COMP 4441 Final Project: Predicting Student Math Success From Self-Reported Attitudes

Luke Sonnanburg

Project Overview

The EQAO (Ontario's chief agency governing public schools) issues a standardized mathematics test to all 9th grade students in Ontario. In the school year of 2013-2014, students also responded to a series of statements that reflected their opinions of mathematics and their relationship to it. While individual student responses and scores are not publicly available, aggregate responses and scores are published by the EQAO on a per-school basis.

The research question I attempt to answer here is: Do students' self-reported attitudes towards math (measured by their aggregate responses to the below statements) accurately predict their scores on EQAO's standardized 9th grade math test?

Students were asked to respond to the following statements * Q1: I like mathematics * Q2: I am good at mathematics * Q3: I am able to answer difficult mathematics questions * Q4: Mathematics is one of my favourite subjects * Q5: I understand most of the mathematics I am taught * Q6: Mathematics is an easy subject * Q7: I do my best in mathematics class * Q8: The mathematics I learn now is useful for everyday life * Q9: The mathematics I learn now helps me do work in other subjects * Q10: I need to do well in mathematics to study what I want later * Q11: I need to keep taking mathematics for the kind of job I want after I leave school

Initial Data View

The variables of interest are a school's aggregate response to a statement, and average scores.

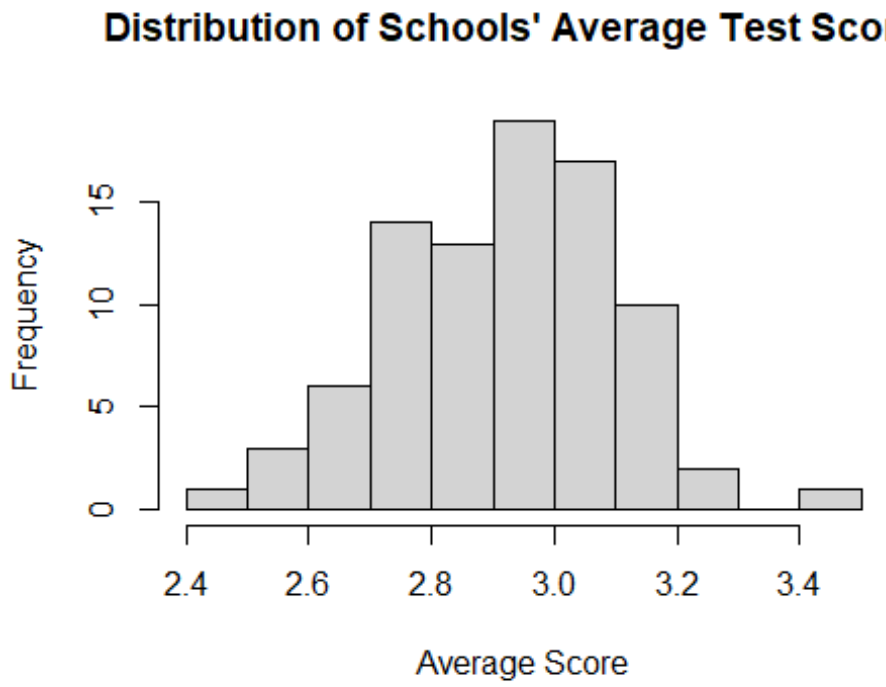
The variables Q1...Q11 correspond to the percentage of students at a school who responded that they either agree or strongly agree to the numbered statement.

Individual students received a score on their standardized test from 1 to 4, with 4 being the best possible score. Average scores are calculated from $\frac{\sum_{s=1}^{4} s n_s}{\sum_{s=s}^{4} n_s}$, where s is a score and n_s is the number of students at a school who received that score.

