Stats 253 Project

Raymond Gu & Howard Cai

3/20/2021

Regression models:

Linear Regression Model

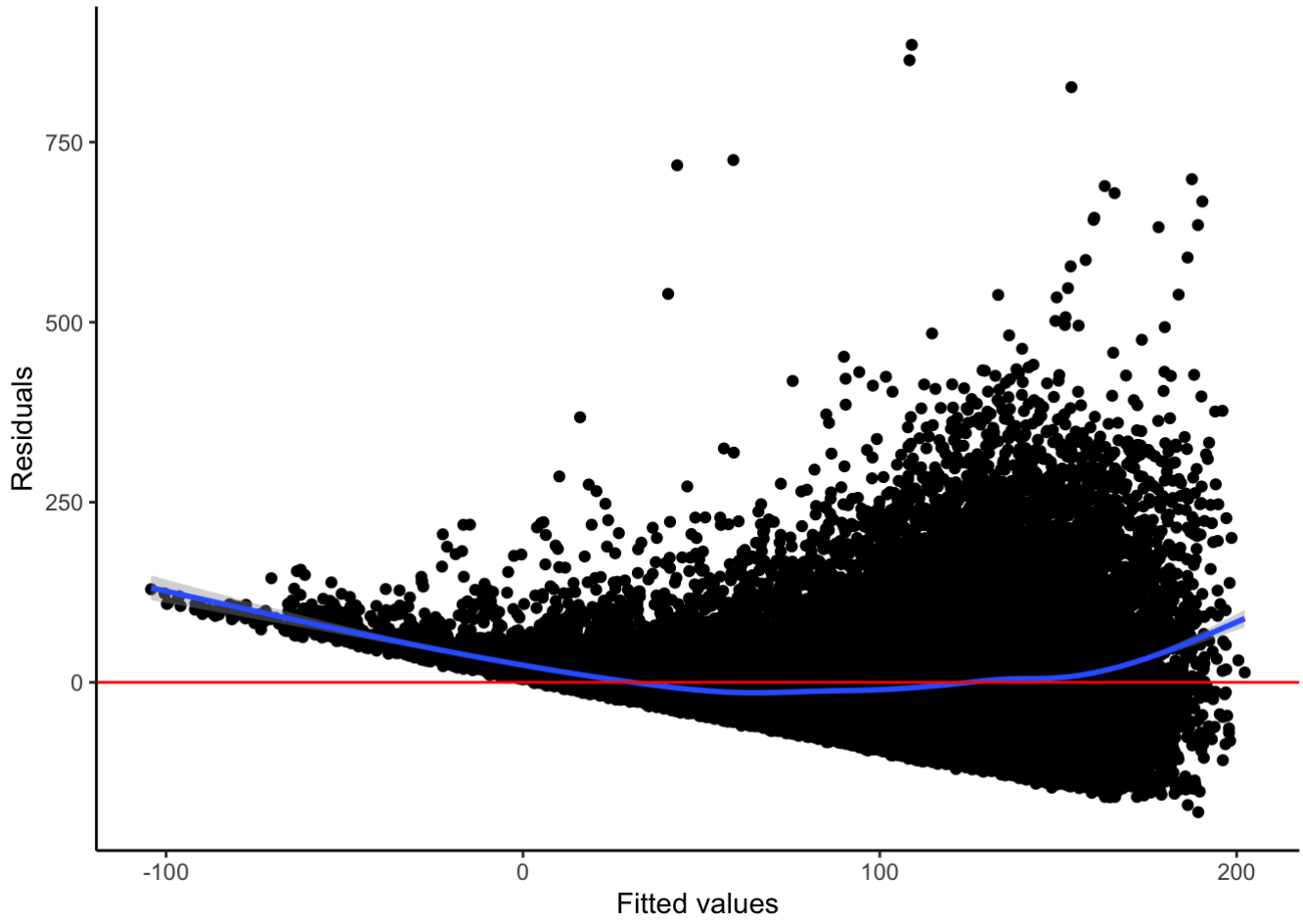
Build a linear regression model with 7 possibly good predicotrs

```
linearModel <- lm(pm2.5 ~ DEWP+TEMP+PRES+cbwd+Iws+Ir+Is, data = PM2.5)

summary(linearModel)
```

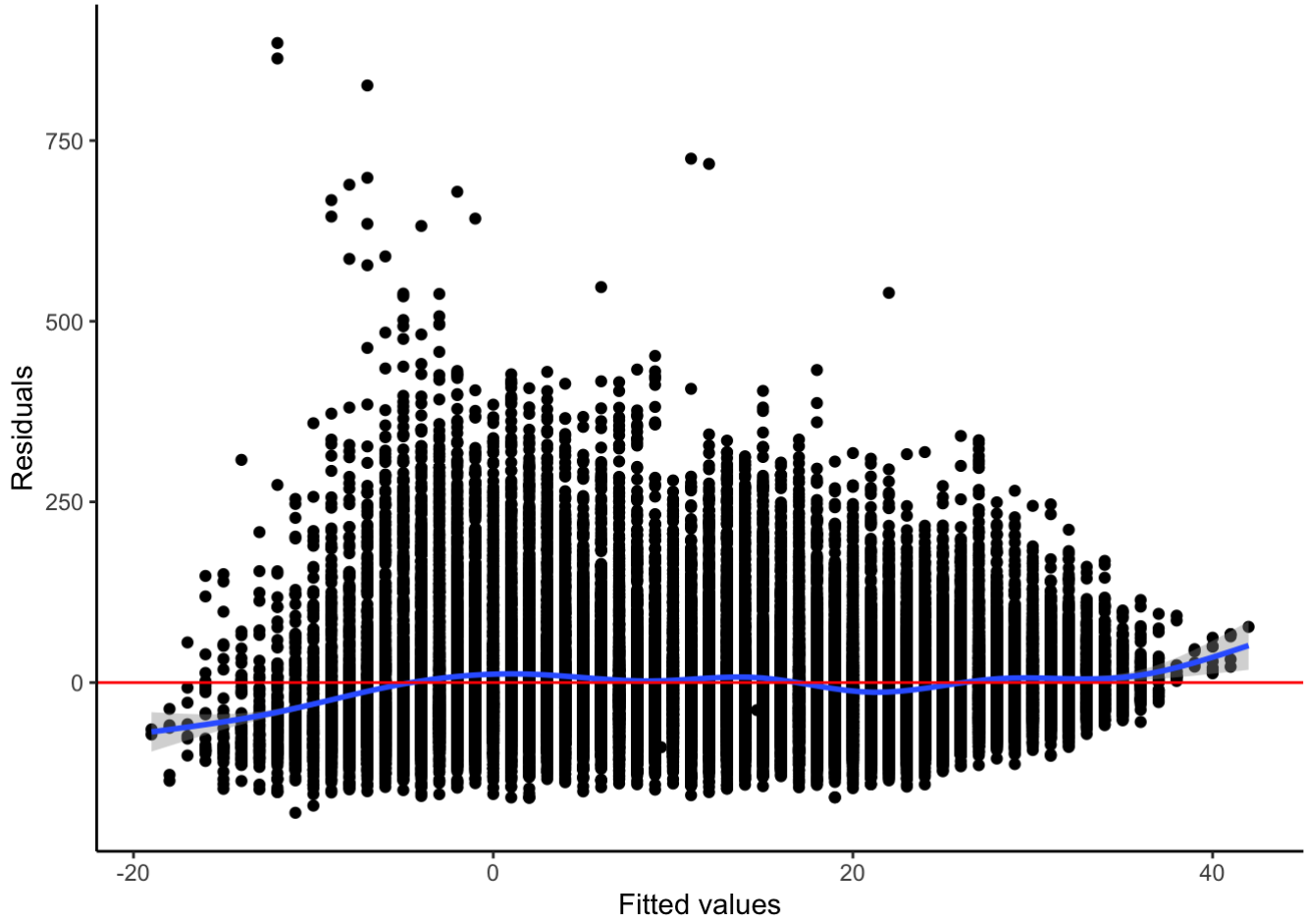
```
##
## Call:
## lm(formula = pm2.5 ~ DEWP + TEMP + PRES + cbwd + Iws + Ir + Is,
##     data = PM2.5)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -180.23  -51.32  -15.80   31.36  885.04
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.829e+03  7.196e+01  25.418  < 2e-16 ***
## DEWP         4.025e+00  5.324e-02  75.597  < 2e-16 ***
## TEMP        -6.254e+00  6.800e-02 -91.969  < 2e-16 ***
## PRES        -1.616e+00  7.033e-02 -22.972  < 2e-16 ***
## cbwdNE      -2.674e+01  1.428e+00 -18.722  < 2e-16 ***
## cbwdNW      -3.011e+01  1.173e+00 -25.667  < 2e-16 ***
## cbwdSE       3.882e+00  1.092e+00   3.555 0.000378 ***
## Iws         -2.013e-01  8.752e-03 -23.003  < 2e-16 ***
## Ir          -6.230e+00  2.783e-01 -22.387  < 2e-16 ***
## Is          -3.301e+00  5.028e-01  -6.564 5.28e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79.16 on 41747 degrees of freedom
## Multiple R-squared:  0.2606, Adjusted R-squared:  0.2604
## F-statistic:  1635 on 9 and 41747 DF,  p-value: < 2.2e-16
```

Plot the residual versus fitted value plot

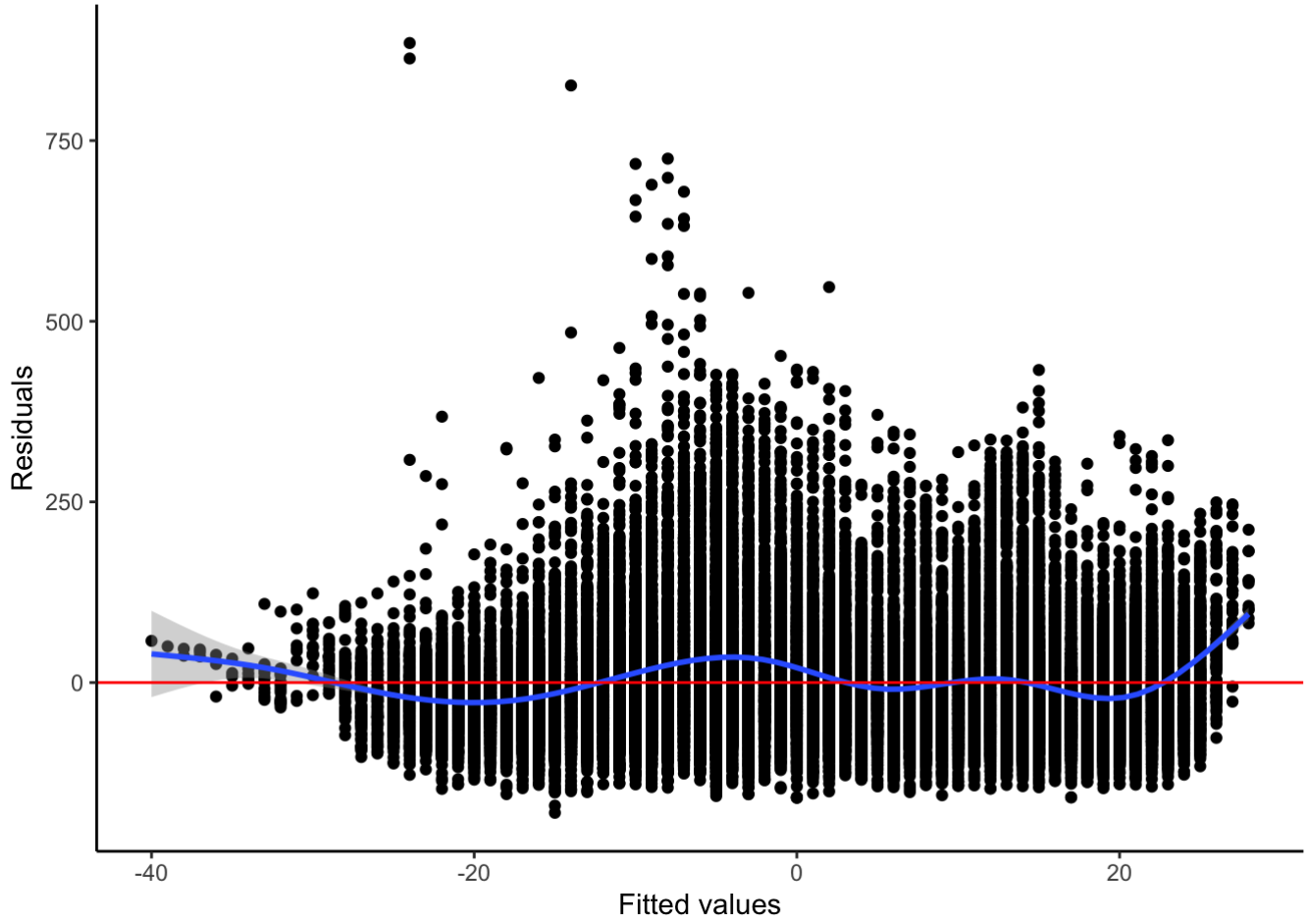
```
linearModel_Output <- broom::augment(linearModel, newdata = PM2.5)

ggplot(linearModel_Output, aes(x = .fitted, y = .resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```



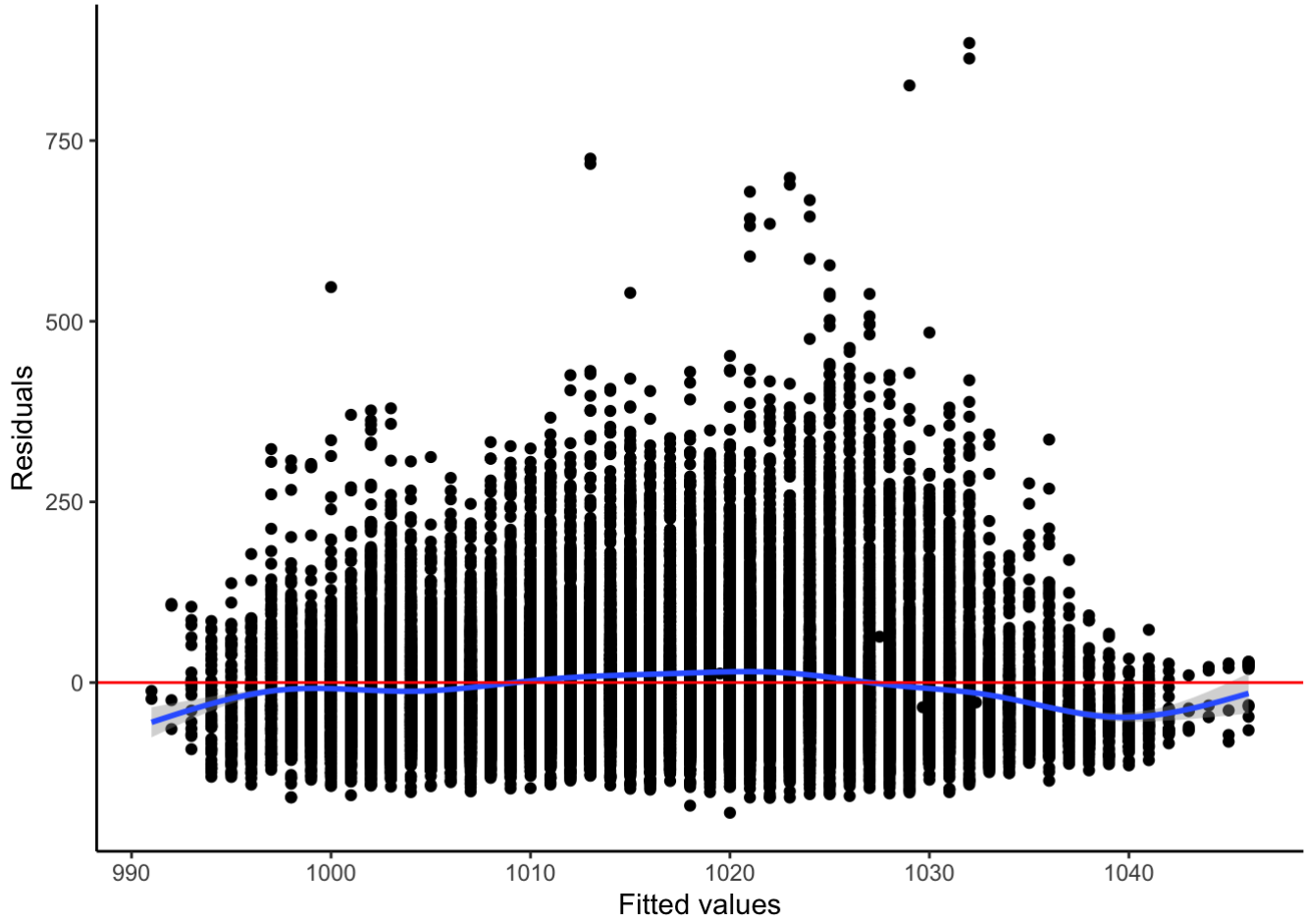

```
ggplot(linearModel_Output, aes(x = TEMP, y = .resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```



