Solution to Question 2.1

(a) Type the following commands in R to load the data and to view the data and its dimension:

```
load(file="eyedata.rda")
View(eyedata)
dim(eyedata)
```

(b) Simply use the following lines:

```
Y <- eyedata[,1]
X <- eyedata[,-1]
X <- as.matrix(X)
X <- scale(X, center=TRUE, scale=FALSE)
Y <- scale(Y, center=TRUE, scale=FALSE)
```

(c) Try the following commands in R:

```
QRdecomp <- qr(X)
rankX <- QRdecomp$rank
solve(t(X)%*%X)
```

Fit the OLS method to the data, ideally without intercept as all the variables are centred around 0, using the following code:

```
summary(lm(Y~-1+X))
```

and the output below shows the OLS estimates, where one can see that only the first 120 parameter estimates are produced and the rest are not calculated. This is because the design matrix is singular as $n < p$ (i.e., does not have full column rank). So the OLS method fails to work properly on this high dimensional data set.

```
Coefficients: (81 not defined because of singularities)
Estimate Std. Error    t value Pr(>|t|)
XX1377   1.809e+00  4.435e-13  4.079e+12 1.56e-13 ***
XX1748  -3.232e-01  9.895e-14 -3.266e+12 1.95e-13 ***
XX2487   1.976e-01  3.455e-14  5.719e+12 1.11e-13 ***
```

XX2679	-9.596e-01	2.682e-13	-3.578e+12	1.78e-13	***
XX2789	5.593e-01	9.037e-14	6.189e+12	1.03e-13	***
XX2875	-2.077e-01	1.056e-13	-1.968e+12	3.24e-13	***
XX3244	6.194e-01	1.244e-13	4.981e+12	1.28e-13	***
XX3375	-1.192e+00	3.133e-13	-3.805e+12	1.67e-13	***
XX3732	3.361e-01	1.110e-13	3.029e+12	2.10e-13	***
XX5892	9.227e-01	2.131e-13	4.330e+12	1.47e-13	***
XX6222	-2.087e-01	1.353e-13	-1.542e+12	4.13e-13	***
XX6242	-9.151e-02	5.142e-14	-1.780e+12	3.58e-13	***
XX6247	-2.395e-01	5.719e-14	-4.188e+12	1.52e-13	***
XX6359	-5.996e-01	2.368e-13	-2.533e+12	2.51e-13	***
XX6690	2.395e-01	5.268e-14	4.547e+12	1.40e-13	***
XX7069	4.630e-01	1.135e-13	4.081e+12	1.56e-13	***
XX7261	-9.329e-01	2.710e-13	-3.442e+12	1.85e-13	***
XX7941	1.135e+00	2.812e-13	4.037e+12	1.58e-13	***
XX8675	-1.337e-01	1.103e-13	-1.212e+12	5.25e-13	***
XX8835	2.231e-01	1.060e-13	2.104e+12	3.03e-13	***
XX9061	-5.572e-01	9.478e-14	-5.879e+12	1.08e-13	***
XX9096	-8.861e-01	2.855e-13	-3.103e+12	2.05e-13	***
XX9187	6.202e-01	1.568e-13	3.955e+12	1.61e-13	***
XX9303	6.058e-01	1.228e-13	4.931e+12	1.29e-13	***
XX9340	5.467e-01	1.571e-13	3.479e+12	1.83e-13	***
XX9972	-3.442e-01	7.363e-14	-4.675e+12	1.36e-13	***
XX10144	1.856e+00	4.662e-13	3.981e+12	1.60e-13	***
XX10196	7.493e-01	2.619e-13	2.861e+12	2.23e-13	***
XX10326	3.990e-01	1.459e-13	2.734e+12	2.33e-13	***
XX10438	2.060e-01	9.258e-14	2.225e+12	2.86e-13	***
XX10540	-1.101e+00	1.710e-13	-6.442e+12	9.88e-14	***
XX10693	1.422e-01	8.964e-14	1.586e+12	4.01e-13	***
XX10780	1.753e+00	2.792e-13	6.280e+12	1.01e-13	***
XX11024	6.141e-01	7.059e-14	8.699e+12	7.32e-14	***
XX11421	-2.779e-01	6.493e-14	-4.280e+12	1.49e-13	***
XX11609	-4.951e-01	2.301e-13	-2.151e+12	2.96e-13	***
XX11711	-2.343e-01	7.639e-14	-3.067e+12	2.08e-13	***

XX11719	2.995e-01	8.163e-14	3.669e+12	1.74e-13	***
XX11928	-3.998e-01	1.353e-13	-2.955e+12	2.15e-13	***
XX11995	1.037e+00	1.502e-13	6.906e+12	9.22e-14	***
XX12081	-5.394e-01	5.804e-14	-9.293e+12	6.85e-14	***
XX12085	2.790e-01	1.107e-13	2.521e+12	2.53e-13	***
XX12205	-4.925e-02	1.101e-13	-4.474e+11	1.42e-12	***
XX12813	-2.599e-01	1.715e-13	-1.515e+12	4.20e-13	***
XX12997	-5.183e-01	1.180e-13	-4.391e+12	1.45e-13	***
XX13092	-4.741e-01	1.164e-13	-4.075e+12	1.56e-13	***
XX13629	-1.252e+00	2.866e-13	-4.370e+12	1.46e-13	***
XX13858	1.057e+00	1.522e-13	6.942e+12	9.17e-14	***
XX13901	-7.876e-01	1.839e-13	-4.281e+12	1.49e-13	***
XX14046	6.151e-01	2.506e-13	2.454e+12	2.59e-13	***
XX14461	-5.679e-01	1.971e-13	-2.882e+12	2.21e-13	***
XX14631	6.083e-01	1.717e-13	3.543e+12	1.80e-13	***
XX14903	3.124e-01	9.139e-14	3.419e+12	1.86e-13	***
XX14949	-2.327e-02	9.160e-14	-2.540e+11	2.51e-12	***
XX15224	-1.161e+00	2.155e-13	-5.388e+12	1.18e-13	***
XX15289	-7.339e-01	2.462e-13	-2.981e+12	2.14e-13	***
XX15368	-7.512e-01	9.341e-14	-8.043e+12	7.92e-14	***
XX15636	-9.665e-02	7.140e-14	-1.354e+12	4.70e-13	***
XX15752	-1.554e-01	6.693e-14	-2.322e+12	2.74e-13	***
XX15787	1.459e+00	4.255e-13	3.430e+12	1.86e-13	***
XX15850	2.833e-01	1.165e-13	2.432e+12	2.62e-13	***
XX15863	-4.452e-01	1.236e-13	-3.602e+12	1.77e-13	***
XX15940	-1.007e-01	9.372e-14	-1.074e+12	5.93e-13	***
XX16014	-5.765e-01	9.955e-14	-5.791e+12	1.10e-13	***
XX16313	-7.438e-01	1.381e-13	-5.385e+12	1.18e-13	***
XX16541	3.168e-02	9.142e-14	3.466e+11	1.84e-12	***
XX16569	-1.194e-01	4.546e-14	-2.626e+12	2.42e-13	***
XX16801	5.004e-01	7.936e-14	6.306e+12	1.01e-13	***
XX16924	3.027e-01	1.227e-13	2.468e+12	2.58e-13	***
XX16964	3.521e-01	7.412e-14	4.751e+12	1.34e-13	***
XX16984	-4.675e-02	6.326e-14	-7.390e+11	8.61e-13	***

