6.3730 PSet 1 Part 2

Stephen Andrews

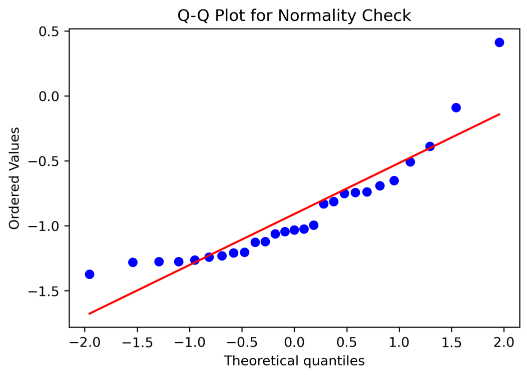
February 21, 2025

1.4:

In this problem we will perform multiple hypothesis testing over all 3051 genes to see if they have statistically different expression levels across the two tumors. For gene i, the test would look like the following

$$H_0 : \mu_L^i = \mu_M^i$$
$$H_A : \mu_L^i \neq \mu_M^i$$

where μ_L^i is the average expression level of gene i across ALL patients and μ_M^i is the average expression level of gene i across AML patients. The intuition for checking the means comes from CLT and treating the expression level of a gene across multiple people as i.i.d. random variables. To model this, we use an unpaired t-test without the assumption that the variances between the two distributions are the same. Welch's test matches this criteria, but before we move forward we create the Q-Q plot as a heuristic to make sure our data follows the normality assumption. For one gene we get the following plot:



While it is not perfect, I believe these points are close enough to the line to go ahead with the Welch's Test. In order to implement this, we used packages such as scipy, numpy, and pandas to make matrix calculations and the multiple hypothesis testing smooth and efficient. The details for calculating the test statistic and degrees of freedom can be found in the Wiki article for the test. Note that the packages imported also calculated p-values giving us the following results:

i Uncorrected p-values: 1078 significant genes

ii Holm-Bonferroni Correction: 103 significant genes

iii Benjamini-Hochberg Correction: 695 significant genes

1.5 a):

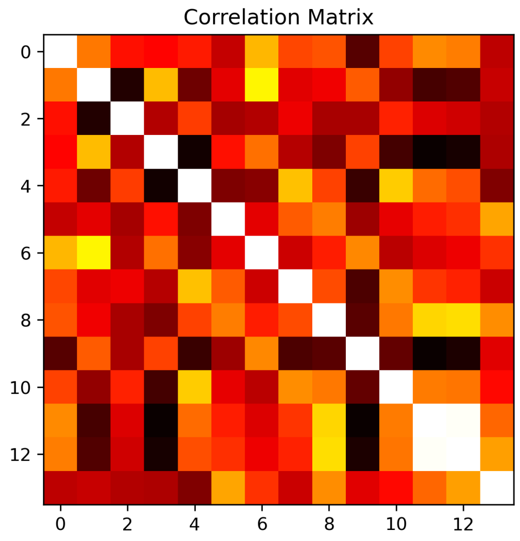
i)

We know that our OLS estimator follows $\hat{\beta} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T y$ where $\tilde{X}$ is our design matrix with column of ones on the left to allow for a constant term in our regression. Solving for this, we get that $\hat{\beta} = \begin{bmatrix} 1.93 \\ 1.26 \\ -4.60 \end{bmatrix}$. Note that 1.93 is the intercept term.

ii)

The first possible problem that we see is that different features are on completely different orders of magnitude. If we do not address this, the regression algorithm would overly prefer to fit towards the features with larger magnitude. This can be easily fixed by normalization. However, on closer inspection, we see that some features are right skewed. In particular, the air pollutant features, density, income features, and some other demographic features suffer from this property. In this case, it would be smarter to apply a log normalization to these features to spread out the densely packed smaller data points.

The next potential problem would come from exploring correlations between different features. Just based on inspection, it would seem that some of these features should definitely be correlated. In particular, the air pollutants was one that stuck out to me from the start. To verify this, we create the correlation matrix. As

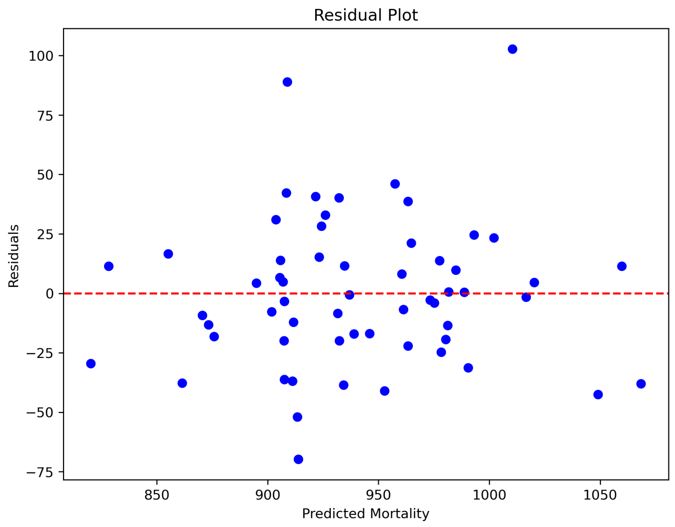


expected we get a correlation of 0.98 between HC and NOx. This collinearity can definitely pose issues during regression time, so in order to fix this, we would remove the HC.

Lastly, the city that stood out the most to me was New Orleans. Despite having some of the lowest air pollutant scores, it had the highest mortality rate providing some evidence against the correlation between air pollutants and mortality. This suggests some otehr factors are at play. In addition, Pittsburgh stands out by having lots of air pollutants but average mortality rate.

iii)

After normalizing as described in the previous part and removing the highly correlated feature we get an R^2 score of 0.748 which suggests the model explains the data decently well. In addition, we have the following residuals plot which are decently normally distributed.



1.5 b):

i)

The main challenge that will arise from this is the sheer size of the matrix. The total number of bits needed to store this matrix is 1.28 trillion which is well above what a standard computer can manage at once. Even if the computer was able to handle such a large matrix, performing the necessary matrix operations to calculate the true OLS estimator would take very long as matrix multiplications and matrix inverses are $O(n^3)$.

ii)

One potential solution to this would be to use an iterative method such as SGD. The pros of this are that we do not ever need to handle the entire matrix all at once for a particular calculation. This is helpful for memory efficiency. We only need to deal with one particular batch of the data at once. On the other hand, with SGD, there is no guarantee that we will converge to the global optimum solution. In addition, the algorithm performance can be sensitive the hyparameter selection such as learning rate.