Spline Regression on Ozone Data

Hannah Pieper

3/30/21

Introduction

This document performs regression using B-splines, natural splines and smoothing splines on the ozone data found at https://web.stanford.edu/~hastie/ElemStatLearn/data.html. This data consists of observations containing four features; ozone, radiation, temperature and wind. In particular, we perform regression with B-splines, natural splines, and smoothing splines.

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

First we load the data.

```
ozone.data<-read.table("ozone_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(ozone.data)
```

```
##   ozone radiation temperature wind
## 1    41       190          67  7.4
## 2    36       118          72  8.0
## 3    12       149          74 12.6
## 4    18       313          62 11.5
## 5    23       299          65  8.6
## 6    19        99          59 13.8
```

We will perform regression with regular splines, natural splines and smoothing splines with the same degree of freedom. This can be set here:

```
degf<-6
```

Functions and Documentation

Any functions that are written specifically for this document are included here. These functions build the models and plot them and the bulk of the functions deals with formatting these plots.

b.or.n.partial()

Description: This function performs linear regression on one feature with either regression splines or natural splines, plots the training data and the predicted values, and returns the linear model.

Inputs:
column - the feature on which we perform regression. Must be entered as a string and should be one of "temperature", "radiation", or "wind".
degf - the degrees of freedom.
type - enter "bs" for regression with regular splines and "ns" for regression with natural splines.

Output: the linear model.

```
b.or.n.partial<-function(column, degf, type){

  # First we extract the feature data.
  feature<-ozone.data[[column]]
  # Construct a grid of values on which we use the model to predict ozone levels
  collims<-range(feature)
  col_grid<-seq(min(collims), max(collims))

  # This block performs regression with regression splines
  if(type == "bs"){
    if(column=="radiation"){
        # This builds the linear model
        model = lm(ozone~bs(radiation, df=degf), data = ozone.data)
        # Use the linear model to predict ozone levels on the grid
        pred<- predict(model, newdata = list(radiation = col_grid), se = TRUE)
    }
    else if(column == "temperature"){
          model = lm(ozone~bs(temperature, df=degf), data = ozone.data)
          pred<- predict(model, newdata = list(temperature = col_grid), se = TRUE)
    }
    else if(column == "wind"){
          model = lm(ozone~bs(wind, df=degf), data = ozone.data)
          pred<- predict(model, newdata = list(wind = col_grid), se = TRUE)
    }
    else{
      print("Incorrect Feature Input")
    }
  }

  # This block performs regression with natural splines.
  else if(type == "ns"){
    if(column=="radiation"){
        model = lm(ozone~ns(radiation, df=degf), data = ozone.data)
        pred<- predict(model, newdata = list(radiation = col_grid), se = TRUE)
    }
    else if(column == "temperature"){
          model = lm(ozone~ns(temperature, df=degf), data = ozone.data)
          pred<- predict(model, newdata = list(temperature = col_grid), se = TRUE)
    }
    else if(column == "wind"){
          model = lm(ozone~ns(wind, df=degf), data = ozone.data)
          pred<- predict(model, newdata = list(wind = col_grid), se = TRUE)
    }
    else{
      print("Incorrect Feature Input")
    }
  }
  else{
      print("Incorrect Spline Input")
  }

  # We construct the error bands as 2*(standard error) from our predictions
```

```
  # using our model
  error_bands<-with(pred, cbind("upper"=fit+2*se.fit, "lower" = fit - 2*se.fit))

  # Create the labels and titles for our plot
  label1<-"SE bands computed from predictions"
  label2<-"Predictions by model"
  Key<-" Training Data"

  if(type=="ns"){
    name<-"Natural Splines on Ozone Data"
  }
  else{
    name<-"Regression Splines on Ozone Data"
  }

  # Construct the plot
  p<-ggplot() +
    geom_point(data = ozone.data, aes(x = feature, y = ozone, color=Key)) +
    geom_line(aes(x = col_grid, y = pred$fit, color=label2)) +
    geom_ribbon(aes(x = col_grid,
      ymin = error_bands[,"lower"],
      ymax = error_bands[,"upper"],
      color=label1),
      alpha = 0.3) +
    xlim(collims) +
    theme(legend.position = 'right') +
    guides(fill=guide_legend(title="New Legend Title"))+
    labs(title="name", x=column, y="ozone")

  # display the plot
  print(p)
  # Return the model
  return(model)
  }
```

ss.partial()

Description: This function performs regression on one feature with smoothing splines, plots the training data and the predicted values, and returns the model. This function builds two models using different smoothing parameters λ. In the first, λ is determined from the specified degrees of freedom and in the second λ is determined via cross validation.

Inputs:
column - the feature on which we perform regression. Must be entered as a string and should be one of "temperature", "radiation", or "wind".
degf - the degrees of freedom.

Output: the model.

```
ss.partial<-function(column, degf){
    # Extract the data for the feature we are interested in
    feature<-ozone.data[[column]]

    # Construct a model using smoothing splines where the smoothing parameter
    # lambda is determined from the degrees of freedom
```

```
    fit_smooth = with(ozone.data, smooth.spline(feature, ozone, df = degf))
    # Construct a model with optimal \lambda determined from CV
    fit_smooth_cv = with(ozone.data, smooth.spline(feature, ozone, cv = TRUE))

    # Extract the effective degrees of freedom
    edf<-round(fit_smooth_cv$df,2)

    # Build the labels
    Key<-paste(degf," degrees of freedom")
    label2<-paste(edf, "effective degrees of freedom")

    # Plot the smoothing splines
    p<- ggplot() +
        geom_point(data = ozone.data, aes(x = feature, y = ozone)) +
        geom_line(aes(x = fit_smooth$x, y = fit_smooth$y,
                                              color = Key))  +
        geom_line(aes(x = fit_smooth_cv$x, y = fit_smooth_cv$y,
                                                color = label2)) +
        theme(legend.position = 'right') +
        labs(title = "Smoothing Splines on Ozone Data", x=column, y="ozone")
  # Print the plot
  print(p)
  # Return the model with \lambda determined via CV
  return(fit_smooth_cv)
}
```

