Methods and Models Comparison

Elena Kargopoltsev

01 03 2021

#1.Read and revise the data set

```
House=read.csv("kc house sales.csv")
House$date=as.Date(House$date,format="%Y%m%d")
House$view=as.factor(House$view)
House$waterfront=as.factor(House$waterfront)
House$zipcode=as.factor(House$zipcode)
House[15871,3]=3
attach(House)
```

#2.Consider a multiple regression with log(price) as the response and all other variables and their two-way interactions as the predictors. Among the variables, exclude sqft_basement since it is linearly dependent on other variables. Also, for zipcode consider the main effect only (do not consider its interaction with any other variables).Find the minimum Cp and the number of predictors in the best model. Plot BIC and mark the minimum point on the plot. Find the minimum BIC and the number of predictors in the best model.

```
library(leaps)
all.fit=regsubsets(log(price)~(.-sqft_basement-zipcode)^2+zipcode, data=House, nvmax=20, method="forward")
```

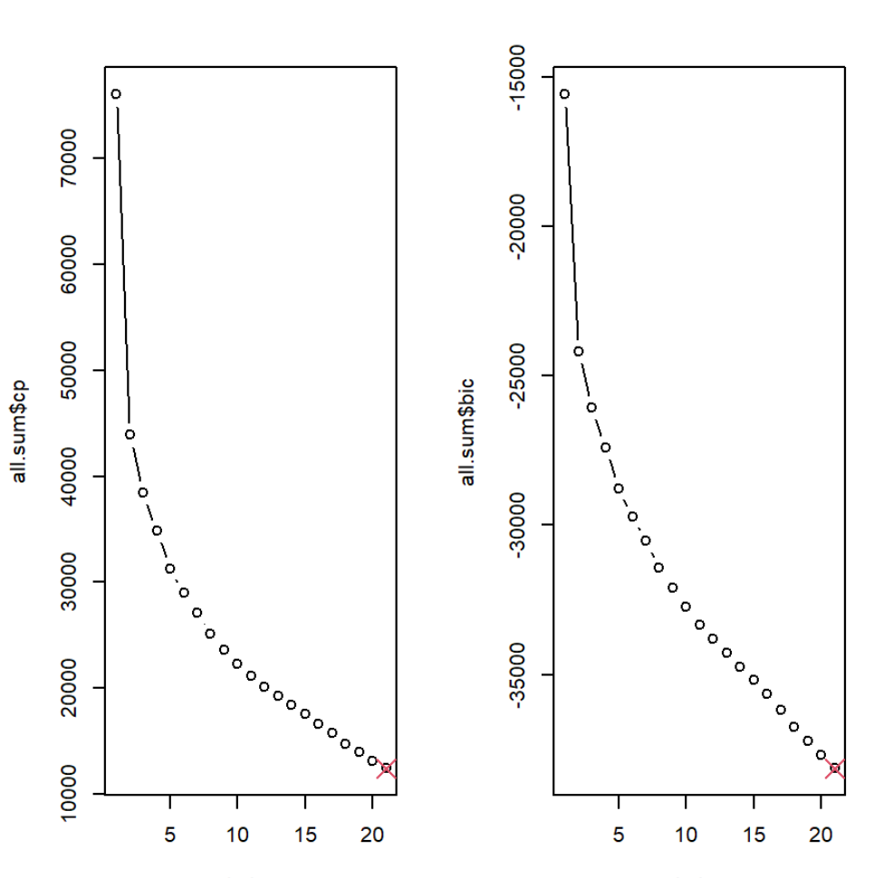
```
## Reordering variables and trying again:
```

```
all.sum=summary(all.fit)
all.sum$cp
```

```
##  [1] 75964.07 43873.04 38396.44 34814.25 31298.66 29002.95 27127.30 25096.23
##  [9] 23667.52 22330.60 21131.72 20179.34 19257.24 18404.42 17551.73 16684.65
## [17] 15769.09 14764.00 13980.81 13183.67 12466.26
```

```
par(mfrow=c(1,3))
plot(all.sum$cp,type="b")
mincp=which.min(all.sum$cp)
points(mincp,all.sum$cp[mincp],pch=4,cex=2,col=2)

plot(all.sum$bic,type="b")
minbic=which.min(all.sum$bic)
points(minbic,all.sum$bic[minbic],pch=4,cex=2,col=2)
```



#3.Repeat question 2 for the backward stepwise selection.

```
all.fit=regsubsets(log(price)~(.-sqft_basement-zipcode)^2+zipcode, data=House, nvmax=20, method="backward")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 1 linear dependencies found
```

