Splines, Statistical Significance and Bone Data

Hannah Pieper

4/12/21

Introduction

This document performs regression using splines on the bone data found at https://web.stanford.edu/~hastie/ElemStatLearn/data.html. This data consists of observations containing four features; ID, age, gender and bone density. In particular, we perform regression with smoothing splines on age and bone density and quantify the statistical significance of our results.

This document uses the following packages: splines, dplyr, MASS, knitr, graphics.

First we load the data.

```
bone.data<-read.table("bone_data.txt", header = TRUE, dec=".")
```

This data frame contains each of the observations in a row. We will consider the ozone levels to be the response variable and temperature, wind, and radiation to be predictors. Visually:

```
head(bone.data)
```

```
##   idnum   age gender      spnbmd
## 1     1 11.70   male 0.018080670
## 2     1 12.70   male 0.060109290
## 3     1 13.75   male 0.005857545
## 4     2 13.25   male 0.010263930
## 5     2 14.30   male 0.210526300
## 6     2 15.30   male 0.040843210
```

Smoothing Spline Regression

First recall that natural splines are required to be linear at the end intervals $[a, \xi_1]$ and $[\xi_K, b]$ and a basis for the set of all natural cubic splines on K knots in one dimension is given by

$$\begin{gathered} N_1(x) = 1, \qquad N_2(x) = x \\ N_{k+2}(x) = d_k(x) - d_{K-1}(x), \qquad k = 1, \dots, K-2 \\ \text{where } d_k(x) = \frac{(x-\xi_k)_+^3 - (x-\xi_K)_+^3}{\xi_K - \xi_k}. \end{gathered}$$

While the functions $d_k(x)$ are not linear, $d_k(x) - d_{K-1}(x)$ are all linear. We then perform linear regression to determine the parameter $\hat{\theta} = (\hat{\theta}_1, \dots, \hat{\theta}_K)$ to construct the model

$$\hat{f}(x) = \sum_{i=1}^{K} \theta_i N_i(x).$$

Smoothing splines use the basis of natural splines and incorporate the smoothing parameter λ and minimize the penalized residual sum squares

$$RSS(f,\lambda) = \sum_{i=1}^{N} \left(y_i - f(x_i)\right)^2 + \lambda \int \left(f''(t)\right)^2 \, dt.$$

We have previously argued that the minimizer is a natural spline, so we know that f has the form

$$f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$$

where $\{N_j\}$ forms a basis for the space of natural splines of order M. If we define the matrix $\mathbf{N}$ to have ijth entry given by $N_j(x_i)$ and the matrix $\mathbf{\Omega}_N$ to have jkth entry $\int N_j'' N_k''$, then we can formulate the residual sum squares as

$$RSS(\theta,\lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\mathbf{\Omega}_N\theta$$

where the minimizing solution is given by

$$\hat{\theta} = \left(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N\right)^{-1}\mathbf{N}^T\mathbf{y}.$$

Smooth splines can be made in R using smooth.spline(). The default degree of the splines is cubic and the smoothing parameter λ is specified from the degrees of freedom. If we set "cv = FALSE", then λ is determined by generalized cross-validation. If we set "cv =TRUE", then λ is determined by leave-one-out cross validation, but since we have repeated age values, we use generalized cross-validation, which uses the singular value decomposition of the data matrix.

First we need to determine how many knots we want to use. We want a knot ξ_k at every value of the age data, we need to determine how many unique ages are present.

```r
num.knots<-length(unique(bone.data[['age']]))
sep="\n"

string<-paste("The number of knots and basis functions we will have is: ", num.knots, ".")
cat(string)
```

```
## The number of knots and basis functions we will have is:  239 .
```

Functions and Documentation

H.matrix()

This function compute an $N \times K$ matrix where N is the number of data points and K is the number of basis vectors. The ijth element of the matrix is given by $N_j(x_i)$ where $N_j(x)$ are the basis functions described above.

Inputs:
data - a data frame consisting of N observations of feature and response pairs

Outputs: H - an $N \times K$ matrix whose entries are the basis functions evaluated at the feature training data

```r
H.matrix<-function(data){
  K<-length(unique(data[['age']]))
  N<-length(data[['age']])
  x<-data[['age']]
  xi<-sort(as.vector(unique(data[['age']])))
  H<-matrix(0,N,K)

  for(i in 1:N){
    for(j in 1:2){
      H[i,j]=x[i]**(j-1)
    }
  }
  for (i in 1:N){
    d_K_1<-(cutoff((x[i]-xi[K-1])**3)-cutoff((x[i]-xi[K])**3))/(xi[K]-xi[K-1])
    for (j in 1:(K-2)){
      d_j<-(cutoff((x[i]-xi[j])**3)-cutoff((x[i]-xi[K])**3))/(xi[K]-xi[j])
      H[i,j+2]=d_j-d_K_1
    }
  }
  return(H)
}
```

cutoff()

This is a helper function that zeros out negative values.

