Exercise 8

Nikolaus Czernin

```r
library("knitr")
# library("ROCit")
# library("ISLR")
# library("klaR")
# library("glmnet")
# install.packages("gclus")
library("gclus")
```

```
## Loading required package: cluster
```

```r
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
set.seed(11721138)
```

Loading, preprocessing & splittting

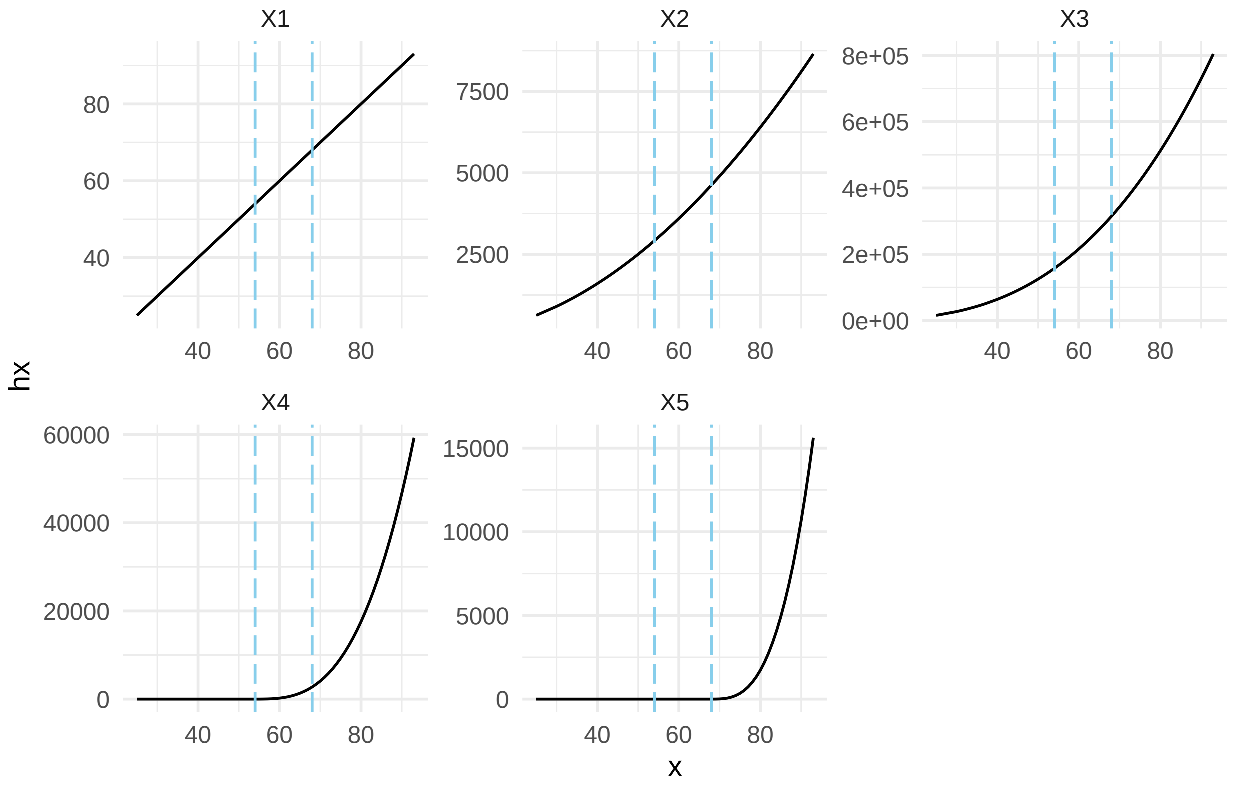
```r
data(ozone)

# perform train set split
train_idx <- sample(1:nrow(ozone), nrow(ozone)%/%3*2, replace=FALSE)
train <- ozone[train_idx, ]
test <-  ozone[-train_idx, ]

# ozone
```

```r
getsplines <- function(x, nknots=2, M=4){
  # nknots ... number of knots -> placed at regular quantiles
  # M ... M-1 is the degree of the polynomial
  n <- length(x)
  # X will not get an intercept column
  X <- matrix(NA, nrow=n, ncol=(M - 1) + nknots)
  for (i in 1:(M - 1)){
    X[,i] <- x^i
    }
  # now the basis functions for the constraints:
  quant <- seq(0, 1, 1 / (nknots + 1))[c(2 : (nknots + 1))]
  qu <- quantile(x,quant)
  for (i in M : (M + nknots - 1)){
    X[,i] <- ifelse(x - qu[i - M + 1] < 0, 0, (x - qu[i - M + 1])^(M - 1))
  }
  list(x=x, X=X, quantiles=quant, xquantiles=qu)
}
```