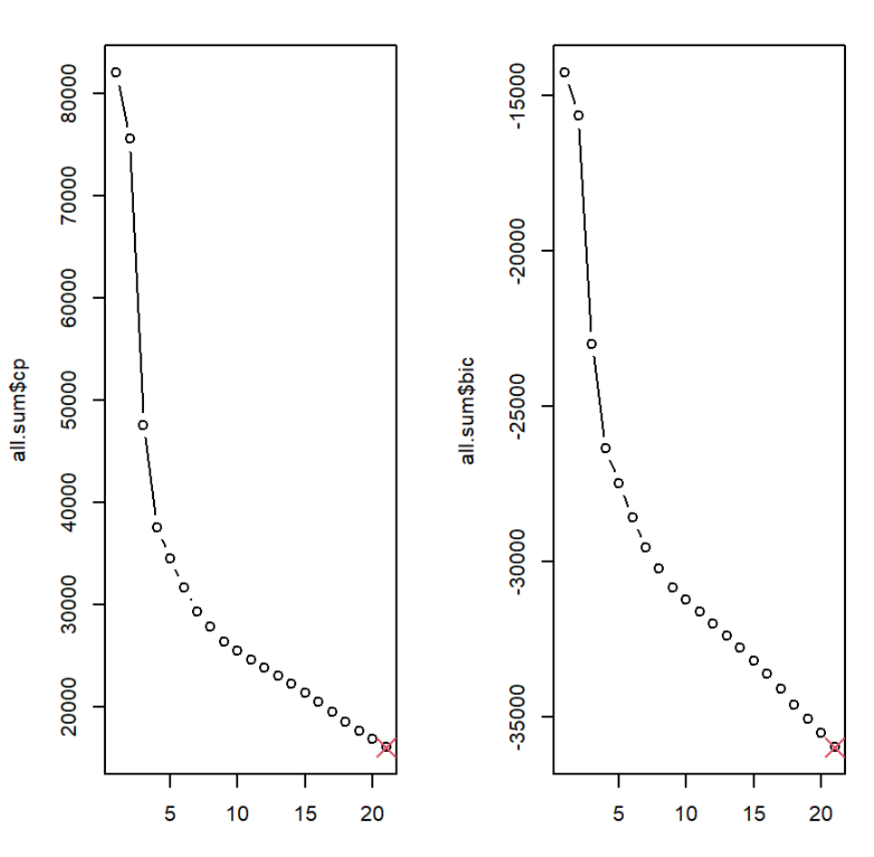
```
## Reordering variables and trying again:
```

```
all.sum=summary(all.fit)
all.sum$cp
```

```
##  [1] 82022.33 75564.04 47581.52 37565.25 34522.03 31717.20 29372.43 27806.55
##  [9] 26413.25 25511.84 24659.96 23816.21 23019.93 22223.45 21370.49 20471.57
## [17] 19559.43 18589.38 17711.15 16875.52 16062.45
```

```
par(mfrow=c(1,3))
plot(all.sum$cp,type="b")
mincp=which.min(all.sum$cp)
points(mincp,all.sum$cp[mincp],pch=4,cex=2,col=2)

plot(all.sum$bic,type="b")
minbic=which.min(all.sum$bic)
points(minbic,all.sum$bic[minbic],pch=4,cex=2,col=2)
```



#4.Compare the two stepwise selection methods. Which of the two selection methods is better

#For a competent comparison of the two sample models, I used the minimum cp. And based on this indicator, the first method would be the best.

#5.Find the AIC of the model.

```
all.glm=glm(log(price)~.-sqft_basement+bedrooms:sqft_living+bedrooms:sqft_lot+bedrooms:floors+bedrooms:grade+bedr
ooms:sqft_above+bedrooms:long+bedrooms:sqft_lot15+bathrooms:sqft_lot+bathrooms:floors+bathrooms:sqft_above+bathro
oms:yr_built+bathrooms:long+bathrooms:sqft_lot15+sqft_living:floors+sqft_living:sqft_above+sqft_living:yr_renovat
ed+sqft_living:lat+sqft_living:long+sqft_living:sqft_living15+sqft_living:view+sqft_lot:yr_built+sqft_lot:sqft_lo
t15+floors:yr_built+floors:lat+condition:grade+condition:yr_built+condition:yr_renovated+condition:sqft_living+gr
ade:long+grade:waterfront+sqft_above:yr_built+sqft_above:lat+yr_built:long+yr_built:sqft_living15+yr_renovated:la
t+yr_renovated:view+lat:sqft_lot15+lat:waterfront+long:date+long:view+long:waterfront+view:waterfront, House, fam
ily=gaussian)
all.glm$aic
```

```
## [1] -14301.49
```

#6.Use the cv.glm() function to perform 10-fold CV on the model used in the previous question. Use set.seed(100) before performing the cross validation.

```
library(boot)
set.seed(100)
all.cvglm=cv.glm(House,all.glm, K=10)
all.cvglm$delta
```

```
## [1] 0.03042268 0.03038860
```

#7.Consider the model discussed in question 2. Plot the results. Find the minimum (given in a variable called lambda.min) and its corresponding test MSE (given in a variable called cvm).

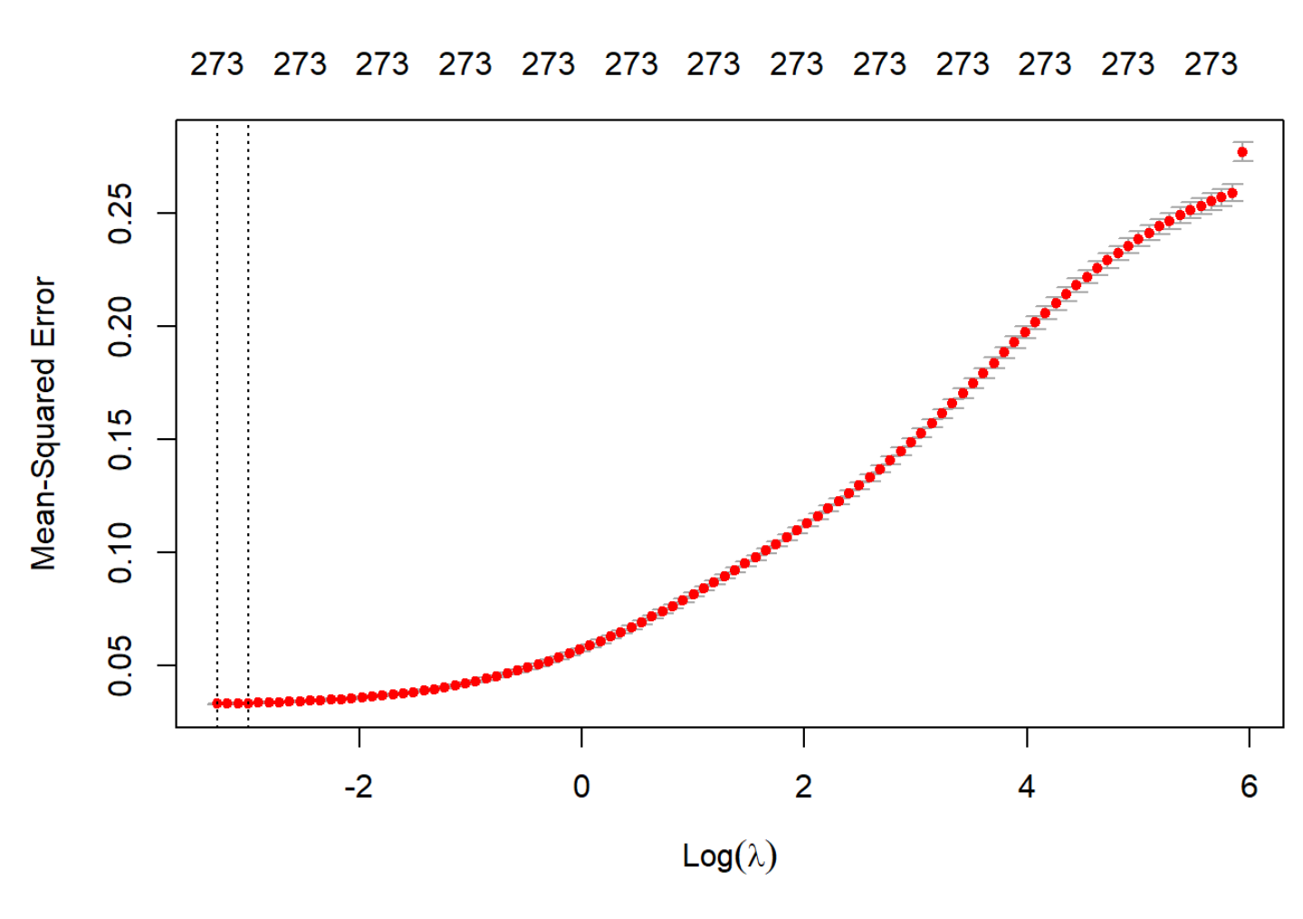
```
library(Matrix)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

