# STAT34800 HW2

*Sarah Adilijiang*

## Problem A

Run the given code first.

### Simulate some data

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

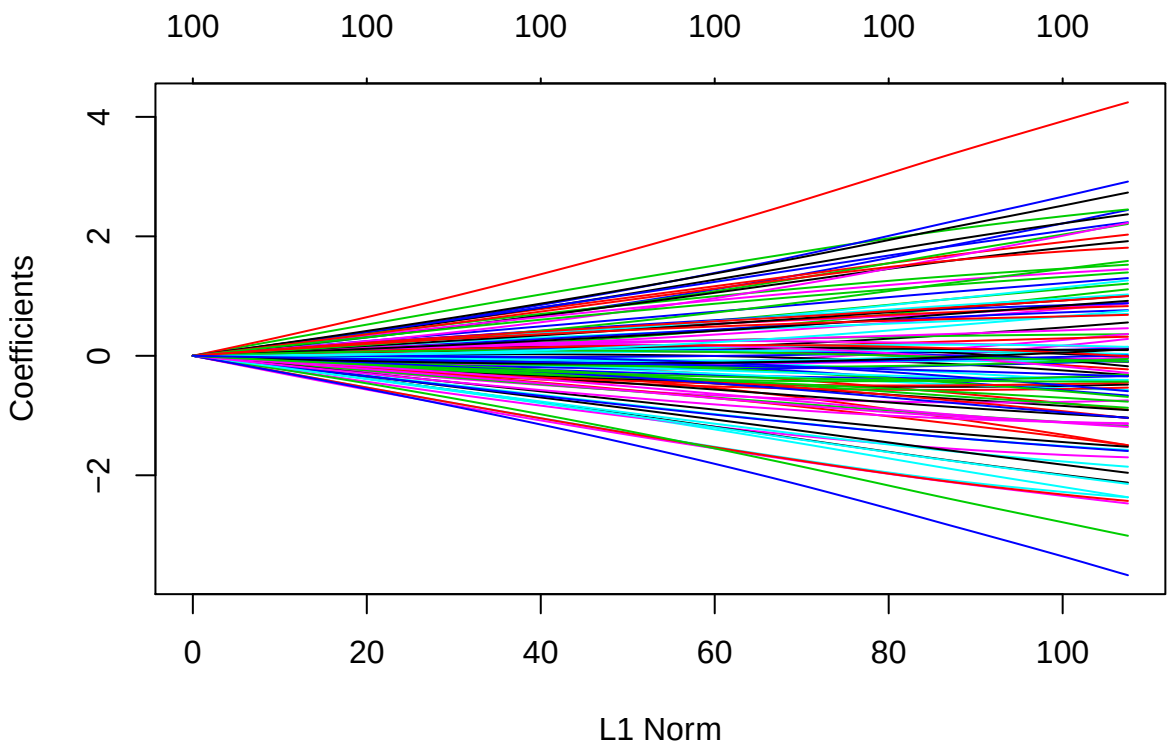
```
## Loaded glmnet 2.0-16
```

```
set.seed(1)
p = 100
n = 500
X = matrix(rnorm(n*p),ncol=p)
b = rnorm(p)
e = rnorm(n,0,sd=25)
Y = X %*% b + e
```

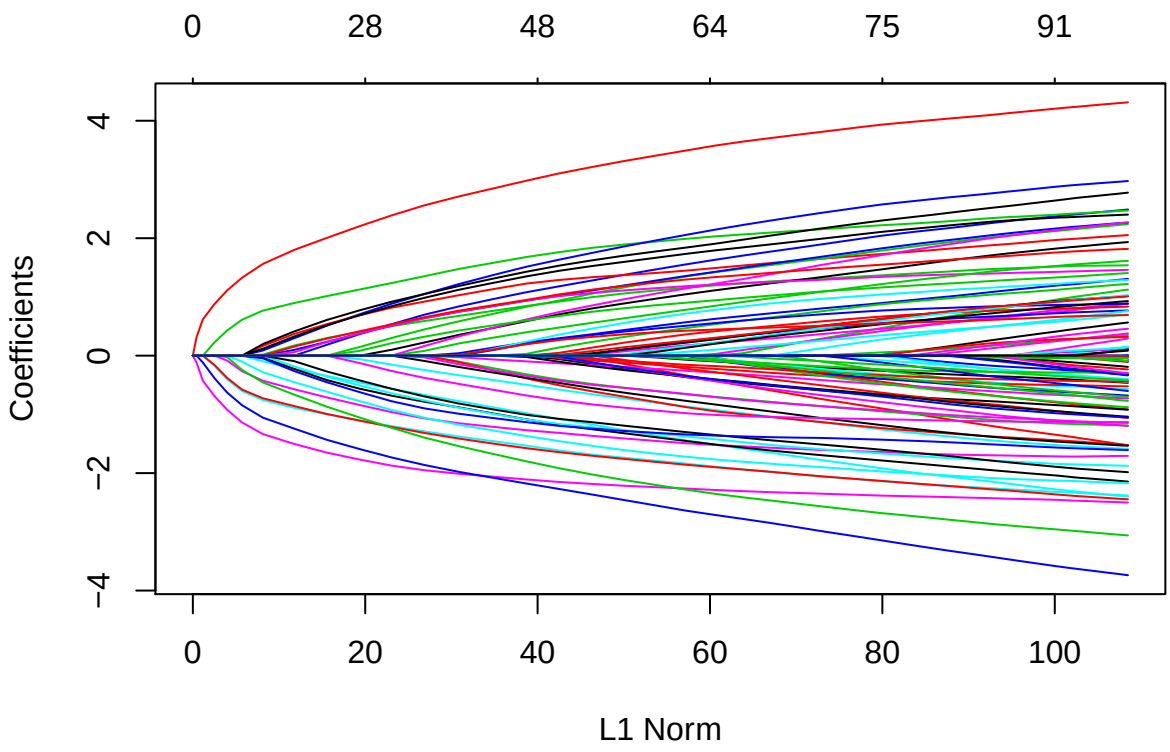
Now fit **OLS**, **Ridge regression** and **Lasso**, and see some basic plots.

```
# OLS
Y.ols = lm(Y~X)

# Ridge
Y.ridge = glmnet(X,Y,alpha=0) # alpha=0 is Ridge penalty
plot(Y.ridge)
```