Task 1: Creating and plotting splines

```r
plotspl <- function(splobj, ..., title=""){
  vertical.lines <- splobj$xquantiles %>% unname()
  splobj %>%
    .$X %>%
    data.frame() %>%
    mutate(x=X1) %>%
    pivot_longer(-c(x), values_to = "hx") %>%
    ggplot(aes(x=x, y=hx)) +
      geom_line() +
      theme_minimal() +
      facet_wrap(~name, scales="free") +
      geom_vline(xintercept=vertical.lines,
                 colour="skyblue",
                 linetype = "longdash") +
      ggtitle(title)
  }

# getsplines(ozone$Temp) %>%
#   plotspl(main="Temp")
```

```r
# These lines create splines for all columns, not just temp
# splines <- train %>%
#   colnames() %>%
#   lapply(function(name){
#     list(name=name, splines=getsplines(train[[name]]))
#   })
#
# splines %>%
#   lapply(function(spline) plotspl(spline$splines, title=spline$name))

# Creating splines for the temperature
splines.temp.train <- getsplines(train$Temp)
splines.temp.test <- getsplines(test$Temp)
plotspl(splines.temp.train, "Temperature Splines")
```



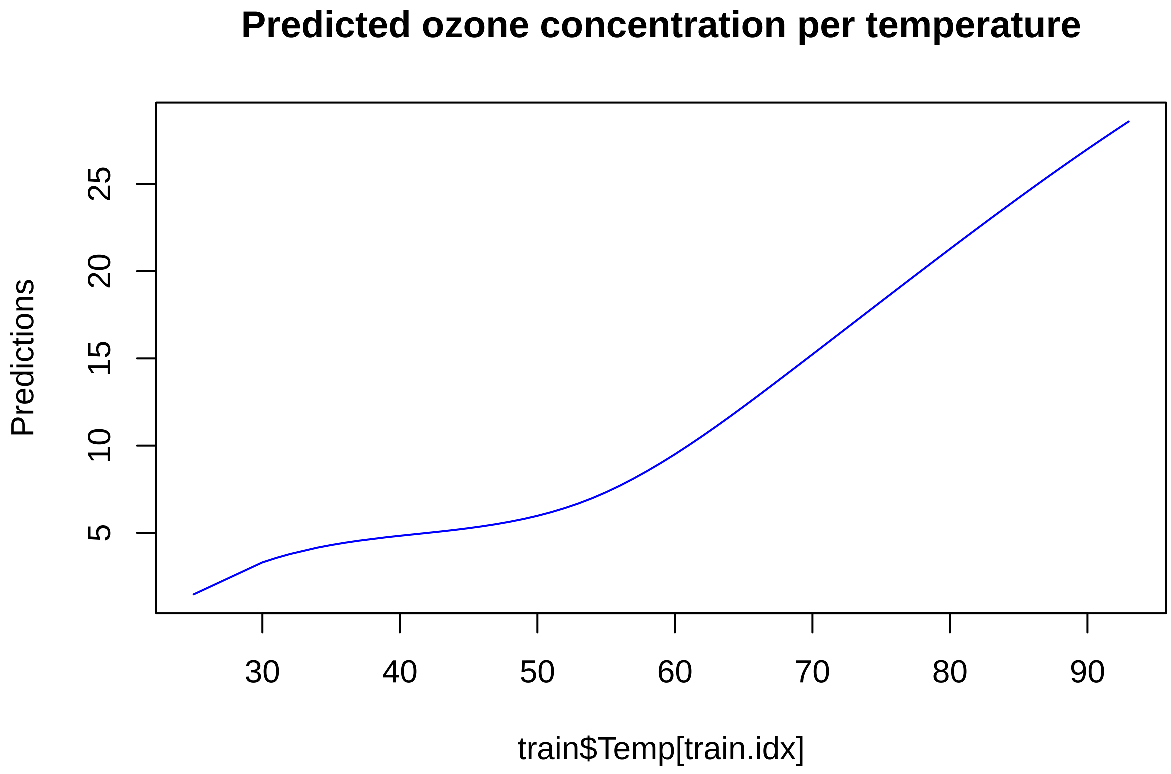
```
# plotspl(splines.temp.test, "Temperature Splines")
```