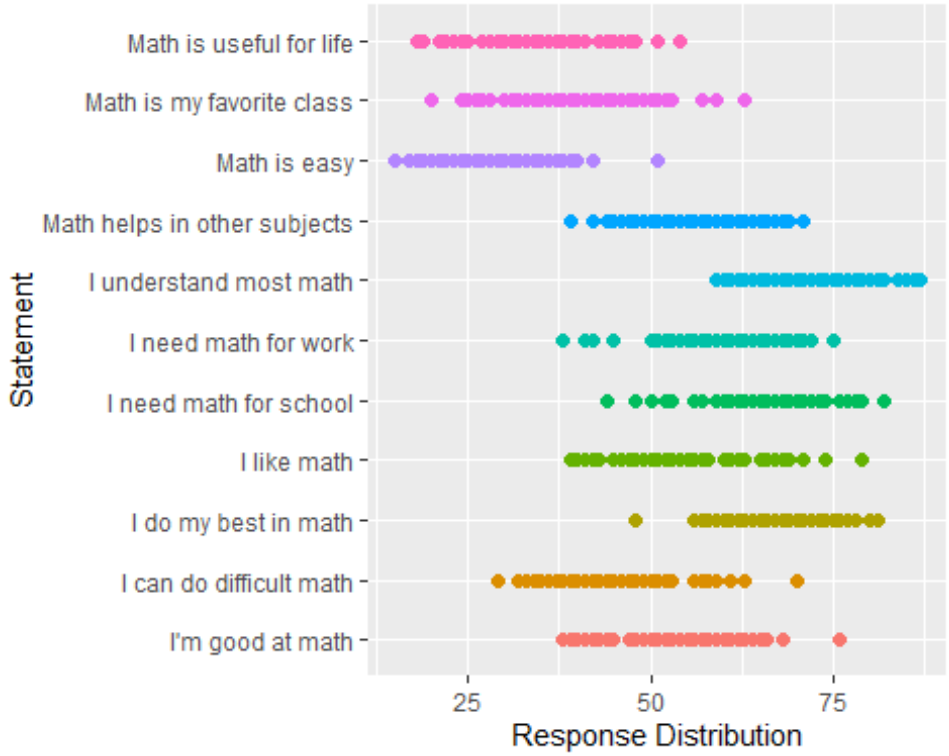
```
main_model <- response_dat
main_model$average_score <- with(school_dat,
                                 (Level.1.... + 2*Level.2.... + 3*Level.3....
+ 4*Level.4....)/
                                  (Level.1.... + Level.2.... + Level.3.... +
Level.4....) )
```

```
columns <-
c("school_id","q1","q2","q3","q4","q5","q6","q7","q8","q9","q10","q11","avera
ge_score")
colnames(main_model) <- columns
head(main_model)
```

```
##   school_id q1 q2 q3 q4 q5 q6 q7 q8 q9 q10 q11 average_score
## 1    890723 43 47 40 27 69 32 59 32 61  64  59      3.090909
## 2    890880 60 48 38 46 72 40 66 48 58  63  57      2.909091
## 3    892203 60 52 37 38 75 23 65 34 57  62  58      2.762887
## 4    895016 46 47 38 39 59 19 68 28 48  53  54      2.659794
## 5    895920 54 49 39 36 70 22 72 35 53  64  61      2.676768
## 6    896055 62 50 48 47 77 27 72 40 60  70  61      2.890000
```




```
## [1] "cols"  "value"
```



