```r
rm(list = ls())
 setwd("G:\\math\\504")
options(scipen=999)
 #require("ggplot2")
bones<-read.table("BoneMassDataF.txt",header=T)
bones<-bones[which(bones[,3] == "female"),]

Fapprox<-function(m,n=10000,x){

  MATT<-matrix(0:(m-1),m,1)

  Gx<-function(v,x){ if(v != 0) {
    return( cos( (2*pi*v*(x-9.4 ) ) /(25.55-9.4 )) )
    } else  {return(rep(1,length(x) ))} }

  a=min(x);b=max(x)
  h=(b-a)/n;i=0:(n-1)
  W<-apply(MATT,1,function(w) Gx(w,a+(i+1)*h) )

  Mat<-matrix(NA,m,m)

  for( i in 1:m){
    Mat[,i] <-   colSums( (  W[  ,i ]*W[  ,1:m]  ) ) * h }

return(Mat)
}

Fapprox(6,n=10000, bones$age)
```

```
##                            [,1]                        [,2]
## [1,] 16.149999999999985789145 -0.000000000000000002216161
## [2,] -0.000000000000000002216161  8.074999999999992894573
## [3,] -0.000000000000000002057479 -0.000000000000000002262779
## [4,] -0.000000000000000002003689 -0.000000000000000001754460
## [5,] -0.000000000000000001406168 -0.000000000000000002019826
## [6,] -0.000000000000000002067341 -0.000000000000000002088857
##                            [,3]                        [,4]
## [1,] -0.000000000000000002057479 -0.000000000000000002003689
## [2,] -0.000000000000000002262779 -0.000000000000000001754460
## [3,]  8.074999999999992894573 -0.000000000000000001924796
## [4,] -0.000000000000000001924796  8.074999999999992894573
## [5,] -0.000000000000000002312535 -0.000000000000000001626260
## [6,] -0.000000000000000001361791 -0.000000000000000001532127
##                             [,5]                         [,6]
## [1,] -0.000000000000000014061682 -0.000000000000000020673407
## [2,] -0.000000000000000020198260 -0.000000000000000020888569
## [3,] -0.000000000000000023125349 -0.000000000000000013617912
## [4,] -0.000000000000000016262602 -0.000000000000000015321272
## [5,]  8.074999999999999928945726 -0.000000000000000002815026
## [6,] -0.000000000000000002815026  8.074999999999999928945726
```

```r
Gapprox<-function(n,j,k,x){

    coeff<-(1/sqrt( diag(Fapprox(6,n=10000, bones$age)) ))
    Fx<-function(l,x){ if(l != 0) {
```

```
        return( coeff[l+1]* cos( (2*pi*l*(x-9.4 ) ) /(25.55-9.4 )) )
        } else  {return( coeff[l+1]* rep(1,length(x) ))} }

    a=min(x);b=max(x)

    h=(b-a)/n;i=0:(n-1)
    return(     sum( Fx(j,a+(i+1)*h)*Fx(k,a+(i+1)*h)*h )    )
}

matt<-matrix(NA,6,6)
for( J in 0:5){for( K in 0:5){ matt[J+1,K+1]<-Gapprox(10000,J,K,bones[,2]) }}
matt;diag(matt )
```

```
##                             [,1]                        [,2]
## [1,]  1.000000000000000044408921 -0.000000000000000001931235
## [2,] -0.000000000000000001931235  0.999999999999999888977770
## [3,] -0.000000000000000001764539 -0.000000000000000002870425
## [4,] -0.000000000000000001656119 -0.000000000000000002258529
## [5,] -0.000000000000000001279359 -0.000000000000000002429968
## [6,] -0.000000000000000001767250 -0.000000000000000002429290
##                             [,3]                        [,4]
## [1,] -0.000000000000000001764539 -0.000000000000000001656119
## [2,] -0.000000000000000002870425 -0.000000000000000002258529
## [3,]  0.999999999999999888977770 -0.000000000000000002406251
## [4,] -0.000000000000000002406251  0.999999999999999888977770
## [5,] -0.000000000000000002953096 -0.000000000000000002110128
## [6,] -0.000000000000000001658829 -0.000000000000000001872282
##                             [,5]                        [,6]
## [1,] -0.000000000000000012793586 -0.000000000000000017672495
## [2,] -0.000000000000000024299681 -0.000000000000000024292905
## [3,] -0.000000000000000029530957 -0.000000000000000016588293
## [4,] -0.000000000000000021101285 -0.000000000000000018722816
## [5,]  0.999999999999999888977698 -0.000000000000000005817422
## [6,] -0.000000000000000005817422  0.999999999999999888977698
```

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

In this step, I am writing a function for any general b_j. If w is 1,...,5, I obtain the normalizing coefficients, apply them to the $b_j = 1, ..., 5$, and return the specified b_j. If 0 is given I return 1's.

