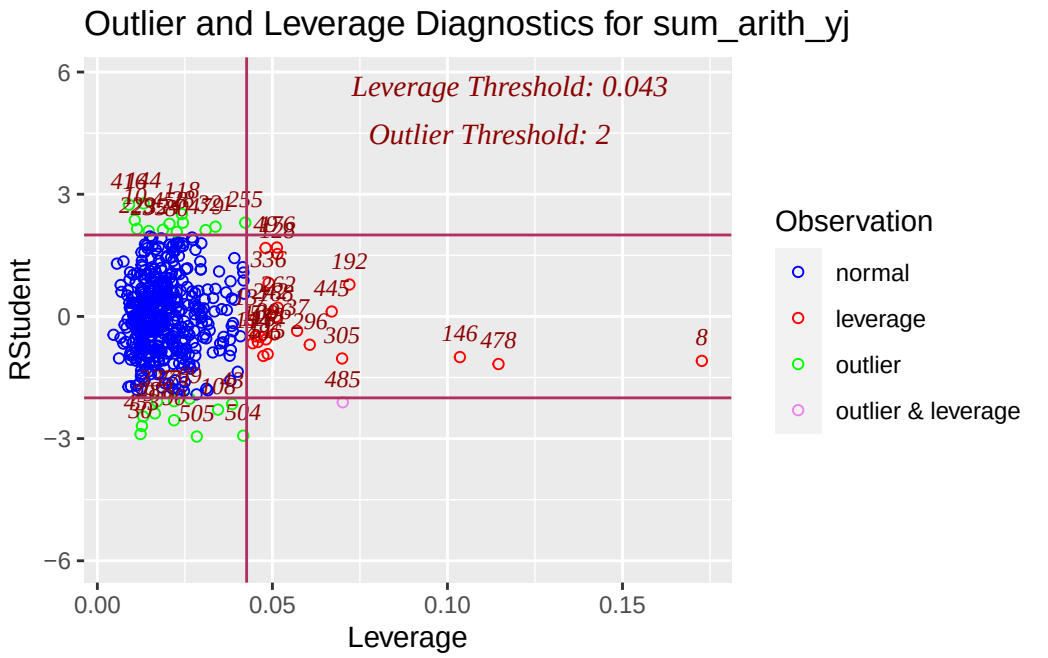


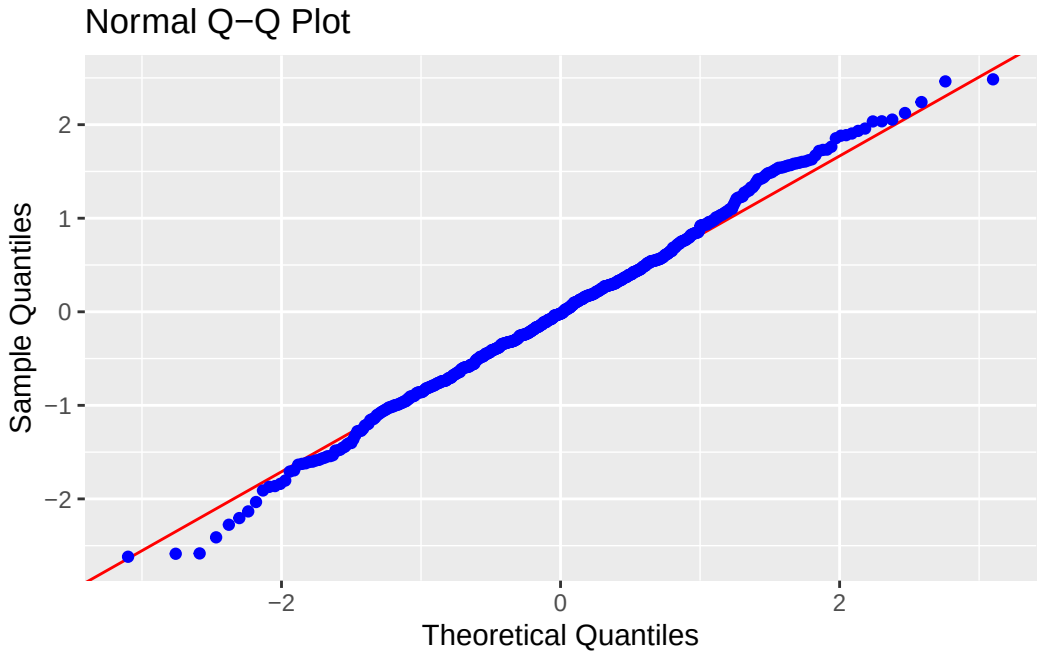

```
##
## Call:
## lm(formula = sum_arith_yj ~ score_AMAS_total + sex + age + score_BFI_N +
##     score_GAD + score_PISA_ME + score_STAI_state_short + score_TAI_short +
##     score_SDQ_M + score_SDQ_L, data = scaled_training2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.61839 -0.59134 -0.02117  0.54692  2.48470
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -0.1506350  0.0497746  -3.026  0.00260 **
## score_AMAS_total       -0.0452919  0.0621963  -0.728  0.46682
## sexm                    0.4768569  0.0939660   5.075 5.46e-07 ***
## age                     0.0005689  0.0415375   0.014  0.98908
## score_BFI_N            -0.0076183  0.0538123  -0.142  0.88748
## score_GAD               0.0216458  0.0543524   0.398  0.69061
## score_PISA_ME           0.1478662  0.0527826   2.801  0.00528 **
## score_STAI_state_short -0.0687982  0.0476976  -1.442  0.14981
## score_TAI_short         0.0498519  0.0461368   1.081  0.28043
## score_SDQ_M             0.1918772  0.0589872   3.253  0.00122 **
## score_SDQ_L             0.0516186  0.0416661   1.239  0.21597
## ---
```

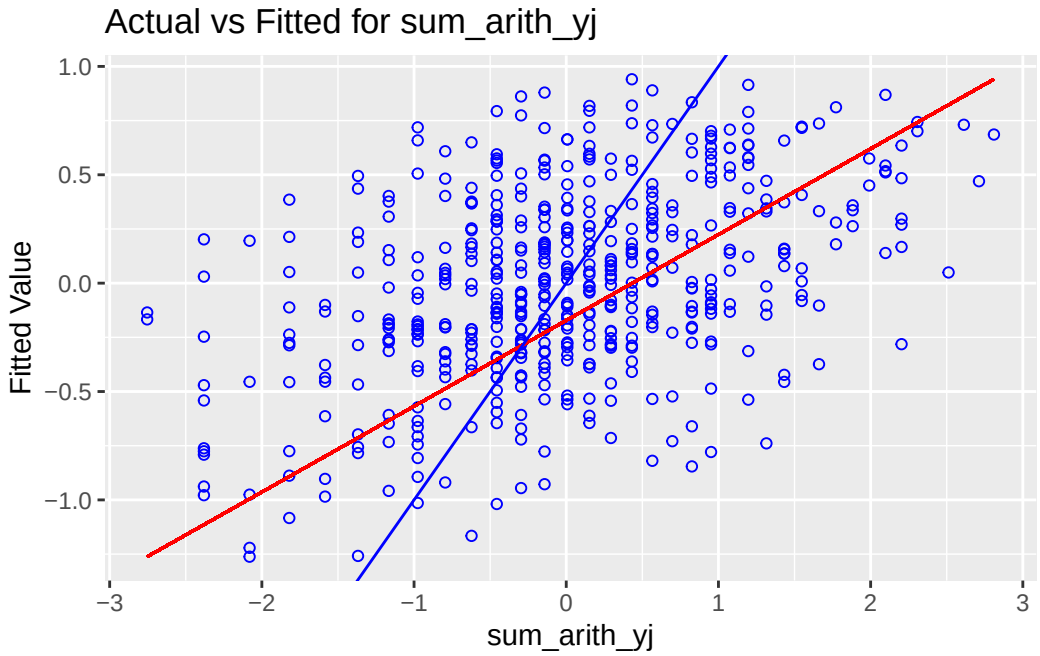
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9076 on 505 degrees of freedom
## Multiple R-squared:  0.1922, Adjusted R-squared:  0.1762
## F-statistic: 12.02 on 10 and 505 DF,  p-value: < 2.2e-16
```

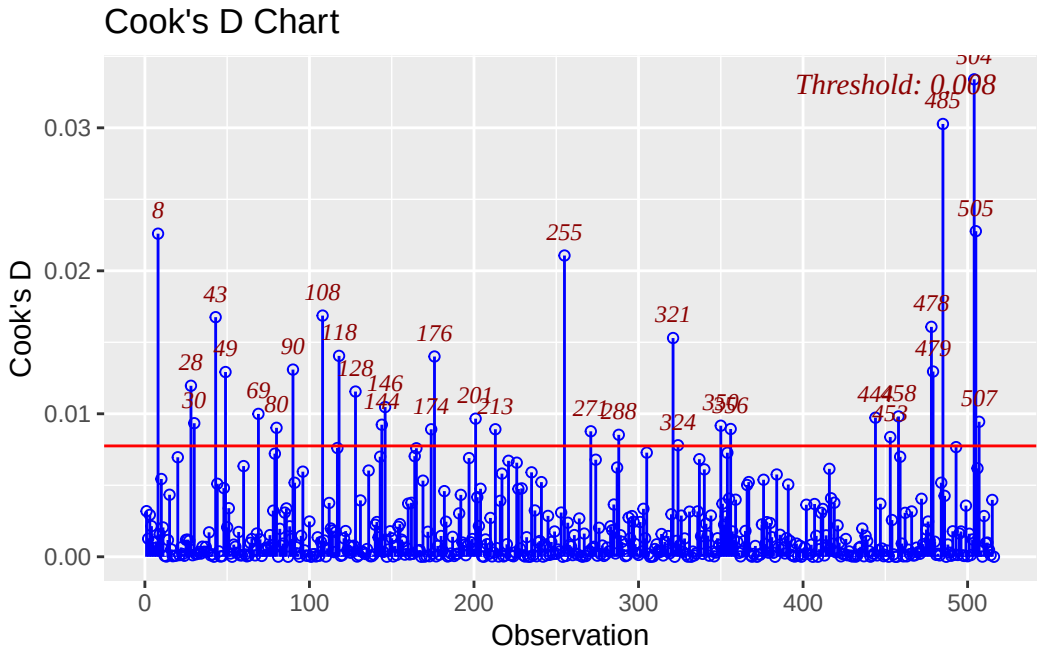


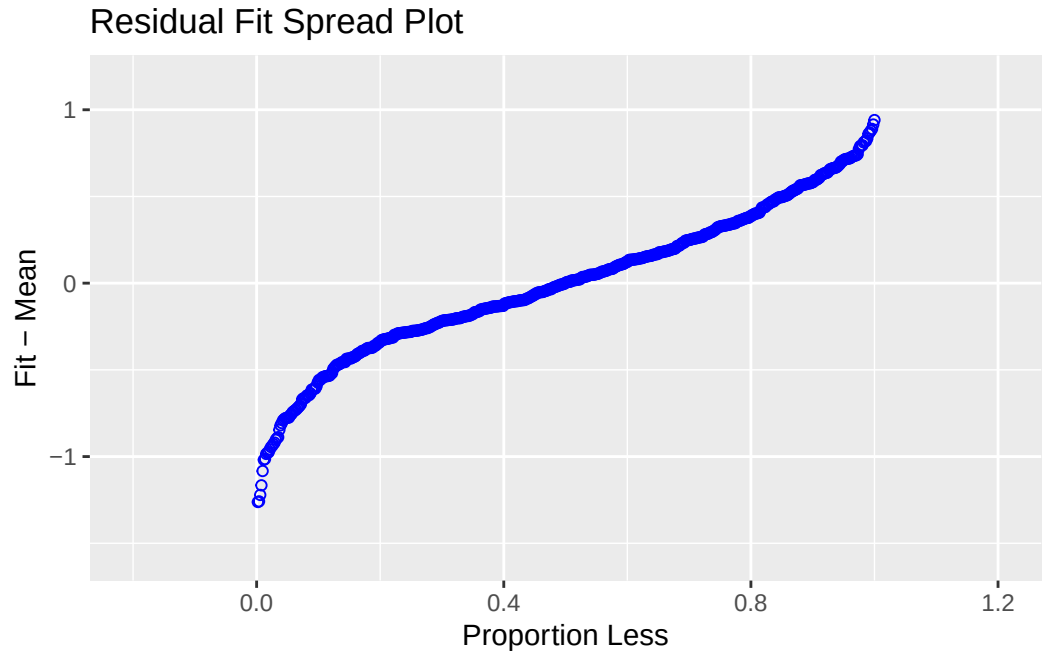


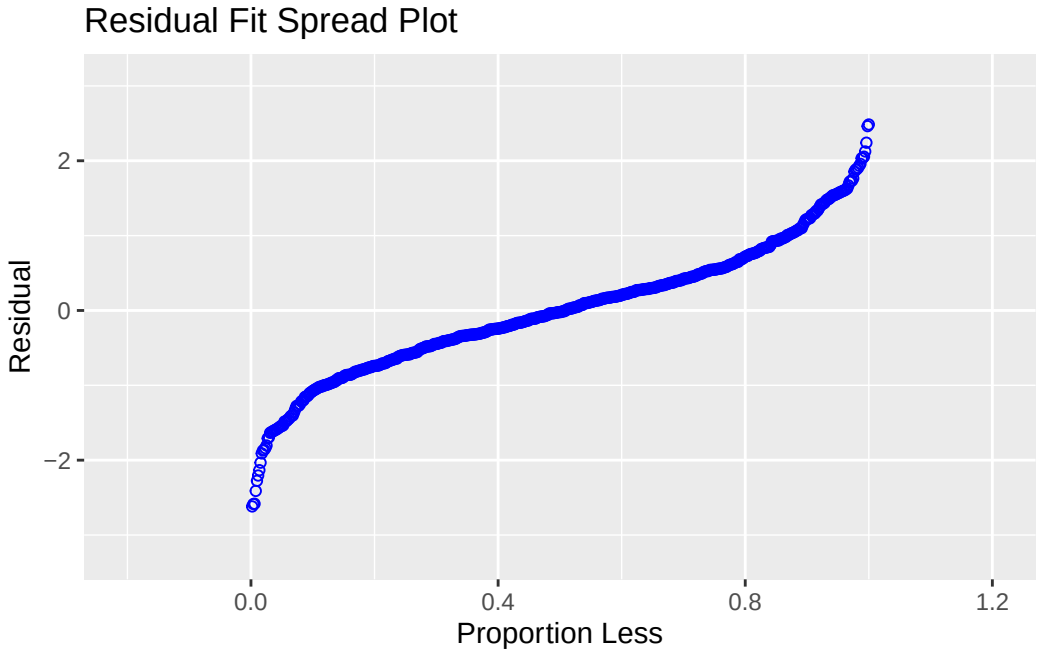

Outlier and Leverage Diagnostics for sum_arith_yj
Leverage Threshold: 0.043
Outlier Threshold: 2
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage

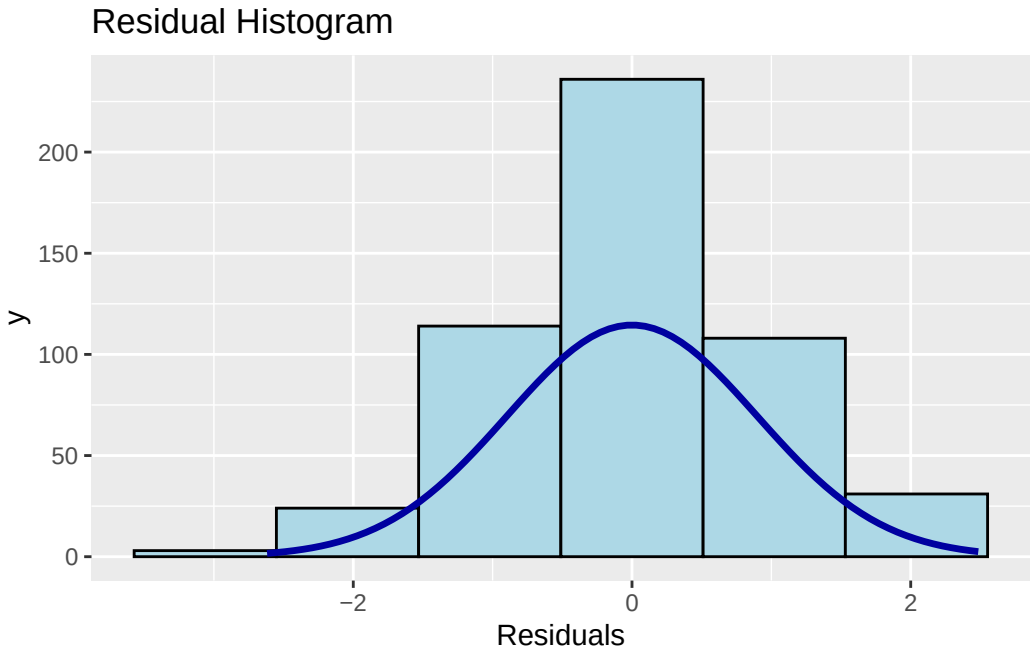

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles

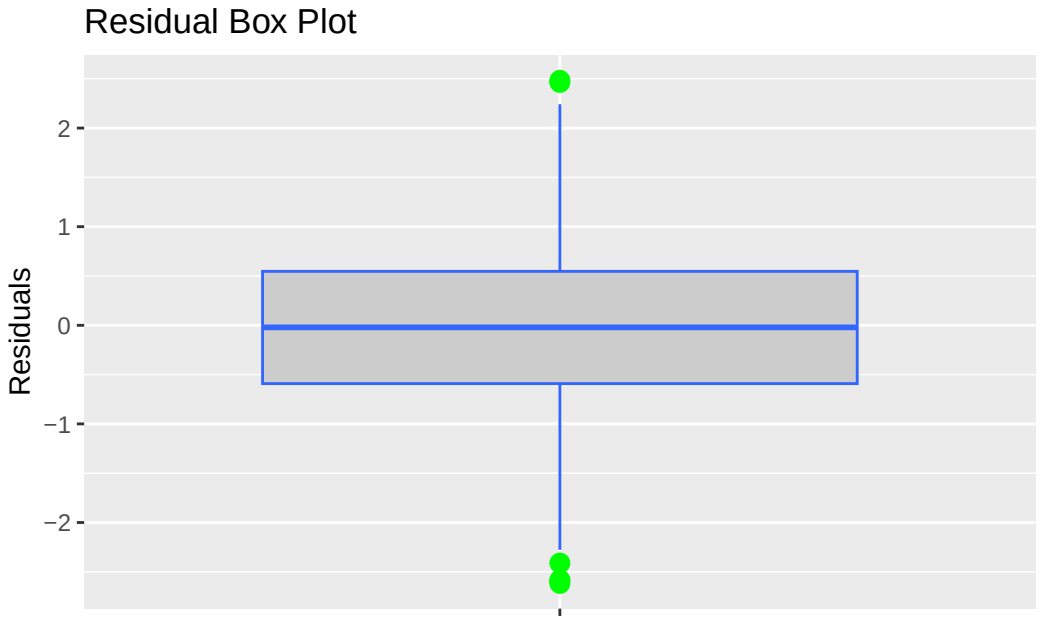

Actual vs Fitted for sum_arith_yj
Fitted Value
sum_arith_yj
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3

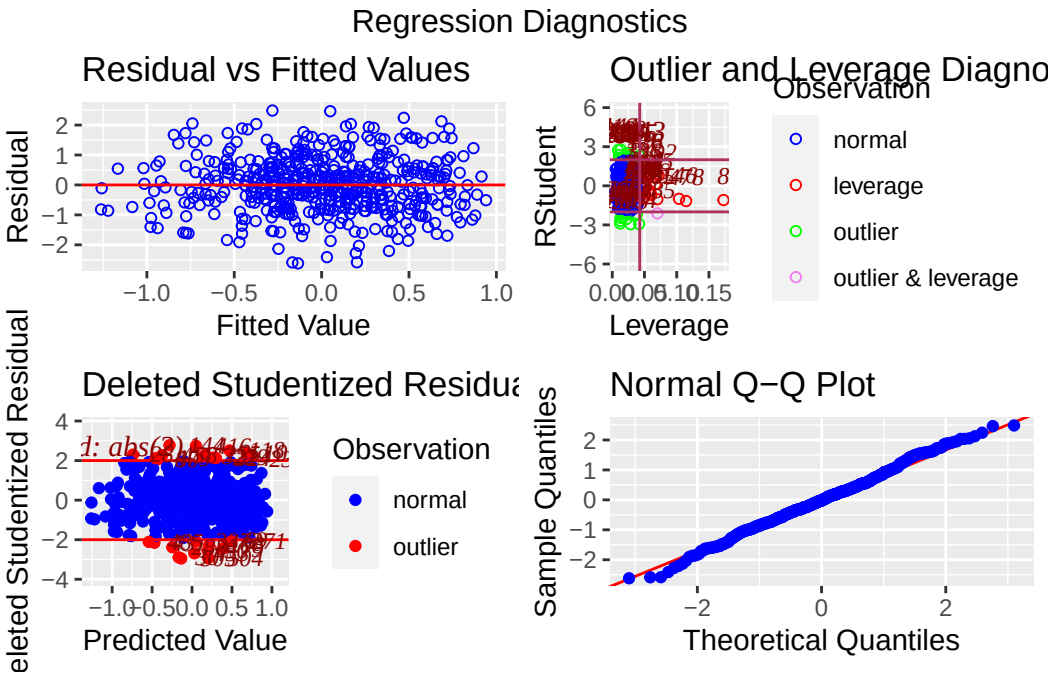

Cook's D Chart
Cook's D
Observation
Threshold: 0.008
0.03
0.02
0.01
0.00
0
100
200
300
400
500
8
43
28
49
30
69
80
90
108
118
128
146
144
176
174
201
213
255
271
288
321
324
350
356
444
453
458
478
479
485
504
505
507

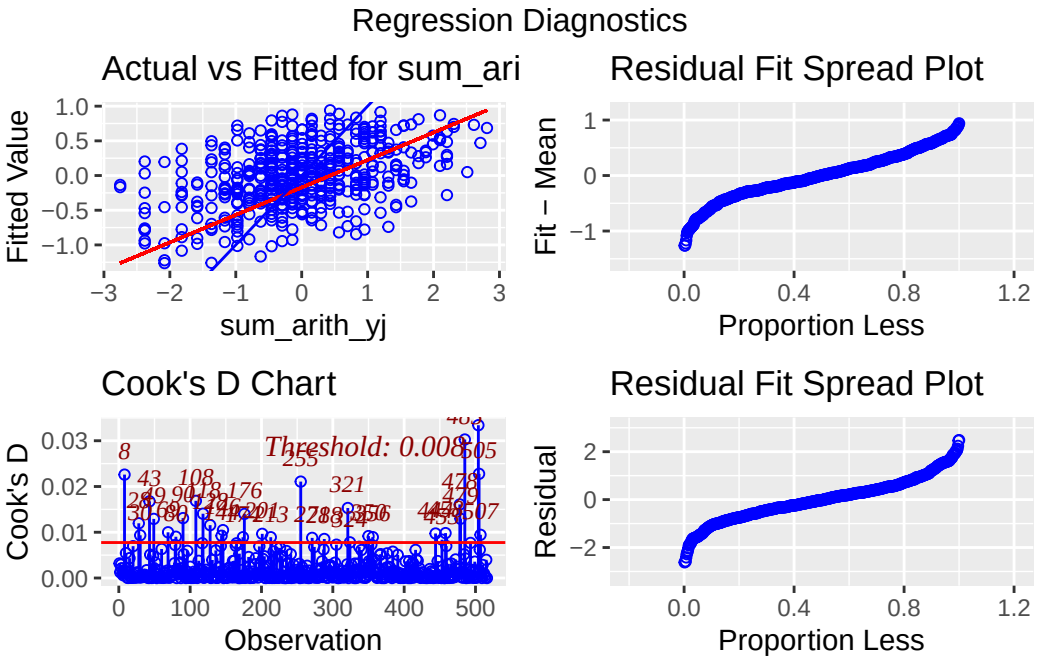

Residual Fit Spread Plot
Fit – Mean
1
0
−1
0.0
0.4
0.8
1.2
Proportion Less

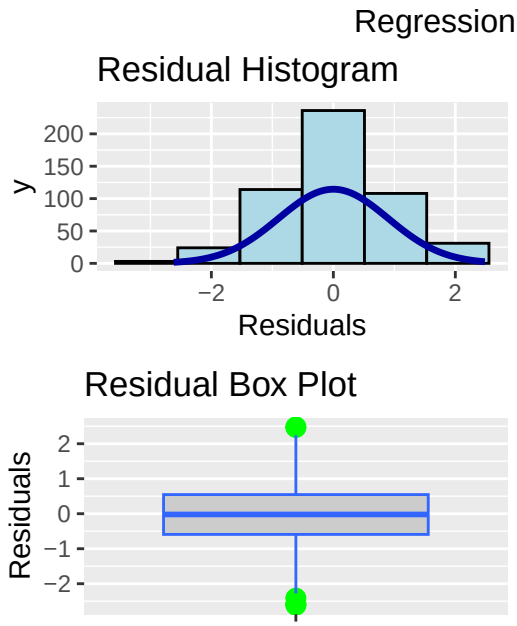

Residual Fit Spread Plot
Residual
2
0
−2
0.0
0.4
0.8
1.2
Proportion Less


Residual Histogram
200
150
100
50
0
y
−2
0
2
Residuals

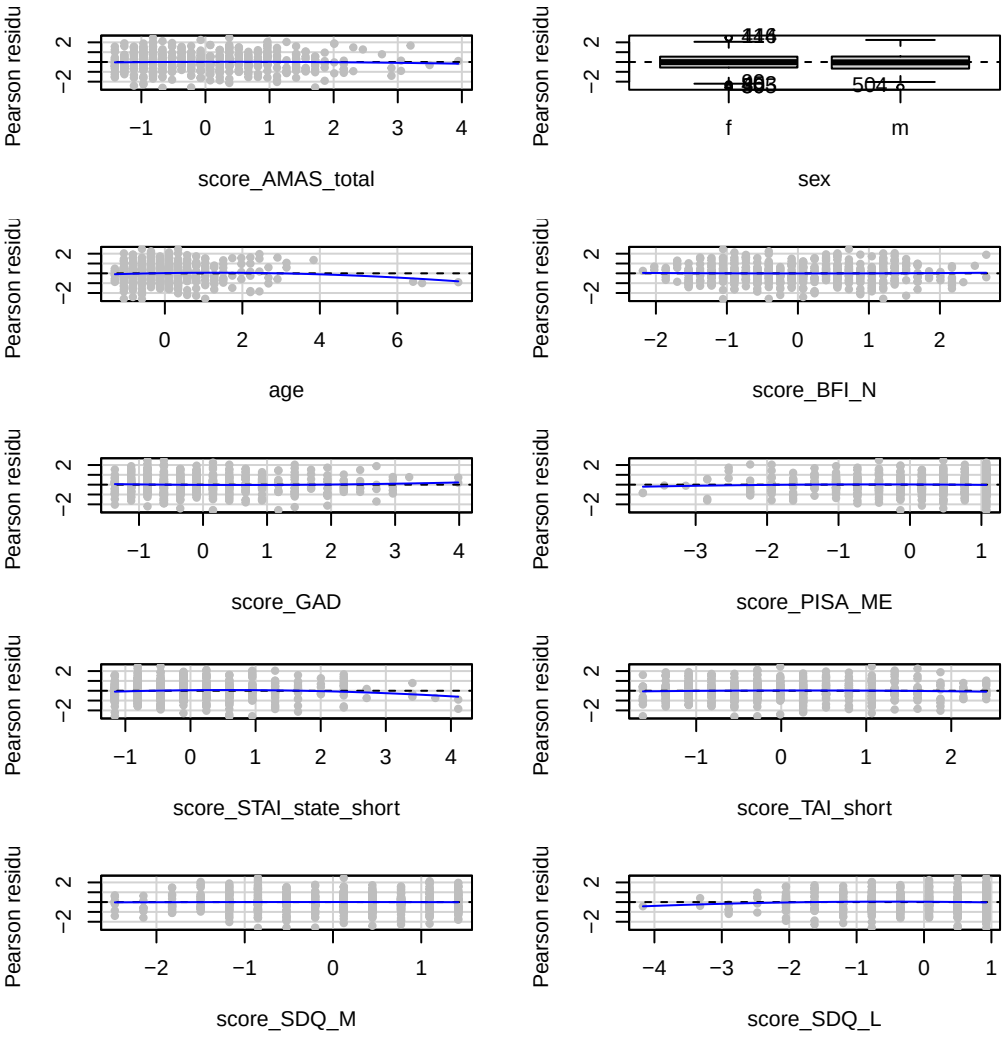

Residual Box Plot
2
1
0
−1
−2
Residuals

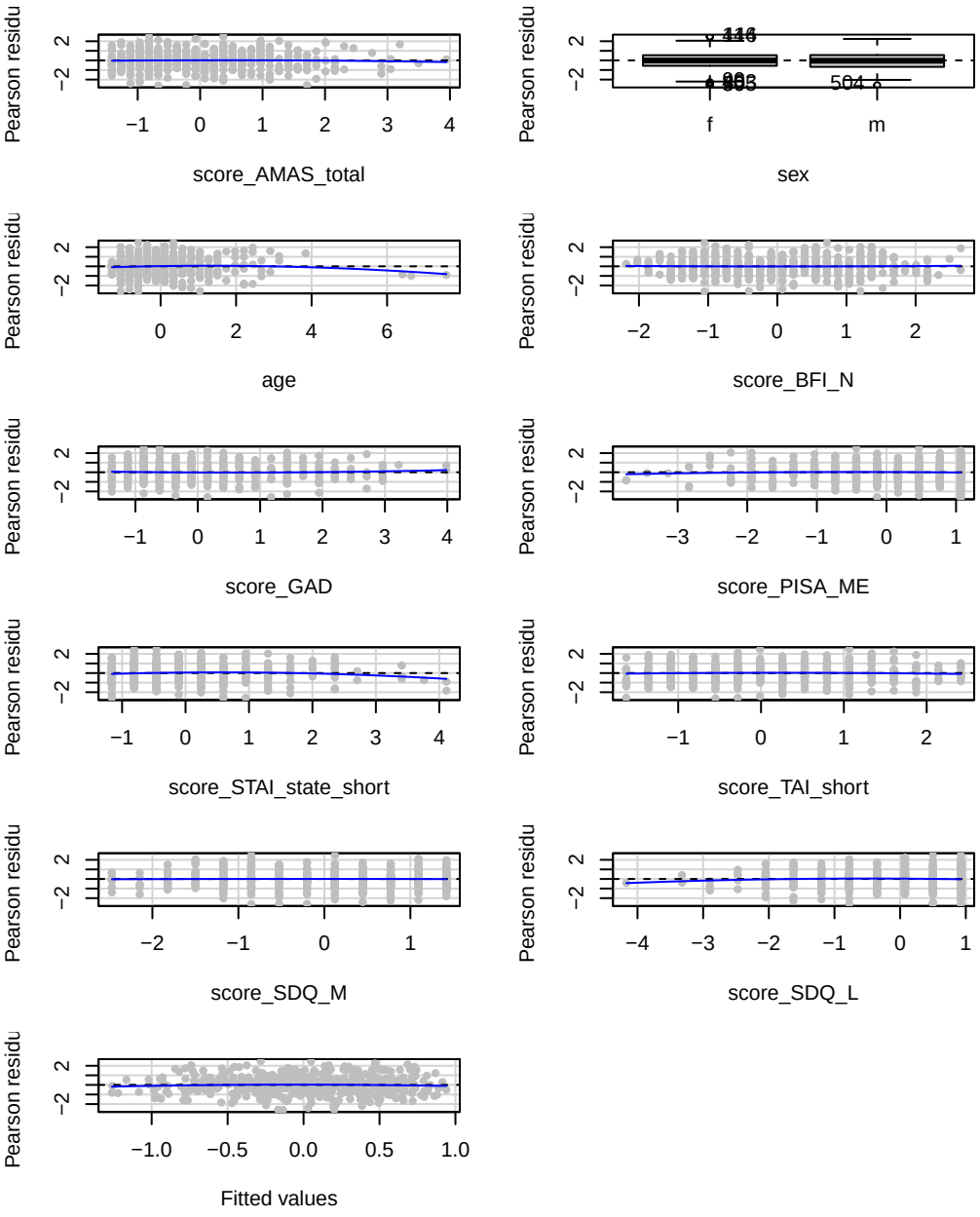

Regression Diagnostics
Residual vs Fitted Values
Residual
Fitted Value
Outlier and Leverage Diagno
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage
Deleted Studentized Residua
eleted Studentized Residual
Predicted Value
Observation
normal
outlier
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles


Regression Diagnostics
Actual vs Fitted for sum_ari
Fitted Value
sum_arith_yj
Residual Fit Spread Plot
Fit − Mean
Proportion Less
Cook's D Chart
Cook's D
Threshold: 0.008
Observation
Residual Fit Spread Plot
Residual
Proportion Less