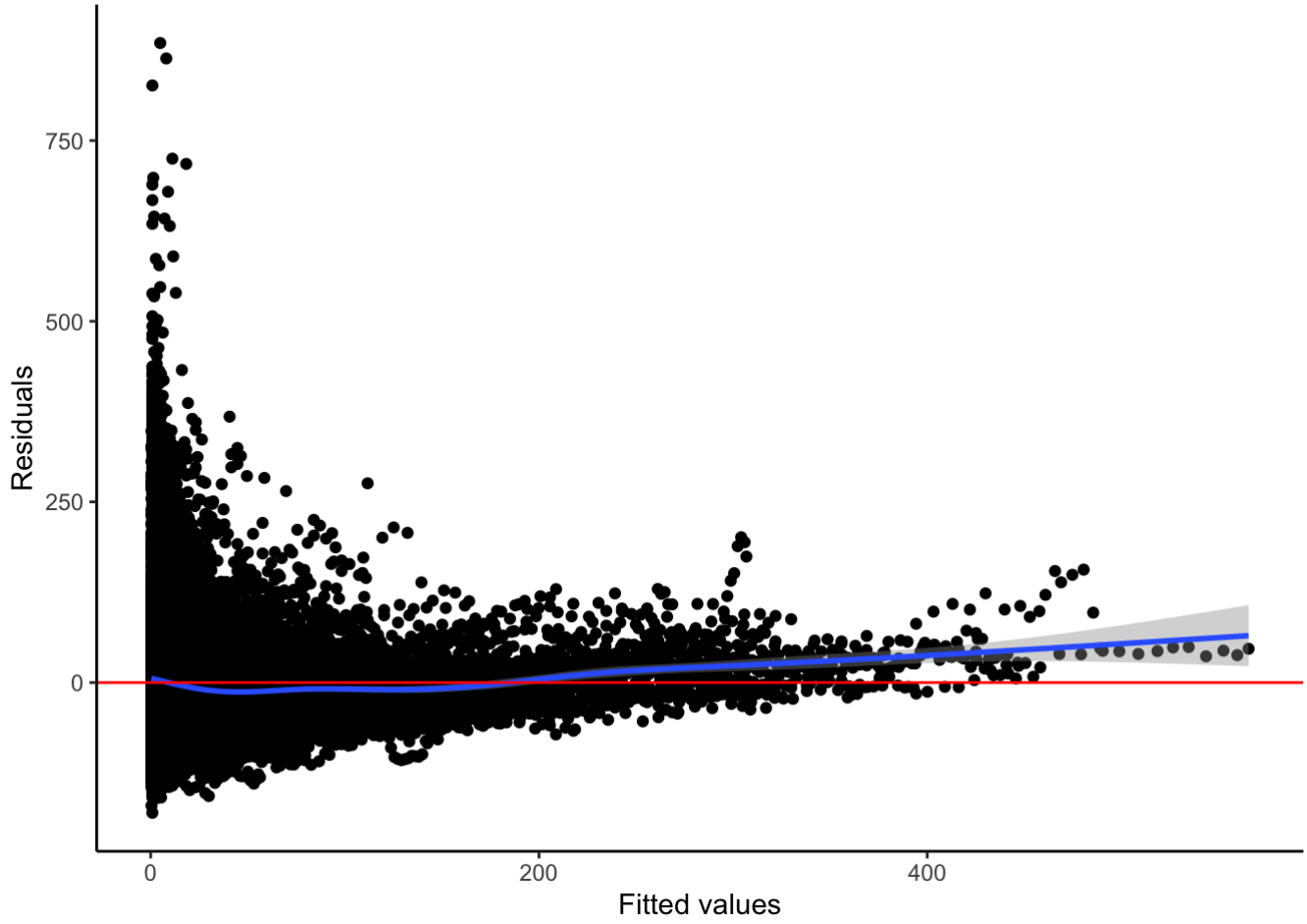

```
ggplot(linearModel_Output, aes(x = DEWP, y = .resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```



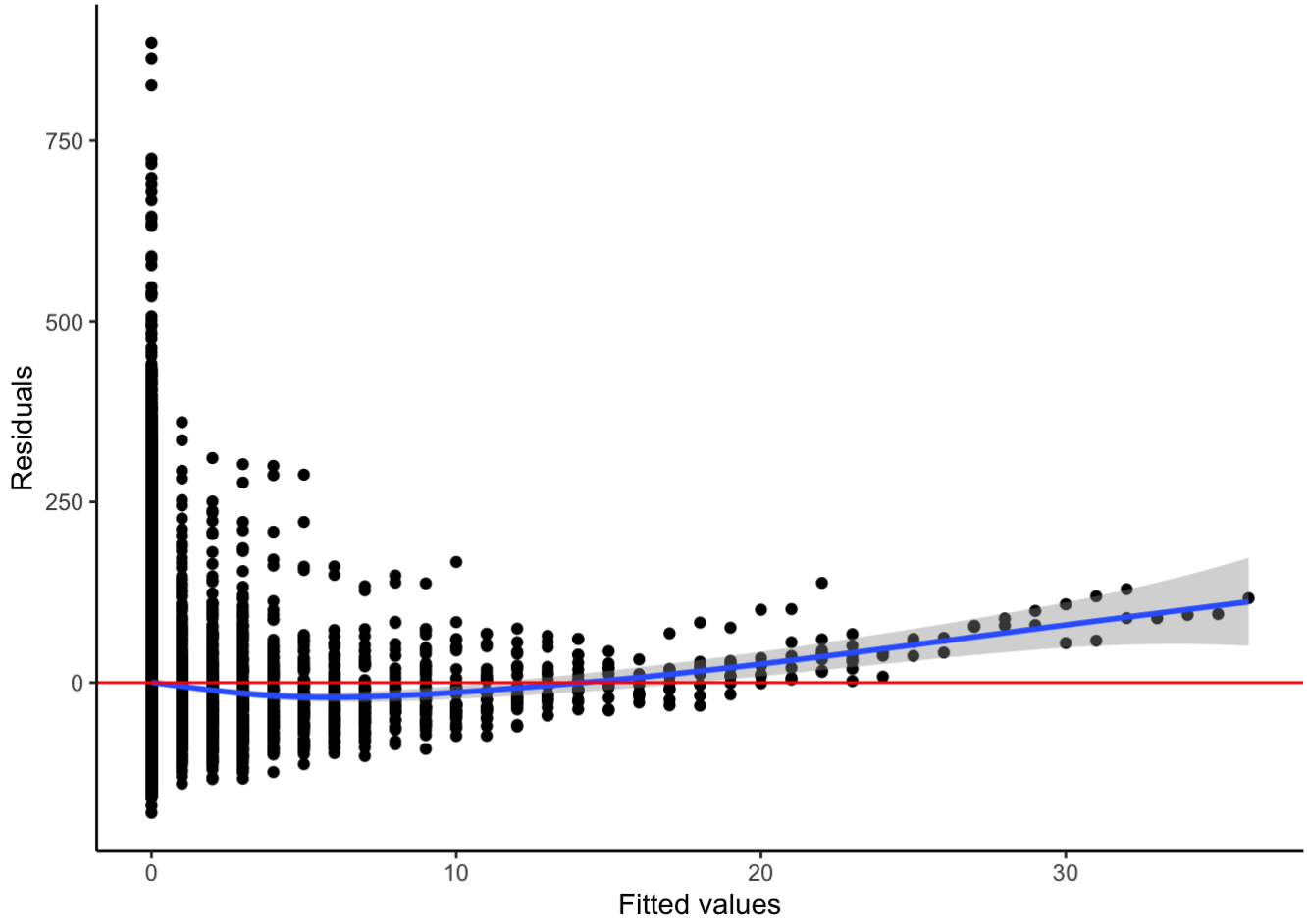

```
ggplot(linearModel_Output, aes(x = PRES, y = .resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```



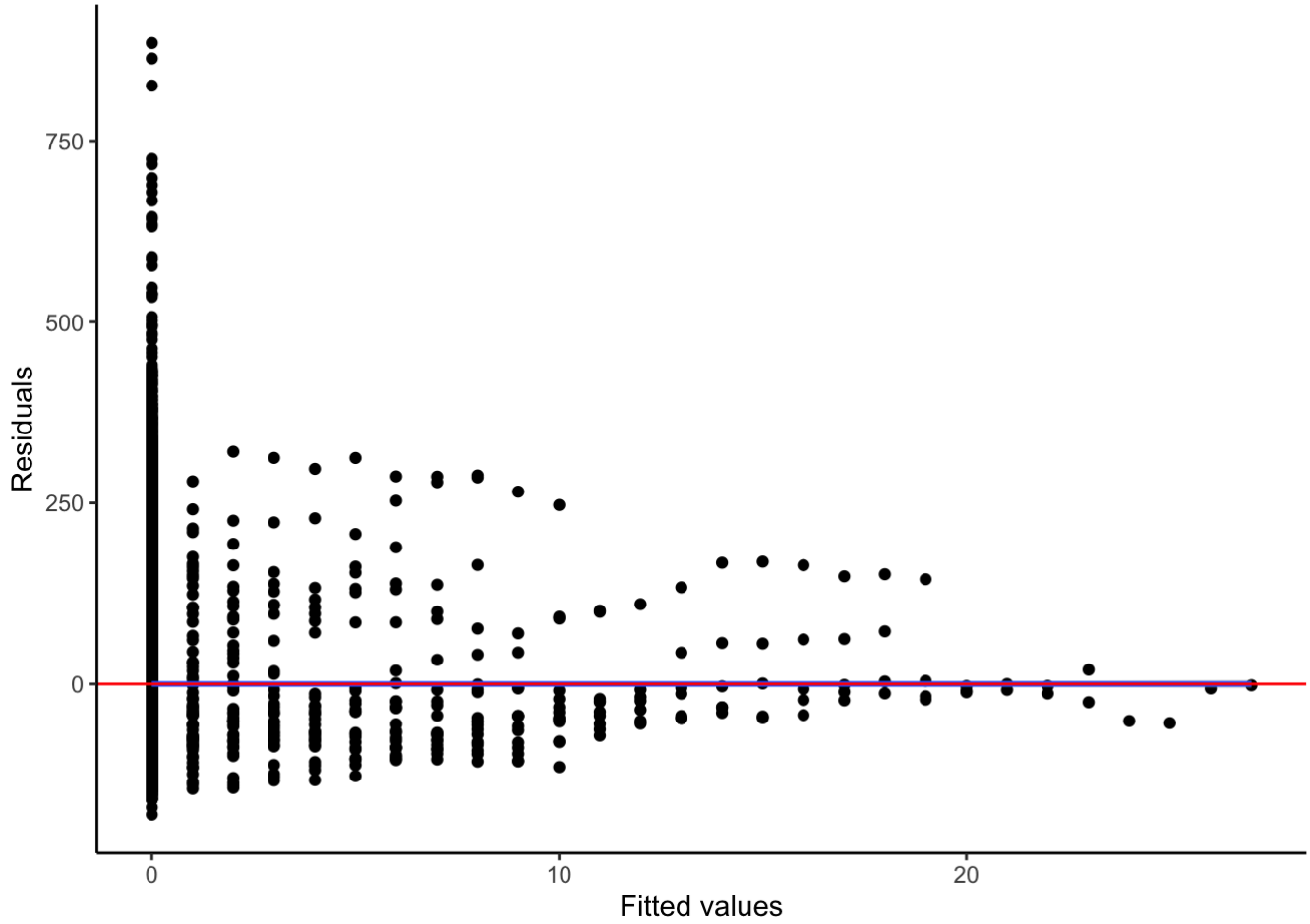

```
ggplot(linearModel_Output, aes(x = Iws, y = .resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```




```
ggplot(linearModel_Output, aes(x = Ir, y = .resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```




```
ggplot(linearModel_Output, aes(x = Is, y = .resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```



From the individual predictor versus residual plot, we see that we might need a non-linear model for DEWP.

Calculate the Training MAE

```
mean(abs(linearModel_Output$.resid))
```

```
## [1] 57.20498
```

On average, we're off in our pm2.5 predictions by about 57.2 ug/m^3 .This value could be better if we incorporate all the predictors into the model.

Perform 10-fold cross-validation for our linearModel to estimate the test MAE, and compare it with a model with every variable.

```
set.seed(253)

linearModel_cv <- train(
    pm2.5 ~ DEWP+TEMP+PRES+cbwd+Iws+Ir+Is,
    data = PM2.5,
    method = "lm",
    trControl = trainControl(method = "cv", number = 10),
    na.action = na.omit
)

summary(linearModel_cv)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -180.23  -51.32  -15.80   31.36  885.04
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.829e+03  7.196e+01  25.418  < 2e-16 ***
## DEWP         4.025e+00  5.324e-02  75.597  < 2e-16 ***
## TEMP        -6.254e+00  6.800e-02 -91.969  < 2e-16 ***
## PRES        -1.616e+00  7.033e-02 -22.972  < 2e-16 ***
## cbwdNE      -2.674e+01  1.428e+00 -18.722  < 2e-16 ***
## cbwdNW      -3.011e+01  1.173e+00 -25.667  < 2e-16 ***
## cbwdSE       3.882e+00  1.092e+00   3.555 0.000378 ***
## Iws         -2.013e-01  8.752e-03 -23.003  < 2e-16 ***
## Ir          -6.230e+00  2.783e-01 -22.387  < 2e-16 ***
## Is          -3.301e+00  5.028e-01  -6.564 5.28e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79.16 on 41747 degrees of freedom
## Multiple R-squared:  0.2606, Adjusted R-squared:  0.2604
## F-statistic:  1635 on 9 and 41747 DF,  p-value: < 2.2e-16
```

```
linearModel_cv$results
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1      TRUE 79.14787 0.2607165 57.21832 1.963477 0.01265153 0.9921828
```

```
set.seed(253)

linearModel1_cv <- train(
    pm2.5 ~ . ,
    data = PM2.5,
    method = "lm",
    trControl = trainControl(method = "cv", number = 10),
    na.action = na.omit
)

summary(linearModel1_cv)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -173.16  -50.94  -15.22   31.51  885.77
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.254e+06  4.854e+05   6.703 2.07e-11 ***
## No           1.846e-01  2.755e-02   6.701 2.09e-11 ***
## year        -1.618e+03  2.415e+02  -6.699 2.13e-11 ***
## month       -1.359e+02  2.013e+01  -6.754 1.46e-11 ***
## day         -3.657e+00  6.613e-01  -5.530 3.22e-08 ***
## hour         1.065e+00  6.862e-02  15.521  < 2e-16 ***
## DEWP         4.366e+00  5.668e-02  77.015  < 2e-16 ***
## TEMP        -6.505e+00  7.297e-02 -89.143  < 2e-16 ***
## PRES        -1.686e+00  7.202e-02 -23.409  < 2e-16 ***
## cbwdNE      -2.549e+01  1.415e+00 -18.006  < 2e-16 ***
## cbwdNW      -2.728e+01  1.168e+00 -23.356  < 2e-16 ***
## cbwdSE       5.809e-01  1.093e+00   0.532    0.595
## Iws         -1.987e-01  8.722e-03 -22.776  < 2e-16 ***
## Is          -3.483e+00  4.990e-01  -6.979 3.01e-12 ***
## Ir          -6.467e+00  2.757e-01 -23.457  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 78.34 on 41742 degrees of freedom
## Multiple R-squared:  0.276,  Adjusted R-squared:  0.2758
## F-statistic:  1137 on 14 and 41742 DF,  p-value: < 2.2e-16
```

```
linearModel1_cv$results
```

```
##   intercept    RMSE  Rsquared     MAE   RMSESD RsquaredSD     MAESD
## 1      TRUE 78.3236 0.2760449 56.7892 1.918812 0.01193944 0.9685252
```

From the results, we see that our model with fewer predictors is better based on cross-validation estimates of test error, though the difference is rather small. We have a MAE of 56.7892 and a MAESD of 0.9685252 for our better model.

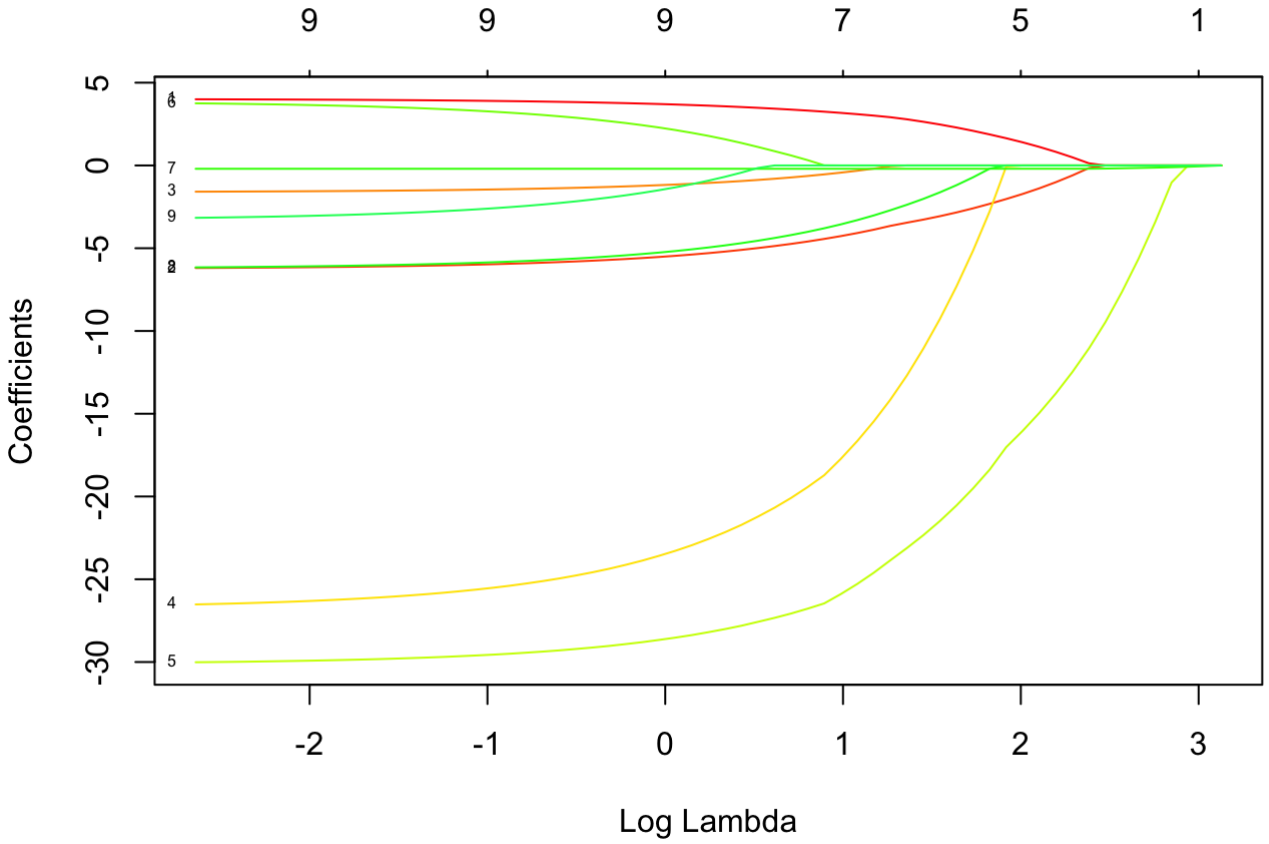
LASSO

Fitting a LASSO model

```
set.seed(253)

lasso_mod <- train(
    pm2.5 ~ DEWP+TEMP+PRES+cbwd+Iws+Ir+Is,
    data = PM2.5,
    method = "glmnet",
    trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"),
    tuneGrid = data.frame(alpha = 1, lambda = seq(0, 10, length.out = 100)),
    metric = "MAE",
    na.action = na.omit
)
```

Plot the coefficient estimates as a function of λ.

