Modern Statistical Methods 2023

Tom Maullin

2023-11-14

Multiple Regression

We'll begin today by performing a linear regression using the `lm` (linear model) package in R. To do so, we first load the "BloodPressure.csv" file.

```
# Read blood pressure file
bp=read.csv("BloodPressure.csv")

# Look at first few lines of file
head(bp)
```

```
##   Systolic Diastolic Gender Age WeightInKg HeightInCm EyeColour
## 1      120        80   male  25         60        165     brown
## 2      119        76 female  31         57        151     brown
## 3      145        85   male  56         80        163     brown
## 4      178        98 female  72         55        140     brown
## 5      132        80   male  35         78        159     brown
## 6      100        78 female  20         58        170     brown
```

```
# Look at file summary
summary(bp)
```

```
##     Systolic       Diastolic         Gender               Age       
##  Min.   : 94.0   Min.   : 67.00   Length:37          Min.   :18.00  
##  1st Qu.:132.0   1st Qu.: 78.00   Class :character   1st Qu.:35.00  
##  Median :145.0   Median : 94.00   Mode  :character   Median :55.00  
##  Mean   :145.9   Mean   : 89.24                      Mean   :52.24  
##  3rd Qu.:168.0   3rd Qu.: 98.00                      3rd Qu.:68.00  
##  Max.   :200.0   Max.   :110.00                      Max.   :88.00  
##    WeightInKg       HeightInCm     EyeColour        
##  Min.   : 54.00   Min.   :140.0   Length:37         
##  1st Qu.: 60.00   1st Qu.:155.0   Class :character  
##  Median : 74.00   Median :163.0   Mode  :character  
##  Mean   : 73.73   Mean   :163.8                     
##  3rd Qu.: 86.00   3rd Qu.:170.0                     
##  Max.   :100.00   Max.   :193.0                     
```

The above code loads in the blood pressure file, prints off the first few rows of the data, and summarises the table. Make sure you understand each of the items in the summary table before proceeding.

We are interested in how a subject's body mass index (BMI), which is their weight (Kg) divided by the square of their height (M), is influenced by age. Try using the `pairs()` function to view all possible scatterplots.

> **Note:** You might find some of the columns of your data cannot be viewed in `pairs`. Make sure you understand why this is. By constructing a second dataframe containing only the numeric columns of `bp`, view the columns you can using `pairs`.

```
# Write your code here...
```

We begin with a little preprocessing. Unfortunately, BMI is currently missing from the `bp` dataframe. To rectify this, compute BMI from the given weight and height data; carefully consider your units and check that you've actually computed realistic BMI values (normal is between about 18 and 25). Use the `summary` function to check your computations. What do these values tell you about the health of the subjects?

```
# Write your code here...
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   17.99   22.04   28.06   27.94   33.78   39.96
```

Using the `lm` function, construct a linear model containing both an intercept and `age` as predictors, and `BMI` as a response.

> **Note:** By default `lm` will include the intercept for you.

```
# Write your code here...
```

Plot the data with the regression line, then check the diagnostic plots (use `plot` with the `lm` model object you're viewing; e.g. if your model object is `mod`, then `par(mfrow=c(2,2));plot(mod)` will show you the 4 default plots all at once). What do you observe? Is it a good fit?

```
# Write your code here...
```

You might want to visit this page for a deeper understanding of these plots.

> **Extra Task:** Try running a model of the form `BMI ~ 0 + Age` in `lm`. This will likely give you different results to your analysis above. Why do you think this is? If you are unsure, look online or feel free to ask one of the tutors.

```
# Write your code here...
```

Now, fit a quadratic model for the same data. You may wish to look up the `poly(X,k)` function to help you get a polynomial for explanatory variable X with order k. (This post may be helpful…)

```
# Write your code here...
```

Try this again for a cubic model (k=3) and then for higher order terms (k=4,5,…).

```
# Write your code here...
```

Use `anova` to compare these models. Which one do you think is best?

```
# Write your code here...
```

In general, are there any issues you should be mindful of when running many models and comparing them, like we have above? (Hint, Hint).

Now, consider the other variables in `bp` **except for** Height and Weight, from which BMI is derived (why do you think it might be a bad idea to include these variables?).

Using `lm` again, but with no polynomials, what do you think is the best model for BMI? Use your common sense knowledge of what you think might be important, trying different models, but be sure to include Age. Try a few models and find a model that is best.

