1 Coordinate Descent Algorithm

In this section we analyse the performance of two regression shrinkage methods – the lasso and the elastic net – by implementing the "one-at-a-time" coordinate descent algorithm specified in Friedman et al. (2007). This section of the report is organised as follows. In the first part, we describe the development of our coordinate descent algorithm for the lasso and elastic net regularization. Next, we explain the data simulation process, followed by a brief example that illustrates how we obtain the tuning parameters $\hat{\lambda}$ for our models using the mean-squared-error (MSE) statistic. Our numerical simulation results are analysed sections 4 and 5. We summarise our findings and conclude in section 6.

2 Coordinate Descent Algorithm for Lasso and Elastic Net

We use the following coordinate descent algorithm to solve the Lasso problem. Our function $lasso\_alg(X, Y, \lambda)$ takes three parameters as arguments and returns the vector of coefficients β.

1. We begin by **scaling X** and initialising β_j and $\beta_j^{old} = 0$, $j = 1, ..., p$
2. We repeat the following steps until the algorithm converges for $j = 1, ..., p$:
   - We let $\beta_j^{old} = \beta_j$. The partial residual is computed by taking $r_{ij} = y_i - \sum_{k \neq j} x_{ik}\beta_k$.
   - We compute the least square coefficients of residuals r_{ij} on the j-th predictor by taking $\beta_j^* = \frac{1}{n} \sum_{i=1}^{n} x_{ij} r_{ij}$
   - We can compute β_j using the *soft thresholding* method as follows:

$$\beta_j = \text{sign}(\beta_j^*) \max((|\beta_j^*| - \lambda), 0)$$

3. In repeating (2), we define convergence as the Root-Mean-Square Error (RMSE) between the current and prior iteration being smaller than 10^{-10}, where

$$RMSE = \sqrt{\frac{(\beta_j - \beta_j^{old})^2}{n}}$$

4. We **re-scale** β and return it as the function output.

The coordinate descent algorithm we use to solve the elastic net is very similar to the algorithm used to solve the lasso. We create a function, $elastic\_net\_alg(X, Y, \lambda_1, \lambda_2)$, that takes five parameters and returns the vector of coefficients β. Since the objective function differs, we make a modest modification to the *soft thresholding* from (2.c) in the algorithm above.

$$\beta_j = \text{sign}(\beta_j^*) \max((|\beta_j^*| - \lambda_1), 0) \ (1 + 2\lambda_2)^{-1}$$

We scale the data at the beginning so that each predictor has mean 0 and variance 1. In our derivation, $\beta_j^* = \frac{1}{\sum_{i=1}^{n}(x_{ij})^2} \sum_{i=1}^{n} x_{ij} r_{ij}$. Since we scale our data, $\sum_{i=1}^{n}(x_{ij})^2$ is close to n, hence we have $\beta_j^* = \frac{1}{n} \sum_{i=1}^{n} x_{ij} r_{ij}$ in the algorithm.

The aim of this project is to report the mean and standard error of the **mean-squared-error** (MSE) and the **number of estimated non-zero coefficients** for the lasso and the elastic net

across a number of cases. To ease implementation of this process, we create multiple functions to run the coordinate descent algorithms, select optimal tuning parameters, and evaluate the performance of optimised models. The final programme returns the three main reporting variables mentioned above, where $MSE = [\sum_{i=1}^{n} \sum_{j=1}^{p} (y_i - x_{ij}\beta_j)^2]/n$.

3 Data Simulation

The next step of our project is to generate simulations from the model $Y = X\beta + \sigma\epsilon$ where $\epsilon \sim N(0, I_n)$. In order to test the functionality of our program, we first set the parameters to the default case as follows.