Regression Splines

Recall that for fixed M, a basis for the space of regression splines in one dimension is given by

$$\begin{aligned} h_j(x) &= x^{j-1}, \qquad j = 1, \ldots, M \\ h_{M+1} &= (x - \xi_\ell)_+^{M-1}, \qquad \ell = 1, \ldots, K, \end{aligned}$$

where ξ_i with $i \in [1, \ldots, K]$ are the fixed knots. This basis has $K + M$ terms; meaning in a spline space of order M with K knots, we have $M + K$ degrees of freedom to specify our approximation functions. With this basis, we then perform ordinary linear regression to determine $\hat{\beta} = \hat{\beta}_0, \ldots, \hat{\beta}_{M+K}$ to form the model

$$\hat{f}(x) = \sum_{k=1}^{M+K} \beta_k h_k(x).$$

In R, regular regression splines of order M with fixed knots are formed via the function bs() with the default being $M = 4$. The knots can be specified; if they are not, the knots are chosen by default at the K quantiles of the data. Furthermore, if there are N training points x_i, this function bs() returns a $N \times (M + K - 1)$ basis matrix of the form

$$\begin{aligned} X &= \begin{pmatrix} h_2(x_1) & \ldots & h_M(x_1) & h_{M+1}(x_1) & \ldots & h_{M+K}(x_1) \\ \vdots & & \vdots & \vdots & & \vdots \\ h_2(x_N) & \ldots & h_M(x_N) & h_{M+1}(x_N) & \ldots & h_{M+K}(x_N) \end{pmatrix} \\ &= \begin{pmatrix} x_1 & \ldots & x_1^{M-1} & (x_1 - \xi_1)_+^{M-1} & \ldots & (x_1 - \xi_K)_+^{M-1} \\ \vdots & & \vdots & \vdots & & \vdots \\ x_N & \ldots & x_N^{M-1} & (x_N - \xi_1)_+^{M-1} & \ldots & (x_N - \xi_K)_+^{M-1} \end{pmatrix}. \end{aligned}$$

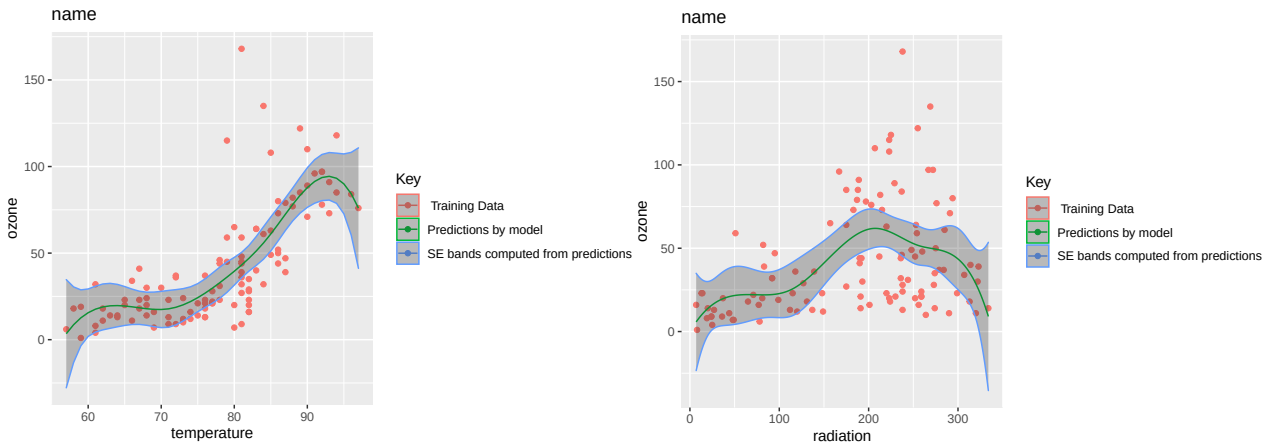
It is important to notice that this basis matrix excludes the constant basis element. Therefore, the function bs() has $M + K - 1$ degrees of freedom. After we form the basis matrix, we perform ordinary linear regression using the lm() function. It is important to note that the call of this function automatically includes the constant basis element to form the intercept.

Regression on 1 Feature

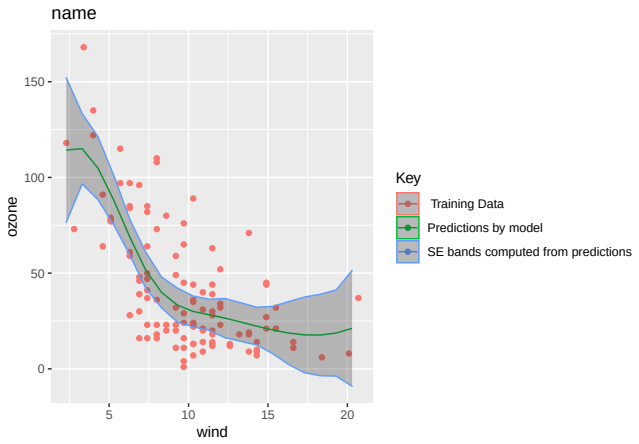
In this section, we work with cubic splines. Therefore if we specify 6 degrees of freedom, this means that 3 interior knots will be chosen; they are fixed at the median and the first and third quartile of the training data.

