PSTAT127_HW7

Celine Mol

March 7, 2017

2. Load library "MASS" into R, and run the ridge regression example at the end of the lm.ridge help file. Write down the model that is being fitted (with assumptions) and explain the results of each line of code.

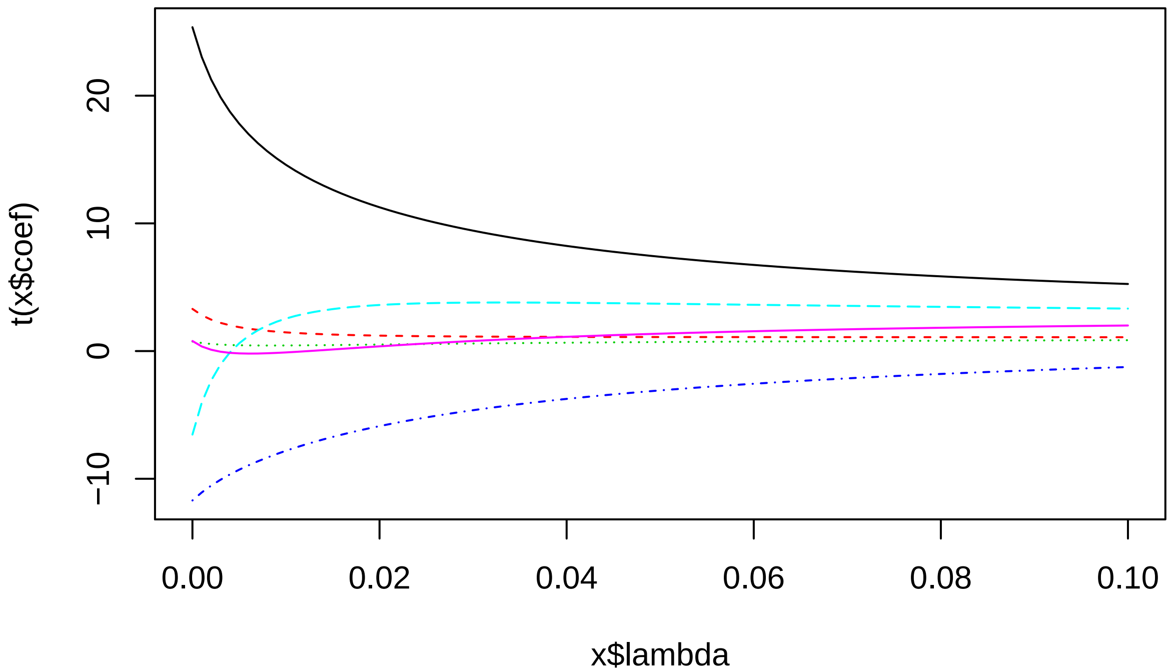
```
library(MASS)
longley # not the same as the S-PLUS dataset
```

```
##      GNP.deflator     GNP Unemployed Armed.Forces Population Year Employed
## 1947         83.0 234.289      235.6        159.0    107.608 1947   60.323
## 1948         88.5 259.426      232.5        145.6    108.632 1948   61.122
## 1949         88.2 258.054      368.2        161.6    109.773 1949   60.171
## 1950         89.5 284.599      335.1        165.0    110.929 1950   61.187
## 1951         96.2 328.975      209.9        309.9    112.075 1951   63.221
## 1952         98.1 346.999      193.2        359.4    113.270 1952   63.639
## 1953         99.0 365.385      187.0        354.7    115.094 1953   64.989
## 1954        100.0 363.112      357.8        335.0    116.219 1954   63.761
## 1955        101.2 397.469      290.4        304.8    117.388 1955   66.019
## 1956        104.6 419.180      282.2        285.7    118.734 1956   67.857
## 1957        108.4 442.769      293.6        279.8    120.445 1957   68.169
## 1958        110.8 444.546      468.1        263.7    121.950 1958   66.513
## 1959        112.6 482.704      381.3        255.2    123.366 1959   68.655
## 1960        114.2 502.601      393.1        251.4    125.368 1960   69.564
## 1961        115.7 518.173      480.6        257.2    127.852 1961   69.331
## 1962        116.9 554.894      400.7        282.7    130.081 1962   70.551
```

```
names(longley)[1] <- "y" #Changing the variable name for the first column to "y"
lm.ridge(y ~ ., longley) #Fitting a ridge regression model
```

```
##                        GNP    Unemployed  Armed.Forces    Population
## 2946.85636017   0.26352725    0.03648291    0.01116105   -1.73702984
##          Year     Employed
##   -1.41879853   0.23128785
```

```
plot(lm.ridge(y ~ ., longley,
              lambda = seq(0,0.1,0.001)))
```