XX16988	9.986e-02	1.865e-13	5.356e+11	1.19e-12	***
XX17200	-2.554e-01	5.931e-14	-4.306e+12	1.48e-13	***
XX17270	-6.769e-01	1.306e-13	-5.183e+12	1.23e-13	***
XX17436	-2.697e-01	8.276e-14	-3.259e+12	1.95e-13	***
XX17599	-3.592e-01	1.159e-13	-3.101e+12	2.05e-13	***
XX17645	-2.040e-01	3.461e-14	-5.896e+12	1.08e-13	***
XX17723	-8.998e-01	3.266e-13	-2.755e+12	2.31e-13	***
XX17803	2.892e-01	4.825e-14	5.993e+12	1.06e-13	***
XX17816	-2.752e-01	6.529e-14	-4.215e+12	1.51e-13	***
XX17986	-4.506e-01	1.076e-13	-4.190e+12	1.52e-13	***
XX18062	-5.008e-01	1.878e-13	-2.667e+12	2.39e-13	***
XX18283	-1.052e+00	2.187e-13	-4.813e+12	1.32e-13	***
XX18389	-6.040e-02	1.337e-13	-4.518e+11	1.41e-12	***
XX18405	-1.124e+00	2.007e-13	-5.599e+12	1.14e-13	***
XX19331	-5.413e-01	9.111e-14	-5.941e+12	1.07e-13	***
XX21092	8.526e-04	1.592e-13	5.355e+09	1.19e-10	***
XX21094	-2.009e-01	1.758e-13	-1.143e+12	5.57e-13	***
XX21469	-7.517e-01	2.297e-13	-3.272e+12	1.95e-13	***
XX21550	2.902e-01	1.512e-13	1.919e+12	3.32e-13	***
XX21564	3.068e-01	1.038e-13	2.957e+12	2.15e-13	***
XX21680	8.079e-02	2.157e-13	3.746e+11	1.70e-12	***
XX21701	-5.263e-01	2.179e-13	-2.416e+12	2.64e-13	***
XX21791	3.351e-01	1.602e-13	2.091e+12	3.04e-13	***
XX21864	-1.258e+00	1.576e-13	-7.982e+12	7.98e-14	***
XX21907	4.182e-01	8.709e-14	4.802e+12	1.33e-13	***
XX21978	1.505e-01	9.712e-14	1.549e+12	4.11e-13	***
XX22016	3.342e-01	1.875e-13	1.782e+12	3.57e-13	***
XX22029	1.739e-01	1.091e-13	1.594e+12	3.99e-13	***
XX22043	-1.755e-01	7.382e-14	-2.378e+12	2.68e-13	***
XX22110	-5.681e-02	8.343e-14	-6.809e+11	9.35e-13	***
XX22140	-4.306e-01	8.318e-14	-5.176e+12	1.23e-13	***
XX22200	-1.209e-01	8.993e-14	-1.344e+12	4.74e-13	***
XX22277	-5.880e-01	1.781e-13	-3.302e+12	1.93e-13	***
XX22304	3.259e-01	1.695e-13	1.923e+12	3.31e-13	***

XX22423	5.073e-01	1.018e-13	4.983e+12	1.28e-13	***
XX22640	7.966e-01	1.245e-13	6.398e+12	9.95e-14	***
XX22694	5.148e-01	2.189e-13	2.352e+12	2.71e-13	***
XX22731	-3.813e-02	1.608e-13	-2.370e+11	2.69e-12	***
XX22813	-5.737e-01	1.598e-13	-3.590e+12	1.77e-13	***
XX22869	-1.218e+00	2.650e-13	-4.596e+12	1.39e-13	***
XX22896	3.493e-01	6.138e-14	5.690e+12	1.12e-13	***
XX22935	6.871e-01	1.059e-13	6.490e+12	9.81e-14	***
XX22938	-5.891e-02	8.299e-14	-7.099e+11	8.97e-13	***
XX22978	2.669e-01	1.183e-13	2.257e+12	2.82e-13	***
XX22980	7.455e-02	1.156e-13	6.449e+11	9.87e-13	***
XX23006	5.429e-01	1.573e-13	3.451e+12	1.85e-13	***
XX23041	-2.483e-01	2.574e-13	-9.646e+11	6.60e-13	***
XX23050	-1.286e+00	3.176e-13	-4.048e+12	1.57e-13	***
XX23110	NA	NA	NA	NA	
XX23161	NA	NA	NA	NA	
XX23206	NA	NA	NA	NA	
XX23288	NA	NA	NA	NA	
XX23348	NA	NA	NA	NA	
XX23404	NA	NA	NA	NA	
XX23618	NA	NA	NA	NA	
XX23804	NA	NA	NA	NA	
XX23805	NA	NA	NA	NA	
XX23877	NA	NA	NA	NA	
XX23942	NA	NA	NA	NA	
XX24087	NA	NA	NA	NA	
XX24198	NA	NA	NA	NA	
XX24225	NA	NA	NA	NA	
XX24245	NA	NA	NA	NA	
XX24282	NA	NA	NA	NA	
XX24353	NA	NA	NA	NA	
XX24396	NA	NA	NA	NA	
XX24413	NA	NA	NA	NA	
XX24422	NA	NA	NA	NA	

XX24565	NA	NA	NA	NA
XX24597	NA	NA	NA	NA
XX24618	NA	NA	NA	NA
XX24653	NA	NA	NA	NA
XX24783	NA	NA	NA	NA
XX24857	NA	NA	NA	NA
XX24892	NA	NA	NA	NA
XX24901	NA	NA	NA	NA
XX25000	NA	NA	NA	NA
XX25014	NA	NA	NA	NA
XX25055	NA	NA	NA	NA
XX25105	NA	NA	NA	NA
XX25109	NA	NA	NA	NA
XX25141	NA	NA	NA	NA
XX25281	NA	NA	NA	NA
XX25367	NA	NA	NA	NA
XX25403	NA	NA	NA	NA
XX25425	NA	NA	NA	NA
XX25439	NA	NA	NA	NA
XX25443	NA	NA	NA	NA
XX25852	NA	NA	NA	NA
XX25903	NA	NA	NA	NA
XX25909	NA	NA	NA	NA
XX26369	NA	NA	NA	NA
XX26672	NA	NA	NA	NA
XX26696	NA	NA	NA	NA
XX26712	NA	NA	NA	NA
XX26725	NA	NA	NA	NA
XX26738	NA	NA	NA	NA
XX26809	NA	NA	NA	NA
XX26868	NA	NA	NA	NA
XX26932	NA	NA	NA	NA
XX27179	NA	NA	NA	NA
XX27244	NA	NA	NA	NA

```
XX27354          NA          NA          NA        NA
XX27408          NA          NA          NA        NA
XX28164          NA          NA          NA        NA
XX28306          NA          NA          NA        NA
XX28343          NA          NA          NA        NA
XX28383          NA          NA          NA        NA
XX28680          NA          NA          NA        NA
XX28738          NA          NA          NA        NA
XX28891          NA          NA          NA        NA
XX28899          NA          NA          NA        NA
XX28964          NA          NA          NA        NA
XX28967          NA          NA          NA        NA
XX28983          NA          NA          NA        NA
XX29041          NA          NA          NA        NA
XX29045          NA          NA          NA        NA
XX29566          NA          NA          NA        NA
XX29665          NA          NA          NA        NA
XX29773          NA          NA          NA        NA
XX29842          NA          NA          NA        NA
XX29896          NA          NA          NA        NA
XX29912          NA          NA          NA        NA
XX29984          NA          NA          NA        NA
XX30031          NA          NA          NA        NA
XX30037          NA          NA          NA        NA
XX30078          NA          NA          NA        NA
XX30116          NA          NA          NA        NA
XX30141          NA          NA          NA        NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.455e-14 on 1 degrees of freedom
Multiple R-squared:      1,Adjusted R-squared:      1
F-statistic: 9.876e+25 on 119 and 1 DF,  p-value: 8.012e-14
```