Here we produce the figures similar to those in Figure 5.4 in the text. These display fitted B-spline functions and included are the point-wise standard error bounds.

```
bs.temp<-b.or.n.partial("temperature", degf,"bs")
bs.rad<-b.or.n.partial("radiation", degf,"bs")
```




```
bs.wind<-b.or.n.partial("wind", degf,"bs")
```



These plots are made as follows:

1. First, a basis of cubic splines with three interior knots at the quartiles of the feature data is generated;
2. we perform linear regression on this basis;
3. this model is used to predict ozone data from the feature;
4. the standard error of these predictions is computed and forms the error bars.

```
t<-toString(attr(bs(ozone.data[["temperature"]],df=degf),"knots"))
r<- toString(attr(bs(ozone.data[["radiation"]],df=degf),"knots"))
```

```
w<- toString(attr(bs(ozone.data[["wind"]],df=degf),"knots"))

sep="\n"
string<- paste("The x values of the knots are fixed at the values: ", sep, "Temperature: ", t, sep,"Radi
cat(string)
```

```
## The x values of the knots are fixed at the values:
##  Temperature:  71, 79, 84.5
##  Radiation:  113.5, 207, 255.5
##  Wind:  7.4, 9.7, 11.5
```

```
t<- round(sum(abs(bs.temp$residual)^2), 2)
r<- round(sum(abs(bs.rad$residual)^2), 2)
w<- round(sum(abs(bs.wind$residual)^2), 2)

sep="\n"
string<- paste("The residual sum squares for each feature are as follow: ", sep, "Temperature: ", t, sep
cat(string)
```

```
## The residual sum squares for each feature are as follow:
##  Temperature:  51716.87
##  Radiation:  90119
##  Wind:  56979.72
```

Multiple Regression

Now we want to predict the ozone levels using all of the features. In this case, we need to construct a large linear regression model using bases for cubic splines associated to each feature. This can be done using the lm() function as well. Now each of the features form a coordinate of our model.

```
bsmodel<- lm(ozone ~  bs(radiation, degf)+bs(temperature, degf)+bs(wind, degf), data=ozone.data)
```

The model summary is as follows:

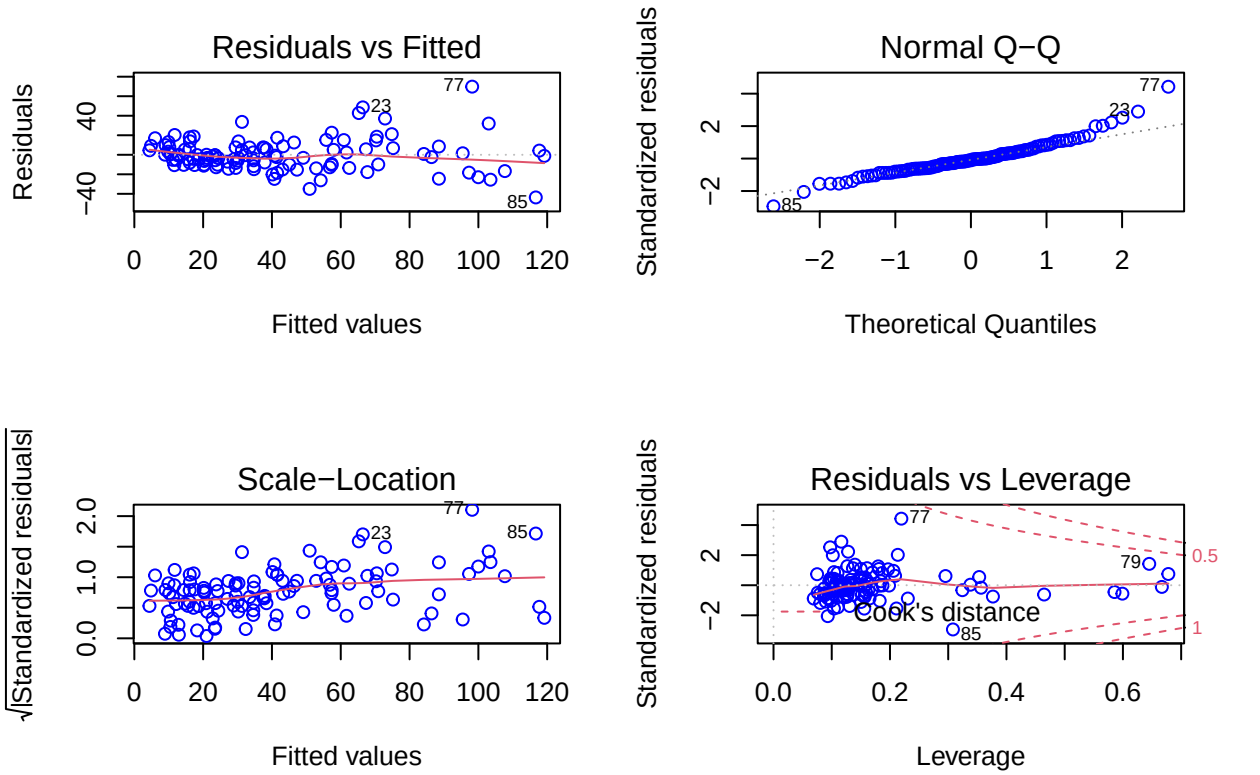
```
tidy(summary(bsmodel))
```

```
## # A tibble: 19 x 5
##    term                    estimate std.error statistic p.value
##    <chr>                      <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)               60.6       23.3    2.60   0.0108
##  2 bs(radiation, degf)1       8.64      20.3    0.426  0.671
##  3 bs(radiation, degf)2      -1.84      14.7   -0.126  0.900
##  4 bs(radiation, degf)3      19.4       14.4    1.34   0.183
##  5 bs(radiation, degf)4      26.0       12.6    2.05   0.0429
##  6 bs(radiation, degf)5       4.87      18.3    0.266  0.791
##  7 bs(radiation, degf)6      13.4       16.6    0.811  0.420
##  8 bs(temperature, degf)1     1.24      26.7    0.0466 0.963
##  9 bs(temperature, degf)2    -5.57      15.9   -0.350  0.727
## 10 bs(temperature, degf)3    -1.72      17.2   -0.100  0.920
## 11 bs(temperature, degf)4    40.9       17.6    2.33   0.0222
## 12 bs(temperature, degf)5    41.9       23.3    1.80   0.0750
## 13 bs(temperature, degf)6    30.5       20.6    1.48   0.142
## 14 bs(wind, degf)1           14.1       27.1    0.520  0.605
## 15 bs(wind, degf)2          -47.0       16.1   -2.92   0.00444
## 16 bs(wind, degf)3          -48.7       17.5   -2.79   0.00636
## 17 bs(wind, degf)4          -57.4       20.4   -2.81   0.00611
```

```
## 18 bs(wind, degf)5          -64.0       28.7   -2.23   0.0282
## 19 bs(wind, degf)6          -43.1       21.6   -2.00   0.0490
```