Task 2: Fitting a linear model

```
# fit the model
training.data <- data.frame(splines.temp.train$X, Ozone=train$Ozone)
model1 <- lm(Ozone ~ ., data=training.data)
summary(model1)
```

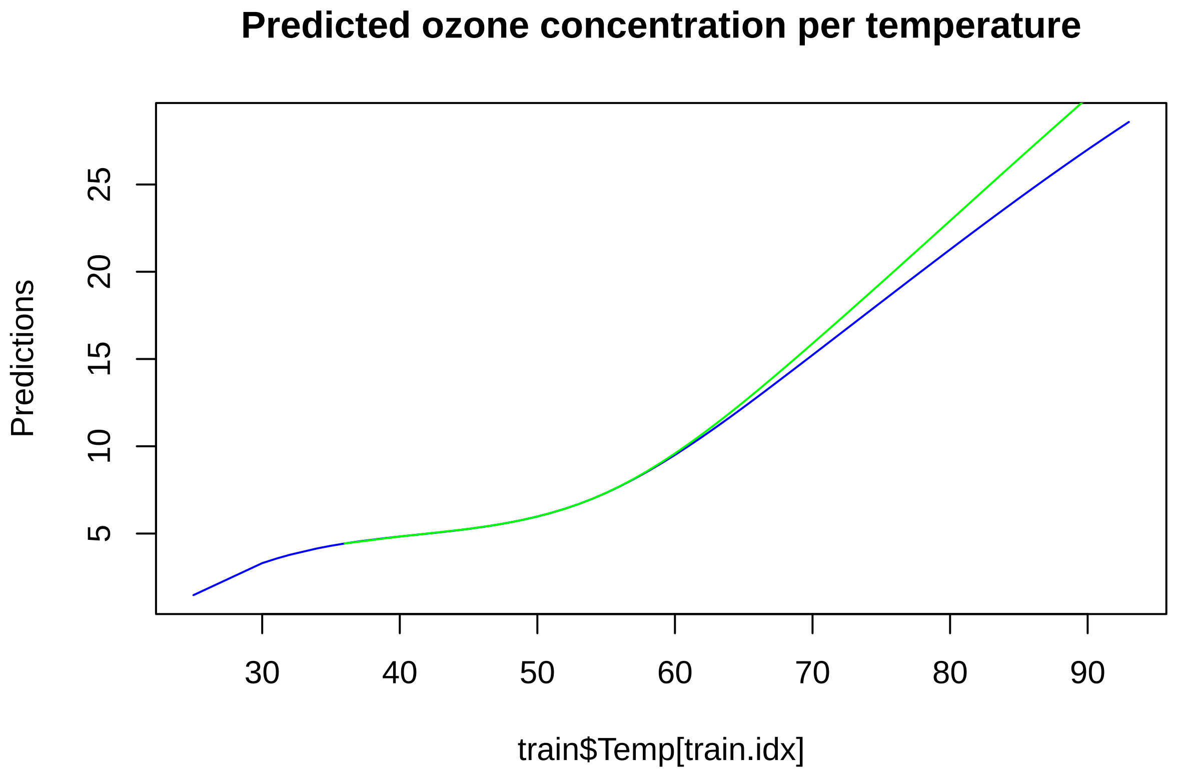
```
##
## Call:
## lm(formula = Ozone ~ ., data = training.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.9982  -2.6875  -0.5925   2.1402  16.3125
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.319e+01  4.344e+01  -0.764    0.446
## X1           2.607e+00  2.916e+00   0.894    0.372
## X2          -6.116e-02  6.359e-02  -0.962    0.337
## X3           4.938e-04  4.517e-04   1.093    0.276
## X4          -9.143e-04  8.977e-04  -1.018    0.310
## X5           3.438e-04  9.614e-04   0.358    0.721
##
## Residual standard error: 4.688 on 214 degrees of freedom
## Multiple R-squared:  0.6822, Adjusted R-squared:  0.6748
## F-statistic: 91.87 on 5 and 214 DF,  p-value: < 2.2e-16
```

```
# make predictions
train.idx <- train$Temp %>% sort(index.return=TRUE) %>% .$ix
test.idx <- test$Temp %>% sort(index.return=TRUE) %>% .$ix
yhat.train <- predict(model1, training.data)

plot(train$Temp[train.idx], yhat.train[train.idx], ylab="Predictions", main="Predicted ozone concentrat
```



