1 Regression Part-A

SSE=$\frac{1}{2}(y - X\beta)'W(y - X\beta)$

$\rightarrow \frac{1}{2}(y' - \beta' X')W(y - X\beta)$

$\rightarrow \frac{1}{2}(y'W - \beta' X'W)(y - X\beta)$

$\rightarrow \frac{1}{2}(y'Wy - y'WX\beta - \beta' X'Wy + \beta' X'WX\beta)$

Now, $\nabla(\frac{1}{2}(y'Wy - y'WX\beta - \beta' X'Wy + \beta' X'WX\beta))$

$\rightarrow \frac{1}{2}(-y'WX - X'Wy + 2X'WX\beta) = 0$

$\rightarrow \frac{1}{2}(-2X'Wy + 2X'WX\beta) = 0$

$\rightarrow X'WX\beta = X'Wy$

2 Regression Part-B

$X'WX\beta = X'Wy$.
Let $X'W^{1/2}$=P and $W^{1/2}y$=u.
Therefore the new system of linear equation is: $P'P\beta = P'u$.
It seems that $P'P$ is a square matrix. The above equation can be viewed as Ax=b form. So this A ($P'P$) matrix can be decomposed via LU decomposition method. The LU decomposition factorizes a matrix into a lower triangular matrix L and an upper triangular matrix U. This decomposition summarizes the process of Gaussian elimination in matrix form.

Singular Value Decomposition: Suppose the system of linear equations: Ax=b.
The case where A is an n x n square matrix is of particular interest. In this case, the Singular Value Decomposition of A is given as A=USV^T, where V and U are orthogonal matrices.

QR Decomposition: Any real square matrix A may be decomposed as A=QR. Where Q is an orthogonal matrix (its columns are orthogonal unit vectors meaning Q^TQ = I) and R is an upper triangular matrix (also called right triangular matrix). If A is invertible, then the factorization is unique if we require the diagonal elements of R to be positive. If instead A is a complex square matrix, then there is a decomposition A = QR where Q is a unitary matrix.

3 Regression Part-C

$##Part–C$

```r
##Call library
library(base)
library(Matrix)

##Initialize the values of N and P:
N=NULL
P=NULL

##Simulating design matrix, response variable and Weight matrix:
data<-function(N,P) {
  Y <- rnorm(N)
  Xmat <- matrix(rnorm(N*P),N,P)
  #W <- diag(runif(N),N,N)
  W <- diag(N)
  data.frame(Y=Y,Xmat=Xmat,W=W)
}

##Example:
##Use the equation (X'WX)beta=X'WY:
    rdata <- data(10,8)
     Y <- rdata[,1]
     X <- as.matrix(rdata[,2:9])
     W <- as.matrix(rdata[,10:19])

##Quadratic form:
  A <- t(X) %*% W %*% X
  b <- t(X) %*% W %*% Y



##Application of SVD:
```

```r
C <- svd(A)
D <- diag(C$d)
U <- C$u
V <- C$v

A <- U %*% D %*% t(V) ##We get the original matrix A
InvA <- U %*% solve(D) %*% t(V) ##Calculating the inverse of
#the matrix A using SVD decomposition

##Solution of the equations:
s <- InvA %*% b
```

4 Regression Part-D

```r
library('Matrix')
library('foreach')
library('glmnet')

N <- 1000
P <- 500

X <- matrix(rnorm(N*P), N, P)
mask=matrix(rbinom(N*P,1,0.05),nrow=N)
X=mask*X

beta <- rnorm(P)
Y <- X %*% beta + rnorm(N)
```

```
glmnet.fit1 <- glmnet(X, Y)
```

This package fits lasso and elastic-net model paths for regression, logistic and multinomial regression using coordinate descent. The algorithm is extremely fast, and exploits sparsity in the input x matrix where it exists. A variety of predictions can be made from the fitted models.