DATA 621 - HW5

Andrew Bowen, Glen Davis, Shoshana Farber, Joshua Forster, Charles Ugiagbe

2023-11-27

Homework 5 - Count Regression

Data Exploration

The dataset to be used in this analysis involves sales of over 12,000 different types of commercially available wine. There are 12,795 records across the training set with a response variable `TARGET` that indicates the number of sample cases purchased by wine distribution companies. It is generally more appropriate to use poisson or negative binomial regression methods to predict a discrete dependent variable, and different variations will be explored later in the analysis.

Below is a short description of all the variables of interest in the data set, including the response variable:

VARIABLE NAME	DEFINITION
`INDEX`	Identification Variable
`TARGET`	Number of Cases Purchased
`AcidIndex`	Proprietary method of testing total acidity of wine by using a weighted average
`Alcohol`	Alcohol Content
`Chlorides`	Chloride content of wine
`CitricAcid`	Citric Acid Content
`Density`	Density of Wine
`FixedAcidity`	Fixed Acidity of Wine
`FreeSulfurDioxide`	Sulfur Dioxide Content of Wine
`LabelAppeal`	Marketing Score indicating the appeal of label design for consumers. High numbers suggest customers like the label design. Negative numbers suggest customers don't like the design.
`ResidualSugar`	Residual Sugar of Wine
`STARS`	Win rating by a team of experts
`Sulphates`	Sulfate content of Wine
`TotalSulfurDioxide`	Total Sulfur Dioxide of Wine
`VolatileAcidity`	Volatile Acide content of Wine
`pH`	pH Level of Wine

Let's review a summary of the dataframe:

```
##      INDEX           TARGET       FixedAcidity     VolatileAcidity
##  Min.   :    1   Min.   :0.000   Min.   :-18.100   Min.   :-2.7900
##  1st Qu.: 4038   1st Qu.:2.000   1st Qu.:  5.200   1st Qu.: 0.1300
##  Median : 8110   Median :3.000   Median :  6.900   Median : 0.2800
```

```
##  Mean   : 8070   Mean   :3.029   Mean   :  7.076   Mean   : 0.3241
##  3rd Qu.:12106   3rd Qu.:4.000   3rd Qu.:  9.500   3rd Qu.: 0.6400
##  Max.   :16129   Max.   :8.000   Max.   : 34.400   Max.   : 3.6800
##
##    CitricAcid      ResidualSugar        Chlorides       FreeSulfurDioxide
##  Min.   :-3.2400   Min.   :-127.800   Min.   :-1.1710   Min.   :-555.00
##  1st Qu.: 0.0300   1st Qu.:  -2.000   1st Qu.:-0.0310   1st Qu.:   0.00
##  Median : 0.3100   Median :   3.900   Median : 0.0460   Median :  30.00
##  Mean   : 0.3084   Mean   :   5.419   Mean   : 0.0548   Mean   :  30.85
##  3rd Qu.: 0.5800   3rd Qu.:  15.900   3rd Qu.: 0.1530   3rd Qu.:  70.00
##  Max.   : 3.8600   Max.   : 141.150   Max.   : 1.3510   Max.   : 623.00
##                    NA's   :616        NA's   :638       NA's   :647
##  TotalSulfurDioxide    Density             pH          Sulphates
##  Min.   :-823.0     Min.   :0.8881   Min.   :0.480   Min.   :-3.1300
##  1st Qu.:  27.0     1st Qu.:0.9877   1st Qu.:2.960   1st Qu.: 0.2800
##  Median : 123.0     Median :0.9945   Median :3.200   Median : 0.5000
##  Mean   : 120.7     Mean   :0.9942   Mean   :3.208   Mean   : 0.5271
##  3rd Qu.: 208.0     3rd Qu.:1.0005   3rd Qu.:3.470   3rd Qu.: 0.8600
##  Max.   :1057.0     Max.   :1.0992   Max.   :6.130   Max.   : 4.2400
##  NA's   :682                         NA's   :395     NA's   :1210
##     Alcohol       LabelAppeal          AcidIndex          STARS
##  Min.   :-4.70   Min.   :-2.000000   Min.   : 4.000   Min.   :1.000
##  1st Qu.: 9.00   1st Qu.:-1.000000   1st Qu.: 7.000   1st Qu.:1.000
##  Median :10.40   Median : 0.000000   Median : 8.000   Median :2.000
##  Mean   :10.49   Mean   :-0.009066   Mean   : 7.773   Mean   :2.042
##  3rd Qu.:12.40   3rd Qu.: 1.000000   3rd Qu.: 8.000   3rd Qu.:3.000
##  Max.   :26.50   Max.   : 2.000000   Max.   :17.000   Max.   :4.000
##  NA's   :653                                          NA's   :3359
```

All of the variables are numeric, including `STARS`, although that predictor could be treated as a factor given that its values correspond to a rating scale. We will coerce it. It also has by far the most `NA` values, and we will add a level representing missingness to the factor. Looking at the means of all of the variables, it appears `CitricAcid` and `FreeSulfurDioxide` have much larger average values than the other predictor variables, which might impact our model coefficients later. `ResidualSugar` appears to have some large outlier values given its interquartile range of -2 to 15.9. It is not immediately obvious what a negative value would represent for this predictor, and this is not the only predictor for which there are negative values with unclear meaning.

We take a closer look at the number of observations that take negative values, broken down by predictor, and identify whether we believe these observations are valid data points or would distort the model. The predictor `LabelAppeal` is excluded since its negative values are interpretable according to its definition: negative numbers for this predictor "suggest customers don't like the design."

Predictor	Negative Values Count	Validity
FixedAcidity	1621	Invalid
VolatileAcidity	2827	Invalid
CitricAcid	2966	Invalid
ResidualSugar	3136	Invalid
Chlorides	3197	Invalid
FreeSulfurDioxide	3036	Invalid
TotalSulfurDioxide	2504	Invalid
Sulphates	2361	Invalid
Alcohol	118	Invalid

None of the above predictors measure quantities for which a negative value would be reasonable in this context, so we consider these predictors' negative observations invalid. Given the proclivity of negative values across most of these predictors, it is likely there were widespread recording or measurement errors. It is worth noting that `Alcohol` contains far fewer negative values than the other predictors affected by this issue, but it's unclear why that might be.

The evaluation data contains negative observations for all the same predictors as well:

Predictor	Negative Values Count	Validity
FixedAcidity	439	Invalid
VolatileAcidity	788	Invalid
CitricAcid	804	Invalid
ResidualSugar	828	Invalid
Chlorides	776	Invalid
FreeSulfurDioxide	774	Invalid
TotalSulfurDioxide	639	Invalid
Sulphates	594	Invalid
Alcohol	25	Invalid

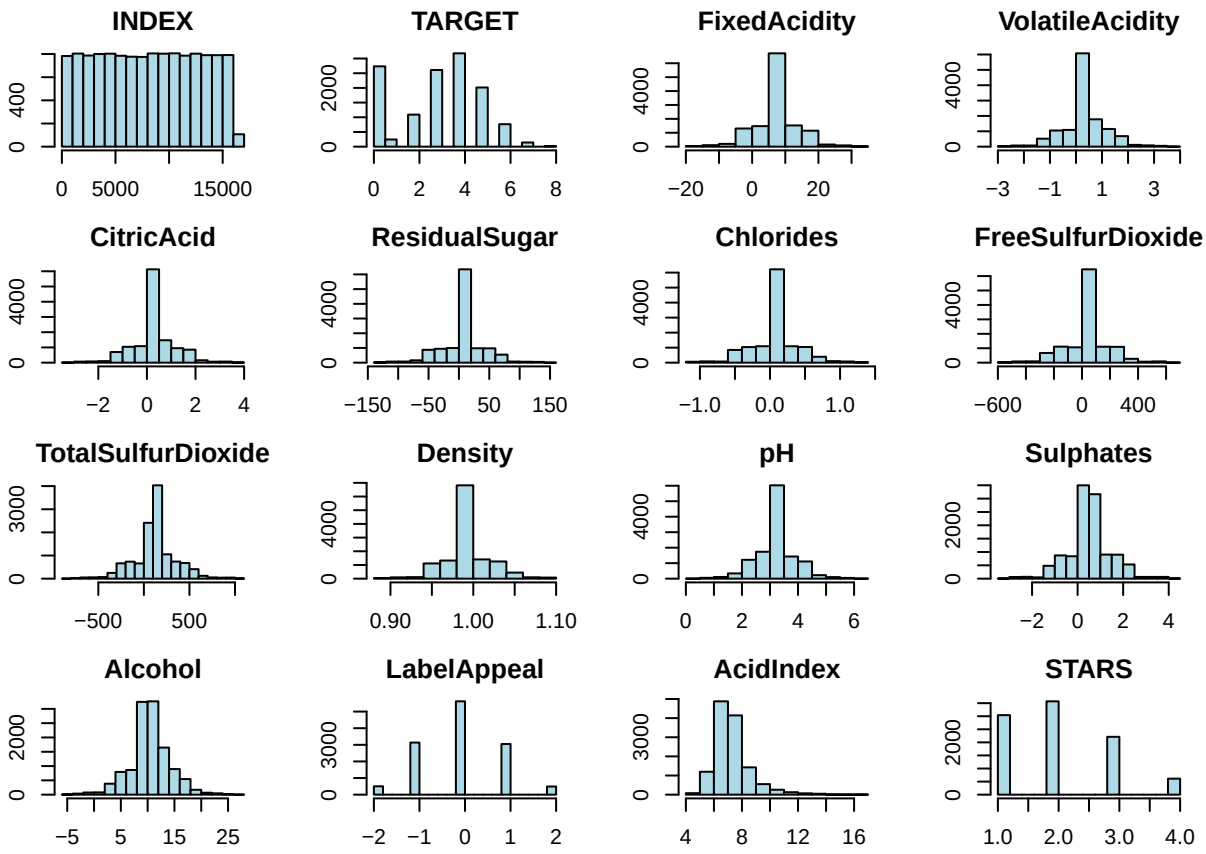
This leads us to the question of how to handle these occurrences appropriately. Although changing the sign for the negative observations might be the simplest way to fix this issue, it could be inappropriate due to the large number of observations this would affect, and it might not actually result in a more accurate representation of the data. Alternatively, excluding that number of observations from the training data might distort the model and make it less likely to extrapolate well to the evaluation data. So could excluding all these predictors.

So we have decided to try three methods: A) one in which we exclude predictors with invalid negative observations from modeling and B) one in which we change the signs of the negative observations. In both of these cases, we will use only complete cases without any imputations. In the latter only, we will make some transformations. Before we make the drastic changes to the data required by Method B, we will continue exploring the data in its original state. lastly in the third method C) we will only use columns without negative values, but will impute missing data points for these predictors.

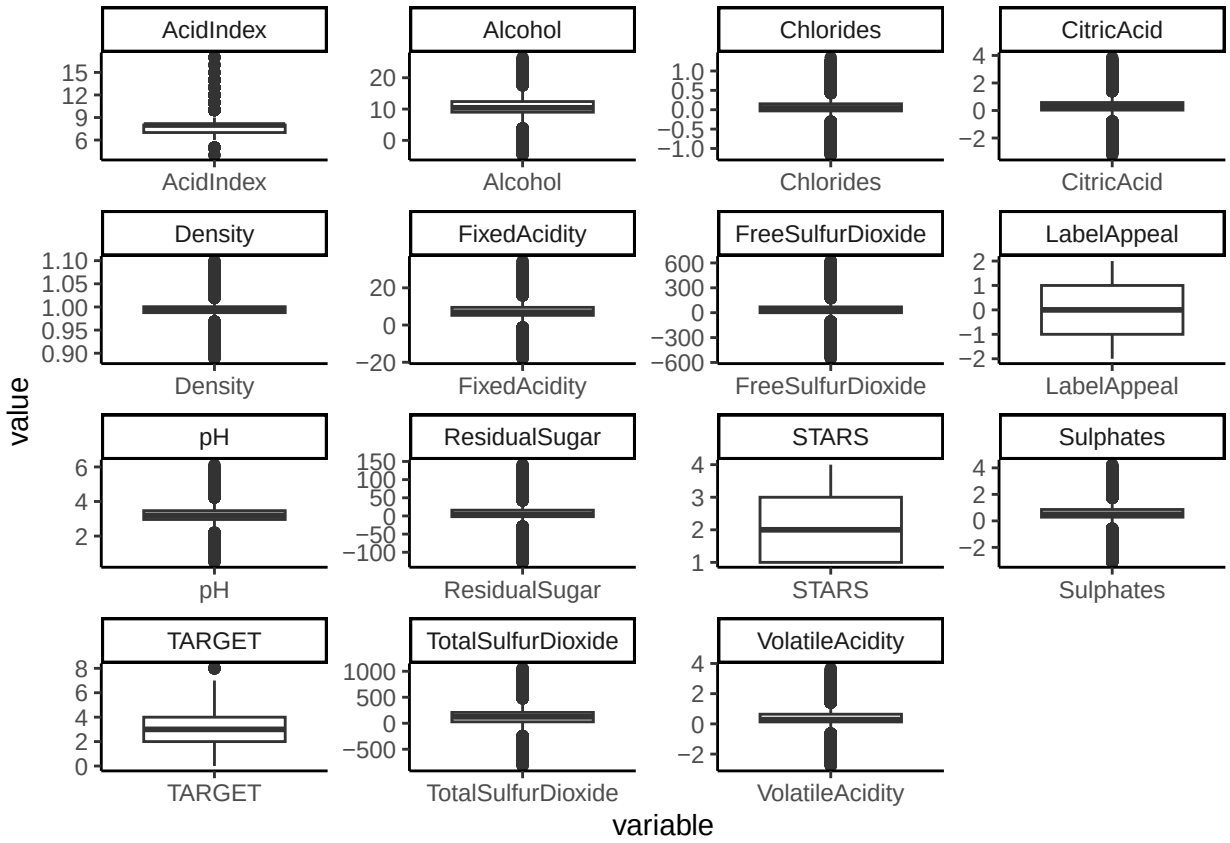
Let's review the frequency of missing values overall across the training dataset:

As mentioned earlier, `STARS` is missing a value for about a quarter of observations, so we will add a missingness level to its factor. `Sulphates` is `NA` for nearly 10 percent of values as well, and some other variables are missing smaller percentages of values. It's possible measurements of these attributes weren't taken for these wines, or the taken measurements were discarded because they were inaccurate.

Review of all variable distributions:

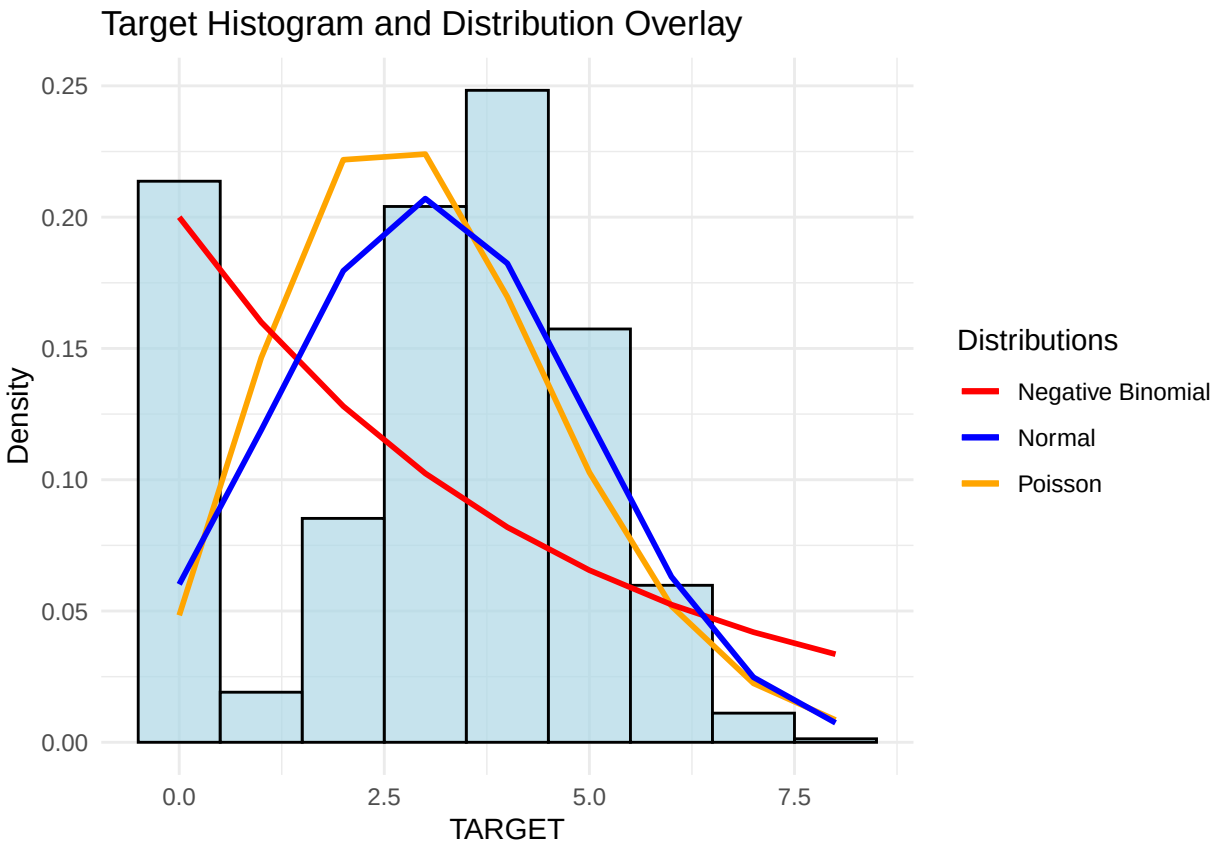


The vast majority of predictor variables appear to have a fairly normal approximation, but the distributions for the predictors that contain what we believe are invalid negative values are probably invalid as well, and their actual distributions are likely to be skewed. `ResidualSugar`, `Chlorides`, `FreeSulfurDioxide`, and `CitricAcid` have some minor skew in their distributions. `AcidIndex` appears to more closely resemble a poisson or maybe exponential distribution, and the skew may require transformation. As discussed previously, the values for `STARS` are discrete in nature, but they are also skewed. `LabelAppeal` is discrete as well and should be similarly be coerced to a factor.



Most of the variables appear to be highly concentrated in the center, in line with the distributions shown in the histograms. Some of the outlier points revealed by the boxplots could be problematic when modeling.

The response variable has a large amount of zero values, which requires some further review:



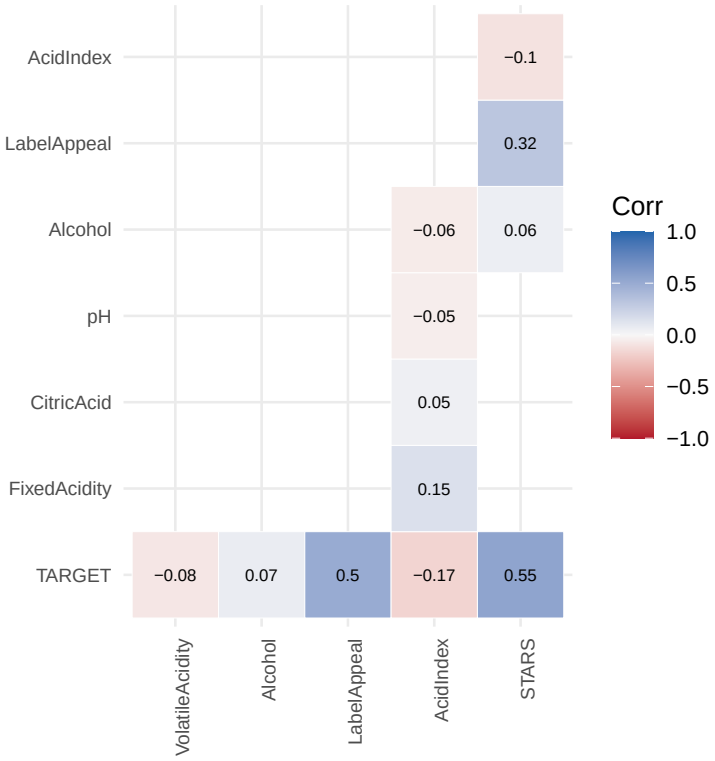
One key assumption when using poisson regression models is that the response is expected to mirror a poisson distribution, which has been plotted on the graph above. An attempt will be made to use a standard poisson model despite the fact that there are many zero values from this distribution. It may be necessary to incorporate zero-inflated modeling techniques to account for this pattern in the data and improve the accuracy/performance of our regression models. The negative binomial function seems to be able to account for the frequency of zero values in the `TARGET` response, but it is not capturing the distribution of the remaining predicted cases very well. The normal distribution was also included to compare against the poisson given that at higher λ values, these two distributions tend to converge, and in this case are not substantially different from one another when evaluating the training data.