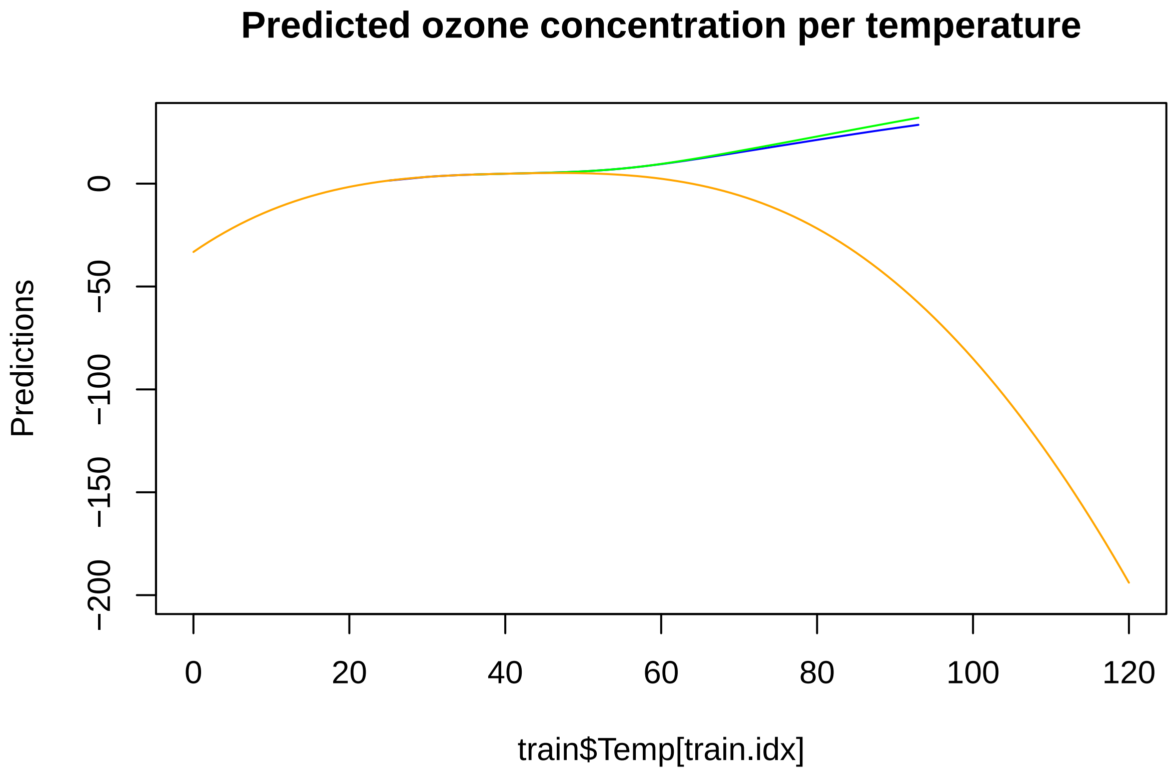
Task 3: Plotting predictions vs observations

```
plot(train$Temp[train.idx], yhat.train[train.idx], ylab="Predictions", main="Predicted ozone concentrat

test.data <- data.frame(splines.temp.test$X, Ozone=test$Ozone)
yhat.test <- predict(model1, test.data)
lines(test$Temp[test.idx], yhat.test[test.idx], col="green")
```



Task 4: Extending the data

```
plot(train$Temp[train.idx], yhat.train[train.idx], ylab="Predictions", main="Predicted ozone concentrati
     xlim=c(0, 120),
     ylim=c(-200, 30)
     )

lines(test$Temp[test.idx], yhat.test[test.idx], col="green")


newtemp <- seq(0, 120)

newtemp %>%
  getsplines() %>%
  .$X %>%
  as.data.frame() %>%
  rename_all(str_replace_all, "V", "X") %>%
  predict(model1, .) %>%
  lines(newtemp, ., col="orange")
```