```
##       score_AMAS_total                   sex                   age
##               2.418458              1.195244              1.078674
##            score_BFI_N             score_GAD          score_PISA_ME
##               1.810396              1.846917              1.741771
## score_STAI_state_short       score_TAI_short            score_SDQ_M
##               1.422337              1.330776              2.175332
##            score_SDQ_L
##               1.085367
```


Pearson residu
score_AMAS_total
sex
f
m
age
score_BFI_N
score_GAD
score_PISA_ME
score_STAI_state_short
score_TAI_short
score_SDQ_M
score_SDQ_L

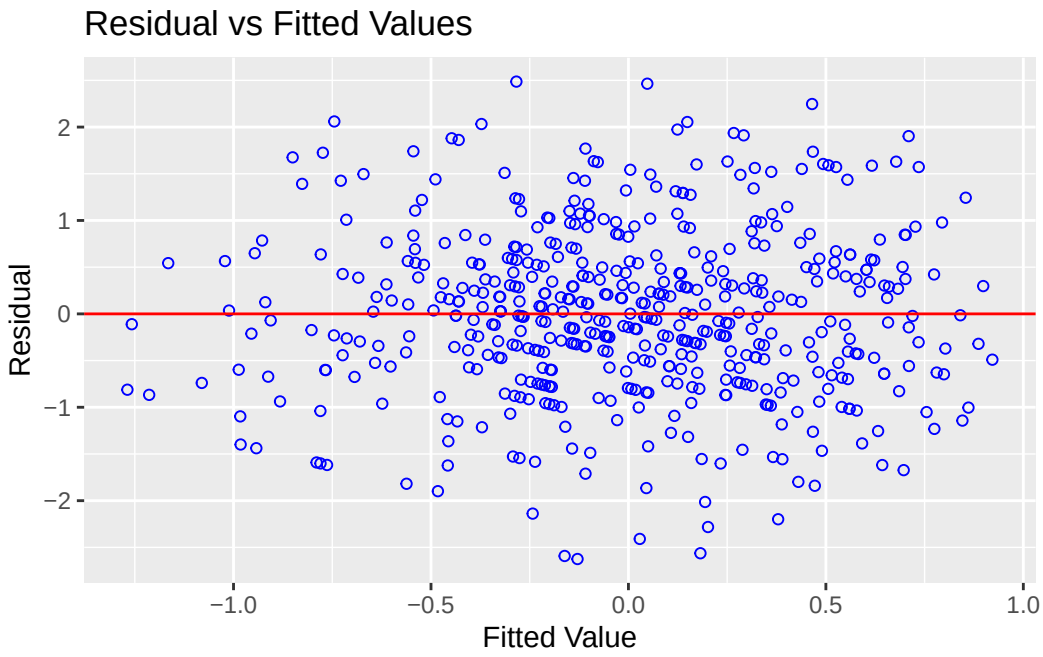


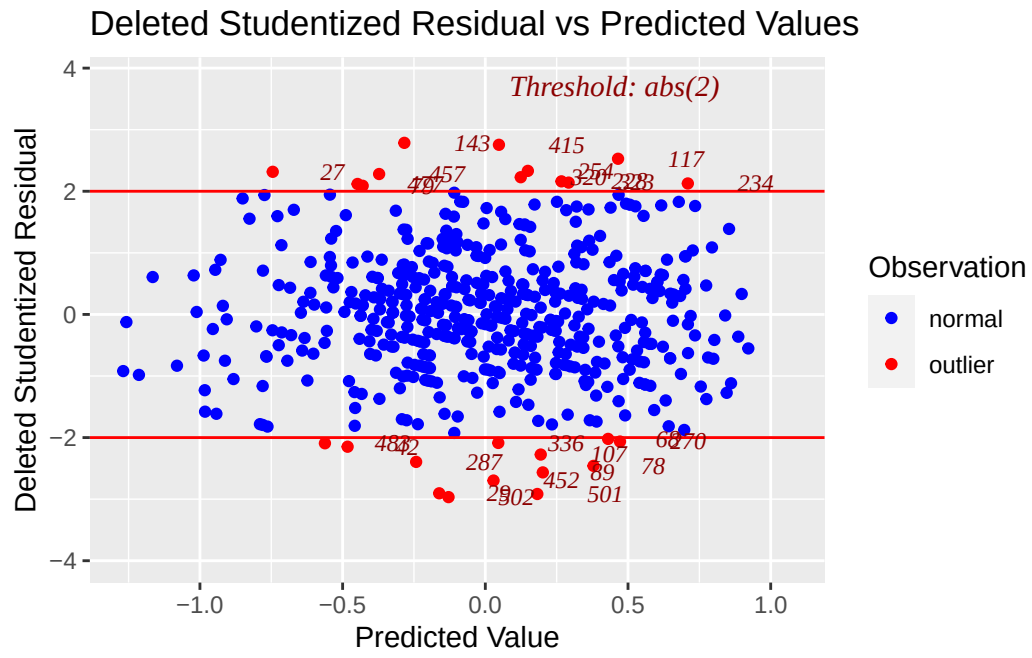
This model looks so much better it's not even funny. Much better resid/fitted plot, much better QQ plot (though with some issues we'll mention in a second), and of course a much more reasonable DV plot.VIF numbers seem reasonable, but we'll keep an eye on it. Notice the residual histogram has an issue - there is a tail of low residual values (which can be spotted near the bottom of the residual/fitted graph.) This indicates the model is systematically underfitting certain values around 0. The QQ plot shows that our distribution is pretty normal, but there are a couple causes for concern, namely the over/under estimation at theoretical

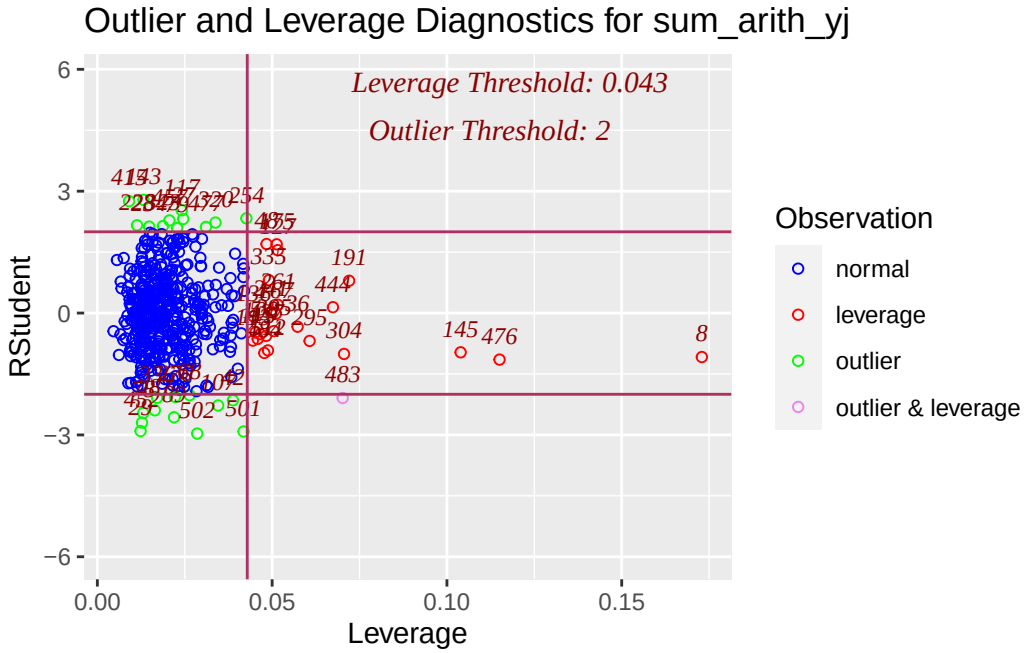
quantile values around 2 and -2, respectively. This probably indicates some polynomial terms may be necessary to add to the model, but I will need to confirm.

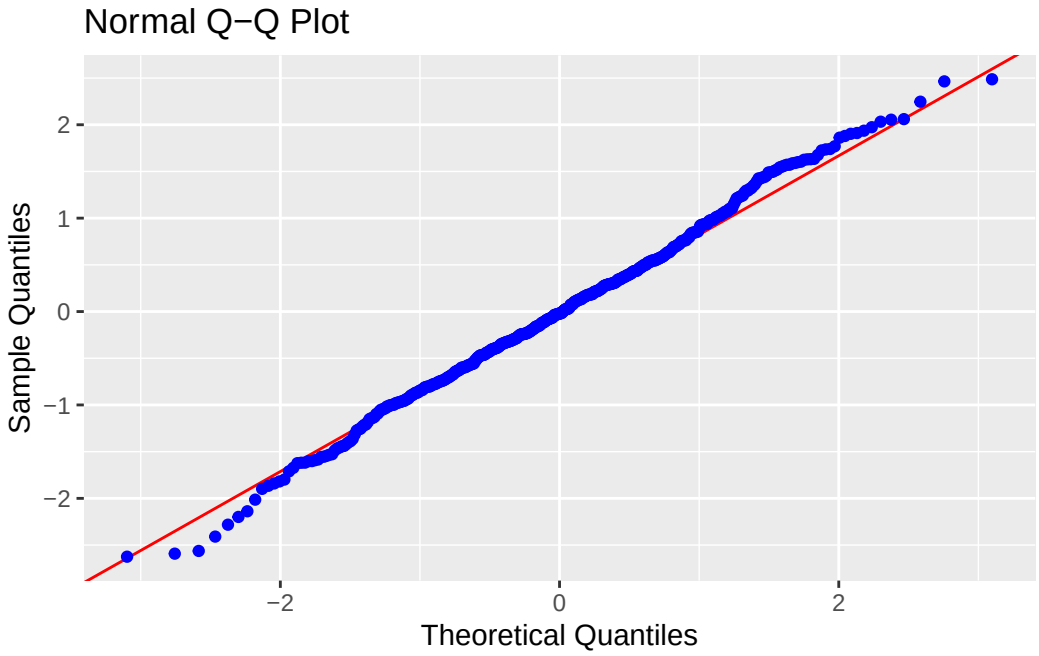
Just for fun, we'll also try a model just cutting the outliers and also run a model with no SDQ_L.

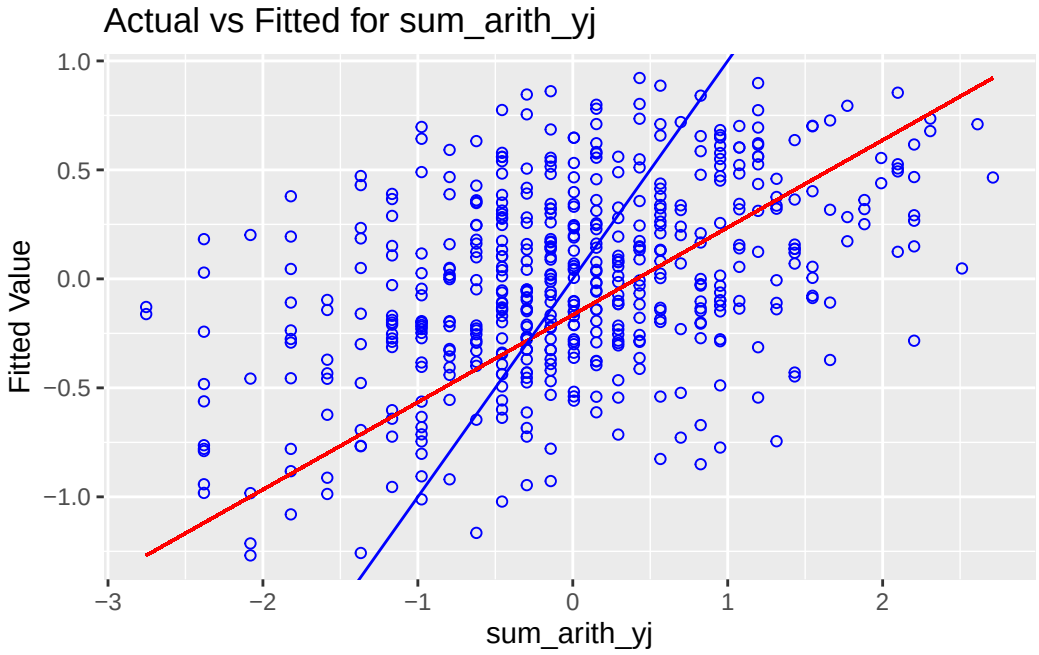
```
# no outlier model.
scaled_training3 <- scaled_training2
scaled_training3 <- scaled_training2[-c(502, 10, 477), ]
transformedmodel2 <- lm(sum_arith_yj ~ score_AMAS_total + sex + age + score_BFI_N + score_GA
ols_plot_diagnostics(transformedmodel2)
```

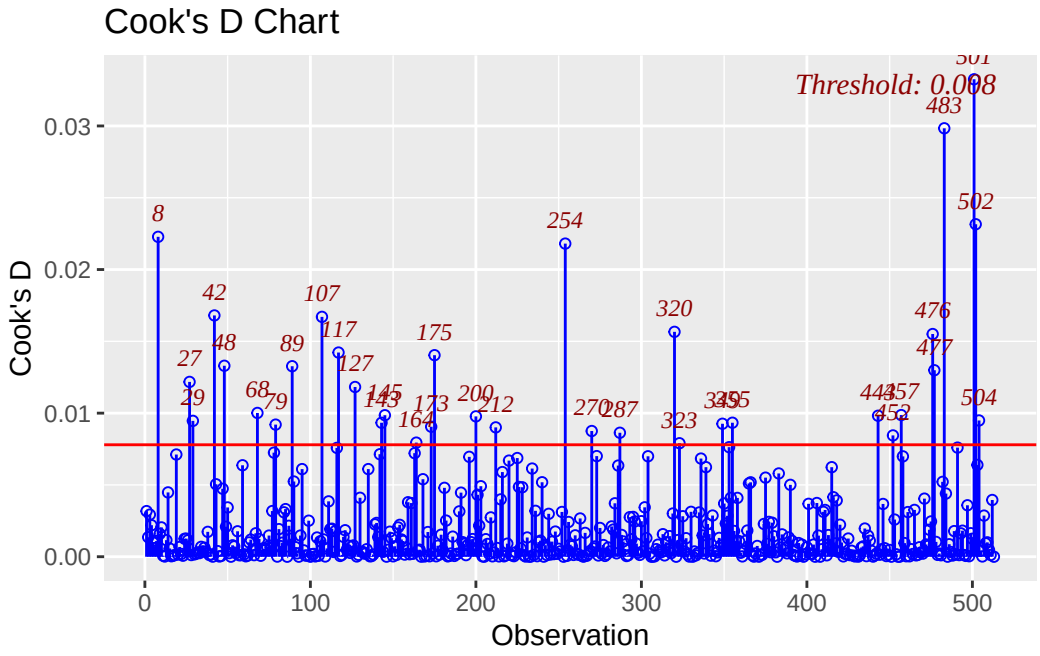


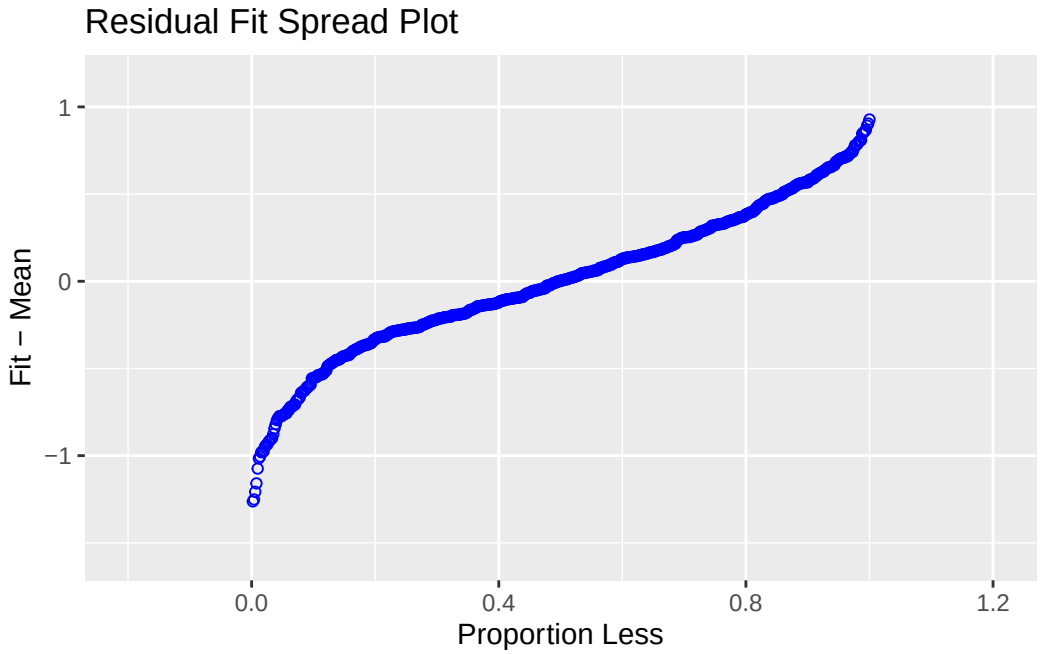

Deleted Studentized Residual vs Predicted Values
Threshold: abs(2)
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier

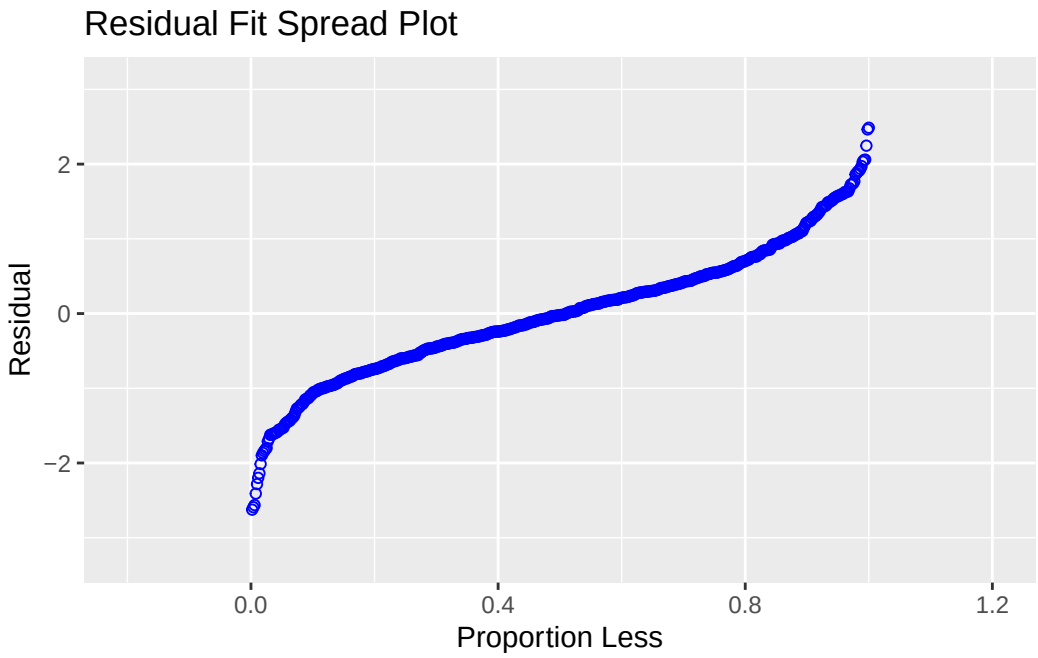

Outlier and Leverage Diagnostics for sum_arith_yj
Leverage Threshold: 0.043
Outlier Threshold: 2
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage

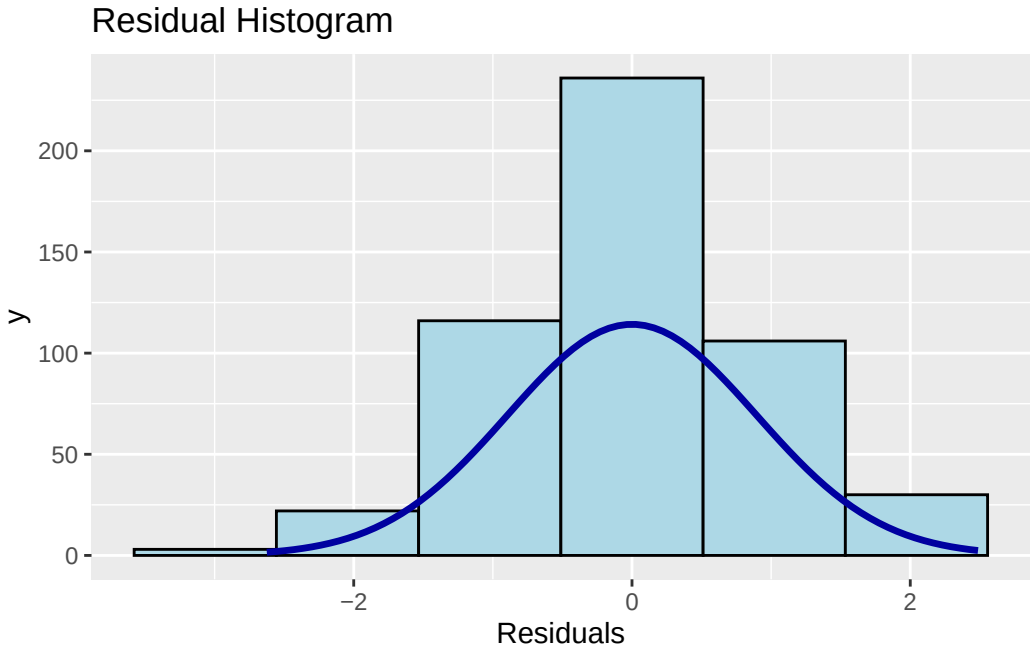

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
2
1
0
−1
−2
−2
0
2

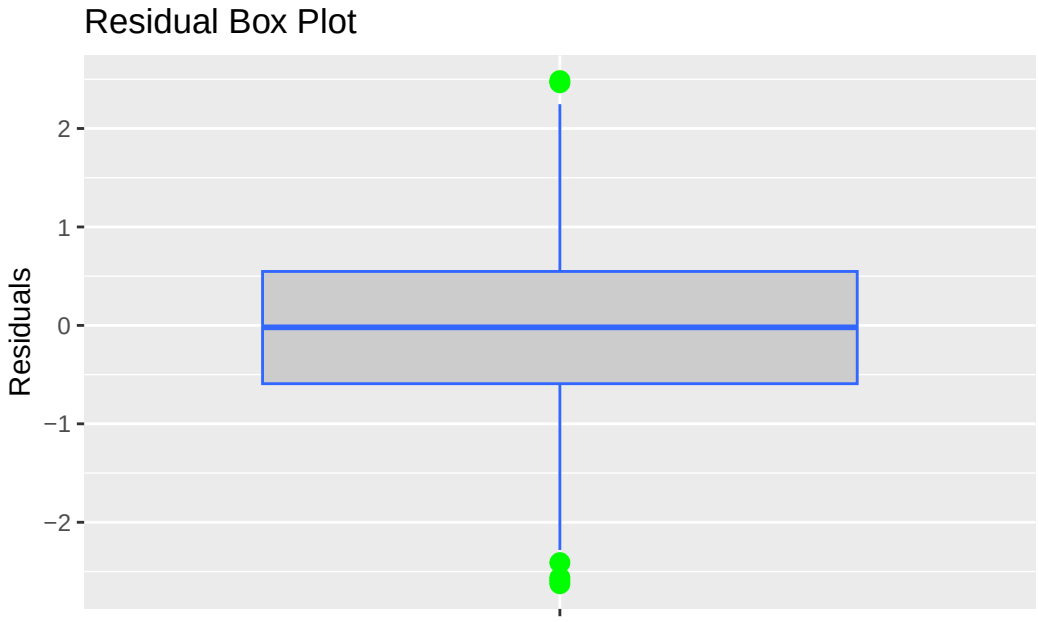

Actual vs Fitted for sum_arith_yj
Fitted Value
sum_arith_yj
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2

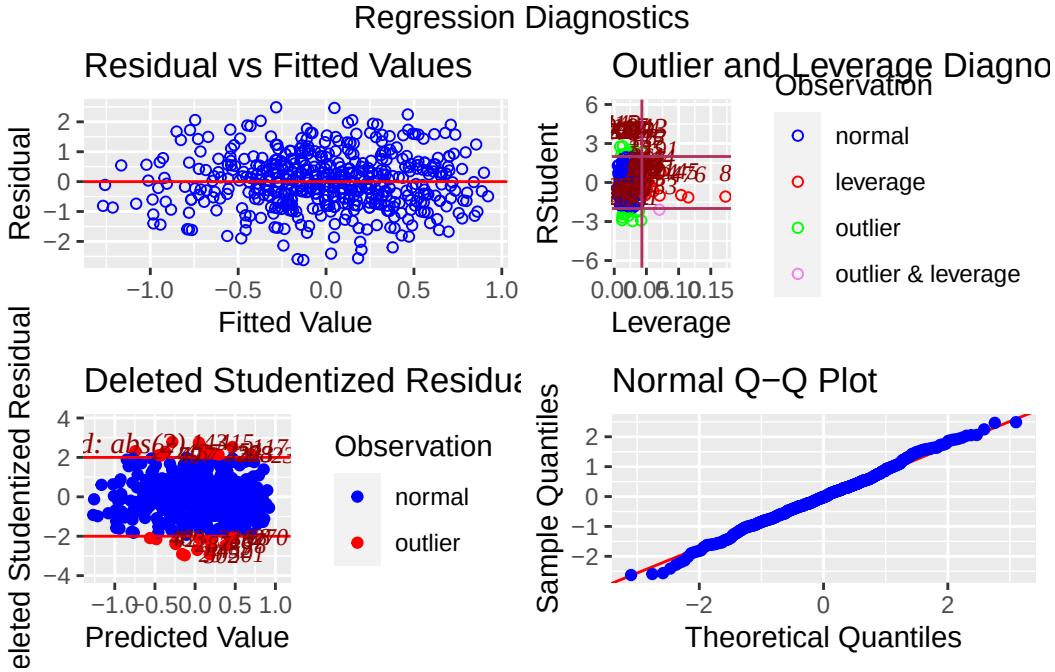

Cook's D Chart
Threshold: 0.008
Cook's D
Observation
0.00
0.01
0.02
0.03
0
100
200
300
400
500
8
254
483
502
42
107
320
476
27
48
89
117
127
175
477
29
68
79
145
164
173
200
212
270
287
323
444
457
504

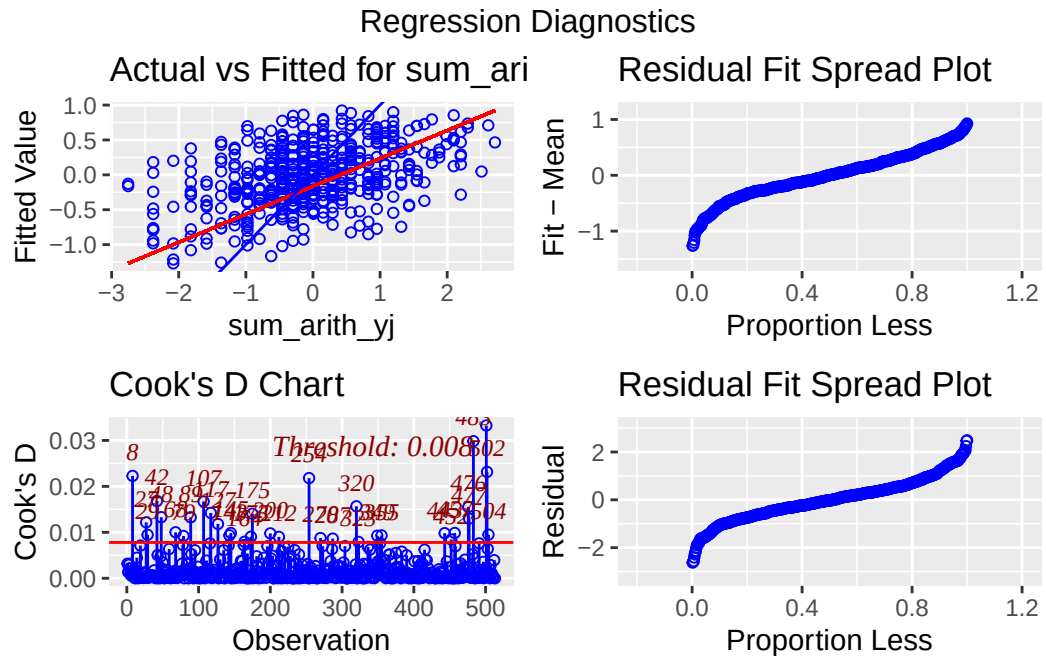

Residual Fit Spread Plot
Fit − Mean
Proportion Less
1
0
−1
0.0
0.4
0.8
1.2

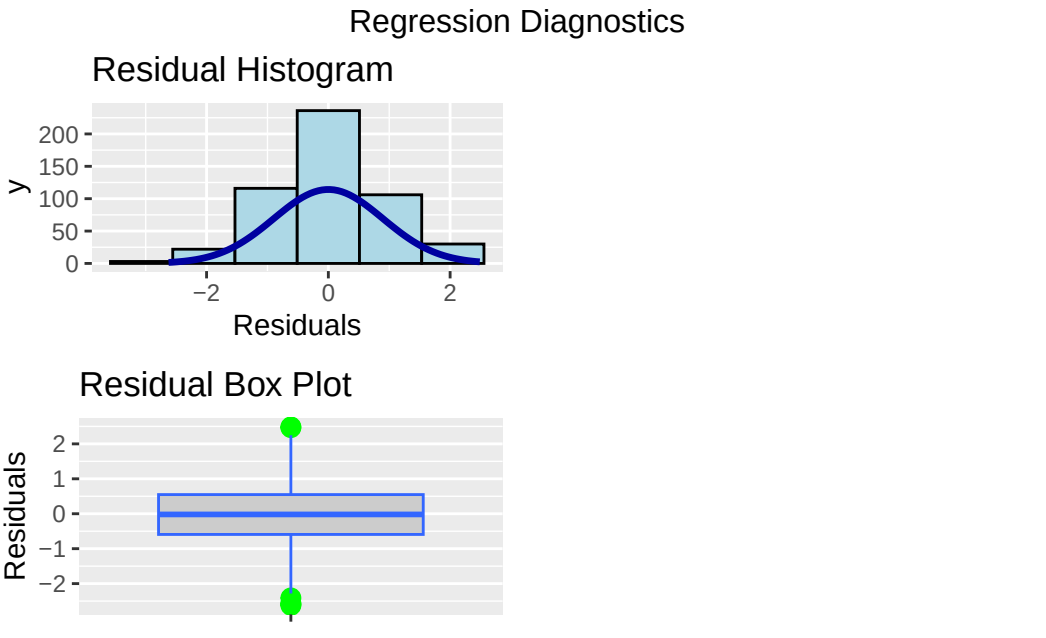

Residual Fit Spread Plot
Residual
2
0
−2
0.0
0.4
0.8
1.2
Proportion Less


Residual Histogram
y
200
150
100
50
0
−2
0
2
Residuals

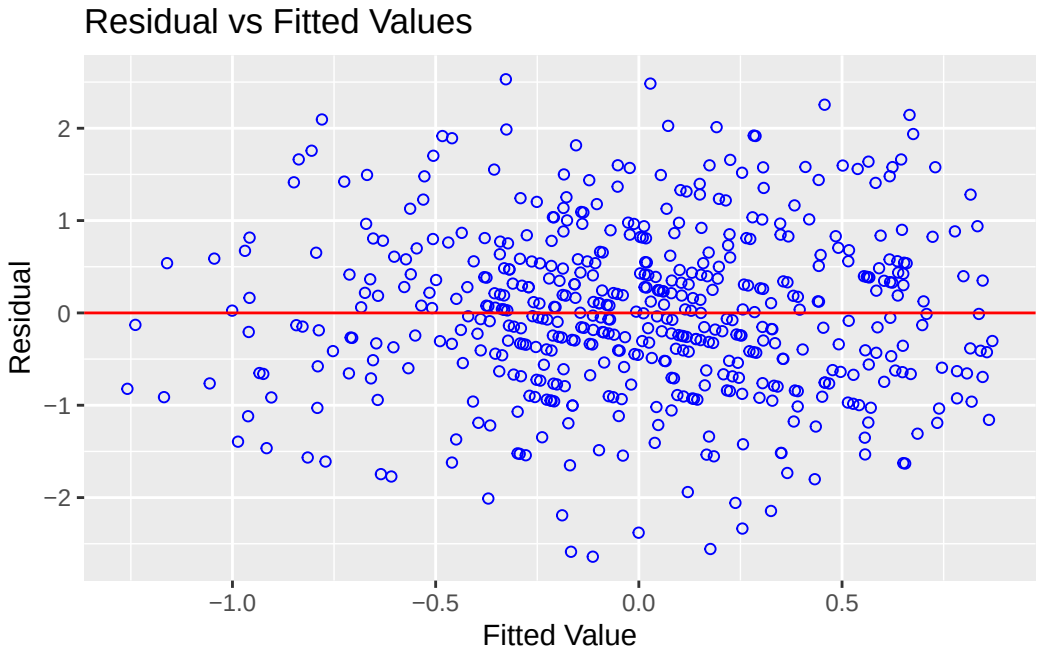

Residual Box Plot
Residuals
2
1
0
−1
−2

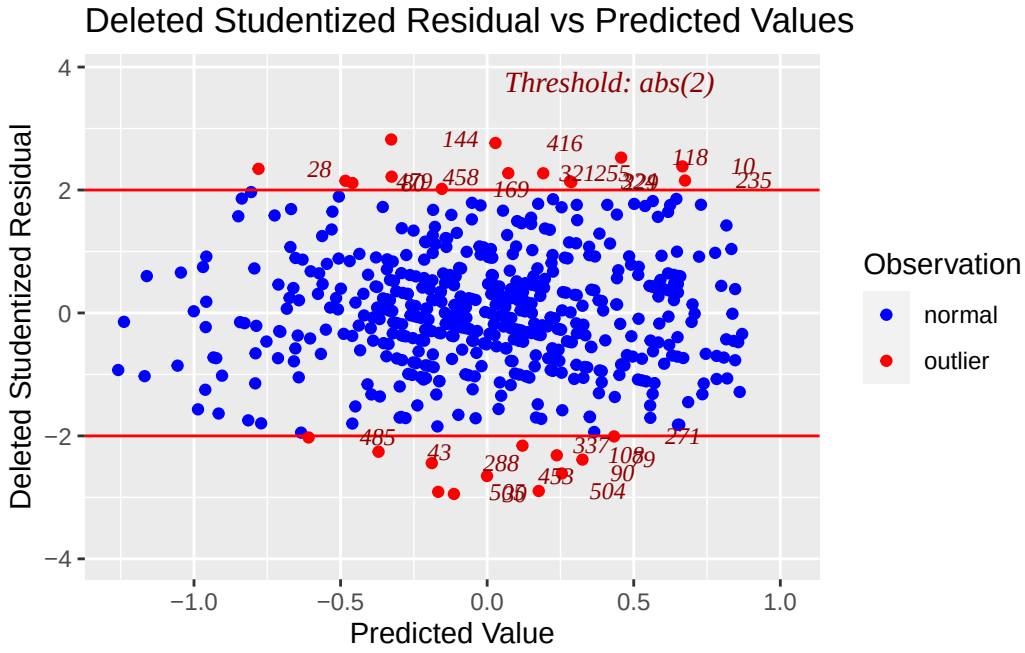

Regression Diagnostics
Residual vs Fitted Values
Residual
Fitted Value
Outlier and Leverage Diagno
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage
Deleted Studentized Residua
eleted Studentized Residual
Predicted Value
Observation
normal
outlier
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles

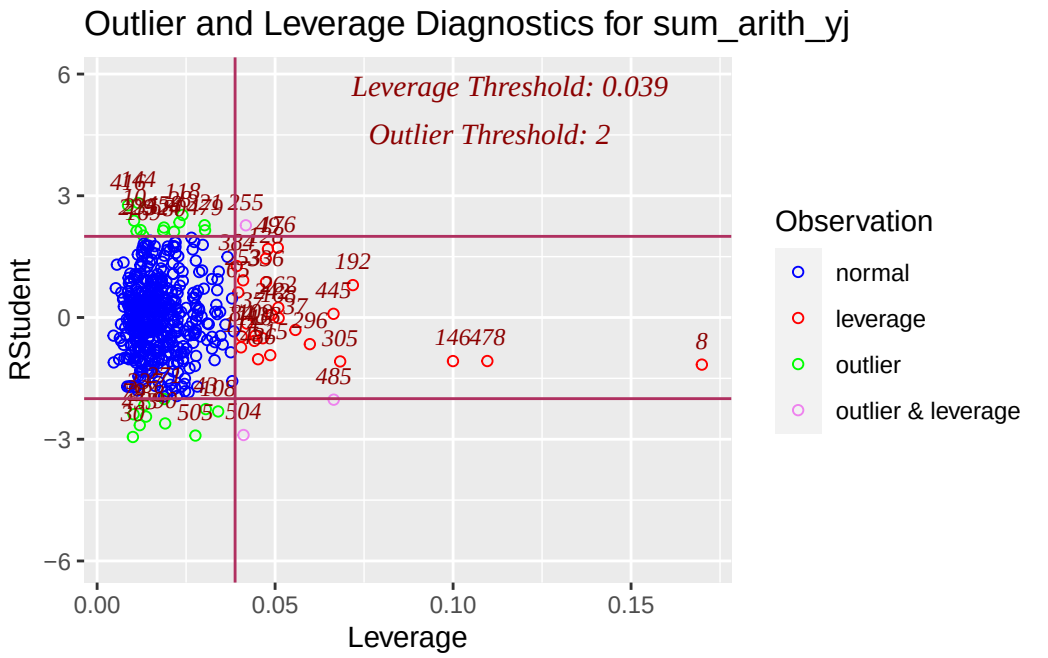


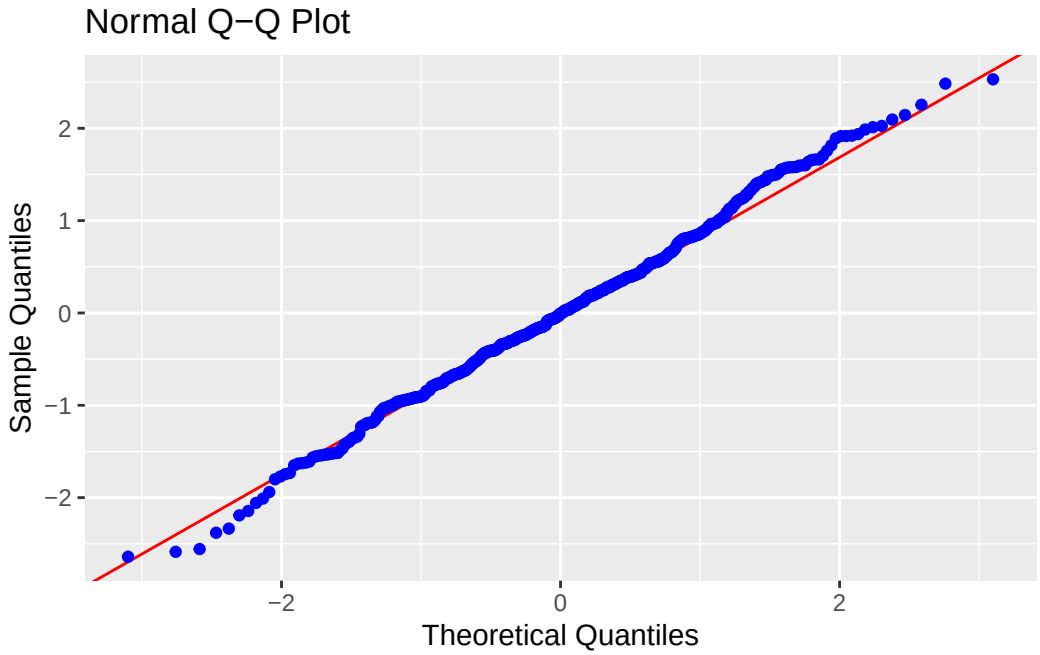


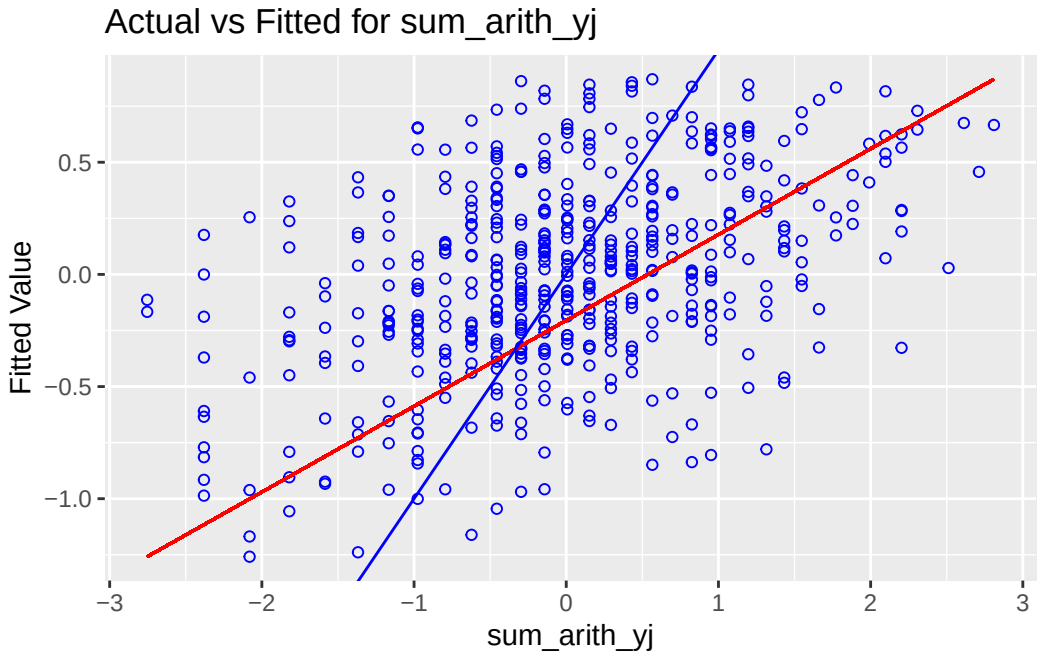