Inputs:
x - a scalar

Outputs:
x - if $x \geq 0$;
0 - if $x < 0$.

```
cutoff<-function(x){
  if(x>=0){
    return(x)}
  else{
    return(0)}
}
```

sigma.hat()

Suppose that our data follows the form $y_i = f(x_i) + \epsilon_i$ with ϵ_i a random variable. Then, this function computes the estimated noise variance of the responses y_i given by

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{\mu}(x_i)\right)^2,$$

where $\hat{\mu}$ is our estimated model.

Inputs:
data - training data $\{(x_i, y_i)\}$

Outputs:
σ^2 - estimated noise variance

```
sigma.hat<-function(data){
  ages<-data[['age']]
  spline<-smooth.spline(x=ages, y=data[,'spnbmd'], cv=FALSE, all.knots=TRUE)
  preds<-predict(spline,x=ages)

  N<-length(ages)
  errorvec<-matrix(1,N)
  for (i in 1:N){
    error<-(preds$y[i]-data[['spnbmd']][i])**2
    errorvec[i]<-error
  }
  return(1/N*(sum(errorvec)))
}
```

little.h()

This computes the vector

$$h(x) = (N_1(x), \ldots, N_K(x))^T$$

where $N_i(x)$ is a basis function.

Inputs:
y - the value at which to evaluate the basis functions
data - the data set from which the knots $\xi_1, \ldots, \xi_K$ are constructed

Outputs:
H - a K dimensional column vector.

```
little.h<-function(y,data){
  K<-length(unique(data[['age']]))
  x<-sort(as.vector(unique(data[['age']])))
  H<-matrix(1, K)

  for(j in 1:2){
      H[j]=y**(j-1)
    }
  d_K_1<-(cutoff((y-x[K-1])**3)-cutoff((y-x[K])**3))/(x[K]-x[K-1])
  for (j in 1:(K-2)){
    d_j<-(cutoff((y-x[j])**3)-cutoff((y-x[K])**3))/(x[K]-x[j])
    H[j+2]=d_j-d_K_1
    }
  return(H)
}
```

standard.error()

Letting $h(x)^T = (N_1(x), \ldots, N_K(x))$, we can define the standard error of a predictor $\hat{\mu}(x) = h(x)^T\hat{\beta}$ to be

$$\hat{se}\big[\hat{\mu}(x)\big] = \left[h(x)^T \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\right]^{1/2} \hat{\sigma}.$$

Inputs:
x - the value at which to evaluate the standard error
H.matrix - the matrix of basis functions evaluated at the predictor values in the training data
sigma - the estimated noise variance
data - the training set

Outputs:
val - the standard error of the model at x; is a scalar

```
standard.error<-function(x, H.matrix, sigma, data){
  h<-little.h(x, data)
  val<-sigma*(t(h)%*%ginv(t(H.matrix)%*%H.matrix)%*%h)^(1/2)
  return(val)
}
```

Plots and Results

First we compute the matrix H of the basis functions evaluated at the unique training data for age.

```
bigH<-H.matrix(bone.data)
sigma<-sigma.hat(bone.data)
```

If $\hat{\mu}(x)$ is our estimated model and $\{(x_i, y_i)\}$ are the training data, then we can estimate the noise variance as