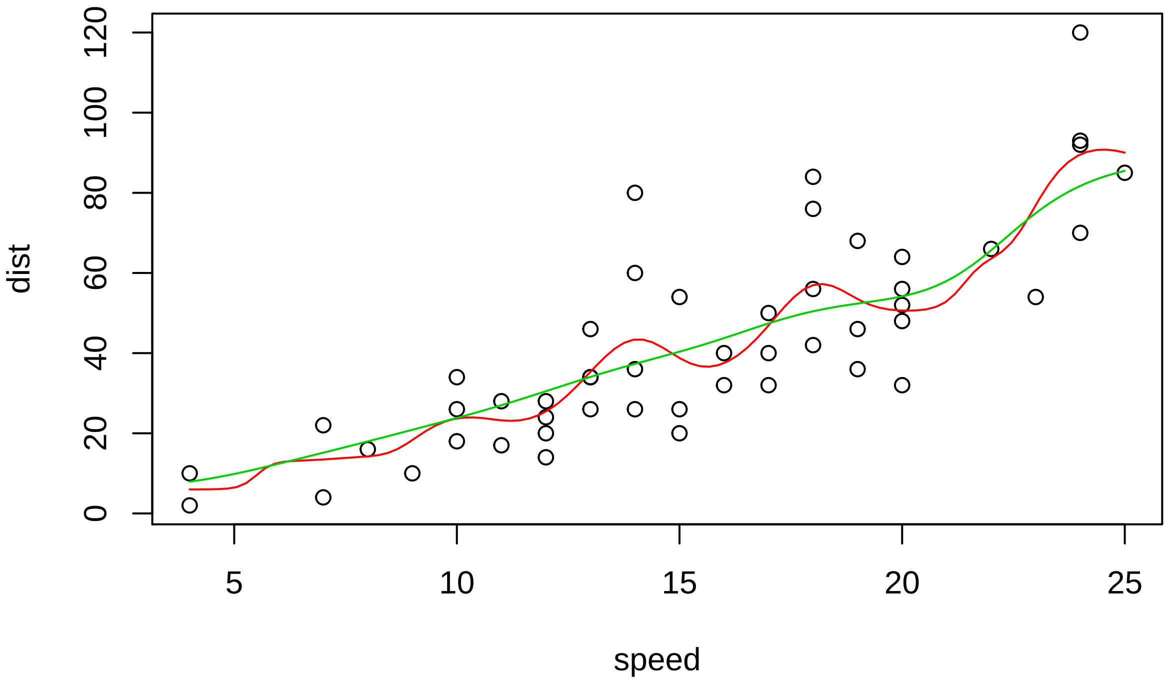

```r
select(lm.ridge(y ~ ., longley,
                lambda = seq(0,0.1,0.0001)))
```

```
## modified HKB estimator is 0.006836982
## modified L-W estimator is 0.05267247
## smallest value of GCV  at 0.0057
```

The model that is being fitted is Yi = Bo + B1(GNP)i + B2(Unemployed)i + B3(Armed.Forces)i + B4(Population)i + B5(Year)i + B6(Employed)i
Model assumptions: e ~ (0, sigma^2*I)
We can choose lambda according to the GCV, which is 0.0057. GCV stands for generalized cross-validation, and is used to automate selecting a value for lambda. In this plot, we are looking at how the value of the estimates/coefficients changes as lambda increases from 0.00 to 0.10, where the coefficients/estimates are represented by the different dotted and dashed lines. If we take the lambda value generated from the GCV for example, and place a vertical line at 0.0057, we could use that line to provide us the parameter estimates that would have been chosen if lambda was chosen by this generalized cross validation method. As lambda increases, the coefficent values, or B^ridge values get closer to 0, but as lambda gets close to 0, the coefficient values get closer to their Ordinary Least Squares values.