Exploratory Analysis

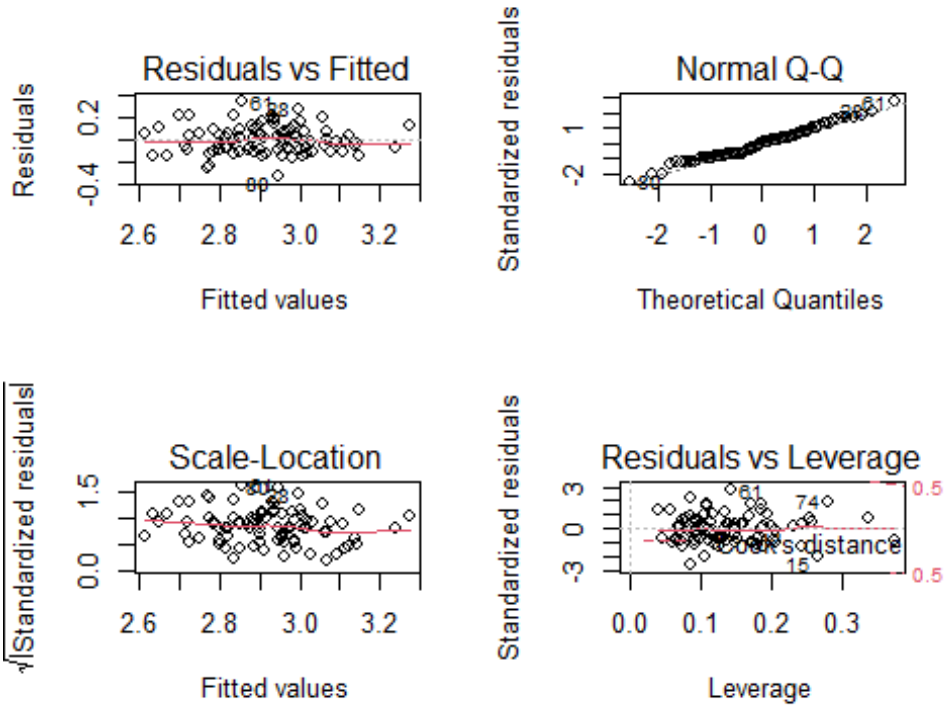
While my intentions are to test a number of statistical methods on this data, they all include some form of linear regression. Before doing that, there are a number of assumptions that must be satisfied for linear regression to be valid. To check that these are satisfied, I first examine a least squares best fit line assuming we use every variable.

```
##
## Call:
## lm(formula = average_score ~ ., data = main_model[2:13])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31787 -0.09233  0.00498  0.07845  0.33532
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.835992   0.236649   7.758 3.71e-11 ***
## q1           0.001735   0.004760   0.365  0.71646
## q2           0.001635   0.004680   0.349  0.72781
## q3           0.011600   0.003919   2.960  0.00413 **
## q4          -0.004719   0.004434  -1.064  0.29060
## q5           0.011596   0.004512   2.570  0.01218 *
## q6          -0.004817   0.003243  -1.486  0.14164
## q7          -0.008722   0.003386  -2.576  0.01199 *
## q8          -0.004599   0.003236  -1.421  0.15947
```

```
## q9           0.005872   0.002894   2.029  0.04603 *
## q10          0.006632   0.005133   1.292  0.20032
## q11         -0.002593   0.004798  -0.540  0.59052
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1329 on 74 degrees of freedom
## Multiple R-squared:  0.5403, Adjusted R-squared:  0.4719
## F-statistic: 7.906 on 11 and 74 DF,  p-value: 6.779e-09
```

From these p-values we can already identify that only a few statements are individually likely to be statistically significant. Particularly Q1, Q2, and Q11 are very unlikely to be significant. However, they will be considered for factor analysis later and are therefore kept for assumption checking.

Graphing to visually check linear regression assumptions are met:

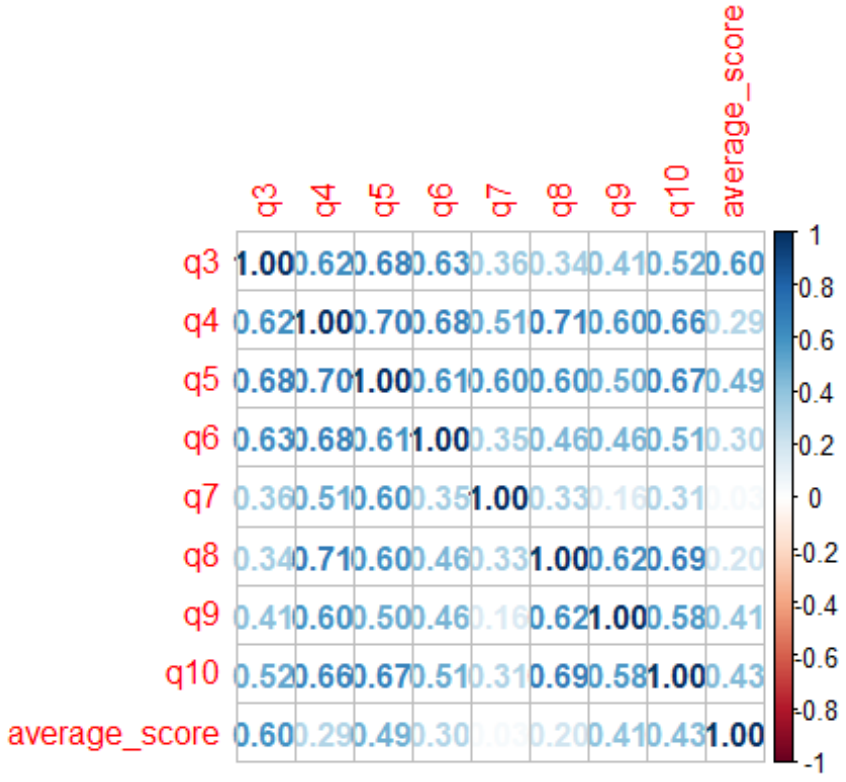


Linear regression's need for a linear relationship between explanatory and output variables is met by our almost flat Residuals vs Fitted graph. The Normal Q-Q plot shows that, while there are a few outliers at the ends, residuals are mostly normally distributed. Our scale-location graph has a very slight downward slope, demonstrating some very minor heteroscedasticity. This was actually made worse by attempts to log transform and flatten the data, so it will be left as is. All data falls well within the boundaries of Cook's distance; we have no significant outliers to consider pruning.

