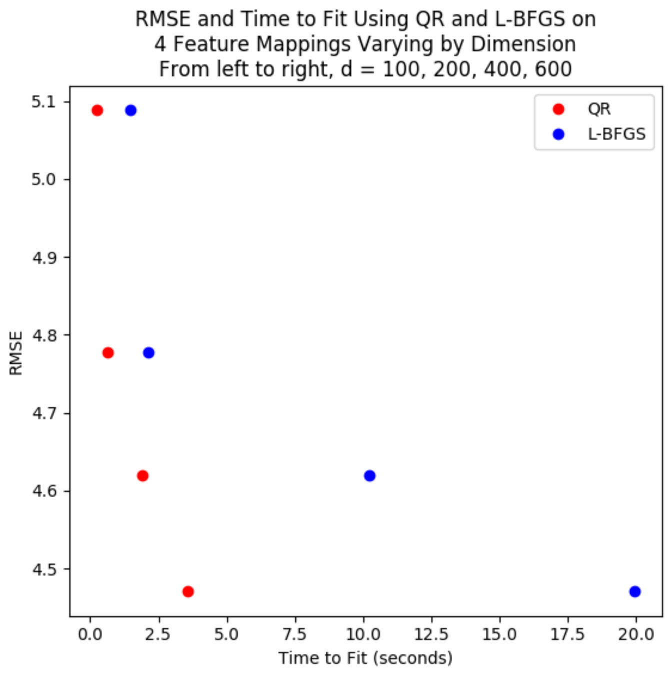

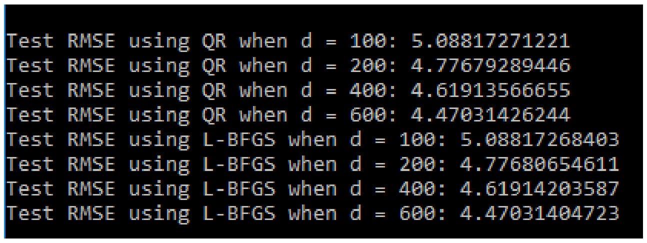

```
Test RMSE using QR when d = 100: 5.08817271221
Test RMSE using QR when d = 200: 4.77679289446
Test RMSE using QR when d = 400: 4.61913566655
Test RMSE using QR when d = 600: 4.47031426244
Test RMSE using L-BFGS when d = 100: 5.08817268403
Test RMSE using L-BFGS when d = 200: 4.77680654611
Test RMSE using L-BFGS when d = 400: 4.61914203587
Test RMSE using L-BFGS when d = 600: 4.47031404723
```

The results indicate that the non-linear transformations of the covariates reduced the prediction error. This is may be because our data follow a non-linear pattern with respect to some covariate(s), which is exactly when non-linear transformations of covariates are useful. When data follow a non-linear pattern, having no non-linear transformations of covariates will result in underfitting of the data. This was likely the case with our original set of covariates. Adding random noise to our original covariates and then using the cosine function to do a non-linear transformation worked well here, though I suspect that there are other transformations that would perform just as well.

The results also indicate that QR was faster than L-BFGS at coming up with the model fit for each d. For each increase in d, the time to complete model fitting using L-BFGS increased relative to the time it took to complete model fitting using QR.

Matrix inversion is normally the slowest part of coming up with the least squares or ridge solution to linear regression, but using QR decomposition means only having to invert an upper triangular matrix, which is relatively fast regardless of the value of d. L-BFGS involves using the gradient and Hessian matrix to update a d-dimensional vector of weights in order to find the set of d weights that minimize loss. When d becomes large, it makes sense that searching for the optimal weights would become slow. Note that it is possible that an alternative set of optimization parameters could speed up L-BFGS.