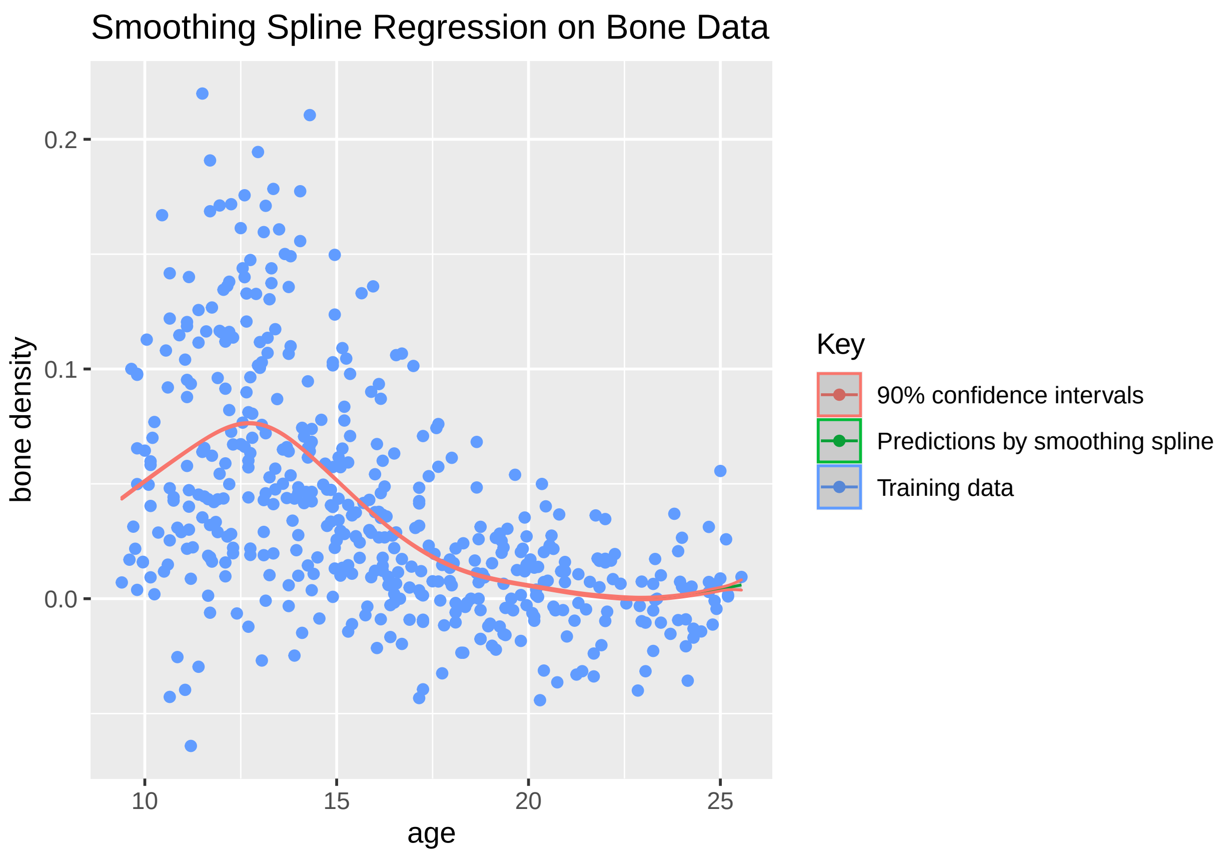
$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{\mu}(x_i)\right)^2.$$

Using this estimate, we can then form the standard error of a predictor as

$$\hat{se}\left[\hat{\mu}(x)\right] = \left[h(x)^T(\mathbf{H}^T\mathbf{H})^{-1}h(x)\right]^{1/2}\hat{\sigma}.$$

The matrix $\mathbf{H}$ has ijth entry $h_j(x_i)$. Since we are using smoothing splines, there is a knot at every unique age value. Thus, if there are N training data points and K unique age values, $\mathbf{H}$ will be an $N \times K$ matrix.

```
ages<-bone.data[['age']]
age.grid<-seq(from=min(ages), to=max(ages), length.out=200)

pts<-matrix(1,200)
for(i in 1:200){
  pts[i]<-standard.error(age.grid[i], bigH, sigma, bone.data)
}

spline<-smooth.spline(x=bone.data[,'age'], y=bone.data[,'spnbmd'],cv=FALSE,  all.knots=TRUE)
preds<-predict(spline, x=age.grid)

lower<-preds$y-1.645*pts
upper<-preds$y+1.645*pts


label1<-'90% confidence intervals'
label2<-'Predictions by smoothing spline'
Key<-'Training data'

p<-ggplot() +
  geom_point(data=bone.data, aes(x=age, y=spnbmd, color=Key))+
  geom_line(aes(x=age.grid, y=preds$y, color=label2))+
  geom_ribbon(aes(x=age.grid, ymin=lower, ymax=upper, color=label1), alpha=.2)+
  labs(title="Smoothing Spline Regression on Bone Data", x="age", y="bone density")

print(p)
```



Posterior Estimates

If our model is of the form $\mu(x) = h(x)\beta^T$, then we can provide a prior distribution on the coefficients of β, which will then implicitly define a prior distribution on $\mu(x)$. We will choose a Gaussian prior with zero mean; $\beta \sim N(0, \tau\Sigma)$ where the prior correlation matrix and τ must be chosen. We will take Σ to be the identity. The posterior distribution for β will also be Gaussian, with mean and covariance

$$E(\beta|\mathbf{Z}) = \left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1}\right)^{-1} \mathbf{H}^T\mathbf{y}$$
$$cov(\beta|\mathbf{Z}) = \left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1}\right)^{-1} \sigma^2,$$

where $\mathbf{Z} = \{z_1, z_2, \ldots, z_N\}$ with $z_i = (x_i, y_i)$ is the training data.

Then the posterior values for the model μ are given as

$$E(\mu|\mathbf{Z}) = h(x)^T \left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1}\right)^{-1} \mathbf{H}^T\mathbf{y}$$
$$cov(\mu(x), \mu(x')|\mathbf{Z}) = h(x)^T \left(\mathbf{H}^T\mathbf{H} + \frac{\sigma^2}{\tau}\Sigma^{-1}\right)^{-1} h(x')\sigma^2$$

Functions and Documentation

posterior.mean()

This function computes the posterior mean for the model $\mu(x)$.