All assumptions of linear regression have been met, so it is safe to proceed. However, before directly applying linear regression models to the data, I will first remove Q1, Q10, and Q11 from consideration based on their t-test results in the basic model summary.

```
##   school_id q3 q4 q5 q6 q7 q8 q9 q10 average_score
## 1    890723 40 27 69 32 59 32 61  64      3.090909
## 2    890880 38 46 72 40 66 48 58  63      2.909091
## 3    892203 37 38 75 23 65 34 57  62      2.762887
## 4    895016 38 39 59 19 68 28 48  53      2.659794
## 5    895920 39 36 70 22 72 35 53  64      2.676768
## 6    896055 48 47 77 27 72 40 60  70      2.890000
```

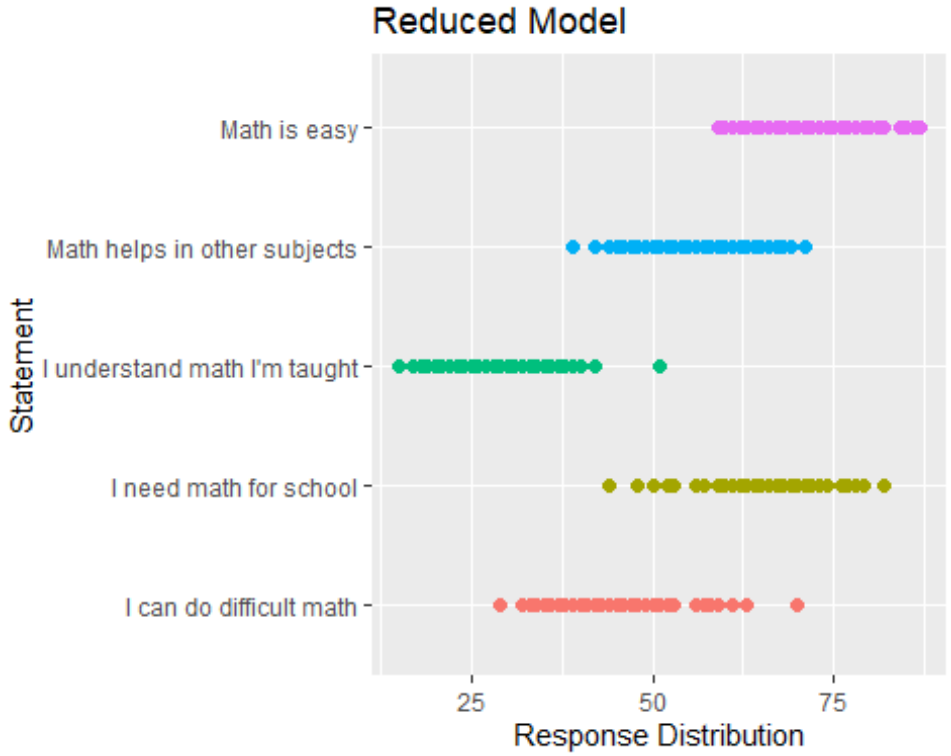
While low collinearity is not necessary assumption of linear regression, it can cause problems for the calculations. Below we can see the correlations between every remaining predictive variable.