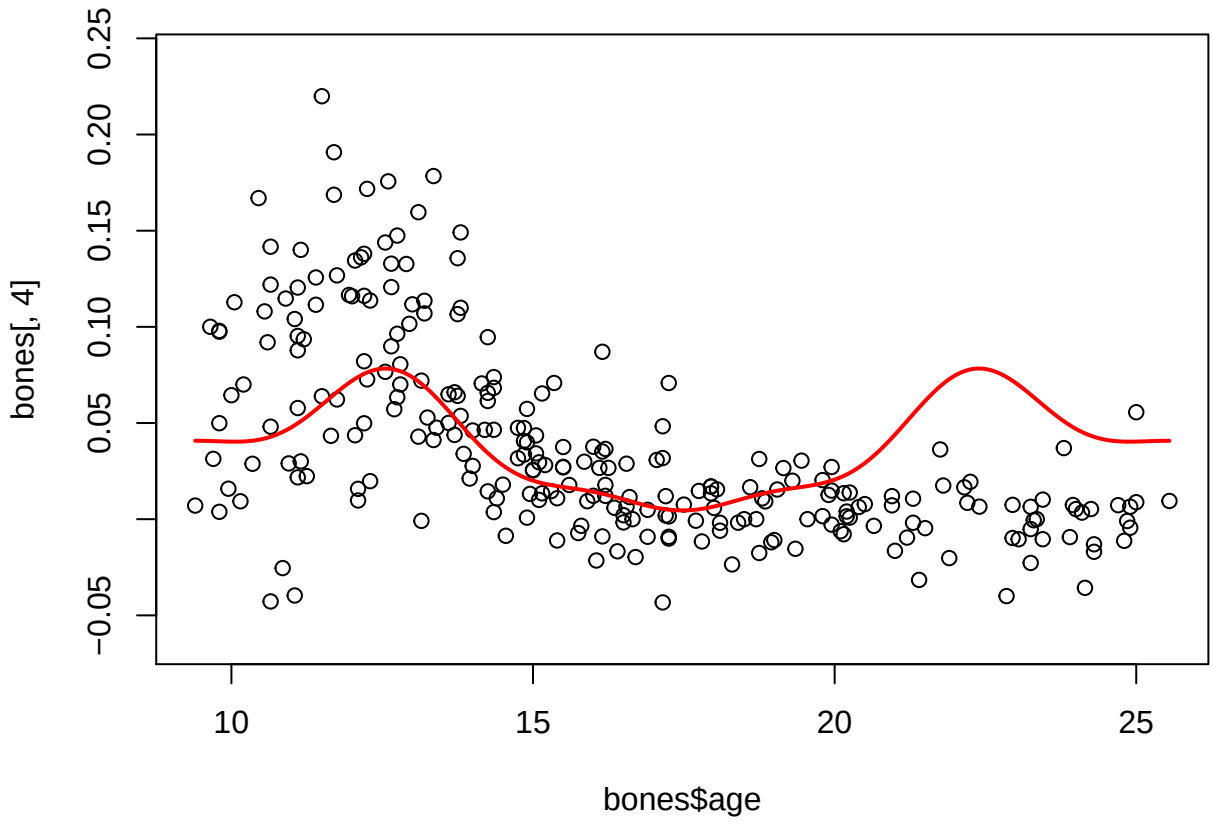
```
Fx<-function(w,x){
    #need this matrix for the normalizing Coefficients
    ncoef<-(1/sqrt( diag(Fapprox(6,n=10000, bones$age)) ))
    if(w == 0 ) {return(   rep(1,length(x) ))  } else {
    return( ncoef[w+1]* cos( (2*pi*w*(x-9.4 ) ) /(25.55-9.4 )) )
        } }
```