> **Note:** When you include `gender` in the model it is included as a `factor`. If you are unsure what a `factor` is, look online to find out before proceeding with the rest of the tutorial. If you are still unsure, feel free to ask one of the tutors present.

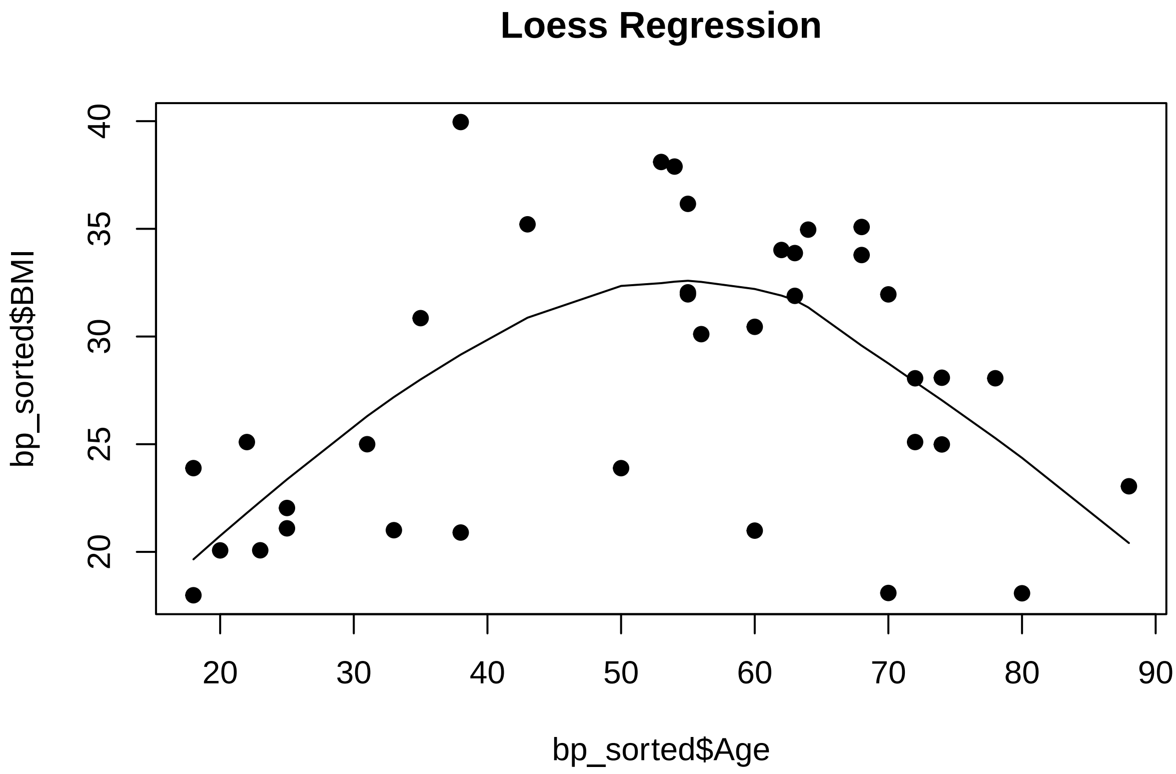
```
# Write your code here...
```

You might find the results aren't quite as expected… what do you think is best to do in this situation? A little common sense may also be required when modelling. After all, *if the model tells you to jump off a cliff…*

General Additive Models

Continuing with the `bp` dataset, we shall now try running some general additive models. Before we look into this, lets use `loess` to see if anything jumps out at us. Below we have provided a `loess` fit for BMI vs Age. Try doing this with a few other variables (perhaps those you found of interest in the last section). Do any trends jump out at you?

```
# Sort the data by Age
bp_sorted <- bp[order(bp$Age),]

# Fit the loess model with the sorted data
loess_mod <- loess(BMI ~ Age, data=bp_sorted)

# Make predictions using the loess model
loess_predict <- predict(loess_mod)

# Plot the sorted data
plot(bp_sorted$Age, bp_sorted$BMI, pch=19, main='Loess Regression')

# Add the loess regression line
lines(bp_sorted$Age, loess_predict)
```