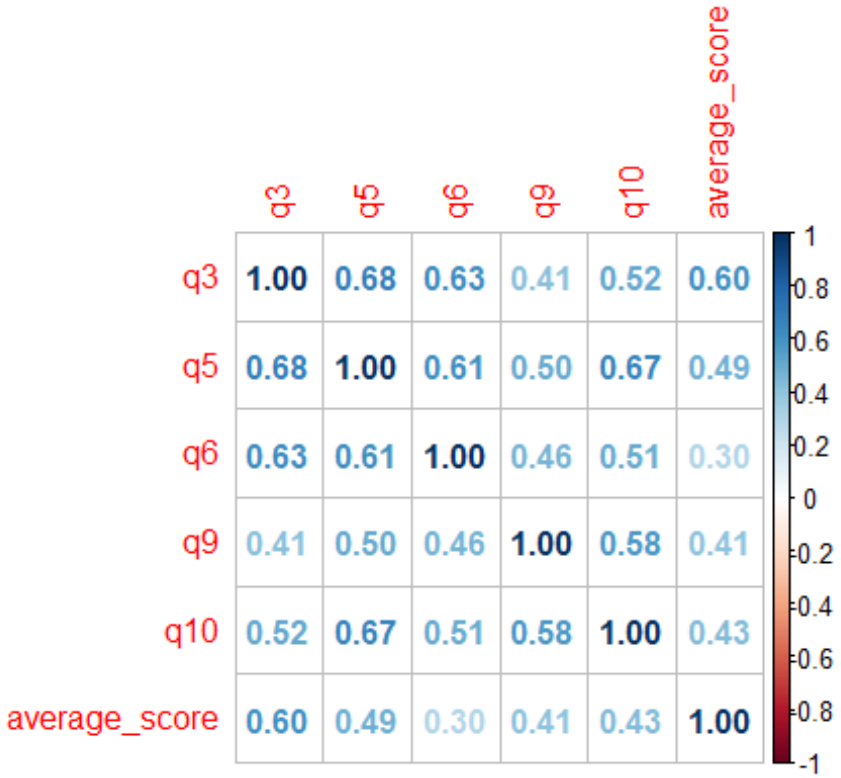


Overall high correlation between variables suggests this data may work well with factor analysis, but high colinearity may be a problem for direct linear regression. To continue my naive approach of manually removing variables, I will also drop q4, q7, and q8. All three have very high correlation with q5, but weak correlation with the output variable.

```
##   school_id q3 q5 q6 q9 q10 average_score
## 1    890723 40 69 32 61  64      3.090909
## 2    890880 38 72 40 58  63      2.909091
## 3    892203 37 75 23 57  62      2.762887
## 4    895016 38 59 19 48  53      2.659794
```

```
## 5    895920 39 70 22 53  64      2.676768
## 6    896055 48 77 27 60  70      2.890000

## [1] "cols"  "value"
```





I then split the data into training/testing sets at an 80/20 ratio, applied several methods of linear regression, and measured their accuracy in predicting scores from the 20% of data in the testing set. The methods were ridge, LASSO, and elastic net regressions.

All three methods operate on the basis of performing linear regression, but shrinking the coefficients of explanatory variables that have a very small contribution. This shrinking of coefficients is performed with a penalizing term. Ridge regression bases this on the L2-norm based on the square of the magnitude of coefficients, lasso regression bases it on the magnitude of the coefficients themselves, and elastic net regression combines them based on an additional hyperparameter that tilts the model closer to either ridge or LASSO regression.

All three were implemented using R's glmnet library for penalized regression. All three are fine-tuned with a λ parameter, for which I used the call:

cv <- cv.glmnet(x, y, alpha = 0) cv$lambda.min

to automatically determine the lambda that minimizes errors.

Elastic Net regression also uses a hyperparameter α to indicate to what extent it should blend ridge or LASSO regression methods. (The same function is used for all three; ridge or LASSO are simply selected by setting $\alpha = 0$ or $\alpha = 1$).

model <- train(medv ~., data = train.data, method = "glmnet", trControl = trainControl("cv", number = 10), tuneLength = 10