Inputs:
x - the value at which to evaluate μ
bigH - the matrix **H** consisting of the basis functions evaluated at points in the training set
sigma - the estimated noise variance
tau - a scalar constant to be chosen

Output: val - the posterior mean of $\mu(x)$

```
posterior.mean<-function(x,bigH,sigma,tau){
  y<-bone.data[['spnbmd']]
  h<-little.h(x, bone.data)
  K<-length(unique(bone.data[['age']]))
  val<-t(h)%*%ginv(t(bigH)%*%bigH + sigma^2/tau*diag(K))%*%t(bigH)%*%y
  return(val)
}
```

posterior.cov()

This function computes the posterior covariance for the model $\mu(x)$.

Inputs:
x, y - the values at which to evaluate μ
bigH - the matrix **H** consisting of the basis functions evaluated at points in the training set
sigma - the estimated noise variance
tau - a scalar constant to be chosen

Output: val - the posterior mean of $\mu(x)$

```
posterior.cov<-function(x,y,bigH,sigma,tau){
  hx<-little.h(x, bone.data)
  hy<-little.h(y, bone.data)
  K<-length(unique(bone.data[['age']]))
  val<-t(hx)%*%ginv(t(bigH)%*%bigH + sigma^2/tau*diag(K))%*%hy*sigma^2
  return(val)
}
```

post.beta.mean()

This function computes the posterior mean for the parameter β.

Inputs:
bigH - the matrix **H** consisting of the basis functions evaluated at points in the training set
sigma - the estimated noise variance
tau - a scalar constant to be chosen

Output: val - the posterior mean of β

```
post.beta.mean<-function(bigH,sigma,tau){
  y<-bone.data[['spnbmd']]
  K<-length(unique(bone.data[['age']]))
  val<-ginv(t(bigH)%*%bigH + sigma^2/tau*diag(K))%*%t(bigH)%*%y
  return(val)
}
```

post.beta.cov()

This function computes the posterior covariance for the parameter β.

Inputs:
bigH - the matrix **H** consisting of the basis functions evaluated at points in the training set
sigma - the estimated noise variance
tau - a scalar constant to be chosen

Output: val - the posterior mean of β

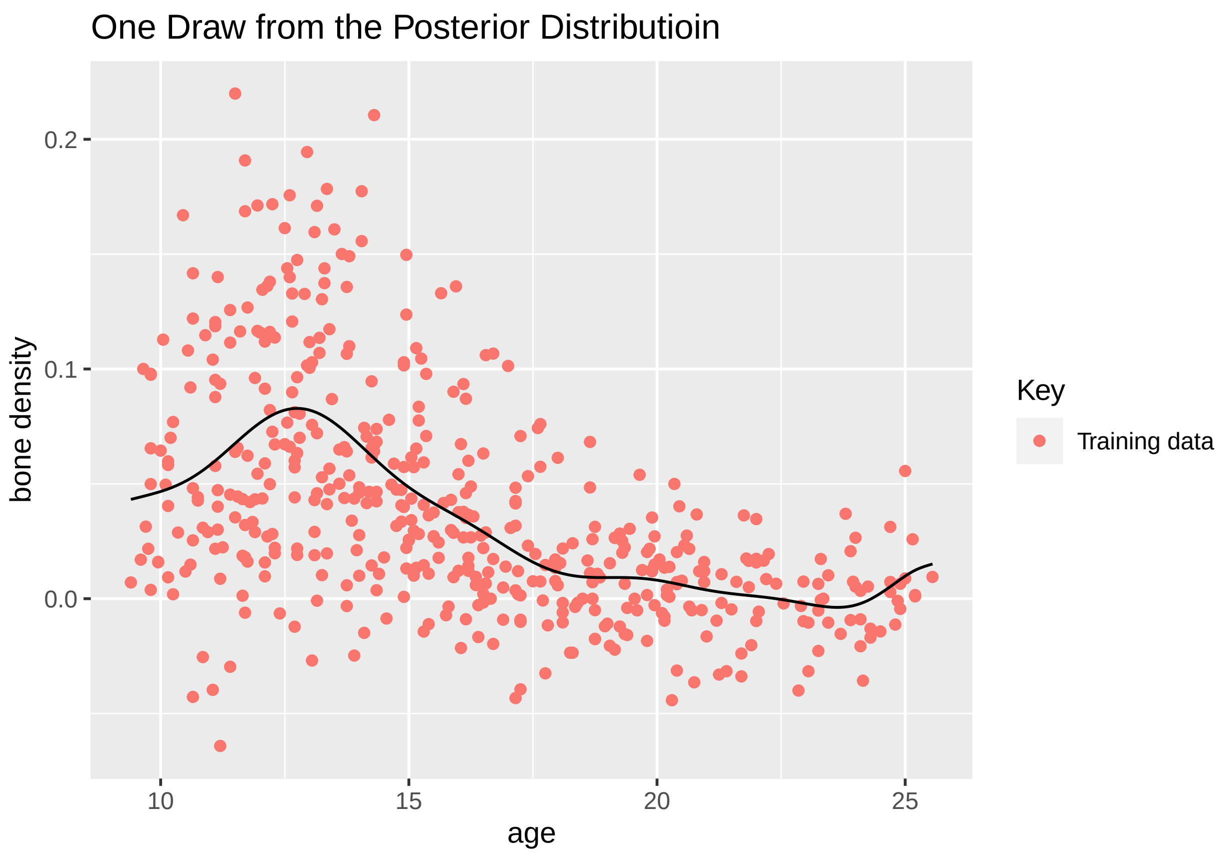
```
post.beta.cov<-function(bigH,sigma,tau){
  K<-length(unique(bone.data[['age']]))
  val<-ginv(t(bigH)%*%bigH + sigma^2/tau*diag(K))*sigma^2
  return(val)
}
```