We'll now fit a General Additive Model. To get you started, here is a very simple model (but you can do better surely).

```
modgam=gam(BMI~Gender+s(Age),data=bp)
summary(modgam)
```

```
##
## Call: gam(formula = BMI ~ Gender + s(Age), data = bp)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1840  -2.3086   0.3142   3.5607  10.9089
##
## (Dispersion Parameter for gaussian family taken to be 26.1535)
##
##     Null Deviance: 1493.25 on 36 degrees of freedom
## Residual Deviance: 810.7523 on 30.9998 degrees of freedom
## AIC: 233.2226
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##           Df Sum Sq Mean Sq F value  Pr(>F)
## Gender     1   0.31   0.311  0.0119 0.91391
## s(Age)     1 112.10 112.105  4.2864 0.04683 *
## Residuals 31 810.75  26.153
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##             Npar Df Npar F   Pr(F)
## (Intercept)
## Gender
## s(Age)             3 6.8308 0.00115 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Try a few models and see which is better.

> **Note:** There is some debate as to the best approach for comparing GAMs. We recommend using comparisons such as `AIC` or `anova` only as rules of thumb here. Be careful not to overstate your results.

Regularised Regression

We shall now run some regularised regression models. In this section, we shall first work through an example analysis in full, complete with data and code. Following this, you will be given a new dataset and asked to analyse it independently.

Prostate Data Worked Example

The dataset we shall use for demonstration is the `prostate` dataset described in the Elements of Statistical Learning textbook, found here or on the Canvas site.

```
prostate=read.table("prostate.data")
summary(prostate)
```

```
##      lcavol           lweight            age             lbph
##  Min.   :-1.3471   Min.   :2.375   Min.   :41.00   Min.   :-1.3863
##  1st Qu.: 0.5128   1st Qu.:3.376   1st Qu.:60.00   1st Qu.:-1.3863
##  Median : 1.4469   Median :3.623   Median :65.00   Median : 0.3001
##  Mean   : 1.3500   Mean   :3.629   Mean   :63.87   Mean   : 0.1004
##  3rd Qu.: 2.1270   3rd Qu.:3.876   3rd Qu.:68.00   3rd Qu.: 1.5581
##  Max.   : 3.8210   Max.   :4.780   Max.   :79.00   Max.   : 2.3263
##       svi              lcp             gleason          pgg45
##  Min.   :0.0000   Min.   :-1.3863   Min.   :6.000   Min.   :  0.00
##  1st Qu.:0.0000   1st Qu.:-1.3863   1st Qu.:6.000   1st Qu.:  0.00
##  Median :0.0000   Median :-0.7985   Median :7.000   Median : 15.00
##  Mean   :0.2165   Mean   :-0.1794   Mean   :6.753   Mean   : 24.38
##  3rd Qu.:0.0000   3rd Qu.: 1.1787   3rd Qu.:7.000   3rd Qu.: 40.00
##  Max.   :1.0000   Max.   : 2.9042   Max.   :9.000   Max.   :100.00
##       lpsa           train
##  Min.   :-0.4308   Mode :logical
##  1st Qu.: 1.7317   FALSE:30
##  Median : 2.5915   TRUE :67
##  Mean   : 2.4784
##  3rd Qu.: 3.0564
##  Max.   : 5.5829
```

This dataset contains the following predictive variables:

- Log Cancer Volume (`lcavol`): The area of cancer, measured from digitized images.
- Log Weight (`lweight`): The log of prostate weight.
- Age (`age`): Subject age.
- Log Vening Prostatic Hyperplasia (`lbph`): Log of the amount of benign prostatic hyperplasia.
- Seminal Vesicle Invasion (`svi`): 0-1 variable representing invasion of the muscular wall of the seminal vesicles by prostate cancer.
- Log Capsular Penetration (`lcp`): Log capsular penetration greater than 1 cm in linear extent.
- The Gleason score (`gleason`): The Gleason score (see here for more detail).
- Percent of Gleason score 4 or 5 (`pgg45`): The percentage of tumor occupied by Gleason grades 4 and 5.
- Train: A binary variable we have created for demonstration purposes to ensure the observations you will use for training match the output we have provided for this example.

The goal is to fit a predictive model for the gold standard value of `lpsa` (log PSA, prostate specific antigen) using the other variables. To begin, we must identify our predictors (`x`) and outcome (`y`). We must then split the data into testing and training data (why must we do this? If you are unsure, please ask one of the helpers).