```
##### No SDQ_L model.
transformedmodel3 <- lm(sum_arith_yj ~ score_AMAS_total + sex + age + score_BFI_N + score_GAI
ols_plot_diagnostics(transformedmodel3)
```

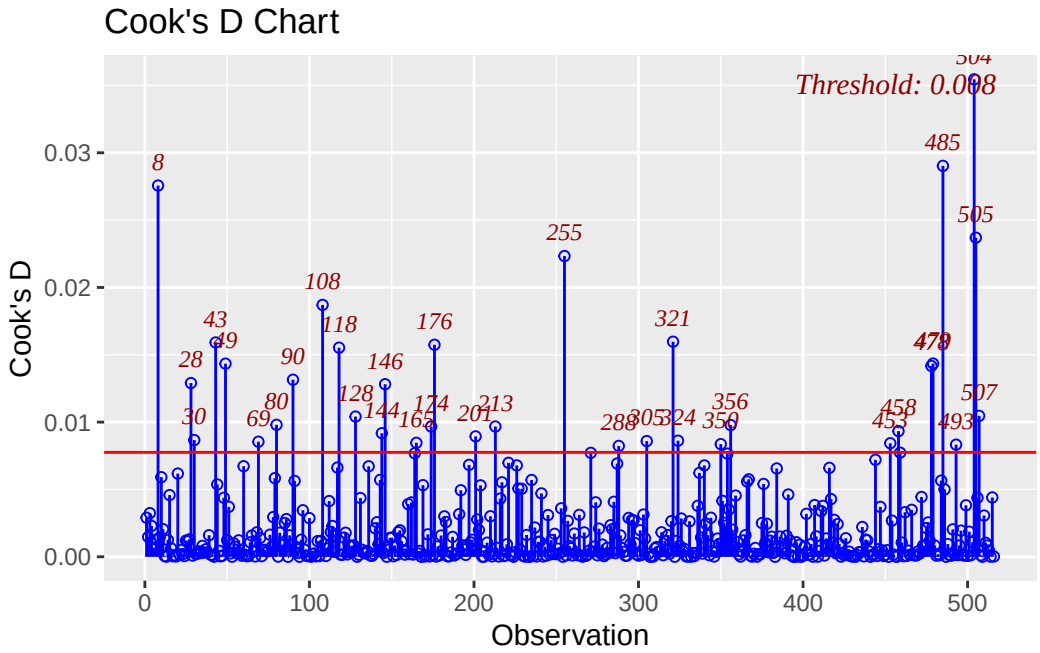

Residual vs Fitted Values
Residual
Fitted Value
2
1
0
−1
−2
−1.0
−0.5
0.0
0.5

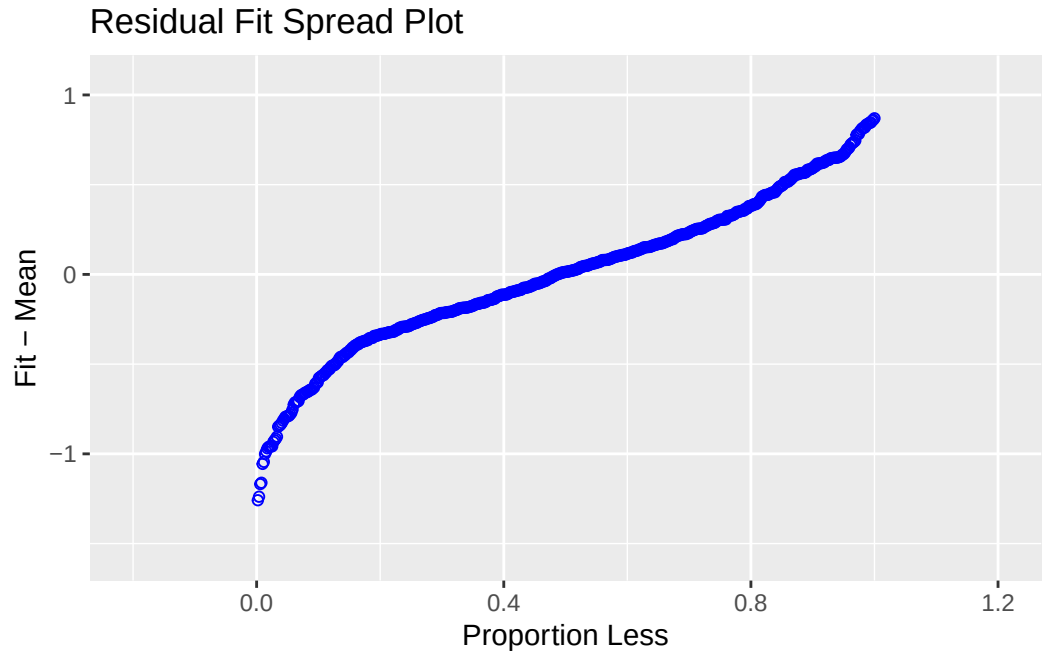

Deleted Studentized Residual vs Predicted Values
Threshold: abs(2)
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier
144
416
118
10
28
458
169
235
485
43
337
271
288
90
504
4
2
0
−2
−4
−1.0
−0.5
0.0
0.5
1.0

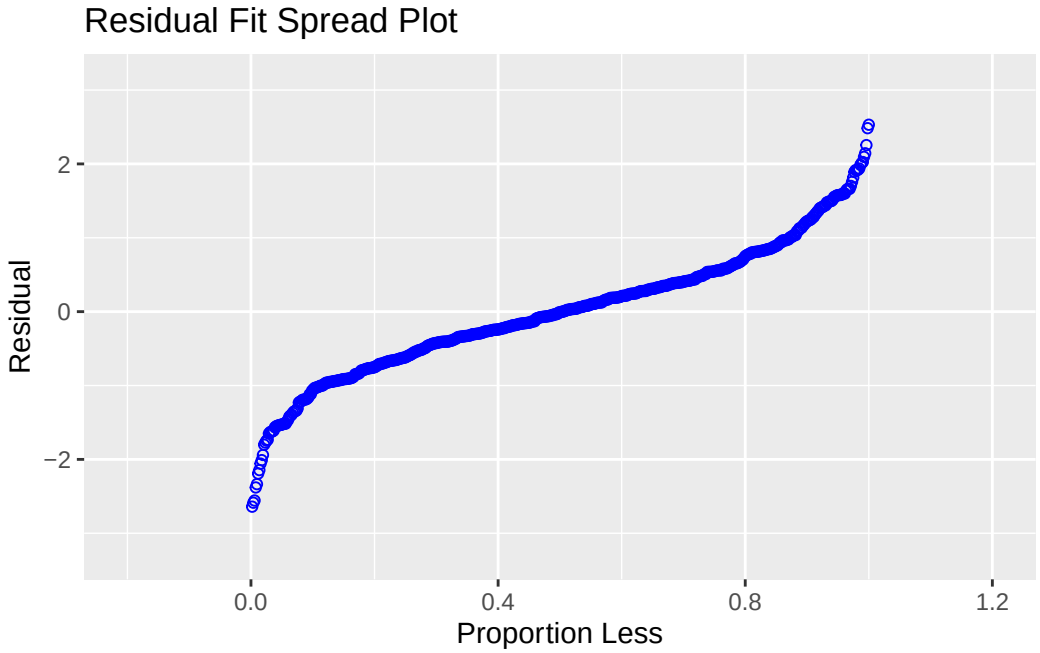

Outlier and Leverage Diagnostics for sum_arith_yj
Leverage Threshold: 0.039
Outlier Threshold: 2
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage

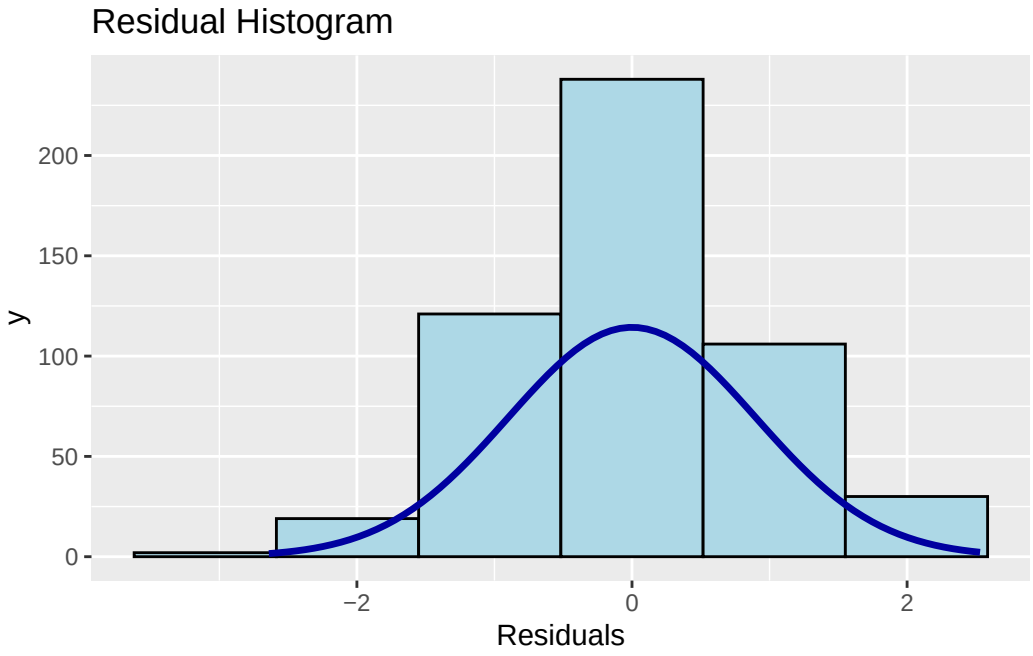

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles

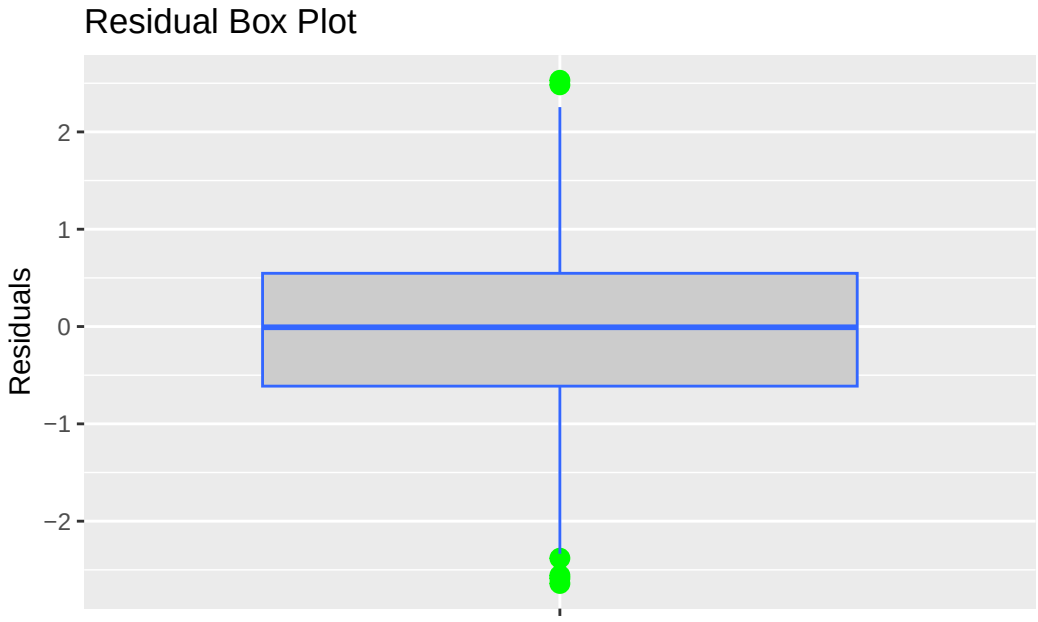

Actual vs Fitted for sum_arith_yj
Fitted Value
sum_arith_yj
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3

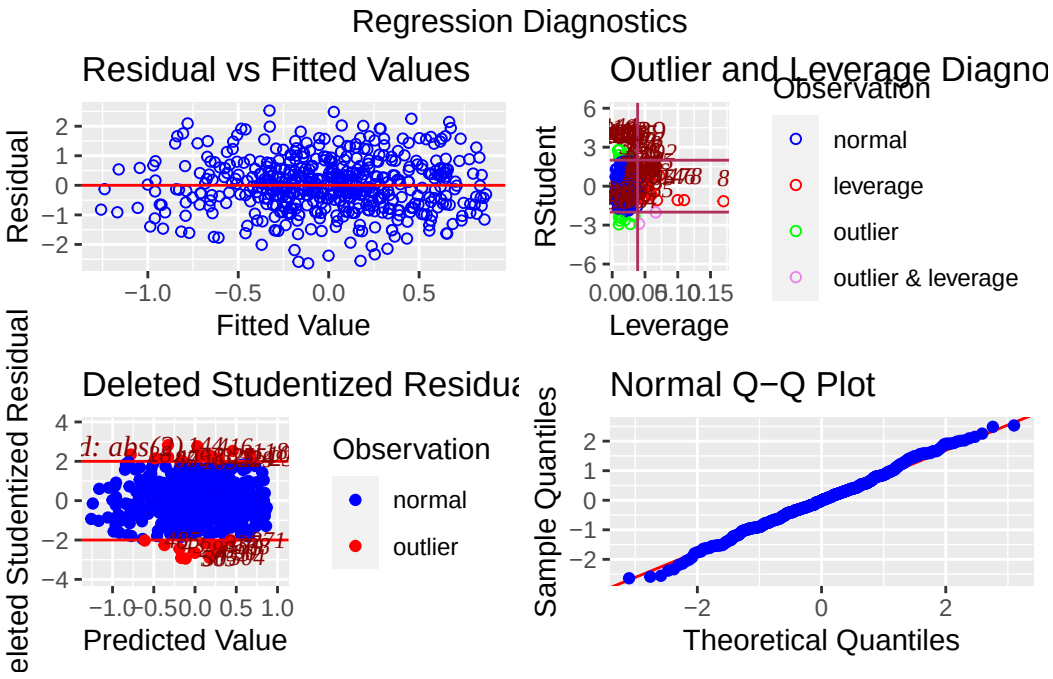

Cook's D Chart
Threshold: 0.008
Cook's D
Observation
0.03
0.02
0.01
0.00
0
100
200
300
400
500
8
255
108
118
176
321
43
49
28
90
146
128
144
174
213
30
69
80
165
201
288
305
324
350
356
458
453
485
505
507
493

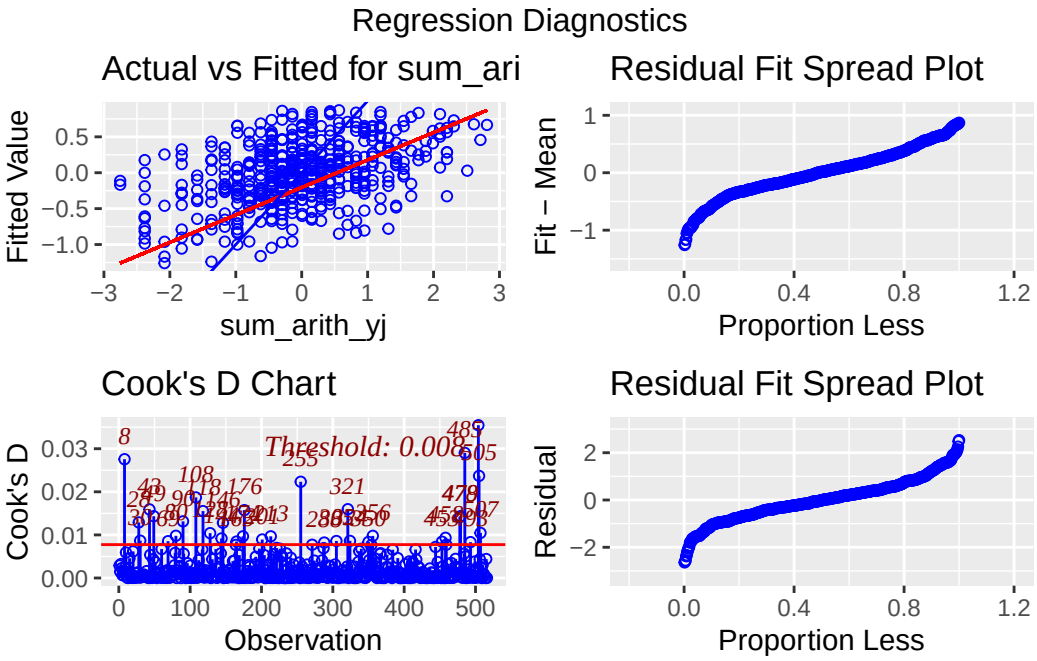

Residual Fit Spread Plot
Fit – Mean
1
0
−1
0.0
0.4
0.8
1.2
Proportion Less

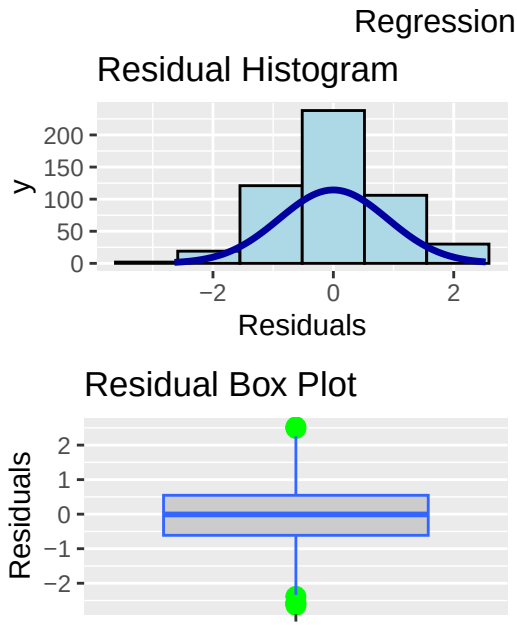

Residual Fit Spread Plot
Residual
2
0
−2
0.0
0.4
0.8
1.2
Proportion Less


Residual Histogram
200
150
100
50
0
y
−2
0
2
Residuals

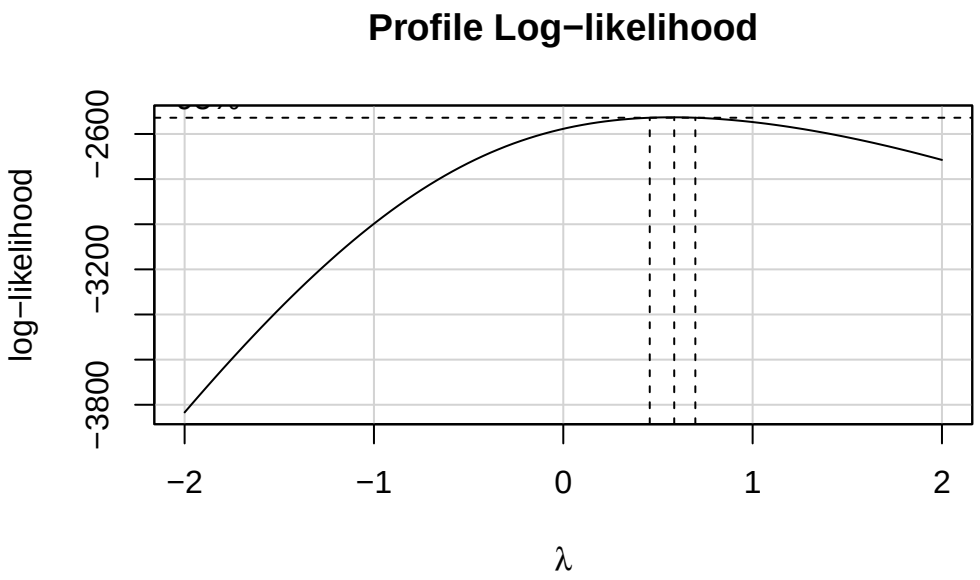

Residual Box Plot
2
1
0
−1
−2
Residuals


Regression Diagnostics
Residual vs Fitted Values
Residual
Fitted Value
Outlier and Leverage Diagno
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage
Deleted Studentized Residua
eleted Studentized Residual
Predicted Value
Observation
normal
outlier
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles


Regression Diagnostics
Actual vs Fitted for sum_ari
Fitted Value
sum_arith_yj
Residual Fit Spread Plot
Fit − Mean
Proportion Less
Cook's D Chart
Cook's D
Threshold: 0.008
Observation
Residual Fit Spread Plot
Residual
Proportion Less