Plots and Results

Using the posterior estimates for β, we can model realizations of the function $\mu(x)$.

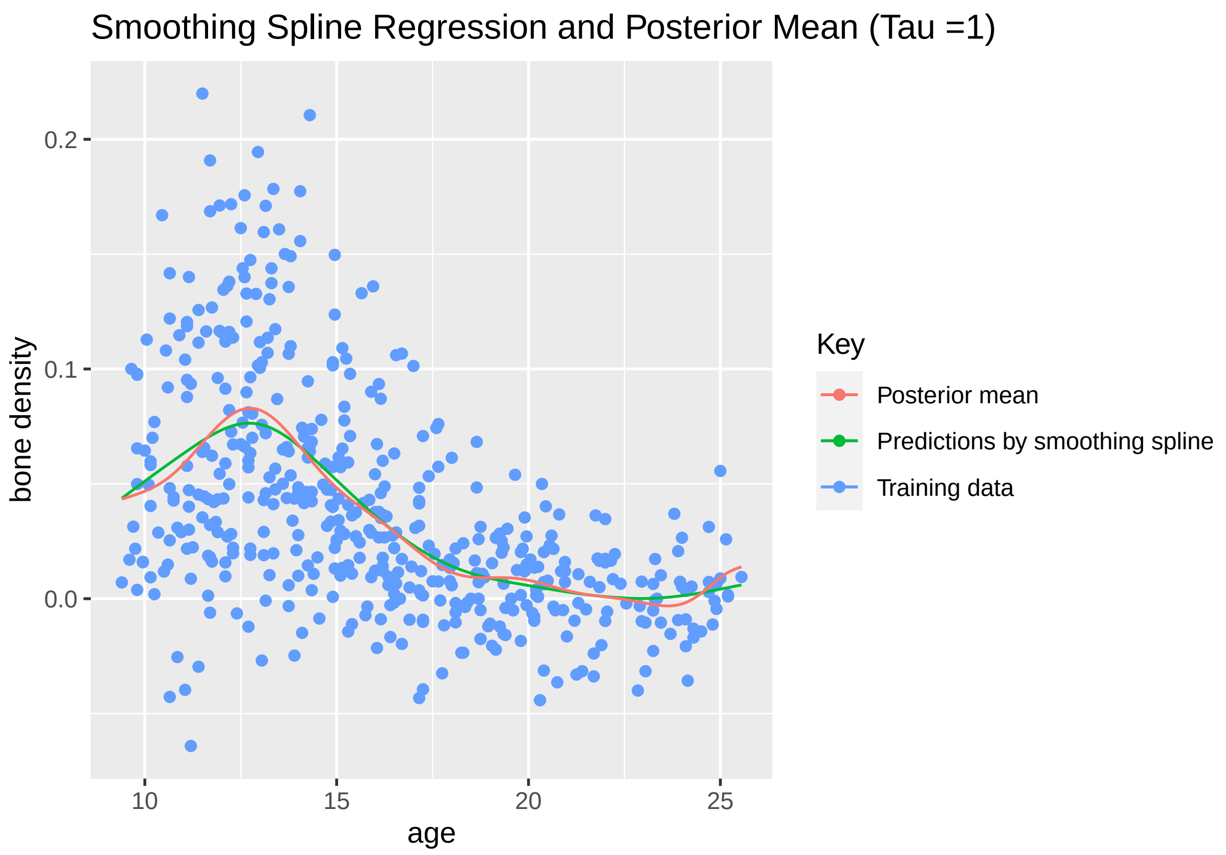
```
mean<-post.beta.mean(bigH,sigma,100)
cov<-post.beta.cov(bigH,sigma,100)
```

```
sampled.betas<-mvrnorm(1,mean,cov)

grid.size<-200

ages<-bone.data[['age']]
age.grid<-seq(from=min(ages), to=max(ages), length.out=grid.size)
K<-length(unique(ages))
hmat<-matrix(1,grid.size,K)
for (i in 1:grid.size){
  hmat[i,]<-t(little.h(age.grid[i], bone.data))
}

label1<-'Sampled mu(x)'
label2<-'Predictions by smoothing spline'
Key<-'Training data'


p<-ggplot() +
  geom_point(data=bone.data, aes(x=age, y=spnbmd, color=Key))+
  geom_line(aes(x=age.grid, y=hmat%*%sampled.betas))+


  labs(title="One Draw from the Posterior Distributioin", x="age", y="bone density")

print(p)
```



```
expts<-matrix(1,200)
varpts<-matrix(1,200)
for(i in 1:200){
  expts[i]<-posterior.mean(age.grid[i], bigH, sigma,100)
  varpts[i]<-posterior.cov(age.grid[i], age.grid[i],bigH,sigma,100)
}
```

```
label2<-'Predictions by smoothing spline'
label3<-'Posterior mean'
Key<-'Training data'

p<-ggplot() +
  geom_point(data=bone.data, aes(x=age, y=spnbmd, color=Key))+
  geom_line(aes(x=age.grid, y=preds$y, color=label2))+
  geom_line(aes(x=age.grid, y=expts, color=label3))+
  labs(title="Smoothing Spline Regression and Posterior Mean (Tau =1)", x="age", y="bone density")

print(p)
```