The residual plots for the full model using regression splines are as follow:

```
par(mfrow = c(2,2))
plot(bsmodel, col="blue")
```



We can also use ANOVA testing to determine whether all three features are necessary to make a good prediction. For example, if we wanted to understand the effect that adding the wind data has on our model, we can compare two models; one that just uses the radiation and temperature data and another that uses all three features.

```
submodel1<-lm(ozone~bs(radiation, degf) + bs(temperature, degf), data=ozone.data)
submodel2<-lm(ozone~bs(radiation, degf) + bs(wind, degf), data=ozone.data)
submodel3<-lm(ozone~bs(wind, degf) + bs(temperature, degf), data=ozone.data)

tidy(anova(bsmodel, submodel1))
```

```
## # A tibble: 2 x 6
##   res.df    rss    df   sumsq statistic      p.value
##    <dbl>  <dbl> <dbl>   <dbl>     <dbl>        <dbl>
## 1     92 29340.    NA      NA       NA    NA
## 2     98 45103.    -6 -15763.      8.24  0.000000383
```

```
tidy(anova(bsmodel, submodel2))
```

```
## # A tibble: 2 x 6
##   res.df    rss    df   sumsq statistic      p.value
##    <dbl>  <dbl> <dbl>   <dbl>     <dbl>        <dbl>
## 1     92 29340.    NA      NA       NA    NA
## 2     98 45760.    -6 -16420.      8.58  0.000000207
```

```
tidy(anova(bsmodel, submodel3))
```

```
## # A tibble: 2 x 6
##   res.df    rss    df  sumsq statistic p.value
##    <dbl>  <dbl> <dbl>  <dbl>     <dbl>   <dbl>
## 1     92 29340.    NA    NA       NA    NA
## 2     98 33874.    -6 -4534.      2.37  0.0357
```

Because the p-value is the largest when adding in radiation, this suggests that radiation is in some sense the least useful feature in determining ozone levels.

Natural Spline Regression

Recall that natural splines are just like regression splines, only now they are required to be linear at the end intervals $[a, \xi_1]$ and $[\xi_K, b]$. A basis for the set of all natural cubic splines on K knots in one dimension is given by

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

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 $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_K)$ to construct the model

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

In R, the function ns() constructs a matrix of these basis functions evaluated at the training data x_N. More specifically, this matrix is of the form

$$X = \begin{pmatrix} N_2(x_1) & N_3(x_1) & \ldots & N_K(x_1) \\ N_2(x_2) & N_3(x_2) & \ldots & N_K(x_2) \\ \vdots & \vdots & & \vdots \\ N_2(x_N) & N_3(x_N) & \ldots & N_K(x_N) \end{pmatrix},$$

where as in the case of regression splines, the constant basis function $N_1(x)$ is excluded. The exclusion of the constant term is then accounted for in the function lm(), which performs ordinary regression on this data matrix.

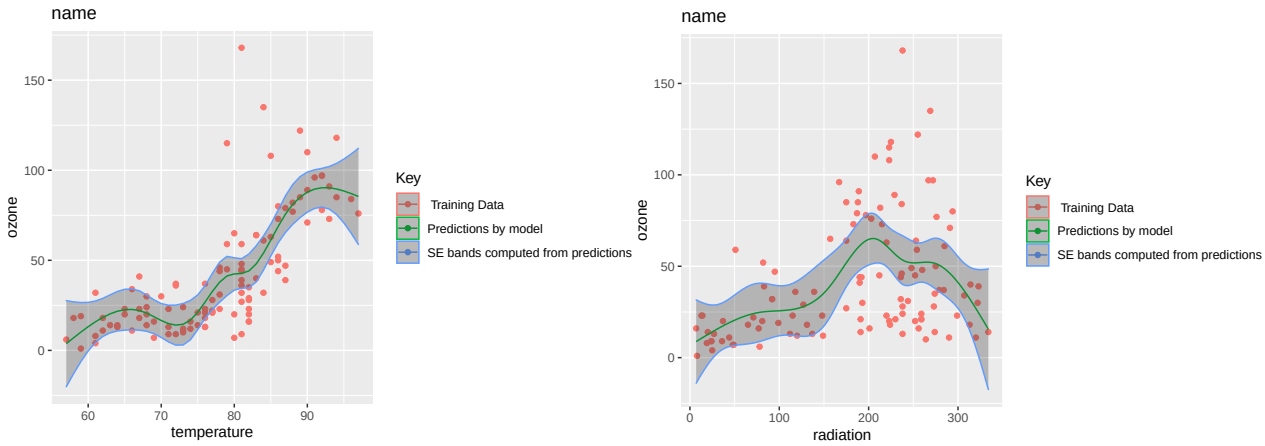
In general, natural cubic splines have $M + K + 4$ degrees of freedom. This is because the requirement that the function is linear beyond the two boundary knots frees up four degrees of freedom, which can be used as interior knots. Therefore, if we fix the degrees of freedom, the model using natural cubic splines will have more interior knots than the model using regression splines.