Best tuning parameter

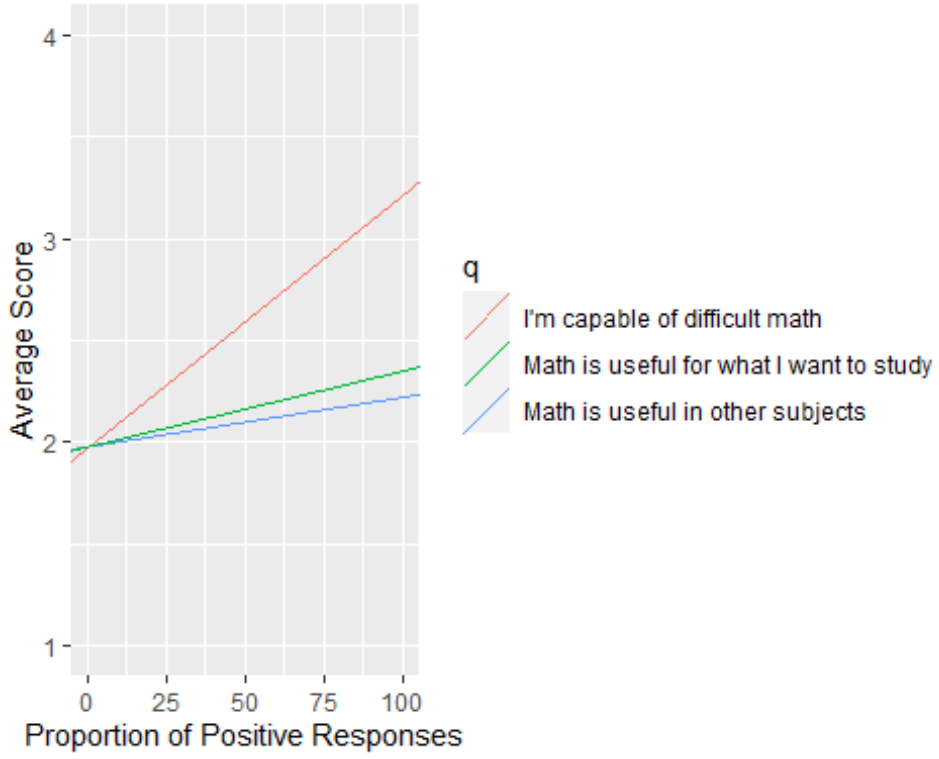
model$bestTune

The final results of the models are as follows:

```
##                             RMSE   Rsquare number of variables
## Ridge Regression       0.1459139 0.3556914                   5
## Lasso Regression       0.1510877 0.3428642                   3
## Elastic Net Regression 0.1500432 0.3440466                   4
```

The ridge regression model is the most accurate for making predictions, but the results for LASSO regression is nearly as accurate with only three variables.

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 1.975639558
## q3          0.012432363
## q5          .
## q6          .
## q9          0.002429859
## q10         0.003729587
```



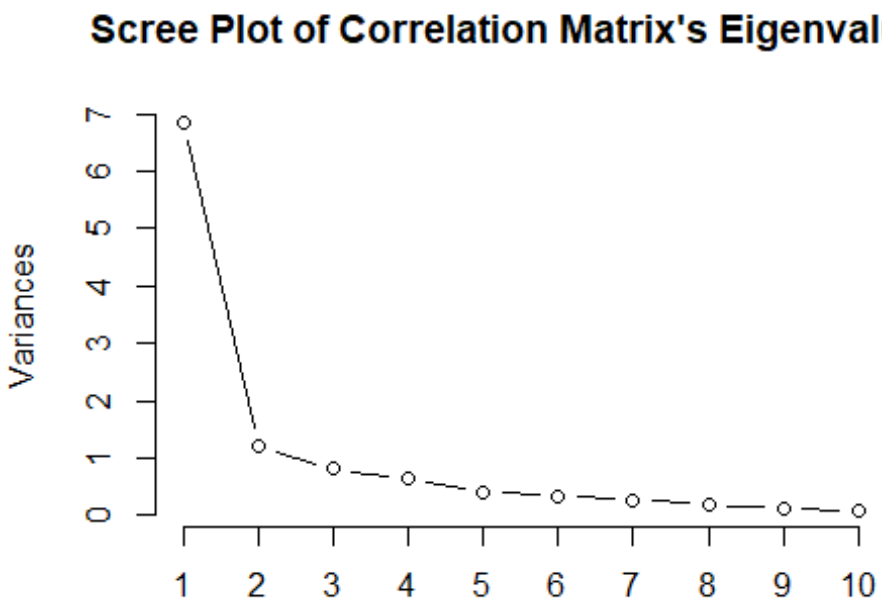
Factor Analysis

I believe that the large number of and correlation between most of these explanatory variables makes this a good case for using factor analysis, in order to reduce the model's dimensionality without losing too much information. For confirmation, I apply the Kaiser, Meyer, Olkin measure of sampling, measuring how much variance is common between variables:

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = mat)
## Overall MSA =  0.88
## MSA for each item =
##   q1   q2   q3   q4   q5   q6   q7   q8   q9  q10  q11
## 0.90 0.85 0.86 0.89 0.91 0.92 0.78 0.92 0.94 0.84 0.84
```