Reviewing λ for the poisson distribution:

```
## The response mean: 3.02907385697538 and variance: 3.71089452283923
```

Another important assumption for poisson models relates to the fact that the λ parameter, which is a critical input for the poisson distribution, expects that the mean and variance must be equal. In practice this is nearly impossible and although there is some overdispersion for the response, it is not bad enough to disqualify using this method.

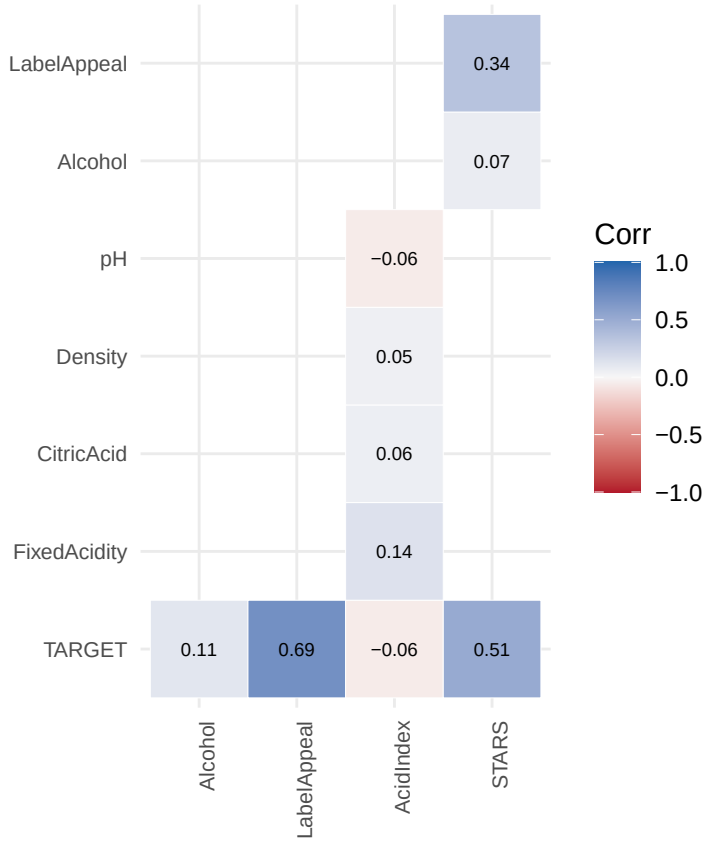
Let's analyze the correlation relationship among predictors and the response. We set a small correlation threshold of 0.05 so nothing below that in absolute value is displayed for readability.



After excluding the NA values, there are weak correlation values across the board except for `TARGET`,`STARS` and `LabelAppeal`. This is not the most encouraging evidence that the predictor variables provided in the dataset are going to be that successful in modeling the response.

One hypothesis for the apparent lack of linear relationships between variables and the response may be driven off the substantial number of zero values in the response. A guess for why there are so many `TARGET` values of zero might be that many brands of wine exist that are minimally distributed on a commercial scale.

What is the relationship among the variables when excluding the zero response values?



After excluding the zero response rows, `LabelAppeal` appears to have a stronger linear relationship with `TARGET` and to a lesser extent `STARS`.

Data Preparation for All Methods:

We coerce the `STARS` and `LabelAppeal` variables to factors as discussed. Wines with NA `STARS` values are likely to be wines that have not been rated by the experts, and it is also likely unrated wines are less interesting to distributors. We will order its factor from `NA` to `4` so that the missingness level is the lowest level. The `LabelAppeal` factor will be ordered naturally from -2 to 2.

Data Preparation for Method A:

Since we will be excluding predictors with invalid negative observations from Method A, we perform no transformations, as the predictors with invalid negative observations are the predictors that would probably benefit most from them.

So all we do for Method A is split the data into train and test sets and remove incomplete cases.

Data Preparation for Method B:

Since we'll be changing the signs of invalid negative observations for Method B, it's possible the data for all predictors will represent valid distributions that could benefit from transformations. So we will undertake those considerations here. (Note that we still believe changing the signs is drastic, and in the real world, we would absolutely need confirmation of the nature of the data entry error to proceed this way. Ideally, the data could be pulled again and delivered to us without these errors.)

First, we change the signs of invalid negative observations. (Recall that negative values for `LabelAppeal` are valid.)

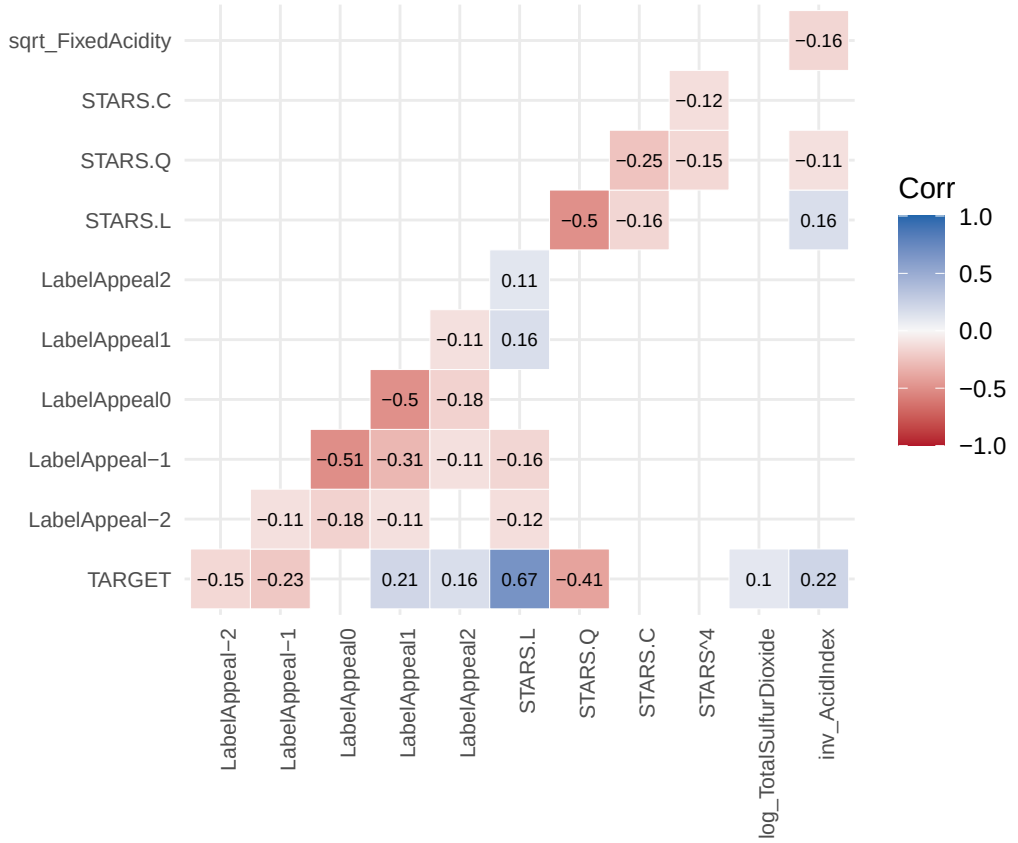
Next, we split the data into train and test sets and eliminate incomplete cases. Having done this for both datasets will allow us to compare metrics like AIC when we run our models, whereas imputing data in one set and trying to only use complete cases in another would prove for problematic evaluation.

We can proceed with a review of numeric predictors that might need transformations.

Variable	Ideal Lambda Proposed by Box-Cox	Reasonable Alternative Transformation
FixedAcidity	0.5	square root
VolatileAcidity	0.2	log
CitricAcid	0.3	log
ResidualSugar	0.15	log
Chlorides	0	log
FreeSulfurDioxide	0.15	log
TotalSulfurDioxide	0.3	log
Density	1.3	none
pH	1.1	none
Sulphates	0.3	log
Alcohol	0.95	none
AcidIndex	-1.2	inverse

The only numeric predictors that might not benefit from transformations are `Density`, `pH`, and `Alcohol`. We transform the data for the remaining numeric predictors.

Before proceeding further let's double check the correlations of the target and predictor variables to see how transformation has impacted the linear relationships. Here, we exclude any correlations less than 0.1 in absolute value for readability.

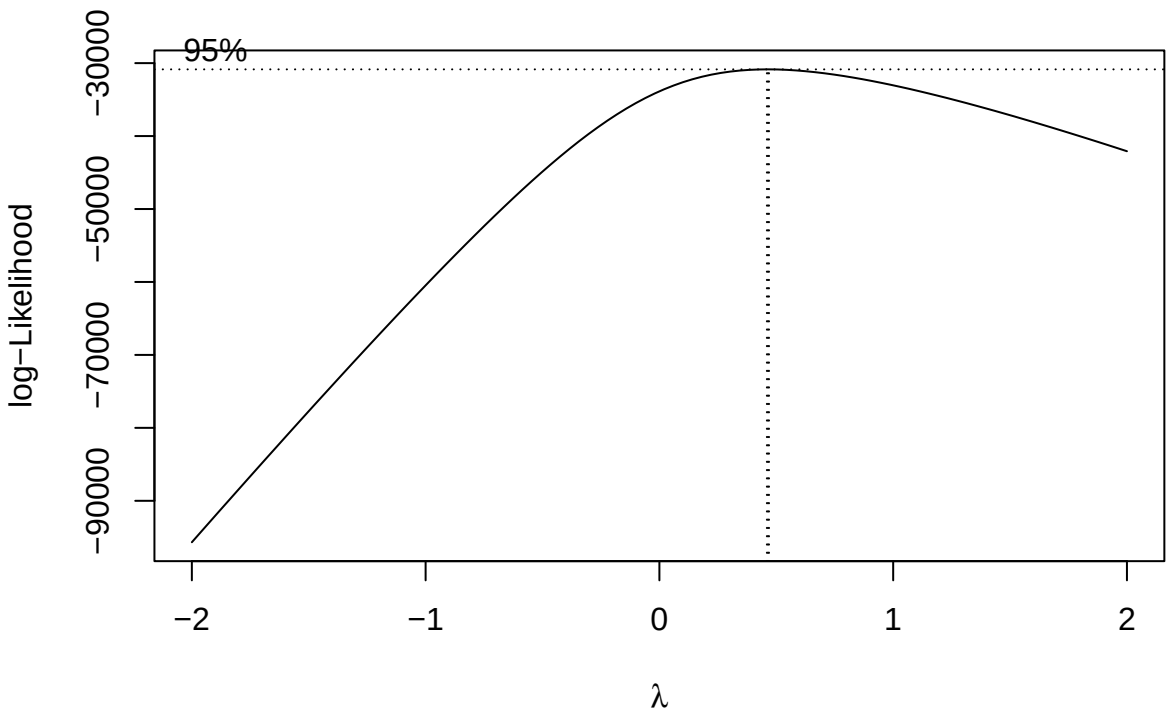


Modifying `STARS` into a factor increased the strength of its positive linear relationship with `TARGET`. Turning `LabelAppeal` into a factor revealed that midling label ratings have no relationship with `TARGET`, but low ratings have a negative correlation, and high ratings have a positive correlation. Low label ratings also negatively correlate with `STARS`, and high label ratings positively correlate with `STARS`.

`inv_AcidIndex` has the strongest positive relationship with `TARGET` among the transformed numeric predictors. It is also slightly negatively correlated with `sqrt_FixedAcidity`, which is not correlated with `TARGET`, so modeling might benefit from only including the former acid measure.

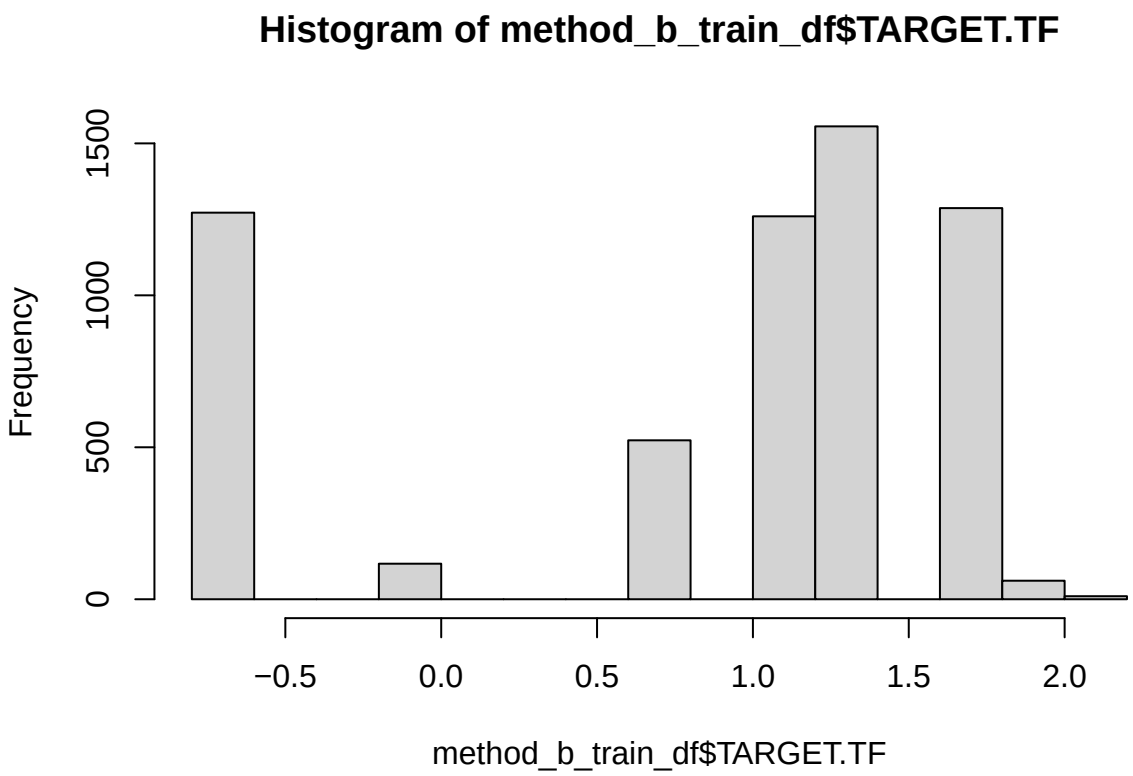
One means of prediction using OLS for count data is taking the ln y as a means of replicating the Poisson linear transformation by applying it on the response. This transformation is based on a UC Davis [publication] (https://cameron.econ.ucdavis.edu/racd/simplepoisson.pdf) on count regression. Therefore we will prepare an alternative response variable, `TARGET.TF`, for Method B although some minor modifications are needed since there are many zero values.

Let's review what the Box-Cox method proposes for the transformation:




```
## [1] 0.4646465
```

The Box-Cox preferred transformation appears to show the square root as the optimal transformation, but given the primer on count data from UC-Davis we will apply the ln y instead. The data for Method B still includes the original `TARGET` variable, so we will be explicit about which models use the alternative response `TARGET.TF` later.



Data Preparation for Method C:

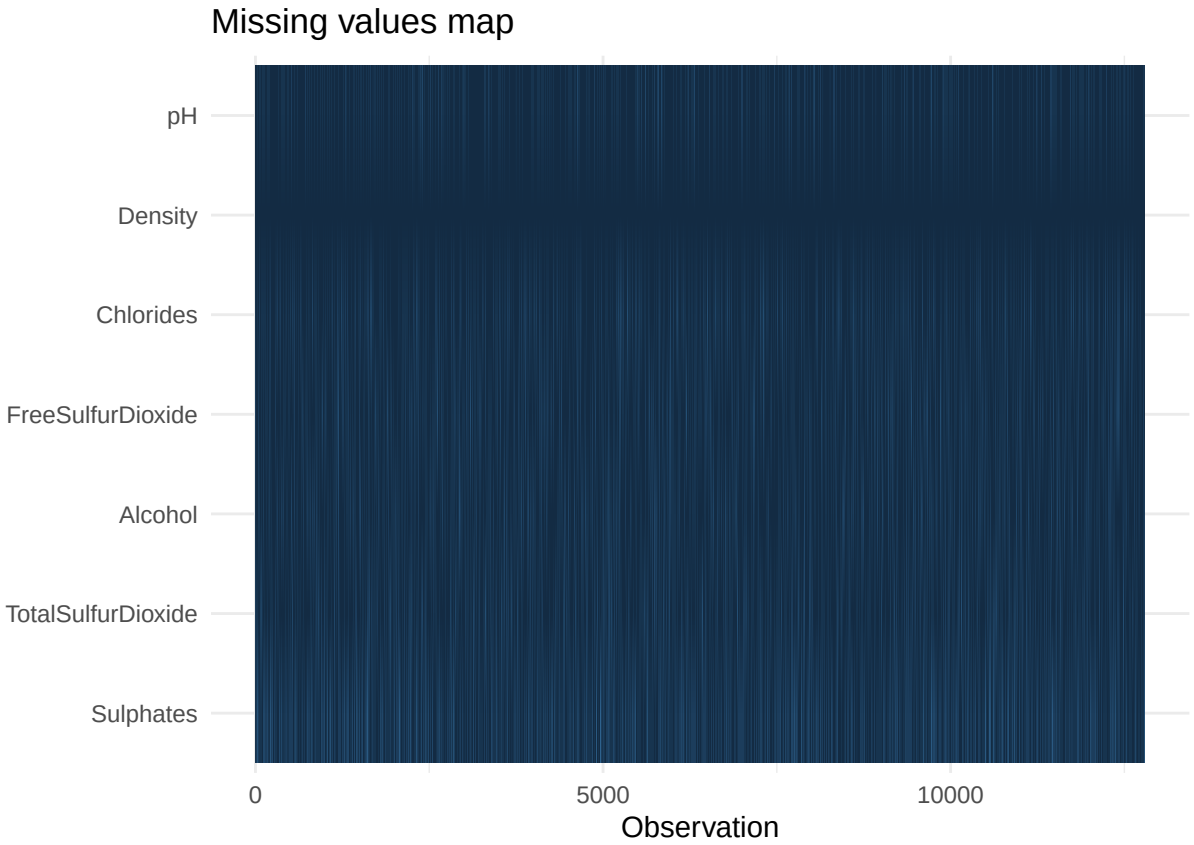
The remaining columns with missing values are far less clear to evaluate and will need a more standardized imputation method:

One more comprehensive method to evaluate if the data is Missing Completely at random (MCAR) is running Little's test although it is unlikely to generate a statistically significant difference for most real world datasets.