Bootstrapping

Bootstrapping is a way to assess uncertainty by sampling from the training data. If the training data is of the form $\mathbf{Z} = \{(x_i, y_i)\}_{i=1}^{N}$ we first draw B data sets of size N with replacement from the training data. To each bootstrap data set, $\mathbf{Z}^{i*}$, we fit a model $\hat{f}^{*,i}$. Then, using the collection of fitted models, we can form point-wise confidence bands from the percentiles at each x value where the models were evaluated.

Functions and Documentation

shuffle()

This function is used to construct the bootstrap samples.

Inputs: none

Outputs: a new data set sampled from the original one with replacement. The size of the new data set and the original data set coincide.

```
shuffle<-function(){
  num<-nrow(bone.data)
  resampled.rows<-sample(1:num, size=num, replace=TRUE)
  new.data<-bone.data[resampled.rows,]
  return(new.data)
}
```

new.spline()

This function constructs a spline estimator for a given data set and returns the predicted values of bone density along evenly spaced age values.

Inputs:
new.data - the data set containing the training data from which we form the model
grid.size - the number of points in the domain (age) on which we evaluate the model

Output: bone.preds - a vector of bone density predictions of size age.grid

```
new.spline<-function(new.data, grid.size){
  spline<-smooth.spline(x=new.data[,'age'], y=new.data[,'spnbmd'], cv=FALSE,  all.knots=TRUE)

  age.grid<-seq(from=min(bone.data[,'age']), to=max(bone.data[,'age']), length.out=grid.size)
  preds<-predict(spline,x=age.grid)

  bone.preds<-preds$y
  return(bone.preds)
}
```

spline.ci

This function forms confidence intervals about the smoothing spline estimator formed from the full training set using the bootstrap estimators. This is done by computing the values of all the bootstrap estimators at a given value of x and taking the value in the $\alpha/2$th percentile and the $1-\alpha/2$th percentile.

Inputs:
N - the number of bootstrap samples
alpha - the α level for the confidence interval
grid.size - the number of points in the domain (age) on which we evaluate the model

Output: vals - a list containing the predictions from the smoothing spline formed on the full training set, the lower bound and upper bounds for the confidence interval, and the values in the domain on which we evaluated the models

```
spline.ci <- function(N,alpha, grid.size){
  first.spline<-new.spline(bone.data, grid.size)
  bootstrapped.splines<-replicate(N, new.spline(shuffle(), grid.size))

  l.ci<-2*first.spline-apply(bootstrapped.splines, 1, quantile, probs=1-alpha/2)
  u.ci<-2*first.spline-apply(bootstrapped.splines, 1, quantile, probs=alpha/2)

  ages<-bone.data[,'age']
  age.grid<-seq(from=min(ages), to=max(ages), length.out=grid.size)

  vals<-list(orig.spline=first.spline, lower.ci=l.ci, upper.ci=u.ci, x=age.grid)
  return(vals)

}
```

plot.splines()

This function plots the training data, the smoothing spline and the confidence intervals formed from the bootstrap samples.