```
y=scale(prostate[,9])
x=scale(as.matrix(prostate[,1:8]))
# Scaling is important to ensure all variables are equally influential for the Lasso/Ridge
train=prostate[,10]
y.train=y[train]
x.train=x[train,]
y.test=y[!train]
x.test=x[!train,]
```

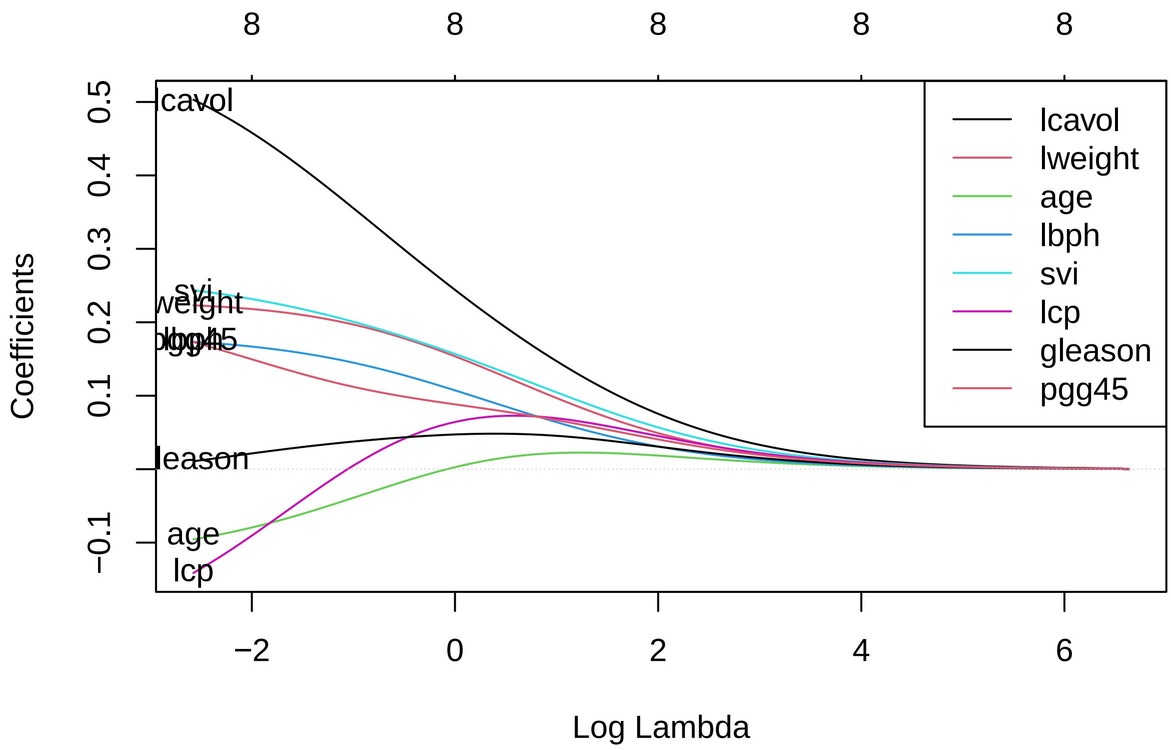
This utility function will be helpful to plot the 'solution' paths.