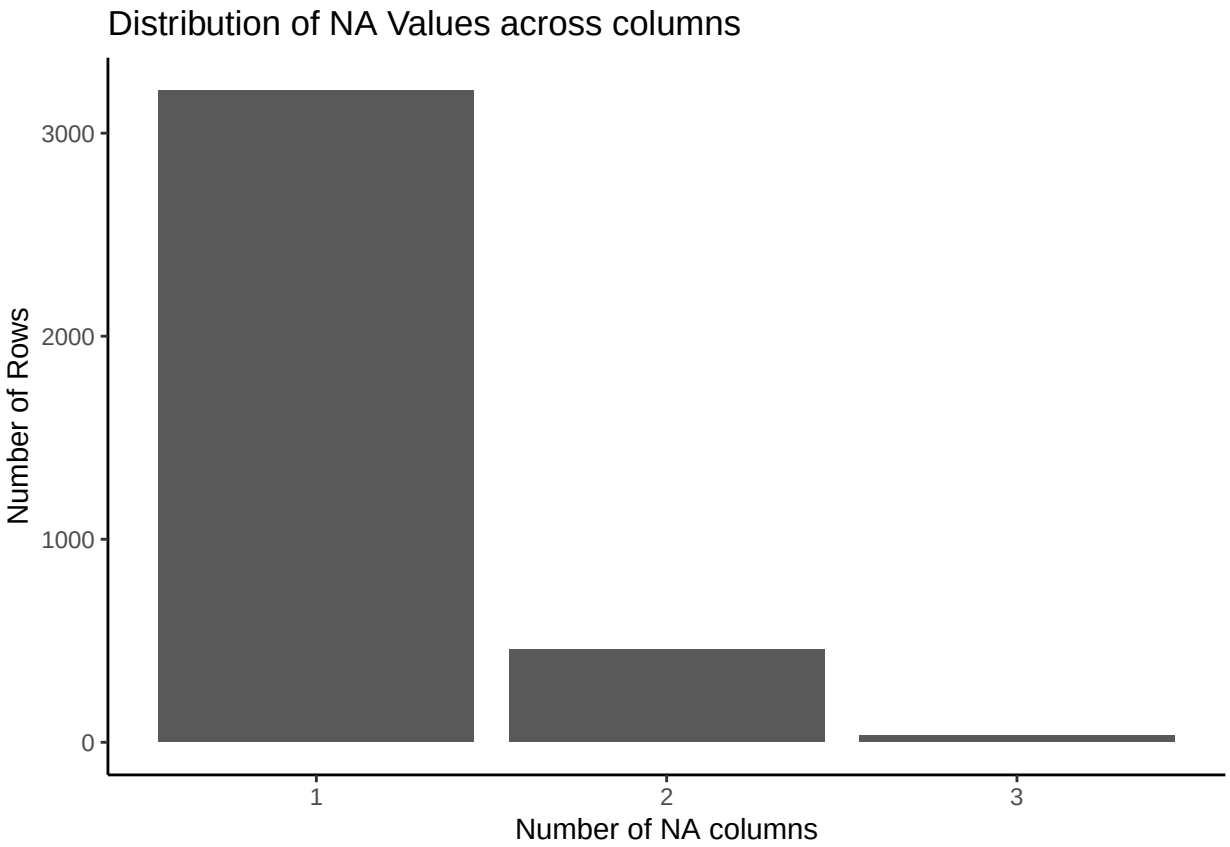
statistic	df	p.value	missing.patterns
694.1909	705	0.6070216	57

The zero in the p-value indicates that the missing values across these columns are not in fact MCAR.

Let's review if the NA values appear in similar observations:



The plot above highlights via a heat map all of the individuals rows where the independent predictors have NA values (shaded in light blue). It's a bit challenging to discern where observations have more than one null value, but there is definitely some overlap across records.




```
## Only 490 rows have more than 1 column with NA, which is only 13.24% of the observations that have NA
```

It does not appear that many observations have more than 1 NA value and the heatmap was a little bit less clear in helping us identify overlapping columns.

Let's split the data into train and test sets to provide some observations to evaluate the selected model as a holdout set:

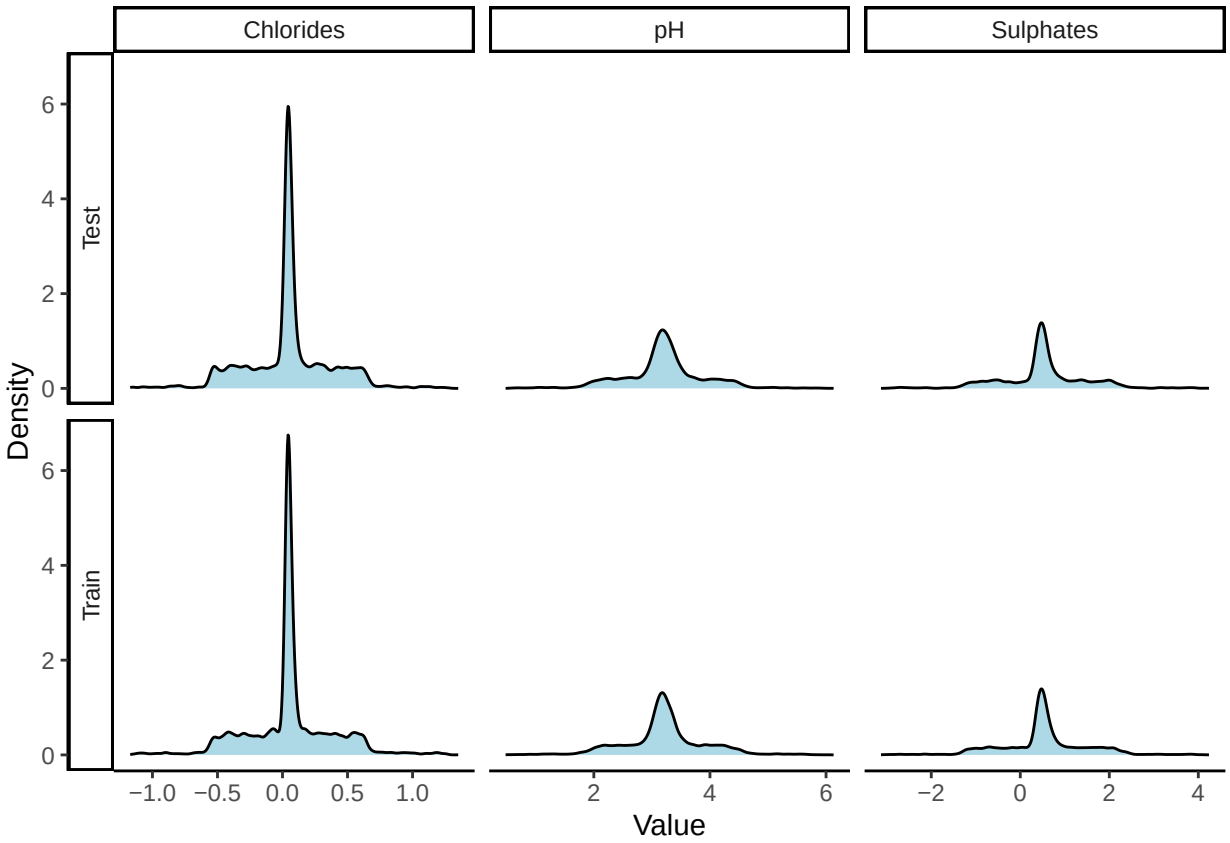
Alternatively, we will use the MICE package to fill the NA values:

Let's confirm that the imputation filled all the null values for MICE:

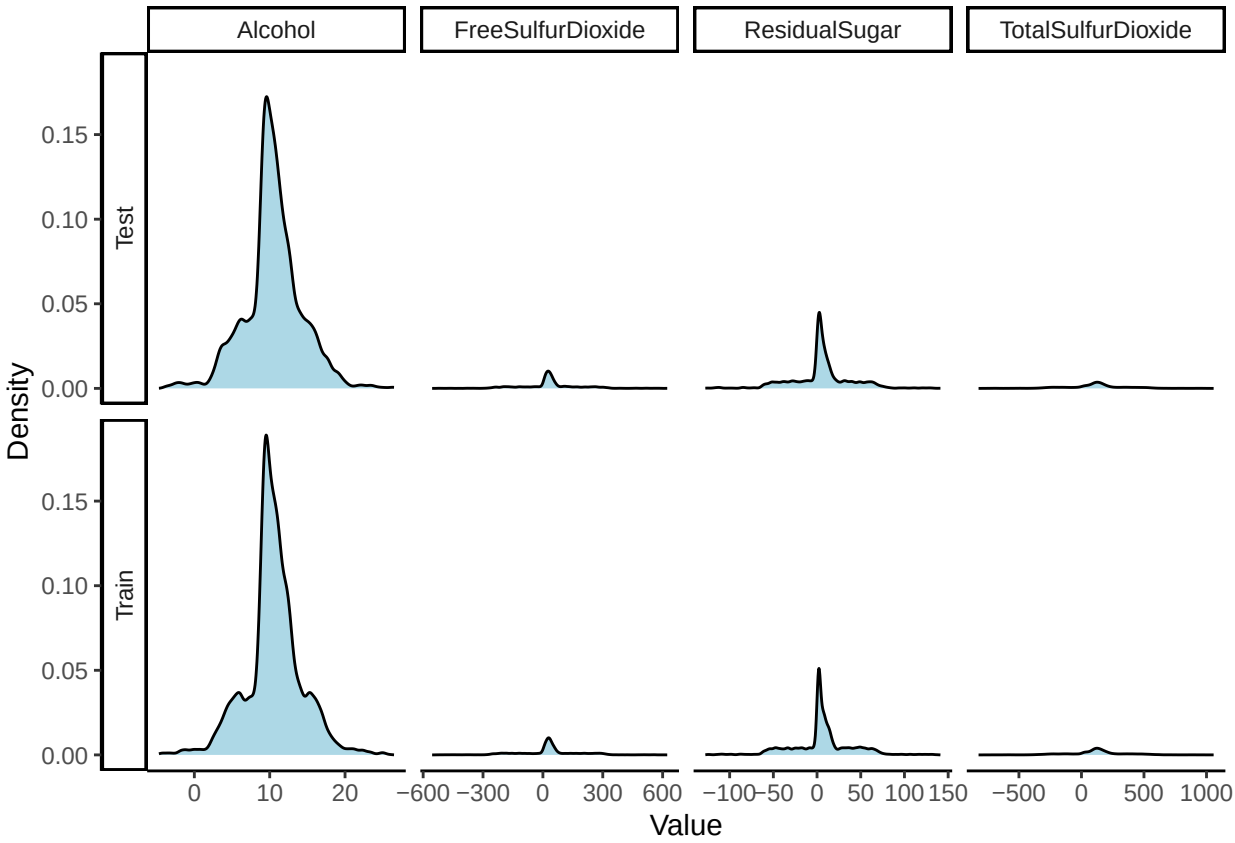
```
## [1] TRUE
```

As expected the MICE imputed estimate values for all the missing fields and we can now include all observations within the model.

Let's review the distributions of the train and test imputed datasets to confirm they remain similar to the initial unmodified dataset:



These columns were separated as they had higher density points distorting the remainder of predictors that had NA values in the density plots. Across train and test sets it still appears to be similarly distributed and not substantially different from the initial input data.



The remaining columns that had imputations also mirror one another in the train and test sets as well as the initial distributions.

Modeling

Model POIS:1 - Poisson Count Model Using Method A Data: Model POIS:1 is a Poisson Count Model using Method A data. Method A data reminders:

- predictors with negative values aside from `LabelAppeal` excluded
- no transformations
- no imputations (i.e. complete cases only)

We start with all predictors not excluded for negative illogical values and perform stepwise model selection to pick the model with the smallest AIC value. The response variable for this model is `TARGET`. A summary of Model POIS:1 is below:

```
##
## Call:
## glm(formula = TARGET ~ STARS + LabelAppeal + AcidIndex, family = "poisson",
##     data = method_a_train_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2201  -0.6157   0.0153   0.4421   3.7934
```

```
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.676756   0.050756  33.036  < 2e-16 ***
## STARS.L        0.972652   0.023847  40.788  < 2e-16 ***
## STARS.Q       -0.378394   0.020609 -18.360  < 2e-16 ***
## STARS.C        0.141238   0.017609   8.021 1.05e-15 ***
## STARS^4       -0.004379   0.014344  -0.305    0.760
## LabelAppeal.L  0.529977   0.039427  13.442  < 2e-16 ***
## LabelAppeal.Q -0.054411   0.033082  -1.645    0.100
## LabelAppeal.C  0.010643   0.023488   0.453    0.650
## LabelAppeal^4  0.022290   0.014979   1.488    0.137
## AcidIndex     -0.077476   0.006427 -12.055  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 10635.5  on 6085  degrees of freedom
## Residual deviance:  6523.9  on 6076  degrees of freedom
## AIC: 21807
##
## Number of Fisher Scoring iterations: 6
```

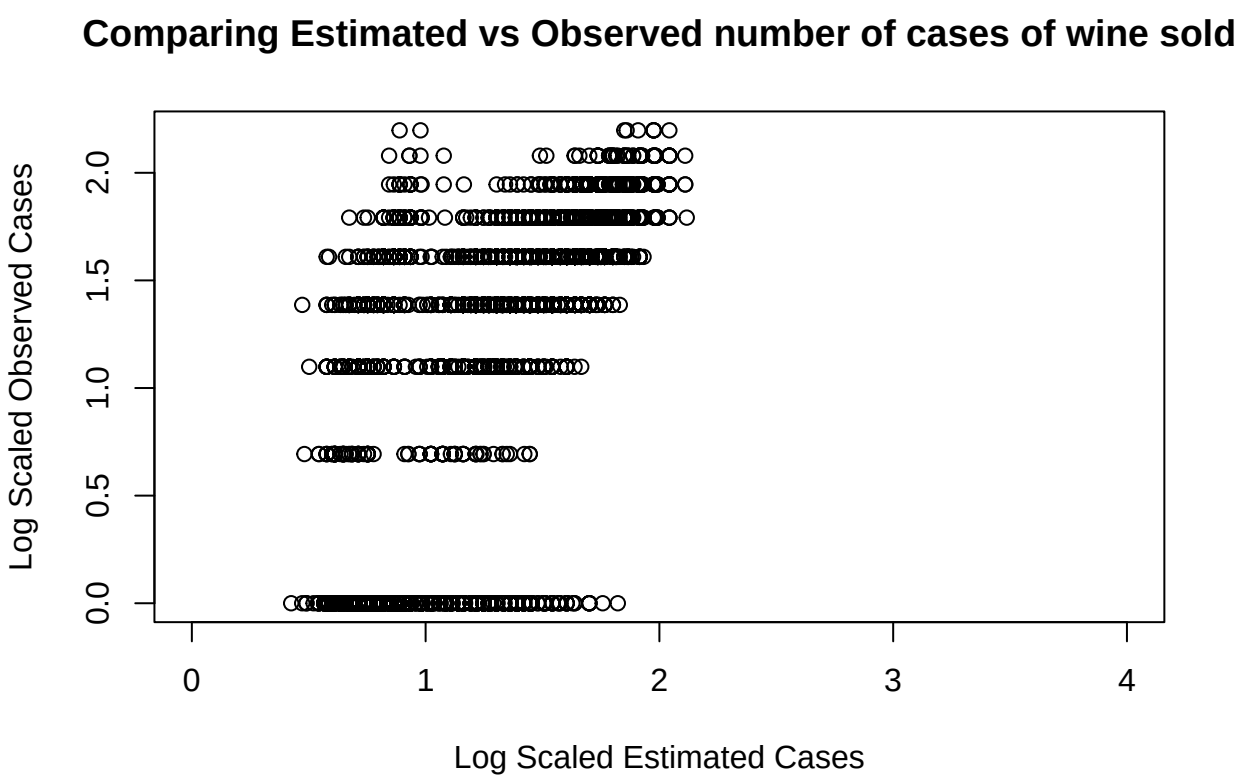
The AIC of Model POIS:1 is 21807. `STARS`, `LabelAppeal`, and `AcidIndex` are the only statistically significant predictors in Model POIS:1. We will assess the predictors more formally after building a zero inflated equivalent as a comparison.

Residual Deviance: Chi-Squared Test:

```
##      res.deviance   df            p
## [1,]     6523.874 6076 3.546715e-05
```

This test measures the goodness of fit of the poisson model and an ideal case is when the test returns a non-significant p-value. It is designed to compare the expected counts (response) returned from the model to the observed values, and if the model did as expected there would not be a statistically significant difference between the two values. Therefore, this significant p-value would indicate that the data does not fit the model well. Model POIS:1 violates the linear distribution of the Poisson, and there was more dispersion than allowed for by the Poisson.

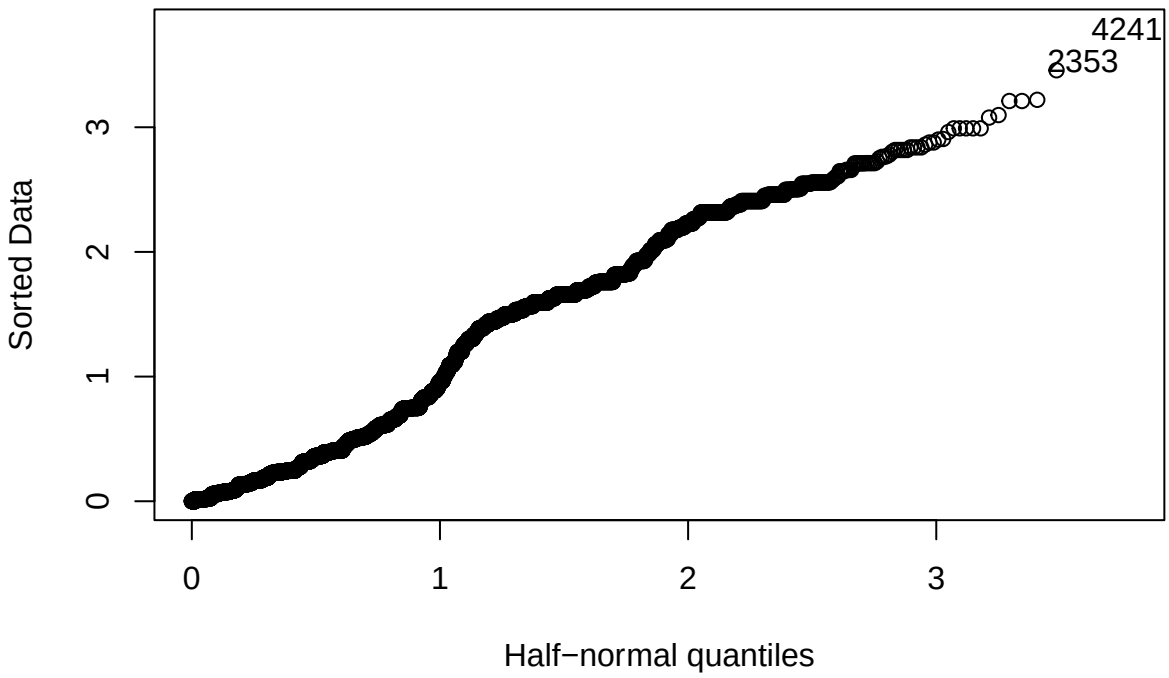
Let's plot the log scaled version of the response and the predicted number of cases:



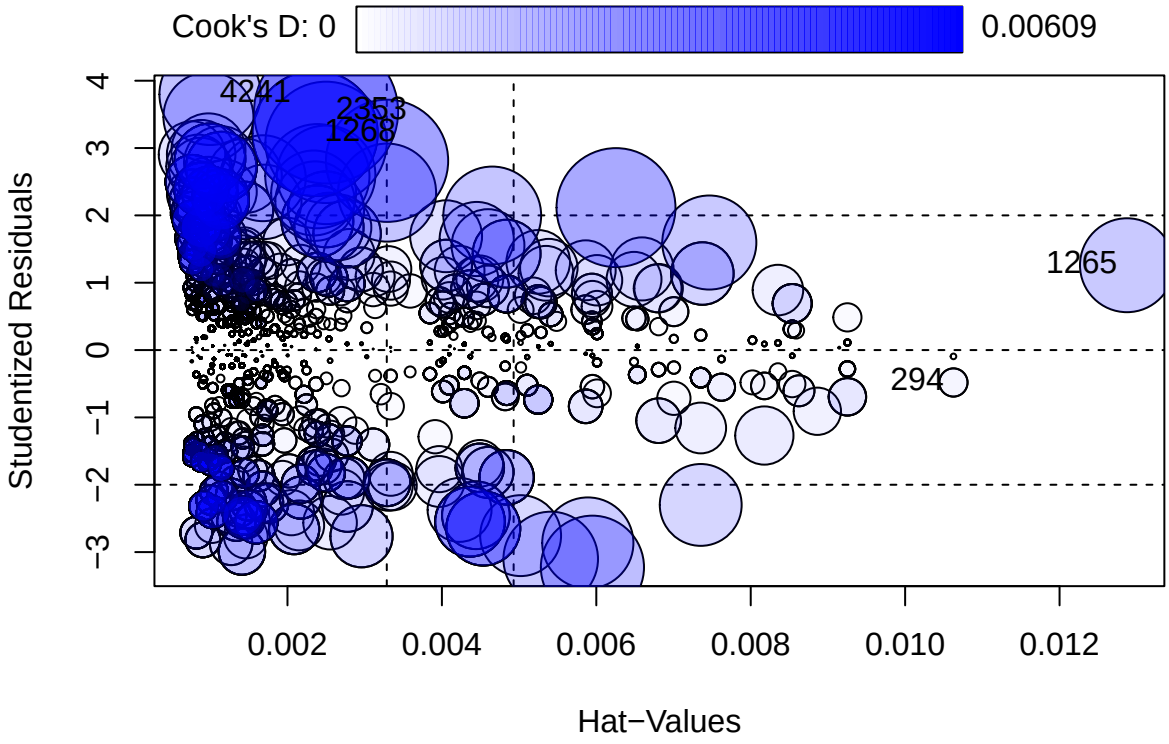
Consistent with the Chi-Squared Test it appears there are too many observed values of zero to fit into the Poisson distribution. This plot is designed to identify where the estimated number of cases on the x-axis diverges from the actual values, particularly emphasizing the concentration of zero values that a Poisson distribution is unable to accurately estimate.