```
plot(lasso_mod$finalModel, xvar = "lambda", label = TRUE, col = rainbow(20))
```



Displaying codebook for which variables the numbers correspond to

```
rownames(lasso_mod$finalModel$beta)
```

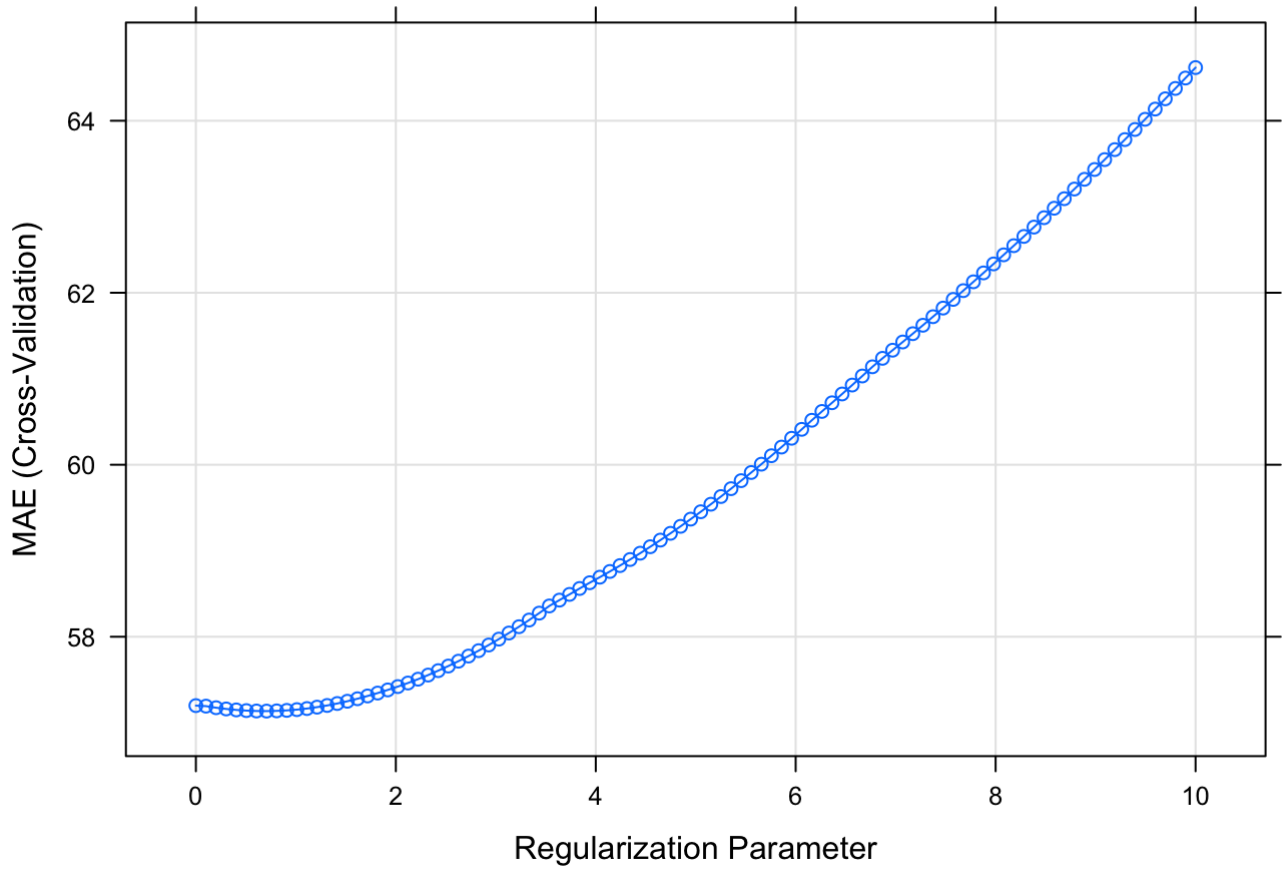
```
## [1] "DEWP"   "TEMP"   "PRES"   "cbwdNE" "cbwdNW" "cbwdSE" "Iws"    "Ir"
## [9] "Is"
```

From the Graph, we can tell that line 5(“cbwdNW”) is the most persistent one, indicating itself as a very strong predictor. Also, IS is the lest persistent one,

indicating itself as a very weak predictor.

Plot CV error estimates for the different models to find the best model

```
plot(lasso_mod)
```



The graph is kind of U-shaped, with lowest MAE at approximately 1

```
lasso_mod$bestTune
```

```
##    alpha   lambda
## 21     1 2.020202
```

So the best model here is with parameter alpha = 1 and lambda = 1.818182

```
coef(lasso_mod$finalModel, lasso_mod$bestTune$lambda)
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 907.5796467
## DEWP          3.3770811
## TEMP         -4.7448951
## PRES         -0.7276076
## cbwdNE      -20.1180734
## cbwdNW      -27.0833413
## cbwdSE        0.6170566
## Iws          -0.2018918
## Ir           -4.2135298
## Is            .
```

The “best” LASSO model here has every predictor that we choose. The coefficient for IS is -0.029. It is so small compared to other coefficients that rstudio chooses to display it with . instead of the actual number.

Subset Selection

Perform backward stepwise selection with cross-validation

```
set.seed(253)

back_step_mod <- train(
    pm2.5  ~ DEWP+TEMP+PRES+Iws+Is+Ir+cbwd,
    data = PM2.5,
    method = "leapBackward",
    tuneGrid = data.frame(nvmax = 1:9),
    trControl = trainControl(method = "cv", number = 10),
    metric = "MAE",
    na.action = na.omit
)
```

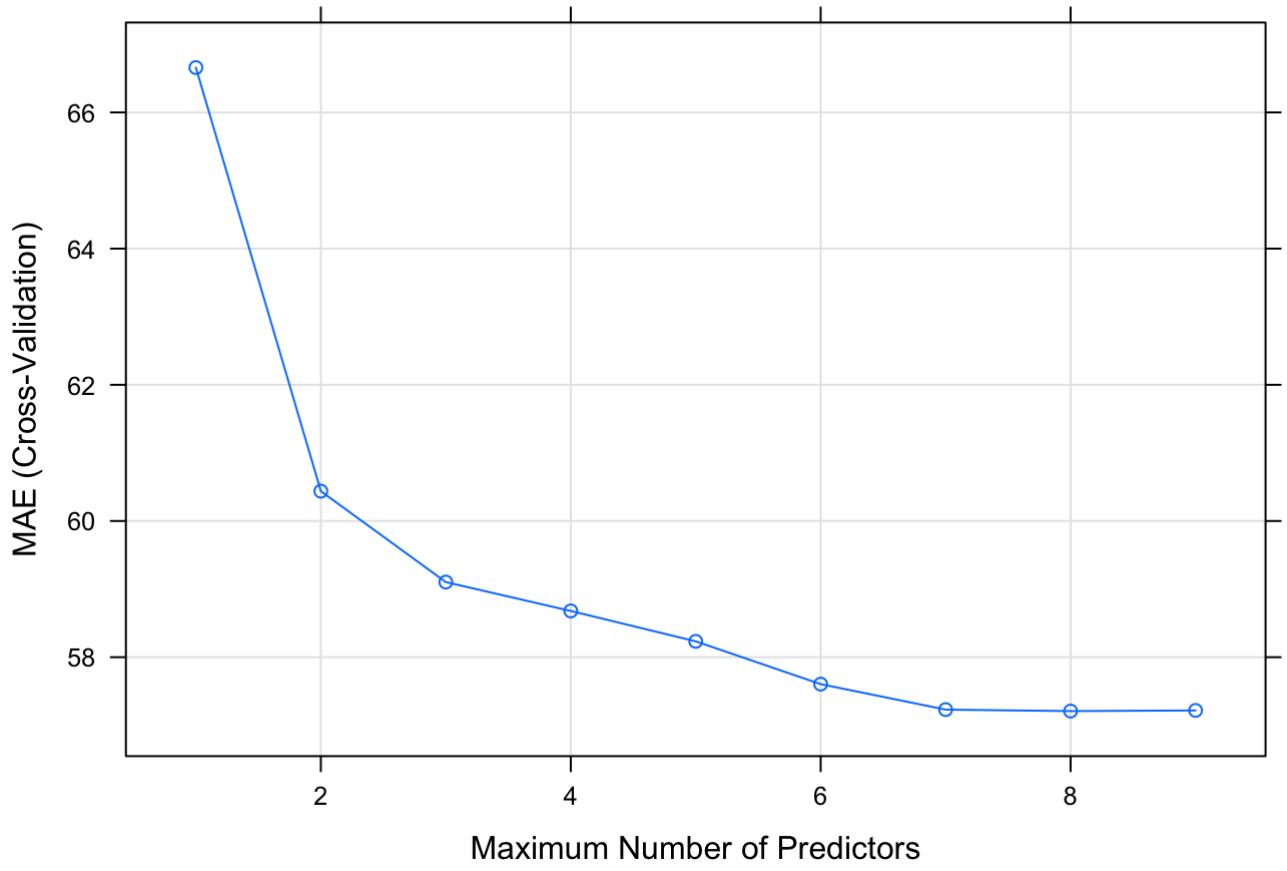
Results

```
summary(back_step_mod)
```

```
## Subset selection object
## 9 Variables  (and intercept)
##        Forced in Forced out
## DEWP       FALSE      FALSE
## TEMP       FALSE      FALSE
## PRES       FALSE      FALSE
## Iws        FALSE      FALSE
## Is         FALSE      FALSE
## Ir         FALSE      FALSE
## cbwdNE     FALSE      FALSE
## cbwdNW     FALSE      FALSE
## cbwdSE     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: backward
##          DEWP TEMP PRES Iws Is  Ir  cbwdNE cbwdNW cbwdSE
## 1  ( 1 ) "*"  " "  " "  " " " " " " " "    " "    " "
## 2  ( 1 ) "*"  "*"  " "  " " " " " " " "    " "    " "
## 3  ( 1 ) "*"  "*"  " "  " " " " " " " "    "*"    " "
## 4  ( 1 ) "*"  "*"  " "  " " " " " " "*"    "*"    " "
## 5  ( 1 ) "*"  "*"  " "  "*" " " " " "*"    "*"    " "
## 6  ( 1 ) "*"  "*"  "*"  "*" " " " " "*"    "*"    " "
## 7  ( 1 ) "*"  "*"  "*"  "*" " " "*" "*"    "*"    " "
## 8  ( 1 ) "*"  "*"  "*"  "*" "*" "*" "*"    "*"    " "
```

Examine the 10-fold CV MAE for each of the 9 models in the backward stepwise sequence:

```
plot(back_step_mod)
```



Find the best model and the coefficients associated with it

```
back_step_mod$bestTune
```

```
##   nvmax
## 8     8
```

```
coef(back_step_mod$finalModel, id = back_step_mod$bestTune$nvmax)
```

```
##  (Intercept)         DEWP         TEMP         PRES          Iws           Is
## 1829.6488575    4.0143553   -6.2216418   -1.6144426   -0.1975703   -3.2102848
##           Ir       cbwdNE       cbwdNW
##   -6.2340130  -29.0207270  -32.5208578
```

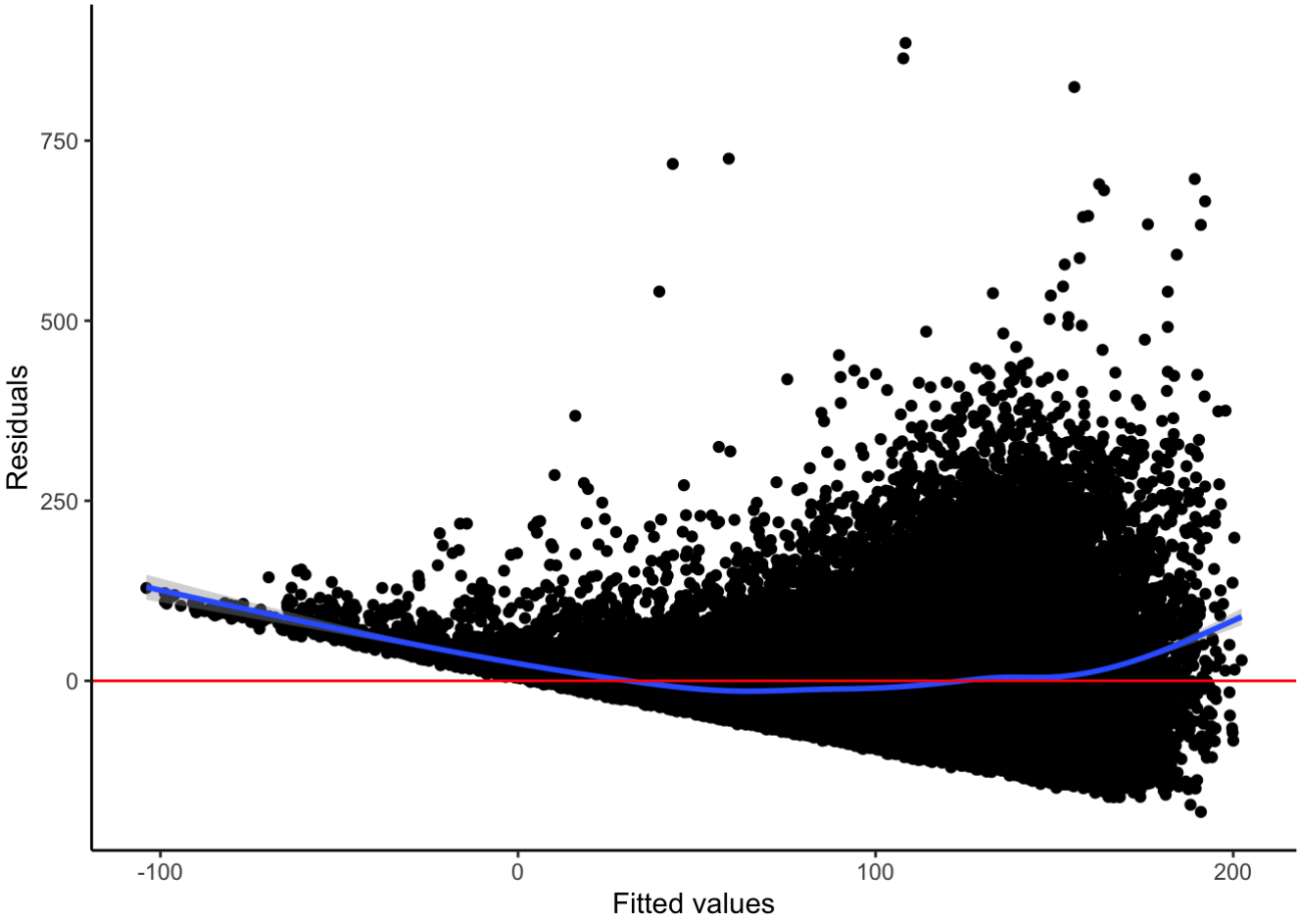
```
back_step_mod$results%>% filter(nvmax==8)
```

```
##   nvmax     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1     8 79.15794 0.2605269 57.20852 1.970284 0.01274628 0.9960053
```

The best model has 8 predictors, every predictor except for “cbwdSE”. However, we will need to include it, because we include cbwd as a categorical predictor and we should include all of its sub-predictors. The MAE of our best backward selection model is 57.21 ug/m^3, and the MAESD is 0.9960053 ug/m^3.

Evaluate the model with residual versus fitted value

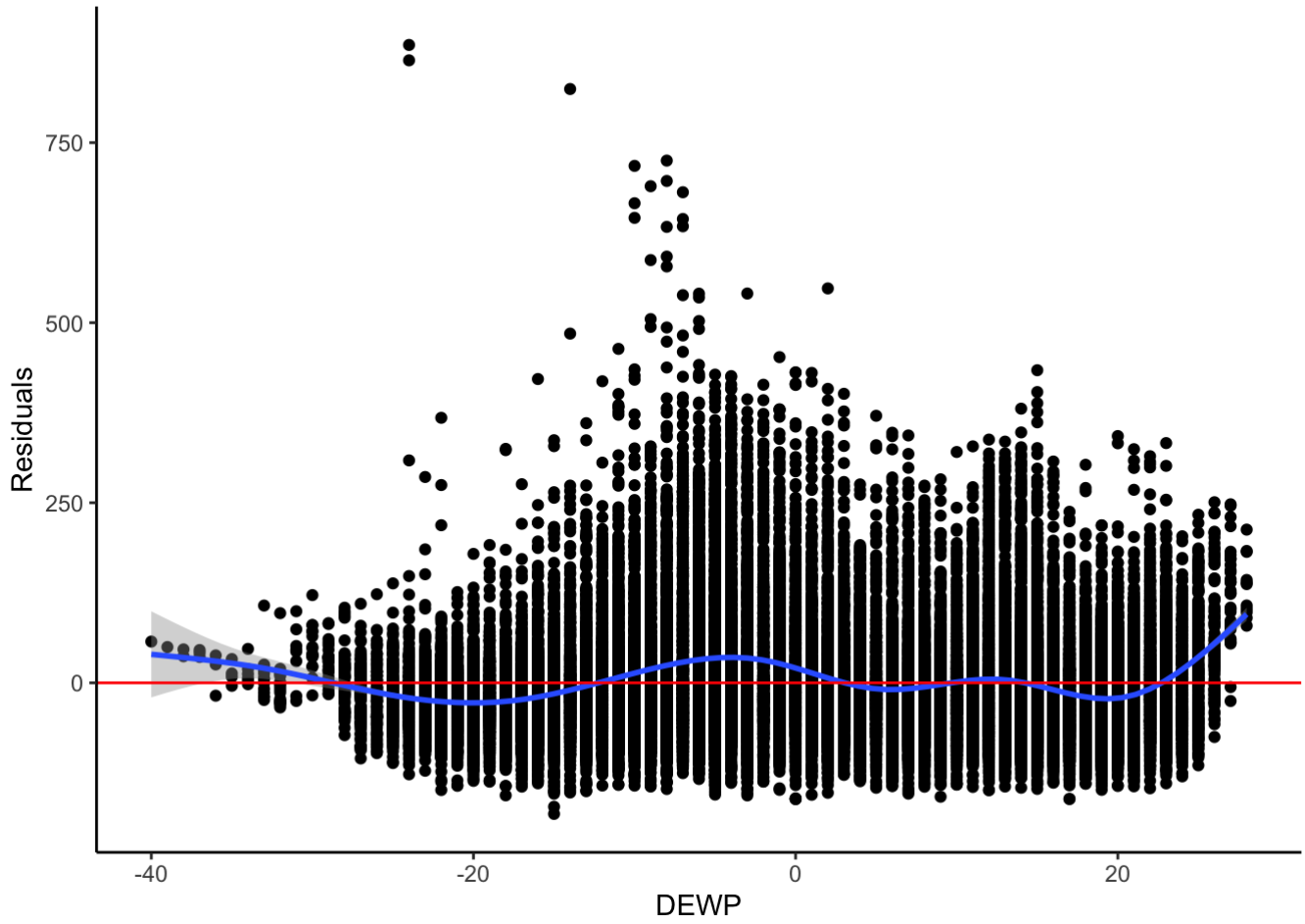
```
back_step_mod_out <- PM2.5 %>%
    mutate(
        fitted = predict(back_step_mod, newdata = PM2.5),
        resid = pm2.5 - fitted
    )