1. Size of training dataset (n_train) = size of validation dataset ($n\_validation$) = 20
2. Size of testing dataset (n_test) = 200
3. $\beta = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$
4. $\sigma = 3$
5. $corr(X_i, X_j) = 0.5^{|i-j|}$
6. Number of simulations of the entire process (r) = 50

We first have to simulate two sets of random variables for X and ϵ. For the noise component ϵ, we use the *rnorm* function within R to generate random normal samples with mean = 0 and sd = 1. The tricky part here is to simulate the X variables as *rnorm* only allows for random variables with zero correlations. In our simulations, we need the pairwise correlation between X_i and X_j to be equal to $0.5^{|i-j|}$. We build a *Sigma_create* function to set up a covariance matrix stated above. Next, we utilise the built-in *mvrnorm* function to simulate our X values, calling the *sigma* that we have just created. Once we have X and ϵ, we use matrix multiplication to obtain Y data set.

The data sets are then split into a training set, an independent validation set and an independent test set. The training set is used to run the coordinate descent algorithms for a range of parameter values, the validation set is used to select the optimal tuning parameters, and the test set is used to estimate the accuracy of the models with optimal tuning parameters.

4 Obtaining Tuning Parameters $\hat{\lambda}$

After simulating the data sets for X and Y, the next step is to find a way to select the optimal tuning parameters: $\hat{\lambda}$ in lasso and $\hat{\lambda}_1$, $\hat{\lambda}_2$ in elastic net. **The general logic here is to perform trial and error with different values of λ on the training data set.** A range of λ is fitted into the *lasso* and *elastic_net* functions described earlier. The returned β coefficients for each respective λ are tested on the validation data using the *evaluation* function that we coded. We select the optimal parameter $\hat{\lambda}$ that produces the smallest error (MSE) on the validation data. **After we have obtained optimal tuning parameters, we re-estimate the model on the training data and obtain our final results by running it on the test data.**

To illustrate how we select the optimal lambda, we plot lambda against the MSE when solved using the coordinate descent algorithm. As shown in Figure 1, the range of lambda is $(0, 5)$ with a length of 25. The analysis is performed with the default case and the optimal point is highlighted in red. We select $\hat{\lambda} = 2.08$ as it gives us the lowest MSE of 13.4, compared to other λ, when evaluated against the testing data.

A similar thought process is applied to generate a pair of most efficient lambdas for the elastic net algorithm. The results is plotted in Figure 2 below, and the lowest MSE of 10.9 is achieved when $\hat{\lambda}_1 = 1.04$ and $\hat{\lambda}_2 = 0.208$.

In the default case, we can say that all tuning parameters are **significantly greater** than zero. If we take a closer look at the error terms (MSE), we can say that the performance of the elastic net dominates the lasso problem. However, the number of estimated non-zero coefficients selected by lasso is the true value 3 whereas elastic net selects a number of 4. In this default case, there exist a trade-off between the prediction accuracy and variable selection. However, we can't draw any conclusion from just one sample run and hence we proceed to repeat the simulation by r (50 by default) amount of times.

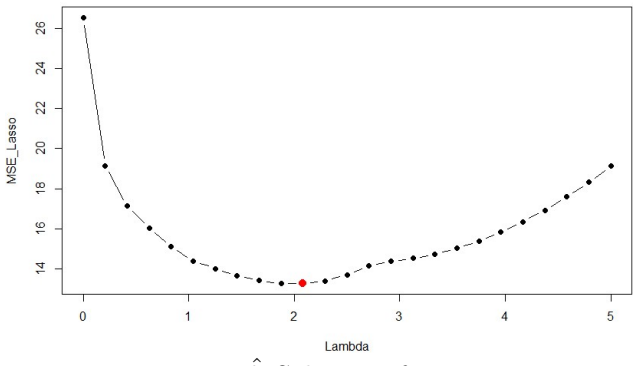

Figure 1: $\hat{\lambda}$ Selection for Lasso

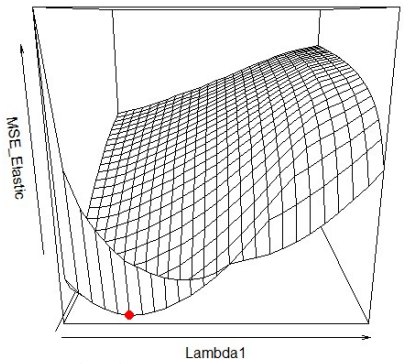

Figure 2: $\hat{\lambda}_1$ $\hat{\lambda}_2$ Selection for Elastic Net

We proceed to observe the different results produced by our algorithms. We can summarise the results into several scenarios when it comes to selecting the best Lambdas:

1. $\hat{\lambda}^{lasso} = \hat{\lambda}_1^{elastic} = \hat{\lambda}_2^{elastic} = 0$: The lasso and elastic net penalties are all reduced to Ordinary Least Square regression (OLS).