The half-norm plot of the residuals and a plot of Cook's D statistic are two different ways of identifying outliers:

The half norm plot assesses the distribution using quantiles similar to the qqplot.

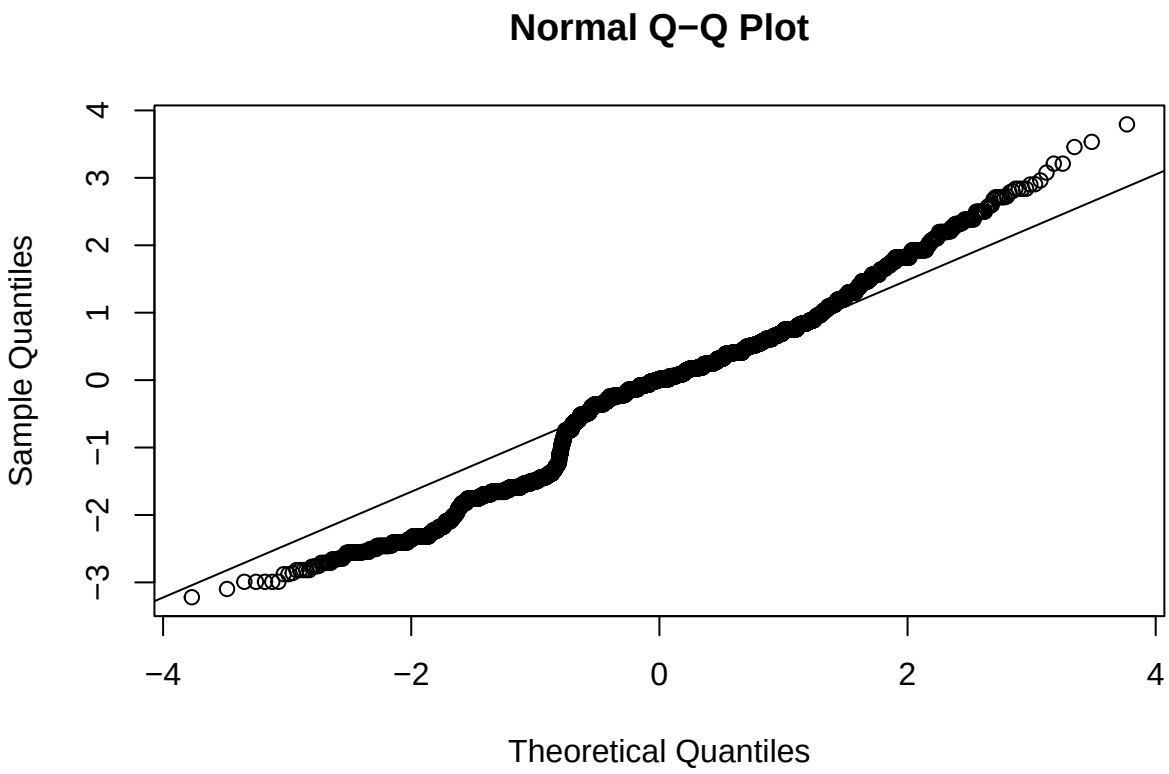


The half-norm plot highlights two points at the top right of the graph, but does not appear to show substantial divergence to warrant classifying observations 4241 and 2353 as outliers.




```
##          StudRes          Hat        CookD
## 294  -0.4812313 0.0106257018 0.0002341174
## 1265  1.2598913 0.0128729662 0.0025882002
## 1268  3.2177877 0.0023570071 0.0046043292
## 2353  3.5421853 0.0025000137 0.0060871675
## 4241  3.7973409 0.0009974284 0.0030106673
```

The influence plot appears to show that point 1265 is a leverage point; however, it doesn't cross the 2 boundary that would classify it as an outlier for the model. There are other points listed below the graph as well that have much larger standardized residual values, but don't have divergent case predictions and therefore do not appear to be outliers as well.



The QQplot would seem to indicate that the deviance residuals are diverging from the expected distribution. We check for over/underdispersion of the model to see if there are any additional violations of assumptions. Let's calculate the dispersion from the model:

```
## [1] 0.8839235
```

It appears there is a decent amount of under-dispersion in this model as the ideal case would return a value of 1, which may limit our abilities to reliably use Poisson Regression for this count data response consistent with other diagnostic methods conducted. Generally, the concern when evaluating Poisson regression is assessing overdispersion violations, but in this case the zero values are driving this issue. It does appear that the frequency of zero values does not follow a standard Poisson distribution.

Model POIS:2 - Zero Inflated Poisson (ZIP) Count Model Using Method B Data: Model POIS:2 is a ZIP model using Method B data. Method B Data Reminders:

- signs of negative observations were changed for all predictors except `LabelAppeal`
- some numeric predictors were transformed, mostly using log transformations
- no imputations (i.e. complete cases only)

We start with all predictors and perform backward stepwise variable elimination. The response variable for this model is `TARGET`. Only the final summary of Model POIS:2 is below:

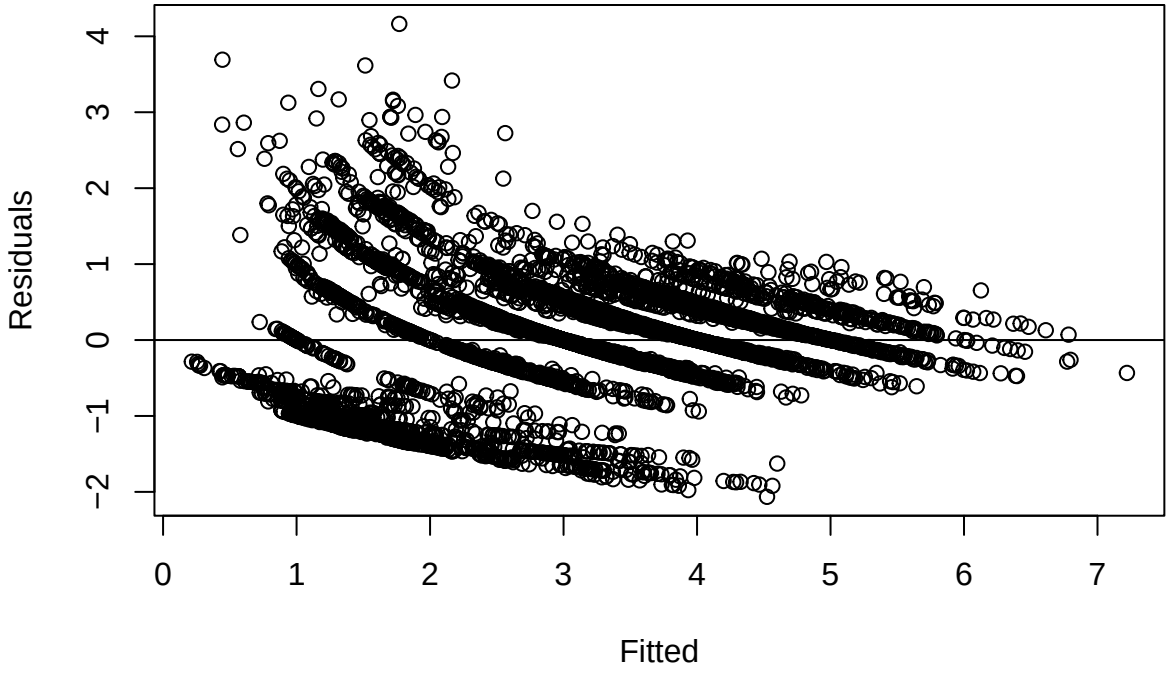
```
##
## Call:
## pscl::zeroinfl(formula = TARGET ~ . - TARGET.TF - log_ResidualSugar -
##     sqrt_FixedAcidity - pH - log_TotalSulfurDioxide - log_Sulphates -
##     log_FreeSulfurDioxide - Density - log_Chlorides | LabelAppeal + inv_AcidIndex,
##     data = method_b_train_df, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -2.0647 -0.4388  0.0799  0.4690  4.1632
##
## Count model coefficients (poisson with log link):
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           0.978618   0.063148  15.497  < 2e-16 ***
## Alcohol               0.005916   0.002137   2.768  0.00564 **
## LabelAppeal.L         0.685690   0.042244  16.232  < 2e-16 ***
## LabelAppeal.Q        -0.088257   0.034805  -2.536  0.01122 *
## LabelAppeal.C         0.013386   0.024667   0.543  0.58736
## LabelAppeal^4         0.021110   0.015804   1.336  0.18164
## STARS.L               0.721200   0.029360  24.564  < 2e-16 ***
## STARS.Q              -0.287559   0.023101 -12.448  < 2e-16 ***
## STARS.C               0.148389   0.018668   7.949 1.89e-15 ***
## STARS^4              -0.032754   0.014779  -2.216  0.02667 *
## log_VolatileAcidity -0.024205   0.008218  -2.946  0.00322 **
## log_CitricAcid        0.031377   0.007585   4.137 3.52e-05 ***
## inv_AcidIndex         0.893793   0.428947   2.084  0.03719 *
##
## Zero-inflation model coefficients (binomial with logit link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     4.7473     0.5864   8.096 5.68e-16 ***
## LabelAppeal.L   1.9943     1.0036   1.987   0.0469 *
## LabelAppeal.Q  -0.9089     0.8506  -1.068   0.2853
## LabelAppeal.C   0.2617     0.5200   0.503   0.6148
## LabelAppeal^4  -0.1415     0.2281  -0.620   0.5351
## inv_AcidIndex -59.3439     4.1245 -14.388  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 58
## Log-likelihood: -1.063e+04 on 19 Df
```

There is some discussion in statistical publications around using robust methods to estimate the standard error when using zero inflation techniques, but we will review other diagnostic methods to evaluate this model although it does not typically impact coefficients. The binomial with logit link portion of the model considered only the strongest predictors (`STARS`, `LabelAppeal`, and `inv_AcidIndex`) to identify the instances when no cases were expected to be sold based on the expectation that negative customer evaluation, missing expert reviews, and high acid index levels would prompt distributors to not purchases cases of these wines. `STARS` was insignificant in that regard. When reviewing the coefficients in the poisson with log link portion of the model, `STARS`, `LabelAppeal`, `Alcohol`, `log_VolatileAcidity`, `log_CitricAcid`, and `inv_AcidIndex` were all significant.

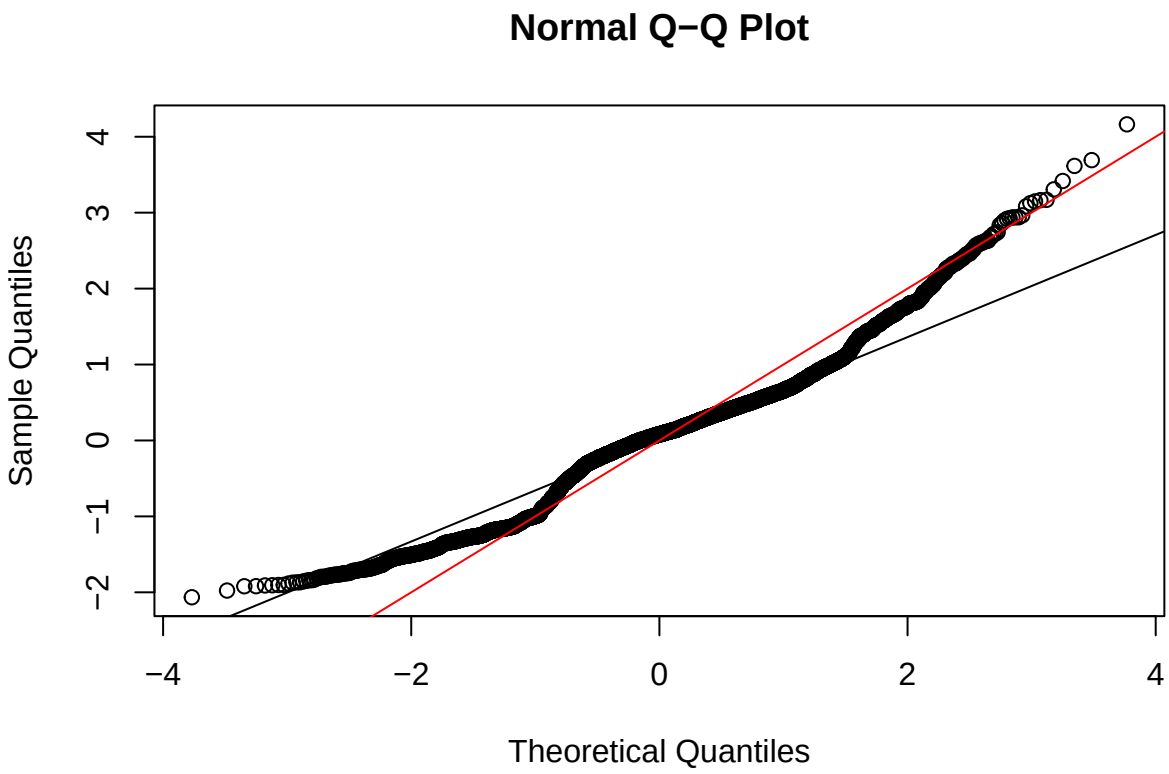
Let's run a Chi-Squared Test to evaluate the goodness of fit:

```
## 'log Lik.' 0 (df=19)
```

A statistically significant result in the Chi-Squared Test would seem to indicate that there is a goodness of fit for the ZIP model. The degrees of freedom is based on the number of predictors in the model excluding the intercept (the null model).



When assessing the fitted vs residuals plot there is some pattern to the residuals, and the largest ones still exist for the zero cases examples, it would appear. They are still centered around zero.



The residuals follow the QQline for some of the quantiles, but appear to diverge on the higher end and follow a line tracking from zero to one.

```
## [1] 0.3122495
```

There is more substantial underdispersion captured in this model than the other poisson regression one and we would like to avoid any under/over dispersion that exists.

Let's compare the Akaike Information Criterion for both poisson based models:

```
##           df      AIC
## mod1_pois 10 21807.01
## mod2_zip  19 21300.95
```

The AIC value for Model POIS:2 (ZIP) is slightly lower than the AIC value for Model POIS: 1.

Model NB:1 - Negative Binomial Count Model Using Method A Data: The negative binomial distribution can sometimes better approximate dispersion when the mean and variance are not equal to one another which would violate the Poisson distribution and related assumptions for Poisson Count regression. Therefore, we will use the same variables and data as selected in Model POIS:1 to try to create a similar comparison. The response variable for this model is `TARGET`.

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
##
## Call:
## glm.nb(formula = TARGET ~ STARS + LabelAppeal + AcidIndex, data = method_a_train_df,
##     init.theta = 41260.37945, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2200  -0.6157   0.0153   0.4421   3.7932
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.676773   0.050758  33.035  < 2e-16 ***
## STARS.L        0.972653   0.023848  40.786  < 2e-16 ***
## STARS.Q       -0.378393   0.020610 -18.360  < 2e-16 ***
## STARS.C        0.141237   0.017610   8.020 1.06e-15 ***
## STARS^4       -0.004379   0.014344  -0.305    0.760
## LabelAppeal.L  0.529974   0.039429  13.441  < 2e-16 ***
## LabelAppeal.Q -0.054410   0.033083  -1.645    0.100
## LabelAppeal.C  0.010644   0.023489   0.453    0.650
## LabelAppeal^4  0.022291   0.014979   1.488    0.137
## AcidIndex     -0.077478   0.006427 -12.055  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(41260.38) family taken to be 1)
##
##     Null deviance: 10634.9  on 6085  degrees of freedom
## Residual deviance:  6523.6  on 6076  degrees of freedom
## AIC: 21809
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  41260
##           Std. Err.:  51076
## Warning while fitting theta: iteration limit reached
##
##  2 x log-likelihood:  -21787.21
```

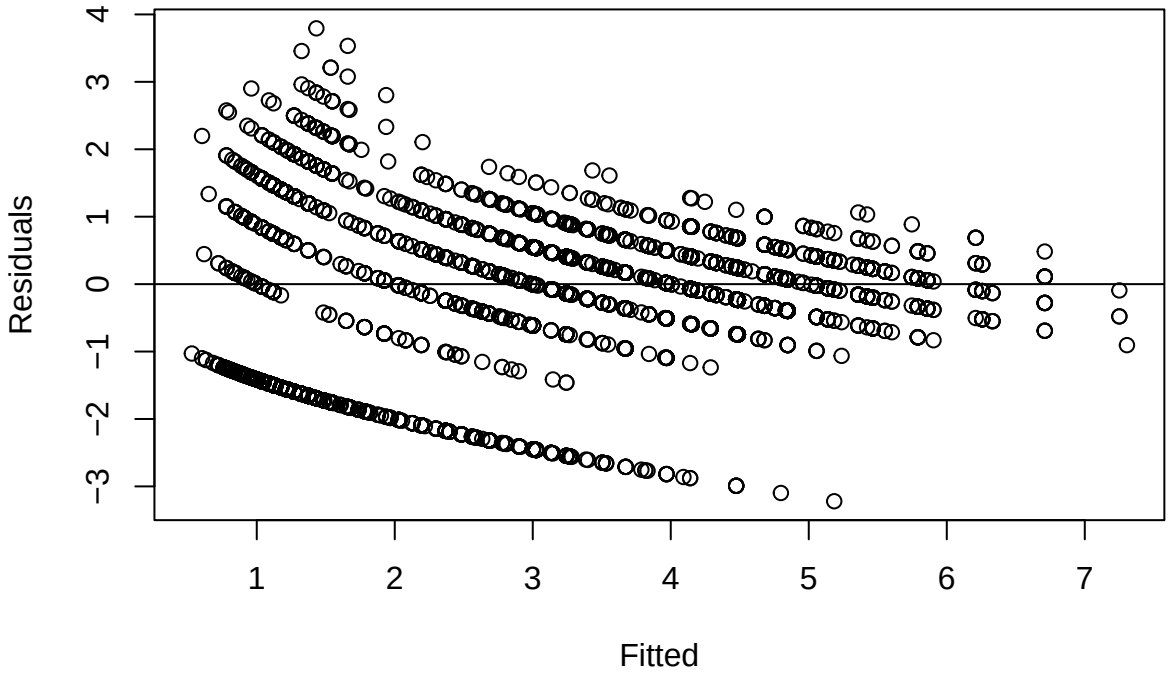
It appears that the algorithm is unable to converge with a negative binomial distribution.

In terms of the strength of the coefficients, the `STARS` ,`LabelAppeal`, and `AcidIndex` all have the signs we saw previously in Model POIS: 1.

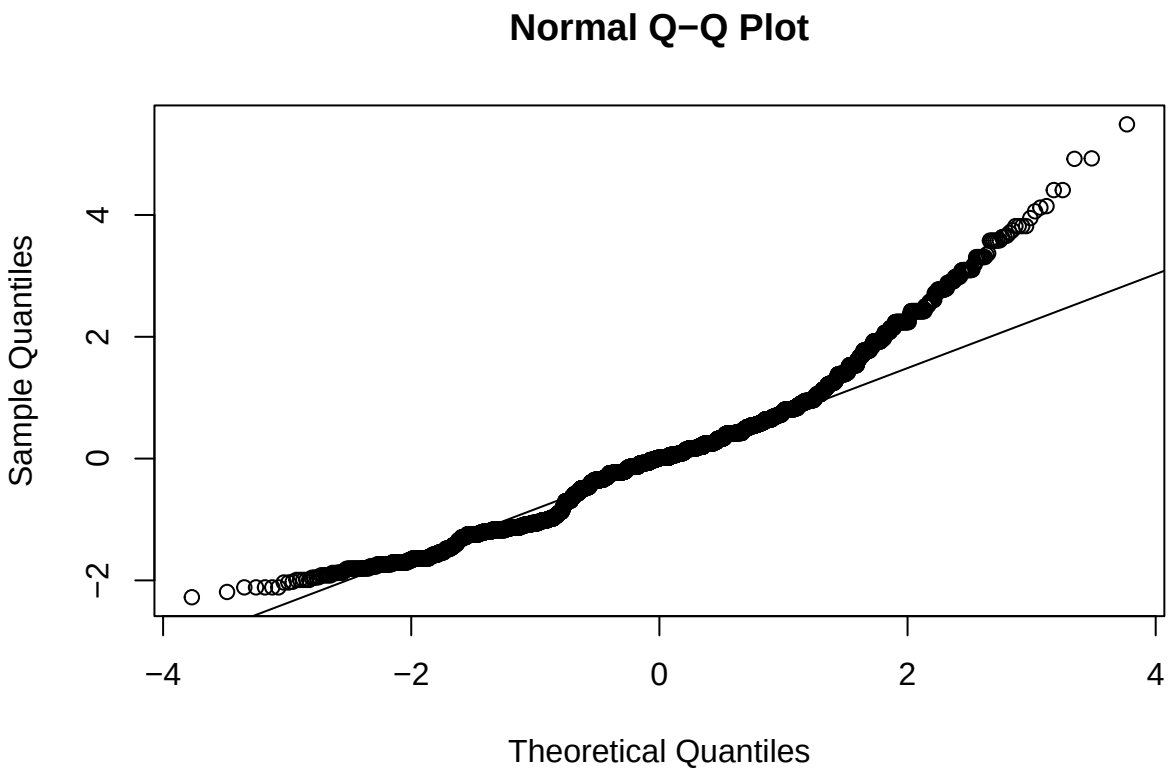
One way to assess the negative binomial model is to run a likelihood ratio test to compare Model NB:1 using the negative binomial family against Model POIS:1 using the Poisson family.