3. Read the help file for function ksmooth, and run the example at the end of that help file. Write down the model being fitted and assumptions.

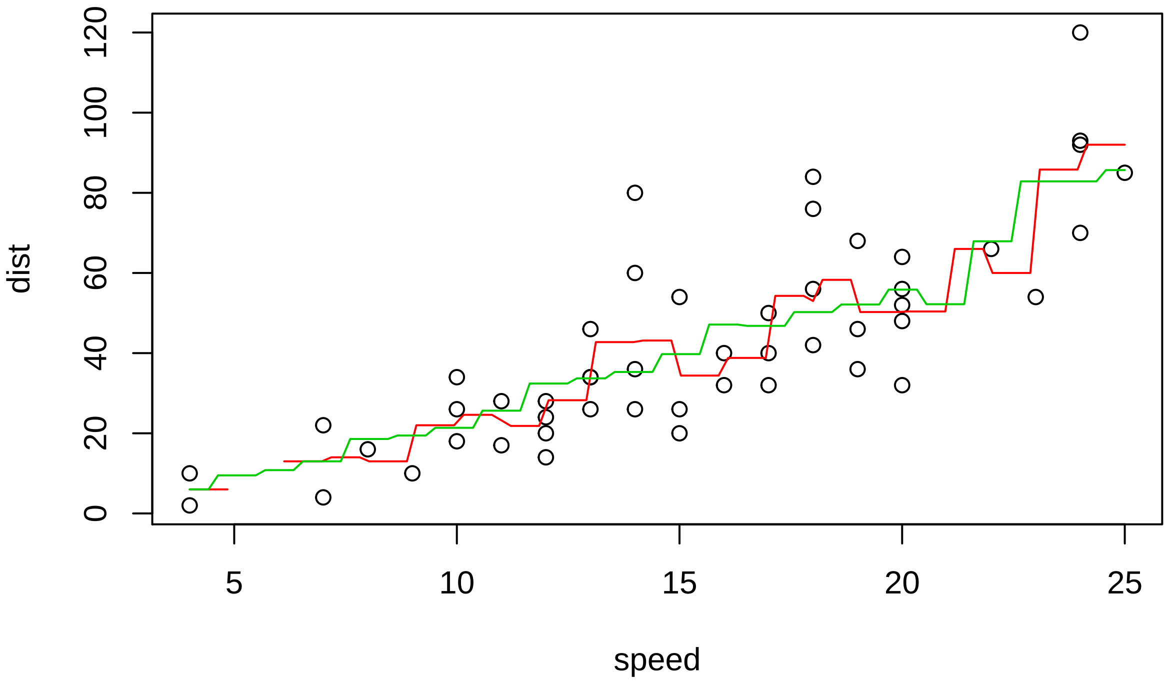
```r
require(graphics)

with(cars, {
    #Plot the data
    plot(speed, dist)
    #Look at a variety of choices for bandwidth
    lines(ksmooth(speed, dist, "normal", bandwidth = 2), col = 2) #The red line
    lines(ksmooth(speed, dist, "normal", bandwidth = 5), col = 3) #The green line
})
```



The model: dist = Bo + B1(speed)
Assumptions: e ~ N(0, sigma^2*I)
Our red line represents a normal kernel with a bandwidth of 2, and our green line represnts a normal kernel with a bandwidth of 5. Since we do not know the true function relating speed and distance, we can only speculate, but it is reasonable to expect this function to be smooth. Since Faraway recommends to pick the least smooth fit that does not show any implausible fluctuations, we can determine that the red normal smoothing kernel seems best.