```
## Loaded glmnet 4.1-4
```

```
x=model.matrix(log(price)~(.-sqft_basement-zipcode)^2+zipcode, data=House)
y=log(price)

cv.out=cv.glmnet(x,y,alpha=0, labda=100)
plot(cv.out)
```



```
cv.out$lambda.min
```

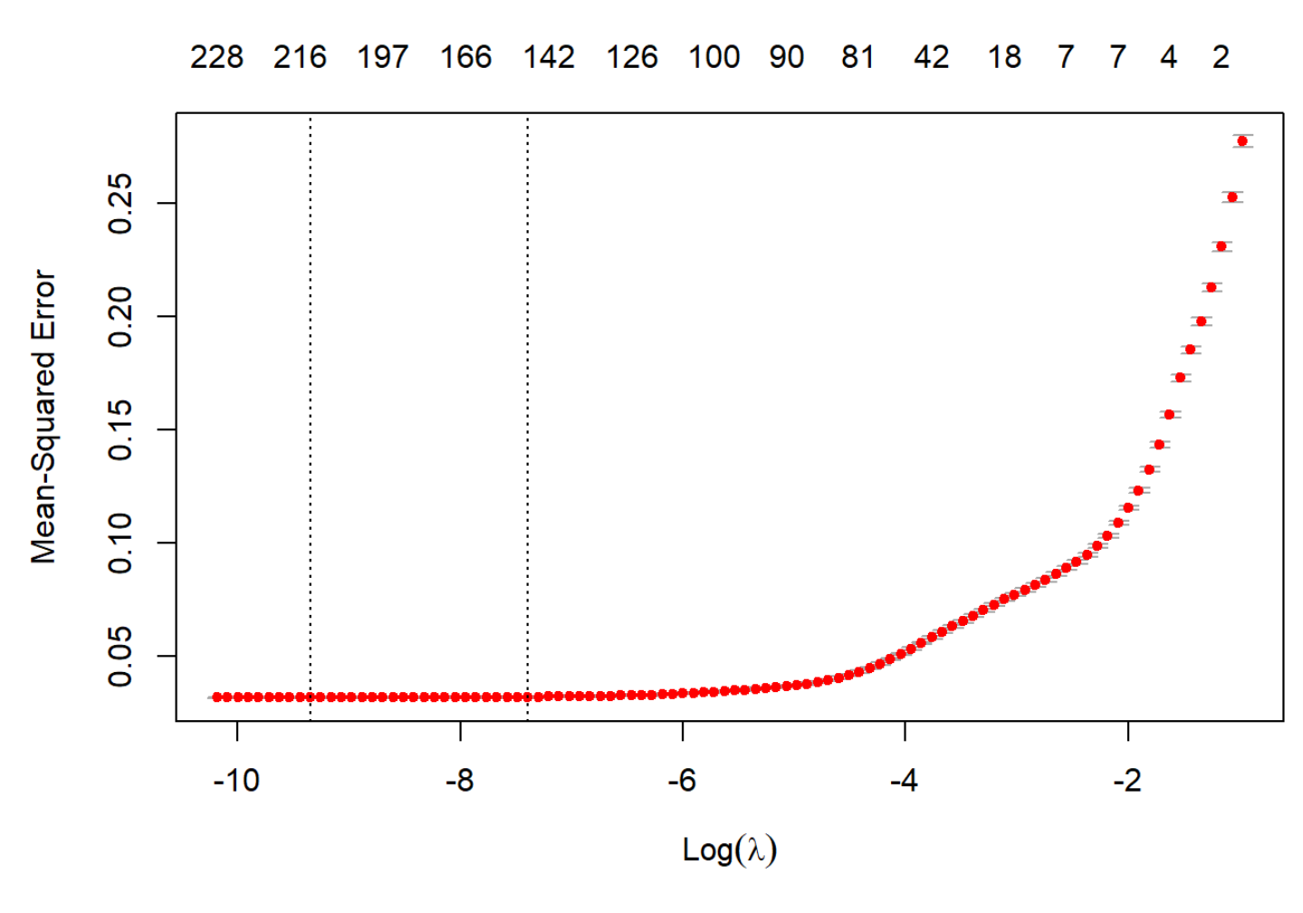
```
## [1] 0.03777521
```

```
min.r=which.min(cv.out$cvm)
cv.out$cvm[min.r]
```

```
## [1] 0.03278221
```

#8.Repeat question 7 for LASSO.