```
## 'log Lik.' 1 (df=11)
```

Based on this test, it would appear that the negative binomial model is not appropriate as the result is above the generally accepted alpha level at 5%.



When reviewing the fitted versus residuals plot, there is some pattern in the residuals which appears to track closely with the `TARGET` shape, where even the negative binomial has the largest residuals for many zero cases. The spread overall doesn't make it particularly clear if the variance is changing, although there is clear separation in the plot due to the discrete nature of the response variable.



When using the pearson residuals we can see that for the interquartile range the negative binomial model tracks well with the expected distribution, but ultimately diverges on both tails somewhat similarly to the other ZIP model. On the higher end, there is substantial skew which would indicate that the residuals are not normally distributed.

Let's calculate the dispersion for this model:

```
## [1] 0.420093
```

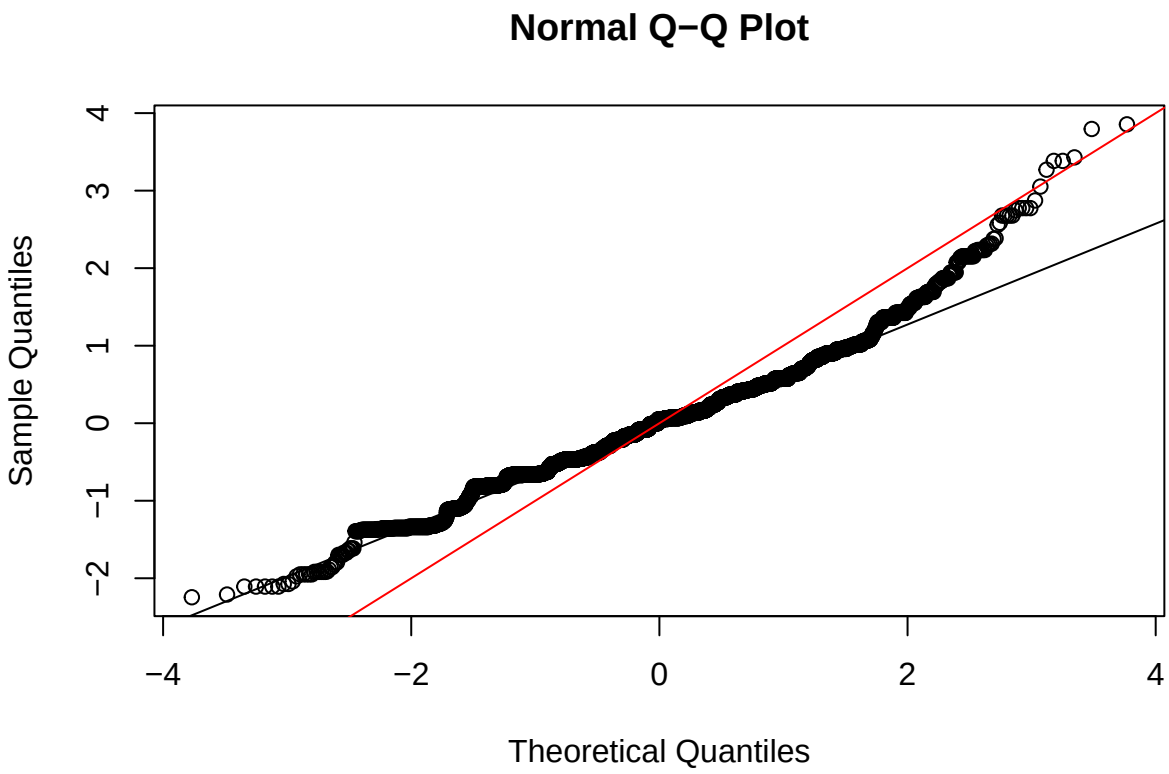
It appears there is more underdispersion in this model than we identified in Model POIS:1, but less than in Model POIS:2 (ZIP).

Model NB:2 - Zero Inflated Negative Binomial (ZINB) Using Method A Data: Let's try to evaluate the data based on a zero inflated negative binomial using Method A data and the same predictors identified as significant by Model POIS:1. The response variable for this model is `TARGET`.

```
##
## Call:
## pscl::zeroinfl(formula = TARGET ~ STARS + LabelAppeal + AcidIndex | STARS +
##     LabelAppeal, data = method_a_train_df, dist = "negbin")
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -2.24467 -0.46593  0.04806  0.41129  3.85506
##
```

```
## Count model coefficients (negbin with log link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.506373   0.054810  27.484  < 2e-16 ***
## STARS.L        0.311074   0.025146  12.371  < 2e-16 ***
## STARS.Q        0.025608   0.021451   1.194    0.233
## STARS.C       -0.011852   0.018328  -0.647    0.518
## STARS^4        0.007720   0.014786   0.522    0.602
## LabelAppeal.L  0.818622   0.042606  19.214  < 2e-16 ***
## LabelAppeal.Q -0.168398   0.035573  -4.734 2.20e-06 ***
## LabelAppeal.C  0.032860   0.024971   1.316    0.188
## LabelAppeal^4  0.003925   0.015607   0.252    0.801
## AcidIndex     -0.033596   0.006996  -4.802 1.57e-06 ***
## Log(theta)    17.062061        NaN     NaN      NaN
##
## Zero-inflation model coefficients (binomial with logit link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -9.6870   264.8765  -0.037   0.9708
## STARS.L       -19.0722   766.2888  -0.025   0.9801
## STARS.Q        -1.9554   647.6320  -0.003   0.9976
## STARS.C         5.1946   538.3262   0.010   0.9923
## STARS^4         4.1236   326.7551   0.013   0.9899
## LabelAppeal.L   2.7522     0.4249   6.477 9.37e-11 ***
## LabelAppeal.Q  -0.7327     0.3604  -2.033   0.0420 *
## LabelAppeal.C   0.2114     0.2289   0.923   0.3558
## LabelAppeal^4  -0.2398     0.1146  -2.093   0.0364 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 25701535.1852
## Number of iterations in BFGS optimization: 25
## Log-likelihood: -9819 on 20 Df
```

From a review of the summary, the residuals appear to be distributed around zero although there is a slight positive skew as compared to the negative. All of the predictors are also significant which is consistent with the Model POIS:1. Removing `STARS` from the binomial with logit link portion of the model makes `LabelAppeal` statistically insignificant strangely, so we leave it in.



The QQplot aligns more closely with the distribution line; however, similar to the other distributions there are a lot of points on the right tail that are skewed.

Let's calculate the dispersion from the model:

```
## [1] 0.2180696
```

The dispersion statistic is the lowest so far, which is a knock against this model.

	df	AIC
mod1_pois	10	21807.01
mod2_zip	19	21300.95
mod3_nb	11	21809.21
mod4_zinb	20	19678.73

However, this model has the lowest AIC so far as well.

Model MLR:1 - Multiple Linear Regression Using Method A Data: The response variable for our first multiple linear regression model, Model MLR: 1, which uses Method A data, is `TARGET`.

```
##
## Call:
## lm(formula = TARGET ~ STARS + LabelAppeal + Density + AcidIndex,
##     data = method_a_train_df)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9177 -0.8617  0.0568  0.8599  6.2789
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.084998   0.642712   9.468  < 2e-16 ***
## STARS.L        2.828955   0.058881  48.046  < 2e-16 ***
## STARS.Q       -0.433617   0.051181  -8.472  < 2e-16 ***
## STARS.C        0.171383   0.043168   3.970 7.27e-05 ***
## STARS^4        0.086565   0.035626   2.430   0.0151 *
## LabelAppeal.L  1.462115   0.080688  18.121  < 2e-16 ***
## LabelAppeal.Q  0.146456   0.068004   2.154   0.0313 *
## LabelAppeal.C  0.002933   0.049411   0.059   0.9527
## LabelAppeal^4  0.071508   0.033019   2.166   0.0304 *
## Density       -1.116136   0.642206  -1.738   0.0823 .
## AcidIndex     -0.194646   0.012870 -15.125  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.319 on 6075 degrees of freedom
## Multiple R-squared:  0.521,  Adjusted R-squared:  0.5203
## F-statistic: 660.9 on 10 and 6075 DF,  p-value: < 2.2e-16
```

When running backward selection on the variables in multiple linear regression we see that the `Density` variable remains despite being classified as insignificant. The signs for each coefficient match Model NB:1 and the most significant predictors remain: `STARS`, `LabelAppeal`, and `AcidIndex`. The F-statistic indicates that the model is better than a pure intercept approach when running MLR.

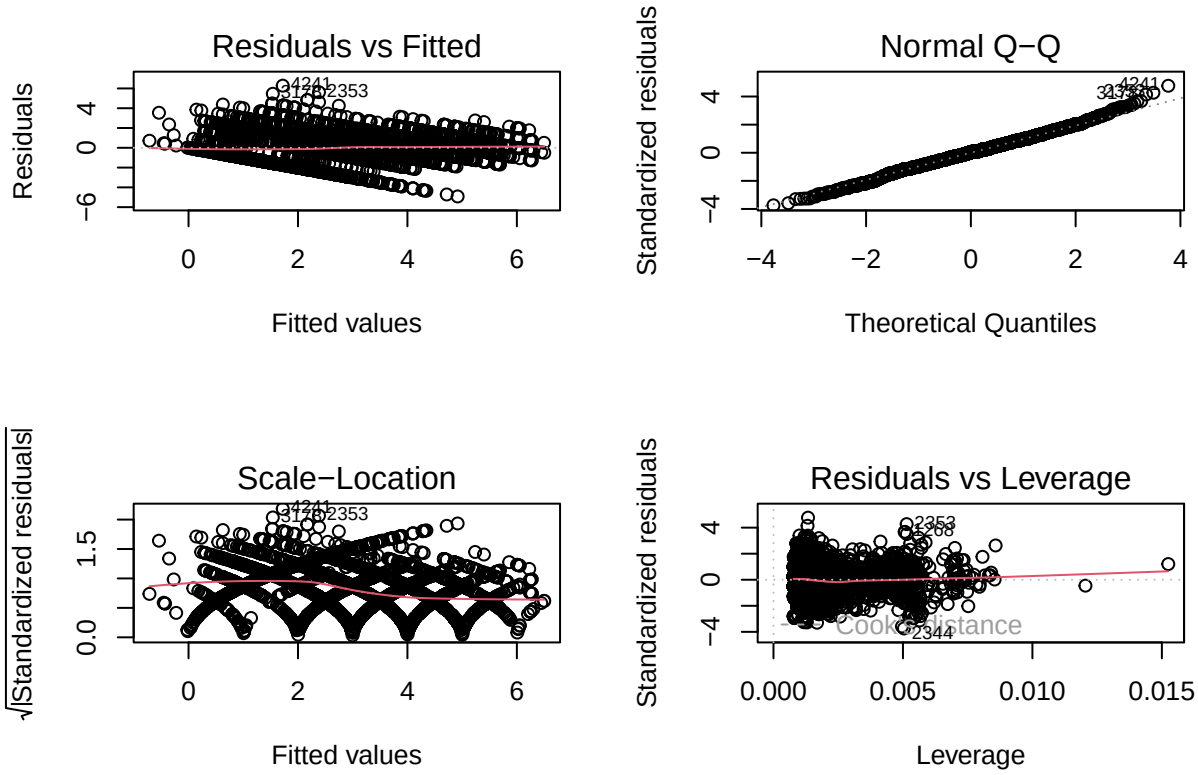
The residuals appear to be somewhat evenly spread around zero although there is a partial right skew. The R^2_{adj} value shows about 52 percent of the variability can be explained by the variation in the predictors.

Are there any multicollinearity concerns in this model?

	x
STARS.L	1.689053
STARS.Q	1.713298
STARS.C	1.270748
STARS^4	1.081520
LabelAppeal.L	1.790042
LabelAppeal.Q	1.266301
LabelAppeal.C	1.695319
LabelAppeal^4	1.256441
Density	1.004795
AcidIndex	1.038685

There are no issues with multicollinearity, which is to be expected given the low correlation values across many of the predictors in the correlation plot from the exploratory section.

Let's review the diagnostic plots to assess if any assumptions are violated:



The Fitted vs. Residuals plot appears to show a slight pattern with the majority of the residuals above zero, although this kind of makes sense given the heavy concentration of zeros and the fact that the predicted number of cases will never be negative. The QQ plot seems to follow the assumption that the residuals follow a normal distribution, and the right skew does not diverge that substantially from the expected case. The standardized residuals show a clear pattern at each discrete value, which is generally one reason why OLS is not encouraged for count data. It does not appear as if a linear model adequately captures the relationship between the predictors and the response variables. There are not a ton of high leverage points.

Model MLR:2 - Multiple Linear Regression Using Method B Data Unlike with all the prior models, the response variable in our second multiple linear regression model, Model MLR:2, is `TARGET.TF`

```
##
## Call:
## lm(formula = TARGET.TF ~ Density + pH + LabelAppeal + STARS +
##     log_VolatileAcidity + log_CitricAcid + log_ResidualSugar +
##     log_Chlorides + log_FreeSulfurDioxide + log_TotalSulfurDioxide +
##     log_Sulphates + inv_AcidIndex, data = method_b_train_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.20806 -0.33793  0.04998  0.33676  2.02718
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.343182   0.315514   1.088  0.27677
## Density                 -0.446912   0.303835  -1.471  0.14137
```

```
## pH                     -0.017650   0.011778  -1.499  0.13404
## LabelAppeal.L           0.316010   0.038162   8.281  < 2e-16 ***
## LabelAppeal.Q          -0.029858   0.032174  -0.928  0.35343
## LabelAppeal.C           0.023816   0.023367   1.019  0.30813
## LabelAppeal^4           0.033790   0.015625   2.163  0.03062 *
## STARS.L                 1.172443   0.027864  42.078  < 2e-16 ***
## STARS.Q                -0.373749   0.024213 -15.436  < 2e-16 ***
## STARS.C                 0.089414   0.020425   4.378 1.22e-05 ***
## STARS^4                 0.033801   0.016855   2.005  0.04497 *
## log_VolatileAcidity    -0.040948   0.008853  -4.625 3.82e-06 ***
## log_CitricAcid          0.042106   0.007240   5.816 6.35e-09 ***
## log_ResidualSugar       0.010402   0.005860   1.775  0.07593 .
## log_Chlorides          -0.014782   0.007017  -2.106  0.03520 *
## log_FreeSulfurDioxide   0.020206   0.006705   3.013  0.00259 **
## log_TotalSulfurDioxide  0.055276   0.008587   6.437 1.31e-10 ***
## log_Sulphates          -0.015516   0.009176  -1.691  0.09090 .
## inv_AcidIndex           5.579903   0.414106  13.475  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6234 on 6067 degrees of freedom
## Multiple R-squared:  0.4858, Adjusted R-squared:  0.4843
## F-statistic: 318.5 on 18 and 6067 DF,  p-value: < 2.2e-16
```

This emphasizes more predictors than any previous model, but does not exclude any of the most important ones: `STARS`, `LabelAppeal`, and `inv_AcidIndex`. `pH` remains in this model although it is not significant. The R^2_{adj} for this model is slightly lower than that of Model MLR:1 at 48.43%, and the residual standard error is about half the equivalent in Model MLR:1. The interpretability of the model is certainly reduced when considering the ln y response because it is a percentage change in cases of wine given a one unit change in a given predictor.

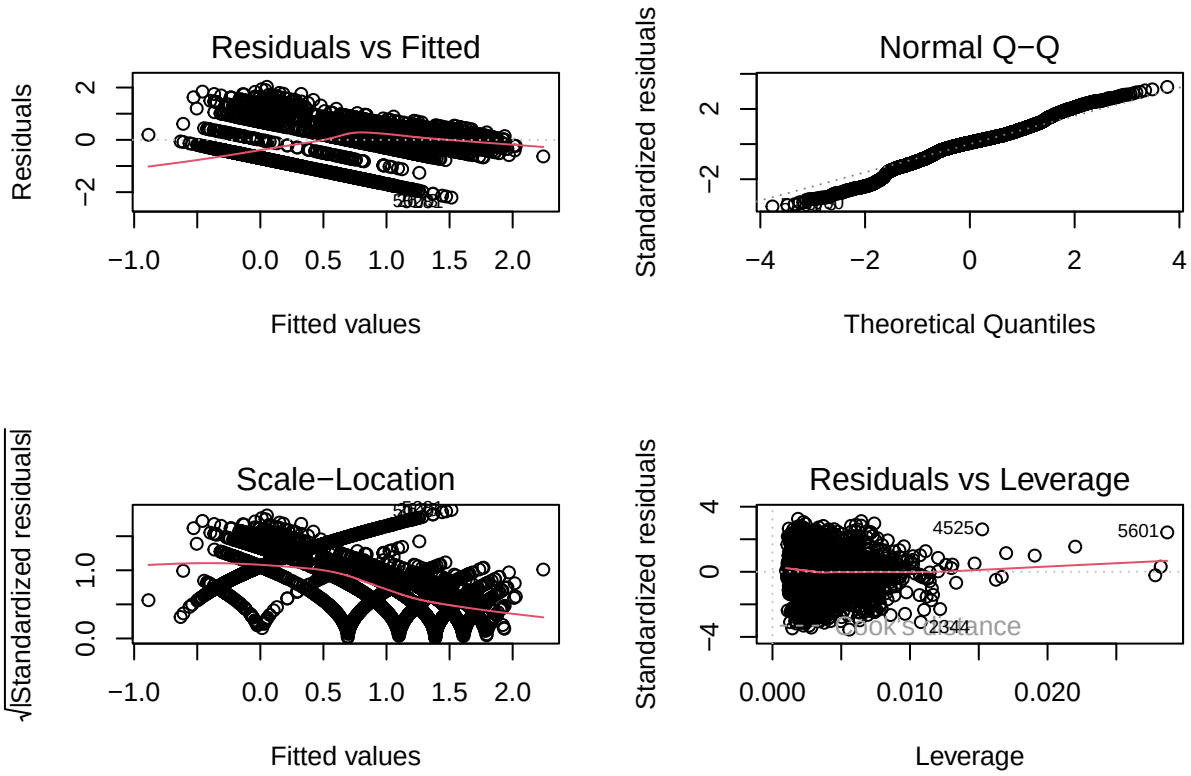
Are there any multicollinearity concerns in this version of the model?

	x
Density	1.006345
pH	1.008322
LabelAppeal.L	1.791631
LabelAppeal.Q	1.268296
LabelAppeal.C	1.696452
LabelAppeal^4	1.258933
STARS.L	1.692453
STARS.Q	1.715681
STARS.C	1.272934
STARS^4	1.083181
log_VolatileAcidity	1.010733
log_CitricAcid	1.014847
log_ResidualSugar	1.002349
log_Chlorides	1.009940
log_FreeSulfurDioxide	1.013291
log_TotalSulfurDioxide	1.017369
log_Sulphates	1.004677
inv_AcidIndex	1.059853

There are no issues with multicollinearity, which is to be expected given the low correlation values across

many of the predictors in the correlation plot from the exploratory section.

Let's review the diagnostic plots to assess if any assumptions are violated:



The fitted residuals are not evenly distributed around zero, although the variance does not appear to change across the range of the predicted values. The QQ plot for the most part conforms with the expected normal distribution of errors except for a left skew which is likely driven off of all the transformed zero values that were in the `TARGET` variable. The scaled version of the fitted versus residuals shows a clear pattern that would violate the assumption that there is a linear relationship between the response and the predictors. It's not clear that it is acceptable to use this model for any inference going forward due to the violations of key assumptions.