2. $\hat{\lambda}^{lasso} \neq 0$, $\hat{\lambda}_1^{elastic} \neq 0$, $\hat{\lambda}_2^{elastic} = 0$: The elastic net regularization will take the form of the lasso penalties instead.

3. $\hat{\lambda}^{lasso} \neq 0$, $\hat{\lambda}_1^{elastic} = 0$, $\hat{\lambda}_2^{elastic} \neq 0$: The elastic net regularization will shrink to a ridge regression problem in this scenario.

4. $\hat{\lambda}^{lasso} \neq 0$, $\hat{\lambda}_1^{elastic} \neq 0$, $\hat{\lambda}_2^{elastic} \neq 0$: The default case is an example of this scenario and it allows us to compare the performance between lasso and elastic net when using the simulated data sets.

5 Main Simulation Results

5.1 Base Case: $n > p$

Our default case sets the parameters exactly as described in Section 3. The bolded row in Table 1 below shows the main results of 50 simulations of our programme, focusing on reporting the three key variables of this project: the mean and standard error of MSE and the number of estimated non-zero coefficients.

Table 1: Results for Base Case with varying n

	Lasso			Elastic net		
No. of obs. (n)	MSE	MSE st. err.	# of vars	MSE	MSE st. err.	# of vars
120	16.53	0.89	4.5	16.10	0.76	5.4
240	**12.11**	**0.39**	**5.9**	**12.02**	**0.37**	**6.5**
480	10.56	0.16	6.2	10.58	0.17	6.4
960	9.60	0.08	5.7	9.62	0.09	5.7
1920	9.40	0.06	5.6	9.40	0.06	5.7
3840	9.21	0.03	6.0	9.21	0.03	6.0

As expected, **the elastic net regularization has better predictive performance than the lasso**, as shown in the smaller mean MSE for the elastic net. It also computes the MSE in a more stable manner, reflected in the lower standard error of the MSE. Importantly, the elastic net not only produces better prediction accuracy; **it also does well at variable selection**. The elastic net selected 6.5 variables on average across 50 simulations, only slightly higher than the 5.9 variables selected on average by the lasso.

We would also like to know how or whether these results vary with a change in the number of observations. As such, we repeat the Base Case simulation for datasets with various numbers of observations n, keeping the proportion of training, validation and testing sets equal.

Table 1 shows that our main findings hold when n is not too large. Given sufficiently large n, the elastic net problem takes the form of the lasso, with $\hat{\lambda}_2^{elastic} = 0$. And the MSE of using elastic net or lasso does not differ very much. The differences between the lasso and the elastic net shrink towards zero as $n \to \infty$, where all $\hat{\lambda} = 0$, and we have an Ordinary Least Square regression (OLS). We find that this is partly because of the small size of p relative to n; indeed, running the scenario with larger values of p yields greater differences between the lasso and elastic net. (We will discuss large values of p in more detail in Section 5.2.)

More generally, we explain this pattern as follows: as $n \to \infty$, we are able to make more reliable predictions (as shown in the standard error of the MSE), hence the MSE is negatively correlated with the number of observations. We therefore conclude that **the superiority of the elastic net over the lasso grows as our training dataset becomes more limited**. Working with a huge amount of data will result in the elastic net converging towards the lasso.