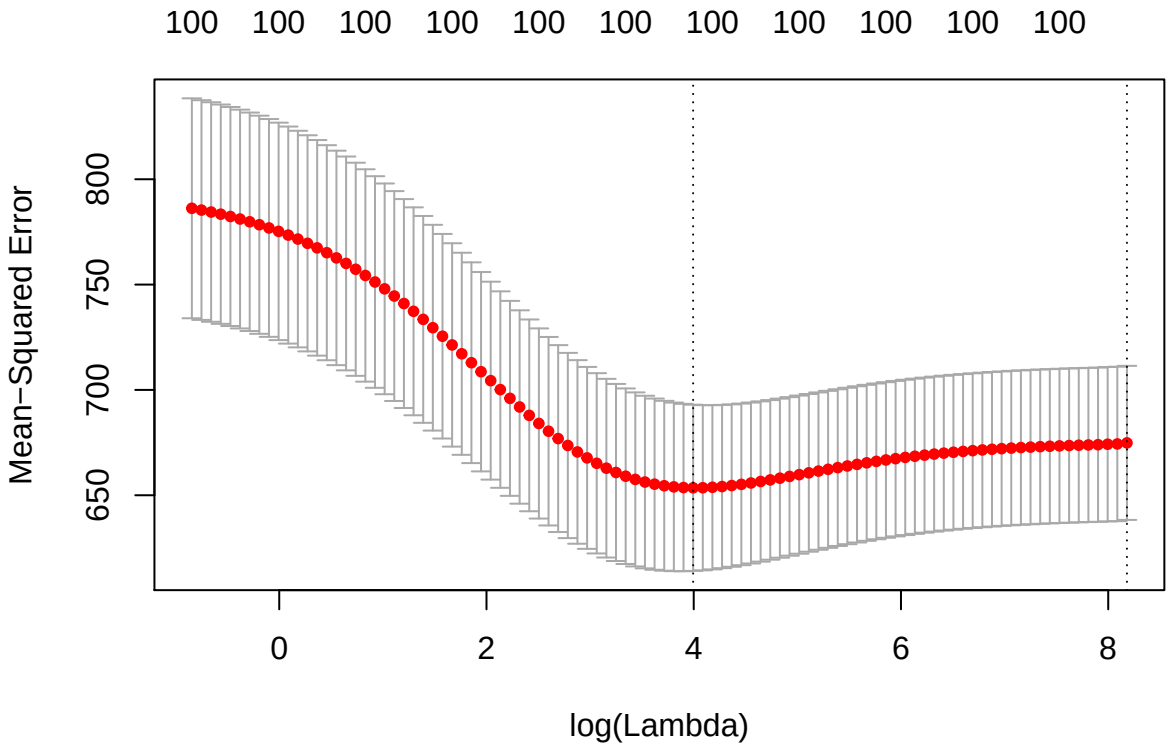



```
# Lasso
Y.lasso = glmnet(X,Y,alpha=1) # alpha=1 is Lasso penalty (default)
plot(Y.lasso)
```

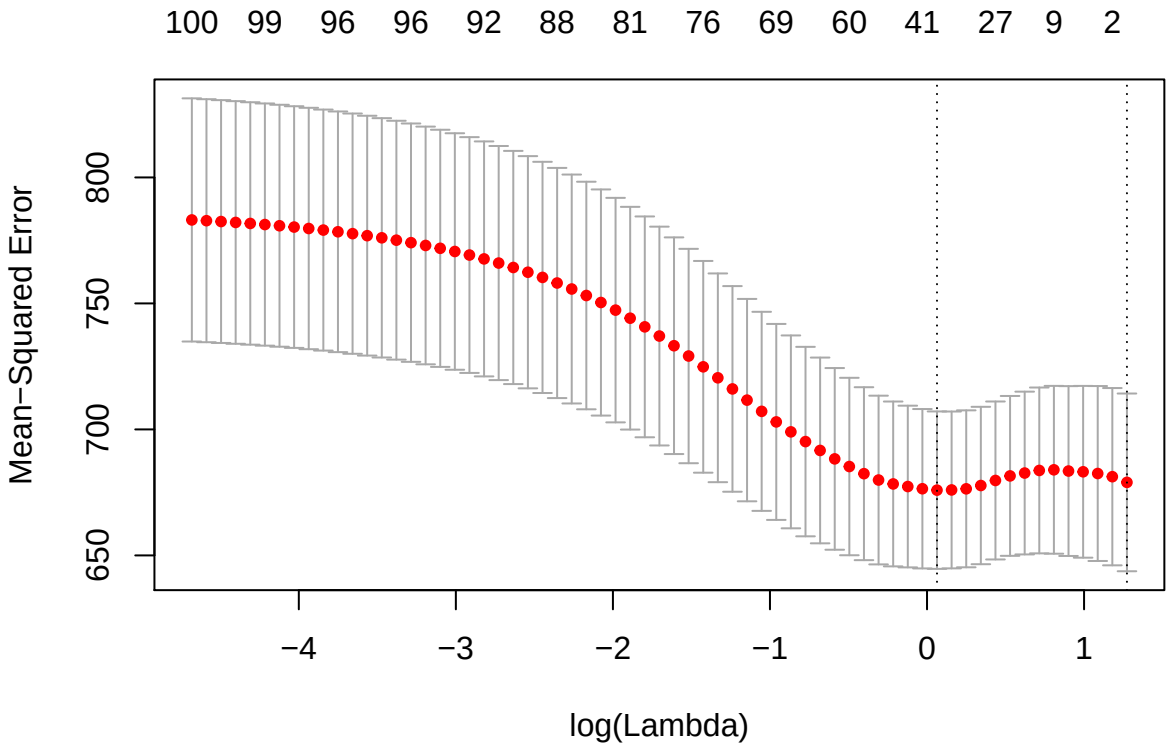



The library also allows you to run **Cross-Validation** easily:

```
# Ridge with 10-fold CV
cv.ridge = cv.glmnet(X,Y,alpha=0) # default: nfolds=10
plot(cv.ridge)
```




```
# Lasso with 10-fold CV
cv.lasso = cv.glmnet(X,Y,alpha=1) # default: nfolds=10
plot(cv.lasso)
```




**Measure accuracy of coefficients**

Extract coefficients from best CV fits.

```
b.ridge = predict(Y.ridge, type="coefficients", s=cv.ridge$lambda.min) # includes intercept
b.lasso = predict(Y.lasso, type="coefficients", s=cv.lasso$lambda.min)
b.ols = Y.ols$coefficients
```

Note that the fits include an **intercept** (unregularized, equal to the **mean of Y**).

```
length(b.lasso)
```

```
## [1] 101
```

```
b.lasso[1]
```

```
## [1] -1.38244
```

```
mean(Y)
```

```
## [1] -1.233957
```

Compare the estimated coefficients with the true coefficients:

```
# true coefficients
btrue = c(0,b) # Here the 0 is the intercept (true value 0)

# This is error if we just estimate 0 for everything and ignore data.
# It is better than OLS!
sum((btrue-0)^2)
```

```
## [1] 85.30411
```

```
# coefficients estimated by OLS, Ridge, Lasso
sum((btrue-b.ols)^2)
```

```
## [1] 138.2715
```

```
sum((btrue-b.ridge)^2)
```

```
## [1] 56.99797
```

```
sum((btrue-b.lasso)^2)
```

```
## [1] 68.7212
```