```
cv.out=cv.glmnet(x,y,alpha=1, labda=100)
plot(cv.out)
```



```
cv.out$lambda.min
```

```
## [1] 8.726563e-05
```

```
min.r=which.min(cv.out$cvm)
cv.out$cvm[min.r]
```

```
## [1] 0.03178324
```

#9.Compare the results of the above three models, backward selection in question 5, the ridge regression, and the LASSO. Which of the three models does provide the smallest test MSE?

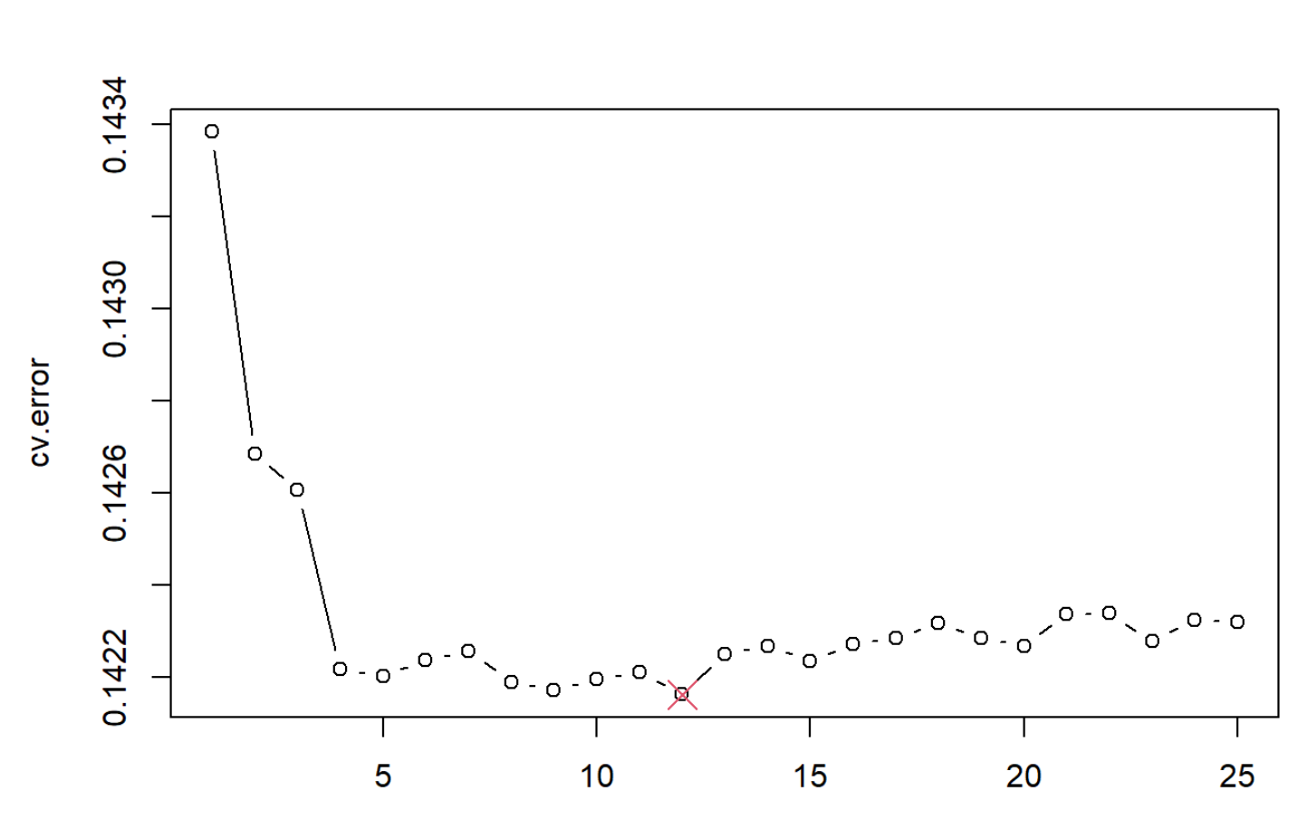
#When comparing the three models, the first model had the smallest test MSE.

#10.Use cross-validation to select the optimal degrees of freedom for a natural cubic spline which predicts log(price) using sqft_living. Use set.seed(100) and vary the degrees of freedom from 1 to 25. Plot the cross-validation errors and mark the minimum point on the plot.

```
library(splines)
cv.error=rep(0, 10)
set.seed(100)
for (i in 1:25) {
ns.fit=glm(log(price)~ns(sqft_living,df=i))
cv.error[i]=cv.glm(House,ns.fit,K=10)$delta[1]
}
cv.error
```

```
##  [1] 0.1433848 0.1426855 0.1426061 0.1422179 0.1422015 0.1422373 0.1422567
##  [8] 0.1421903 0.1421717 0.1421960 0.1422113 0.1421627 0.1422511 0.1422674
## [15] 0.1422359 0.1422721 0.1422853 0.1423176 0.1422848 0.1422678 0.1423372
## [22] 0.1423384 0.1422786 0.1423241 0.1423195
```

```
plot(cv.error, type="b")
df.min=which.min(cv.error)
points(df.min,cv.error[df.min],col=2,pch=4,cex=2)
```



#11.Fit a natural cubic spline with the optimal degrees of freedom obtained in the previous question. Find the location of the knots using attr(). Use the summary() function to print the results of the fit. Evaluate the results.