Inputs:
c.ints - a list containing the predictions from the smoothing spline formed on the full training set, the lower

bound and upper bounds for the confidence interval, and the values in the domain on which we evaluated the models. This is designed to be the output from spline.ci()

Outputs:
p - a plot

```r
plot.splines<-function(c.ints){

label1<-'90% confidence intervals'
label2<-'Predictions by smoothing spline'
Key<-'Training data'

p<-ggplot() +
  geom_point(data=bone.data, aes(x=age, y=spnbmd, color=Key))+
  geom_line(aes(x=c.ints$x, y=c.ints$orig.spline, color=label2))+
  geom_ribbon(aes(x=c.ints$x, ymin=c.ints$lower.ci, ymax=c.ints$upper.ci, color=label1), alpha=.2)+
  labs(title="Smoothing Spline Regression on Bone Data", x="age", y="bone density")
return(p)
}
```

Plots and Results

```r
spline<-smooth.spline(x=bone.data[,'age'], y=bone.data[,'spnbmd'], cv=FALSE,  all.knots=TRUE)

sep<-"\n"

string1<- paste("Using generalized cross validation, the value of lambda used is", spline$lambda,".")
string2<-paste(sep, "The effective degrees of freedom are: ", spline$df,".")

cat(string1)
```

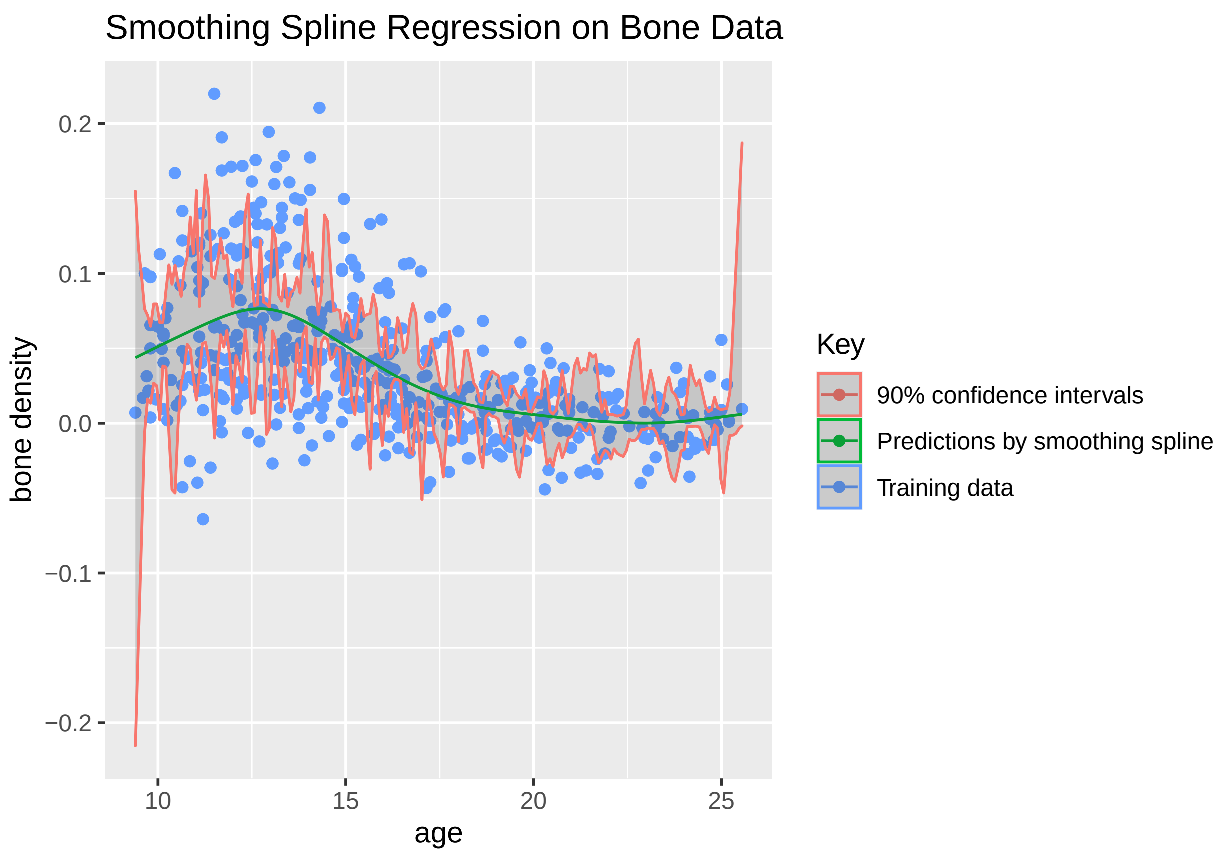
```
## Using generalized cross validation, the value of lambda used is 0.00737761533151351 .
```

```r
cat(string2)
```

```
##
##  The effective degrees of freedom are:  6.48906159411871 .
```

Using 100 bootstrap samples, we can compute a point-wise 90% confidence interval by taking the middle 90% of the bootstrap values at each value of x.

```r
alpha<-.1
num.boots<-100
c.ints<-spline.ci(num.boots,alpha,200)

p<-plot.splines(c.ints)
print(p)
```



#

Cross Validation

Suppose that $\mathbf{Z}^{*1}, \ldots, \mathbf{Z}^{*,B}$ are B bootstrap samples from the training data $\mathbf{Z} = \{(x_i, y_i)\}$. Then if the estimate $\hat{f}^{*,b}$ is formed from the data set $\mathbf{Z}^{*,b}$, we can define the aggregated bootstrap estimator to be

$$\hat{f}^{*}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*,b}(x).$$

This is also sometimes referred to as the bagging estimate.