ggplot(back_step_mod_out, aes(x = fitted, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Fitted values", y = "Residuals")
```



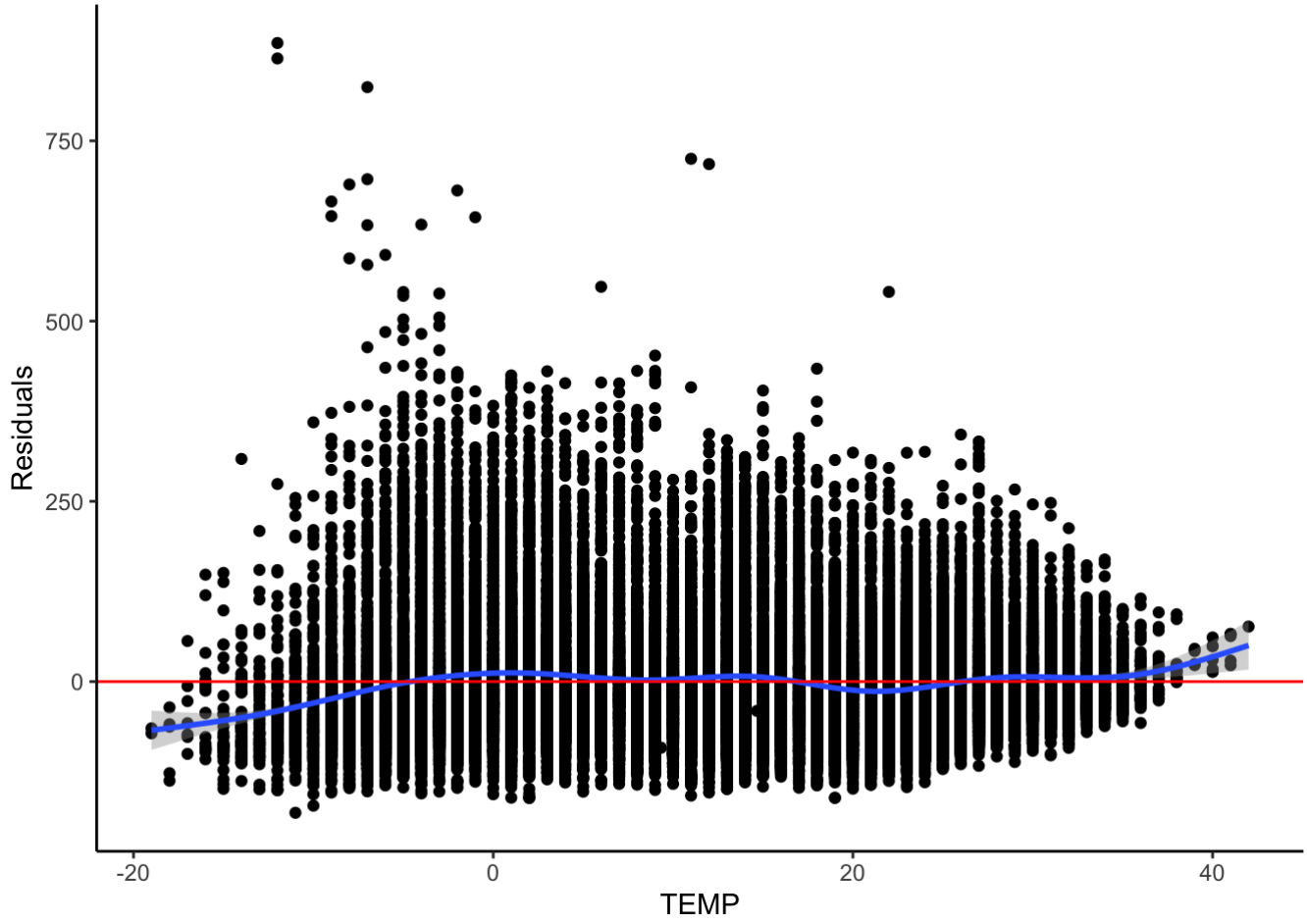
From the plot, we see that most of the residual are above 0, indicating a underestimation of our model.

Plot the individual preditor's residual plot

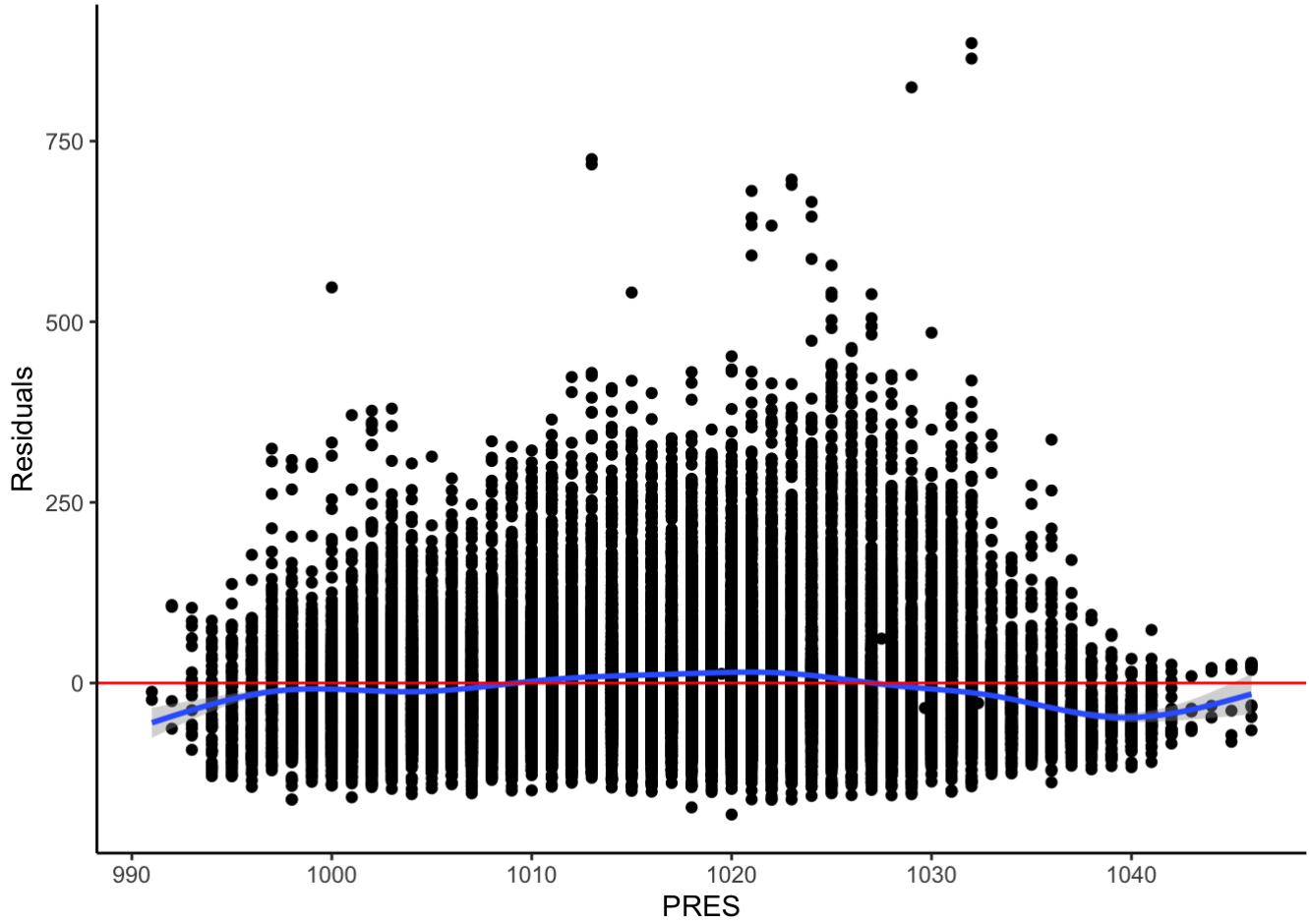
```
#DEWP+TEMP+PRES+Iws+Is+Ir

ggplot(back_step_mod_out, aes(x = DEWP, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "DEWP", y = "Residuals")
```



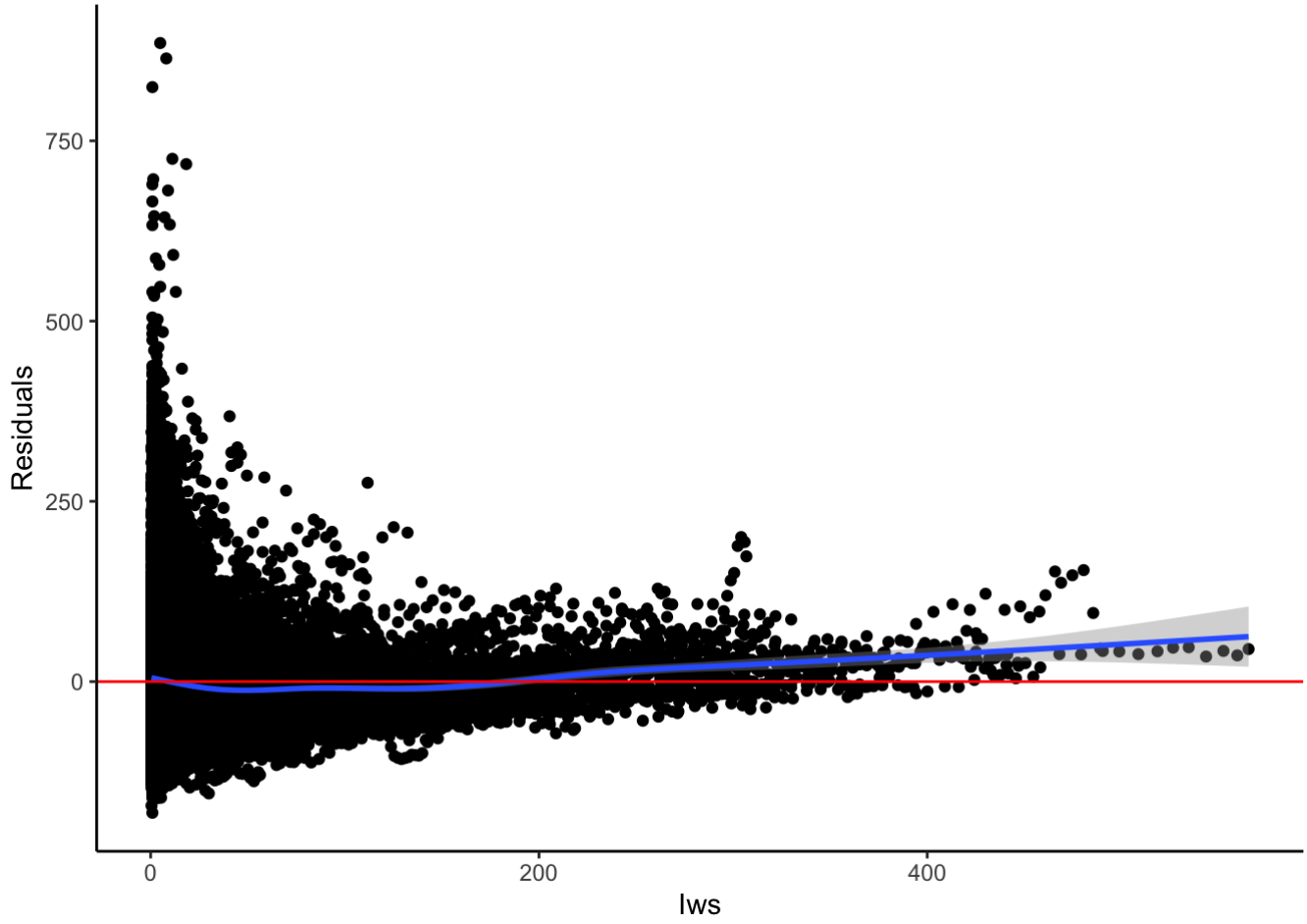

```
ggplot(back_step_mod_out, aes(x = TEMP, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "TEMP", y = "Residuals")
```



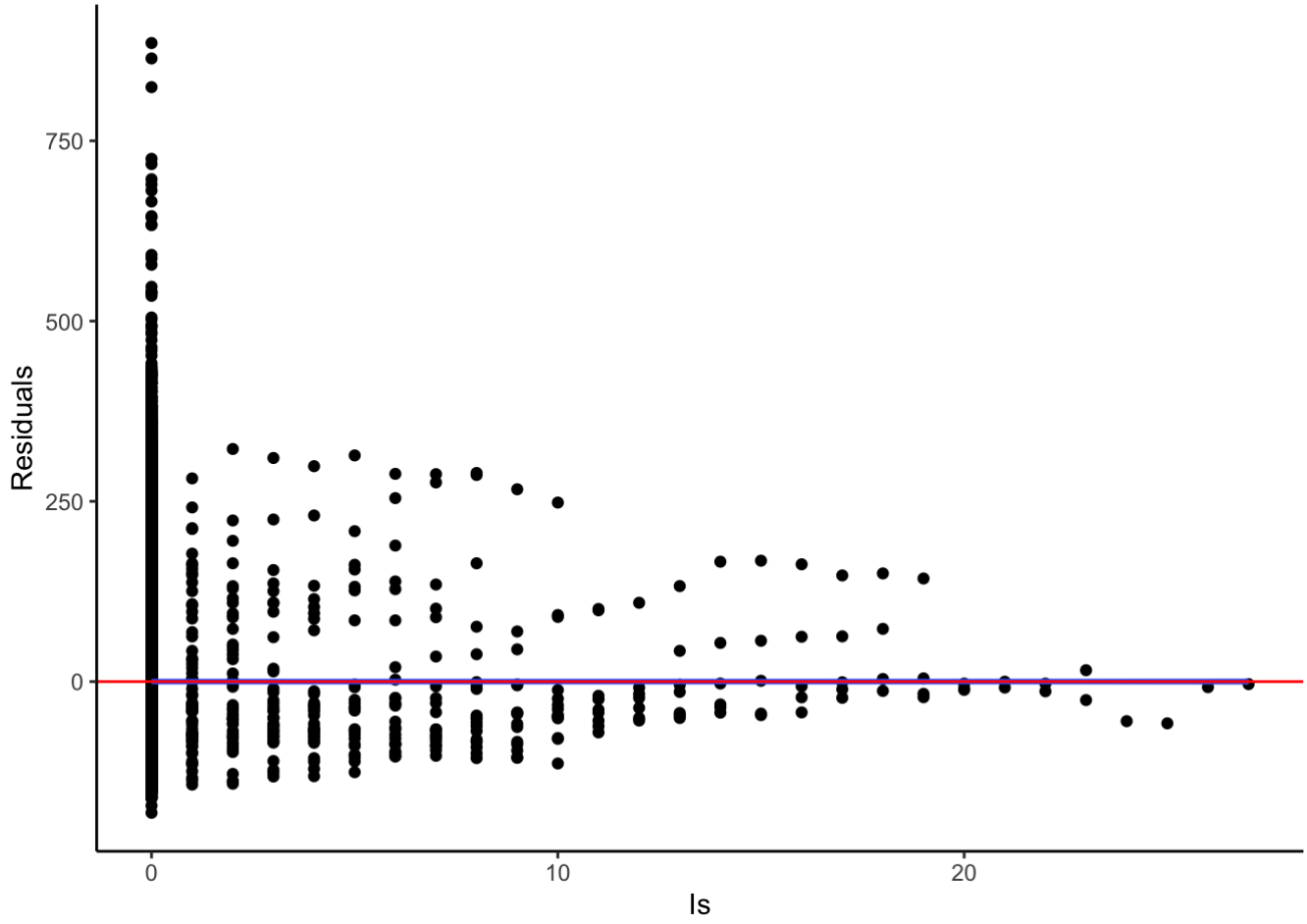

```
ggplot(back_step_mod_out, aes(x = PRES, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "PRES", y = "Residuals")
```



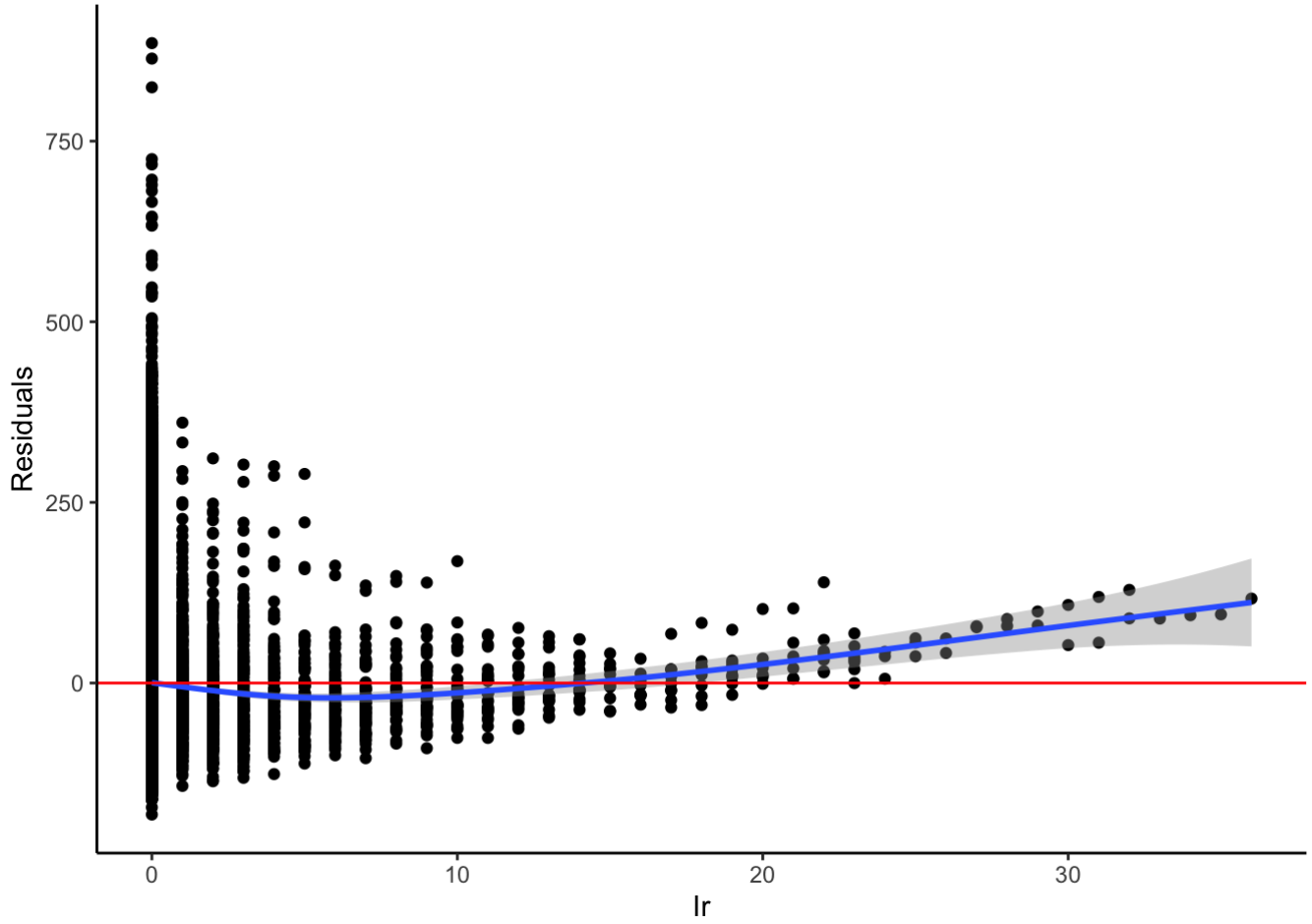