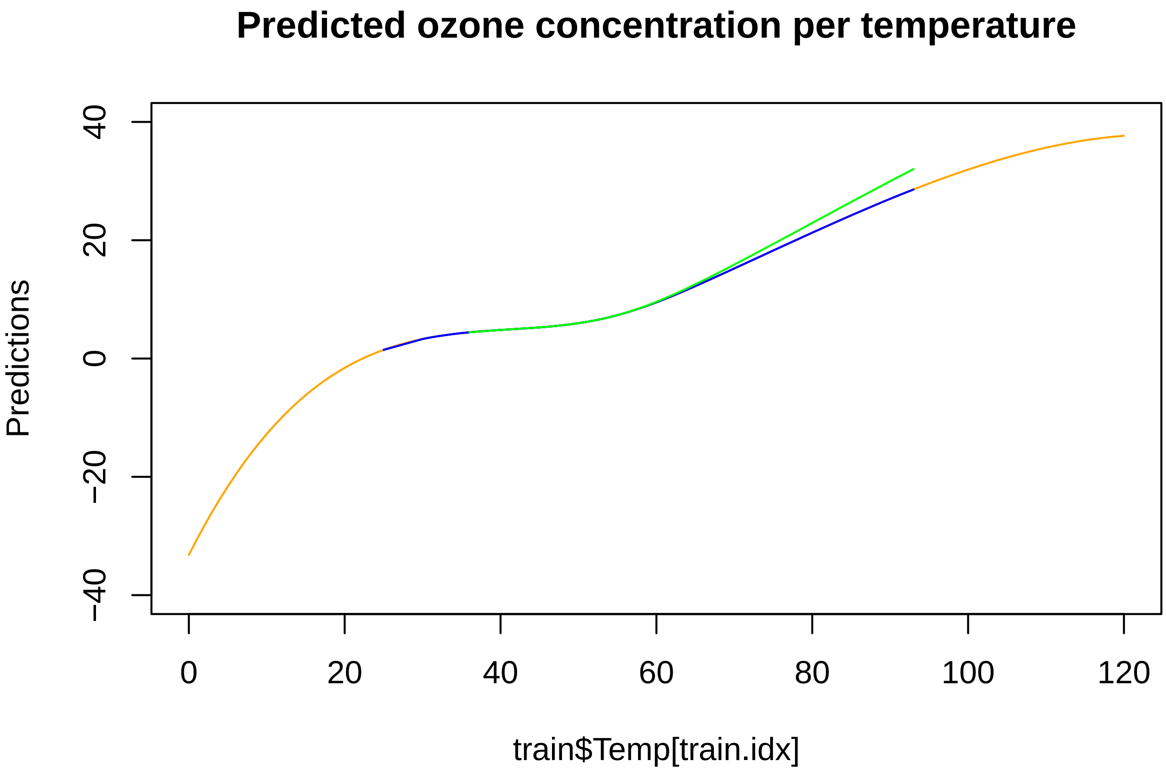


Task 5: Knots at custom points

```
getsplines <- function(x, nknots=2, M=4, knots=NULL){
  # nknots ... number of knots -> placed at regular quantiles
  # M ... M-1 is the degree of the polynomial
  n <- length(x)
  # X will not get an intercept column
  X <- matrix(NA, nrow=n, ncol=(M - 1) + nknots)
  for (i in 1:(M - 1)){
    X[,i] <- x^i
    }
  # now the basis functions for the constraints:
  if (is.null(knots)){
    print("a")
    # create knots from quantiles
    quant <- seq(0, 1, 1 / (nknots + 1))[c(2 : (nknots + 1))]
    qu <- quantile(x,quant)
  } else {
    # if custom knots have been given, use them as knots
    qu = knots
    # and calculate the quantile percentages of the knots
    quant <- ecdf(x)(knots)
  }
  for (i in M : (M + nknots - 1)){
    X[,i] <- ifelse(x - qu[i - M + 1] < 0, 0, (x - qu[i - M + 1])^(M - 1))
  }
  list(x=x, X=X, quantiles=quant, xquantiles=qu)
}
```

The function is now modiffied to accept custom knot values as an optional argument.

```
plot(train$Temp[train.idx], yhat.train[train.idx], ylab="Predictions", main="Predicted ozone concentrati
     xlim=c(0, 120),
     ylim=c(-40, 40)
     )



# get the knots of the train data splines
training_knots <- splines.temp.train$xquantiles %>% unname()

newtemp <- seq(0, 120)
newtemp %>%
  getsplines(knots = training_knots) %>%
  .$X %>%
  as.data.frame() %>%
  rename_all(str_replace_all, "V", "X") %>%
  predict(model1, .) %>%
  lines(newtemp, ., col="orange")

lines(train$Temp[train.idx], yhat.train[train.idx], col="blue")
lines(test$Temp[test.idx], yhat.test[test.idx], col="green")
```