Rerun this example, but this time use the "box" kernel instead of the "normal" kernel. Submit the resulting figure, and the code you ran to obtain this.

```
with(cars, {
    plot(speed, dist)
    lines(ksmooth(speed, dist, "box", bandwidth = 2), col = 2)
    lines(ksmooth(speed, dist, "box", bandwidth = 5), col = 3)
})
```



The resulting figure is not very smooth, shown through the stepped-looking fit in both the red and the green box smoothing kernels. We can assume that this is not a choice of kernel that we are interested in because, as stated earlier, the relationship between speed and distance is expected to be fairly smooth.

Which of these two kernels do you prefer for these data? Explain why.

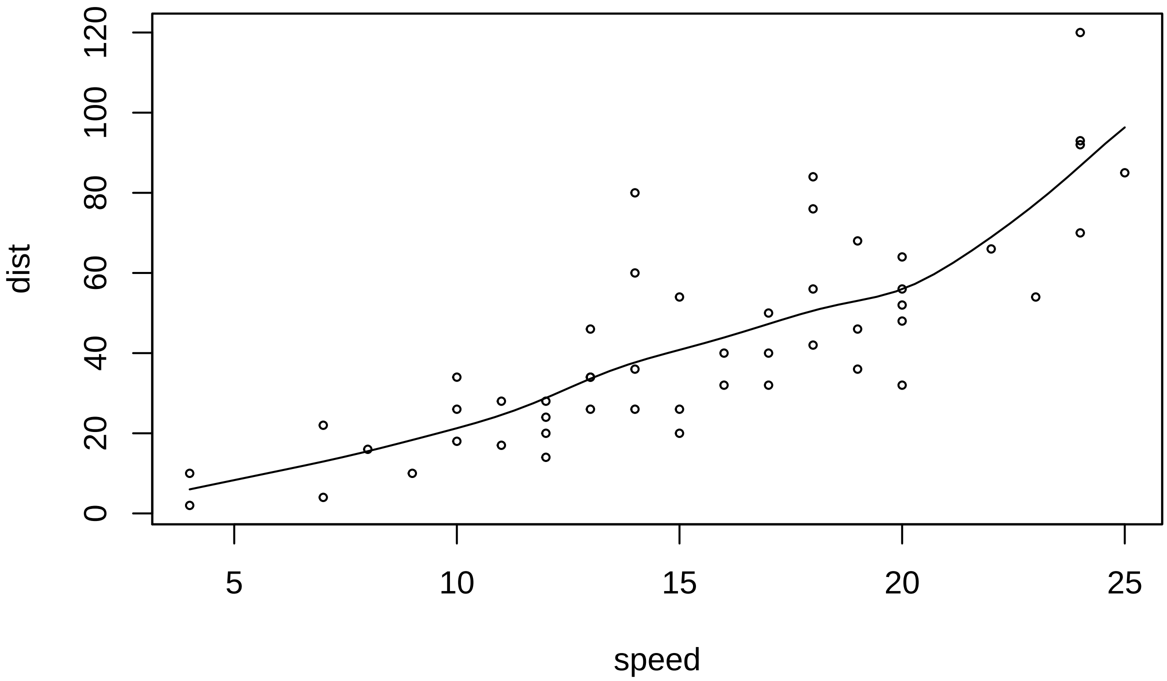
For the choice in kernels, smoothness and compactness are desirable, to ensure that the resulting estimator is smooth and that only data local to the point at which f is estimated is used in the fit. We also want to avoid a stepped-looking fit. Because of this, we prefer the normal kernel on this data set.

4. Again fit a kernel smoother for the "cars" data set used in the previous exercise, but now use function sm.regression in R library "sm", with cross-validation choice of smoothing parameter. Submit your plot, and the code you ran with comments.

```r
library(sm)
```

```
## Package 'sm', version 2.2-5.4: type help(sm) for summary information

##
## Attaching package: 'sm'

## The following object is masked from 'package:MASS':
##
##     muscle
```

```r
with(cars, sm.regression(speed, dist, h=h.select(speed, dist)))
```



The sm library allows us to compute the cross-validated choice of smoothing parameter. When we perform regression with "sm", we find the CV choice of the smoothing parameter using a Gaussian kernel where the smoothing parameter is the standard deviation of the kernel. Here, we can see the kernel estimated smooth of the cars data.