In this step I form the model matrix. By using our function Fx above. Then I solve for α.

```
model_matrix <- function(x,m) {
    nx <- length(x)
```

```
    A<-matrix(NA,nx,m+1)
    for( i in 0:m){
        A[,(i+1)] <- Fx(i,x)  }
    colnames(A )<-NULL
    return(A)   }


B<- model_matrix(bones$age,5)


Alpha <- solve(t(B) %*% B , t(B) %*% as.matrix(bones[,4]))
```

```
x_grid <- seq(min(bones$age), max(bones$age), .01)
B_grid <- model_matrix(x_grid,5)
y_grid <- B_grid %*% Alpha

par(mar=c(4.1,4.1,2,1))
plot(bones$age, bones[,4], ylim=c(min(bones[,4])-.02,max(bones[,4]) + .02) )
lines(x_grid, y_grid, col="red", lwd=2)
```



We have some data, $x^{(i)} \in \mathbb{R}$ and $y_i \in \mathbb{R}$, and we wish to approximate y by a function $f(x)$, when $f(x) \in \mathcal{F}$ and $f(x) : \mathbb{R}^n \to \mathbb{R}$.

Let $\mathcal{F}$ be:

$$\mathcal{F} = \{f(x) : f(x) : \mathbb{R}^n \to \mathbb{R}, f(x) = 5^{th} \text{ harmonic}\}.$$

We use least squares to find the $f(x) \in \mathcal{F}$

$$\min_{f(x)\in\mathcal{F}} \sum_{i=1}^{N} \left| y_i - f\left(x^{(i)}\right) \right| \tag{1}$$

It is key to note that $\mathcal{F}$ is a linear function / vector space. It is true that for any $g(x) \in \mathcal{F}$ and $h(x) \in \mathcal{F}$, $c_1 g(x) + c_2 h(x) \in \mathcal{F}$, where $c_1, c_2 \in \mathbb{R}$

Our function $f(x)$ can be written as:

$$f(x) = \sum_{j=0}^{n} \alpha_j b_j(x) = \alpha_0(1) + \alpha_1 \cos\left(\frac{2\pi(1)\left(x - x_{max}\right)}{\left(x_{max} - x_{min}\right)}\right) + ... + \alpha_5 \cos\left(\frac{2\pi(5)\left(x - x_{max}\right)}{\left(x_{max} - x_{min}\right)}\right)$$

so we can rewrite (1) as:

$$\min_{f(x)\in\mathcal{F}} \sum_{i=1}^{N} \left| y_i - f\left(x^{(i)}\right) \right| = \min_{\alpha\in\mathbb{R}^6} \sum_{i=1}^{N} \left| \alpha_0 + \alpha_1 \cos\left(\frac{2\pi\left(x_i - x_{max}\right)}{\left(x_{max} - x_{min}\right)}\right) + ... + \alpha_5 \cos\left(\frac{10\pi\left(x_i - x_{max}\right)}{\left(x_{max} - x_{min}\right)}\right) \right|^2 . \tag{2}$$

The sum in the RHS of (2) can be put into matrix form:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} b_0\left(x^{(1)}\right) & b_1\left(x^{(1)}\right) & ... & b_5\left(x^{(1)}\right) \\ b_0\left(x^{(2)}\right) & b_1\left(x^{(2)}\right) & ... & b_5\left(x^{(2)}\right) \\ \vdots & \vdots & \ddots & \vdots \\ b_0\left(x^{(N)}\right) & b_1\left(x^{(N)}\right) & ... & b_5\left(x^{(N)}\right) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{pmatrix} \tag{3}$$