Task 6: Repeating the analyses with the log response

```
training.data <- data.frame(splines.temp.train$X, Ozone=train$Ozone)
# if specify using log for every variable in the data
# i also add a small constant to every cell to avoid lm(0)
model2 <- lm(Ozone ~ log(X1) + log(X2) + log(X3) + log(X4) + log(X5),
             data=training.data+1e-12)
# predict train data
yhat.train <- predict(model2, training.data) %>% exp()
```

```
## Warning in predict.lm(model2, training.data): prediction from a rank-deficient
## fit may be misleading
```

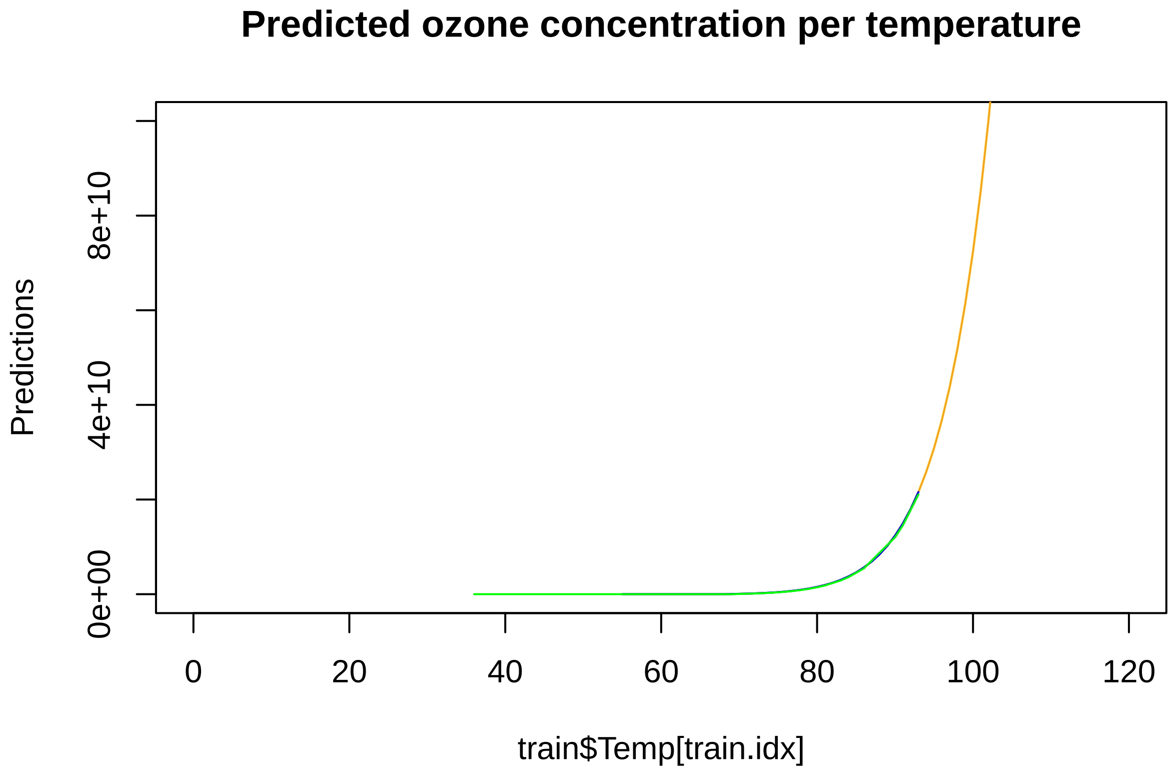
```
# predict test data
test.data <- data.frame(splines.temp.test$X, Ozone=test$Ozone)
yhat.test <- predict(model2, test.data+1e-12) %>% exp()
```

```
## Warning in predict.lm(model2, test.data + 1e-12): prediction from a rank-
## deficient fit may be misleading
```

```
plot(train$Temp[train.idx], yhat.train[train.idx], ylab="Predictions", main="Predicted ozone concentrati
     xlim=c(0, 120),
     ylim=c(0, 10e10)
     )


newtemp <- seq(0, 120)
newtemp %>%
  getsplines(knots = training_knots) %>%
  .$X %>%
  as.data.frame() %>%
  rename_all(str_replace_all, "V", "X") %>%
  predict(model2, .) %>%
  exp() %>%
  lines(newtemp, ., col="orange")
```

```
## Warning in predict.lm(model2, .): prediction from a rank-deficient fit may be
## misleading
```

```
lines(train$Temp[train.idx], yhat.train[train.idx], col="blue")
lines(test$Temp[test.idx], yhat.test[test.idx], col="green")
```



Now the predicted ozone concentrations are all above 0, but they explode for temperatures higher than 80.