(d) For $\lambda = 2$, use the following lines in R to fit the ridge regression to the data and get the parameter estimates:

```
M1 <- glmnet(X, Y, alpha=0, lambda=2)
M1$beta
```

However, it is important to note that one should be cautious here because `glmnet` asks for a sequence of λ values and not a single value of λ, and the result can be unreliable when using a single λ in `glmnet` (see the package `glmnet` online for further details on this warning). With this in mind, the ridge estimates with $\lambda = 2$ are as follows:

```
X1377  -1.757781e-03
X1748  -2.179735e-03
X2487  -8.427059e-04
X2679  -1.895241e-03
X2789  -2.185677e-03
X2875  -1.392777e-03
X3244   3.520188e-04
X3375  -1.246042e-03
X3732   2.079562e-03
X5892   9.990418e-04
X6222   4.355170e-03
X6242   1.868401e-03
X6247   2.300378e-03
X6359   1.190963e-03
X6690   1.175061e-03
X7069   1.360316e-03
X7261  -4.985680e-04
X7941  -4.760170e-04
X8675   1.369642e-04
X8835   7.849194e-04
X9061   1.476784e-03
X9096   7.994346e-04
X9187   8.619300e-04
X9303   1.655661e-03
X9340   1.876337e-03
X9972   2.802471e-03
```

```
X10144  4.084296e-04
X10196  8.911061e-04
X10326  2.109087e-03
X10438 -4.845175e-04
X10540 -1.928527e-03
X10693  9.924179e-04
X10780  1.699098e-03
X11024  1.385099e-03
X11421  1.898531e-03
X11609  2.396479e-03
X11711  1.967548e-03
X11719  1.704888e-03
X11928  1.269615e-03
X11995  2.323145e-03
X12081  3.690622e-04
X12085  3.483409e-03
X12205  8.141726e-04
X12813  5.437416e-04
X12997  8.261571e-04
X13092  1.747423e-03
X13629  6.377585e-04
X13858 -6.999613e-04
X13901 -1.809244e-03
X14046  8.638731e-04
X14461  3.040811e-05
X14631  2.076620e-03
X14903  2.053416e-04
X14949  4.411621e-03
X15224  2.346691e-03
X15289  6.422313e-04
X15368  1.489356e-03
X15636  1.998763e-03
X15752  9.346162e-04
X15787  3.493949e-03
```

```
X15850  1.377032e-04
X15863 -4.494107e-03
X15940  8.665501e-04
X16014  2.265048e-03
X16313  2.897697e-03
X16541 -3.435170e-06
X16569  2.346012e-03
X16801  5.539301e-05
X16924  9.592379e-04
X16964  2.721698e-04
X16984  7.854293e-05
X16988  1.713076e-03
X17200  1.243661e-03
X17270  2.551424e-03
X17436  2.883488e-03
X17599 -3.428217e-04
X17645  2.527724e-04
X17723  3.013868e-04
X17803 -2.321862e-03
X17816  9.879588e-04
X17986  6.344344e-05
X18062  1.560954e-03
X18283 -1.106524e-03
X18389  1.406642e-03
X18405  2.793666e-03
X19331  1.166959e-03
X21092 -7.858065e-03
X21094  3.538280e-03
X21469 -3.375625e-03
X21550 -5.660478e-03
X21564 -9.298735e-04
X21680 -4.936924e-03
X21701 -5.298933e-03
X21791  1.555478e-03
```

```
X21864  2.181490e-03
X21907  3.918551e-03
X21978  8.665585e-04
X22016  3.293681e-03
X22029  7.187960e-03
X22043  2.749609e-03
X22110  8.161955e-03
X22140 -7.474404e-03
X22200 -4.098955e-03
X22277 -3.086791e-03
X22304  2.516250e-03
X22423  2.415695e-03
X22640  2.897925e-03
X22694 -3.265112e-03
X22731 -3.779806e-03
X22813 -3.025461e-03
X22869 -4.766301e-03
X22896 -7.016286e-03
X22935 -7.327513e-03
X22938 -4.202154e-04
X22978 -3.477874e-03
X22980  4.736004e-04
X23006 -4.268401e-03
X23041  3.592722e-03
X23050  2.610018e-03
X23110  1.926959e-03
X23161  2.550137e-03
X23206  1.335358e-03
X23288  1.801659e-03
X23348 -5.946580e-03
X23404 -3.970945e-03
X23618 -6.390653e-04
X23804 -7.432657e-03
X23805  1.214504e-03
```

```
X23877 -1.613968e-03
X23942 -8.079755e-04
X24087  2.138647e-03
X24198 -3.918856e-03
X24225  3.813725e-03
X24245  6.912042e-03
X24282 -2.582483e-03
X24353 -7.881285e-03
X24396  2.652309e-03
X24413  1.446420e-03
X24422 -2.261850e-03
X24565  8.969519e-03
X24597 -5.440183e-03
X24618  4.236345e-03
X24653  4.792698e-03
X24783  2.150461e-03
X24857 -1.758426e-03
X24892  4.135683e-03
X24901  4.566216e-03
X25000  4.887242e-03
X25014  2.298893e-03
X25055 -2.298929e-03
X25105 -4.207619e-03
X25109  4.346833e-03
X25141  8.954323e-03
X25281 -4.251913e-03
X25367  3.836186e-03
X25403  1.721763e-03
X25425  5.989296e-03
X25439 -4.160741e-03
X25443 -3.504680e-03
X25852  2.607710e-03
X25903  5.578759e-03
X25909  7.117279e-03
```

```
X26369  1.245654e-03
X26672 -4.186170e-03
X26696 -3.586104e-03
X26712 -1.961074e-03
X26725 -5.388604e-04
X26738  2.821001e-03
X26809 -2.265180e-03
X26868  2.463868e-03
X26932  3.284391e-03
X27179  5.960105e-03
X27244 -3.696357e-03
X27354  1.028386e-03
X27408 -4.638022e-03
X28164 -4.545836e-03
X28306 -3.577029e-03
X28343 -1.698654e-03
X28383 -4.834677e-03
X28680  6.169940e-03
X28738 -7.423912e-03
X28891 -4.443347e-03
X28899 -5.721145e-03
X28964  3.698130e-03
X28967 -1.001495e-02
X28983 -1.765604e-03
X29041 -6.860544e-03
X29045 -4.848918e-03
X29566 -3.592618e-03
X29665 -3.474710e-03
X29773 -3.640880e-03
X29842  3.898001e-03
X29896 -2.218553e-03
X29912 -1.574216e-03
X29984  2.267048e-03
X30031  5.600521e-03
```

```
X30037 -4.220734e-03
X30078  2.689125e-03
X30116  5.573006e-03
X30141 -7.785276e-03
```