```
ggplot(back_step_mod_out, aes(x = Iws, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Iws", y = "Residuals")
```




```
ggplot(back_step_mod_out, aes(x = Is, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Is", y = "Residuals")
```




```
ggplot(back_step_mod_out, aes(x = Ir, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red") +
    theme_classic() +
    labs(x = "Ir", y = "Residuals")
```



From the plot, we see that every predictor seems to be an underestimation, but not too much, indicating possibly better model with non-linear models.

In conclusion, stepwise and Lasso give us almost the same final results.The difference is that stepwise selection excludes “cbwdSE” in the results. Both methods shows that IS should be the last predictor to incorporate into the model. This makes sense because unlike Minnesota, it snows only about 10 days per year in Beijing. The best predictor in Lasso is cbwdNW, which also makes sense because seasonal tornados from northwest(inner Mongolia) can cause PM2.5 in Beijing to rise and persist for a very long time.

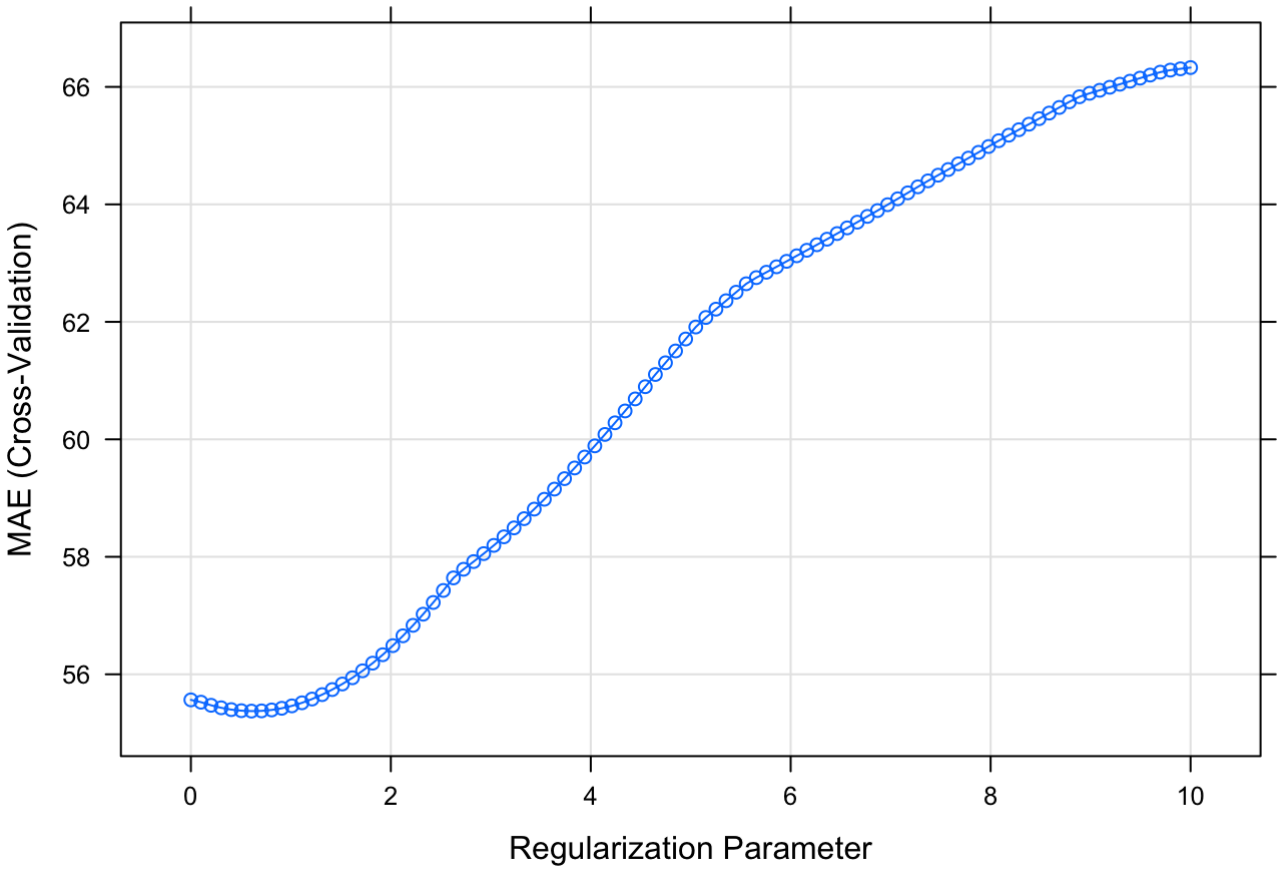
Investigation 2: Non-linear model

Splines with Lasso

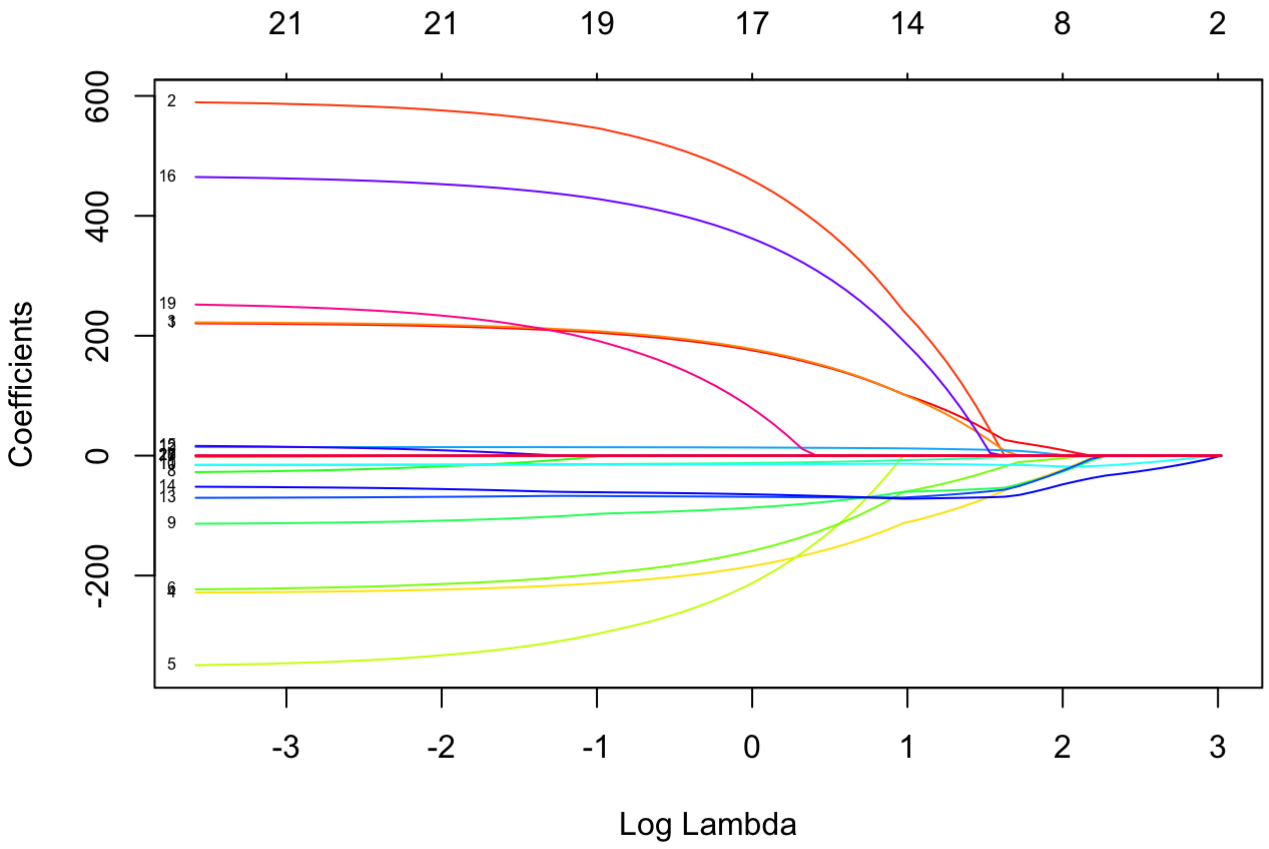
```
set.seed(253)
splinelasso_mod <- train(
    pm2.5 ~ ns(DEWP, 3) + ns(TEMP, 3) + ns(PRES, 3) +cbwd + ns(Iws, 3) + ns(Ir, 3)+ ns(I
s, 3),
    data = PM2.5,
    method = "glmnet",
    trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"),
    tuneGrid = data.frame(alpha = 1, lambda = seq(0, 10, length.out = 100)),
    metric = "MAE",
    na.action = na.omit
)
```

Plot CV error estimates for the different models to find the best model

```
plot(splinelasso_mod)
```




```
plot(splinelasso_mod$finalModel, xvar = "lambda", label = TRUE, col = rainbow(20))
```



Displaying codebook for which variables the numbers correspond to

```
rownames(splinelasso_mod$finalModel$beta)
```

```
##  [1] "ns(DEWP, 3)1" "ns(DEWP, 3)2" "ns(DEWP, 3)3" "ns(TEMP, 3)1" "ns(TEMP, 3)2"
##  [6] "ns(TEMP, 3)3" "ns(PRES, 3)1" "ns(PRES, 3)2" "ns(PRES, 3)3" "cbwdNE"
## [11] "cbwdNW"       "cbwdSE"       "ns(Iws, 3)1"  "ns(Iws, 3)2"  "ns(Iws, 3)3"
## [16] "ns(Ir, 3)1"   "ns(Ir, 3)2"   "ns(Ir, 3)3"   "ns(Is, 3)1"   "ns(Is, 3)2"
## [21] "ns(Is, 3)3"
```

```
splinelasso_mod$bestTune
```

```
##    alpha   lambda
## 14     1 1.313131
```

```
coef(splinelasso_mod$finalModel, lasso_mod$bestTune$lambda)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)  -1.175445e+02
## ns(DEWP, 3)1  1.297579e+02
## ns(DEWP, 3)2  3.231508e+02
## ns(DEWP, 3)3  1.305414e+02
## ns(TEMP, 3)1 -1.394703e+02
## ns(TEMP, 3)2 -7.954334e+01
## ns(TEMP, 3)3 -9.752722e+01
## ns(PRES, 3)1  .
## ns(PRES, 3)2  .
## ns(PRES, 3)3 -7.043338e+01
## cbwdNE       -9.384317e+00
## cbwdNW       -1.380190e+01
## cbwdSE        1.264129e+01
## ns(Iws, 3)1  -6.973457e+01
## ns(Iws, 3)2  -6.927128e+01
## ns(Iws, 3)3   .
## ns(Ir, 3)1    2.563946e+02
## ns(Ir, 3)2    7.557278e-08
## ns(Ir, 3)3   -1.513960e-08
## ns(Is, 3)1    .
## ns(Is, 3)2    .
## ns(Is, 3)3    .
```

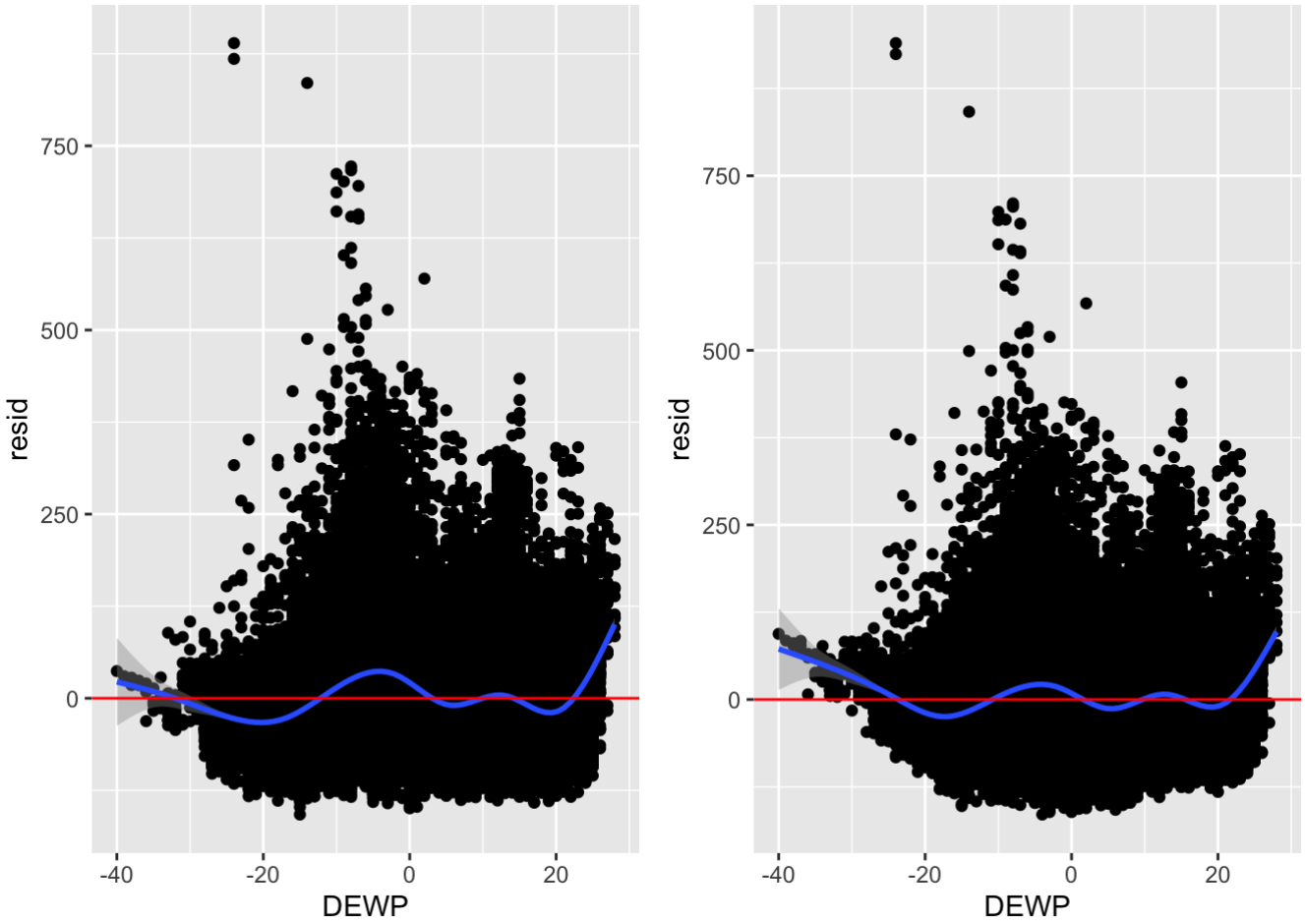
Make the residual plots to see if Splines improve the model

```
lasso_mod_data <- PM2.5 %>%
    mutate(
        pred = predict(lasso_mod, newdata = PM2.5),
        resid = pm2.5 - pred
    )

p1 <- ggplot(lasso_mod_data, aes(x = DEWP, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")

p2 <- ggplot(lasso_mod_data, aes(x = TEMP, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")

p3 <- ggplot(lasso_mod_data, aes(x = PRES, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")

p4 <- ggplot(lasso_mod_data, aes(x = cbwd, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
p5 <- ggplot(lasso_mod_data, aes(x = Iws, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
p6 <- ggplot(lasso_mod_data, aes(x = Ir, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
p7 <- ggplot(lasso_mod_data, aes(x = Is, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
```

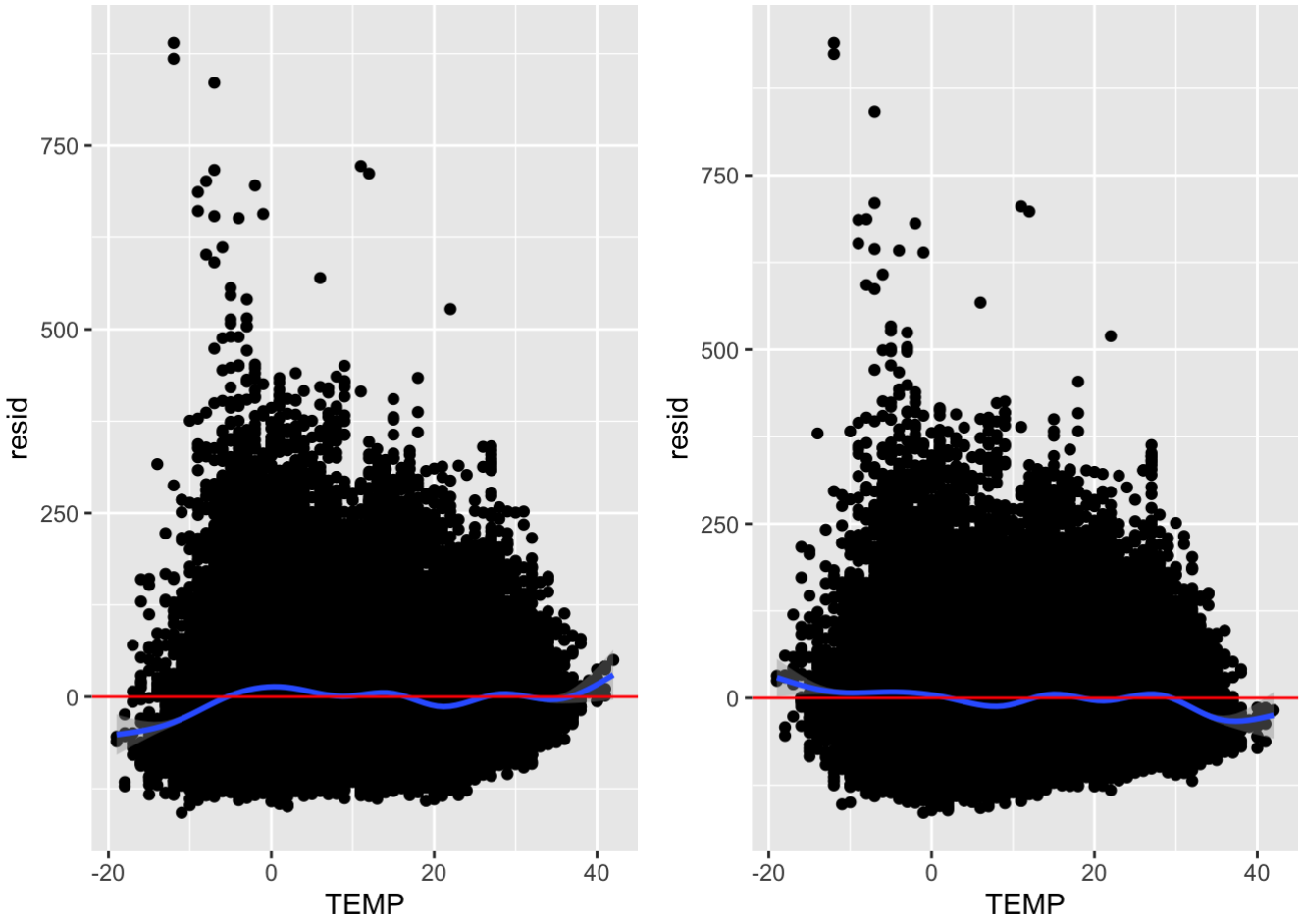
```
splinelasso_mod_data <- PM2.5 %>%
    mutate(
        pred = predict(splinelasso_mod, newdata = PM2.5),
        resid = pm2.5 - pred
    )

