To compute Lasso regression, \frac{1}{2}\|\mathbf{y}-\mathbf{X}\mathbf{\beta}\|_{\ell_2}^2+\lambda\|\mathbf{\beta}\|_{\ell_1}define the soft-thresholding functionS(z,\gamma)=\text{sign}(z)\cdot(|z|-\gamma)_+=\begin{cases}z-\gamma&\text{ if }\gamma>|z|\text{ and }z<0\\z+\gamma&\text{ if }\gamma<|z|\text{ and }z<0\\0&\text{ if }\gamma\geq|z|\end{cases}[/latex]The R function would be</p> <p> a19b6d1f6b98b5b2e91bd9a5f66f7f8f000</p> <p>To solve our optimization problem, set[latex display="true"]\mathbf{r}_j=\mathbf{y} - \left(\beta_0\mathbf{1}+\sum_{k\neq j}\beta_k\mathbf{x}_k\right)=\mathbf{y}-\widehat{\mathbf{y}}^{(j)}
so that the optimization problem can be written, equivalently
\min\left\lbrace\frac{1}{2n}\sum_{j=1}^p [\mathbf{r}_j-\beta_j\mathbf{x}_j]^2+\lambda |\beta_j|\right\rbrace
hence\min\left\lbrace\frac{1}{2n}\sum_{j=1}^p \beta_j^2\|\mathbf{x}_j\|-2\beta_j\mathbf{r}_j^T\mathbf{x}_j+\lambda |\beta_j|\right\rbrace
and one gets
\beta_{j,\lambda} = \frac{1}{\|\mathbf{x}_j\|^2}S(\mathbf{r}_j^T\mathbf{x}_j,n\lambda)
or, if we develop
\beta_{j,\lambda} = \frac{1}{\sum_i x_{ij}^2}S\left(\sum_ix_{i,j}[y_i-\widehat{y}_i^{(j)}],n\lambda\right)
Again, if there are weights \mathbf{\omega}=(\omega_i), the coordinate-wise update becomes
\beta_{j,\lambda,{\color{red}{\omega}}} = \frac{1}{\sum_i {\color{red}{\omega_i}}x_{ij}^2}S\left(\sum_i{\color{red}{\omega_i}}x_{i,j}[y_i-\widehat{y}_i^{(j)}],n\lambda\right)
The code to compute this componentwise descent is

```
lasso_coord_desc = function(X,y,beta,lambda,tol=1e-6,maxiter=1000){
  beta = as.matrix(beta)
  X = as.matrix(X)
  omega = rep(1/length(y),length(y))
  obj = numeric(length=(maxiter+1))
  betalist = list(length(maxiter+1))
  betalist[[1]] = beta
  beta0list = numeric(length(maxiter+1))
  beta0 = sum(y-X%*%beta)/(length(y))
  beta0list[1] = beta0
  for (j in 1:maxiter){
    for (k in 1:length(beta)){
      r = y - X[,-k]%*%beta[-k] - beta0*rep(1,length(y))
      beta[k] = (1/sum(omega*X[,k]^2))*
        soft_thresholding(t(omega*r)%*%X[,k],length(y)*lambda)
    }
    beta0 = sum(y-X%*%beta)/(length(y))
    beta0list[j+1] = beta0
    betalist[[j+1]] = beta
    obj[j] = (1/2)*(1/length(y))*norm(omega*(y - X%*%beta -
           beta0*rep(1,length(y))),'F')^2 + lambda*sum(abs(beta))
    if (norm(rbind(beta0list[j],betalist[[j]]) -
              rbind(beta0,beta),'F') &lt; tol) { break }
  }
  return(list(obj=obj[1:j],beta=beta,intercept=beta0)) }
```

For instance, consider the following (simple) dataset, with three covariates

```
chicago = read.table("http://freakonometrics.free.fr/chicago.txt",header=TRUE,sep=";")
```

that we can “normalize”

```
X = model.matrix(lm(Fire~.,data=chicago))[,2:4]
```

```r
for(j in 1:3) X[,j] = (X[,j]-mean(X[,j]))/sd(X[,j])
y = chicago$Fire
y = (y-mean(y))/sd(y)
```

To initialize the algorithm, use the OLS estimate

```r
beta_init = lm(Fire~0+.,data=chicago)$coef
```

For instance

```r
lasso_coord_desc(X,y,beta_init,lambda=.001)
$obj
[1] 0.001014426 0.001008009 0.001009558 0.001011094 0.001011119 0.001011119

$beta
           [,1]
X_1  0.0000000
X_2  0.3836087
X_3 -0.5026137

$intercept
[1] 2.060999e-16
```

and we can get the standard Lasso plot by looping,