5.2 Case 2: $n < p$

The number of variables p has an opposite effect as the number of data n in relation with the estimation result. With a **large p relative to n, the elastic net penalty proves to be more advantageous than the lasso** as data become more scarce to estimate a large number of coefficients. Alternatively, the elastic net regularization will converge to the lasso regularization with a small number of variables. In this subsection, we are more interested in testing out a special case where the number of predictors is greater than the data size used to estimate the model.

Our second case is different from the default in one major way: we set the number of predictors, p, to be greater than the number of observations in the training data (n_train). Specifically, we maintain the size of training, validation and testing sets from our base case (20 training, 20 validation, 200 testing) and increase p to 25. Given the larger size of p, we generate non-overlapping coefficients β_j randomly by sampling without replacement and we leave the other parameters unchanged.

Theory predicts that the **lasso in this scenario selects at most n_train variables** before saturating. Our simulations results align with this finding. Across our 50 simulations, the average number of predictors selected by the lasso is 13.5, considerably lower than the 20.8 predictors chosen on average by the elastic net specification. It is also worth pointing out that error term for the elastic net specification ($MSE = 15.88$) is significantly lower than the lasso ($MSE = 18.55$) in this scenario. This makes intuitive sense, as there may be cases in which the optimal choice of non-zero coefficients is greater than n_train and elastic net can capture this better than the lasso.

5.3 Case 3: High pairwise correlations

In our third scenario, we vary the correlation between the first two predictors, X_1 and X_2, and record the results. Firstly, we discuss the situation when $X_1 = X_2$, then we proceed to run 50 simulations each for $corr(X_1, X_2)$ ranging from 0.1 to 0.9 (increasing by increments of 0.1). The main results in terms of prediction performance continue to hold: the elastic net performs slightly better in terms of the mean MSE, even though the difference are not that significant. The elastic net also has a slightly lower standard error than the lasso, and picks on average more variables than the lasso. For full results, please see Table 5 in the appendix.

The more interesting results in this scenario are the estimated values of β_1 and β_2, summarised in the Table 2 below. **When $X_1 = X_2$, we find that $\hat{\beta}_1 = \hat{\beta}_2$, which could be explained by theory.** Because when $X_1 = X_2$, $\hat{\beta}_1 + \hat{\beta}_2$ is close to $\beta_1 + \beta_2$. In elastic net model, the MSE keeps same no matter how $\hat{\beta}_1$ changes, but the penalty is minimal when $\hat{\beta}_1 = \hat{\beta}_2$. In lasso, both the MSE and penalty keep same because $\hat{\beta}_1 + \hat{\beta}_2$ is close to a constant, which means there are many possible solutions and our algorithm provides only one solution. **Besides, as the correlation increases, we find that the difference between the lasso's estimates of β_1 and β_2 narrows. The same is true for the elastic net, though to a lesser extent.** (See the Appendix for numerical results.)

Table 2: Varying Pairwise Correlations

	Lasso		Elastic net	
Pairwise Correlations	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_1$	$\hat{\beta}_2$
0.1	2.4	1.1	2.3	1.1
0.2	2.5	1.1	2.4	1.1
0.3	2.5	1.2	2.4	1.2
0.4	2.5	1.2	2.4	1.2
0.5	**2.5**	**1.2**	**2.4**	**1.3**
0.6	2.5	1.3	2.4	1.3
0.7	2.5	1.3	2.4	1.4
0.8	2.5	1.4	2.3	1.4
0.9	2.3	1.6	2.3	1.5
$X_1 = X_2$	2.0	2.0	1.9	1.9

6 Additional Simulation Settings

6.1 Variance Parameter σ

We want to investigate the relationship between the parameter sigma σ and both the lasso penalty and the elastic net penalty. This parameter introduce randomness and volatility into our simulated data Y. The higher the coefficient of σ, the more unpredictable and random our Y is. We have tabulated the result in Table 3 of running the algorithm with different levels of sigma while keeping the other parameters constant.