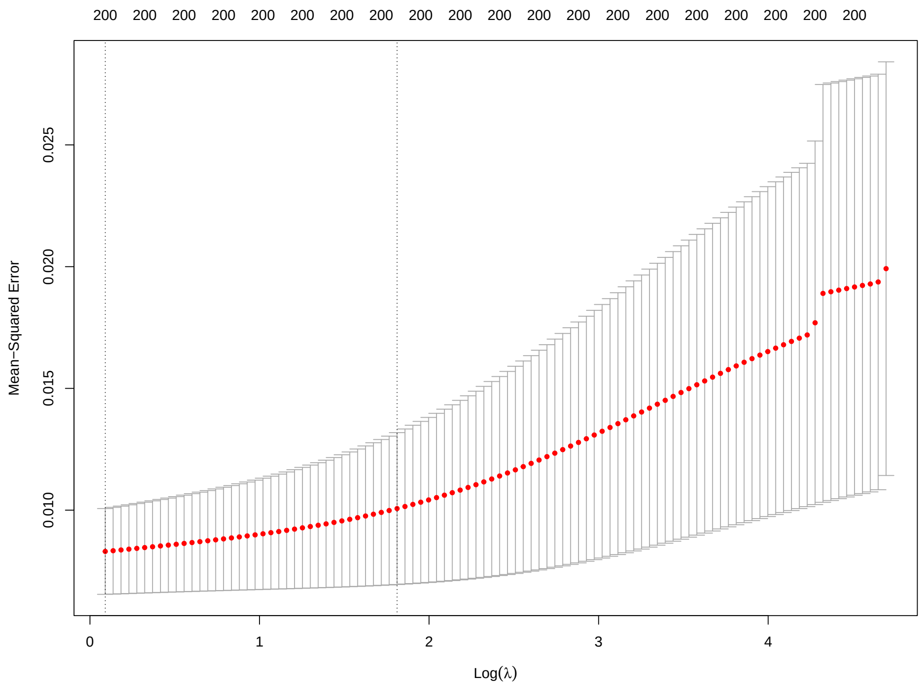
Also, the following line of code carries out the default cross validation (CV) for ridge regression with glmnet to select an optimal value of λ:

```
CV1 <- cv.glmnet(X, Y, alpha=0)
lambda_ridge <- CV1$lambda.min
```

The optimal value of λ would be 1.094429. The following line plots the results of the above cross validation:

```
plot(CV1)
```

which produces the following plot which suggests that the lambda value that minimises the CV error should be around $\log(1.094429) = 0.09023277$.



Use the following code to fit the ridge regression with the CV optimal value of λ and to get the estimates of parameters:

```
M1 <- glmnet(X, Y, alpha=0)
betahat_ridge <- coef(M1, s=lambda_ridge)
betahat_ridge_nointercept <- betahat_ridge[-1]
```

Note that using a single value of λ inside function coef is the right way to avoid the issue mentioned above about using a single value of λ inside function `glmnet`. The ridge estimates would then be as follows (the intercept estimate is very small as all variables are centred around 0):

```
(Intercept) -3.651802e-16
X1377       -1.771047e-03
X1748       -2.419708e-03
X2487       -1.396172e-04
X2679       -1.795420e-03
X2789       -2.287780e-03
X2875       -1.292992e-03
X3244       -5.347211e-04
X3375       -1.139430e-03
X3732        2.218167e-03
X5892        4.203318e-04
X6222        6.068241e-03
X6242        2.146372e-03
X6247        3.125317e-03
X6359        7.782829e-04
X6690        8.566690e-04
X7069        1.148970e-03
X7261        4.490281e-05
X7941        1.099493e-04
X8675        1.303005e-03
X8835        1.725936e-04
X9061        1.493298e-03
X9096        4.359575e-04
X9187        3.595637e-04
X9303        1.701681e-03
X9340        2.066601e-03
X9972        3.586485e-03
X10144      -2.439209e-04
X10196       3.745878e-04
X10326       2.513534e-03
X10438      -1.777322e-03
```

X10540	-2.305948e-03
X10693	3.802953e-04
X10780	2.105224e-03
X11024	1.560123e-03
X11421	2.099136e-03
X11609	2.964020e-03
X11711	2.426153e-03
X11719	1.898253e-03
X11928	1.481388e-03
X11995	2.916812e-03
X12081	-4.282684e-04
X12085	4.822099e-03
X12205	2.730633e-04
X12813	5.202400e-05
X12997	6.968049e-04
X13092	2.206198e-03
X13629	3.712737e-05
X13858	-4.835900e-05
X13901	-1.886284e-03
X14046	2.395927e-03
X14461	-7.831751e-04
X14631	2.500862e-03
X14903	-4.854127e-04
X14949	6.604953e-03
X15224	2.910487e-03
X15289	2.501480e-04
X15368	1.596365e-03
X15636	2.488027e-03
X15752	4.716231e-04
X15787	4.701200e-03
X15850	-3.478257e-04
X15863	-6.711334e-03
X15940	7.695773e-04
X16014	2.962102e-03

X16313	3.711273e-03
X16541	-9.032034e-04
X16569	3.082951e-03
X16801	-8.558583e-04
X16924	9.168967e-04
X16964	-2.892106e-04
X16984	-1.023331e-03
X16988	1.913507e-03
X17200	1.236905e-03
X17270	3.164068e-03
X17436	3.766028e-03
X17599	-1.796883e-03
X17645	-4.220052e-04
X17723	-5.424377e-04
X17803	-3.005871e-03
X17816	6.576213e-04
X17986	-6.817224e-04
X18062	1.679371e-03
X18283	-7.757276e-04
X18389	1.267870e-03
X18405	3.320436e-03
X19331	1.022358e-03
X21092	-1.150334e-02
X21094	3.944739e-03
X21469	-4.075325e-03
X21550	-7.658843e-03
X21564	4.172846e-04
X21680	-6.625395e-03
X21701	-6.777305e-03
X21791	1.268806e-03
X21864	2.406407e-03
X21907	5.159407e-03
X21978	-3.260784e-04
X22016	3.353137e-03