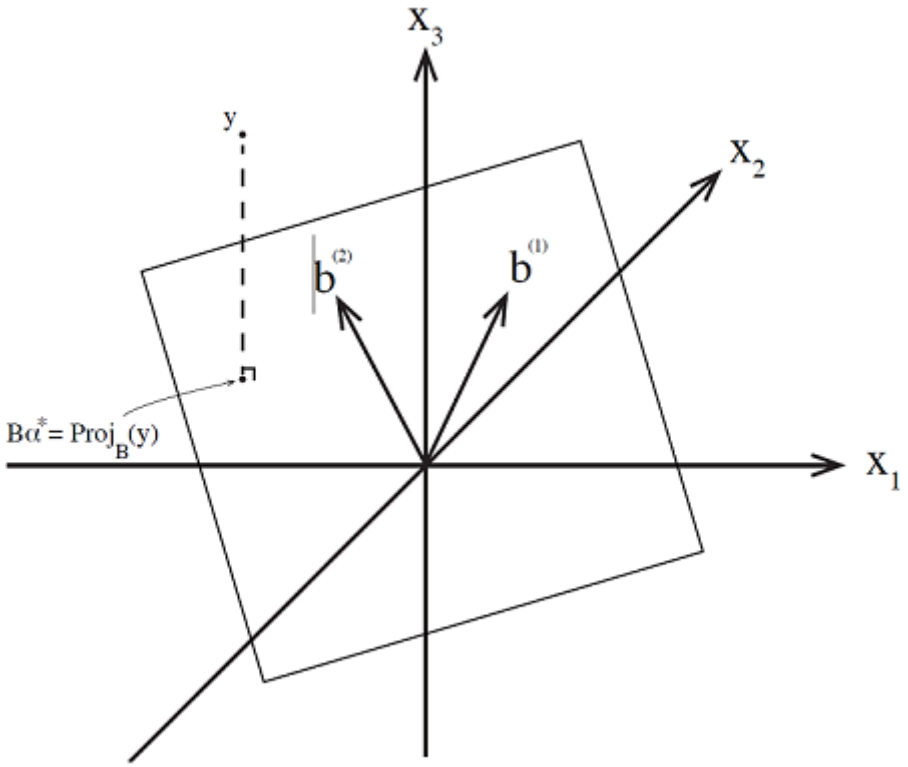
and (2) is equivalent to

$$\min_{\alpha\in\mathbb{R}^6} ||\boldsymbol{y} - B\boldsymbol{\alpha}||^2. \tag{4}$$

So given some data we are trying to fit through that data the best 5^{th} harmonic. We are not doing regression in the sense of fitting a line because we are fitting a 5^{th} harmonic. But we end up doing the same exact thing that we do for linear regression (minimize sum of squares, replace with model matrix).

The important part of linear regression is not that the functions are linear, it is that the space of the functions we are considering are a linear function space. We are still linear in the parameters. Thus in this context, determining $f(x)$ corresponds to a linear regression.

Now for projection consider the $\boldsymbol{y}$, B and $\boldsymbol{\alpha}$ from (3) above. We wish to choose $\boldsymbol{\alpha}$ so that $B\boldsymbol{\alpha}$ is as close as possible to $\boldsymbol{y}$. Or put another way, find the point in the $\text{Span}(B) = \text{Span}\left(b^{(0)} \quad b^{(1)} \quad \dots \quad b^{(5)}\right)$ closest to $\boldsymbol{y}$. If $n = 2$, we would have a visual like this:



For $n = 6$, b_0 through b_5 are linearly combining to form a plane Again, we want to find the point closest to $\boldsymbol{y}$. It will be the point that is orthogonal and this point will be the $\text{Proj}_B(y)$.

The projection of $x \in \mathbb{R}$ onto Ω is the closest point in Ω to x.

$$P_\Omega(x) = \min_{z \in \Omega} ||z - x||$$

Now suppose $\Omega = \text{Span}\left(b^{(0)} \quad b^{(1)} \quad \dots \quad b^{(5)}\right)$, and $P_\Omega(x) = \min_{z \in \Omega} ||z - y||$, where

$$z = \alpha_0 b^{(0)} + \alpha_1 b^{(1)} + ... + \alpha_5 b^{(5)} = B\boldsymbol{\alpha} = \left(b^{(0)} \quad b^{(1)} \quad \dots \quad b^{(5)}\right)\left(\alpha_0 \quad \alpha_1 \quad \dots \quad \alpha_5\right)^T$$

and

$$\min_{z\in\Omega} ||z-y|| = \min_{z\in\Omega} ||\alpha_0 b^{(0)} + \alpha_1 b^{(1)} + ... + \alpha_5 b^{(5)} - y|| \tag{5}$$

So we have $\min_{\alpha\in\mathbb{R}} ||B\alpha - y||^2$. To tie projection and linear regression together, we can view linear regression as a projection onto the span of the columns of the model matrix.