```
a <- boxCox(lambdamodel, family = "yjPower", plotit = TRUE)
```

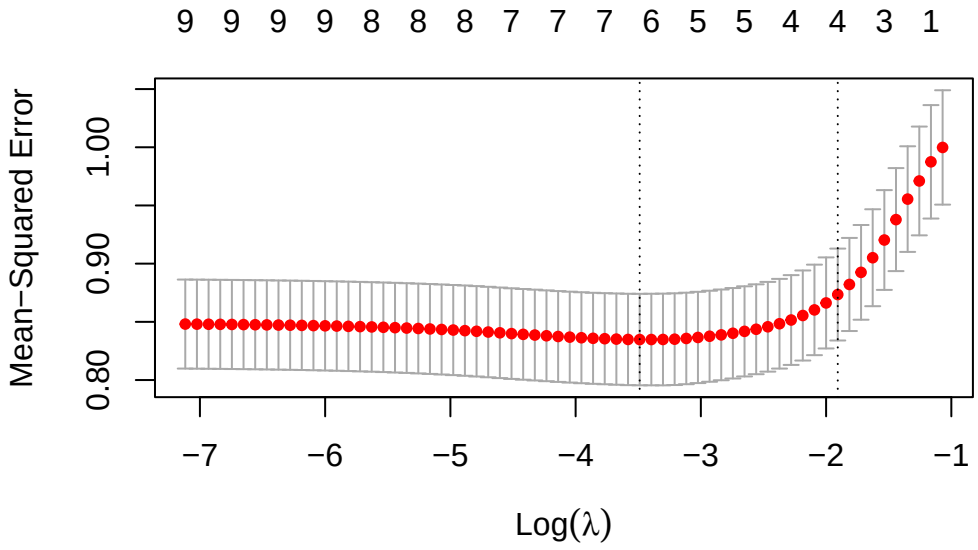