```
ns.fit=glm(log(price)~ns(sqft_living,df=12))
attr(ns(sqft_living,df=12),"knots")
```

```
## 8.333333% 16.66667%       25% 33.33333% 41.66667%       50% 58.33333% 66.66667% 
##      1040      1250      1427      1580      1740      1910      2100      2300 
##       75% 83.33333% 91.66667% 
##      2550      2880      3400
```

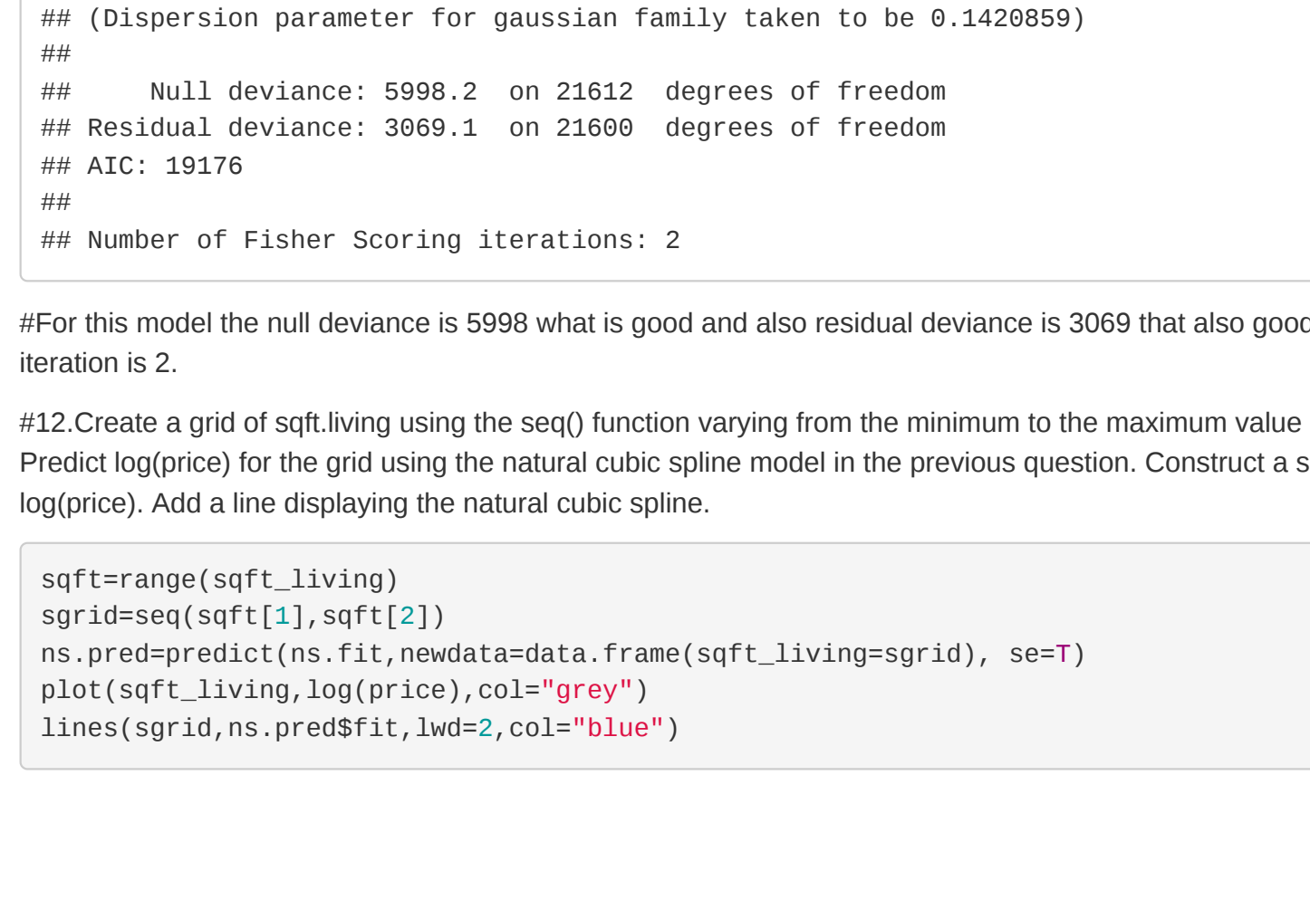
```
summary(ns.fit)
```

```
## 
## Call:
## glm(formula = log(price) ~ ns(sqft_living, df = 12))
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -1.22606  -0.28482   0.01267   0.26152   1.27019  
## 
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                12.14237    0.06735 180.287   <2e-16 ***
## ns(sqft_living, df = 12)1   0.58139    0.06201   9.376   <2e-16 ***
## ns(sqft_living, df = 12)2   0.63208    0.07331   8.622   <2e-16 ***
## ns(sqft_living, df = 12)3   0.74925    0.06780  11.050   <2e-16 ***
## ns(sqft_living, df = 12)4   0.75321    0.07106  10.599   <2e-16 ***
## ns(sqft_living, df = 12)5   0.80095    0.06950  11.524   <2e-16 ***
## ns(sqft_living, df = 12)6   0.91725    0.07012  13.081   <2e-16 ***
## ns(sqft_living, df = 12)7   1.00108    0.06951  14.403   <2e-16 ***
## ns(sqft_living, df = 12)8   1.10500    0.06928  15.951   <2e-16 ***
## ns(sqft_living, df = 12)9   1.27341    0.06823  18.663   <2e-16 ***
## ns(sqft_living, df = 12)10  2.68240    0.05876  45.651   <2e-16 ***
## ns(sqft_living, df = 12)11  3.40100    0.17397  19.549   <2e-16 ***
## ns(sqft_living, df = 12)12  3.01637    0.21019  14.351   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 0.1420859)
## 
##     Null deviance: 5998.2  on 21612  degrees of freedom
## Residual deviance: 3069.1  on 21600  degrees of freedom
## AIC: 19176
## 
## Number of Fisher Scoring iterations: 2
```

#For this model the null deviance is 5998 what is good and also residual deviance is 3069 that also good, aic 19176 and number of fisher scoring iteration is 2.

#12.Create a grid of sqft.living using the seq() function varying from the minimum to the maximum value in the data set with an increment of one. Predict log(price) for the grid using the natural cubic spline model in the previous question. Construct a scatter plot between sqft_living and log(price). Add a line displaying the natural cubic spline.