```
##          df      AIC
## mod5_mlr 12 20650.64
## mod6_mlr 20 11539.43
```

Despite the fact that Model MLR:2 has the lowest AIC value, it would probably be unwise to use this model given the violation of assumptions we just mentioned.

Model POIS:3 - Zero Inflated Poisson Count Model (ZIP) using Method C

- Ignoring predictors with negative values
- Imputing missing values using MICE

```
##
```

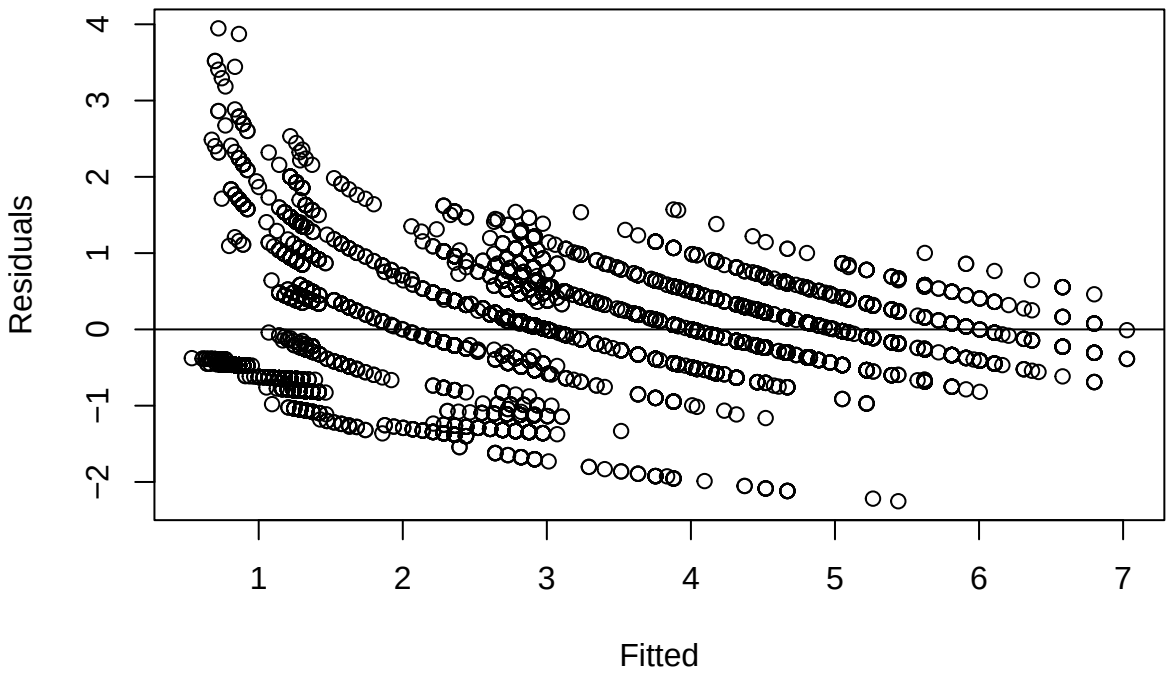
```
## Call:
## pscl::zeroinfl(formula = TARGET ~ STARS + LabelAppeal + AcidIndex | STARS +
##     LabelAppeal, data = method_c_train_df, dist = "poisson")
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -2.25266 -0.46625  0.04954  0.42235  3.94666
##
## Count model coefficients (poisson with log link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.494145   0.045280  32.998  < 2e-16 ***
## STARS.L        0.317696   0.020635  15.396  < 2e-16 ***
## STARS.Q        0.018800   0.017533   1.072   0.2836
## STARS.C       -0.015113   0.015052  -1.004   0.3154
## STARS^4        0.010394   0.012203   0.852   0.3943
## LabelAppeal.L  0.838363   0.035705  23.480  < 2e-16 ***
## LabelAppeal.Q -0.183616   0.029788  -6.164 7.09e-10 ***
## LabelAppeal.C  0.041409   0.020763   1.994   0.0461 *
## LabelAppeal^4 -0.000153   0.012898  -0.012   0.9905
## AcidIndex     -0.032837   0.005799  -5.662 1.49e-08 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -9.71917  217.05517  -0.045   0.9643
## STARS.L       -19.11089  625.30683  -0.031   0.9756
## STARS.Q        -1.89996  528.48060  -0.004   0.9971
## STARS.C         5.18900  444.86251   0.012   0.9907
## STARS^4         4.09296  272.13061   0.015   0.9880
## LabelAppeal.L   2.81390    0.35541   7.917 2.43e-15 ***
## LabelAppeal.Q  -0.72866    0.30124  -2.419   0.0156 *
## LabelAppeal.C   0.23827    0.19107   1.247   0.2124
## LabelAppeal^4  -0.18875    0.09522  -1.982   0.0474 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 25
## Log-likelihood: -1.447e+04 on 19 Df
```

This predictors included in this model were based on the items that were significant to the model rather than including a full one with all of the variables. The logit model used the predictors (`STARS` and `LabelAppeal`) to identify the instances when no cases were expected to be sold based on the expectation that negative customer evaluation and missing expert reviews would prompt distributors to not purchases cases of these wines. Only some levels of both predictors were signnificant in the model.

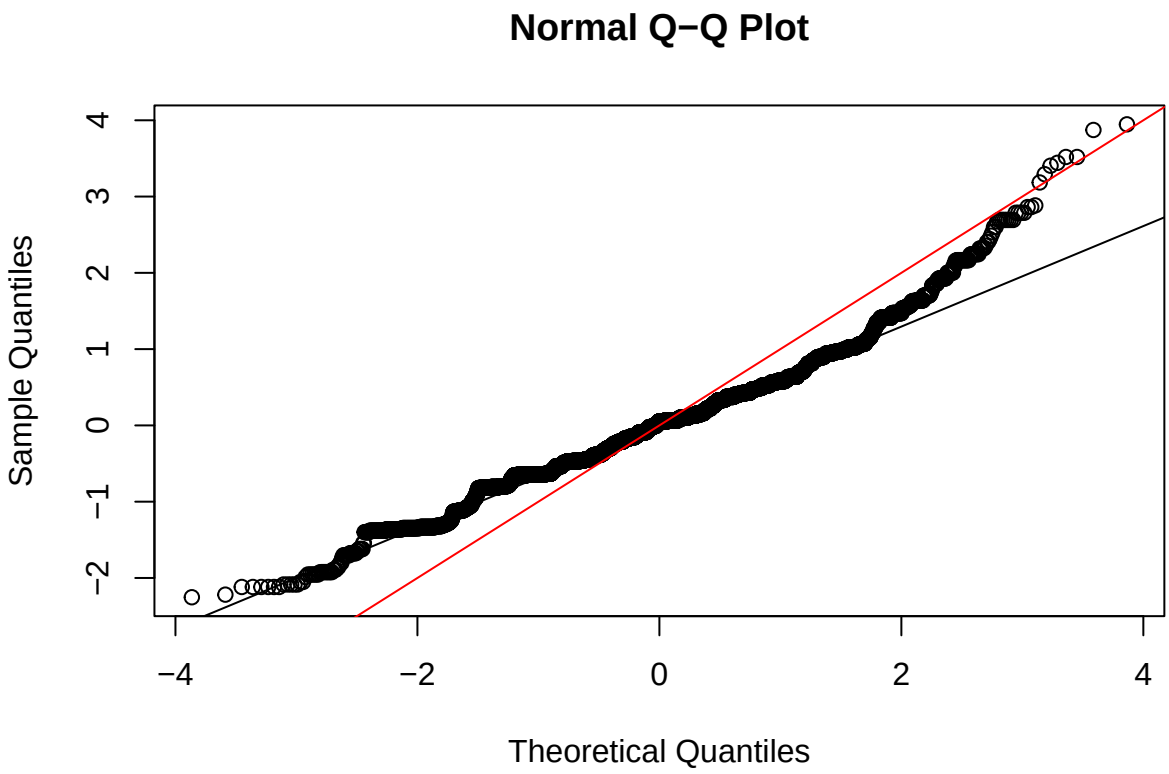
Let's run a Chi-Squared Test to evaluate the goodness of fit:

```
## 'log Lik.' 0 (df=19)
```

A statistically significant result in the Chi-Squared Test would seem to indicate that there is a goodness of fit for the ZIP model. The degrees of freedom is based on the number of predictors in the model excluding the intercept (the null model)



When assessing the fitted vs residuals plot there is some pattern to the residuals and the largest ones still exist for the zero cases examples it would appear. They are still centered around zero.



The residuals primarily follow the QQline for most of the quantiles, but appear to diverge on the higher end and follow a line tracking from zero to one.

```
## [1] 0.3122495
```

There is more substantial underdispersion captured in this model than the other poisson regression one and we would like to avoid any under/over dispersion that exists.

Let's review the Akaike Information Criterion for this ZIP model using method C:

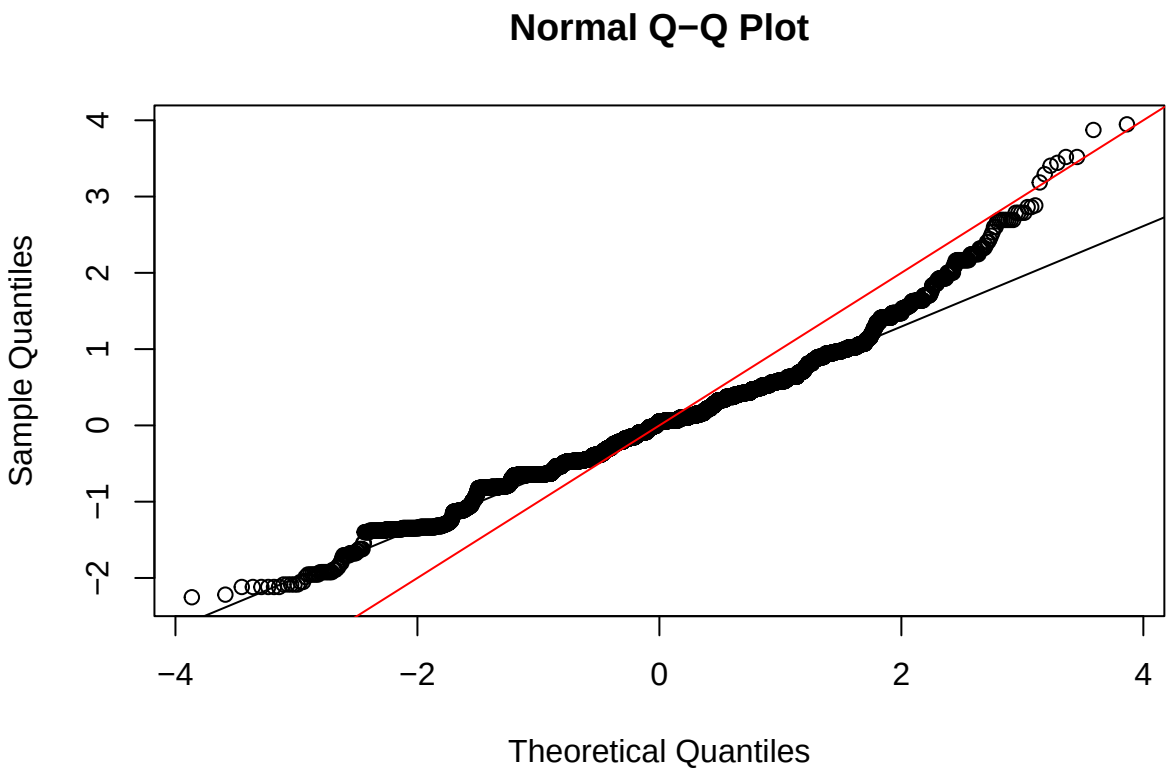
```
## [1] 28972.99
```

Model NB:3 - Zero Inflated Negative Binomial (ZINB) using Method C Let's try to evaluate the data based on a zero inflated negative binomial:

```
##
## Call:
## pscl::zeroinfl(formula = TARGET ~ STARS + LabelAppeal + AcidIndex | STARS +
##     LabelAppeal, data = method_c_train_df, dist = "negbin")
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -2.25266 -0.46626  0.04954  0.42235  3.94665
##
## Count model coefficients (negbin with log link):
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    1.494145   0.045280  32.998  < 2e-16 ***
## STARS.L        0.317696   0.020635  15.396  < 2e-16 ***
## STARS.Q        0.018800   0.017533   1.072   0.2836
## STARS.C       -0.015113   0.015052  -1.004   0.3154
## STARS^4        0.010394   0.012203   0.852   0.3943
## LabelAppeal.L  0.838363   0.035705  23.480  < 2e-16 ***
## LabelAppeal.Q -0.183616   0.029788  -6.164 7.09e-10 ***
## LabelAppeal.C  0.041409   0.020763   1.994   0.0461 *
## LabelAppeal^4 -0.000153   0.012898  -0.012   0.9905
## AcidIndex     -0.032837   0.005799  -5.662 1.49e-08 ***
## Log(theta)    17.305698   1.727092  10.020  < 2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -9.71917  217.05621  -0.045   0.9643
## STARS.L       -19.11089  625.31012  -0.031   0.9756
## STARS.Q        -1.89995  528.48330  -0.004   0.9971
## STARS.C         5.18900  444.86410   0.012   0.9907
## STARS^4         4.09295  272.13135   0.015   0.9880
## LabelAppeal.L   2.81389    0.35541   7.917 2.43e-15 ***
## LabelAppeal.Q  -0.72865    0.30123  -2.419   0.0156 *
## LabelAppeal.C   0.23826    0.19106   1.247   0.2124
## LabelAppeal^4  -0.18875    0.09522  -1.982   0.0474 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 32792091.6805
## Number of iterations in BFGS optimization: 25
## Log-likelihood: -1.447e+04 on 20 Df
```

We tried to be consistent with the zero inflated model for the negative binomial using this method and used the same independent variables as what was identified as significant in zero inflated poisson model using method C. From a review of the summary, the residuals appear to be distributed around zero although there is a slight positive skew as compared to the negative. `STARS` appears to be less significant in the logistic model although only the lowest level is significant for the count model.



The QQplot aligns more closely with the distribution line; however, similar to the other distributions there are a lot of points on the right tail that are skewed.

Let's calculate the dispersion from the model:

```
## [1] 0.3231091
```

The dispersion statistic is in line with the other zero inflated model, which appears to be a knock on both models.