X22029	9.789838e-03
X22043	2.343799e-03
X22110	1.120998e-02
X22140	-1.056665e-02
X22200	-4.417768e-03
X22277	-3.406848e-03
X22304	2.303025e-03
X22423	2.941978e-03
X22640	3.199501e-03
X22694	-3.909799e-03
X22731	-4.521871e-03
X22813	-3.685793e-03
X22869	-5.716177e-03
X22896	-9.330379e-03
X22935	-9.502517e-03
X22938	6.931096e-04
X22978	-3.470781e-03
X22980	-4.240925e-04
X23006	-5.220866e-03
X23041	3.251652e-03
X23050	2.442118e-03
X23110	1.419531e-03
X23161	2.583683e-03
X23206	9.984385e-04
X23288	1.320079e-03
X23348	-7.435098e-03
X23404	-4.697015e-03
X23618	1.140957e-03
X23804	-9.817228e-03
X23805	8.698893e-04
X23877	-9.938035e-04
X23942	6.770647e-05
X24087	1.914323e-03
X24198	-4.377996e-03

X24225	3.882210e-03
X24245	8.995475e-03
X24282	-2.554818e-03
X24353	-1.021546e-02
X24396	1.835617e-03
X24413	1.280177e-03
X24422	-1.930170e-03
X24565	1.248891e-02
X24597	-6.788419e-03
X24618	4.356472e-03
X24653	5.795518e-03
X24783	1.829843e-03
X24857	-7.533029e-05
X24892	5.260114e-03
X24901	5.531260e-03
X25000	6.182925e-03
X25014	1.539520e-03
X25055	-1.948683e-03
X25105	-4.901906e-03
X25109	4.616298e-03
X25141	1.244708e-02
X25281	-4.781829e-03
X25367	4.997324e-03
X25403	9.598882e-04
X25425	7.756664e-03
X25439	-5.066568e-03
X25443	-3.804787e-03
X25852	2.958079e-03
X25903	6.981002e-03
X25909	9.180886e-03
X26369	5.396213e-04
X26672	-5.298105e-03
X26696	-3.486283e-03
X26712	-1.140648e-03

X26725	1.486162e-03
X26738	2.926532e-03
X26809	-1.861278e-03
X26868	2.730214e-03
X26932	2.695037e-03
X27179	8.058111e-03
X27244	-4.230637e-03
X27354	-1.062564e-03
X27408	-4.951159e-03
X28164	-4.949542e-03
X28306	-3.798929e-03
X28343	-6.797162e-04
X28383	-5.564326e-03
X28680	8.465511e-03
X28738	-9.565259e-03
X28891	-4.818008e-03
X28899	-6.764534e-03
X28964	4.674579e-03
X28967	-1.374199e-02
X28983	-7.659781e-04
X29041	-9.310652e-03
X29045	-6.234939e-03
X29566	-4.260735e-03
X29665	-4.001692e-03
X29773	-3.430904e-03
X29842	4.238934e-03
X29896	-1.583343e-03
X29912	-5.378749e-05
X29984	1.157673e-03
X30031	7.299805e-03
X30037	-4.915714e-03
X30078	2.428071e-03
X30116	6.792172e-03
X30141	-1.079238e-02

It can be seen that the results are not much different than the case with $\lambda = 2$.

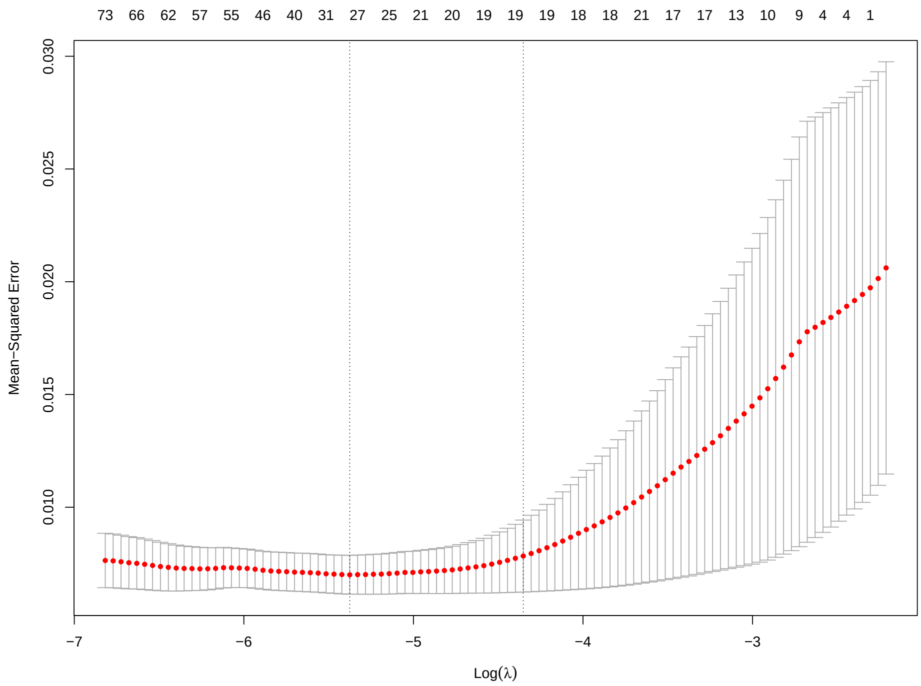
(e) The following lines carry out the default cross validation (CV) for lasso with `glmnet` to select an optimal value of λ:

```
CV2 <- cv.glmnet(X, Y, alpha=1)
lambda_lasso <- CV2$lambda.min
```

The optimal value of λ would be 0.0046286. The following line plots the results of the above cross validation:

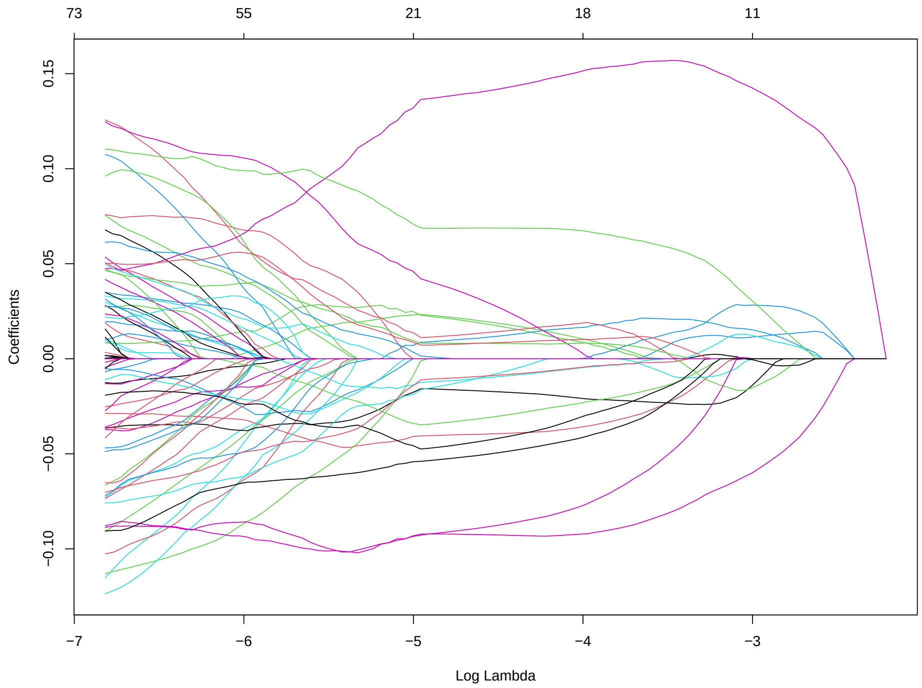
```
plot(CV2)
```

which produces the following plot which suggests that the lambda value that minimises the CV error should be around $\log(0.0046286) = -5.375501$.