```
sqft=range(sqft_living)
sgrid=seq(sqft[1],sqft[2])
ns.pred=predict(ns.fit,newdata=data.frame(sqft_living=sgrid), se=T)
plot(sqft_living,log(price),col="grey")
lines(sgrid,ns.pred$fit,lwd=2,col="blue")
```

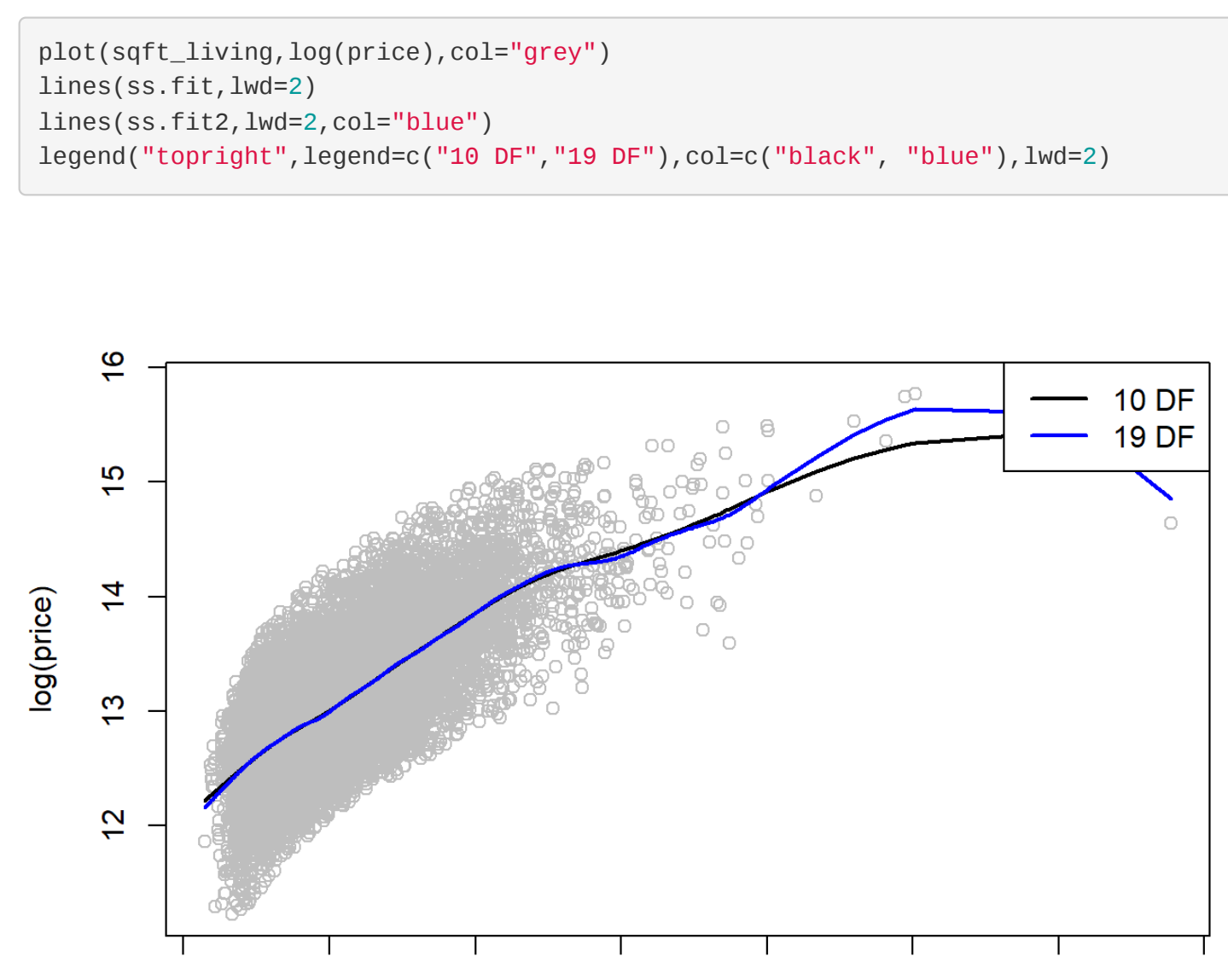


#13.Use the smooth.spline() function to fit a smoothing spline to predict log(price) using sqft_living. Use CV=T in the function to find the optimal degrees of freedom for the spline using cross validation. Construct a scatter plot between sqft_living and log(price). Add a line displaying the smoothing spline with the optimal degrees of freedom. Add another line displaying the smoothing spline with 10 degrees of freedom. There may be warnings due to non-unique values.

```
ss.fit=smooth.spline(sqft_living,log(price),df=10)
ss.fit2=smooth.spline(sqft_living,log(price),cv=T)
ss.fit2$df
```