# Questions

## (i)

### (1) Simulate test data & check prediction error

```
# simulate test data from same model
p = 100       # same data dimension
n.test = 200  # change the number of test size
X.test = matrix(rnorm(n.test*p),ncol=p)
b.test = b    # same b as in the training data set
e.test = rnorm(n.test,0,sd=25)
Y.test = X.test %*% b.test + e.test
```

```
# prediction error MSE of OLS on test data
Y.test_pred.ols = cbind(rep(1,n.test),X.test) %*% b.ols
sum((Y.test_pred.ols-Y.test)^2)/n.test
```

```
## [1] 810.2715
```

```
# prediction error MSE of Ridge on test data
Y.test_pred.ridge = predict(Y.ridge, X.test, type="response", s=cv.ridge$lambda.min)
sum((Y.test_pred.ridge-Y.test)^2)/n.test
```

```
## [1] 695.8569
```

```
# prediction error MSE of Lasso on test data
Y.test_pred.lasso = predict(Y.lasso, X.test, type="response", s=cv.lasso$lambda.min)
sum((Y.test_pred.lasso-Y.test)^2)/n.test
```

```
## [1] 710.5617
```

```
# average MSE of leave-one-out CV OLS on training data
MSE_store = rep(NA,n)
for (i in 1:n) {
    model = lm(Y[-i]~X[-i,])
    pred = c(1,X[i,]) %*% model$coefficients
    MSE_store[i] = (pred-Y[i])^2
}
mean(MSE_store)
```

```
## [1] 764.0673
```

```
# minimal average MSE of CV Ridge on training data
min(cv.ridge$cvm)
```

```
## [1] 653.5249
```

```
# minimal average MSE of CV Lasso on training data
min(cv.lasso$cvm)
```

```
## [1] 675.8959
```

**Answer**:

(1) The prediction error MSE of OLS model on the test data is 810.2715, while the average MSE of leave-one-out CV OLS model on the training data is 764.0673.

(2) The prediction error MSE of Ridge regularization model on the test data is 695.8569, while the minimal average MSE of 10-fold CV Ridge model on the training data is 653.5249.

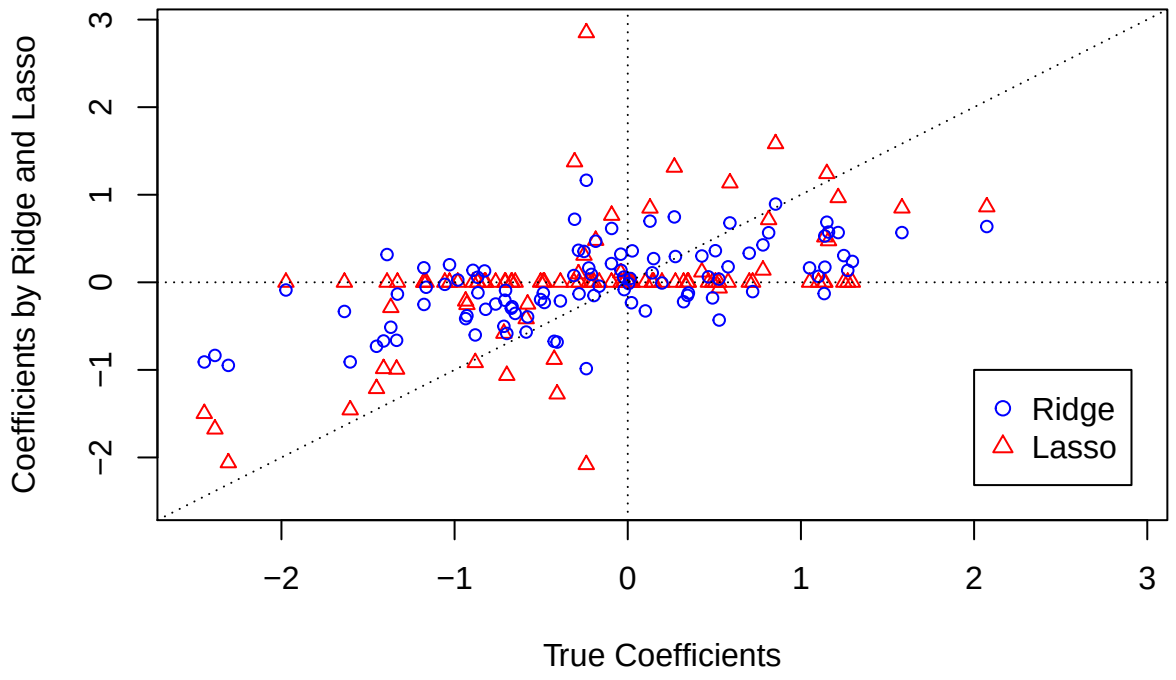
(3) The prediction error MSE of Lasso regularization model on the test data is 710.5617, while the minimal average MSE of 10-fold CV Lasso model on the training data is 675.8959.

As a result, we can see that the prediction error of different methods is comparable with the average error of CV results. Therefore, the CV method provides a good way to estimate the prediction error in the test set.

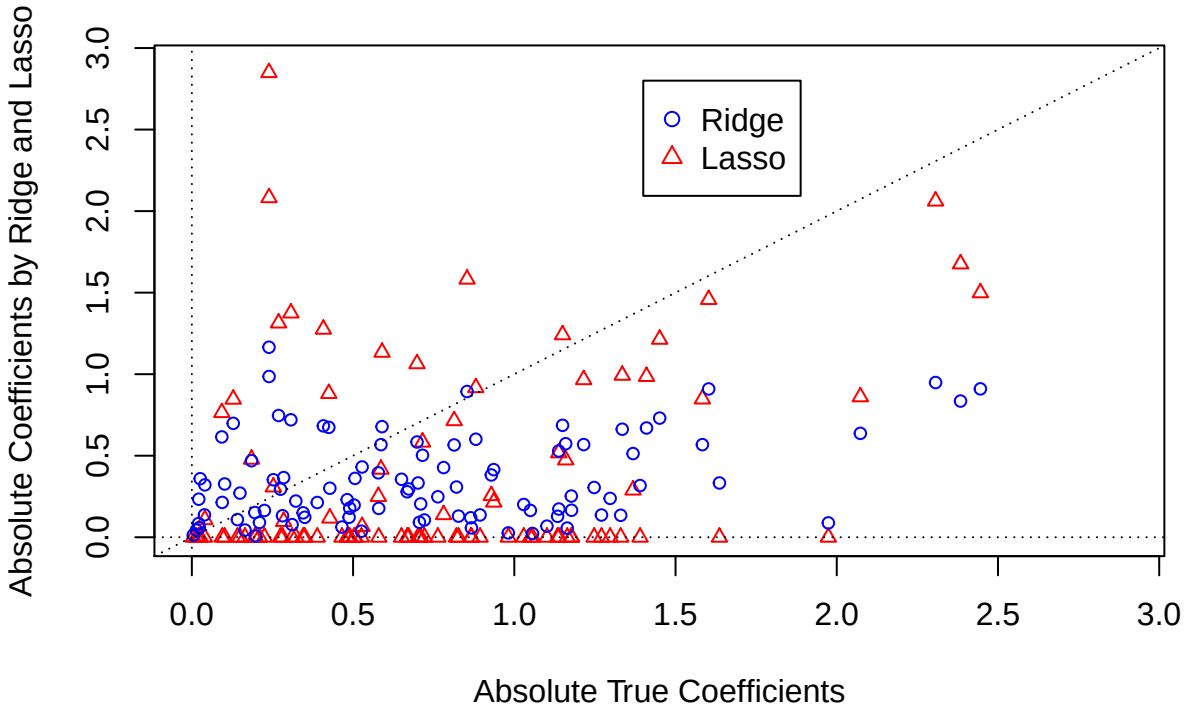
Also, we can see that both Ridge and Lasso regularization methods provide lower prediction error on the test data than the OLS model, thus they are both good ways to avoid overfitting.