```
optimal_lambda2 <- a$x[which.max(a$y)]
optimal_lambda2
## [1] 0.5858586
```

Removing the outliers cleaned up the numbers a little but didn't address the issue of mild nonlinearity at certain points for the regression model. Cutting SDQ_L did the same thing, made the QQ plot a little nicer but not so much that we don't notice that weird overfitting at the higher end of the theoretical quantiles. We'll see if we can clean up these variables a little bit more before running our models.

Okay, LASSO. Let's just run it and see what we get.

```
R.version.string
## [1] "R version 4.2.3 (2023-03-15)"
if (!requireNamespace("BiocManager", quietly = TRUE)) {
  install.packages("BiocManager")
}
library(glmnet)
## we need to turn the dataframe into a matrix
x2 <- model.matrix(~ score_AMAS_total + sex + age + score_BFI_N + score_GAD + score_PISA_ME

x2 <- x2[, -3]
y2 <- scaled_training2$sum_arith_yj

set.seed(151)
cv_model <- cv.glmnet(x2, y2, alpha = 1)

plot(cv_model)
```




```
# The upper and lower bars are the MSE +- standard error.
set.seed(151)
best_lambda <- cv_model$lambda.min
print(best_lambda)
## [1] 0.03054552
# after we set the lambda, we run the model!
set.seed(152)
lassomodel <- glmnet(x2, y2, alpha = 1, lambda = best_lambda)

lassomodel
##
## Call:  glmnet(x = x2, y = y2, alpha = 1, lambda = best_lambda)
##
##   Df  %Dev  Lambda
## 1  6 18.62 0.03055
lassomodel$dev.ratio
## [1] 0.1861926
lassomodel # is the fraction of (null) deviance explained. Pretty good GOF estimator but we'l
##
## Call:  glmnet(x = x2, y = y2, alpha = 1, lambda = best_lambda)
##
##   Df  %Dev  Lambda
## 1  6 18.62 0.03055

coef(lassomodel)
## 11 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                                 s0
## (Intercept)             0.278185654
## score_AMAS_total       -0.023246211
## sexf                   -0.406639652
## age                     .
## score_BFI_N             .
## score_GAD               .
## score_PISA_ME           0.144757064
## score_STAI_state_short -0.036017334
## score_TAI_short         .
## score_SDQ_M             0.168384645
## score_SDQ_L             0.006036009


# SECOND TRY - NOW WITH LAMBDA that minimizes CV error plus one standard error.
set.seed(151)
best_lambda2 <- cv_model$lambda.1se
print(best_lambda2)
## [1] 0.1485307
# after we set the lambda, we run the model!
set.seed(152)
lassomodel2 <- glmnet(x2, y2, alpha = 1, lambda = best_lambda2)

lassomodel2
##
## Call:  glmnet(x = x2, y = y2, alpha = 1, lambda = best_lambda2)
##
##   Df  %Dev Lambda
## 1  4 14.14 0.1485
print(coef(lassomodel2))
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                 s0
## (Intercept)             0.138003983
## score_AMAS_total       -0.001860801
## sexf                   -0.201728201
## age                     .
## score_BFI_N             .
## score_GAD               .
## score_PISA_ME           0.113073420
## score_STAI_state_short  .
## score_TAI_short         .
## score_SDQ_M             0.095257893
```

```
## score_SDQ_L              .

lassomodel2$dev.ratio # is the fraction of (null) deviance explained. Pretty good GOF estimat
## [1] 0.1413626
# This may be a better model, tbh. Let's test later.