pl1 <- ggplot(splinelasso_mod_data, aes(x = DEWP, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")

pl2 <- ggplot(splinelasso_mod_data, aes(x = TEMP, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")

pl3 <- ggplot(splinelasso_mod_data, aes(x = PRES, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")

pl4 <- ggplot(splinelasso_mod_data, aes(x = cbwd, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
pl5 <- ggplot(splinelasso_mod_data, aes(x = Iws, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
pl6 <- ggplot(splinelasso_mod_data, aes(x = Ir, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
pl7 <- ggplot(splinelasso_mod_data, aes(x = Is, y = resid)) +
    geom_point() +
    geom_smooth() +
    geom_hline(yintercept = 0, color = "red")
```

Comparisons

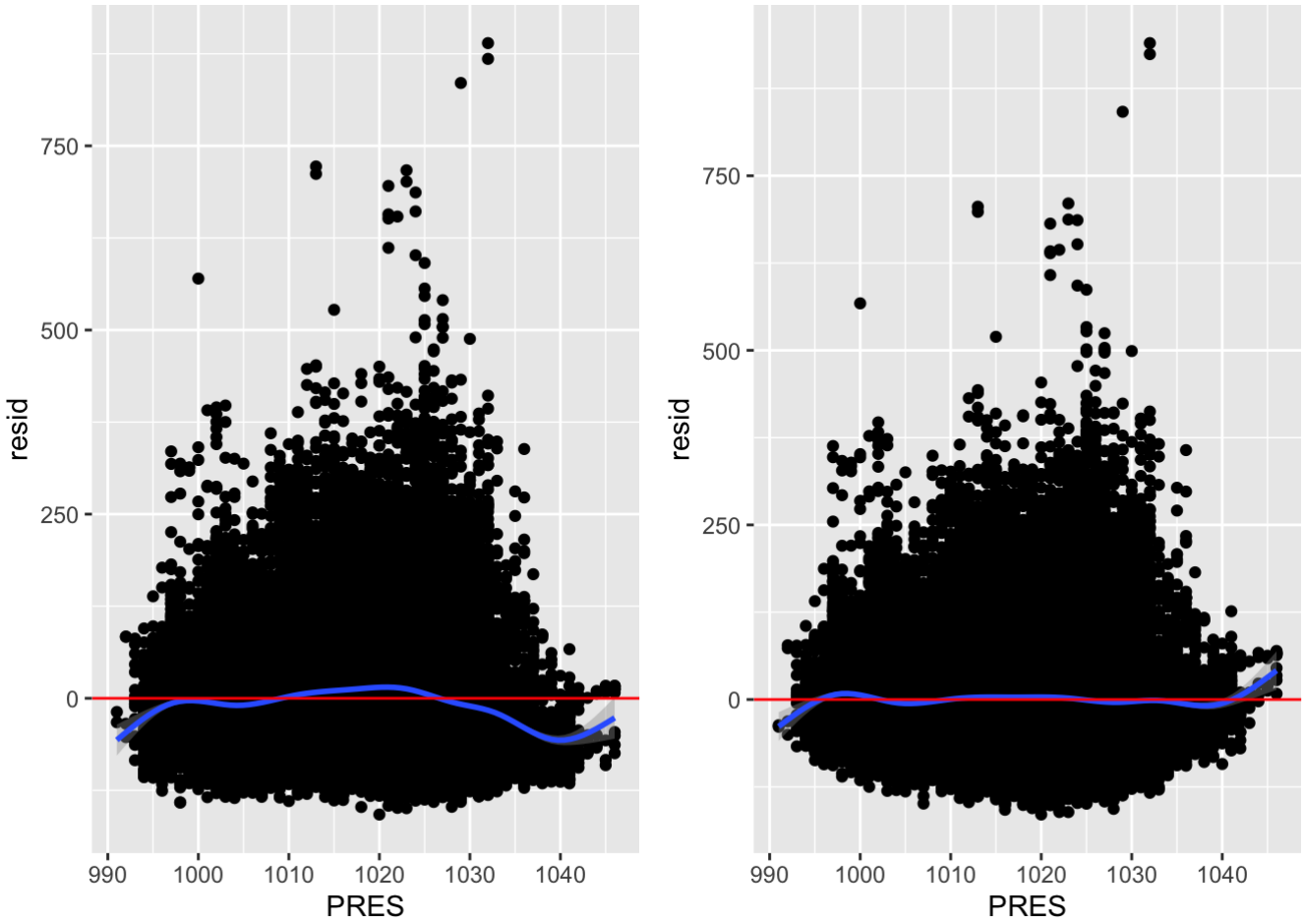
```
grid.arrange(p1, pl1, nrow = 1, ncol = 2)
```



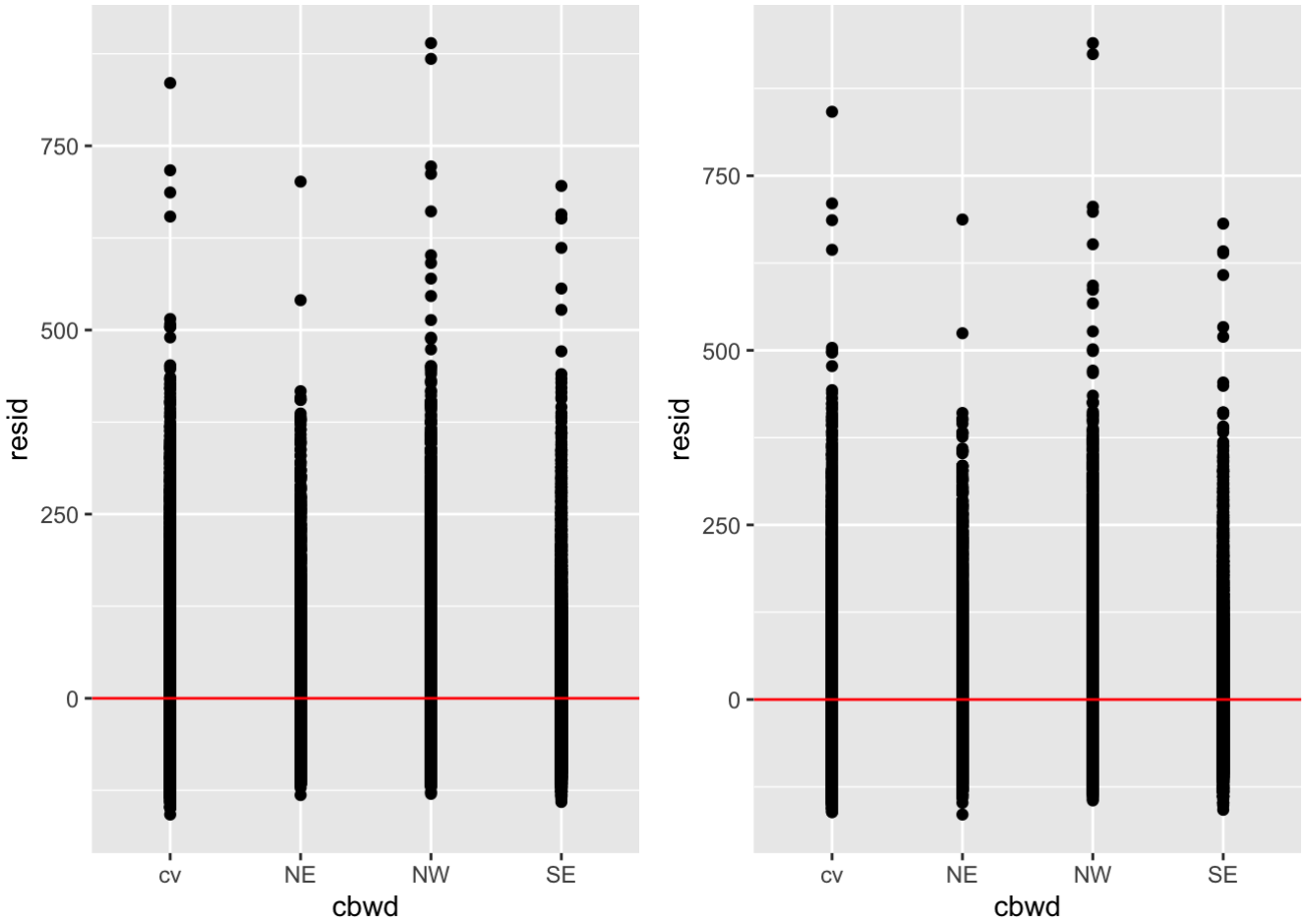

```
grid.arrange(p2, pl2, nrow = 1, ncol = 2)
```



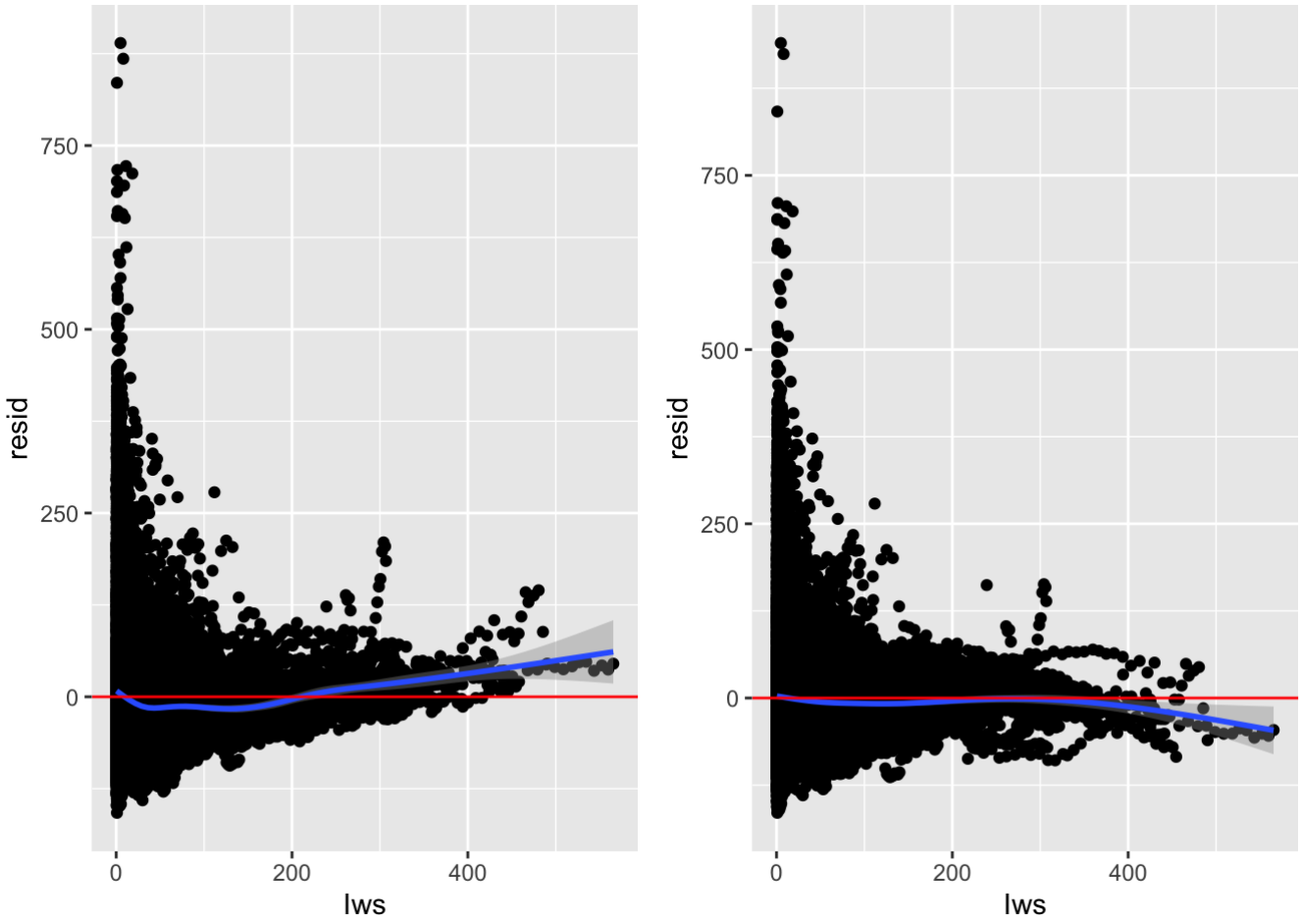

```
grid.arrange(p3, pl3, nrow = 1, ncol = 2)
```



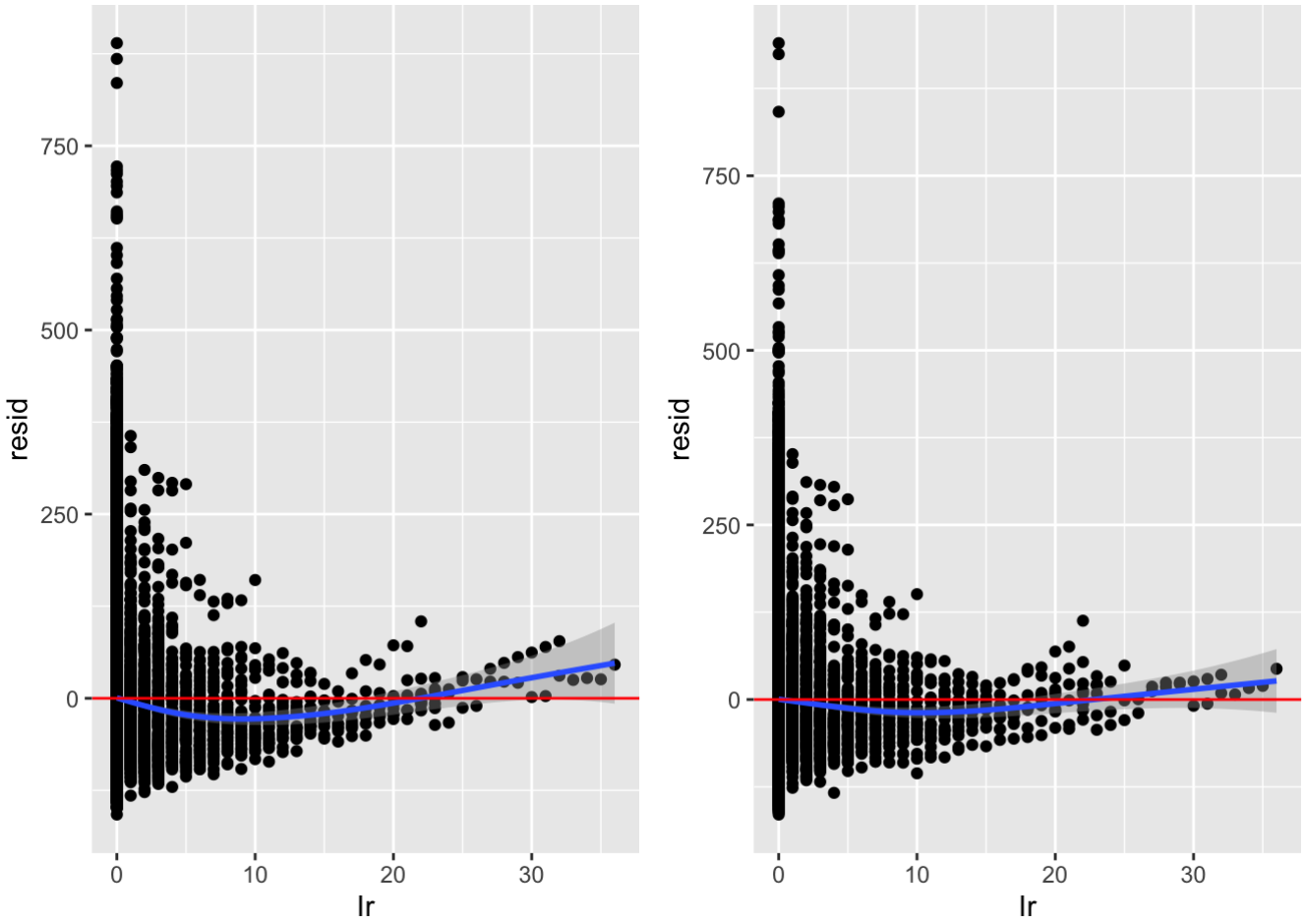

```
grid.arrange(p4, pl4, nrow = 1, ncol = 2)
```



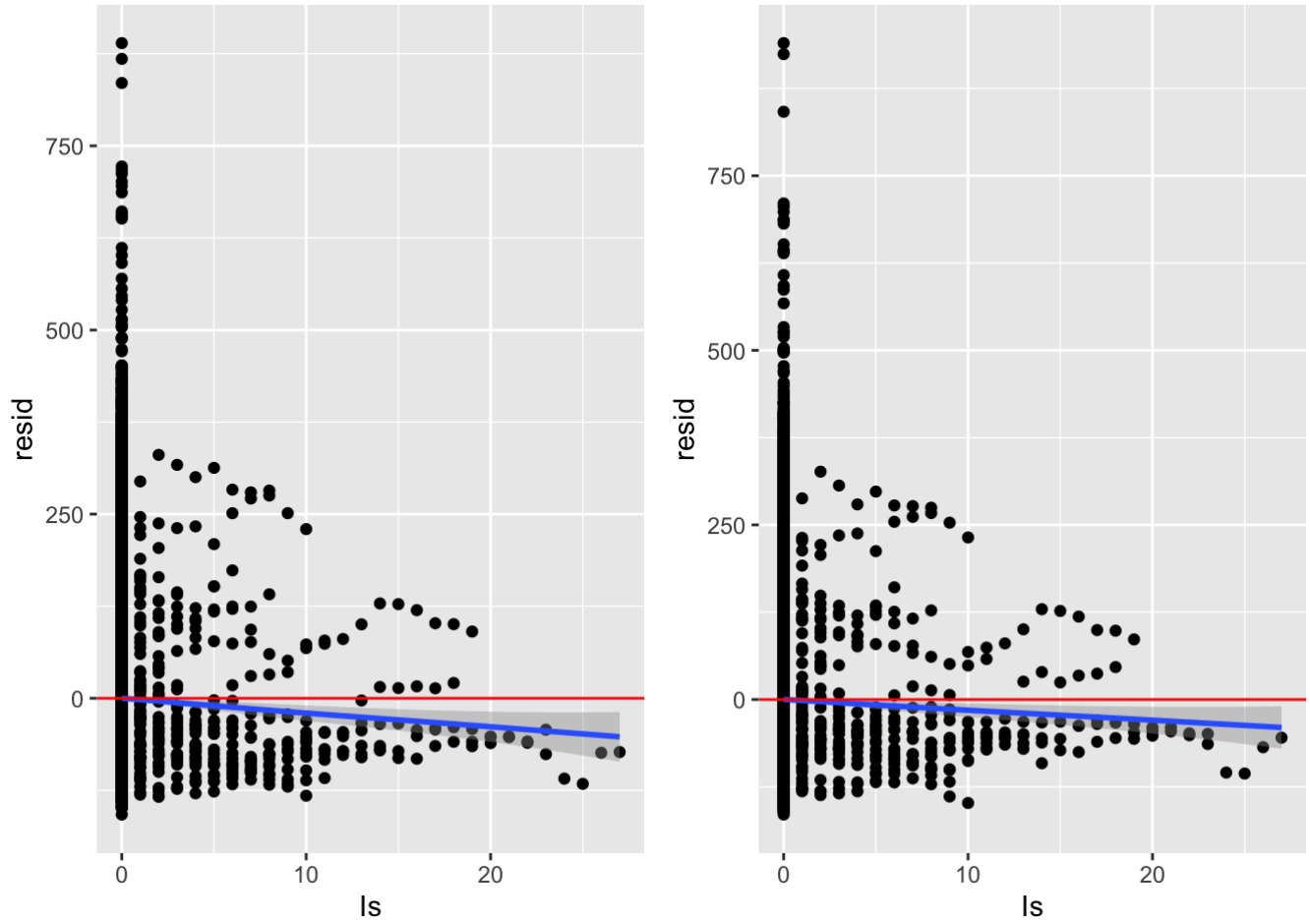