Regression on 1 Feature

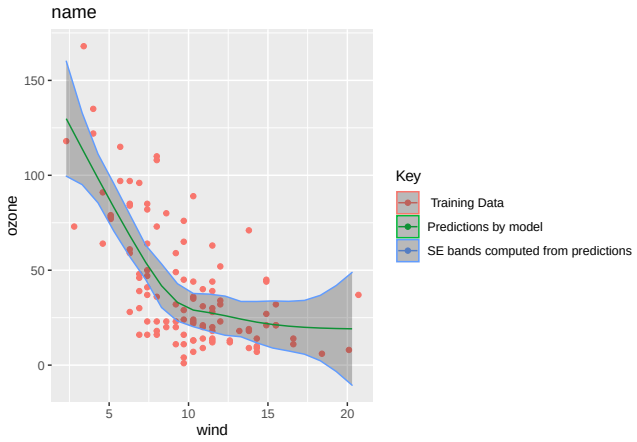
In this section, we use cubic splines $M = 4$ and we want the same number of degrees of freedom as in the case of regression splines. Note that since the constant basis element is excluded from these calculations using the function ns(), we will only have two more interior knots than we did with regression splines.

First, we perform linear regression with natural cubic splines on each feature individually.

```
ns.temp<-b.or.n.partial("temperature", degf,"ns")
ns.rad<-b.or.n.partial("radiation", degf,"ns")
```




```
ns.wind<-b.or.n.partial("wind", degf,"ns")
```




```
t<-toString(attr(ns(ozone.data[["temperature"]],df=degf),"knots"))
r<- toString(attr(ns(ozone.data[["radiation"]],df=degf),"knots"))
w<- toString(attr(ns(ozone.data[["wind"]],df=degf),"knots"))

sep="\n"
string<- paste("The x values of the knots for these models are at the following values: ", sep, "Tempera
cat(string)
```

```
## The x values of the knots for these models are at the following values:
##  Temperature:  67.3333333333333, 74, 79, 82, 87
##  Radiation:  77.3333333333333, 154.333333333333, 207, 238, 272.666666666667
##  Wind:  6.9, 8, 9.7, 11.1, 13.8
```

```
t<- round(sum(abs(ns.temp$residual)^2), 2)
r<- round(sum(abs(ns.rad$residual)^2), 2)
w<- round(sum(abs(ns.wind$residual)^2), 2)

sep="\n"
string<- paste("The residual sum squares for each feature are as follow: ", sep,"Temperature: ", t, sep
cat(string)
```

```
## The residual sum squares for each feature are as follow:
```

```
##  Temperature:  50807.24
##  Radiation:  89594.26
##  Wind:  57887.48
```

The process for making these plots is virtually identical to the one used in the regression spline case, only now we use ns() instead of bs().

Multiple Regression with Natural Splines

Now we want to predict the ozone levels using all of the features. Here, we are just creating a large linear regression model with an appropriate choice of bases where each feature forms a coordinate of the model.

```
nsmodel<- lm(ozone ~ ns(radiation, degf)+ns(temperature, degf)+ns(wind, degf), data=ozone.data)
```