```
plotglmnet <- function(fit, lams=NULL,  ...) {
        L <- length(fit$lambda)
        x <- log(fit$lambda[L])
        y <- fit$beta[, L]
        labs <- names(y)
        text(x, y, labels=labs, ...)
        abline(h=0,lty=3,col='gray',lwd=0.5)
        if (!is.null(lams))
            abline(v=log(lams),lty=3)
        legend('topright', legend=labs, col=1:6, lty=1)
}
```

We shall now use the `glmnet` package to perform a ridge regression on the training data. The `glmnet` package supports Ridge, Lasso and Elastic Net (mixture of L0 and L1 penalties) regression. These are controlled by the `alpha` parameter: Ridge `alpha=0`, Lasso `alpha=1`, and Elastic net is any value between 0 and 1.

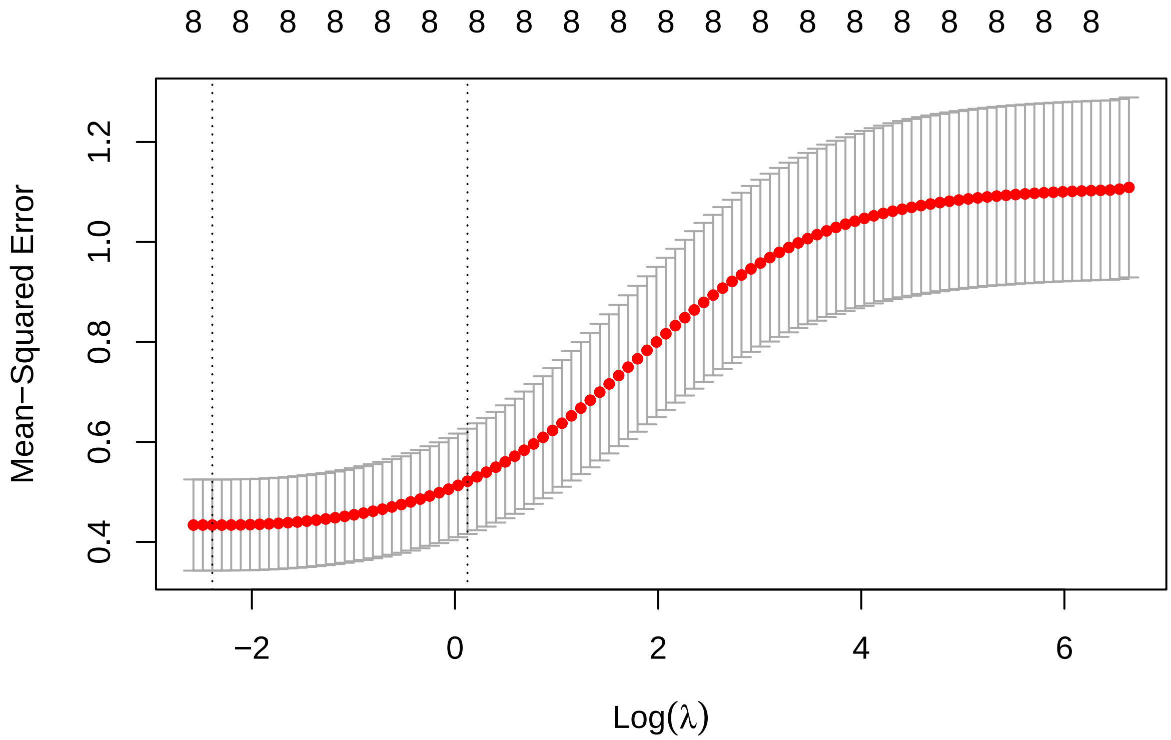
Below is a ridge regression.

```
# Fit the Ridge Regression
fit=glmnet(x.train,y.train,alpha=0)

# Plot the results
plot(fit,xvar="lambda")
plotglmnet(fit)
```



We can now find the optimal penalty parameter using the `cv_fit` function.

```
cv_fit <- cv.glmnet(x.train, y.train, alpha = 0)
opt_lambda <- cv_fit$lambda.min
plot(cv_fit)
```



See if you can answer the following questions: - What do these plots tell you? - Why is the number 8 repeated across the top of the figure? (Hint: look back at this mornings slides if you are unsure). - What are the two vertical lines in the last plot, and how can you obtain their values from the `cv_fit` object? (Again, you may wish to check this mornings slides).

Here are the 'internal' predictions, predictions for the training data with a model fit on the training data:

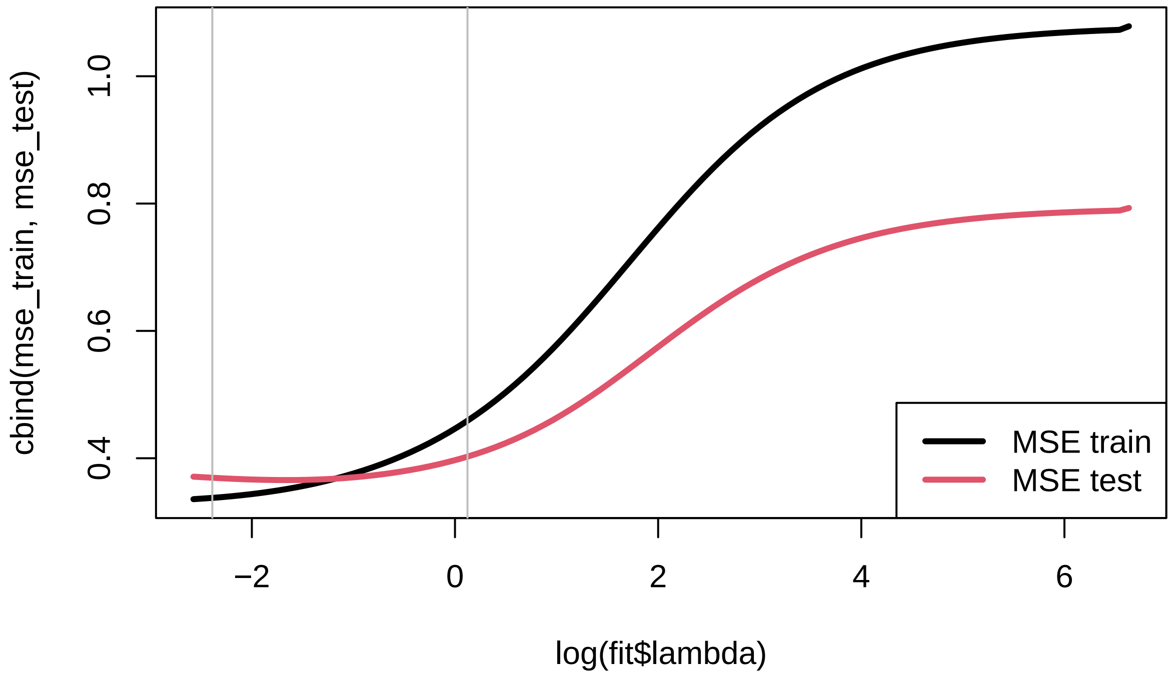
```
y_train_predicted <- predict(fit, s = opt_lambda, newx = x.train)
```

In a similar manner to the above, we can use the predict function to predict the `lpsa` values for the test data using the covariates of the test set.