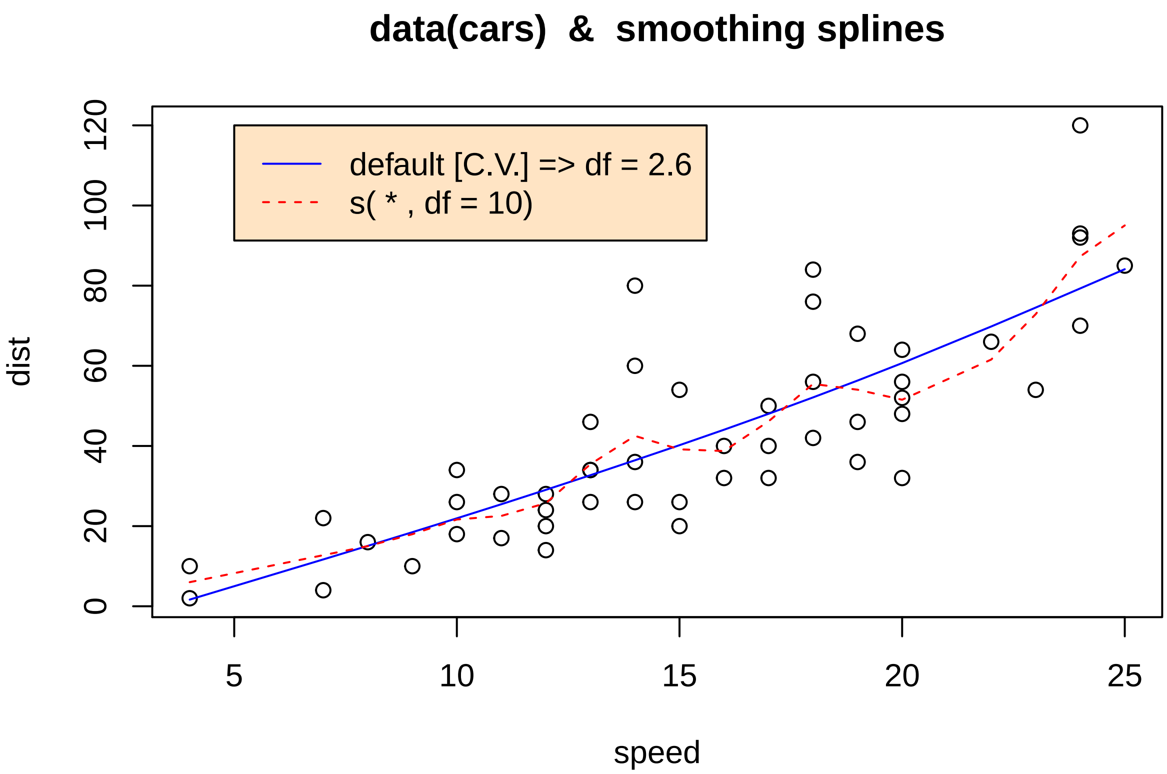
5. Now run the first of the cubic smoothing spline examples at the end of the R help file for function smooth.spline (with data set cars). Also run the code to produce a residual plot for this cars smoothing spline fit. Hand in your code with comments, and figures with explanation.

```
require(graphics)

attach(cars)
plot(speed, dist, main = "data(cars)  &  smoothing splines") #Regular plot of our x vs. y
cars.spl <- smooth.spline(speed, dist)
#Using default - cross-validation - to select the smoothing parameter
(cars.spl)
```

```
## Call:
## smooth.spline(x = speed, y = dist)
##
## Smoothing Parameter  spar= 0.7801305  lambda= 0.1112206 (11 iterations)
## Equivalent Degrees of Freedom (Df): 2.635278
## Penalized Criterion: 4187.776
## GCV: 244.1044
```

```
## This example has duplicate points, so avoid cv = TRUE

lines(cars.spl, col = "blue")
#Default CV is used to select smoothing parameter (using variable created above)
lines(smooth.spline(speed, dist, df = 10), lty = 2, col = "red")
```