### (2) Plot coefficients of Ridge & Lasso against true coefficients

```
# plot the coefficients of Ridge & Lasso against true coefficients
plot(b.lasso[-1]~b, pch=2, col=2, cex=0.8, xlim=c(-2.5,2.9), ylim=c(-2.5,2.9),
     xlab="True Coefficients", ylab="Coefficients by Ridge and Lasso")
points(b.ridge[-1]~b, pch=1, col=4, cex=0.8)
abline(0,1, lty=3)  # line y=x
abline(h=0, lty=3)  # line y=0
abline(v=0, lty=3)  # line x=0
legend(2,-1, legend=c("Ridge","Lasso"), pch=c(1,2), col=c(4,2))
```




```
# plot the absolute value of coefficients
plot(abs(b.lasso[-1])~abs(b), pch=2, col=2, cex=0.8, xlim=c(0,2.9), ylim=c(0,2.9),
     xlab="Absolute True Coefficients", ylab="Absolute Coefficients by Ridge and Lasso")
points(abs(b.ridge[-1])~abs(b), pch=1, col=4, cex=0.8)
abline(0,1, lty=3)  # line y=x
abline(h=0, lty=3)  # line y=0
abline(v=0, lty=3)  # line x=0
legend(1.4,2.8, legend=c("Ridge","Lasso"), pch=c(1,2), col=c(4,2))
```




```
# check how many coefficients are cut to zero & how many get shrunk
c(sum(b.ridge[-1]==0), sum(b.ridge[-1]!=0), sum(abs(b.ridge[-1])<abs(b) & b.ridge[-1]!=0))
```

```
## [1]   0 100  75
```

```
c(sum(b.lasso[-1]==0), sum(b.lasso[-1]!=0), sum(abs(b.lasso[-1])<abs(b) & b.lasso[-1]!=0))
```

```
## [1] 61 39 23
```

**Discussion**:

(1) For Ridge regression model, none (0/100) of the coefficients are cut to zero, but most (75/100) of the coefficients get shrunk to lower values. Especially when the absolute values of true coefficients are large, they get shrunk much more than those small ones.

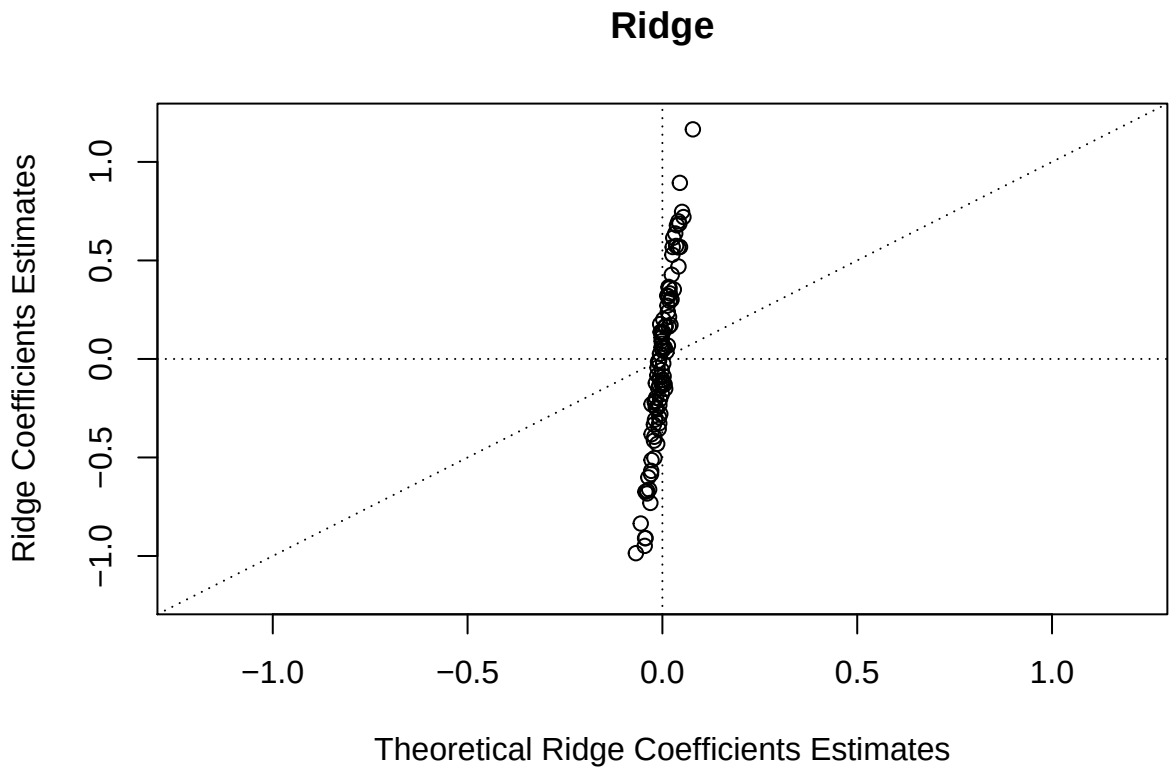
(2) For Lasso model, most (61/100) of the coefficients are cut to zero, and within the remaining coefficients, most (23/39) of them get shrunk as well. However, we can see that these coefficients get shrunk by a strength that is much less than in the Ridge model. So the Ridge model shrank the coefficients harder than the Lasso model. Plus, on the contrary, some of small coefficients in the Lasso model get manified to larger values.