```
y_test_predicted <- predict(fit, s = opt_lambda, newx = x.test)
```

We now calculate the mean squared (MSE) prediction error for both the training and testing data, using all values of lambda instead of just the optimal one. Note you can find all the lambda's used in `fit$lambda`, and, MSE can be computed like `mse_train = mean((y_train_predicted-y.train)^2)`.

```
# Initialize empty arrays
mse_test=mse_train=c()

# Loop through possible lambda values
for (i in 1:length(fit$lambda)){

  # Get the current lambda value
  lam=fit$lambda[i]

  # Get the training data predication
  y_train_predicted <- predict(fit, s = lam, newx = x.train)
  # Get the testing data predication
  y_test_predicted <- predict(fit, s = lam, newx = x.test)

  # Get the mean squared error for training
  mse_train[i]=mean((y.train-y_train_predicted)^2)

  # Get the mean squared error for testing
  mse_test[i]=mean((y.test-y_test_predicted)^2)
}

# Plot the lambda values against the training and testing mean square errors
matplot(log(fit$lambda),cbind(mse_train,mse_test),type='l',lwd=3,lty=1)

# Add a legend
legend('bottomright', legend=c('MSE train','MSE test'),col=1:2,lwd=3,lty=1)

# Add a vertical line at the lambda values
abline(v=log(c(opt_lambda,cv_fit$lambda.1se)),col='gray')
```



Our analysis concluded that we should use low values of log(lambda) for our analysis. Around these values (shown as horizontal lines) we have a low MSE for both the training and, more importantly test data. For larger values of lambda, we saw a larger MSE on both the training set and the test set, thus confirming our choice of lambda as reasonable.

By changing the value of alpha in the `glmnet` function, see if you can now run an LASSO analysis for the above data.

```r
# Write your code here...
```

Ensure you understand the outputs of your analysis before proceeding.

Finally, by changing the value of alpha once more, perform a elastic net regression on this dataset.

```r
# Write your code here...
```

By investigating this dataset, and based on the context you have been given, which of the three regression methods do you believe might be most appropriate for this analysis? Which variables have predictive power under your chosen model?

It has been suggested that maybe the above model was not accounting for various interactions between the predictors. To rectify this, a new file has been created, composed of 64 predictor variables, each composed of combinations of the predictors you worked with above.

> **Warning:** The variables in the new file do not have the same scaling as the first file. Note that it is important to scale your variables (see the prostate example) before performing a regularised regression, as your penalty term is sensitive to the magnitude of your data.

```r
diabetes_with_quad <- read.csv("diabetes_with_quad.csv", header = TRUE)
```

Using these new variables, alongside `diabetes$y` run new ridge, LASSO and elastic net analyses. Does the inclusion of these variables in your analysis improve the model fit?

```r
# Write your code here...
```

Acknowledgements

Portions of this lab are drawn from materials by Thomas E. Nichols, Marco Palma & Chieh-Hsi (Jessie) Wu.