Now tying both of them to quadratic optimization. When we have a linear subspace and project onto it, it leads to a minimization of a quadratic $\min_{\alpha\in\mathbb{R}^6} ||B\boldsymbol{\alpha} - \boldsymbol{y}||^2$, (5) above. When $\mathcal{F}$ is a linear function space then the regression is a minimization of a quadratic $\min_{\alpha\in\mathbb{R}^6} ||\boldsymbol{y} - B\boldsymbol{\alpha}||^2$, (4) above.

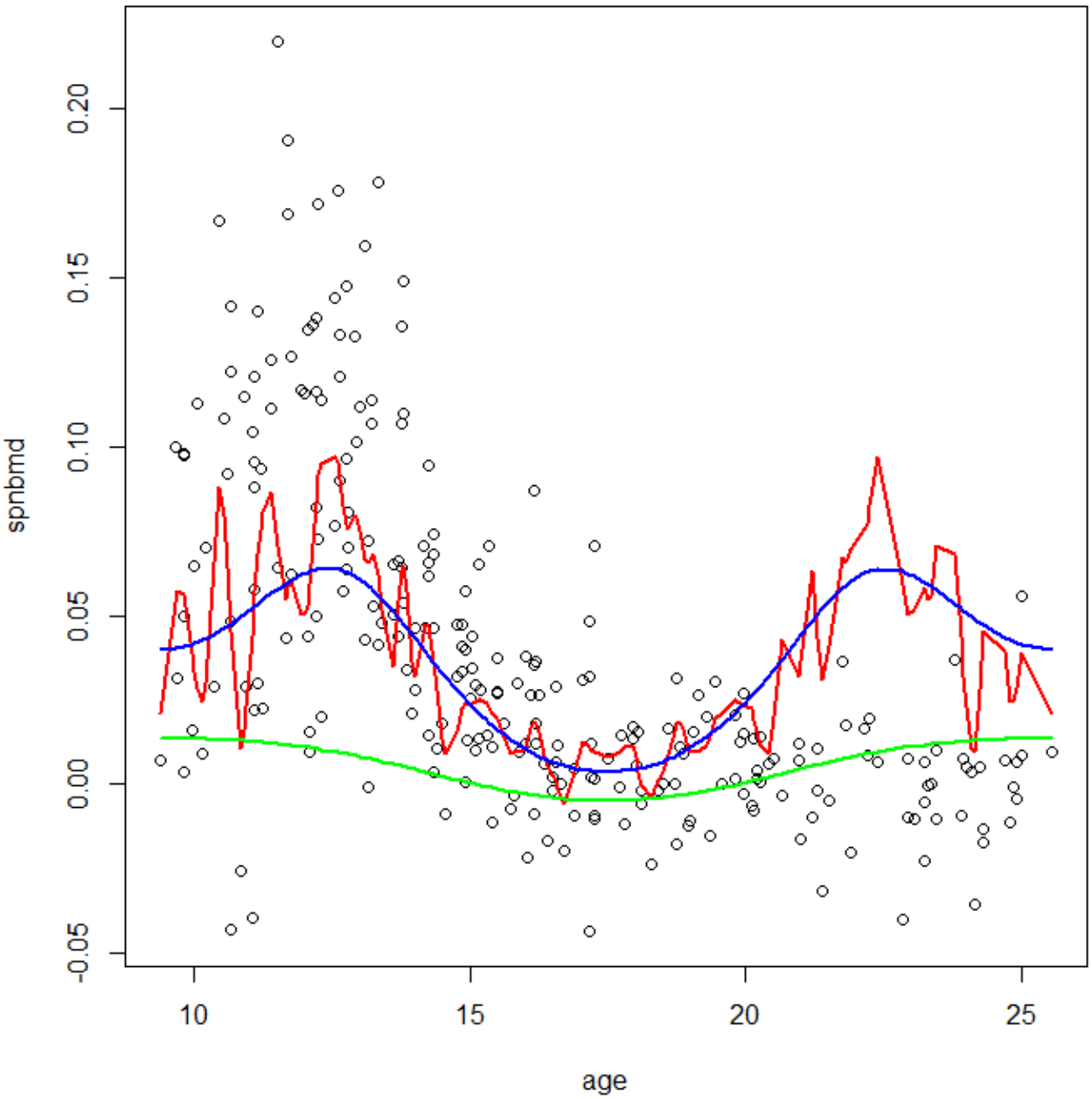
Both (4) and (5) are solved with the normal equations.