### (3) Plot coefficients of Ridge & Lasso against theoretical coefficients

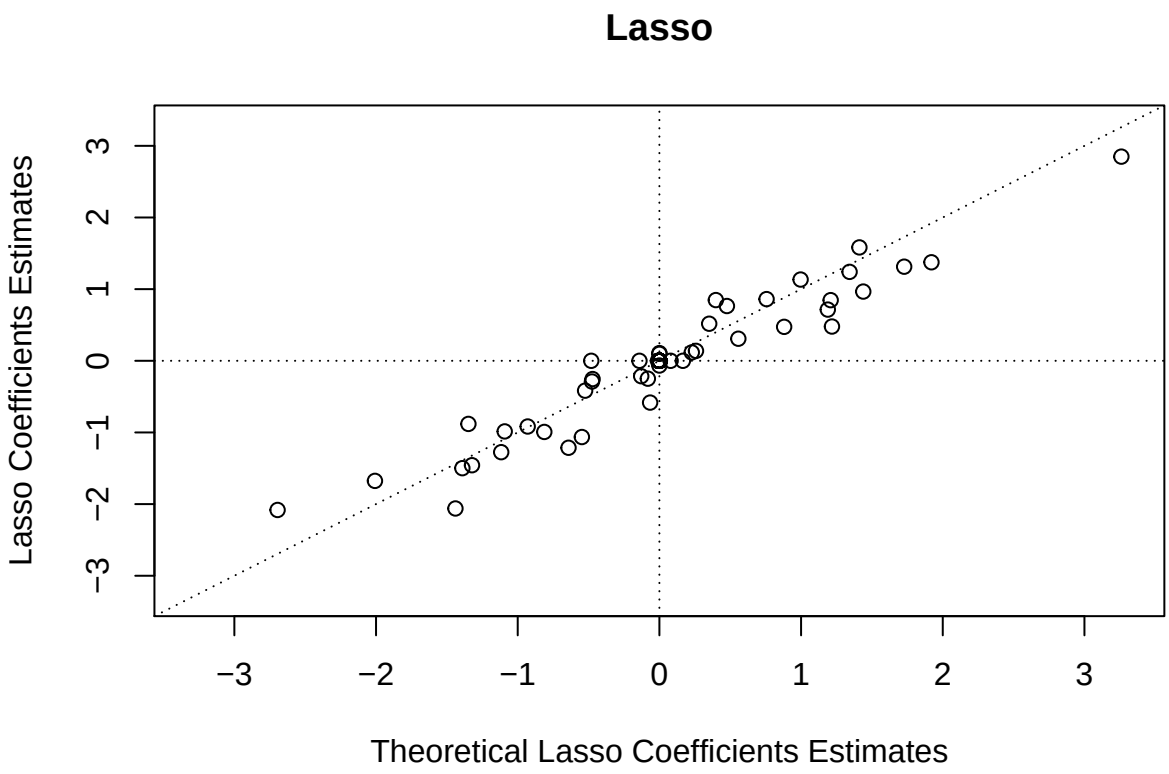
```
# calculate the theoretical estimates of Ridge & Lasso
b.lasso.theo = sign(b.ols)*ifelse((abs(b.ols)-cv.lasso$lambda.min)>0,abs(b.ols)-cv.lasso$lambda.min,0)
b.ridge.theo = b.ols/(1+cv.ridge$lambda.min)

# plot coefficients of Ridge against its theoretical coefficients
plot(b.ridge[-1]~b.ridge.theo[-1], main="Ridge", xlim=c(-1.2,1.2), ylim=c(-1.2,1.2),
```

```
     xlab="Theoretical Ridge Coefficients Estimates", ylab="Ridge Coefficients Estimates")
abline(0,1, lty=3)  # line y=x
abline(h=0, lty=3)  # line y=0
abline(v=0, lty=3)  # line x=0
```




```
# plot coefficients of Lasso against its theoretical coefficients
plot(b.lasso[-1]~b.lasso.theo[-1], main="Lasso", xlim=c(-3.3,3.3), ylim=c(-3.3,3.3),
     xlab="Theoretical Lasso Coefficients Estimates", ylab="Lasso Coefficients Estimates")
abline(0,1, lty=3)  # line y=x
abline(h=0, lty=3)  # line y=0
abline(v=0, lty=3)  # line x=0
```




**Discussion**:

In the special case of $X$ having orthonormal columns, the theoretical estimates of Ridge and Lasso should be:

$$L_1\ (Lasso):\quad \hat{\beta}_j^1 = sign(\hat{\beta}_j)\ (|\hat{\beta}_j| - \lambda)_+$$

$$L_2\ (Ridge):\quad \hat{\beta}_j^2 = \hat{\beta}_j\ /\ (1+\lambda)$$

where $\hat{\beta}_j$ is the estimates of non-regularized OLS model.

Here in our simulated data, the predictors are not orthogonal, so the theory will certainly not hold precisely.

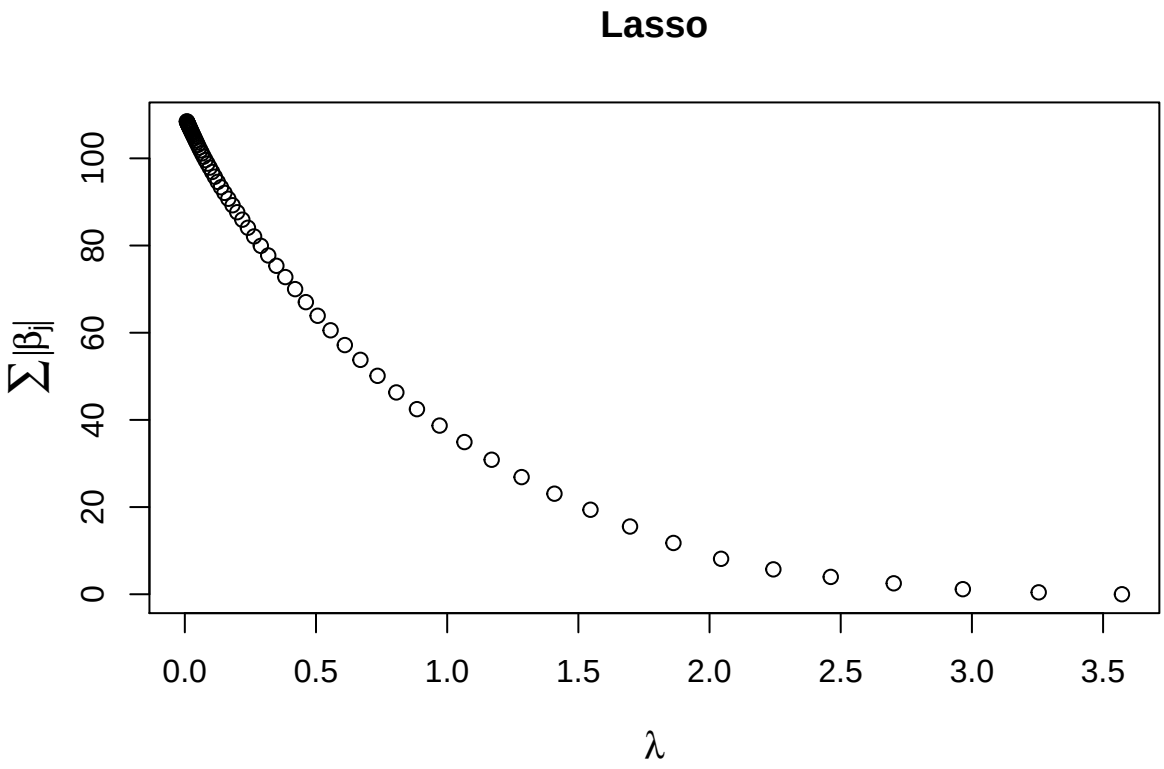
(1) But in the Lasso model, we see that the points lie mostly close to the y=x line. Therefore, the theory holds approximately when the predictors are not orthogonal.

(2) However, in the Ridge regression model, the points lie mostly in a line that is far away form the y=x line and the slope is much larger than 1. Therefore, the theory does not hold approximately when the predictors are not orthogonal, and the coefficients are much larger than their corresponding theoretical estimates when the predictors are actually orthogonal.