```
## [1] 18.84337
```

```
plot(sqft_living,log(price),col="grey")
lines(ss.fit,lwd=2)
lines(ss.fit2,lwd=2,col="blue")
legend("topright",legend=c("10 DF","19 DF"),col=c("black", "blue"),lwd=2)
```



#14.Repeat question 13 using lat as the predictor. There may be warnings due to non-unique values.

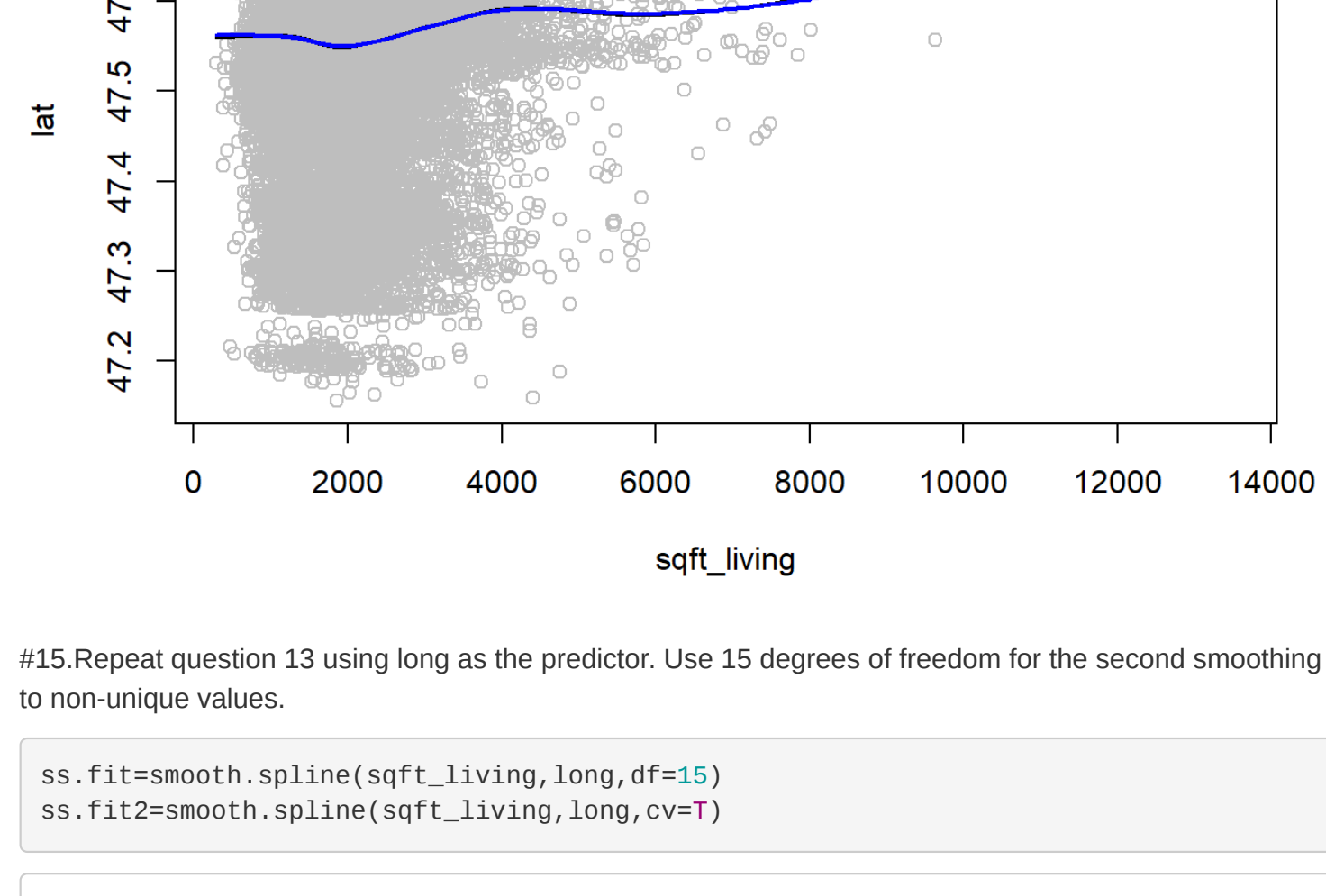
```
ss.fit=smooth.spline(sqft_living,lat,df=10)
ss.fit2=smooth.spline(sqft_living,lat,cv=T)
```

```
## Warning in smooth.spline(sqft_living, lat, cv = T): cross-validation with non-
## unique 'x' values seems doubtful
```