$$\begin{aligned}
L(\alpha) &= \sum_{i=1}^{N} \left(y_i - \alpha_0 + \alpha_1 x_1^{(i)} + \alpha_2 x_2^{(i)} + ...\alpha_n x_n^{(i)} \right)^2 \\
&= \sum_{i=1}^{N} r_i^2 \\
&= \mathbf{r}\cdot\mathbf{r} \\
&= (y - B\alpha)\cdot(y - B\alpha) \\
&= (y - B\alpha)^T (y - B\alpha) \\
&= \left(y^T - (B\alpha)^T\right)(y - B\alpha) \\
&= y^T y - (B\alpha)^T y - y^T B\alpha + (B\alpha)^T B\alpha \\
&= y^T y - 2(B^T y)^T \alpha + \alpha^T B^T B\alpha
\end{aligned}$$

$$\begin{aligned}
\nabla L(\alpha) &= -\frac{1}{2}\left(B^T B\right)^{-1}\left(-2B^T y\right) \\
&= \left(B^T B\right)^{-1} B^T y
\end{aligned}$$

We have shown that determining $f(x)$ corresponds to a linear regression, projection of a vector onto a linear space, and a quadratic optimization.

```
Fx<-function(w,x){
    #need this matrix for the normalizing Coefficients
    ncoef<-(1/sqrt( diag(Fapprox(1001,n=10000, bones$age)) ))
    if(w == 0 ) {return(   rep(1,length(x) ))  } else {
    return( ncoef[w+1]* cos( (2*pi*w*(x-9.4 ) ) /(25.55-9.4 )) )
        } }
dmat<-function(m,n=10000,x){

MATT<-matrix(0:(m-1),m,1)

dFx<-function(v,x){ if(v != 0) {
    return( -((4*pi^2*v^2)/(16.15^2))* cos( (2*pi*v*(x-9.4 ))/16.15)
    )   } else  {return(rep(1,length(x) ))} }

a=min(x);b=max(x)
h=(b-a)/n;i=0:(n-1)
W<-apply(MATT,1,function(w) dFx(w,a+(i+1)*h) )
}

Bpp<-dmat(1000+1,n=10000, bones$age)
z<-seq(min(bones$age),max(bones$age),   length.out = 10000 )
h<-z[2]-z[1]


omega<-h*(t(Bpp) %*% Bpp )

B<- model_matrix(bones$age,1000)
n=1001
y<-as.matrix( bones[,4] )
A<-matrix(NA,(n ),3)
A[,1]<-solve( t(B)%*%B + (.0001 * omega) ) %*% t(B)%*% y
A[,2]<-solve( t(B)%*%B + (1 * omega) ) %*% t(B)%*% y
A[,3]<-solve( t(B)%*%B + (100 * omega) ) %*% t(B)%*% y


par(mar=c(4.1,4.1,1,1))
datt<-data.frame(bones$age, B%*% A[,1], B%*% A[,2], B%*% A[,3] )
datt<-datt[with(datt, order(bones$age)), ]

plot(bones[,2], bones[,4], xlab="age", ylab="spnbmd")
lines( datt[,1],datt[,2],col="red", lwd=2)
lines( datt[,1],datt[,3],col="blue", lwd=2)
lines( datt[,1],datt[,4],col="green", lwd=2)
```



As ρ increases the regression becomes more linear in order to reduce the penalty term, the result is that the fitting term increases, meaning that the fit becomes poorer.

The neural net is a nonconvex optimization, we may go to a local minimum. We want the hessian of the negative log likelihood to be positive definite, or alternatively we want all the eigenvalues to be positive. However, for neural nets the hessian of negative log likelihood is not going to be positive definite because it is not a convex function.

However we can modify the hessian such that it is positive definite and similar to the hessian. We can rewrite the hessian as a decomposition, and add a constant so the minimum eigenvalue becomes positive

$$H = Q \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} Q^T$$

$$H + \lambda I = Q \begin{pmatrix} \lambda_1 + \lambda & & & \\ & \lambda_2 + \lambda & & \\ & & \ddots & \\ & & & \lambda_n + \lambda \end{pmatrix} Q^T$$

$$= Q \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} Q^T + Q \begin{pmatrix} \lambda & & & \\ & \lambda & & \\ & & \ddots & \\ & & & \lambda \end{pmatrix} Q^T.$$

We should raise λ if we are doing poorly, because raising the lambdas enough will get all of our eigenvalues positive. When the eigenvalues are all positive we are in a region of the function that is locally convex and we will go down.