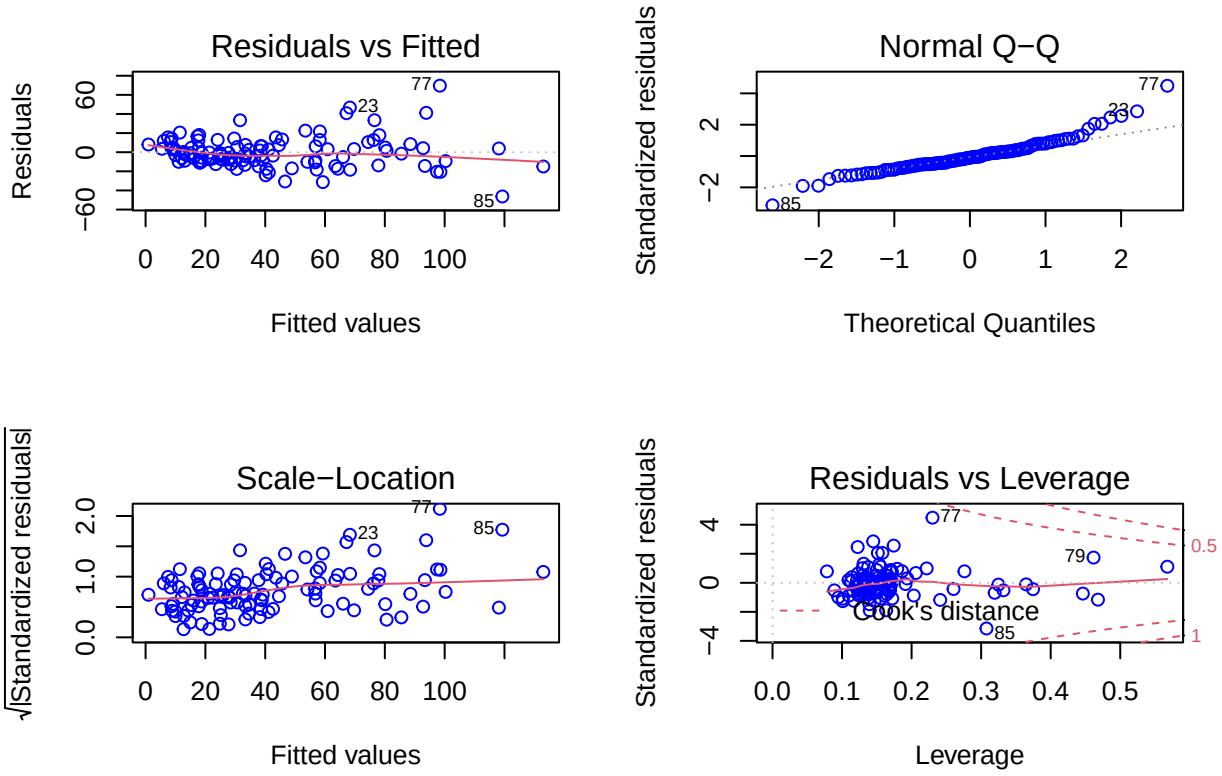
Here is the model summary:

```
tidy(summary(nsmodel))
```

```
## # A tibble: 19 x 5
##    term                    estimate std.error statistic    p.value
##    <chr>                      <dbl>     <dbl>     <dbl>      <dbl>
##  1 (Intercept)                72.3       17.8     4.05  0.000106
##  2 ns(radiation, degf)1        7.52      12.5     0.601 0.549
##  3 ns(radiation, degf)2       10.7       12.7     0.839 0.404
##  4 ns(radiation, degf)3       28.2       10.2     2.76  0.00701
##  5 ns(radiation, degf)4        4.56      10.4     0.438 0.663
##  6 ns(radiation, degf)5       11.6       20.6     0.562 0.576
##  7 ns(radiation, degf)6       10.6       12.0     0.884 0.379
##  8 ns(temperature, degf)1     -8.66      11.9    -0.729 0.468
##  9 ns(temperature, degf)2     13.8       15.5     0.890 0.376
## 10 ns(temperature, degf)3     15.6       12.7     1.23  0.221
## 11 ns(temperature, degf)4     56.3       12.4     4.53  0.0000178
## 12 ns(temperature, degf)5     48.2       28.8     1.67  0.0976
## 13 ns(temperature, degf)6     25.8       13.3     1.94  0.0550
## 14 ns(wind, degf)1           -59.9       12.4    -4.82  0.00000573
## 15 ns(wind, degf)2           -62.6       15.1    -4.16  0.0000726
## 16 ns(wind, degf)3           -66.8       14.1    -4.73  0.00000820
## 17 ns(wind, degf)4           -59.2       12.1    -4.90  0.00000406
## 18 ns(wind, degf)5           -97.8       29.4    -3.32  0.00129
## 19 ns(wind, degf)6           -36.0       15.0    -2.40  0.0183
```

We can also look at the residual plots:

```
par(mfrow = c(2,2))
plot(nsmodel, col="blue")
```



Smoothing Spline Regression

Recall that smoothing splines incorporate the smoothing parameter λ and minimize the penalized residual sum squares

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

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

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

If $\hat{\mathbf{f}}$ is the N vector of fitted values $\hat{f}(x_i)$, then we can write

$$\begin{aligned}\hat{\mathbf{f}} &= \mathbf{N}\big(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N\big)^{-1}\mathbf{N}^T\mathbf{y} \\ &:= S_\lambda y.\end{aligned}$$

Recalling that the degrees of freedom for a linear regression model $\hat{f}(x) = Hy$ can be computed as $\mathrm{Tr}(H)$, we define the effective degrees of freedom of a smoothing spline to be

$$df_\lambda = \mathrm{Tr}\big(S_\lambda\big).$$

Note that in the case of linear models, \$df = Tr(H) will result in the same number of degrees of freedom as obtained in the discussion for regression splines and natural splines.

This is however a nonlinear problem so our linear regression techniques do not apply.

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 the degree of freedom is not specified, or alternatively we set "cv = TRUE", then λ is determined by leave-one-out cross validation (LOOCV).

Regression on 1 Feature

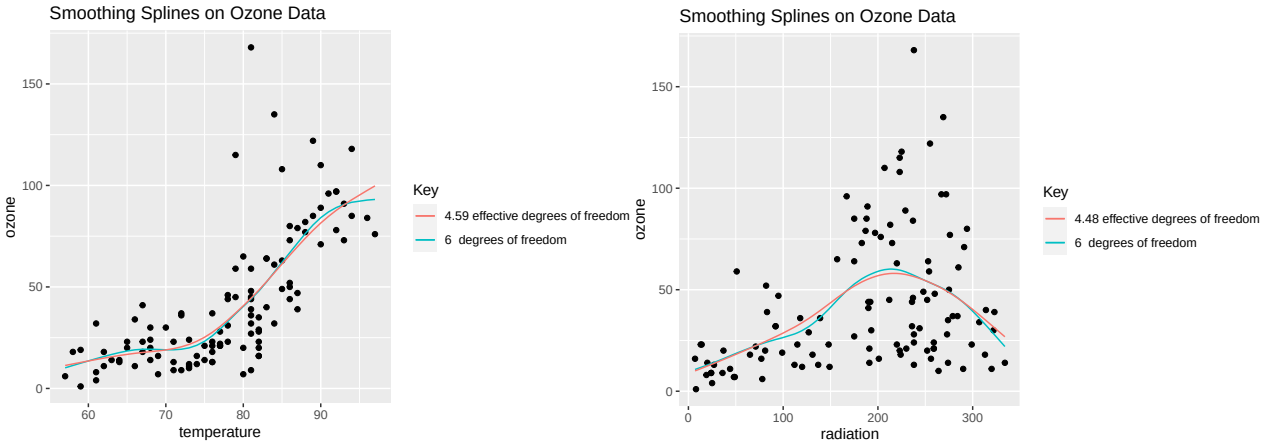
In this section, we perform regression on one feature at a time to predict the ozone levels. We plot the predictor function where λ is determined from the specified degree of freedom and the predictor function where λ is determined via cross-validation. Note that since there are multiple ozone measurements for a particular feature value, the function tells us to be cautious with trusting the λ computed from cross validation.

```
ss.temp<-ss.partial("temperature", degf)
```

```
## Warning in smooth.spline(feature, ozone, cv = TRUE): cross-validation with non-
## unique 'x' values seems doubtful
```