Also, the following code produces the coefficient profile plot with the different values of lambda from CV showing the variable selection of lasso with different values of lambda:

```
plot(CV2$glmnet.fit, xvar="lambda")
```



Then, use the following lines to fit the lasso regression with the CV optimal value of λ and to get the estimates of parameters:

```
M2 <- glmnet(X, Y, alpha=1)
betahat_lasso <- coef(M2, s=lambda_lasso)
betahat_lasso_nointercept <- betahat_lasso[-1]
```

The lasso estimates are given below, where one can see that the lasso provides a sparse solution, that is, the estimates of many parameters are zero which indicates those variables are likely unimportant or less important. Also, the estimate of intercept is very close to 0, as expected.

```
(Intercept) -4.616236e-16
X1377           .
X1748           .
X2487           .
X2679           .
X2789           .
X2875           .
X3244           .
X3375           .
X3732           .
```

X5892	.
X6222	2.698587e-02
X6242	.
X6247	.
X6359	.
X6690	.
X7069	.
X7261	.
X7941	.
X8675	.
X8835	.
X9061	.
X9096	.
X9187	.
X9303	.
X9340	.
X9972	.
X10144	.
X10196	.
X10326	.
X10438	.
X10540	.
X10693	.
X10780	.
X11024	.
X11421	.
X11609	.
X11711	.
X11719	.
X11928	.
X11995	.
X12081	-1.021245e-03
X12085	.
X12205	.

X12813	.
X12997	.
X13092	.
X13629	.
X13858	.
X13901	.
X14046	3.860108e-02
X14461	.
X14631	.
X14903	.
X14949	1.924685e-02
X15224	.
X15289	.
X15368	.
X15636	.
X15752	.
X15787	.
X15850	.
X15863	-4.661526e-02
X15940	.
X16014	.
X16313	.
X16541	.
X16569	.
X16801	.
X16924	.
X16964	.
X16984	.
X16988	.
X17200	.
X17270	.
X17436	.
X17599	-4.789629e-02
X17645	.

X17723	.
X17803	.
X17816	.
X17986	.
X18062	.
X18283	.
X18389	.
X18405	.
X19331	.
X21092	-1.016628e-01
X21094	.
X21469	.
X21550	-3.233661e-02
X21564	.
X21680	.
X21701	.
X21791	.
X21864	.
X21907	1.403022e-03
X21978	.
X22016	.
X22029	.
X22043	.
X22110	.
X22140	-2.639269e-02
X22200	.
X22277	.
X22304	.
X22423	.
X22640	.
X22694	.
X22731	.
X22813	-1.773927e-02
X22869	.

X22896	.
X22935	.
X22938	.
X22978	.
X22980	.
X23006	.
X23041	.
X23050	.
X23110	.
X23161	.
X23206	.
X23288	.
X23348	.
X23404	.
X23618	.
X23804	-1.437885e-02
X23805	.
X23877	.
X23942	.
X24087	.
X24198	.
X24225	.
X24245	2.795895e-02
X24282	.
X24353	-2.132312e-02
X24396	.
X24413	.
X24422	.
X24565	6.430032e-02
X24597	.
X24618	.
X24653	.
X24783	.
X24857	.

X24892	2.147669e-02
X24901	.
X25000	.
X25014	.
X25055	.
X25105	.
X25109	.
X25141	1.056130e-01
X25281	.
X25367	1.962807e-02
X25403	.
X25425	2.731083e-03
X25439	.
X25443	.
X25852	.
X25903	1.436770e-02
X25909	.
X26369	.
X26672	-1.959333e-02
X26696	.
X26712	.
X26725	.
X26738	.
X26809	.
X26868	.
X26932	.
X27179	.
X27244	.
X27354	-7.881859e-03
X27408	.
X28164	.
X28306	.
X28343	.
X28383	.

```
X28680       8.967634e-02
X28738      -2.331786e-03
X28891       .
X28899       .
X28964       7.692939e-03
X28967      -1.015534e-01
X28983       .
X29041      -3.540357e-02
X29045      -3.818062e-02
X29566       .
X29665       .
X29773       .
X29842       .
X29896       .
X29912       .
X29984       .
X30031       .
X30037       .
X30078       .
X30116       .
X30141      -6.037323e-02
```

(f) Try the following lines to randomly split the data to training and test data and to calculate the prediction errors for the ridge and lasso methods.

```
set.seed(2)
n <- nrow(eyedata)
train_rows <- sample(1:n, 0.7*n)
X.train <- X[train_rows, ]
X.test <- X[-train_rows, ]
Y.train <- Y[train_rows]
Y.test <- Y[-train_rows]

CV1 <- cv.glmnet(X.train, Y.train, alpha=0)
lambda_ridge_train <- CV1$lambda.min
M1 <- glmnet(X.train, Y.train, alpha=0)
```

```
Yhat_ridge <- predict(M1,X.test,s=lambda_ridge_train)
MSPE_ridge <- mean((Y.test - Yhat_ridge)^2)

CV2 <- cv.glmnet(X.train, Y.train, alpha=1)
lambda_lasso_train <- CV2$lambda.min
M2 <- glmnet(X.train, Y.train, alpha=1)
Yhat_lasso <- predict(M2,X.test,s=lambda_lasso_train)
MSPE_lasso <- mean((Y.test - Yhat_lasso)^2)
```

The mean squared prediction error for the ridge regression is 0.004749165 and for the lasso is 0.004974798, so the two methods provide similar prediction error. This is probably because the data are not sparse and there are many coefficients with small values (i.e., many genes with small effects), so the lasso does not improve the prediction performance.

Solution to Question 2.2

(a) Use the following lines to load the data into R and to define the response variable and the covariates (centred around 0) as required:

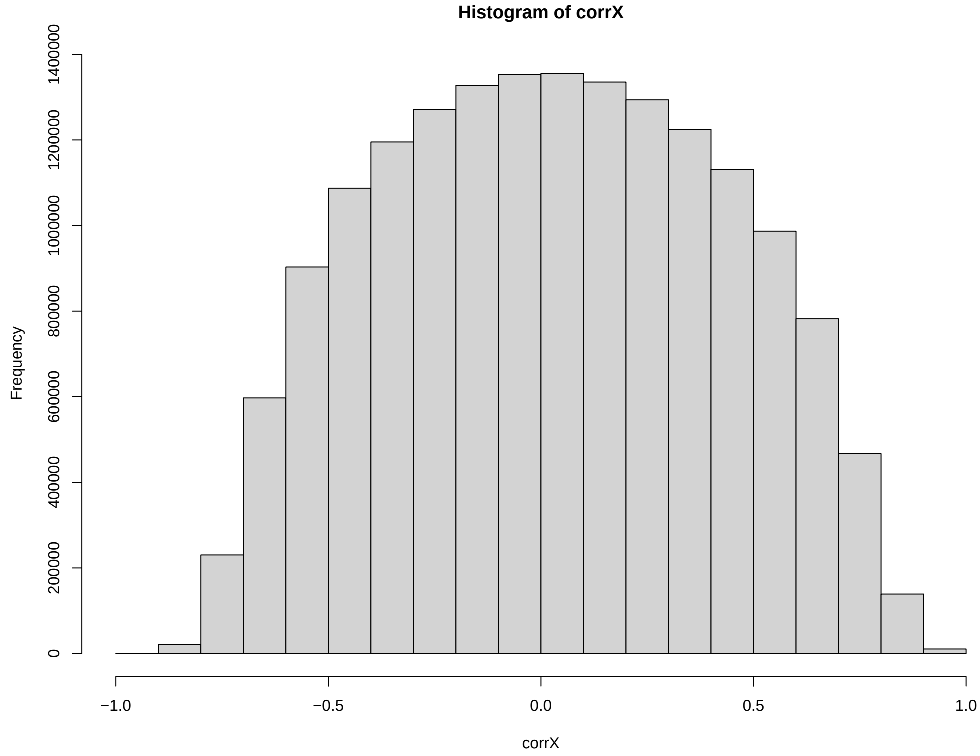
```
load(file="riboflavin.rda")
Y <- riboflavin$y
X <- riboflavin$x
X <- scale(X,center=TRUE,scale=FALSE)
n <- nrow(X)
p <- ncol(X)
```

(b) Calculate the pairwise correlation between covariates (genes) using the following line of code:

```
corrX <- cor(X)
```

Try the following to get a histogram of the pairwise correlations (rounded for some simplicity)

```
corrX <- round(corrX,4)
hist(corrX)
```

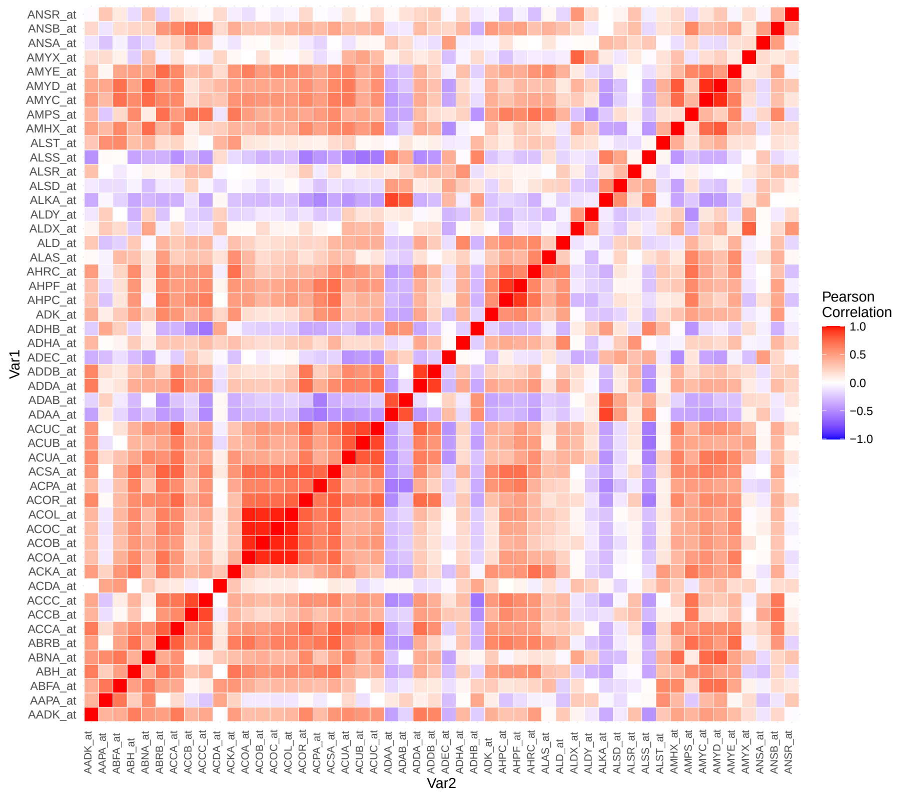


From this histogram, it is clear that the genes (covariates) are correlated, which should not be overlooked throughout the analysis. There are nicer ways of visualising the pairwise correlations. For example, the following lines produce a heat map of the pairwise correlations for a subset of 50 genes for clearer presentation.

```
library(reshape2)
corrX <- corrX[1:50,1:50]
melted_corrX <- melt(corrX)
head(melted_corrX)

library(ggplot2)
ggplot(data = melted_corrX, aes(Var2, Var1, fill = value))+
geom_tile(color = "white")+
scale_fill_gradient2(low = "blue", high = "red", mid = "white",
midpoint = 0, limit = c(-1,1), space = "Lab",
name="Pearson\nCorrelation") +
theme_minimal()+
theme(axis.text.x = element_text(angle = 90, vjust = 0,
```

```
size = 8, hjust = 0))+
coord_fixed()
```



The heat map highlights some of the high correlations among genes.

(c) The ridge and lasso methods can be applied to the data, similar to the previous question, using the following lines in R:

```
CV1 <- cv.glmnet(X, Y, alpha=0)
lambda_ridge <- CV1$lambda.min
plot(CV1)
M1 <- glmnet(X, Y, alpha=0)
betahat_ridge <- coef(M1, s=lambda_ridge)

CV2 <- cv.glmnet(X, Y, alpha=1)
lambda_lasso <- CV2$lambda.min
plot(CV2)
M2 <- glmnet(X, Y, alpha=1)
betahat_lasso <- coef(M2, s=lambda_lasso)
betahat_lasso_final <- betahat_lasso[betahat_lasso[,1]!=0,]
```

```
length(betahat_lasso_final)
```

Note that the above code also produces the non-zero estimates with the lasso (here 50 non-zero estimates), which are as follows:

```
(Intercept)      ARAN_at      ARGF_at      CTAA_at      DNAJ_at
-7.159432119  0.027154198 -0.196995942  0.005714725 -0.065710915
GAPB_at      LACA_at      LYSC_at      PRIA_at      sigM_at
0.012937101  0.004360216 -0.338690888  0.164420216  0.001671189
SPOIIAA_at    SPOVAA_at     THIA_at      THIK_at      XHLB_at
0.007773816  0.300356009 -0.026522354 -0.021234095  0.138756670
XKDB_at      YACN_at      YBFI_at      YCKE_at      YCLB_at
0.016656209 -0.072574008  0.156763252  0.005074657  0.214347996
YCLF_at      YDAO_at      YDDH_at      YDDK_at      YEBC_at
-0.059494672 -0.014102478 -0.062901412 -0.115225991 -0.600951844
YFHE_r_at     YFIO_at     YHDS_r_at     YISU_at      YKBA_at
0.148150427  0.299078921  0.202167625  0.023314583  0.115281459
YKNV_at      YKVJ_at      YLXW_at    YMAH_i_at      YMFE_at
0.007801508  0.147270239  0.106623127 -0.009626700  0.067260662
YOAB_at      YOPS_at      YPGA_at      YQJT_at      YQJU_at
-0.781742907  0.003663372 -0.068891958  0.110121760  0.233089487
YRVJ_at      YTGB_at      YUID_at      YWRO_at      YXIB_at
-0.055194037 -0.056342108  0.048743354 -0.104783243 -0.020937226
YXLD_at      YXLE_at      YYBG_at      YYCO_at      YYDA_at
-0.220130586 -0.095521657 -0.080735553 -0.097232931 -0.046532363
```