First, let's write down $Hf(\eta)d$

$$Hf(\eta)d = \begin{pmatrix} \frac{\partial^2 f(\eta)}{\partial \eta_1^2} & \frac{\partial^2 f(\eta)}{\partial \eta_2 \partial \eta_1} & \dots & \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_1} \\ \frac{\partial^2 f(\eta)}{\partial \eta_2 \partial \eta_1} & \frac{\partial^2 f(\eta)}{\partial \eta_2^2} & \dots & \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_1} & \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_2} & \dots & \frac{\partial^2 f(\eta)}{\partial \eta_n^2} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} \tag{6}$$

$$= \begin{pmatrix} \frac{\partial^2 f(\eta)}{\partial \eta_1^2} d_1 + \frac{\partial^2 f(\eta)}{\partial \eta_2 \partial \eta_1} d_2 + \dots + \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_1} d_n \\ \frac{\partial^2 f(\eta)}{\partial \eta_2 \partial \eta_1} d_1 + \frac{\partial^2 f(\eta)}{\partial \eta_2^2} d_2 + \dots + \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_2} d_n \\ \vdots \\ \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_1} d_1 + \frac{\partial^2 f(\eta)}{\partial \eta_n \partial \eta_2} d_2 + \dots + \frac{\partial^2 f(\eta)}{\partial \eta_n^2} d_n \end{pmatrix} \tag{7}$$

$$= \begin{pmatrix} \nabla \left(\frac{\partial}{\partial \eta_1} f(\eta) \right) \cdot \boldsymbol{d} \\ \nabla \left(\frac{\partial}{\partial \eta_2} f(\eta) \right) \cdot \boldsymbol{d} \\ \vdots \\ \nabla \left(\frac{\partial}{\partial \eta_i} f(\eta) \right) \cdot \boldsymbol{d} \\ \vdots \\ \nabla \left(\frac{\partial}{\partial \eta_n} f(\eta) \right) \cdot \boldsymbol{d} \end{pmatrix} \tag{8}$$

Now we can show that

$$Hf(\eta)d = \lim_{\epsilon \to 0} \frac{\nabla f(\eta + \epsilon d) - \nabla f(\eta)}{\epsilon}$$

is equal to $Hf(\eta)d$. Consider the ith coordinate of the vector $Hf(\eta)d$.

$$\Big[Hf(\eta)d \Big]_i = \lim_{\epsilon \to 0} \frac{\frac{\partial}{\partial \eta_i} f(\eta + \epsilon \boldsymbol{d}) - \frac{\partial}{\partial \eta_i} f(\eta)}{\epsilon} \tag{9}$$

First-order Taylor series expansion about the base point η:

$$\frac{\partial}{\partial \eta_i} f(\eta + \epsilon \boldsymbol{d}) \approx \frac{\partial}{\partial \eta_i} f(\eta) + \nabla \left(\frac{\partial}{\partial \eta_i} f(\eta) \right) \cdot \epsilon \boldsymbol{d}.$$

Plug back the expansion back into (9)

$$\begin{aligned}
\Big[H f(\eta) d \Big]_i &= \lim_{\epsilon \to 0} \frac{\frac{\partial}{\partial \eta_i} f(\eta + \epsilon \boldsymbol{d}) - \frac{\partial}{\partial \eta_i} f(\eta)}{\epsilon} \\
&\approx \lim_{\epsilon \to 0} \frac{\cancel{\frac{\partial}{\partial \eta_i} f(\eta)} + \nabla \left(\frac{\partial}{\partial \eta_i} f(\eta) \right) \cdot \cancel{\epsilon} \boldsymbol{d} - \cancel{\frac{\partial}{\partial \eta_i} f(\eta)}}{\cancel{\epsilon}} \\
&= \nabla \left(\frac{\partial}{\partial \eta_i} f(\eta) \right) \cdot \boldsymbol{d}
\end{aligned}$$

We see that for the ith coordinate is equal to the ith row in (8) above. Other coordinates will follow by the same argument.