```
##           df      AIC
## mod7_zip  19 28972.99
## mod8_zinb 20 28974.99
```

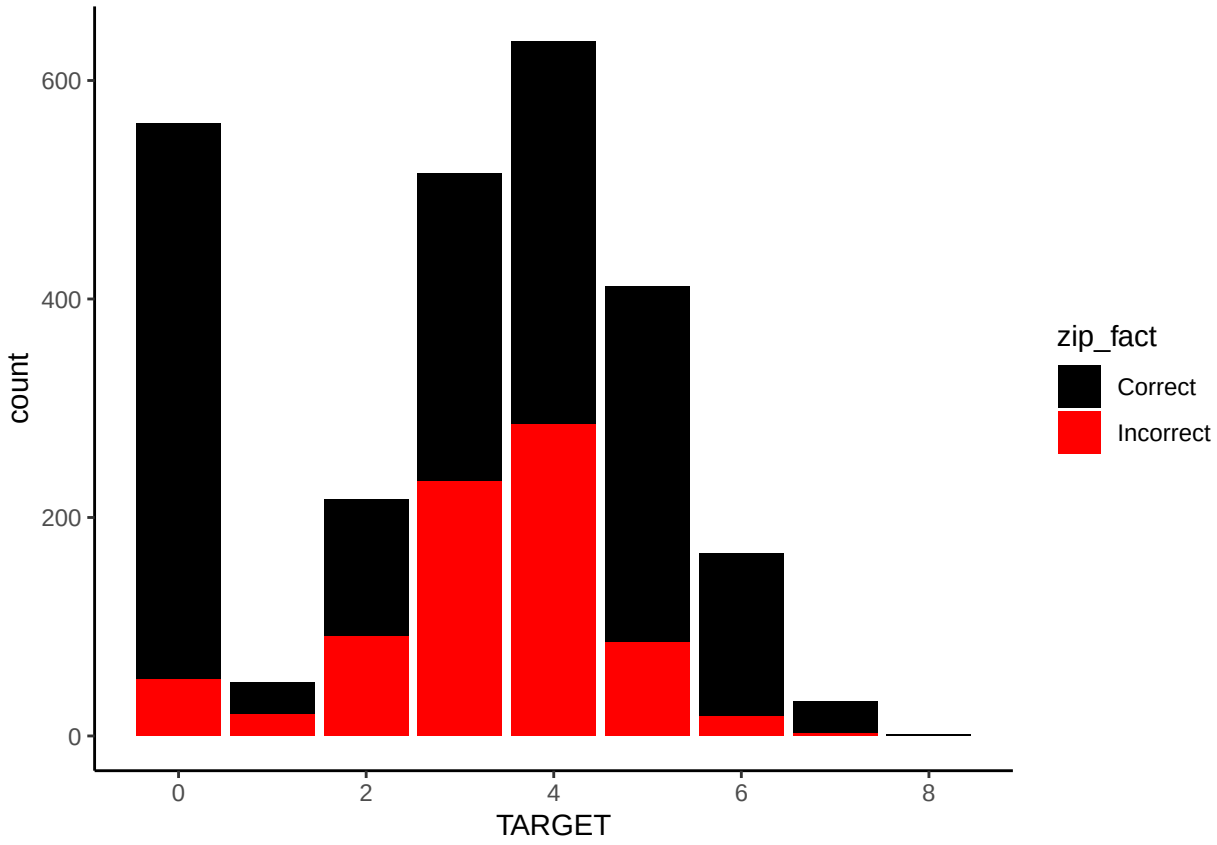
Both AIC values for these models are much higher than the other methods. It appears as though the imputation that was done may not be valid with the negative values present in so many potential predictors.

Select Models Based on the fact that multiple models that were tested in the prior section had somewhat serious flaws, we are going to evaluate the zero inflated models against the hold out set in order to make a final selection of the "best" model. It still appears as though there are some challenges in accurately identifying all of the zero cases scenarios.

The process for determining the correct prediction is to first determine if the logistic model classified zero values. After rounding the predicted values to the closest discrete value for both variations of the model, we can see that Model POIS:2 (ZIP) and Model NB:2 (ZINB) perform very differently on the holdout set.

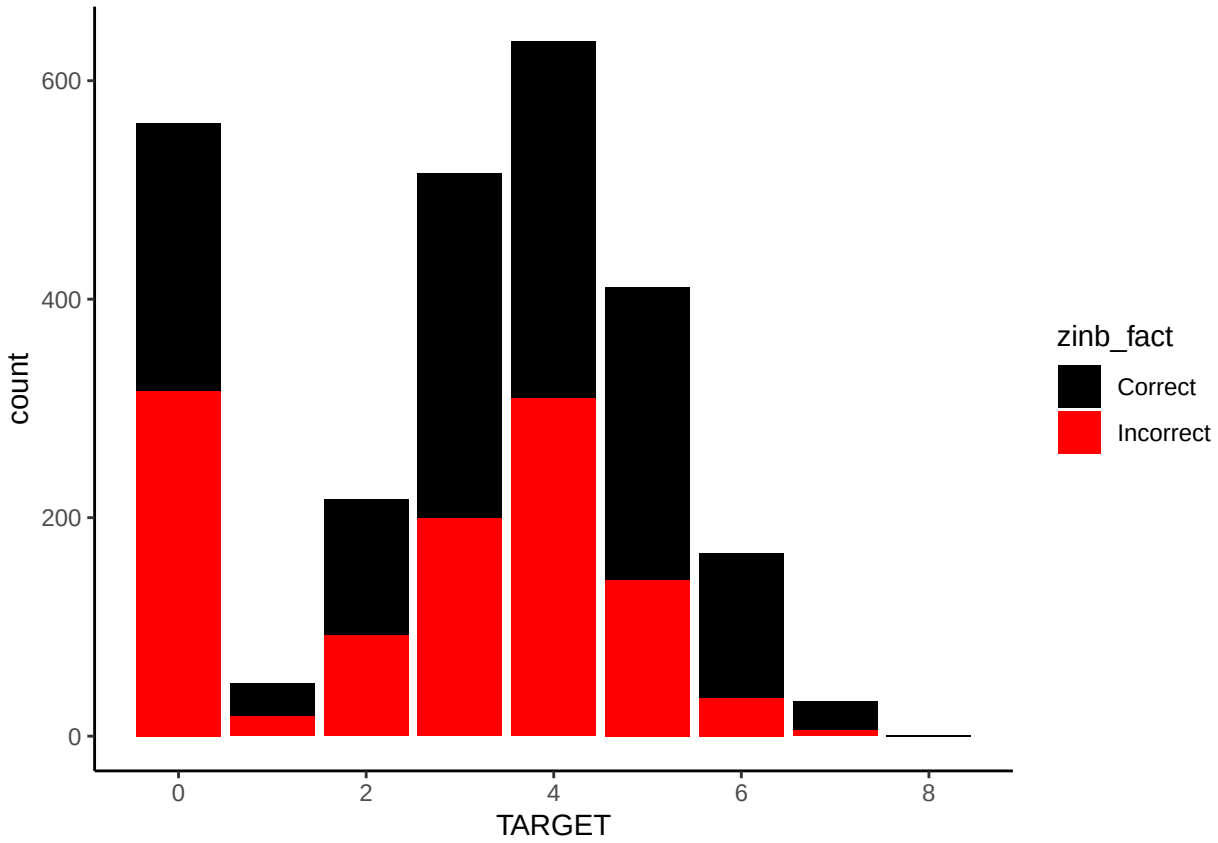
Model POIS:2 (ZIP) Accuracy:

zip_match	cnt
0	1802
1	787



Model NB:2 (ZINB) Accuracy:

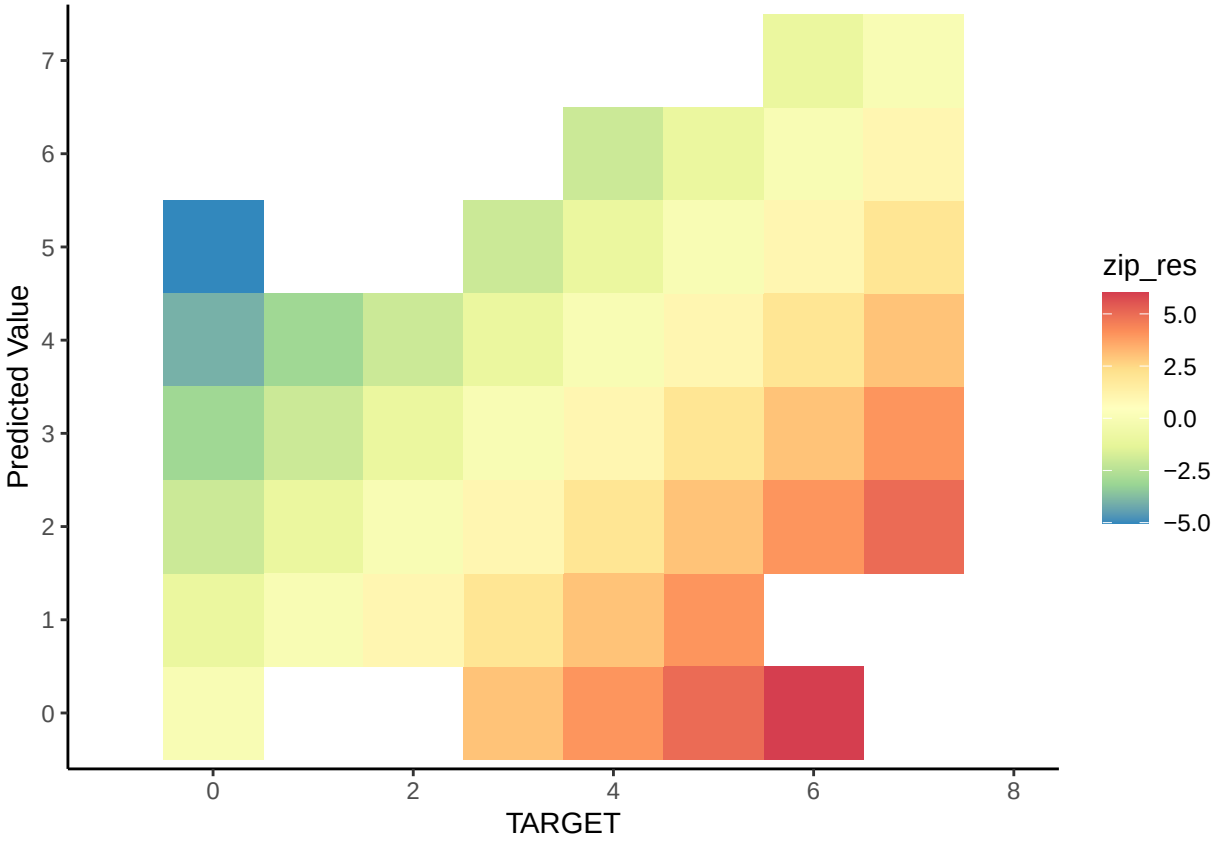
zinb_match	cnt
0	1472
1	1117



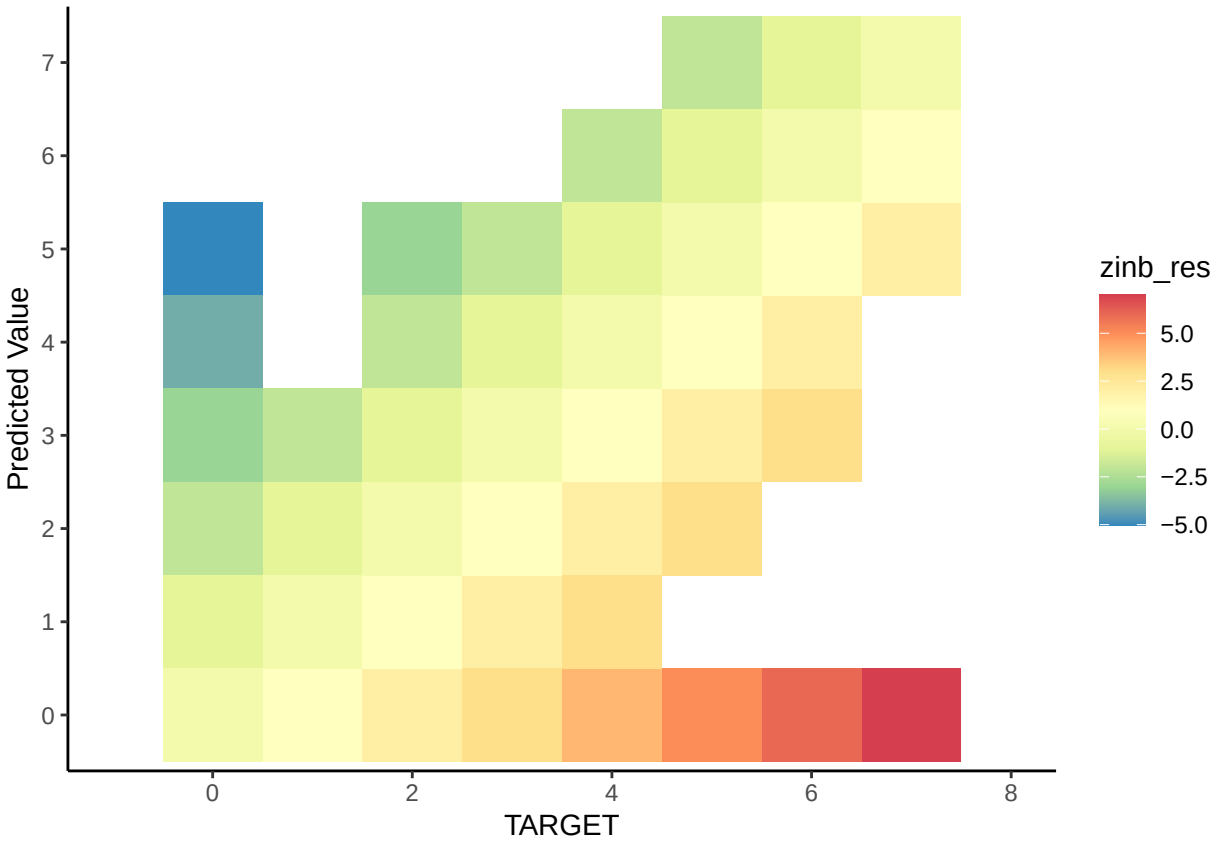
Model POIS:2 (ZIP) has a 30% accuracy rate, whereas Model NB:2 (ZINB) has a 43% accuracy rate. So Model NB:2 (ZINB) is the better zero predictor.

Let's review a comparison of the Target to the predicted value.