### (4) Plot penalty terms of the coefficients against Lasso & Ridge lambda's

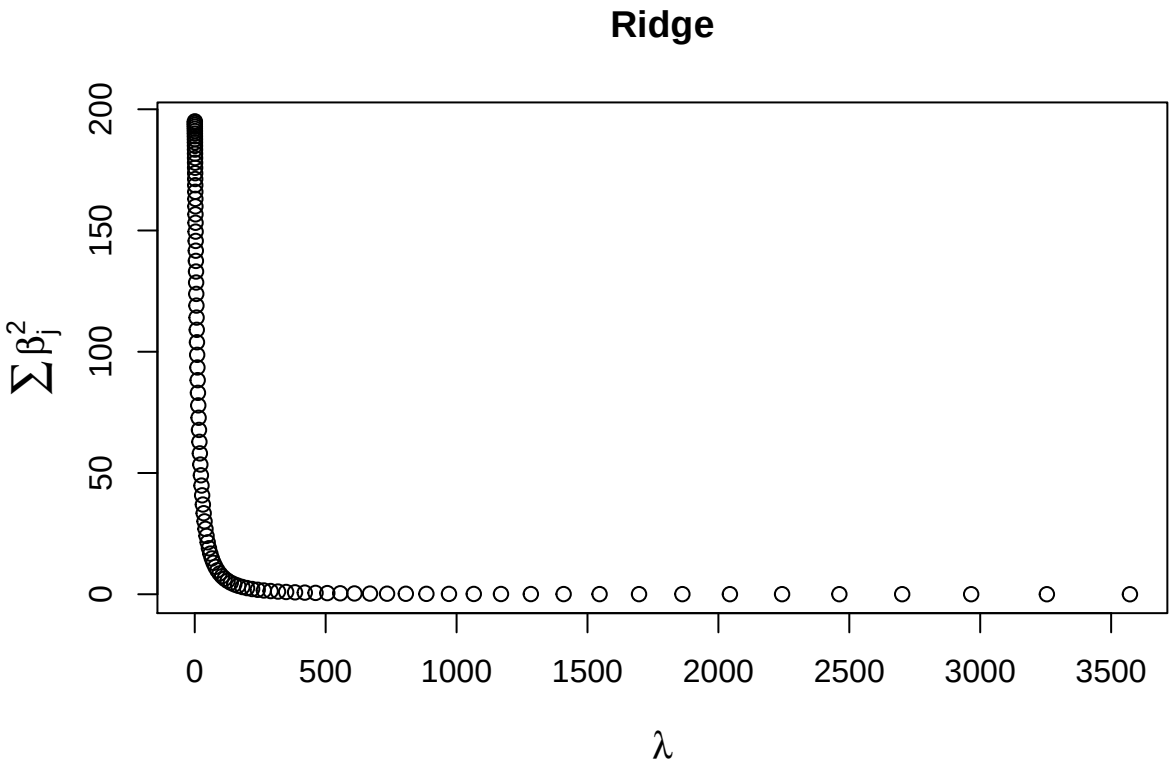
```
# Y.lasso$lambda # 67 lambda values (vector)
# Y.lasso$beta   # 100*67 matrix  (100 beta, 67 lambda)
sum_abs_b.lasso = apply(Y.lasso$beta, 2, function(x){sum(abs(x))})

# plot sum of absolute values of the coefficients against Lasso lambda's
plot(sum_abs_b.lasso~Y.lasso$lambda, main="Lasso", xlab=expression(lambda), ylab="", cex.lab=1.2)
title(ylab=expression(sum(abs(beta[j]))), line=2.3, cex.lab=1.2)
```




```
# Y.ridge$lambda  # 100 lambda values (vector)
# Y.ridge$beta    # 100*100 matrix  (100 beta, 100 lambda)
sum_square_b.ridge = apply(Y.ridge$beta, 2, function(x){sum(x^2)})

# plot sum of absolute values of the coefficients against Ridge lambda's
plot(sum_square_b.ridge~Y.ridge$lambda, main="Ridge", xlab=expression(lambda), ylab="", cex.lab=1.2)
title(ylab=expression(sum(beta[j]^2)), line=2.3, cex.lab=1.2)
```




**Answer**:

We can see from the plot that indeed the sum of absolute values of the coefficients is decreasing as the $\lambda$ value of Lasso increases, and it finally gets to zero.

Also, I plotted a similar figure for Ridge regression. It is also shown that the sum of squares of the coefficients is decreasing as the $\lambda$ value of Ridge regression increases, and it also finally gets to zero.

### (5) Brief Summary

In the previous questions, I learned several things:

(1) In question (1), I evaluated the prediction error of different models on the test data, and found that the prediction error of different methods are comparable with the average error of CV results. Therefore, the CV method provides a good way to estimate the prediction error on the test data.

(2) In question (2), I examined the coefficients of Ridge and Lasso methods against the true coefficients, and found that both methods have a shrinkage bias, though in different ways. Lasso prefers to cut the coefficients to zeros, while Ridge regression prefers to shrink the coefficients harder than the Lasso model, however, not cutting them to zeros.

(3) In question (3), I examined the coefficients of Ridge and Lasso methods against their theoretical values if the predictors are orthogonal. The results show that in the Lasso model the theory will hold approximately, while in the Ridge model it will not.

(4) In question (4), I checked the trends of penalty terms of Ridge and Lasso methods when the $\lambda$ value increases. The results show that indeed both the L1 & L2 penalty terms in two models, respectively, decreases as $\lambda$ increases.

### (ii)

```
# simulate new data with sparse setting in new b
set.seed(123)
p = 100
n = 500
X.new = matrix(rnorm(n*p),ncol=p)
b.new = c(rnorm(10), rep(0,p-10)) # only 10 non-zero true coefficients
e.new = rnorm(n,0,sd=25)
Y.new = X.new %*% b.new + e.new
```

```
# fit different methods
Y.ols.new = lm(Y.new~X.new)
Y.ridge.new = glmnet(X.new,Y.new,alpha=0) # alpha=0 is Ridge penalty
Y.lasso.new = glmnet(X.new,Y.new,alpha=1) # alpha=1 is Lasso penalty (default)

# find best lambda by 10-fold CV
cv.ridge.new = cv.glmnet(X.new,Y.new,alpha=0) # default: nfolds=10
cv.lasso.new = cv.glmnet(X.new,Y.new,alpha=1) # default: nfolds=10

# extract coefficients from best CV fits
b.ridge.new = predict(Y.ridge.new, type="coefficients", s=cv.ridge.new$lambda.min) # includes intercept
b.lasso.new = predict(Y.lasso.new, type="coefficients", s=cv.lasso.new$lambda.min)
b.ols.new = Y.ols.new$coefficients
```

#### (1) Measure accuracy of coefficients

```
# true coefficients
btrue.new = c(0,b.new)    # 0 is the true value of intercept

# the error if we just estimate 0 for everything and ignore data
sum((btrue.new-0)^2)
```

```
## [1] 13.81173
```

```
# coefficients estimated by OLS, Ridge, Lasso
sum((btrue.new-b.ols.new)^2)
```

```
## [1] 169.0517
```

```
sum((btrue.new-b.ridge.new)^2)
```

```
## [1] 14.26103
```

```
sum((btrue.new-b.lasso.new)^2)
```

```
## [1] 12.12208
```

#### (2) Compare the average MSE error of CVs

```
# average MSE of leave-one-out CV OLS
MSE_store = rep(NA,n)
for (i in 1:n) {
    model = lm(Y.new[-i]~X.new[-i,])
    pred = c(1,X.new[i,]) %*% model$coefficients
    MSE_store[i] = (pred-Y.new[i])^2
```

```
}
mean(MSE_store)
```

```
## [1] 804.5315
```

```
# minimal average MSE of CV Ridge
min(cv.ridge.new$cvm)
```

```
## [1] 676.9052
```

```
# minimal average MSE of CV Lasso
min(cv.lasso.new$cvm)
```

```
## [1] 672.471
```

**Discussion**:

(1) Here, I first measured the accuracy of coefficients. The results show that, in this sparse setting, Lasso provides more accurate coefficients than Ridge regression, while the OLS model is much worse than these two regularization methods.