```
grid.arrange(p5, pl5, nrow = 1, ncol = 2)
```




```
grid.arrange(p6, pl6, nrow = 1, ncol = 2)
```




```
grid.arrange(p7, pl7, nrow = 1, ncol = 2)
```



Using splines for non-linearity does make our curves smoother. And compared to without using splines, the model here has one less predictor, namely LS.

Splines with subset selection

####Train the model

```
spline_back_step_mod <- train(
    pm2.5  ~  cbwd + ns(TEMP, df=3)+ ns(PRES, df=3)+ ns(Iws, df=3)+ ns(Is, df=3)+ ns(Ir,
df=3)+ ns(DEWP, df=3),
    data = PM2.5,
    method = "leapBackward",
    tuneGrid = data.frame(nvmax = 1:18),
    trControl = trainControl(method = "cv", number = 10),
    metric = "MAE",
    na.action = na.omit
)
```

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

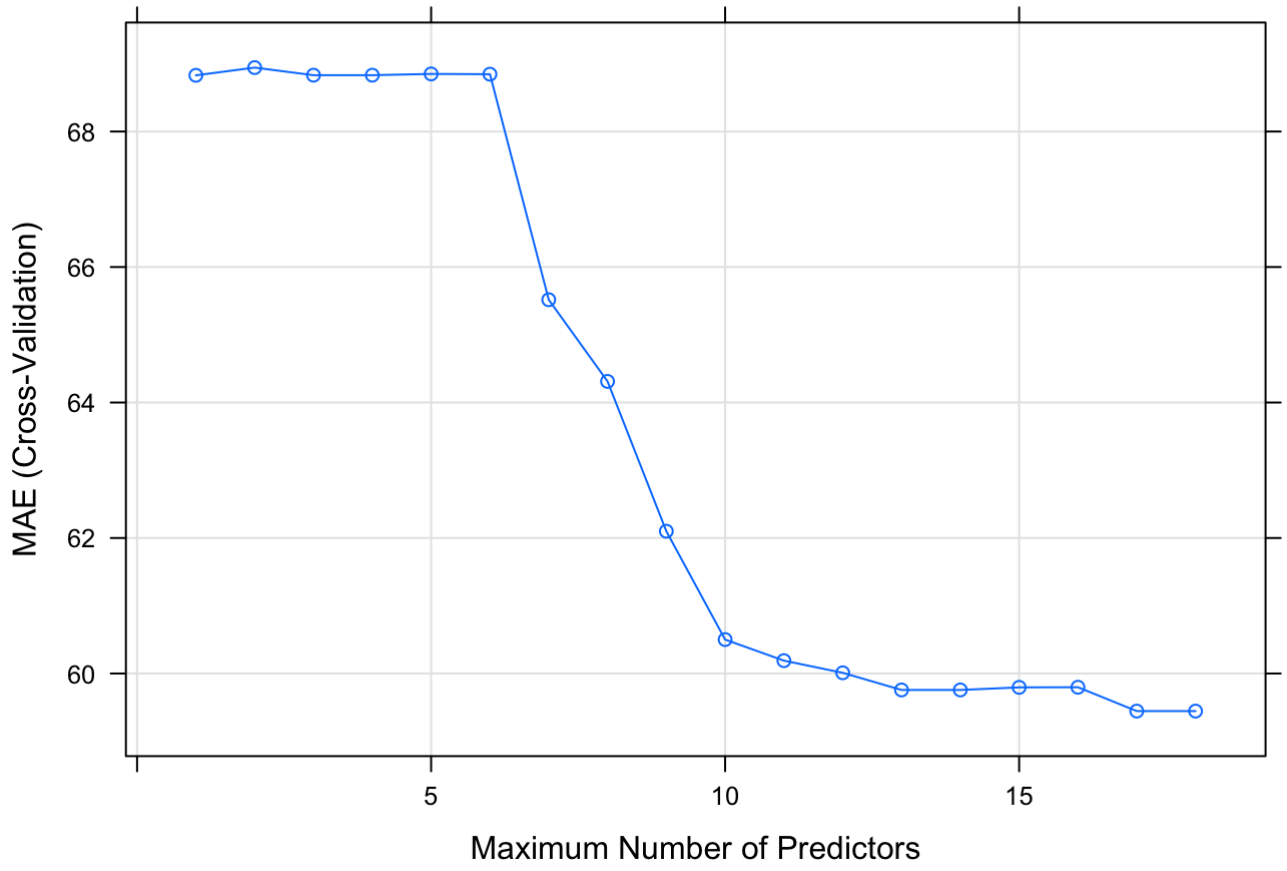
```
summary(spline_back_step_mod)
```

```
## Subset selection object
## 21 Variables  (and intercept)
##                   Forced in Forced out
## cbwdNE                FALSE      FALSE
## cbwdNW                FALSE      FALSE
## cbwdSE                FALSE      FALSE
## ns(TEMP, df = 3)1     FALSE      FALSE
## ns(TEMP, df = 3)2     FALSE      FALSE
## ns(TEMP, df = 3)3     FALSE      FALSE
## ns(PRES, df = 3)1     FALSE      FALSE
## ns(PRES, df = 3)2     FALSE      FALSE
## ns(PRES, df = 3)3     FALSE      FALSE
## ns(Iws, df = 3)1      FALSE      FALSE
## ns(Iws, df = 3)2      FALSE      FALSE
## ns(Iws, df = 3)3      FALSE      FALSE
## ns(Is, df = 3)1       FALSE      FALSE
## ns(Ir, df = 3)1       FALSE      FALSE
## ns(DEWP, df = 3)1     FALSE      FALSE
## ns(DEWP, df = 3)2     FALSE      FALSE
## ns(DEWP, df = 3)3     FALSE      FALSE
## ns(Is, df = 3)2       FALSE      FALSE
## ns(Is, df = 3)3       FALSE      FALSE
## ns(Ir, df = 3)2       FALSE      FALSE
## ns(Ir, df = 3)3       FALSE      FALSE
## 1 subsets of each size up to 17
## Selection Algorithm: backward
##           cbwdNE cbwdNW cbwdSE ns(TEMP, df = 3)1 ns(TEMP, df = 3)2
## 1  ( 1 )  " "    " "    " "    " "                " "
## 2  ( 1 )  " "    " "    " "    "*"                " "
## 3  ( 1 )  " "    " "    " "    "*"                " "
## 4  ( 1 )  " "    " "    " "    "*"                " "
## 5  ( 1 )  " "    " "    " "    "*"                " "
## 6  ( 1 )  " "    " "    " "    "*"                "*"
## 7  ( 1 )  " "    " "    " "    "*"                "*"
## 8  ( 1 )  " "    " "    "*"    "*"                "*"
## 9  ( 1 )  " "    " "    "*"    "*"                "*"
## 10  ( 1 ) " "    " "    "*"    "*"                "*"
## 11  ( 1 ) " "    " "    "*"    "*"                "*"
## 12  ( 1 ) " "    "*"    "*"    "*"                "*"
## 13  ( 1 ) "*"    "*"    "*"    "*"                "*"
## 14  ( 1 ) "*"    "*"    "*"    "*"                "*"
## 15  ( 1 ) "*"    "*"    "*"    "*"                "*"
## 16  ( 1 ) "*"    "*"    "*"    "*"                "*"
## 17  ( 1 ) "*"    "*"    "*"    "*"                "*"
##           ns(TEMP, df = 3)3 ns(PRES, df = 3)1 ns(PRES, df = 3)2
## 1  ( 1 )  " "               " "               " "
## 2  ( 1 )  " "               " "               " "
## 3  ( 1 )  " "               " "               " "
## 4  ( 1 )  " "               " "               " "
## 5  ( 1 )  "*"               " "               " "
## 6  ( 1 )  "*"               " "               " "
## 7  ( 1 )  "*"               " "               " "
## 8  ( 1 )  "*"               " "               " "
```

```
## 9  ( 1 )  "*"                " "                 " "
## 10  ( 1 ) "*"                " "                 " "
## 11  ( 1 ) "*"                " "                 " "
## 12  ( 1 ) "*"                " "                 " "
## 13  ( 1 ) "*"                " "                 " "
## 14  ( 1 ) "*"                " "                 " "
## 15  ( 1 ) "*"                " "                 "*"
## 16  ( 1 ) "*"                " "                 "*"
## 17  ( 1 ) "*"                "*"                 "*"
##           ns(PRES, df = 3)3 ns(Iws, df = 3)1 ns(Iws, df = 3)2 ns(Iws, df = 3)3
## 1  ( 1 )  " "                " "                 " "                 " "
## 2  ( 1 )  " "                " "                 " "                 " "
## 3  ( 1 )  " "                " "                 " "                 " "
## 4  ( 1 )  " "                " "                 " "                 " "
## 5  ( 1 )  " "                " "                 " "                 " "
## 6  ( 1 )  " "                " "                 " "                 " "
## 7  ( 1 )  "*"                " "                 " "                 " "
## 8  ( 1 )  "*"                " "                 " "                 " "
## 9  ( 1 )  "*"                " "                 "*"                 " "
## 10  ( 1 ) "*"                "*"                 "*"                 " "
## 11  ( 1 ) "*"                "*"                 "*"                 " "
## 12  ( 1 ) "*"                "*"                 "*"                 " "
## 13  ( 1 ) "*"                "*"                 "*"                 " "
## 14  ( 1 ) "*"                "*"                 "*"                 " "
## 15  ( 1 ) "*"                "*"                 "*"                 " "
## 16  ( 1 ) "*"                "*"                 "*"                 "*"
## 17  ( 1 ) "*"                "*"                 "*"                 "*"
##           ns(Is, df = 3)1 ns(Is, df = 3)2 ns(Is, df = 3)3 ns(Ir, df = 3)1
## 1  ( 1 )  " "               " "                 " "                 " "
## 2  ( 1 )  " "               " "                 " "                 " "
## 3  ( 1 )  " "               " "                 " "                 " "
## 4  ( 1 )  " "               " "                 " "                 " "
## 5  ( 1 )  " "               " "                 " "                 " "
## 6  ( 1 )  " "               " "                 " "                 " "
## 7  ( 1 )  " "               " "                 " "                 " "
## 8  ( 1 )  " "               " "                 " "                 " "
## 9  ( 1 )  " "               " "                 " "                 " "
## 10  ( 1 ) " "               " "                 " "                 " "
## 11  ( 1 ) " "               " "                 " "                 "*"
## 12  ( 1 ) " "               " "                 " "                 "*"
## 13  ( 1 ) " "               " "                 " "                 "*"
## 14  ( 1 ) "*"               " "                 " "                 "*"
## 15  ( 1 ) "*"               " "                 " "                 "*"
## 16  ( 1 ) "*"               " "                 " "                 "*"
## 17  ( 1 ) "*"               " "                 " "                 "*"
##           ns(Ir, df = 3)2 ns(Ir, df = 3)3 ns(DEWP, df = 3)1 ns(DEWP, df = 3)2
## 1  ( 1 )  " "               " "                 "*"                 " "
## 2  ( 1 )  " "               " "                 "*"                 " "
## 3  ( 1 )  " "               " "                 "*"                 " "
## 4  ( 1 )  " "               " "                 "*"                 "*"
## 5  ( 1 )  " "               " "                 "*"                 "*"
## 6  ( 1 )  " "               " "                 "*"                 "*"
## 7  ( 1 )  " "               " "                 "*"                 "*"
## 8  ( 1 )  " "               " "                 "*"                 "*"
```

```
## 9  ( 1 )  " "               " "               "*"               "*"
## 10  ( 1 ) " "               " "               "*"               "*"
## 11  ( 1 ) " "               " "               "*"               "*"
## 12  ( 1 ) " "               " "               "*"               "*"
## 13  ( 1 ) " "               " "               "*"               "*"
## 14  ( 1 ) " "               " "               "*"               "*"
## 15  ( 1 ) " "               " "               "*"               "*"
## 16  ( 1 ) " "               " "               "*"               "*"
## 17  ( 1 ) " "               " "               "*"               "*"
##           ns(DEWP, df = 3)3
## 1  ( 1 )  " "
## 2  ( 1 )  " "
## 3  ( 1 )  "*"
## 4  ( 1 )  "*"
## 5  ( 1 )  "*"
## 6  ( 1 )  "*"
## 7  ( 1 )  "*"
## 8  ( 1 )  "*"
## 9  ( 1 )  "*"
## 10  ( 1 ) "*"
## 11  ( 1 ) "*"
## 12  ( 1 ) "*"
## 13  ( 1 ) "*"
## 14  ( 1 ) "*"
## 15  ( 1 ) "*"
## 16  ( 1 ) "*"
## 17  ( 1 ) "*"
```

```
plot(spline_back_step_mod)
```



Similar to previous models, the best model here has a tuning parameter of nvmax=17, which is using all predictors. However, MAE starts to decrease more slowly after 10 predictors, indicating a possible more reasonable model with 10 predictors.

```
spline_back_step_mod$bestTune
```

```
##    nvmax
## 17    17
```

```
coef(spline_back_step_mod$finalModel, id = spline_back_step_mod$bestTune$nvmax)
```

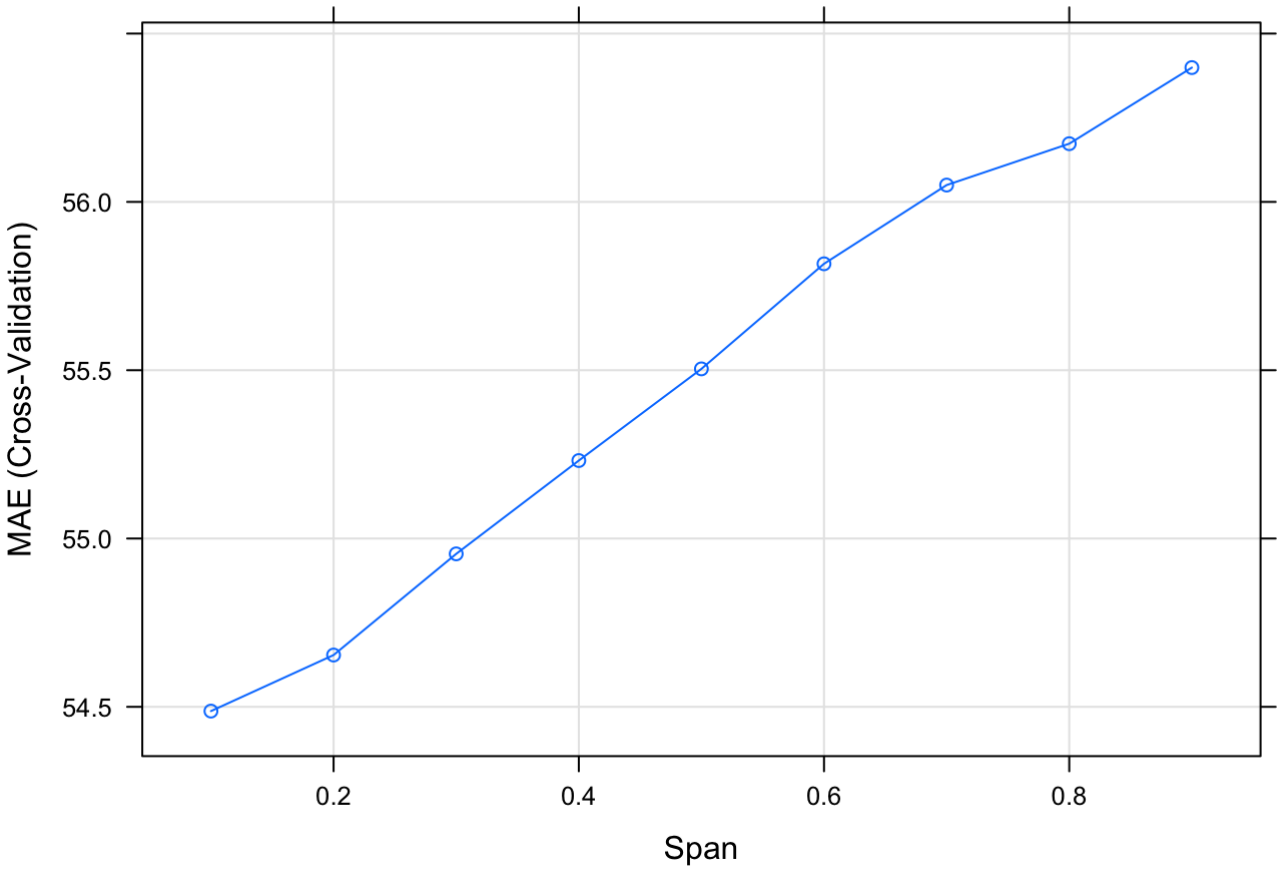
```
##       (Intercept)             cbwdNE             cbwdNW             cbwdSE
##          4.194272         -19.755942         -18.835646          22.080708
## ns(TEMP, df = 3)1 ns(TEMP, df = 3)2 ns(TEMP, df = 3)3 ns(PRES, df = 3)1
##       -135.456764       -219.367878       -142.824739        -42.591279
## ns(PRES, df = 3)2 ns(PRES, df = 3)3  ns(Iws, df = 3)1  ns(Iws, df = 3)2
##       -105.175256       -189.852351       -138.019515       -119.430012
##  ns(Iws, df = 3)3   ns(Is, df = 3)1 ns(DEWP, df = 3)2   ns(Is, df = 3)3
##         -7.332031         -32.022129         -29.679334           0.000000
##   ns(Ir, df = 3)2   ns(Ir, df = 3)3
##         593.745224           0.000000
```

GAM

```
gam_mod <- train(
    pm2.5 ~ DEWP+TEMP+PRES+Iws+Is+Ir+cbwd,
    data=PM2.5,
    method = "gamLoess",
    tuneGrid = data.frame(degree = 1, span = seq(0.1, 0.9, by = 0.1)),
    trControl = trainControl(method = "cv", number = 10, selectionFunction = 'best'),
    metric = "MAE",
    na.action = na.omit,
    warning = FALSE
)
```

We see that our model has the smallest MAE when our tuning parameter span=0.1, where 10% of data are used for this local fit.