Let $\kappa : \{1, \ldots, N\} \to \{1, \ldots, K\}$ be an indexing function that indicates the partition of the data for which observation i is assigned to. If $\hat{f}^{-k}(x)$ corresponds to the fitted function with the kth part of the data removed, the cross-validation estimate of prediction error for a model $\hat{f}$ is defined as

$$CV(\hat{f}) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, \hat{f}^{-\kappa(i)}(x_i)\right).$$

Given a set of models $f(x, \alpha)$ indexed by a tuning parameter α, we can denote $\hat{f}^{-k}(x)$ as the αth model fit with the kth part of the data removed.Then, the cross-validation estimate for this set of models is given by

$$CV(\hat{f}, \alpha) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha)\right).$$

We can think of the bagging estimate as an aggregated model where α is a coefficient weight of $1/B$ on each bootstrap estimate that forms a component of the model. In the following, we use an L^2 loss function, meaning

$$L\left(y_i, \hat{f}^{-\kappa(i)}(x_i, \alpha)\right) = \left(y_i - \hat{f}^{-\kappa(i)}(x_i, \alpha)\right)^2.$$

Functions and Documentation

cv.fit()

This function computes the cross validation estimate of prediction error for a single model $\hat{f}$ constructed via smoothing spline regression. The value λ is chosen via generalized cross validation.

Inputs:
None

Outputs:
val - the cross validation estimate for prediction error using the L^2 loss function.

```r
cv.fit<-function(){
  N<-nrow(bone.data)
  sum.loss<-0
  for(i in 1:N){
    yi<-bone.data[['spnbmd']][i]
    xi<-bone.data[['age']][i]
    new.data<-bone.data[bone.data$'age'!=xi,]
    fitj<-smooth.spline(x=new.data[['age']], y=new.data[['spnbmd']], cv=FALSE, all.knots=TRUE)
    est.yi<-predict(fitj,xi)$y
    sum.loss<-sum.loss+(yi-est.yi)^2
  }
  val<-sum.loss/N
  return(val)
}
```

get.bootstrap.samples()

This function performs B samples of size N with replacement from the training data.

Inputs:
B - the number of samples

Outputs:
samples - a matrix where each row corresponds to a bootstrap sample. The entries in the row correspond to the indices i of the sampled data.

```r
get.bootstrap.samples<-function(B){
  num<-nrow(bone.data)
  samples<-matrix(1,B,num)
  for(i in 1:B){
    resampled.rows<-sample(1:num, size=num, replace=TRUE)
    samples[i,]<-resampled.rows
  }
  return(samples)
}
```

cv.bag()

This function computes the cross validation estimate for prediction error on the bootstrap aggregation model.

Inputs:
samples - a matrix whose rows correspond to one bootstrap sample. Designed to be the output of get.bootstrap.samples()

Outputs:
val - the cross validation estimate for prediction error using the L^2 loss function.

```
cv.bag<-function(samples){
  B<-nrow(samples)
  N<-nrow(bone.data)
  sum.loss<-0
  for(i in 1:N){
    yi<-bone.data[['spnbmd']][i]
    xi<-bone.data[['age']][i]
    est.yi<-0

    for(j in 1:B){
      new.data<-bone.data[samples[j,],]
      new.data<-new.data[new.data$'age'!=xi,]
      fitj<-smooth.spline(x=new.data[['age']], y=new.data[['spnbmd']], cv=FALSE, all.knots=TRUE)
      est.yi<-est.yi+predict(fitj,xi)$y
    }
    hat.f<-1/B*est.yi
    sum.loss<-sum.loss+(yi-hat.f)^2
  }
  val<-sum.loss/N
  return(val)
}
```

Results and Plots

```
B<-100
samples<-get.bootstrap.samples(B)
error<-cv.bag(samples)

string<-paste("The estimated error for the aggregated bootstrap estimator with ", B, " samples is: ", e
cat(string)
```

```
## The estimated error for the aggregated bootstrap estimator with  100  samples is:  0.001986201315946
```

```
error<-cv.fit()
string<-paste("The estimated error for the smoothing spline estimator with is: ", error)
cat(string)
```

```
## The estimated error for the smoothing spline estimator with is:  0.00166378569965835
```