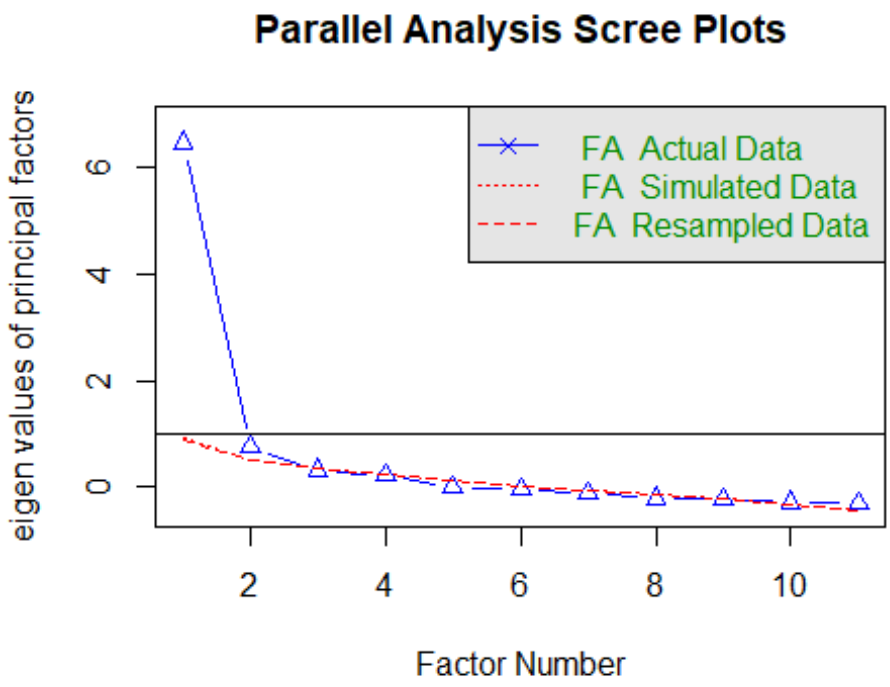
According to Kaiser, who the test is partly named for, an overall MSA of .87, the implementation of factor analysis for this data is "meritorious", so I proceed with this plan. Standards vary, but all sources seem to agree that an MSA greater than .8 is sufficient for factor analysis.

Factor analysis will group weights of explanatory variables together according to their correlation with each other and the outcome variable, but we need to first determine how many factors are desirable.




```
## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.
```

```
## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs =
np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
: An
## ultra-Heywood case was detected. Examine the results carefully
```




```
## Parallel analysis suggests that the number of factors =  2  and the number
of components =  NA
```

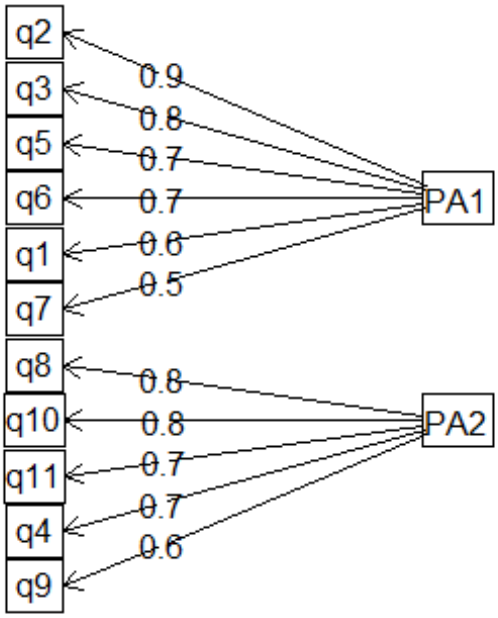
Judging from the graph, only the eigenvalues of the first two principal factors are above those of the simulated factor analysis data. This suggests that only two factors will be relevant to a good model. Since a third is borderline, I will explore the possibility of using a third factor with low expectations.

```
## [1] "Factor loadings with two factors:"

##
## Loadings:
##     PA1   PA2
## q1  0.642 0.631
## q2  0.945 0.209
## q3  0.794 0.269
## q4  0.589 0.656
## q5  0.661 0.534
## q6  0.657 0.385
## q7  0.487 0.245
## q8  0.206 0.839
## q9  0.253 0.644
## q10 0.341 0.798
## q11 0.392 0.728
##
##                  PA1   PA2
## SS loadings    3.764 3.723
```