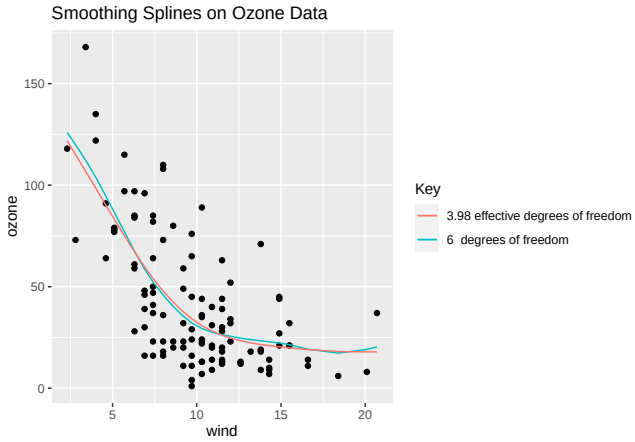
```
ss.rad<-ss.partial("radiation", degf)
```

```
## Warning in smooth.spline(feature, ozone, cv = TRUE): cross-validation with non-
## unique 'x' values seems doubtful
```




```
ss.wind<-ss.partial("wind", degf)
```

```
## Warning in smooth.spline(feature, ozone, cv = TRUE): cross-validation with non-
## unique 'x' values seems doubtful
```




```
t<- round(ss.temp$lambda, 5)
r<- round(ss.rad$lambda, 5)
w<-  round(ss.wind$lambda, 5)

sep="\n"
string<- paste("The optimal smoothing parameters lambda as determined by LOOCV for each feature are as
cat(string)
```

```
## The optimal smoothing parameters lambda as determined by LOOCV for each feature are as follow:
##  Temperature:  0.00942
##  Radiation:  0.01111
##  Wind:  0.01658
```

```
t<- round(ss.temp$pen.crit, 2)
r<- round(ss.rad$pen.crit, 2)
w<- round(ss.wind$pen.crit, 2)

sep="\n"
string<- paste("The residual sum squares for each feature are as follow: ", sep, "Temperature: ", t, sep
cat(string)
```

```
## The residual sum squares for each feature are as follow:
##  Temperature:  15928.29
##  Radiation:  65662.54
##  Wind:  14861.52
```

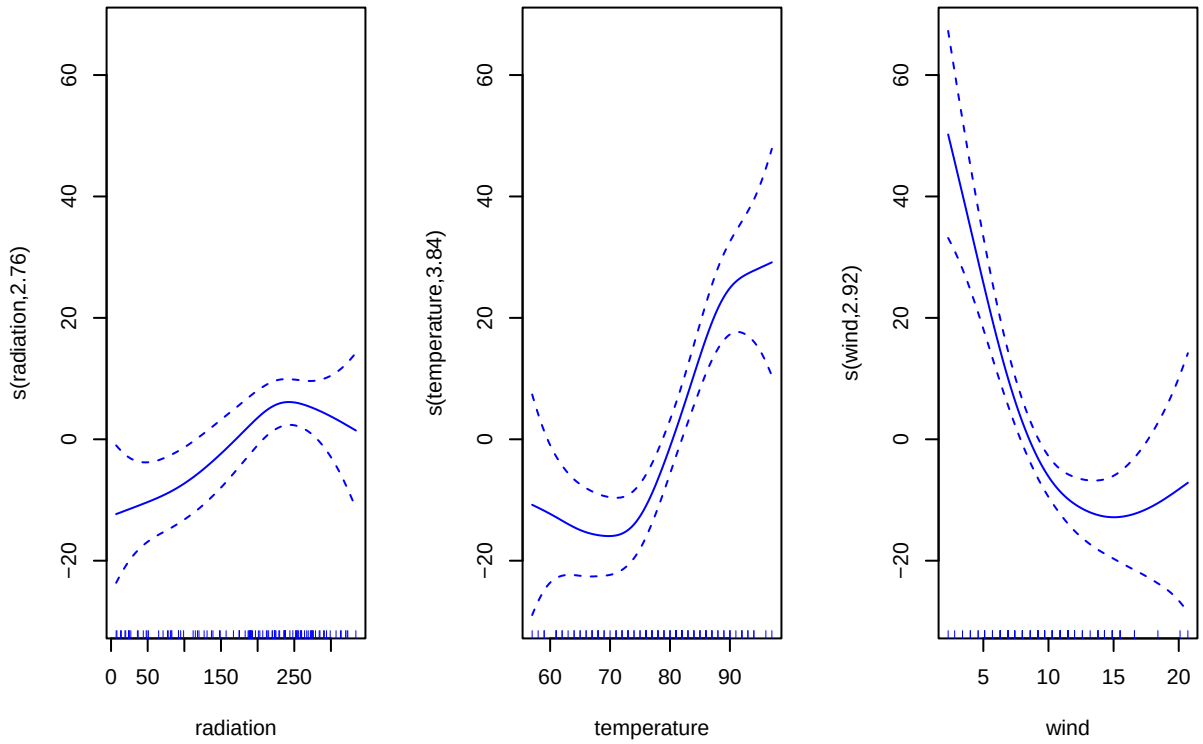
Multiple Regression with Smoothing Splines