We observ that when we decrease the variance of our simulation, the elastic penalties converge into the lasso penalties with $\hat{\lambda}_2^{elastic} = 0$. And the elastic net doesn't yield results greater than the lasso problem. However, if we were to reverse the process and **increase the randomness parameter σ, the mean MSE of elastic net shows superior result compared to the MSE of lasso**. This is proven to be true when $\hat{\lambda}_2^{elastic}$ is statistically greater than zero. The parameter σ also negatively correlates with the number of estimated non-zero predictors. **High σ will result in a better variable selection** as we can see that the number decreases to the true number of variables. We can conclude that the **higher the parameter σ, the more likely we are to choose elastic net over lasso penalties**.

Table 3: Varying the Variance Parameter σ

	Lasso			Elastic net		
σ	MSE	MSE st. err.	# of vars	MSE	MSE st. err.	# of vars
0.5	0.52	0.03	7.2	0.52	0.03	7.2
1	1.55	0.07	6.2	1.55	0.07	6.2
2	5.56	0.22	6.2	5.57	0.21	6.3
3	**12.10**	**0.39**	**5.9**	**12.00**	**0.37**	**6.5**
4	20.95	0.62	5.3	20.42	0.55	6.4
5	31.80	0.81	4.6	30.84	0.67	6.0

6.2 Sparsity Level of β

We also investigate the impact of sparsity on the performance of the lasso and elastic net. Here we measure sparsity as the number of coefficients that are equal to zero. The process is simple: we begin with a coefficient vector that contains only non-zero coefficients (sparsity = 0). We run 50 simulations on this model and then increase sparsity to 1. We repeat this process until sparsity = $p - 1$ (all coefficients except one is equal to zero).

Results of these simulations are visualised in Figure 3 and Figure 4 below. The MSE of predictions is consistently lower under the elastic net specification, although this result may not be statistically significant. The similar performance of the lasso is interesting considering that the lasso consistently comes closer to choosing the "true" number of non-zero coefficients. **Both models systematically select too many non-zero coefficients (not enough sparsity), but the lasso consistently picks a more sparse model than the elastic net.**

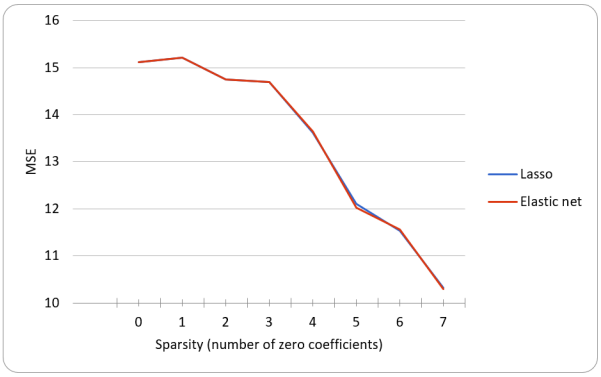

Figure 3: Mean MSE by Sparsity

Figure 4: Non-Zero Coefficients by Sparsity

The reason that the lasso picks a sparser model is illustrated in Figure 5. The elastic net typically selects a value for $\hat{\lambda}_1^{elastic}$ that is smaller than $\hat{\lambda}^{lasso}$. Accordingly, the elastic net assigns some small, non-zero value to $\hat{\lambda}_2^{elastic}$ (the tuning parameter representing the Ridge regression). Since Ridge regression does not do variable selection (i.e. it does not force coefficients to be exactly zero), this re-weighting of tuning parameters biases the elastic net towards less sparsity.