```
ss.fit2$df
```

```
## [1] 8.798056
```

```
plot(sqft_living,lat,col="grey")
lines(ss.fit,lwd=2)
lines(ss.fit2,lwd=2,col="blue")
legend("topright",legend=c("10 DF","9 DF"),col=c("black", "blue"),lwd=2)
```



#15.Repeat question 13 using long as the predictor. Use 15 degrees of freedom for the second smoothing spline line. There may be warnings due to non-unique values.

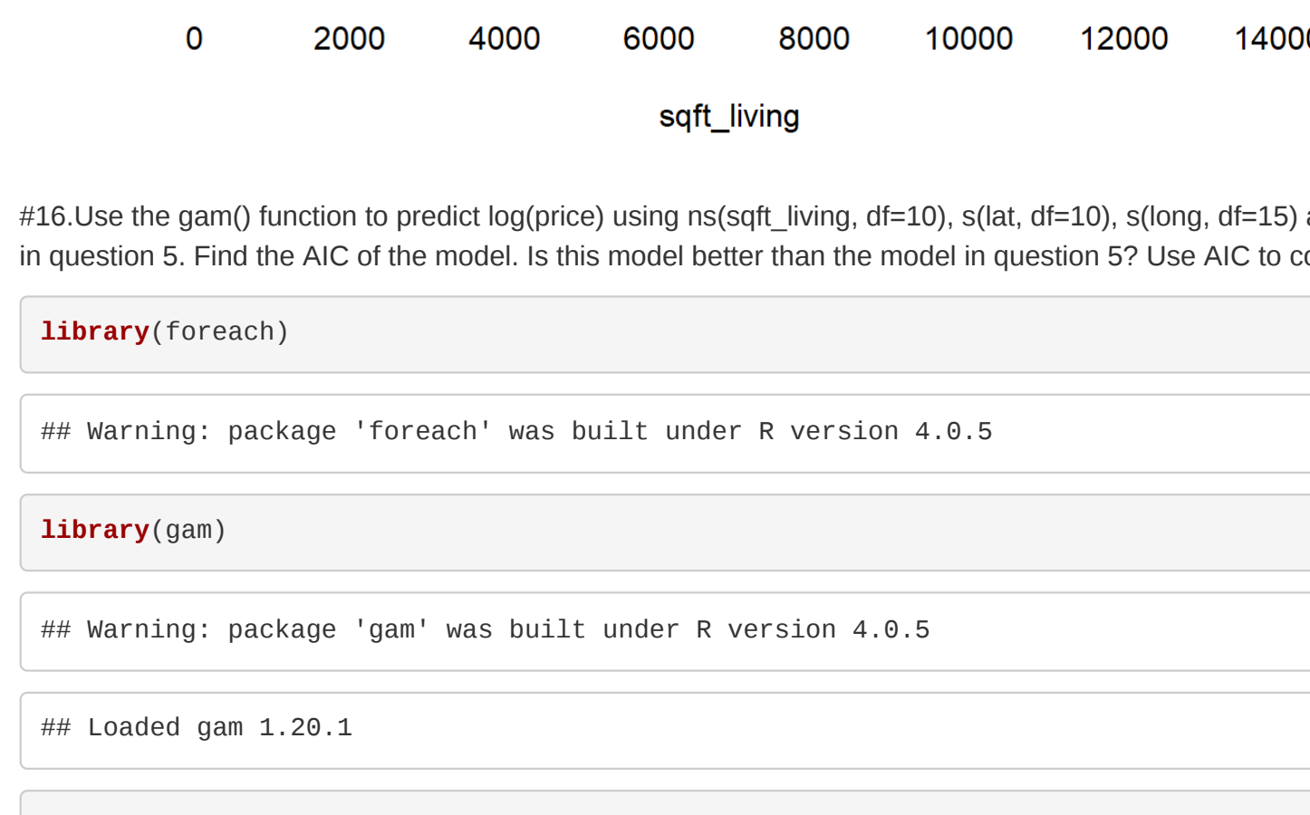
```
ss.fit=smooth.spline(sqft_living,long,df=15)
ss.fit2=smooth.spline(sqft_living,long,cv=T)
```

```
## Warning in smooth.spline(sqft_living, long, cv = T): cross-validation with non-
## unique 'x' values seems doubtful
```

```
ss.fit2$df
```

```
## [1] 4.911937
```

```
plot(sqft_living,long,col="grey")
lines(ss.fit,lwd=2)
lines(ss.fit2,lwd=2,col="blue")
legend("topright",legend=c("15 DF","5 DF"),col=c("black", "blue"),lwd=2)
```



#16.Use the gam() function to predict log(price) using ns(sqft_living, df=10), s(lat, df=10), s(long, df=15) and all other main effects and interactions in question 5. Find the AIC of the model. Is this model better than the model in question 5? Use AIC to compare the models.

```
library(foreach)
```

```
## Warning: package 'foreach' was built under R version 4.0.5
```

```
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.0.5
```

```
## Loaded gam 1.20.1
```

```
gam.fit=gam(log(price)~.+ns(sqft_living,df=10)+s(lat,df=10)+s(long,df=15)-sqft_basement+bedrooms:sqft_living+bedr
ooms:sqft_lot+bedrooms:floors+bedrooms:grade+bedrooms:sqft_above+bedrooms:long+bedrooms:sqft_lot15+bathrooms:sqft
_lot+bathrooms:floors+bathrooms:sqft_above+bathrooms:yr_built+bathrooms:long+bathrooms:sqft_lot15+sqft_living:flo
ors+sqft_living:sqft_above+sqft_living:yr_renovated+sqft_living:lat+sqft_living:long+sqft_living:sqft_living15+sq
ft_living:view+sqft_lot:yr_built+sqft_lot:sqft_lot15+floors:yr_built+floors:lat+condition:grade+condition:yr_buil
t+condition:yr_renovated+condition:sqft_living+grade:long+grade:waterfront+sqft_above:yr_built+sqft_above:lat+yr_
built:long+yr_built:sqft_living15+yr_renovated:lat+yr_renovated:view+lat:sqft_lot15+lat:waterfront+long:date+lon
g:view+long:waterfront+view:waterfront, data=House)
gam.fit$aic
```

```
## [1] -15629.64
```

#This model is best because has a smaller aic.

#Extra

```
set.seed(10)
cv.error=rep(0, 10)
k=10
for(i in 1:k){
house.fit=glm(log(price)~.+ns(sqft_living,df=10)+s(lat,df=10)+s(long,df=15)-sqft_basement+bedrooms:sqft_living+be
drooms:sqft_lot+bedrooms:floors+bedrooms:grade+bedrooms:sqft_above+bedrooms:long+bedrooms:sqft_lot15+bathrooms:sq
ft_lot+bathrooms:floors+bathrooms:sqft_above+bathrooms:yr_built+bathrooms:long+bathrooms:sqft_lot15+sqft_living:f
loors+sqft_living:sqft_above+sqft_living:yr_renovated+sqft_living:lat+sqft_living:long+sqft_living:sqft_living15+
sqft_living:view+sqft_lot:yr_built+sqft_lot:sqft_lot15+floors:yr_built+floors:lat+condition:grade+condition:yr_bu
ilt+condition:yr_renovated+condition:sqft_living+grade:long+grade:waterfront+sqft_above:yr_built+sqft_above:lat+y
r_built:long+yr_built:sqft_living15+yr_renovated:lat+yr_renovated:view+lat:sqft_lot15+lat:waterfront+long:date+lo
ng:view+long:waterfront+view:waterfront, data=House, family=gaussian)

cv.error[i]=cv.glm(House,house.fit,K=10)$delta[1]
}
cv.error
```

```
##  [1] 0.03031268 0.03026746 0.03030724 0.03025463 0.03029429 0.03028774
##  [7] 0.03035161 0.03032994 0.03034355 0.03028529
```

#This model is best because has a smaller test MSE.