(2) Since in the previous question, we have checked that the prediction error of different methods on a new test data is comparable with the CV results, so here I also compared the average MSE error of CV results in different methods. The results again show that Lasso provides better predictions than Ridge regression in the sparse setting, and the OLS model works much worse than these two regularization methods in predictions.

# Problem B

## (i)

### Conjugate Prior

Prior distribution of $\mu$ is: $\mu \sim Gamma(n,\ \lambda)$, so:

$$P(\mu) \ = \ \frac{\lambda^n}{\Gamma(n)}\mu^{n-1}e^{-\lambda\mu} \qquad (\mu \geq 0)$$

Since $X|\mu \sim Poisson(\mu)$, so the likelihood function is:

$$P(X = x|\mu) \ = \ \frac{e^{-\mu}\ \mu^x}{x!}$$

Thus the posterior distribution of $\mu$ is:

$$P(\mu|X = x) \propto \ P(X = x|\mu)P(\mu) \propto \ \frac{e^{-\mu}\mu^x}{x!}\frac{\lambda^n}{\Gamma(n)}\mu^{n-1}e^{-\lambda\mu} \propto \ \frac{\lambda^n}{x!\ \Gamma(n)}\mu^{n+x-1}e^{-(\lambda+1)\mu} \qquad (\mu \geq 0)$$

It is obvious that this posterior distribution of $\mu$ is also a Gamma distribution: $\mu|X = x \sim Gamma(n+x, \lambda+1)$, which is the same family as the prior distribution of $\mu$.

Therefore, we have proved that: "the Gamma distribution is the conjugate prior distribution for a Poisson mean".

## (ii)

### (1) Posterior Distribution for Normal Mean

Prior distribution of $\theta$ is: $\theta \sim N(\mu_0,\ 1/\tau_0)$, so:

$$P(\theta) \propto \exp\{-0.5\tau_0(\theta - \mu_0)^2\} \qquad (\theta \in (-\infty, \infty))$$

Since $X|\theta \sim N(\theta, 1/\tau)$, so the likelihood function:

$$P(X = x|\theta) \propto \exp\{-0.5\tau(x-\theta)^2\}$$

Thus the posterior distribution of $\theta$ is:

$$P(\theta|X = x) \ \propto \ P(X = x|\theta)\ P(\theta) \ \propto \ \exp\{-0.5\tau(x-\theta)^2\}\ \exp\{-0.5\tau_0(\theta-\mu_0)^2\}$$

$$\propto \exp\{-0.5(\tau+\tau_0)\theta^2 + (\tau x + \tau_0\mu_0)\theta\} \qquad (\theta \in (-\infty, \infty))$$

If $P(x) \propto exp\{-0.5Ax^2 + Bx\} \ \ (x \in (-\infty, \infty))$, then $X$ has a Normal distribution: $X \sim N(\mu, 1/\tau)$, where $\tau = A$ and $\mu = B/A$.

Therefore, the posterior distribution of $\theta$ is: $\theta|X \sim N(\mu_1, 1/\tau_1)$, where

$$\tau_1 = \tau + \tau_0$$

$$\mu_1 = \frac{\tau X + \tau_0\mu_0}{\tau+\tau_0} = \frac{\tau}{\tau+\tau_0}X + \frac{\tau_0}{\tau+\tau_0}\mu_0 \ = \ wX + (1-w)\mu_0$$

and

$$w = \frac{\tau}{\tau+\tau_0} = \tau/\tau_1$$

**(2) Shrinkage towards Prior Mean**

The posterior mean of $\theta$ is:

$$E(\theta|X) = \mu_1 = wX + (1-w)\mu_0 = \mu_0 + (X-\mu_0)w, \qquad w = \frac{\tau}{\tau+\tau_0} \in (0,1)$$

So when $X > \mu_0$, we have: $\mu_0 < \mu_1 < X$; when $X = \mu_0$, we have: $\mu_1 = \mu_0$; when $X < \mu_0$, we have: $X < \mu_1 < \mu_0$.

Note that the maximum likelihood estimate of $\theta$ from the data is: $\hat{\theta} = X$.

Therefore, the posterior mean $\mu_1$ is always shrinking the estimate of $\theta$ towards prior mean $\mu_0$.

# Problem C

## (a) Log-likelihood

Prior distribution of $\theta_j$ is: $\theta_j|\mu, \sigma \sim N(\mu, \sigma^2)$, so:

$$P(\theta_j|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\ \exp\{-\frac{(\theta_j-\mu)^2}{2\sigma^2}\} \qquad (\theta_j \in (-\infty,\infty))$$

Since $X_j|\theta_j, s_j \sim N(\theta_j, s_j^2)$, so:

$$P(X_j = x_j|\theta_j, s_j) = \frac{1}{\sqrt{2\pi s_j^2}}\ \exp\{-\frac{(x_j-\theta_j)^2}{2s_j^2}\}$$

Hence:

$$P(X_j = x_j|\mu,\sigma,s_j) = \int_{-\infty}^{\infty} P(X_j = x_j|\theta_j, s_j)\ P(\theta_j|\mu,\sigma)\ d\theta_j$$

$$= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi s_j^2}} \exp\{-\frac{(x_j-\theta_j)^2}{2s_j^2}\}\ \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(\theta_j-\mu)^2}{2\sigma^2}\}\ d\theta_j$$

$$= \frac{1}{2\pi\sigma s_j} \int_{-\infty}^{\infty} \exp\{-\frac{(x_j-\theta_j)^2}{2s_j^2} - \frac{(\theta_j-\mu)^2}{2\sigma^2}\}\, d\theta_j$$

$$= \frac{1}{2\pi\sigma s_j} \int_{-\infty}^{\infty} \exp\{-\frac{\left(\theta_j - \frac{\sigma^2 x_j + s_j^2\mu}{\sigma^2+s_j^2}\right)^2}{2\frac{\sigma^2 s_j^2}{\sigma^2+s_j^2}} - \frac{(x_j-\mu)^2}{2(\sigma^2+s_j^2)}\}\, d\theta_j$$

$$= \frac{1}{\sqrt{2\pi(\sigma^2+s_j^2)}} \exp\{-\frac{(x_j-\mu)^2}{2(\sigma^2+s_j^2)}\} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2 s_j^2/(\sigma^2+s_j^2)}} \exp\{-\frac{\left(\theta_j - \frac{\sigma^2 x_j + s_j^2\mu}{\sigma^2+s_j^2}\right)^2}{2\frac{\sigma^2 s_j^2}{\sigma^2+s_j^2}}\}\, d\theta_j$$

$$= \frac{1}{\sqrt{2\pi(\sigma^2+s_j^2)}} \exp\{-\frac{(x_j-\mu)^2}{2(\sigma^2+s_j^2)}\}$$

Therefore, we have:

$$X_j|\mu,\sigma,s_j \;\sim\; N(\mu,\; \sigma^2+s_j^2)$$

Then the log-likelihood is:

$$l(\mu,\sigma) = \log L(\mu,\sigma) = \log \prod_j P(X_j = x_j|\mu,\sigma,s_j) = \sum_j \log P(X_j = x_j|\mu,\sigma,s_j)$$

$$= \sum_j \log\left(\frac{1}{\sqrt{2\pi(\sigma^2+s_j^2)}} \exp\{-\frac{(x_j-\mu)^2}{2(\sigma^2+s_j^2)}\}\right) \;=\; \sum_j \left(-\frac{1}{2}\log(2\pi(\sigma^2+s_j^2)) - \frac{(x_j-\mu)^2}{2(\sigma^2+s_j^2)}\right)$$