A regression model for the ozone using smoothing splines and all three parameters can be constructed using the function gam(). This function creates a generalized additive model using the ozone data. Previously, we used lm() because the components could be expressed in terms of basis functions and then the parameters could be fit using ordinary least squares regression. Since this is not the case for smoothing splines, we must use a more general sort of additive model.

```
ssmodel = gam(ozone ~ s(radiation) + s(temperature) + s(wind), data = ozone.data)
```

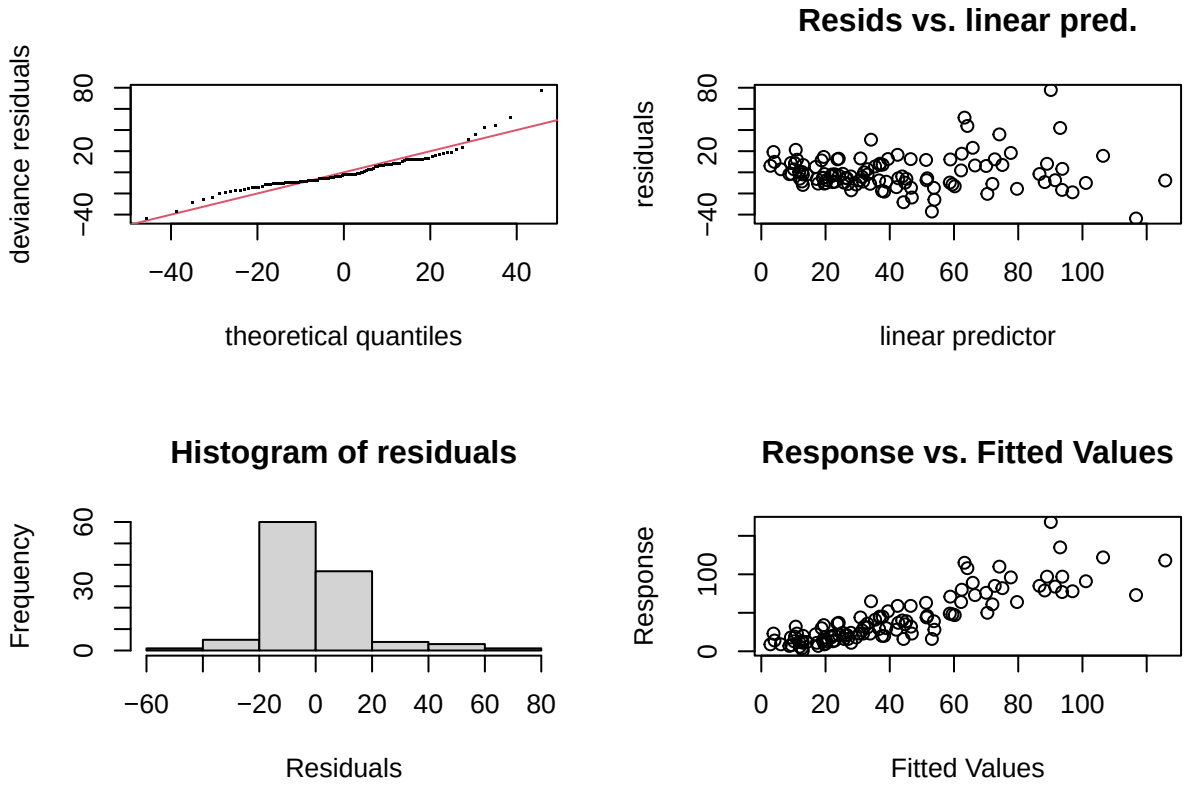
The predictions for each component and the model summary are as follow:

```
par(mfrow = c(1,3))
plot(ssmodel, se = TRUE, col = "blue")
```



We can also look at the residual plots for this model:

```
par(mfrow = c(2,2))
gam.check(ssmodel)
```




```
##
```

```
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 5 iterations.
## The RMS GCV score gradient at convergence was 5.567617e-05 .
## The Hessian was positive definite.
## Model rank =  28 / 28
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                  k'  edf k-index p-value
## s(radiation)   9.00 2.76    0.96    0.29
## s(temperature) 9.00 3.84    0.79    0.02 *
## s(wind)        9.00 2.92    0.96    0.29
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comparison

In this section, we compare the residual sum squares between the ozone predictions by each spline model using all three features and the actual ozone data. This is given by

$$RSS(\hat{f}) = \sum_{i=1}^{N} (\hat{f}(x_i) - y_i)^2.$$

```
reg.rss<-round(sum(abs(bsmodel$residuals)^2),2)
nat.rss<-round(sum(abs(nsmodel$residuals)^2),2)
ss.rss<-round(sum(abs(ssmodel$residuals)^2),2)

sep="\n"
string<- paste("The residual sum squares for each full model are as follow: ", sep, "Regression Splines
cat(string)
```

```
## The residual sum squares for each full model are as follow:
##  Regression Splines:  29340.08
##  Natural Splines:  28778.58
##  Smoothing Splines:  30742.41
```

It makes sense that the model using natural splines has a lower RSS than the model using regression splines since there are more interior knots in the former model. Furthermore, since the model using smoothing splines encourages a smoother $\hat{f}$, it makes sense that this characteristic comes at the expense of increasing the RSS beyond that of the model using natural cubic splines.

In all of these models, the residuals get larger as the predictions increase. This suggests that our model could be improved; perhaps there is a variable missing or we could transform a variable to get a better fit.