## GOOD IDEA: CHECK FOR DIAGNOSTIC PLOTS (AND STATS) OF YOUR TWO LASSO MODELS. SEE IF THERE
## CHECK FOR OTHER THINGS LIKE ROBUST LASSO/ADAPTIVE LASSO ETC. SEE WHAT WOULD FIT YOUR DATA
# THE ROBUSTIFICATION OF THE LASSO AND ELASTIC NET DISSERTATION BY (?MATT MULTACH)
```

Keep in mind that the cross-validation we used is attempting to estimate the value of lambda that leads to the minimal prediction error. In other words, the lambda is being selected to predict the best, NOT to select the "best" model. The lambda that is chosen using cross validation is chosen using MSE, which may lead to a "bigger" model than if we were trying to optimize the lambda to "produce the correct model"/"select the relevant predictors". This is a huge caveat, and something you need to consider when you are thinking about WHY you are using LASSO. You can read more at the following link:

(more here: https://stats.stackexchange.com/questions/353185/why-using-cross-validation-is-not-a-good-option-for-lasso-regression)

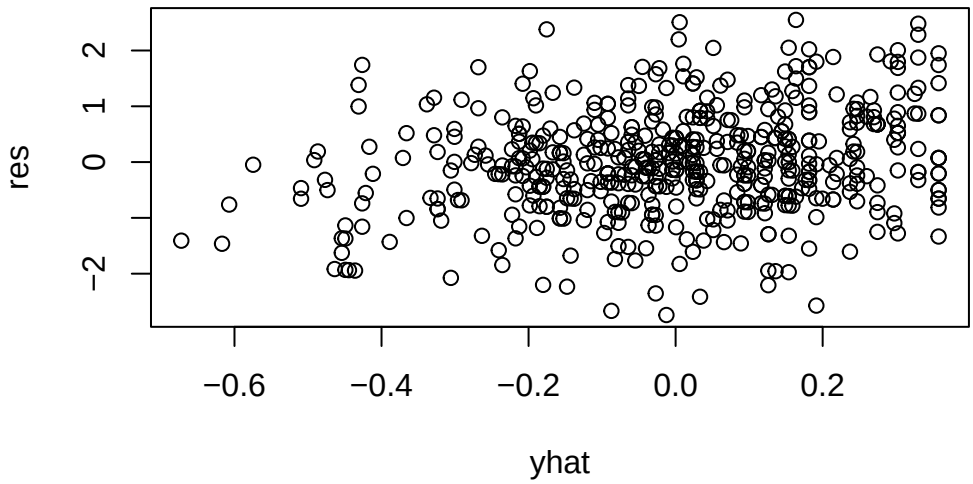
You can "I have had much better luck applying Lasso( ) to select the model and then fitting by least squares, then applying cross-validation to this entire procedure to select . It's still not ideal, but it is a big improvement."

or this "for the particular training fold, CV is applied again to find an optimal in that iteration; or apply CV in two batches, first a set of optimization-CV to find an optimal then fix that value when fitting LASSO models in a another batch set of validation-CV steps to assess model error."

###might be better to just see what variables are being used in the lowest prediction error model? or which variables tend to have the lowest prediction error? Can always try this other perspective, or at LEAST discuss it further than what I currently have. Dr. Lai doesn't really believe in "true" models, for example.

```
# Design matrix and response vector


# Cross validation
set.seed(1)
lasso.cv <- cv.glmnet(x2, y2)

# Prediction
```

```r
yhat <- predict(lasso.cv, newx = x2)

# residuals
res <- y2- yhat

plot(res ~ yhat)
```



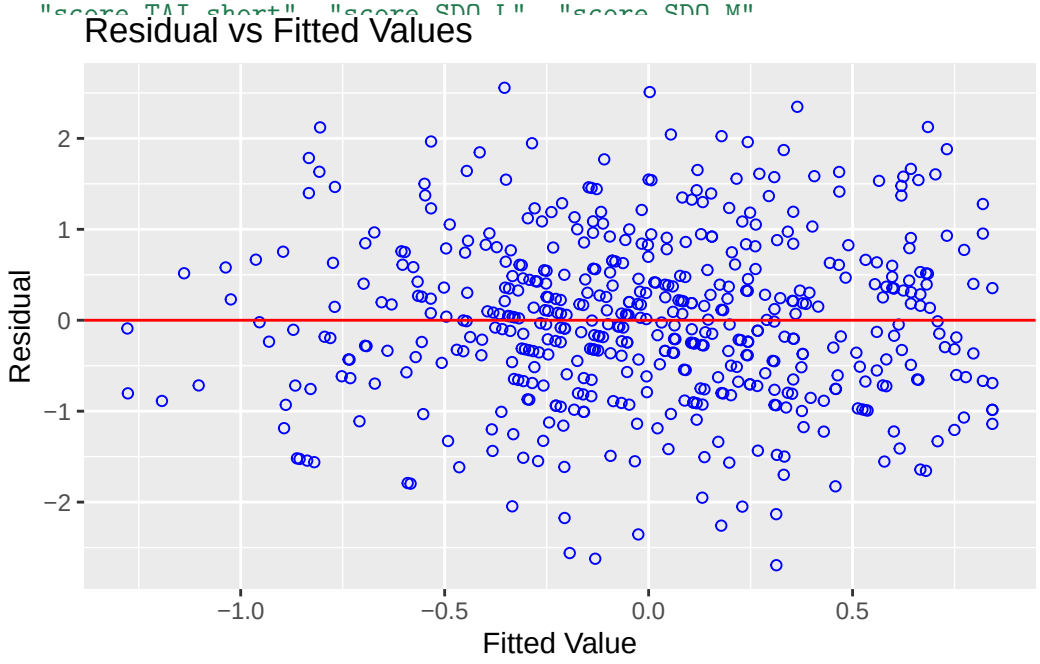
As you can see, there is a slight pattern in the residuals, namely that lower predicted values tend to have lower residuals and higher predicted values tend to have higher residuals. However, LASSO does not call for any particular distribution of its residuals. Recall what LASSO does - it is not a model, with a linear model being denoted by y = XB + e. It is a coefficient estimation method/variable selection method that uses a shrinkage term to anticipate the decrease in model fit when used on test data, which introduces some bias but reduces variance. Because the LASSO function only takes y (the dependent variable) and X (the predictors) while producing B (the coefficients, there are no assumptions regarding the distribution of e, errors. Thus, this residual pattern is not concerning. In fact, it may even be expected due to the fact that LASSO will shift predicted values closer to the mean, meaning (get it?) residuals may display a pattern like the one we see above.

Let's keep looking at both models with the differing values of lambda. I think it's probably fair to say that the model with the lambda.min value may be better at prediction, but may not contain the "correct" subset of variables, whereas the lambda.1se value produces a more sparse model that is more likely to contain the true set of variables.

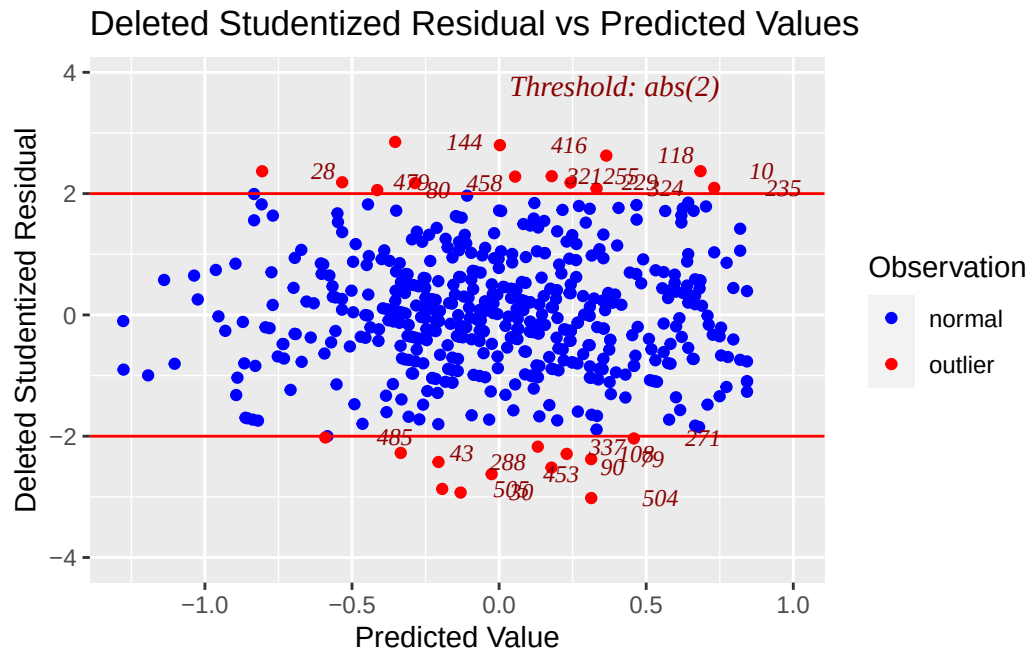
Let's try including the selected variables of both values of lambda in a random forest model and see the models' stats.

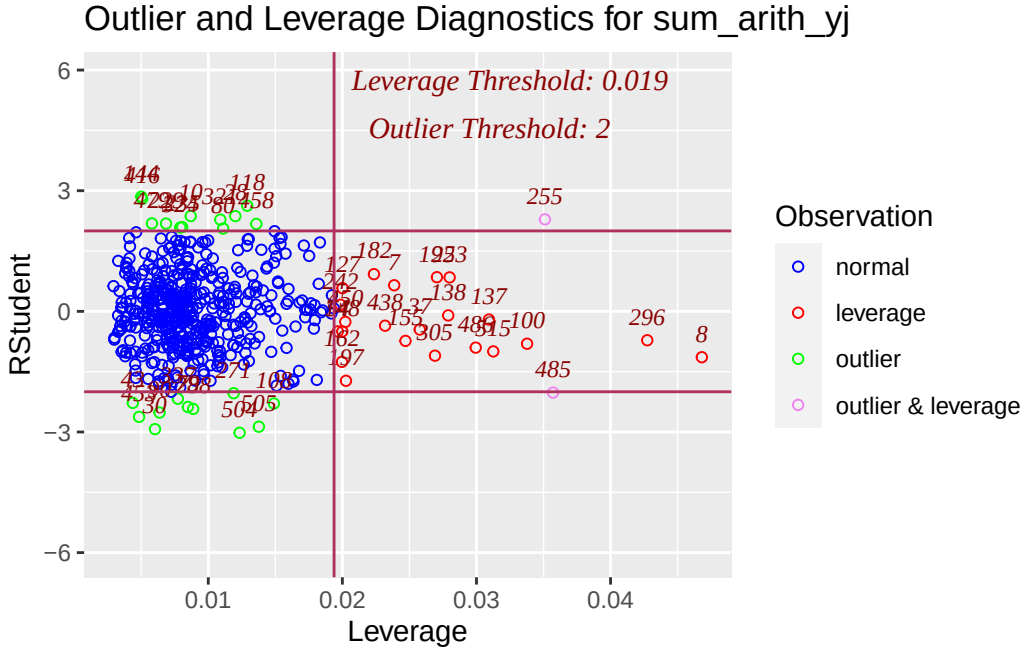
```r
library(caret)
#### TESTING ACCURACY OF MODEL ON TEST SET!
```

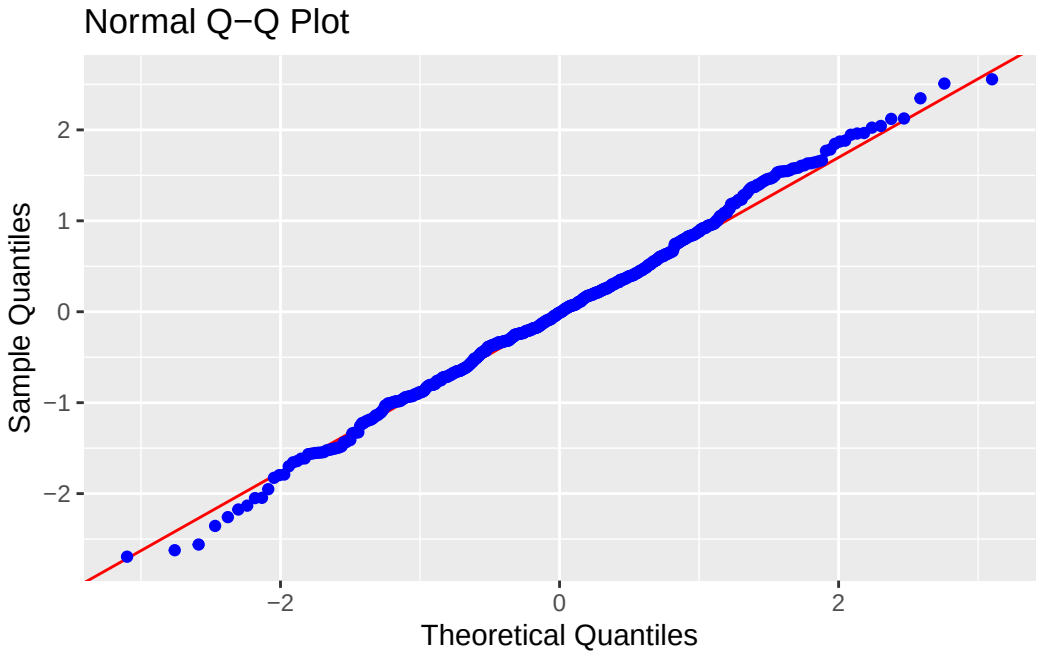
```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.69392 -0.61634 -0.01085  0.55041  2.55653
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.14685    0.04902   2.995  0.00287 **
## sexm                     0.46488    0.09048   5.138 3.96e-07 ***
## score_PISA_ME            0.15290    0.05059   3.022  0.00263 **
## score_STAI_state_short -0.06615    0.04105  -1.611  0.10772
## score_SDQ_H              0.20047    0.04815   4.164 3.68e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9048 on 511 degrees of freedom
## Multiple R-squared:  0.1878, Adjusted R-squared:  0.1814
## F-statistic: 29.53 on 4 and 511 DF,  p-value: < 2.2e-16
ols_plot_diagnostics(linearassomodel2)
```

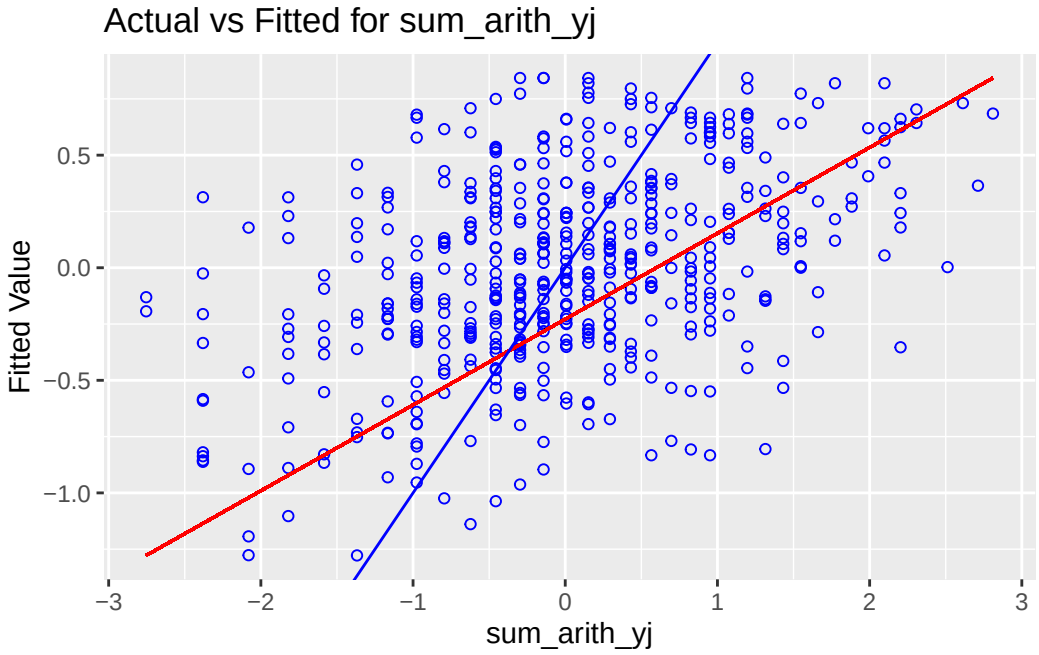


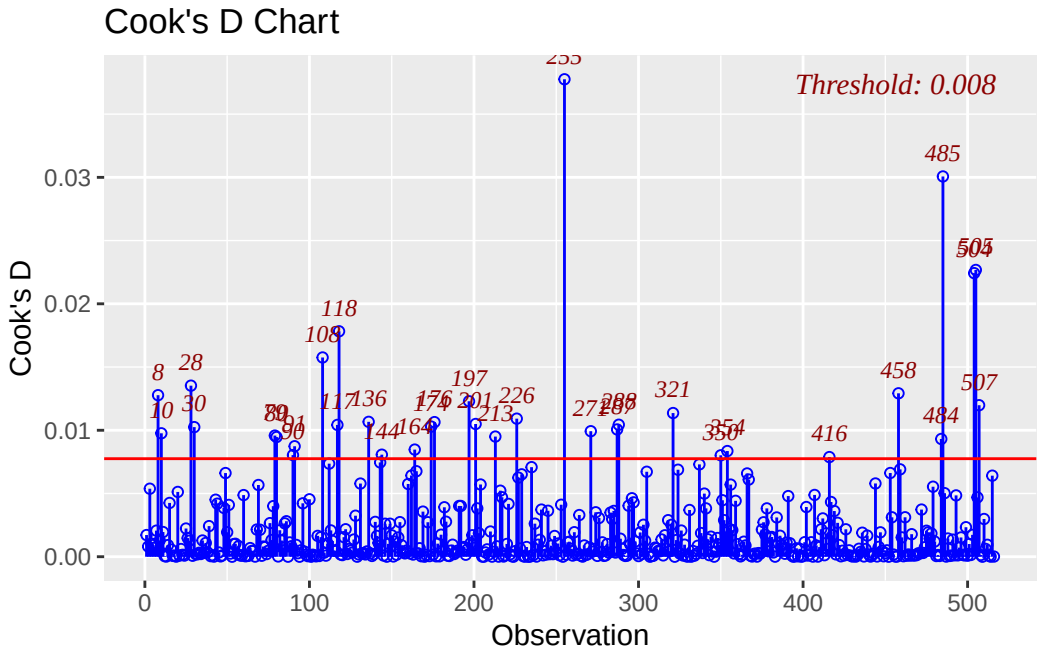

```
# first, find scaling parameters.