## (b) Posterior Mean

From question (a), we can easily compute the posterior density of $\theta_j$, which is:

$$P(\theta_j|X_j = x_j,\mu,\sigma) \;\propto\; P(X_j = x_j|\theta_j)\; P(\theta_j|\mu,\sigma)$$

$$\propto \frac{1}{\sqrt{2\pi s_j^2}} \exp\{-\frac{(x_j-\theta_j)^2}{2s_j^2}\} \;\frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{(\theta_j-\mu)^2}{2\sigma^2}\}$$

$$\propto \frac{1}{\sqrt{2\pi(\sigma^2+s_j^2)}} \exp\{-\frac{(x_j-\mu)^2}{2(\sigma^2+s_j^2)}\} \;\frac{1}{\sqrt{2\pi\sigma^2 s_j^2/(\sigma^2+s_j^2)}} \exp\{-\frac{\left(\theta_j - \frac{\sigma^2 x_j + s_j^2\mu}{\sigma^2+s_j^2}\right)^2}{2\frac{\sigma^2 s_j^2}{\sigma^2+s_j^2}}\}$$

Since $\int_{-\infty}^{\infty} P(\theta_j|X_j = x_j,\mu,\sigma) = 1$, thus we can get the posterior density:

$$P(\theta_j|X_j = x_j,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2 s_j^2/(\sigma^2+s_j^2)}} \exp\{-\frac{\left(\theta_j - \frac{\sigma^2 x_j + s_j^2\mu}{\sigma^2+s_j^2}\right)^2}{2\frac{\sigma^2 s_j^2}{\sigma^2+s_j^2}}\} \qquad (\theta_j \in (-\infty,\infty))$$

This means that the posterior distribution of $\theta_j$ is also a Normal distribution:

$$\theta_j|X_j,\mu,\sigma \;\sim\; N(\frac{\sigma^2 X_j + s_j^2\mu}{\sigma^2+s_j^2},\; \frac{\sigma^2 s_j^2}{\sigma^2+s_j^2})$$

Therefore, the posterior mean of $\theta_j$ is:

$$E(\theta_j|X_j,\mu,\sigma) \;=\; \frac{\sigma^2 X_j + s_j^2\mu}{\sigma^2+s_j^2}$$

### (c) Empirical Bayes Shrinkage function

```
## Empirical Bayes function for Normal Mean mu
## input: x=(x_1, ..., x_n), s=(s_1, ..., s_n)
## output: MLE prior parameters (mu_hat, sigma_hat); posterior mean


ebnm_normal = function(x,s) {
    # -log-likelihood function
    minus_log_Lik = function(pars) {
        - sum(-0.5*log(2*pi*(exp(2*pars[2])+s^2)) - 0.5*(x-pars[1])^2/(exp(2*pars[2])+s^2))
        # pars[1] = mu
        # pars[2] = eta = log(sigma)    #sigma^2 = exp(2*eta)   #non-negative constraint: sigma>0
    }

    # find MLE parameters that maximize log-likelihood
    mu = mean(x)       # initial mu
    sigma = mean(s)    # initial sigma
    mu_hat =  optim(par=c(mu,log(sigma)), minus_log_Lik)$par[1]
    eta_hat = optim(par=c(mu,log(sigma)), minus_log_Lik)$par[2]
    sigma_hat = sqrt(exp(2*eta_hat))

    # compute posterior mean
    pm = (sigma_hat^2*x + s^2*mu_hat)/(sigma_hat^2 + s^2)

    # output
    results = list(mu.hat=mu_hat, sigma.hat=sigma_hat, posterior.mean=pm)
    return(results)
}
```

### (d) Demonstration by simulation

```
set.seed(123)
compare_simulation = function(n, mu, sigma){
    # data simulation function
    theta = rnorm(n,mu,sigma)
    s = abs(rnorm(n))    # non-negative constraint: s>0
    x = rep(NA,n)
    for (i in 1:n) {
        x[i] = rnorm(1,theta[i],s[i])
    }

    # Empirical Bayes function for Normal Mean
    results = ebnm_normal(x=x, s=s)

    # truth vs estimates
    prior_par = data.frame(truth=c(mu, sigma),
                           estimates=c(results$mu.hat, results$sigma.hat))
    rownames(prior_par) = c("mu","sigma")
    print(prior_par)

    # accuary of estimates
    est_acc = data.frame( mle_MSE=sum((theta-x)^2),
```

```
                          pm_MSE=sum((theta-results$posterior.mean)^2) )
    print(est_acc)
}
```

```
# compare on several simulation data
compare_simulation(n=500, mu=1, sigma=1)
```

```
##       truth estimates
## mu        1  1.031561
## sigma     1  1.007480
##    mle_MSE   pm_MSE
## 1 556.0382 166.0288
```

```
compare_simulation(n=500, mu=1, sigma=5)
```

```
##       truth estimates
## mu        1  1.312205
## sigma     5  5.159448
##    mle_MSE   pm_MSE
## 1 471.8893 434.8586
```

```
compare_simulation(n=500, mu=1, sigma=10)
```

```
##       truth estimates
## mu        1  0.760178
## sigma    10  9.895828
##    mle_MSE   pm_MSE
## 1 439.3945 425.8921
```

**Commments**:

I tested on tree simulation data, with different settings, but the results are similar as shown below.

(1) Prior parameters: truth vs estimates

The estimates of $\mu$ and $\sigma$ (i.e. $\hat{\mu}$ and $\hat{\sigma}$) are sensible compared with the true value of $\mu$ and $\sigma$, especially when the true variance of prior distribution, $\sigma^2$, is small.

(2) Normal Mean estimation: MLE vs Posterior Mean

The maximum likelihood estimate of $\theta_j$ is: $\hat{\theta}_j(MLE) = X_j$

The posterior mean of of $\theta_j$ is:

$$\hat{\theta}_j(PM) \;=\; E(\theta_j|X_j,\hat{\mu},\hat{\sigma}) \;=\; \frac{\hat{\sigma}^2 X_j + s_j^2\hat{\mu}}{\hat{\sigma}^2 + s_j^2}$$

The sum of square results show that the posterior means provide better accuracy (i.e. lower sum of square of errors) than the maximum likelihood estimates, especially when the true variance of prior distribution, $\sigma^2$, is small.

## (e) 8-schools problem

```
x = c(28, 8,-3, 7,-1, 1,18,12)
s = c(15,10,16,11, 9,11,10,18)
results = ebnm_normal(x=x, s=s)
results
```

```
## $mu.hat
## [1] 7.688397
##
## $sigma.hat
## [1] 0.00678962
##
## $posterior.mean
## [1] 7.688401 7.688397 7.688395 7.688397 7.688392 7.688394 7.688402 7.688398
```

**Commments**:

The Empirical Bayes estimated prior distrution is: $\theta \sim N(7.6884, 0.0068^2)$. And the posterior mean estimation of true effect size of School A is about 7.6884.