```
## Warning: Removed 1 rows containing missing values (‘geom_tile()‘).
```




```
## Warning: Removed 1 rows containing missing values (‘geom_tile()‘).
```

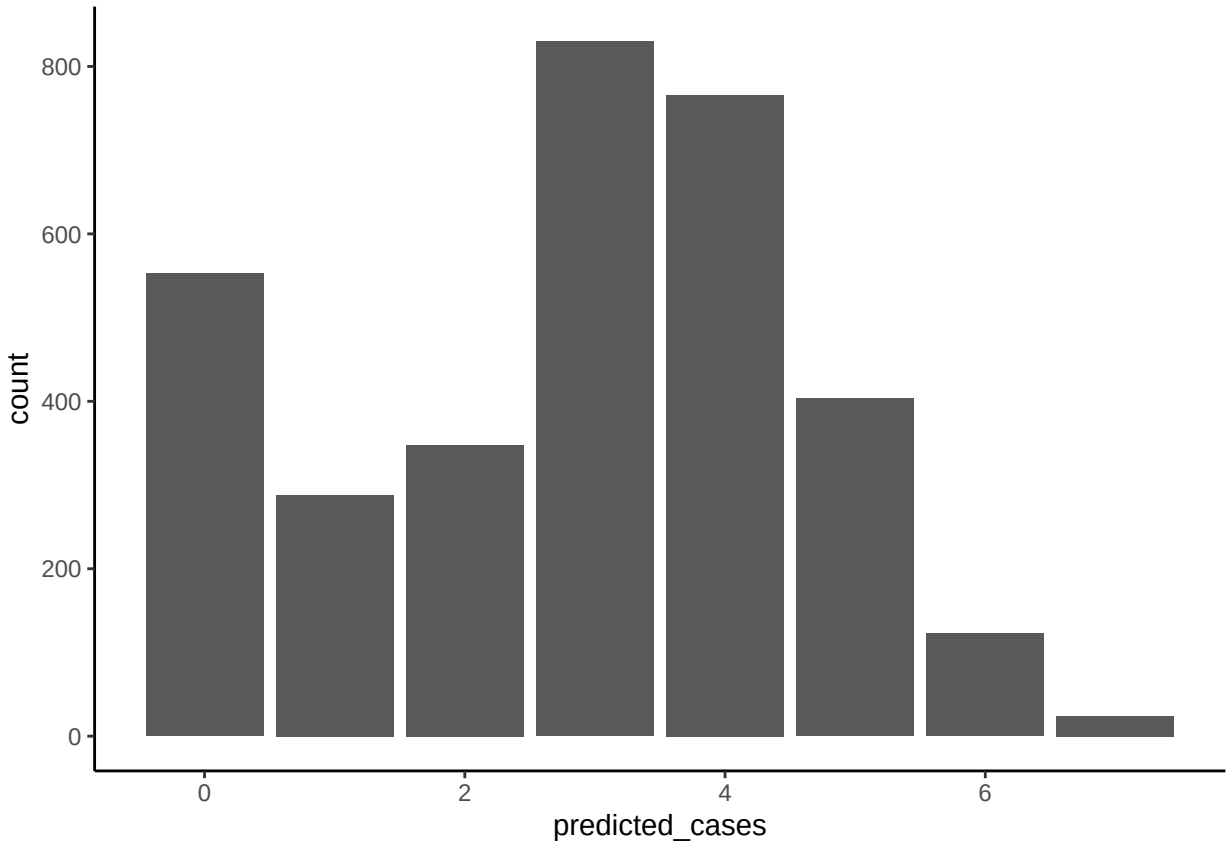


Both models struggled to correctly identify the skewed ends of the target and were most divergent for zero values and the high end number of cases sold.

Based on all of the above criteria, it would appear that Model NB:2 (ZINB) edges out Model POIS:2 (ZIP) given its slightly lower AIC metric and better accuracy at predicting zeroes. All the other criteria stacks up fairly evenly. We select Model NB:2 (ZINB) for evaluation.

Evaluation:

We make predictions on the evaluation dataframe using our selected model, Model NB:2 (ZINB).



The final predicted distribution appears to expect less zero values than what was in the training dataset; however, it is in line with prior data and appears to follow the general shape of the other data available on wines.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
library(tidyverse)
library(DataExplorer)
library(knitr)
library(mice)
library(cowplot)
library(scales)
library(MASS)
library(glue)
library(corrplot)
library(naniar)
library(car)
library(finalfit)
library(pscl)
library(faraway)
library(caret)
library(ggcorrplot)
library(RColorBrewer)
```

```r
cur_theme <- theme_set(theme_classic())

git_link = 'https://raw.githubusercontent.com/andrewbowen19/businessAnalyticsDataMiningDATA621/main/data
eval_link = 'https://raw.githubusercontent.com/andrewbowen19/businessAnalyticsDataMiningDATA621/main/da
input_df <- read_csv(git_link,show_col_types = FALSE)
eval_df <- read_csv(eval_link,show_col_types=FALSE)
summary(input_df)
excl_neg <- c('INDEX','TARGET','pH','STARS', 'LabelAppeal')
neg_cnts <- sapply(input_df |> dplyr::select(-all_of(excl_neg)), function(x) sum(!is.na(x) & x<0))
neg_cnts <- as.data.frame(neg_cnts) |>
    rownames_to_column() |>
    filter(neg_cnts > 0) |>
    mutate(validity = "Invalid")
cols <- c("Predictor", "Negative Values Count", "Validity")
colnames(neg_cnts) <- cols
knitr::kable(neg_cnts, format = "simple")
excl_eval_neg <- c('IN','TARGET', 'LabelAppeal')
neg_eval_cnts <- sapply(eval_df |> dplyr::select(-all_of(excl_eval_neg)), function(x) sum(!is.na(x) & x<
neg_eval_cnts <- as.data.frame(neg_eval_cnts) |>
    rownames_to_column() |>
    filter(neg_eval_cnts > 0) |>
    mutate(validity = "Invalid")
cols <- c("Predictor", "Negative Values Count", "Validity")
colnames(neg_eval_cnts) <- cols
knitr::kable(neg_eval_cnts, format = "simple")
p1 <- plot_missing(input_df, missing_only = TRUE,
                   ggtheme = theme_classic(), title = "Missing Values")


#input_df <- input_df |> dplyr::select(-INDEX)

numeric_train <- input_df[,sapply(input_df, is.numeric)]
par(mfrow=c(4,4))
par(mai=c(.3,.3,.3,.3))
variables <- names(numeric_train)
for (i in 1:(length(variables))) {
  hist(numeric_train[[variables[i]]], main = variables[i], col = "lightblue")
}

gather_df <- input_df %>% dplyr::select(-INDEX) %>%
  gather(key = 'variable', value = 'value')
ggplot(gather_df, aes(variable, value)) +
  geom_boxplot() +
  facet_wrap(. ~variable, scales='free', ncol=4)

ggplot(input_df, aes(x = TARGET, y = after_stat(density))) +
  geom_histogram(binwidth = 1, fill = 'lightblue', color = 'black', alpha = 0.7,
                 aes(color = "Histogram")) +
  geom_line(aes(y = dpois(TARGET, mean(TARGET), log = FALSE), color = "Poisson"),
            linewidth = 1) +
  geom_line(aes(y = dnbinom(x = TARGET, size = 1, prob = 0.2), color = "Negative Binomial"),
            linewidth = 1) +
  geom_line(aes(y = dnorm(x = TARGET, mean = mean(TARGET), sd = sd(TARGET), log = FALSE), color = "Norma
```

```
  labs(title = 'Target Histogram and Distribution Overlay',
       x = 'TARGET',
       y = 'Density') +
  scale_color_manual(values = c( "red", "blue", "orange"),
                     labels = c("Negative Binomial","Normal","Poisson")) +
  guides(color = guide_legend(title = "Distributions")) +
  theme_minimal()


print(glue("The response mean: {mean(input_df$TARGET)} and variance: {var(input_df$TARGET)}"))
m <- cor(na.omit(input_df |> dplyr::select(-INDEX)))
m <- as.data.frame(m) |>
    mutate(across(everything(), function(x){replace(x, which(abs(x) < 0.05),
                                                    NA)}))
m <- as.matrix(m)
palette <- brewer.pal(n = 7, name = "RdBu")[c(1, 4, 7)]
ggcorrplot(m, show.diag = FALSE, type = "lower", lab = TRUE, lab_size = 2.5,
               tl.cex = 8, tl.srt = 90,
               colors = palette, outline.color = "white")

nonzero_df <- na.omit(input_df |> dplyr::select(-INDEX)) |> filter(TARGET!=0)
m <- cor(nonzero_df)
m <- as.data.frame(m) |>
    mutate(across(everything(), function(x){replace(x, which(abs(x) < 0.05),
                                                    NA)}))
m <- as.matrix(m)
ggcorrplot(m, show.diag = FALSE, type = "lower", lab = TRUE, lab_size = 2.5,
               tl.cex = 8, tl.srt = 90,
               colors = palette, outline.color = "white")

mod_df <- input_df |>
    dplyr::select(-INDEX) |>
    mutate(STARS=factor(STARS, levels = c(NA, 1, 2, 3, 4),
                        exclude = "",
                        ordered = TRUE),
           LabelAppeal=factor(LabelAppeal, levels = c(-2, -1, 0, 1, 2),
                        ordered = TRUE))
mod_df_a <- mod_df
set.seed(19)
rows <- sample(nrow(mod_df_a))
sample <- sample(c(TRUE, FALSE), nrow(mod_df_a), replace=TRUE,
                 prob=c(0.7,0.3))
method_a_train_df <- mod_df_a[sample, ]
method_a_test_df <- mod_df_a[!sample, ]
remove_rows_na <- function(df){
    na_row_sums <- rowSums(is.na(df))
    row_has_na <- ifelse(na_row_sums > 0, TRUE, FALSE)
    copy <- df[!row_has_na, ]
    copy
}
method_a_train_df <- remove_rows_na(method_a_train_df)
method_a_test_df <- remove_rows_na(method_a_test_df)
```

```
mod_df_b <- mod_df
sel <- c("STARS", "LabelAppeal")
cols <- colnames(mod_df_b |> dplyr::select(-all_of(sel)))
mod_df_b <- mod_df_b |>
    mutate(across(all_of(cols), function(x){replace(x, which(x < 0),
                                                     abs(x[which(x < 0)]))}))

method_b_train_df <- mod_df_b[sample, ]
method_b_test_df <- mod_df_b[!sample, ]
method_b_train_df <- remove_rows_na(method_b_train_df)
method_b_test_df <- remove_rows_na(method_b_test_df)

sel <- c("TARGET", "STARS", "LabelAppeal")
cols <- colnames(method_b_train_df |> dplyr::select(-all_of(sel)))
for (i in 1:(length(cols))){
    #Add a small constant to columns with any 0 values
    if (sum(method_b_train_df[[cols[i]]] == 0) > 0){
        method_b_train_df[[cols[i]]] <-
            method_b_train_df[[cols[i]]] + 0.001
    }
}
for (i in 1:(length(cols))){
    if (i == 1){
        lambdas <- c()
    }
    bc <- boxcox(lm(method_b_train_df[[cols[i]]] ~ 1),
                 lambda = seq(-2, 2, length.out = 81),
                 plotit = FALSE)
    lambda <- bc$x[which.max(bc$y)]
    lambdas <- append(lambdas, lambda)
}
lambdas <- as.data.frame(cbind(cols, lambdas))
adj <- c("square root", "log", "log", "log", "log", "log", "log",
         "none", "none", "log", "none", "inverse")
lambdas <- cbind(lambdas, adj)
cols <- c("Variable", "Ideal Lambda Proposed by Box-Cox", "Reasonable Alternative Transformation")
colnames(lambdas) <- cols
kable(lambdas, format = "simple")

remove <- c("FixedAcidity", "VolatileAcidity", "CitricAcid", "ResidualSugar",
            "Chlorides", "FreeSulfurDioxide", "TotalSulfurDioxide", "Sulphates",
            "AcidIndex")
method_b_train_df <- method_b_train_df |>
    mutate(sqrt_FixedAcidity = FixedAcidity^0.5,
           log_VolatileAcidity = log(VolatileAcidity),
           log_CitricAcid = log(CitricAcid),
           log_ResidualSugar = log(ResidualSugar),
           log_Chlorides = log(Chlorides),
           log_FreeSulfurDioxide = log(FreeSulfurDioxide),
           log_TotalSulfurDioxide = log(TotalSulfurDioxide),
           log_Sulphates = log(Sulphates),
           inv_AcidIndex = AcidIndex^-1) |>
    dplyr::select(-all_of(remove))
```

```
sel <- c("TARGET", "STARS", "LabelAppeal")
cols <- colnames(method_b_test_df |> dplyr::select(-all_of(sel)))
for (i in 1:(length(cols))){
    #Add a small constant to columns with any 0 values
    if (sum(method_b_test_df[[cols[i]]] == 0) > 0){
        method_b_test_df[[cols[i]]] <-
            method_b_test_df[[cols[i]]] + 0.001
    }
}
method_b_test_df <- method_b_test_df |>
    mutate(sqrt_FixedAcidity = FixedAcidity^0.5,
           log_VolatileAcidity = log(VolatileAcidity),
           log_CitricAcid = log(CitricAcid),
           log_ResidualSugar = log(ResidualSugar),
           log_Chlorides = log(Chlorides),
           log_FreeSulfurDioxide = log(FreeSulfurDioxide),
           log_TotalSulfurDioxide = log(TotalSulfurDioxide),
           log_Sulphates = log(Sulphates),
           inv_AcidIndex = AcidIndex^-1) |>
    dplyr::select(-all_of(remove))

palette <- brewer.pal(n = 7, name = "RdBu")[c(1, 4, 7)]
r <- model.matrix(~0+., data = method_b_train_df) |>
    cor(use = "pairwise.complete.obs")
is.na(r) <- abs(r) < 0.1
r |>
    ggcorrplot(show.diag = FALSE, type = "lower", lab = TRUE, lab_size = 2.5,
               tl.cex = 8, tl.srt = 90,
               colors = palette, outline.color = "white")
method_b_train_df <- method_b_train_df |> mutate(TARGET.TF=ifelse(TARGET==0,TARGET+0.001,TARGET))
target_bc <- boxcox(lm(method_b_train_df$TARGET.TF  ~ 1))
target_lambda <- target_bc$x[which.max(target_bc$y)]
target_lambda
method_b_train_df <- method_b_train_df |>
    mutate(TARGET.TF=log(ifelse(TARGET==0,0.5,TARGET)))
method_b_test_df <- method_b_test_df |>
    mutate(TARGET.TF=log(ifelse(TARGET==0,0.5,TARGET)))
hist(method_b_train_df$TARGET.TF)
littles_test <- mod_df |>
    mcar_test()
knitr::kable(littles_test, format = "simple")

na_cols <- c("pH", "Density", "Chlorides", "FreeSulfurDioxide",'Alcohol','TotalSulfurDioxide','Sulphates
na_col_review <- mod_df |>
    dplyr::select(all_of(na_cols)) |>
    missing_plot()
na_col_review


na_freq <- mod_df |> dplyr::select(all_of(na_cols)) |> mutate(row_index=1:nrow(mod_df)) |> pivot_longer

ggplot(na_freq,aes(x=num_na)) +
    geom_bar() +
```

```
    labs(x='Number of NA columns',y='Number of Rows',title='Distribution of NA Values across columns')

print(glue("Only {pull(na_freq |> filter(num_na>1)|>summarise(total=n()))} rows have more than 1 column

set.seed(19)
rows <- sample(nrow(mod_df))
sample <- sample(c(TRUE, FALSE), nrow(mod_df), replace=TRUE,
                 prob=c(0.7,0.3))
method_c_train_df <- mod_df[sample, ]
method_c_test_df <- mod_df[!sample, ]


df_names <- colnames(method_c_train_df)
non_na_cols <- df_names[!df_names %in% na_cols]

if (file.exists("imputed_wine_test_df.csv") & file.exists("imputed_wine_train_df.csv")){
    train_df_imputed_input <- read.csv("imputed_wine_train_df.csv", na.strings = "")
    test_df_imputed_input <- read.csv("imputed_wine_test_df.csv", na.strings = "")
    method_c_train_df_imp <- train_df_imputed_input |> mutate(STARS=factor(STARS, levels = c(NA, 1, 2, 3
                        exclude = "",
                        ordered = TRUE),
           LabelAppeal=factor(LabelAppeal, levels = c(-2, -1, 0, 1, 2),
                        ordered = TRUE))
    method_c_test_df_imp <- test_df_imputed_input |> mutate(STARS=factor(STARS, levels = c(NA, 1, 2, 3,
                        exclude = "",
                        ordered = TRUE),
           LabelAppeal=factor(LabelAppeal, levels = c(-2, -1, 0, 1, 2),
                        ordered = TRUE))

}else{
        #Train Data Imputation First
    init = mice(train_df, maxit=0)
    meth = init$method
    predM = init$predictorMatrix


    meth[non_na_cols] = ""

    meth[c("pH")] = "pmm"
    meth[c("ResidualSugar")] = "pmm"
    meth[c("Chlorides")] = "pmm"
    meth[c("FreeSulfurDioxide")] = "pmm"
    meth[c("Alcohol")] = "pmm"
    meth[c("TotalSulfurDioxide")] = "pmm"
    meth[c("Sulphates")] = "pmm"

    imputed_train = mice(train_df, method=meth, predictorMatrix=predM, m=5,
                    printFlag = FALSE)
    train_df_imputed <- complete(imputed_train)
    write.csv(train_df_imputed, "imputed_wine_train_df.csv", row.names = FALSE,
              fileEncoding = "UTF-8")

    #Repeat for test_df
```

```
    init = mice(test_df, maxit=0)
    meth = init$method
    predM = init$predictorMatrix
    meth[non_na_cols] = ""
    meth[c("pH")] = "pmm"
    meth[c("ResidualSugar")] = "pmm"
    meth[c("Chlorides")] = "pmm"
    meth[c("FreeSulfurDioxide")] = "pmm"
    meth[c("Alcohol")] = "pmm"
    meth[c("TotalSulfurDioxide")] = "pmm"
    meth[c("Sulphates")] = "pmm"
    imputed_test = mice(test_df, method=meth, predictorMatrix=predM, m=5,
                    printFlag = FALSE)
    imputed_test_df <- complete(imputed_test)
    write.csv(imputed_test_df, "imputed_wine_test_df.csv", row.names = FALSE,
              fileEncoding = "UTF-8")

}

x <- sapply(train_df_imputed_input, function(x) sum(is.na(x)))
y <- sapply(test_df_imputed_input, function(x) sum(is.na(x)))
sum(x, y) == 0
#need to fix for spacing
sub_vars <- c('Chlorides','pH','Sulphates')
other_nas <- c('ResidualSugar','FreeSulfurDioxide','Alcohol','TotalSulfurDioxide')
impute_train_input <- method_c_train_df_imp |>
    dplyr::select(all_of(sub_vars)) |>
    mutate(Set = "Train")
impute_test_input <- method_c_test_df_imp |>
    dplyr::select(all_of(sub_vars)) |>
    mutate(Set = "Test")
impute_both <- impute_train_input |>
    bind_rows(impute_test_input)
impute_pivot <- impute_both |>
    pivot_longer(!Set, names_to = "Variable", values_to = "Value")
impute_plot <- impute_pivot |>
    ggplot(aes(x = Value)) +
    geom_density(fill = "lightblue", color = "black") +
    labs(y = "Density") +
    facet_grid(rows = vars(Set), cols = vars(Variable),
               switch = "y", scales = "free_x")
impute_plot



impute_train_input2 <- method_c_train_df_imp |>
    dplyr::select(all_of(other_nas)) |>
    mutate(Set = "Train")
impute_test_input2 <- method_c_test_df_imp |>
    dplyr::select(all_of(other_nas)) |>
    mutate(Set = "Test")
impute_both2 <- impute_train_input2 |>
    bind_rows(impute_test_input2)
```

```
impute_pivot2 <- impute_both2 |>
    pivot_longer(!Set, names_to = "Variable", values_to = "Value")
impute_plot2 <- impute_pivot2 |>
    ggplot(aes(x = Value)) +
    geom_density(fill = "lightblue", color = "black") +
    labs(y = "Density") +
    facet_grid(rows = vars(Set), cols = vars(Variable),
               switch = "y", scales = "free_x")
impute_plot2

mod1_pois <- glm(TARGET ~ STARS + LabelAppeal + pH + Density + AcidIndex, family = 'poisson', data = me
mod1_pois <- stepAIC(mod1_pois, trace = 0)
summary(mod1_pois)

with(mod1_pois, cbind(res.deviance = deviance, df = df.residual,
  p = pchisq(deviance, df.residual, lower.tail=FALSE)))
e_y <- predict(mod1_pois,type="response")
plot(log(e_y+1),log(method_a_train_df$TARGET+1),xlim=c(0,4),xlab='Log Scaled Estimated Cases',ylab='Log
title(main='Comparing Estimated vs Observed number of cases of wine sold')
faraway::halfnorm(residuals(mod1_pois))

car::influencePlot(mod1_pois)
res <- residuals(mod1_pois, type="deviance")
#abline(h=0, lty=2)
qqnorm(res)
qqline(res)
## Check for over/underdispersion in the model
E2 <- resid(mod1_pois, type = "pearson")
N  <- nrow(method_a_train_df)
p  <- length(coef(mod1_pois))
sum(E2^2) / (N - p)
mod2_zip <- pscl::zeroinfl(formula=TARGET ~ . - TARGET.TF - log_ResidualSugar - sqrt_FixedAcidity - pH
summary(mod2_zip)
m2null <- update(mod2_zip, . ~ 1)
pchisq(2 * (logLik(mod2_zip) - logLik(m2null)), df = 13, lower.tail = FALSE)
plot(residuals(mod2_zip) ~
      fitted(mod2_zip),xlab="Fitted",ylab="Residuals")
abline(h=0)
res_mod2 <- residuals(mod2_zip,'pearson')
qqnorm(res_mod2)
qqline(res_mod2)
abline(0, 1, col = "red")
E2_zip2 <- resid(mod2_zip, type = "pearson")
N_zip2 <- nrow(input_df)
p_zip2 <- length(coef(mod2_zip)) # '+1' is for variance parameter in NB
sum(E2_zip2^2) / (N_zip2 - p_zip2)
AIC(mod1_pois,mod2_zip)
mod3_nb <- glm.nb(TARGET ~ STARS + LabelAppeal + AcidIndex,data = method_a_train_df)
# summary of results
summary(mod3_nb)

pchisq(2 * (logLik(mod3_nb) - logLik(mod1_pois)), df = 1, lower.tail = FALSE)
plot(residuals(mod3_nb) ~
```

```
      fitted(mod3_nb),xlab="Fitted",ylab="Residuals")
abline(h=0)
res_mod_nb <- residuals(mod3_nb,'pearson')
qqnorm(res_mod_nb)
qqline(res_mod_nb)

E2_mod3 <- resid(mod3_nb, type = "pearson")
N  <- nrow(input_df)
p_mod3  <- length(coef(mod3_nb)) + 1 # added for variance parameter
sum(E2_mod3^2) / (N - p_mod3)
mod4_zinb <- pscl::zeroinfl(formula=TARGET ~ STARS + LabelAppeal + AcidIndex | STARS + LabelAppeal,data=

summary(mod4_zinb)
res_mod4 <- residuals(mod4_zinb,'pearson')
qqnorm(res_mod4)
qqline(res_mod4)

abline(0, 1, col = "red")
E2_mod4 <- resid(mod4_zinb, type = "pearson")
N  <- nrow(input_df)
p_mod4  <- length(coef(mod4_zinb)) + 1 # added for variance parameter
sum(E2_mod4^2) / (N - p_mod4)
aic_compare2<- AIC(mod1_pois,mod2_zip,mod3_nb,mod4_zinb)
knitr::kable(aic_compare2, format = "simple")
mod5_mlr <- lm(TARGET ~ STARS + LabelAppeal + pH + Density + AcidIndex,data=method_a_train_df)
mod5_mlr <- stepAIC(mod5_mlr, trace = 0)
summary(mod5_mlr)
knitr::kable(vif(mod5_mlr), format = "simple")
par(mfrow=c(2,2))
plot(mod5_mlr)
mod6_mlr <- lm(TARGET.TF ~ . - TARGET,data=method_b_train_df)
mod6_mlr <- stepAIC(mod6_mlr, trace = 0)
summary(mod6_mlr)
knitr::kable(vif(mod6_mlr), format = "simple")
par(mfrow=c(2,2))
plot(mod6_mlr)
AIC(mod5_mlr,mod6_mlr)
mod7_zip <- pscl::zeroinfl(formula=TARGET ~ STARS + LabelAppeal + AcidIndex |STARS + LabelAppeal,data=m


summary(mod7_zip)
m7null <- update(mod7_zip, . ~ 1)

pchisq(2 * (logLik(mod7_zip) - logLik(m7null)), df = 7, lower.tail = FALSE)
plot(residuals(mod7_zip) ~
      fitted(mod7_zip),xlab="Fitted",ylab="Residuals")
abline(h=0)
res_mod7 <- residuals(mod7_zip,'pearson')
qqnorm(res_mod7)
qqline(res_mod7)

abline(0, 1, col = "red")
E2_zip7 <- resid(mod2_zip, type = "pearson")
```

```
N_zip7 <- nrow(input_df)
p_zip7 <- length(coef(mod7_zip)) # '+1' is for variance parameter in NB
sum(E2_zip7^2) / (N_zip7 - p_zip7)
AIC(mod7_zip)
mod8_zinb <- pscl::zeroinfl(formula=TARGET ~ STARS + LabelAppeal + AcidIndex| STARS + LabelAppeal,data=

summary(mod8_zinb)
res_mod8 <- residuals(mod8_zinb,'pearson')
qqnorm(res_mod8)
qqline(res_mod8)

abline(0, 1, col = "red")
E2_mod8 <- resid(mod8_zinb, type = "pearson")
N  <- nrow(input_df)
p_mod8  <- length(coef(mod8_zinb)) + 1 # added for variance parameter
sum(E2_mod8^2) / (N - p_mod8)
AIC(mod7_zip,mod8_zinb)
zip_pred <- predict(mod2_zip, method_b_test_df,
                    type='response')
zip_zero <- predict(mod2_zip,method_b_test_df,
                    type='zero')
zinb_pred <- predict(mod4_zinb,method_a_test_df,type='response')
zinb_zero <- predict(mod4_zinb,method_a_test_df,type='zero')
holdout_zi1 <- as.data.frame(cbind(method_b_test_df,zip_pred,zip_zero)) |>
    dplyr::select(c("TARGET","zip_pred","zip_zero")) |>
    mutate(zip_pred_rnd=ifelse(zip_zero>=0.5,0,round(zip_pred,0)),
           zip_match=ifelse(zip_zero>0.5,ifelse(TARGET==0,1,0),ifelse(TARGET==zip_pred_rnd,1,0)),
           zip_res=TARGET-zip_pred_rnd,zip_fact=as.factor(zip_match),
           is_zip_zero=ifelse(zip_zero>0.5,'0',as.character(zip_pred_rnd)),
           is_zero=ifelse(TARGET==0,'Y','N'))
holdout_zi2 <- as.data.frame(cbind(method_a_test_df, zinb_pred, zinb_zero)) |>
    dplyr::select(c("TARGET","zinb_pred","zinb_zero")) |>
    mutate(zinb_pred_rnd=ifelse(zinb_zero>=0.5,0,round(zinb_pred,0)),
           zinb_match=ifelse(zinb_zero>0.5,ifelse(TARGET==0,1,0),ifelse(TARGET==zinb_pred_rnd,1,0)),
           zinb_res=TARGET-zinb_pred_rnd,
           zinb_fact=as.factor(zinb_match),
           is_zinb_zero=ifelse(zinb_zero>0.5,'0',as.character(zinb_pred_rnd)),
           is_zero=ifelse(TARGET==0,'Y','N'))

knitr::kable(holdout_zi1 |> group_by(zip_match) |> summarize(cnt=n()),
             format = "simple")

ggplot(data=holdout_zi1,aes(x=TARGET,fill=zip_fact)) +
    geom_bar() +
    scale_fill_manual(values=c('black', 'red'),labels = c("Correct","Incorrect")) +
    guides(color = guide_legend(title = "Discrete Predictions"))
knitr::kable(holdout_zi2 |> group_by(zinb_match) |> summarize(cnt=n()),
             format = "simple")
ggplot(data=holdout_zi2,aes(x=TARGET,fill=zinb_fact)) +
    geom_bar() +
    scale_fill_manual(values=c('black', 'red'),labels = c("Correct","Incorrect")) +
    guides(color = guide_legend(title = "Discrete Predictions"))
ggplot(holdout_zi1,aes(x=TARGET,y=is_zip_zero,fill=zip_res)) +
```

```
    xlim(-1,8) +
    geom_tile() +
    scale_fill_distiller(palette = "Spectral") +
    labs(y='Predicted Value') +
    guides(color = guide_legend(title = "Target - Prediction"))
ggplot(holdout_zi2,aes(x=TARGET,y=is_zinb_zero,fill=zinb_res)) +
    xlim(-1,8) +
    geom_tile() +
    scale_fill_distiller(palette = "Spectral") +
    labs(y='Predicted Value')+
    guides(color = guide_legend(title = "Target - Prediction"))
in_model_vars <- c('STARS','LabelAppeal','AcidIndex')
dist_eval_df <- eval_df |> dplyr::select(all_of(in_model_vars))
mod_eval_df <- dist_eval_df |>
    mutate(STARS=factor(STARS, levels = c(NA, 1, 2, 3, 4),
                        exclude = "",
                        ordered = TRUE),
           LabelAppeal=factor(LabelAppeal, levels = c(-2, -1, 0, 1, 2),
                        ordered = TRUE))
eval_pred <- predict(mod4_zinb,mod_eval_df,type='response')
eval_pred_zero <- predict(mod4_zinb,mod_eval_df,type='zero')
final_pred_df <- as.data.frame(cbind(mod_eval_df,eval_pred,eval_pred_zero)) |> mutate(predicted_cases=ifelse(eval_pred_zero>.5,0,eval_pred))
ggplot(final_pred_df,aes(x=predicted_cases)) +
    geom_bar()

write.csv(final_pred_df,'HW5_Eval_Predictions_Alternate.csv')
```