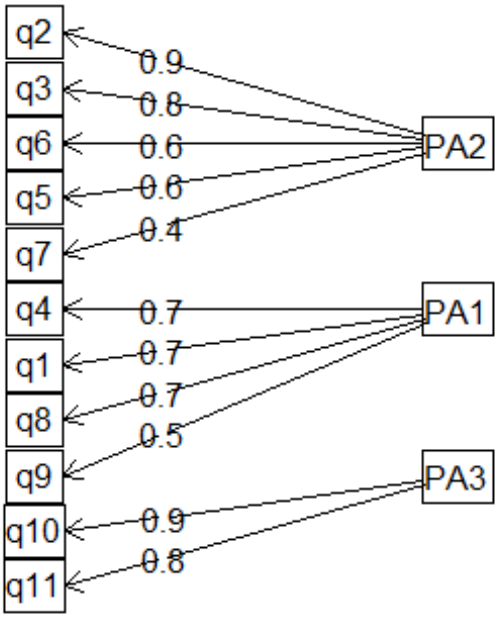
```
## Proportion Var 0.342 0.338
## Cumulative Var 0.342 0.681

## [1] "Factor loadings with three factors:"

##
## Loadings:
##     PA2   PA1   PA3
## q1  0.565 0.706 0.292
## q2  0.948 0.154 0.229
## q3  0.783 0.186 0.272
## q4  0.505 0.733 0.302
## q5  0.612 0.429 0.395
## q6  0.617 0.358 0.261
## q7  0.448 0.319
## q8  0.130 0.678 0.523
## q9  0.201 0.503 0.422
## q10 0.287 0.323 0.870
## q11 0.357 0.282 0.796
##
##                  PA2   PA1   PA3
## SS loadings    3.310 2.404 2.381
## Proportion Var 0.301 0.219 0.216
## Cumulative Var 0.301 0.520 0.736
```

Using factor analysis, we can reduce the dimensions of the explanatory variables from 11 to 2, keeping 67.4% of the variability, or to 3 keeping 73.6%. Varimax rotation was chosen to more evenly divide the number of variables with loadings between factors; without rotation nearly all variables had loadings on the first factor.

ıctor Analysis With Varimax Rotation and Two Factoı



ctor Analysis With Varimax Rotation and Three Factc




```
#head(fanone2$scores)
regdata <- data.frame(fanone2$scores,main_model$average_score)
fanone2model = lm(main_model.average_score~., regdata)
summary(fanone2model)
```

```
##
## Call:
## lm(formula = main_model.average_score ~ ., data = regdata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36802 -0.11160 -0.00979  0.11969  0.41375
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.91838    0.01742 167.555  < 2e-16 ***
## PA1          0.07958    0.01800   4.420 2.96e-05 ***
## PA2          0.04282    0.01859   2.304   0.0237 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1615 on 83 degrees of freedom
## Multiple R-squared:  0.2387, Adjusted R-squared:  0.2204
## F-statistic: 13.01 on 2 and 83 DF,  p-value: 1.213e-05
```

```
regdata2 <- data.frame(fanone3$scores,main_model$average_score)
fanone3model = lm(main_model.average_score~., regdata2)
summary(fanone3model)
```

```
##
## Call:
## lm(formula = main_model.average_score ~ ., data = regdata2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37322 -0.09040 -0.00134  0.11051  0.37963
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.918375   0.016897 172.713  < 2e-16 ***
## PA2         0.078939   0.017179   4.595 1.55e-05 ***
## PA1         0.002377   0.018147   0.131 0.896111
## PA3         0.062302   0.017646   3.531 0.000683 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1567 on 82 degrees of freedom
## Multiple R-squared:  0.2921, Adjusted R-squared:  0.2663
## F-statistic: 11.28 on 3 and 82 DF,  p-value: 2.853e-06
```

```
regdata2 <- data.frame(fanone3$scores,main_model$average_score)
fanone3model = lm(main_model.average_score~., regdata2)
summary(fanone3model)
```

```
##
## Call:
```

```
## lm(formula = main_model.average_score ~ ., data = regdata2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37322 -0.09040 -0.00134  0.11051  0.37963
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.918375   0.016897 172.713  < 2e-16 ***
## PA2         0.078939   0.017179   4.595 1.55e-05 ***
## PA1         0.002377   0.018147   0.131 0.896111
## PA3         0.062302   0.017646   3.531 0.000683 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1567 on 82 degrees of freedom
## Multiple R-squared:  0.2921, Adjusted R-squared:  0.2663
## F-statistic: 11.28 on 3 and 82 DF,  p-value: 2.853e-06
```

Model Selection

After five attempts, the overall best R-squared score came from the Elastic Net Regression model, with an R-squared value of 0.3440466 and seven retained explanatory variables. The best-fitting factor analysis model is the one with three factors. Those factors can be summarized as believing one's self is good at math, liking math, or acknowledging that math will be useful later in life. In both cases, the most influential feature is belief that the student is good at math.

Declaring one's self good at math predicting good math results is predictable, and likely the least interesting result. Still, it may suggest the potential of boosting students' self-confidence in mathematical mastery as a means of improving learning and test performance.