scale(training2$score_BFI_N) # mean 23.41085, 6.200986
##               [,1]
##   [1,]  0.55739363
##   [2,]  0.23495516
##   [3,]  1.04105134
##   [4,] -0.73236024
##   [5,]  0.23495516
##   [6,]  0.71861287
##   [7,]  0.71861287
##   [8,]  2.33080522
##   [9,]  0.07373593
##  [10,] -0.89357948
##  [11,]  0.71861287
##  [12,]  1.36348981
##  [13,]  1.36348981
##  [14,] -1.53845642
##  [15,] -0.57114101
```

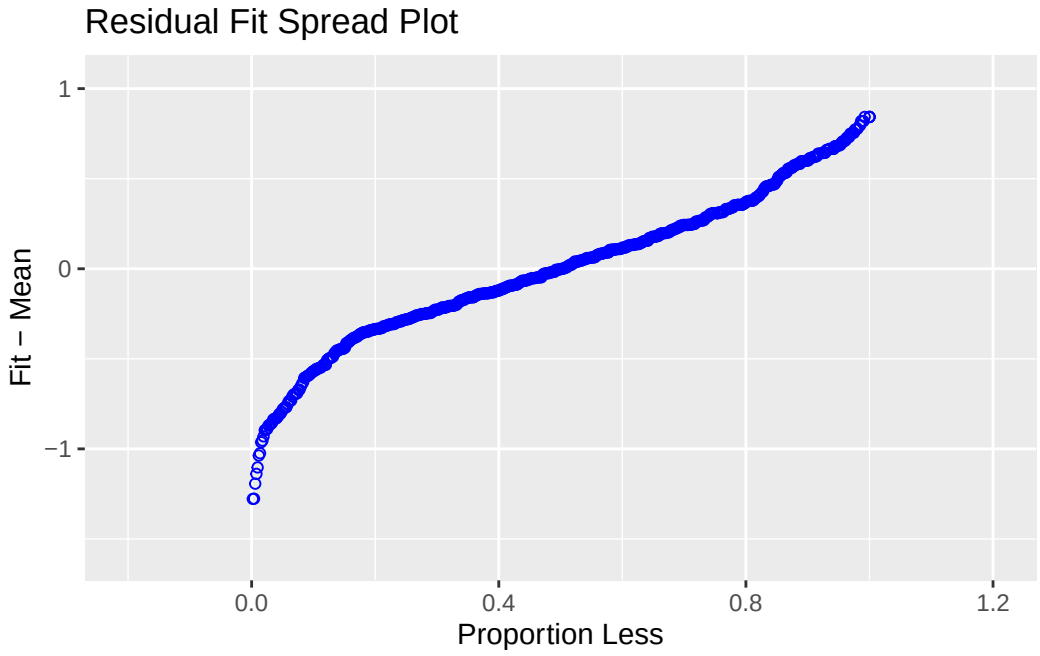

Deleted Studentized Residual vs Predicted Values
Threshold: abs(2)
Deleted Studentized Residual
Predicted Value
Observation
normal
outlier

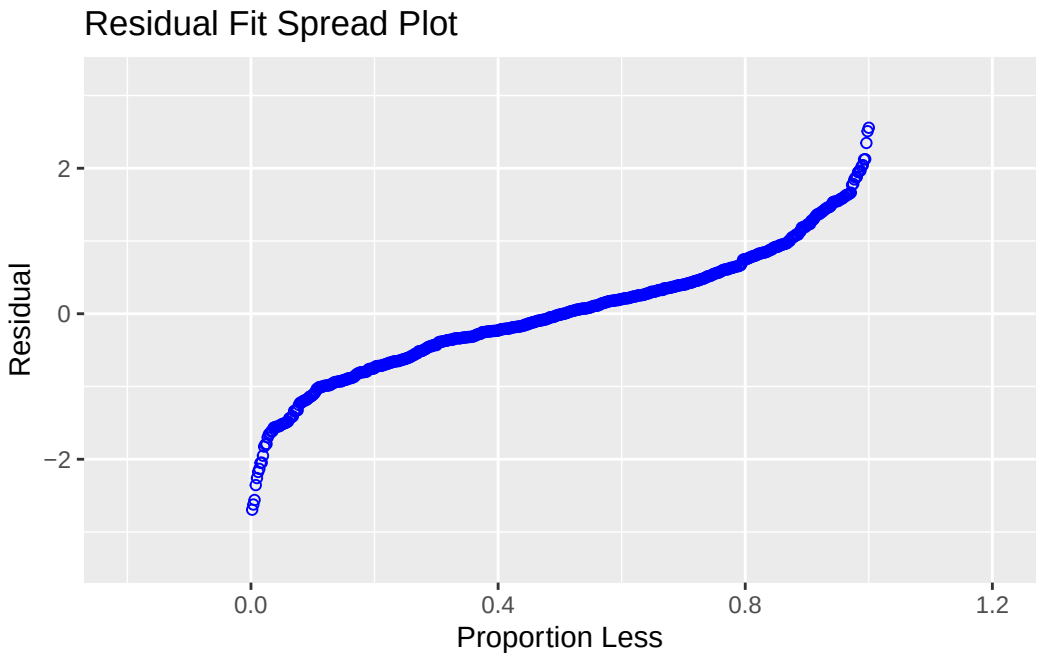

Outlier and Leverage Diagnostics for sum_arith_yj
Leverage Threshold: 0.019
Outlier Threshold: 2
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage

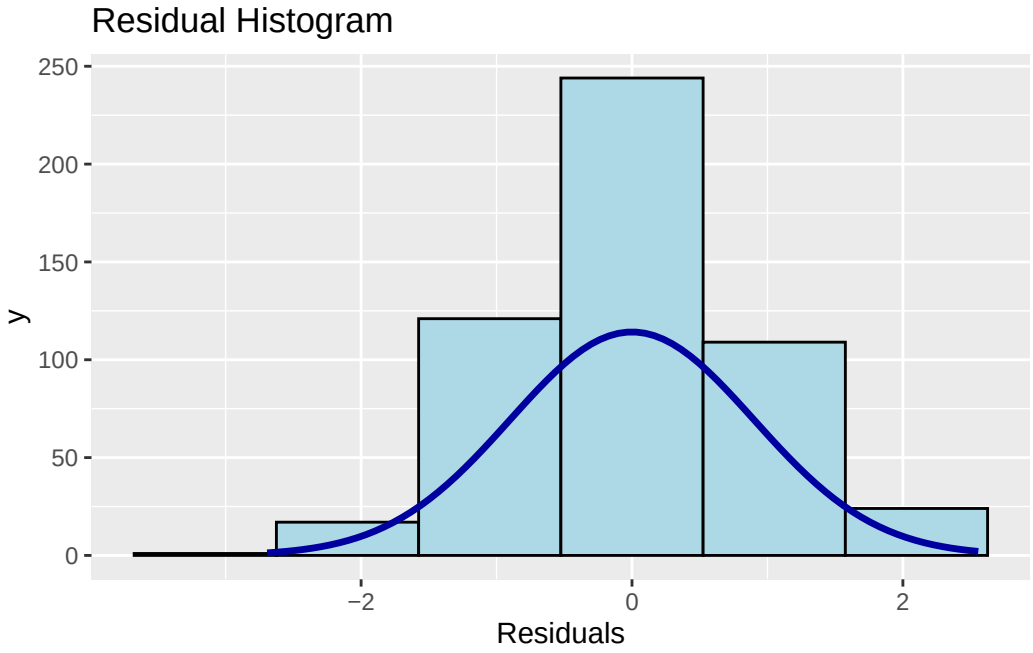

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
2
1
0
−1
−2
−2
0
2

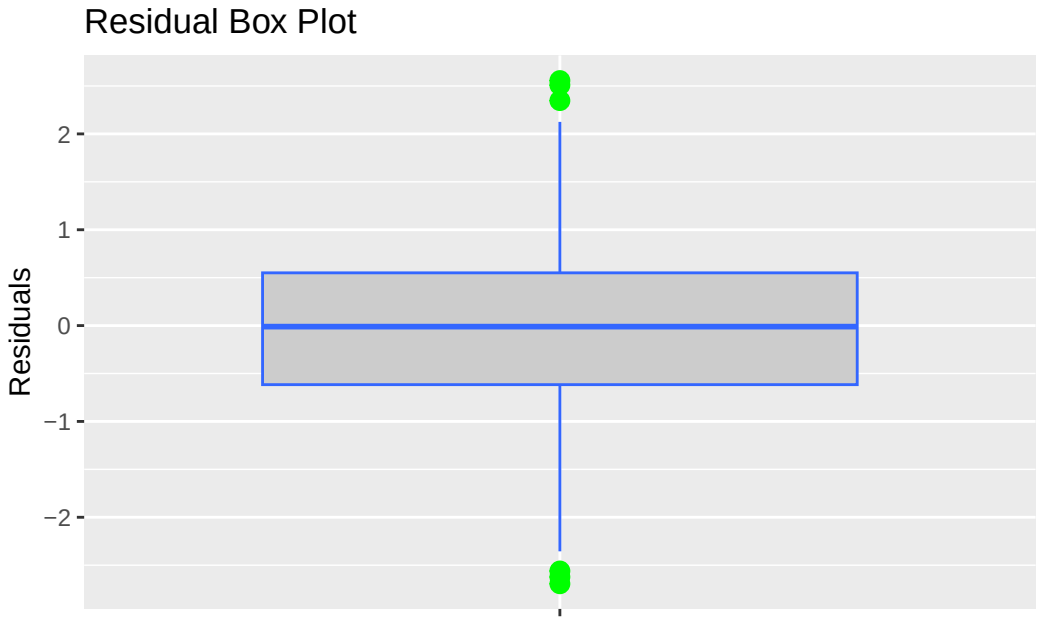

Actual vs Fitted for sum_arith_yj
Fitted Value
sum_arith_yj
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3

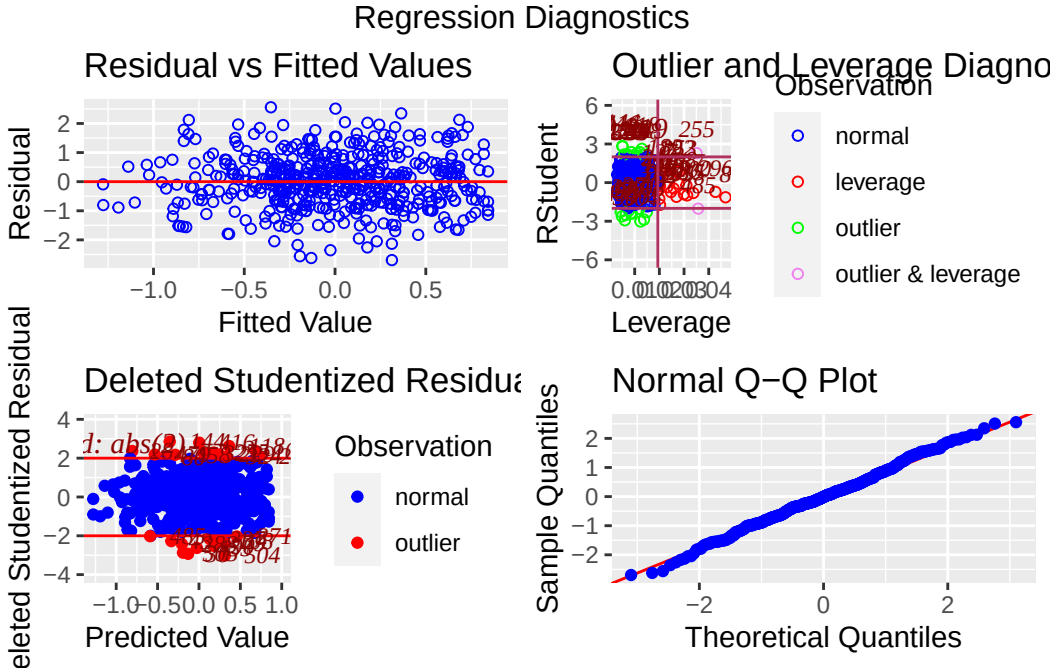

Cook's D Chart
Threshold: 0.008
Cook's D
Observation
0.00
0.01
0.02
0.03
0
100
200
300
400
500

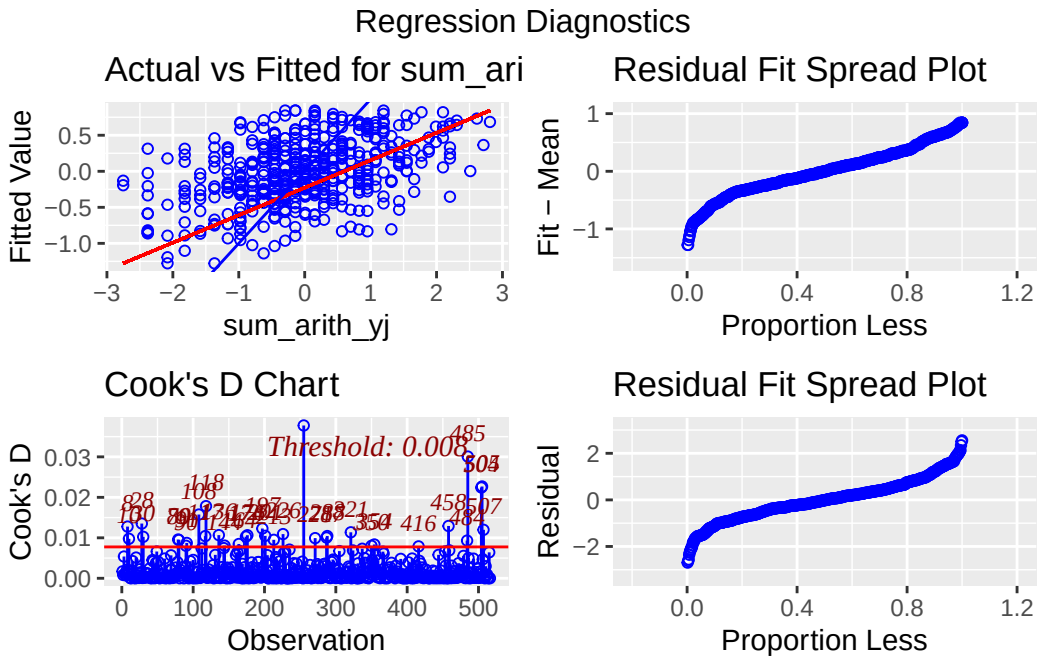

Residual Fit Spread Plot
Fit − Mean
Proportion Less
1
0
−1
0.0
0.4
0.8
1.2

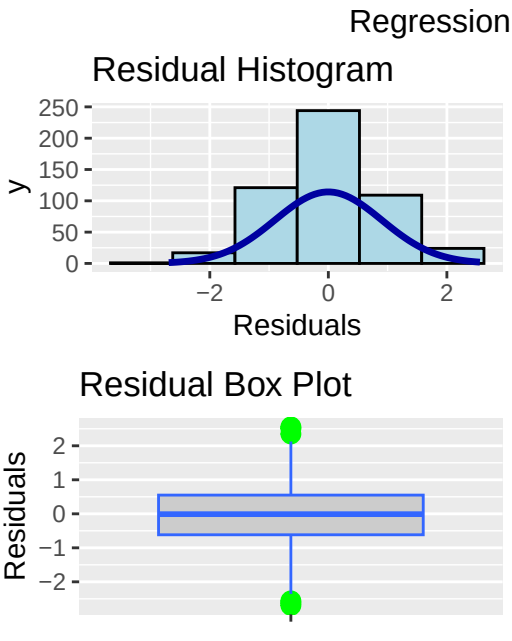

Residual Fit Spread Plot
Residual
2
0
−2
0.0
0.4
0.8
1.2
Proportion Less


Residual Histogram
y
250
200
150
100
50
0
−2
0
2
Residuals


Residual Box Plot
Residuals
2
1
0
−1
−2


Regression Diagnostics
Residual vs Fitted Values
Residual
Fitted Value
Outlier and Leverage Diagno
RStudent
Leverage
Observation
normal
leverage
outlier
outlier & leverage
Deleted Studentized Residua
eleted Studentized Residual
Predicted Value
Observation
normal
outlier
Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles






```
# 1se version
set.seed(101)
rf_lasso_1se <- train(sum_arith_perf ~ sex + score_PISA_ME + score_SDQ_M, data = scaled_trai
## note: only 2 unique complexity parameters in default grid. Truncating the grid to 2 .
```

```
print(rf_lasso_1se)
## Random Forest
##
## 516 samples
##   3 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 463, 465, 462, 466, 465, 466, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared    MAE
##   2     6.593140  0.11157109  5.212328
##   3     6.800425  0.09706039  5.384470
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.


predictionslasso1semodel <- predict(rf_lasso_1se, scaled_test2)
validationlasso1semodel <- data.frame(
  R2 = R2(predictionslasso1semodel, scaled_test2$ sum_arith_perf),
  RMSE = RMSE(predictionslasso1semodel, scaled_test2$ sum_arith_perf),
  MAE = MAE(predictionslasso1semodel, scaled_test2$ sum_arith_perf)
) # these statistical functions are in caret technically under trainControl(), which passes t
print(validationlasso1semodel)
##           R2     RMSE      MAE
## 1 0.06008094 6.448351 5.052809
```

As we have determined across the two analysis methods we have used, age may be a useful variable to include in a strictly predictive model, but it does not seem to be a "key" variable in the sense that it has very little relationship with the dependent variable. Age may help clear up the picture of the dependent variable, but without sex, math self efficacy and math self-concept, it is essentially worthless. However, it does in fact seem that age can be useful in predicting the test data, so one should lean towards including the variable if they are strictly interested in prediction. In fact, given the relative weakness of most of our predictors, it is not unlikely that the full 10-predictor model (or 9-predictor, sans SDQ_L) may have the best predictive power on a larger test set, even if it includes several relatively unimportant variables.

It appears clear that the second LASSO model only reports the necessary variables to explain the DV's variance, whereas the first model includes covariates to produce an "optimal" model

without too many covariates, but enough to have better predictive power than say, a 5 variable model. In fact, such a model is likely better at prediction than the fourth-selection model, which despite having the highest R squared of any model so far, may not generalize predictive ability as well as the first LASSO model outside of the relatively small test sample size. Across models, the inclusion of the age variable does not seem to contribute highly to the R squared of our models. The other seemingly key covariates of state anxiety and possibly text anxiety seem to add greater benefit in predictive modeling after we have naturally included the three main variables of sex, math self-efficacy, and math self-concept.

I also want to check out another LASSO model, one that popped up when I ran a different cv.glm for the lambda value. This one includes the seemingly key measures of anxiety.

```
library(caret)



tunegrid <- expand.grid(.mtry = c(2:5))
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "grid")

set.seed(101)
rf_lasso2 <- train(sum_arith_perf ~ sex + score_AMAS_total + score_STAI_state_short + score_
print(rf_lasso2)
## Random Forest
##
## 516 samples
##   5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 463, 465, 462, 466, 465, 466, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared    MAE
##   2     6.557890  0.10511501  5.091829
##   3     6.653343  0.09503871  5.156656
##   4     6.705255  0.09045477  5.202158
##   5     6.738521  0.08725472  5.230872
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.



predictionslassomodel2 <- predict(rf_lasso2, scaled_test2)
```

```
validationlassomodel2 <- data.frame(
  R2 = R2(predictionslassomodel2, scaled_test2$ sum_arith_perf),
  RMSE = RMSE(predictionslassomodel2, scaled_test2$ sum_arith_perf),
  MAE = MAE(predictionslassomodel2, scaled_test2$ sum_arith_perf)
) # these statistical functions are in caret technically under trainControl(), which passes t
print(validationlassomodel2)
##          R2     RMSE      MAE
## 1 0.0655036 6.353721 4.845875
# Insanely strong prediction. This is likely the best model if you're looking for 4-5 variabl
```

Probably should've assumed this model would be poor, but I wanted to test it anyways just to see if LASSO had missed something.

Remember that with the way we selected the lambda value, LASSO is attempting to optimize for prediction. LASSO is NOT attempting to select only non-noise predictors given the "optimal" lambda. Therefore, if you were looking to use a smaller subset of predictors but retain great predictive accuracy, LASSO has you covered. If you're just looking to identify the "important" variables, we will soon look at another LASSO-adjacent method that purports to do just that with the optimal lambda.

Before we try that final method however, let's do a little post-selection inference and try and see how our actual LASSO models look.