(d) To split the data to training and test data and to calculate the mean squared prediction errors of the ridge and lasso methods, use the following lines in R:

```
train_rows <- sample(1:n, 0.7*n)
X.train <- X[train_rows, ]
X.test <- X[-train_rows, ]
Y.train <- Y[train_rows]
Y.test <- Y[-train_rows]

Yhat_ridge <- predict(M1,X.test,s=lambda_ridge)
MSPE_ridge <- mean((Y.test - Yhat_ridge)^2)
```

```
Yhat_lasso <- predict(M2,X.test,s=lambda_lasso)
MSPE_lasso <- mean((Y.test - Yhat_lasso)^2)
```

The mean squared prediction error for the ridge regression is 0.4163754 and for the lasso is 0.406183. The lasso does not improve predictions which is probably because the covariates (genes) are correlated.

(e) To apply the elastic net with $\alpha = 0.5$, use the following code:

```
CV3 <- cv.glmnet(X, Y, alpha=0.5)
lambda_elastic <- CV3$lambda.min
plot(CV3)
M3 <- glmnet(X, Y, alpha=0.5)
betahat_elastic <- coef(M3, s=lambda_elastic)
betahat_elastic_final <- betahat_elastic[betahat_elastic[,1]!=0,]
length(betahat_elastic_final)
```

The output has a similar trend as the lasso in terms of providing a sparse solution. The non-zero estimates will be produced from the above code.

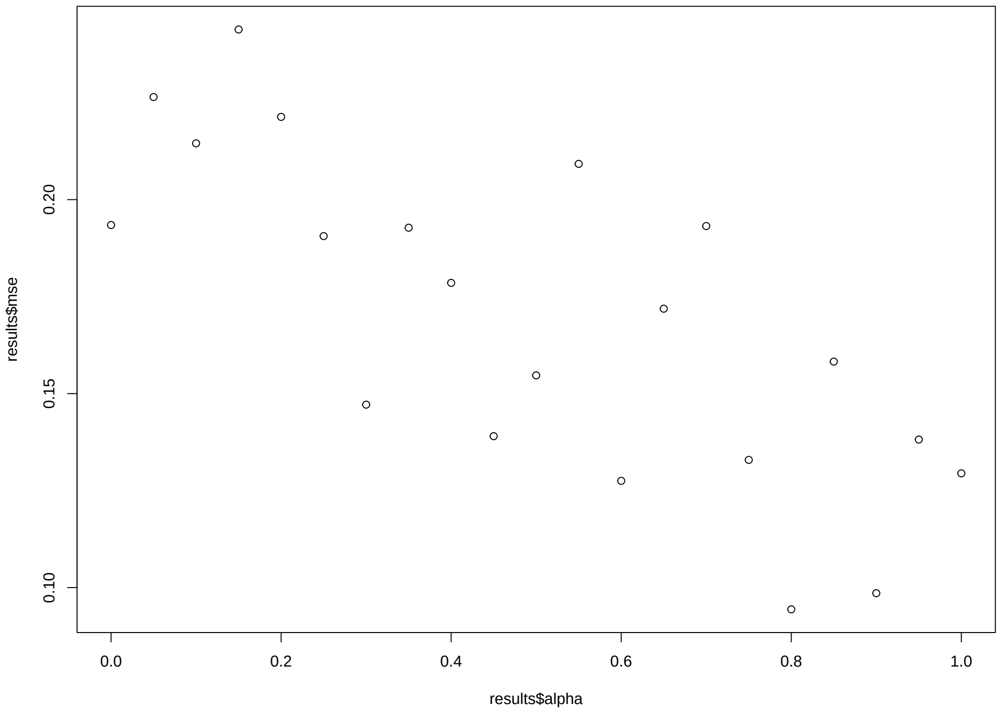
(f) To tune the additional parameter α in the elastic net problem using the Cross Validation requires a bit of additional work. The following code carries out this process:

```
models <- list()
for (i in 0:20)
{
name <- paste0("alpha", i/20)
models[[name]] <- cv.glmnet(X,Y,alpha=i/20)
}

results <- data.frame()
for (i in 0:20) {
name <- paste0("alpha", i/20)
predicted <- predict(models[[name]],
s=models[[name]]$lambda.1se, newx=X.test)
mse <- mean((Y.test - predicted)^2)
temp <- data.frame(alpha=i/20, mse=mse, name=name)
```

```
results <- rbind(results, temp)
}
plot(results$alpha, results$mse)
```

The resulting plot for an optimal value of λ that minimises the CV error is shown below. It can be seen that $\alpha = 0.8$ provides the smallest prediction error, so this can be used as an optimal value for λ in the elastic net for this data set.



Another important conclusion from this analysis and the plot is that neither of the ridge (when $\alpha = 0$) and lasso (when $\alpha = 1$) led to a smaller prediction error, suggesting that the elastic net performs better than ridge and lasso in terms of prediction performance. This is likely because the covariates or genes are correlated. The following lines in R produce the elastic net estimates with this optimal $\alpha = 0.8$:

```
elastic_net_est <- predict(models[["alpha0.8"]], type = "coef")
elastic_net_est <- elastic_net_est[elastic_net_est[,1]!=0,]
length(elastic_net_est)
```

The corresponding elastic net estimates (40 non-zero estimates) are as follows:

```
  (Intercept)        ARGF_at        CARA_at        DNAJ_at        GAPB_at
-7.159432e+00 -1.373598e-01 -1.954361e-05 -1.226862e-01  1.995645e-02
```

```
LYSC_at       PCKA_at       PKSA_at       RPLL_at    SPOIISA_at
-3.180017e-01  2.623722e-03  7.783027e-02 -5.583670e-03  4.337282e-02
SPOVAA_at       XHLB_at       XKDS_at       XLYA_at       XTRA_at
1.284764e-01  1.324097e-01  2.674957e-02  1.497694e-02  8.070677e-02
YBFI_at       YCDH_at       YCGO_at       YCKE_at       YCLB_at
1.718087e-01 -4.538235e-03 -1.204105e-02  6.302197e-02  1.653865e-01
YCLF_at       YDDH_at       YDDK_at       YDDM_at       YEBC_at
-5.646364e-02 -4.664725e-03 -1.347145e-01 -1.898880e-02 -4.207721e-01
YEZB_at     YFHE_r_at       YFII_at       YFIR_at     YHDS_r_at
5.276095e-02  9.485770e-02  1.433775e-02  1.728916e-02  7.067689e-02
YHZA_at       YKBA_at       YOAB_at       YQJU_at       YRVJ_at
-4.000391e-03  3.829271e-02 -6.159279e-01  7.803749e-02 -4.937386e-02
YTGB_at       YURQ_at       YXLD_at       YXLE_at       YYDA_at
-1.720924e-02  1.360024e-01 -1.636034e-01 -1.152646e-01 -9.590789e-02
```