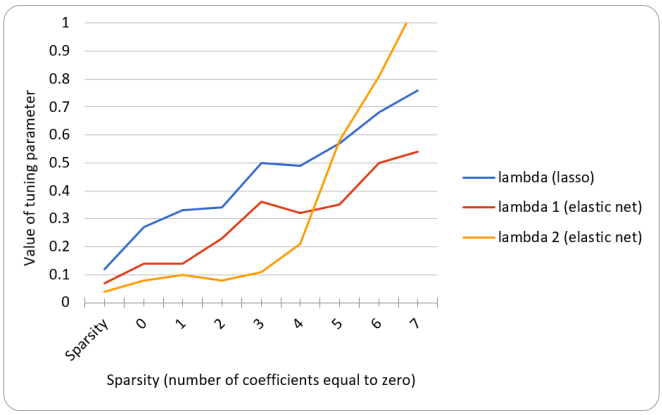

Figure 5: Tuning Parameters by Sparsity

6.3 Number of Simulations

In this subsection, we investigated how our model estimates shift if we change the number of times the process is looped. In the results tabulated in Table 4, we can see that the MSEs maintain roughly the same value disregarding the number of times it is looped. This is because the number of iterations doesn't heavily affect the estimated parameters. Instead, **the number of iterations is positively correlated to the reliability of the estimated parameters**. For example, one can take a look at the results simulated with 150 runs against the base case. The standard error of the MSE computed is strongly reduced for both lasso and elastic net. Not only that, the number of estimated non-zero coefficients also falls towards to the true value, which is 3.

Table 4: Varying the Number of Simulations r

	Lasso			Elastic net		
r	MSE	MSE st. err.	# of vars	MSE	MSE st. err.	# of vars
10	13.00	1.22	6.0	12.60	1.26	6.5
25	12.20	0.56	5.8	12.00	0.57	6.4
50	**12.10**	**0.39**	**5.9**	**12.00**	**0.37**	**6.5**
75	12.20	0.30	5.8	12.10	0.28	6.5
100	12.30	0.26	5.6	12.10	0.24	6.3
150	12.40	0.21	5.5	12.20	0.19	6.2

7 Conclusion

Summarising our findings, we first simulated datasets using the default case provided in the assignment. The default case is useful as it can act as a benchmark against which to compare other simulation settings. We set a constant seed of 16 throughout our entire project in order to generate reproducible results. The datasets are then split into three subsets with different purposes: training data to fit models, validation data to select tuning parameters and testing data to examine performance indicators.

We first implemented and adjusted a coordinate descent algorithm to solve the lasso and the elastic net problems. The coordinate descent algorithm solved using the $soft thresholding$ rule assumes orthonormality. In order to apply our algorithm, we **scaled** X to generate standardized values.

To select the optimal tuning parameters $\hat{\lambda}$, we ran the algorithm using in the range of $(0, 5)$. The best $\hat{\lambda}$ is associated with the lowest MSE when tested against the validation data. In the interest of limiting computing time, we choose a sequence of lambdas that is not very granular. A more granular sequence of lambdas would give us more precise values for the tuning parameters, but the benefit of this approach would likely be outweighed significantly increased runtime. We could also have increased the reliability of our estimated parameters by performing Monte Carlo simulations with a sizeable number of runs.

Our results demonstrated the superiority of the elastic net compared to the lasso in most scenarios. This is reflected in a lower standard error of the MSE over 50 iterations, pointing to a **more stable computation**. The mean of the MSE is also lower in most elastic net simulations, suggesting that **the elastic net dominates the lasso in terms of prediction accuracy**. We also observe that **the elastic net performs fairly well at variable selection**.

We run the algorithm across a number of settings and record the results. For the $n<p$ scenario, we observed that the lasso selects at most n variables before converging. In the special case where our predictors are highly correlated, we find that the difference between the lasso and the elastic net's estimates of β_1 and β_2 narrows as the correlation increases.

We also carried out additional simulation settings to investigate the relationship of various parameters with our results. Looking at the size of the dataset, the elastic net appears to be more powerful than the lasso when the dataset is relatively small. The elastic net is also the preferred regularization as the number of predictors increases. Lastly, the prediction accuracy of the elastic net compared to the lasso is improved when the simulated data is highly random.

One interesting point to note is that when we increase the sparsity of β, we face a trade-off between prediction accuracy and variable selection. The elastic net will generally provide higher prediction accuracy compared to the lasso. However, when the vector of coefficients β is sparse enough, the lasso comes closer to choosing the "true" number of non-zero coefficients due to its variable selection property. Since the elastic net is a combination of the lasso and the ridge penalties, and since the ridge regression is not capable of performing variable selection, the elastic net's ability to correctly identify a sparse coefficient vector is reduced.