```
library(selectiveInference)
## Loading required package: intervals
##
## Attaching package: 'intervals'
## The following object is masked from 'package:Matrix':
##
##     expand
## The following object is masked from 'package:purrr':
##
##     reduce
## The following object is masked from 'package:tidyr':
##
##     expand
## Loading required package: survival
##
## Attaching package: 'survival'
## The following object is masked from 'package:caret':
##
##     cluster
## Loading required package: adaptMCMC
```

```
## Loading required package: parallel
## Loading required package: coda
n <- nobs(lassomodel)
fixedLassoInf(x2, y2, beta = coef(lassomodel, s = best_lambda / n)[-1], lambda = best_lambda
## Warning in fixedLassoInf(x2, y2, beta = coef(lassomodel, s =
## best_lambda/n)[-1], : Solution beta does not satisfy the KKT conditions (to
## within specified tolerances)
##
## Call:
## fixedLassoInf(x = x2, y = y2, beta = coef(lassomodel, s = best_lambda/n)[-1],
##     lambda = best_lambda)
##
## Standard deviation of noise (specified or estimated) sigma = 0.908
##
## Testing results at lambda = 0.031, with alpha = 0.100
##
##  Var   Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
##    1 -0.020  -0.349   0.727    -0.097    0.465       0.050      0.000
##    2 -0.465  -5.060   0.000    -0.633   -0.312       0.049      0.048
##    6  0.155   3.029   0.002     0.068    0.241       0.050      0.049
##    7 -0.057  -1.274   0.181    -0.207    0.058       0.050      0.050
##    9  0.196   3.381   0.001     0.106    0.474       0.048      0.050
##   10  0.045   1.094   0.275    -0.077    0.112       0.050      0.048
##
## Note: coefficients shown are partial regression coefficients
##

n2 <- nobs(lassomodel2)
fixedLassoInf(x2, y2, beta = coef(lassomodel2, s = best_lambda2 / n2)[-1], lambda = best_lam
## Warning in fixedLassoInf(x2, y2, beta = coef(lassomodel2, s =
## best_lambda2/n2)[-1], : Solution beta does not satisfy the KKT conditions (to
## within specified tolerances)
##
## Call:
## fixedLassoInf(x = x2, y = y2, beta = coef(lassomodel2, s = best_lambda2/n2)[-1],
##     lambda = best_lambda2)
##
## Standard deviation of noise (specified or estimated) sigma = 0.908
##
## Testing results at lambda = 0.149, with alpha = 0.100
##
##  Var   Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
```

```
##    1 -0.052  -0.953   0.341    -0.139    0.129       0.049      0.050
##    2 -0.456  -4.970   0.000    -0.608   -0.304       0.049      0.049
##    6  0.156   3.075   0.002     0.070    0.240       0.049      0.050
##    9  0.178   3.120   0.002     0.082    0.296       0.050      0.050
##
## Note: coefficients shown are partial regression coefficients

n3 <- nobs(lassomodel2)
fixedLassoInf(x2, y2, beta = coef(lassomodel2, s = best_lambda2 / n2)[-1], lambda = best_lamb
## Warning in fixedLassoInf(x2, y2, beta = coef(lassomodel2, s =
## best_lambda2/n2)[-1], : Solution beta does not satisfy the KKT conditions (to
## within specified tolerances)
##
## Call:
## fixedLassoInf(x = x2, y = y2, beta = coef(lassomodel2, s = best_lambda2/n2)[-1],
##     lambda = best_lambda2)
##
## Standard deviation of noise (specified or estimated) sigma = 0.908
##
## Testing results at lambda = 0.149, with alpha = 0.100
##
##  Var   Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
##    1 -0.052  -0.953   0.341    -0.139    0.129       0.049      0.050
##    2 -0.456  -4.970   0.000    -0.608   -0.304       0.049      0.049
##    6  0.156   3.075   0.002     0.070    0.240       0.049      0.050
##    9  0.178   3.120   0.002     0.082    0.296       0.050      0.050
##
## Note: coefficients shown are partial regression coefficients
```

Lastly, we will take a look at one more final method for regularization and variable selection in regression: Adaptive Lasso. Adaptive LASSO is an "evolution" of the LASSO (it's not really built on the LASSO though, as the LASSO is itself built on elastic net procedures) that uses a weighted (by coefficient) L1 penalty rather than a simple L1 penalty. Unlike LASSO, Adaptive LASSO appears to have "oracle" properties when the "correct" value of lambda, which means that it has the following properties - it will identify the "correct" subset model (won't include noise variables) and has an optimal estimation rate, which essentially just means that coefficients are estimated as accurately as can be from the given data. **This is extremely exciting, particularly because we estimated our LASSO model using the best lambda for prediction, rather than for to have the "correct" model.**

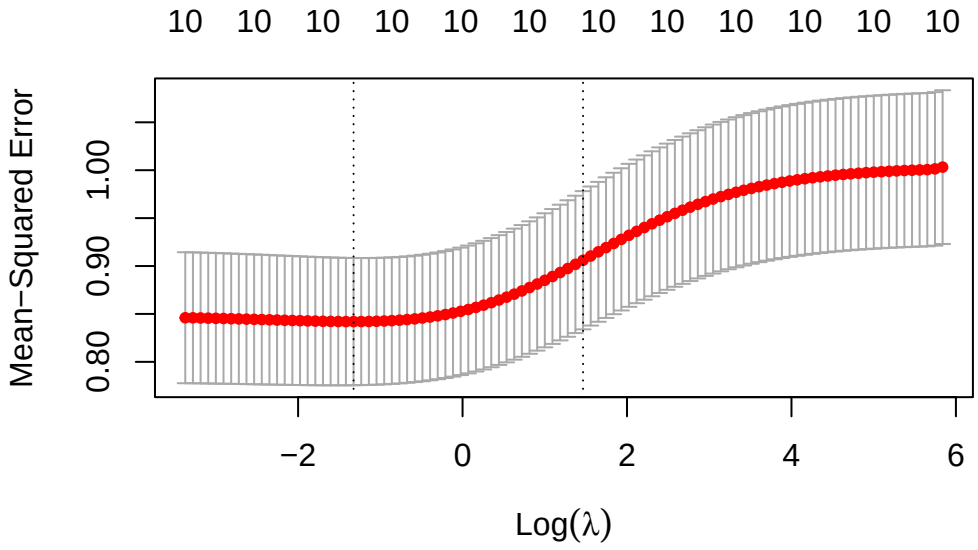
As you'll recall, LASSO and ridge regression introduce bias to counteract high variance, so adaptive LASSO can work around this by weighting penalties on coefficients.

Adaptive LASSO requires a two-step procedure, where coefficient estimates are estimated from a previously estimated model. The original author suggests using the OLS Beta estimates unless collinearity is a concern, in which case we will use ridge regression coefficients, which are more stable in the presence of multicollinearity. We'll do both of course.

To briefly summarize ridge vs. LASSO, ridge regression uses an L2 penalty which simply means that the penalty term squares the vector of Beta values whereas LASSO uses absolute deviation. Furthermore, ridge regression will retain all features in the model, some with very low coefficients whereas LASSO will reduce unimportant features to 0. Lastly, ridge regression will handle multicollinearity and outliers better than LASSO, but LASSO's reduce-coefficient-to-0 capabilities make it extremely attractive for feature selection, particularly in situations where you are not concerned about severe multicollinearity or severe P»N (where ridge is often better).

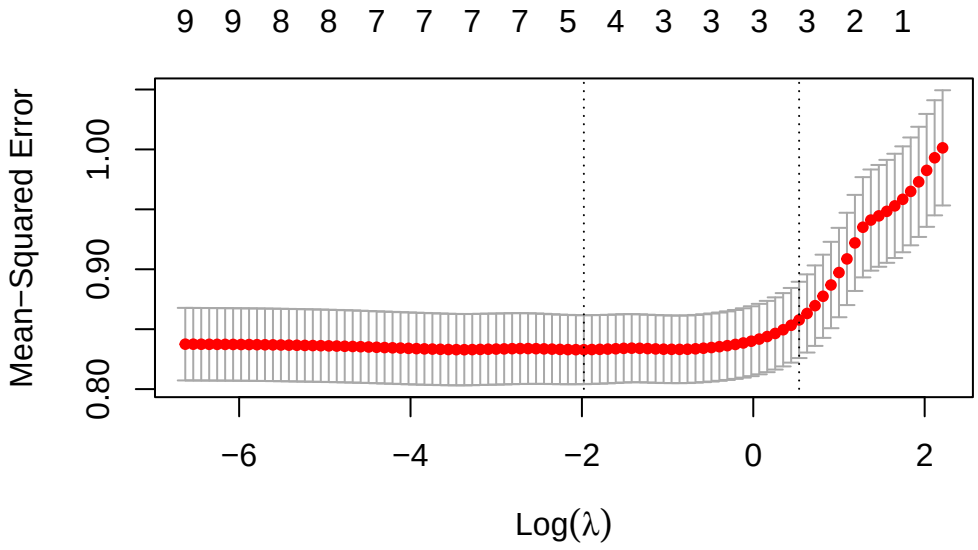
Let's work out the OLS and ridge coefficients. We'll start with ridge since we've already done LASSO, not to mention some of our predictors actually are relatively correlated (probably not enough that we'd actually want to start with ridge if this were a real analysis, but still).

```
set.seed(200)
ridge1 <- cv.glmnet(
  x = x2, y = y2,
  ## measure to optimize for cross validation
  type.measure = "mse",
  ## K = 10 is the default.
  nfold = 10,
  ## 'alpha = 1' is the lasso elastic net penalty, and 'alpha = 0' the ridge elastic net pen
  alpha = 0
)


## Penalty vs CV MSE plot
plot(ridge1)
```




```
## 1se and 1min
ridge1$lambda.min
## [1] 0.2656693
ridge1$lambda.1se
## [1] 4.329748

# coefficients
coef(ridge1, s = ridge1$lambda.min)
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)              0.253303209
## score_AMAS_total        -0.066119152
## sexf                    -0.370267580
## age                      0.001686684
## score_BFI_N             -0.015515118
## score_GAD                0.011340967
## score_PISA_ME            0.138284728
## score_STAI_state_short -0.050882981
## score_TAI_short          0.032004414
## score_SDQ_M              0.142881467
## score_SDQ_L              0.028708546
coef(ridge1, s = ridge1$lambda.1se)
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)              0.068920252
## score_AMAS_total        -0.037963817
```

```
## sexf                    -0.100744617
## age                      0.003497474
## score_BFI_N             -0.016511058
## score_GAD               -0.008537669
## score_PISA_ME            0.051511820
## score_STAI_state_short -0.020000712
## score_TAI_short         -0.003173425
## score_SDQ_M              0.046613828
## score_SDQ_L             -0.002454640
## Oops, gotta drop intercept estimate, we don't need that right now.
best_ridge_coef <- as.numeric(coef(ridge1, s = ridge1$lambda.min))[-1]

best_ridge_coef2 <- as.numeric(coef(ridge1, s = ridge1$lambda.1se))[-1]

## okay guess we'll try adaptive lasso.
set.seed(201)
alasso1 <- cv.glmnet(
  x = x2, y = y2,
  type.measure = "mse",
  nfold = 10,
  alpha = 1,
  # penalty factors can be different for each coefficient. This number is multiplied by lambd
  penalty.factor = 1 / abs(best_ridge_coef),
  ## prevalidated array is returned
  keep = TRUE
)
## Penalty vs CV MSE plot
plot(alasso1)
```




```
## Extract coefficients at the two lambda values we like
alasso1$lambda.min
## [1] 0.1384722
alasso1$lambda.1se
## [1] 1.70715

alassocoef1 <- coef(alasso1, s = alasso1$lambda.min)
alassocoef1
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)              0.316220469
## score_AMAS_total        -0.005398052
## sexf                    -0.462237287
## age                      .
## score_BFI_N              .
## score_GAD                .
## score_PISA_ME            0.149902422
## score_STAI_state_short -0.034444375
## score_TAI_short          .
## score_SDQ_M              0.193514871
## score_SDQ_L              .
alassocoef2 <- coef(alasso1, s = alasso1$lambda.1se)
alassocoef2
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)              0.29302903
```

```
## score_AMAS_total         .
## sexf                    -0.42833706
## age                      .
## score_BFI_N              .
## score_GAD                .
## score_PISA_ME            0.07424765
## score_STAI_state_short   .
## score_TAI_short          .
## score_SDQ_M              0.11913325
## score_SDQ_L              .
```

The less stringent lambda produced almost the same model as the regular LASSO, but without GAD. We are probably happy with that - we know GAD hasn't really helped our models in the past. Seems like that adaptive lasso created a somewhat more parsimonious model, but still with some possible noise variables (indicating we did not use the "optimal" level of lambda, which is understandable of course).

Seems that the more stringent lambda value actually may have selected a model that contains the crucial variables in terms of explaining the data. - WRITE MORE HERE ITS INCOMPLETEEEEEE This could be because we're using ridge regression coefficients, and it may not be the best true fit for our predictors given that our predictors are not extremely correlated.

Let's do OLS now.

```
lm1 <- lm(y2 ~ x2) # simple linear model
summary(lm1) # kind of interesting that SDQ_L keeps popping up as like, mildly relevant? But
##
## Call:
## lm(formula = y2 ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.61839 -0.59134 -0.02117  0.54692  2.48470
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.3262219  0.0756883   4.310 1.96e-05 ***
## x2score_AMAS_total       -0.0452919  0.0621963  -0.728  0.46682
## x2sexf                   -0.4768569  0.0939660  -5.075 5.46e-07 ***
## x2age                     0.0005689  0.0415375   0.014  0.98908
## x2score_BFI_N            -0.0076183  0.0538123  -0.142  0.88748
## x2score_GAD               0.0216458  0.0543524   0.398  0.69061
## x2score_PISA_ME           0.1478662  0.0527826   2.801  0.00528 **
```

```
## x2score_STAI_state_short -0.0687982  0.0476976  -1.442  0.14981
## x2score_TAI_short         0.0498519  0.0461368   1.081  0.28043
## x2score_SDQ_M             0.1918772  0.0589872   3.253  0.00122 **
## x2score_SDQ_L             0.0516186  0.0416661   1.239  0.21597
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9076 on 505 degrees of freedom
## Multiple R-squared:  0.1922, Adjusted R-squared:  0.1762
## F-statistic: 12.02 on 10 and 505 DF,  p-value: < 2.2e-16
```

```
ols_coef <- coef(lm1)[-1] # Exclude the intercept
best_ols_coef <- 1 / abs(ols_coef)

# We'll skip cross validation because it would just be bootstrap, which is written about else

# Fit lasso w ols weights
set.seed(300)
alasso_ols <- cv.glmnet(x2, y2,
  type.measure = "mse",
  nfold = 10,
  alpha = 1,
  # penalty factors can be different for each coefficient. This
  penalty.factor = best_ols_coef,
  ## prevalidated array is returned
  keep = TRUE
)



## Extract coefficients at the two lambda values we like
alasso_ols$lambda.min
## [1] 0.5531049
alasso_ols$lambda.1se
## [1] 8.213433

alassocoef3 <- coef(alasso_ols, s = alasso_ols$lambda.min)
alassocoef3
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                    s1
## (Intercept)             0.31872588
## score_AMAS_total        .
```

```
## sexf                    -0.46589959
## age                      .
## score_BFI_N              .
## score_GAD                .
## score_PISA_ME            0.14374736
## score_STAI_state_short -0.02882554
## score_TAI_short          .
## score_SDQ_M              0.19831690
## score_SDQ_L              .
alassocoef4 <- coef(alasso_ols, s = alasso_ols$lambda.1se)
alassocoef4
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                                  s1
## (Intercept)              0.28357814
## score_AMAS_total         .
## sexf                    -0.41452216
## age                      .
## score_BFI_N              .
## score_GAD                .
## score_PISA_ME            .
## score_STAI_state_short   .
## score_TAI_short          .
## score_SDQ_M              0.08742007
## score_SDQ_L              .
```

The more stringent lambda reported the same coefficients as the ridge adaptive lasso, which means including SDQ_M but not PISA_ME. The model with the less stringent lambda simply included PISA_ME as the third predictor. Like we mentioned way back at the beginning of the worksheet, PISA_ME and SDQ_M naturally correlate quite strongly. Thus, it's possible that the stringent adaptive LASSO selected SDQ_M and then saw little further value in including PISA_ME due to their correlation. The less stringent adaptive LASSO did seem to find unique value in including PISA_ME regardless of its correlation with SDQ_M.

YOU WANNA ADD LASSO-SPECIFIC PREDICTIONS INSTEAD OF PUTTING THEM IN A RF AND LETTING THEM GO WILD. COMPARE AND CONTRAST THE VARIABLES LASSO SELECTED (HINT: THEY'RE THE REAL VARIABLES BUT WORSE PREDICTION!) CHECK WHAT YOU DID ON CHATGPT

YOU ALSO WANTED TO COMPARE PREDICTIVE POWER OF ALASSO VS LASSO, IF POSSIBLE (AKA JUST USING PREDICT FUNCTION ON EACH ONE BRO)

COMPARE AND CONTRAST THE LAMBDA VALUES, TOO.

MAKE SURE EVERY RANDOM OPERATION (CV.GLMNET, LITERALLY ANYTHING ELSE) HAS SET SEED RIGHT BEFORE IT!!!!!!!

References

Allaire, J.J., Teague, C., Scheidegger, C., Xie, Y., & Dervieux C. (2024). Quarto version 1.4. https://github.com/quarto-dev/quarto-cli

Brownlee, J. (2017, December 12th). *Project-oriented Workflow.* Tidyverse.org. https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/ .

Bryan, J. (2020, July 30). *Tune machine learning algorithms in R (random forest case study).* MachineLearningMastery.com.https://www.tidyverse.org/blog/2017/12/workflow-vs-script/

Cipora, K., Lunardon, M., Masson, N., Georges, C., Nuerk, H.-C., & Artemenko, C. (2024). The AMATUS Dataset: Arithmetic Performance, Mathematics Anxiety and Attitudes in Primary School Teachers and University Students. *Journal of Open Psychology Data*, *12*. https://doi.org/10.5334/jopd.115

Chatterjee, T., & Chowdhury, R. (2017). Chapter 11 - Improved Sparse Approximation Models for Stochastic Computations. In *Handbook of Neural Computation* (pp. 201–223). essay, Academic Press.

Friedman J., Tibshirani R., Hastie T. (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, *33*(1), 1–22. doi:10.18637/jss.v033.i01.

Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2015). VSURF: An R package for variable selection using random forests. *The R Journal*, *7*(2), 19–33. https://doi.org/10.32614/rj-2015-018

Ilma, H. (2019, December 18). Ridge and LASSO Regression. https://algotech.netlify.app/blog/ridge-lasso/

Kuhn, M. (2008). "Building Predictive Models in R Using the caret Package." *Journal of Statistical Software*, *28*(5), 1–26. doi:10.18637/jss.v028.i05, https://www.jstatsoft.org/index.php/jss/article/view/v028i05.

Nahhas, R.W. (2024). *Introduction to Regression Methods for Public Health Using R.* CRC Press. https://bookdown.org/rwnahhas/RMPH/

Speiser, J. L., Miller, M. E., Tooze, J., & Ip, E. (2019). A Comparison of Random Forest Variable Selection Methods for Classification Prediction Modeling. *Expert systems with applications*, *134*, 93–101. https://doi.org/10.1016/j.eswa.2019.05.028

Tay, J.K. (2021, February 19). *The box-cox and Yeo-Johnson transformations for continuous variables.* Statistical Odds and Ends. https://statisticaloddsandends.wordpress.com/2021/02/19/the-box-cox-and-yeo-johnson-transformations-for-continuous-variables/

Tay J.K, Narasimhan B., Hastie T. (2023). "Elastic Net Regularization Paths for All Generalized Linear Models." *Journal of Statistical Software*, *106*(1), 1–31. doi:10.18637/jss.v106.i01.

Tibshirani, R.J., Taylor, J., Lockhart, R., & Tibshirani, R. (2016). Exact Post-Selection Inference for Sequential Regression Procedures. *Journal of the American Statistical Association*, *111*(514), 600–620. https://doi.org/10.1080/01621459.2015.1108848

Yoshida, K. (2017). Adaptive Lasso. *Rpubs by RStudio. https://rpubs.com/kaz_yos/alasso*

Zou, H. (2006). The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, *101*(476), 1418–1429. https://doi.org/10.1198/016214506000000735

And a host of Stack Exchange questions as references:

https://stats.stackexchange.com/questions/174976/why-does-the-intercept-column-in-model-matrix-replace-the-first-factor

https://datascience.stackexchange.com/questions/39932/feature-scaling-both-training-and-test-data

https://stats.stackexchange.com/questions/350484/why-is-r-squared-not-a-good-measure-for-regressions-fit-using-lasso

https://stats.stackexchange.com/questions/522265/residual-analysis-for-the-lasso-estimator

https://stats.stackexchange.com/questions/6502/lasso-assumptions

https://stats.stackexchange.com/questions/291409/inference-after-using-lasso-for-variable-selection

https://math.stackexchange.com/questions/2162932/big-picture-behind-how-to-use-kkt-conditions-for-constrained-optimization

https://stats.stackexchange.com/questions/156098/cross-validating-lasso-regression-in-r

https://stats.stackexchange.com/questions/403310/per-cent-increase-in-mse-incmse-random-forests-importance-measure-why-is-mea

https://stackoverflow.com/questions/71900366/randomforest-error-this-function-only-works-for-objects-of-class-rfsrc-grow

https://stats.stackexchange.com/questions/485471/when-in-adaptive-lasso-process-does-it-make-sense-to-constrain-control-variable

https://stats.stackexchange.com/questions/487412/is-this-the-correct-way-to-run-an-adaptive-lasso