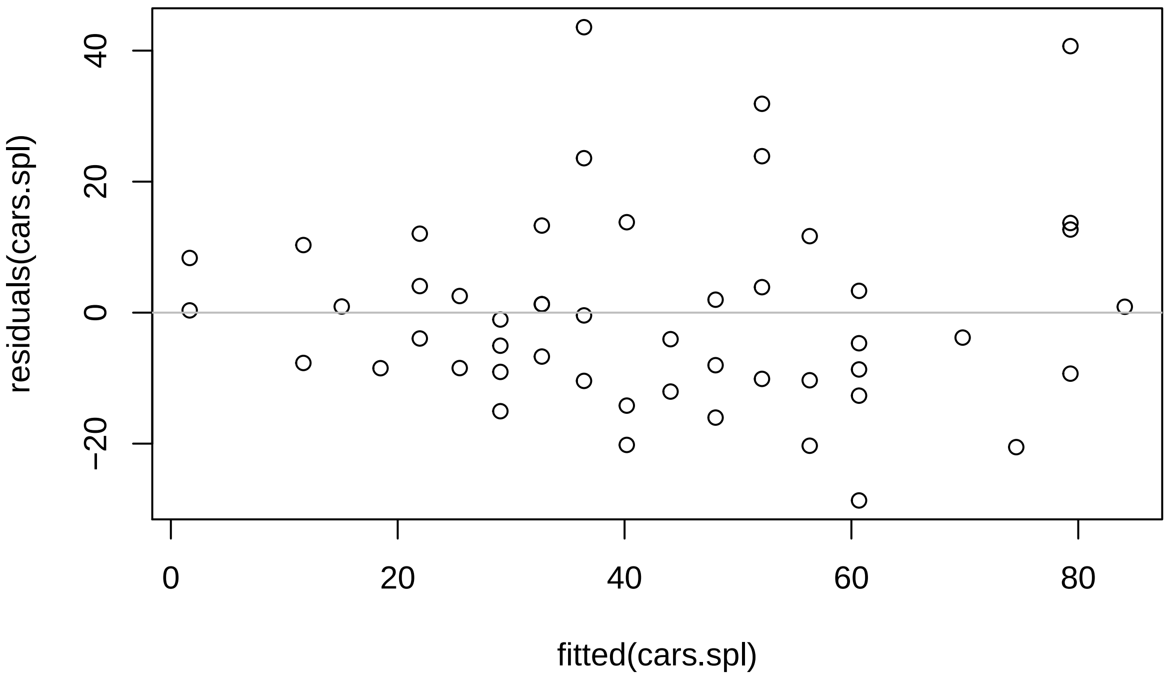
```r
#Setting specific degrees of freedom for smoothing spline fit,
#greater than what CV recommended
legend(5,120,c(paste("default [C.V.] => df =",round(cars.spl$df,1)),
               "s( * , df = 10)"), col = c("blue","red"), lty = 1:2,
       bg = 'bisque')
```




```r
detach()
```

In the first figure above, the blue line represents a smoothing spline using the degrees of freedom recommended by the cross-validation technique, which is 2.6, and creates a fit that is quite smooth. The red line represents a smoothing spline using 10 degrees of freedom, and creates a fit that is a little bit more rough. This makes sense because, like we've seen in class, as the degrees of freedom of a ridge regression model with penalty parameter lambda increases, the model gets more complex, and is directed towards an OLS fit.

```r
## Residual (Tukey Anscombe) plot:
plot(residuals(cars.spl) ~ fitted(cars.spl))
#Plotting residuals vs fitted values from the smoothing spline fit we created
abline(h = 0, col = "gray") #Setting a line at the mean
```



In the second figure, we use the residuals vs the fitted values to check the assumptions that the the mean of the random errors is 0 with a constant variance. We can see that the points are centered around 0, and the variance looks relatively constant, so we can assume that the random errors follow a normal disribution.