Appendices

A References

Friedman, J., Hastic. and Hofling, H. (2007). Pathwise coordinate optimization, *The Annals of Applied Statistics* **1**: 302 - 332.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso, *Journal of Royal Statistical Society, Series B* **58**: 267 - 288.

Tseng, P.(1988). Coordinate ascent for maximizing nondifferentiable concave functions, *Technical Report.*

Tibshirani, R. (2001) *Convergence of block coordinate descent method for nondifferentiable maximization, J.Opt. Theory Appl.* **109**: 474 - 494.

B Figures and Tables

Table 5: Varying the Pairwise Correlations

	Lasso			Elastic net		
Pairwise Correlations	MSE	MSE st. err.	# of vars	MSE	MSE st. err.	# of vars
0.1	12.13	0.38	5.8	12.10	0.37	6.1
0.2	12.10	0.38	5.8	12.00	0.37	6.2
0.3	12.00	0.38	5.8	11.89	0.37	6.2
0.4	11.94	0.38	5.7	11.86	0.38	6.2
0.5	**12.1**	**0.39**	**5.9**	**12.0**	**0.37**	**6.5**
0.6	12.03	0.45	5.8	11.95	0.42	6.2
0.7	11.76	0.38	5.7	11.75	0.38	6.2
0.8	11.65	0.38	5.8	11.67	0.37	6.2
0.9	11.59	0.40	5.7	11.58	0.39	6.2
$X_1 = X_2$	11.57	0.38	5.9	11.73	0.38	6.4

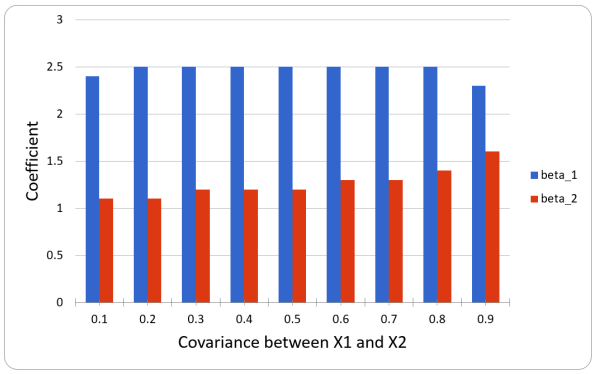

Figure 6: Estimated β_1 and β_2 for Lasso

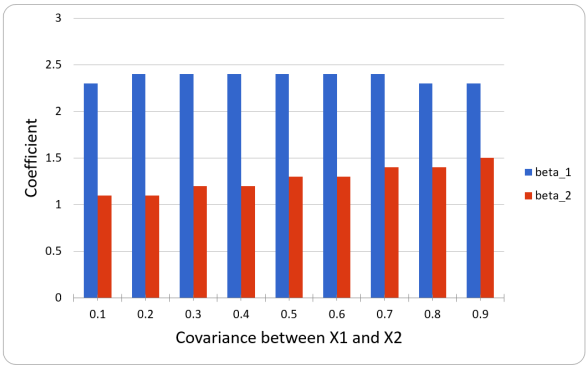

Figure 7: Estimated β_1 and β_2 for Elastic Net

Table 6: Varying Sparsity of β

	Lasso			Elastic net		
sparsity of β	MSE	MSE st. err.	# of vars	MSE	MSE st. err.	# of vars
0	15.10	0.67	7.9	15.10	0.67	7.9
1	15.20	0.67	7.8	15.20	0.67	7.8
2	14.80	0.65	7.5	14.80	0.65	7.5
3	14.70	0.71	7.3	14.70	0.71	7.3
4	13.60	0.52	6.8	13.60	0.51	6.9
5	**12.10**	**0.39**	**5.9**	**12.00**	**0.37**	**6.5**
6	11.30	0.28	4.8	11.10	0.26	6.0
7	10.40	0.22	3.1	10.10	0.18	4.5