```
# Plot CV-estimated test performance versus the tuning parameter
plot(gam_mod)
```




```
# Identify which tuning parameter is "best"
gam_mod$bestTune
```

```
##   span degree
## 1  0.1      1
```

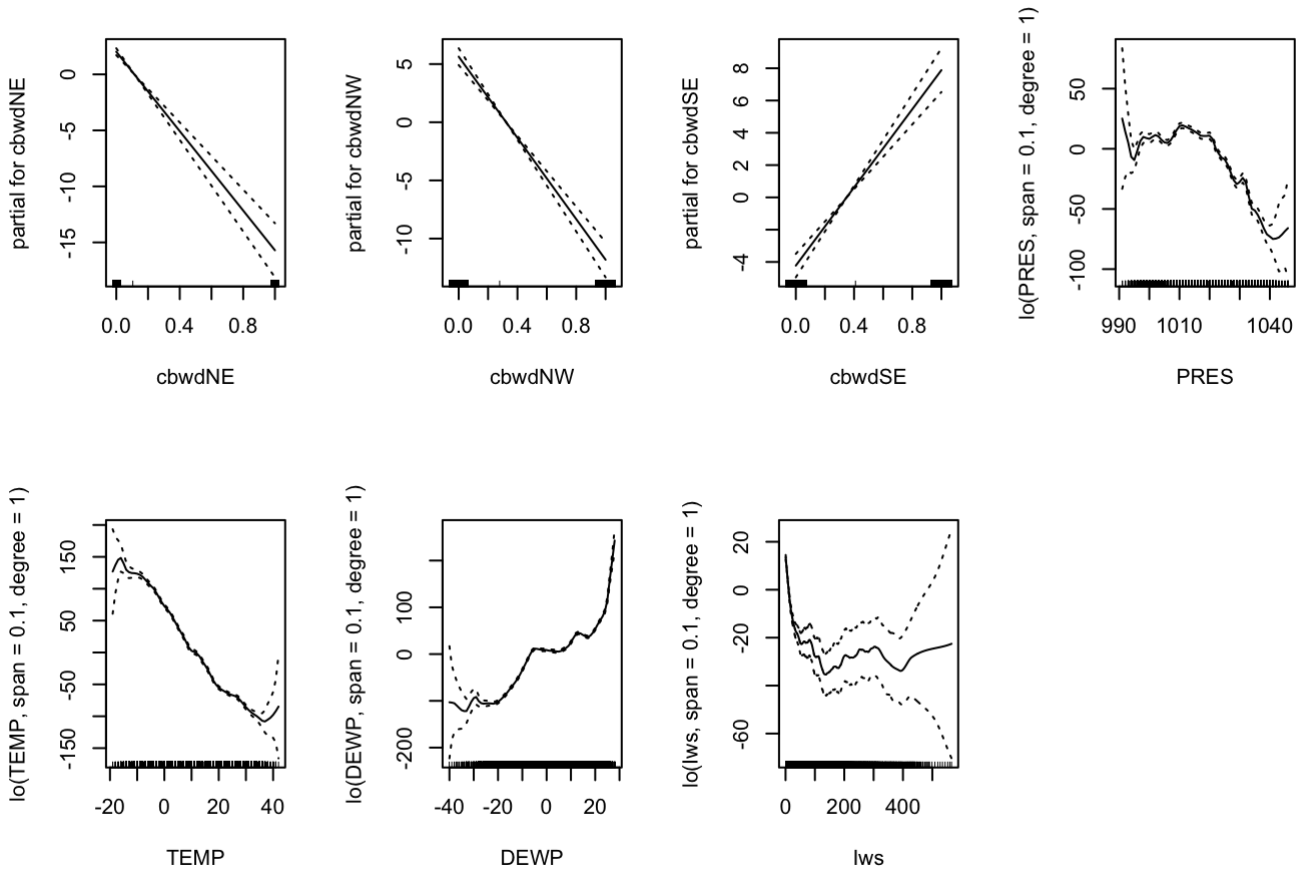
```
# CV metrics for each model
gam_mod$results
```

```
##   degree span     RMSE  Rsquared      MAE   RMSESD  RsquaredSD     MAESD
## 1      1  0.1 75.61126 0.3253993 54.48736 1.296879 0.014097371 0.7178142
## 2      1  0.2 75.88823 0.3205411 54.65384 1.291930 0.013060282 0.7180392
## 3      1  0.3 76.29304 0.3134236 54.95442 1.297135 0.012001092 0.7126157
## 4      1  0.4 76.63212 0.3073194 55.23160 1.312998 0.011474189 0.7183224
## 5      1  0.5 76.96754 0.3012231 55.50400 1.327008 0.010846454 0.7311092
## 6      1  0.6 77.27230 0.2955935 55.81614 1.323666 0.010529235 0.7161367
## 7      1  0.7 77.58978 0.2897703 56.05000 1.318405 0.010274962 0.7128413
## 8      1  0.8 77.88146 0.2844987 56.17291 1.325822 0.009987906 0.7188053
## 9      1  0.9 78.26642 0.2774046 56.39896 1.336003 0.009950151 0.7189023
```

```
# CV metrics for just the "best" model
gam_mod$results %>%
    filter(span==gam_mod$bestTune$span)
```

```
##   degree span     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1      1  0.1 75.61126 0.3253993 54.48736 1.296879 0.01409737 0.7178142
```

```
par(mfrow = c(2,4)) # Sets up a grid of plots
plot(gam_mod$finalModel, se = TRUE) # Dashed lines are +/- 2 SEs
```



The GAM model is the best model so far. For categorical variables cbwdNE, cbwdNW, cbwdSE, the curves are linear. Specifically, if the wind comes from NE and NW, the pm2.5 concentration will increase. And a decrease can be seen if there is wind from SE direction. Besides these categorical variables, other quantative ones are not linear: PRES, TEMP, Lws are negatively related to pm2.5 and DEWP is positively related. These results agree with results from other methods. From the error metrics, we see that this GAM model is better than others for its smallest MAE value.

Summary of investigations

Striving for predictive accuracy, we select gam_mod as our best model, since it has the lowest test error(MAE). This model is relatively interpretable, too. All the predictors are common weather measures in Beijing, and the model shows that they all relate to pm2.5 in some way.

Societal impact

This hourly data set contains the PM2.5 data of US Embassy in Beijing. Meanwhile, meteorological data from Beijing Capital International Airport are also included. So the data is actually collected at two different places, which may lead to some corresponding issues considering the size of Beijing city. Also, the MAE has a value about roughlt 50ug/m^3. It's not that our estimation is 50% different from the true value. The range of our response variable pm2.5 is from 0 to 994. So our estimation is roughly 50/1000 = 5% from the true pm2.5.

Classification model

Our main goal will be to use the predictor we have to predict whether or not a particular day in Beijing is polluted or not.

Some Data rendering. Here, I define a polluted day to have a pm2.5 concentration larger than 100ug/m^3 and not polluted if the pm2.5 concentration is less than 100ug/m^3. A also define a season categorical from the month category. So January Febuary and March are spring and so on.

```
#Change quantative variables into categorical ones
pm2.5Level <- cut(PM2.5$pm2.5, breaks = c(0,50,100,200,300,1000), labels = c("great air
 quality","not bad air quality", "slight pollution","medium pollution","heavy pollution"
))
PM2.5$pm2.5Level <- pm2.5Level

season <- cut(PM2.5$month, breaks = c(1,4,7,10,13), labels = c("Spring","Summer","Fall",
"Winter"))
PM2.5$season <- season

pollution <- cut(PM2.5$pm2.5, breaks = c(0,100,1000), labels = c("no pollution","polluti
on"))
PM2.5$pollution <- pollution

PM2.5 <- PM2.5 %>%
    mutate(pollution = ifelse(pollution=="pollution", "pollution", "not_pollution"))
```

Logistic Regression

Our goal is to fit a logistic regression model with different predictors. The variables I choose here are temperature, wind direction, wind speed, and season.

```
set.seed(253)
logistic_mod <- train(
    pollution ~ TEMP+cbwd+Iws+season,
    data = PM2.5,
    method = "glm",
    family = "binomial",
    trControl = trainControl(method = "cv", number = 10),
    metric = "Accuracy",
    na.action = na.omit
)
```

```
summary(logistic_mod)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4683  -1.0126  -0.7181   1.2244   3.4797
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.2070331  0.0307144   6.741 1.58e-11 ***
## TEMP         -0.0175772  0.0016691 -10.531  < 2e-16 ***
## cbwdNE       -0.6150215  0.0398591 -15.430  < 2e-16 ***
## cbwdNW       -0.7518740  0.0337986 -22.246  < 2e-16 ***
## cbwdSE        0.1919430  0.0301760   6.361 2.01e-10 ***
## Iws          -0.0184003  0.0006064 -30.344  < 2e-16 ***
## seasonSummer -0.0117056  0.0424409  -0.276  0.78269
## seasonFall   -0.0736020  0.0379458  -1.940  0.05242 .
## seasonWinter  0.1035901  0.0353322   2.932  0.00337 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 50339  on 38283  degrees of freedom
## Residual deviance: 46576  on 38275  degrees of freedom
## AIC: 46594
##
## Number of Fisher Scoring iterations: 5
```

Making predictions:

If we consider a day where the Temperature is 0 degrees, 1m/s wind is blowing from southeast and it's winter time.

```
#odds:
exp(0.2070331-0.0175772+0.1919430-0.0184003+0.1035901)
```

```
## [1] 1.594545
```

```
#Probability
1.594545/(1+1.594545)
```

```
## [1] 0.614576
```

We can also use the predict() function to arrive at the same result:

```
predict(logistic_mod, newdata = data.frame(TEMP = 1, cbwd="SE", Iws = 1, season = "Winte
r"), type = "prob")
```

```
##   not_pollution pollution
## 1      0.385424  0.614576
```

From our soft prediction, we see a 61.5% of the chance that the day is polluted, given the meteorological data we assumed. If we use a probability threshold of 50%, we would make a hard prediction of this day being polluted. So we can warn people to prepare and wear a mask on this day.

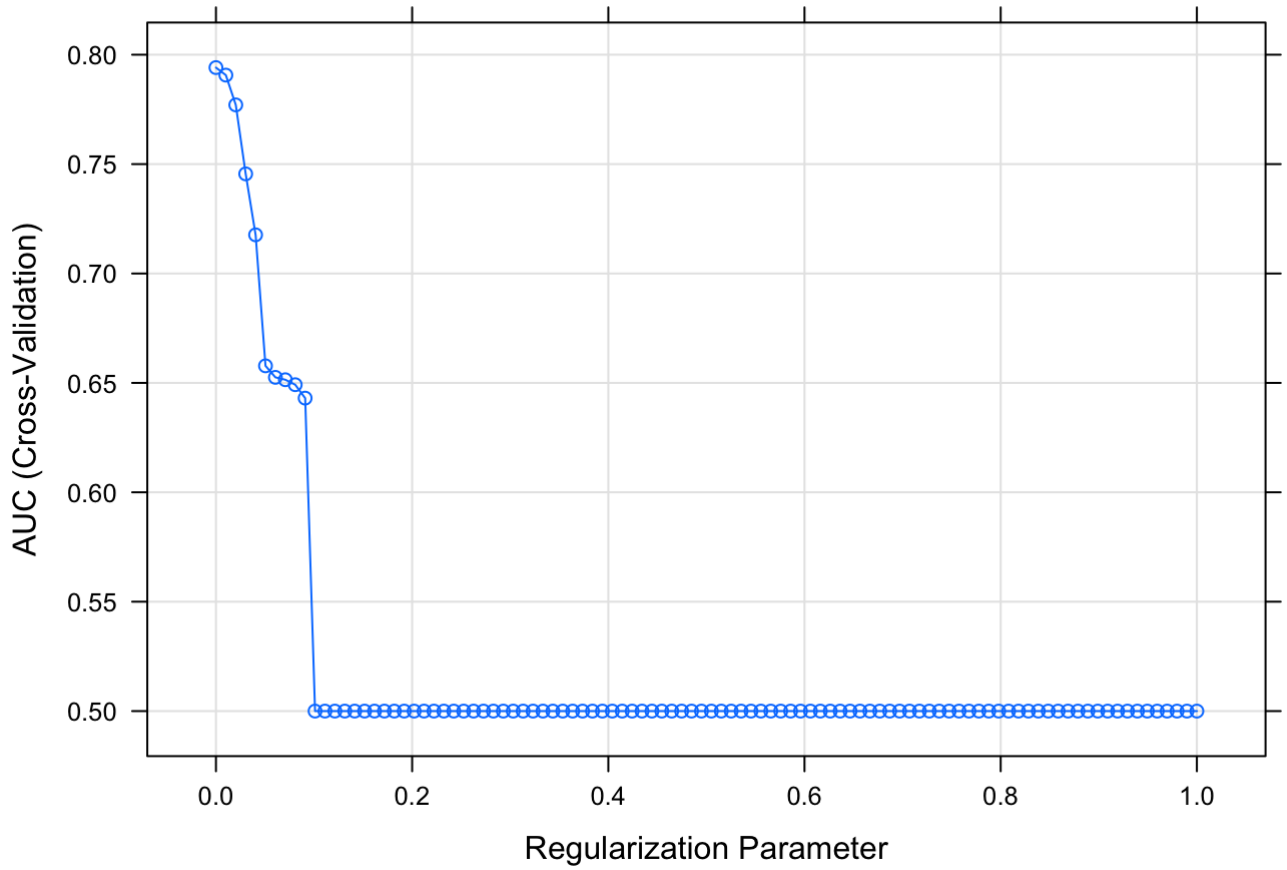
We also want to implement LASSO logistic regression:

```
twoClassSummaryCustom <- function (data, lev = NULL, model = NULL) {
    if (length(lev) > 2) {
        stop(paste("Your outcome has", length(lev), "levels. The twoClassSummary() funct
ion isn't appropriate."))
    }
    caret:::requireNamespaceQuietStop("pROC")
    if (!all(levels(data[, "pred"]) == lev)) {
        stop("levels of observed and predicted data do not match")
    }
    rocObject <- try(pROC::roc(data$obs, data[, lev[1]], direction = ">",
        quiet = TRUE), silent = TRUE)
    rocAUC <- if (inherits(rocObject, "try-error"))
        NA
    else rocObject$auc
    out <- c(rocAUC, sensitivity(data[, "pred"], data[, "obs"],
        lev[1]), specificity(data[, "pred"], data[, "obs"], lev[2]))
    out2 <- postResample(data[, "pred"], data[, "obs"])
    out <- c(out, out2[1])
    names(out) <- c("AUC", "Sens", "Spec", "Accuracy")
    out
}
```

```
set.seed(253)
lasso_logistic_mod <- train(
    pollution ~ Is+Ir+DEWP+PRES+TEMP+cbwd+Iws+season,
    data = PM2.5,
    method = "glmnet",
    family = "binomial",
    tuneGrid = data.frame(alpha = 1, lambda = seq(0, 1, length.out = 100)),
    trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE", cl
assProbs = TRUE, summaryFunction = twoClassSummaryCustom),
    metric = "AUC",
    na.action = na.omit
)
```

The plot shows our AUC at different parameter values. A higher AUC is associated with a better model.

```
plot(lasso_logistic_mod)
```



The best model is when lambda equals 0, when no penalty is applied.

```
lasso_logistic_mod$bestTune
```

```
##   alpha lambda
## 1     1      0
```

We can also see the metrics associated with this best model:

```
lasso_logistic_mod$results %>%
    filter(lambda==lasso_logistic_mod$bestTune$lambda)
```

```
##   alpha lambda      AUC      Sens      Spec  Accuracy       AUCSD      SensSD
## 1     1      0 0.794112 0.8090145 0.5704647 0.7214242 0.009006999 0.008045849
##       SpecSD  AccuracySD
## 1 0.01734545 0.009612734
```

Compute the NIR:

```
PM2.5 %>%
    count(pollution)
```

```
##       pollution     n
## 1 not_pollution 26388
## 2     pollution 15367
## 3          <NA>     2
```

```
26388/(26388+15367)
```

```
## [1] 0.6319722
```

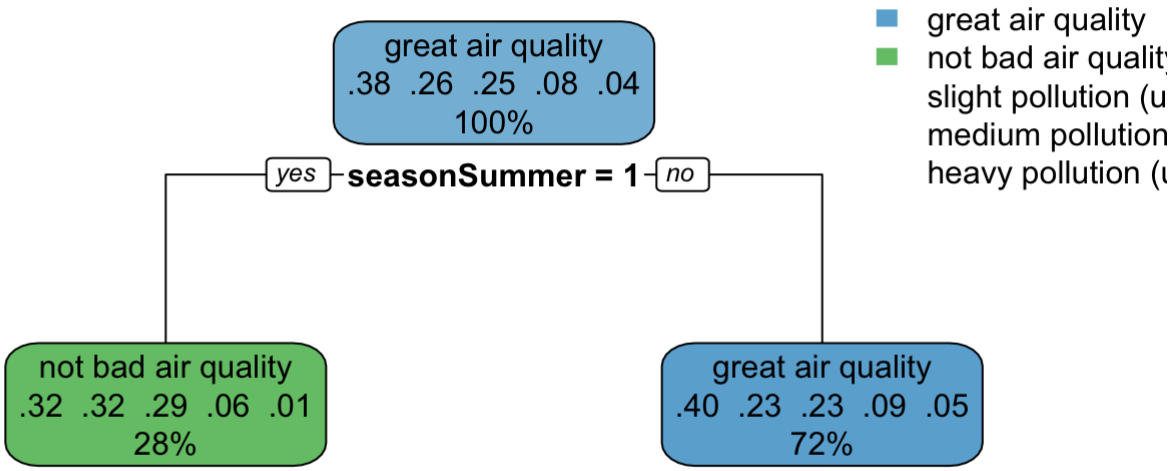
Interpretation

Sensitivity: 80.9% of truly polluted days are expected to be caught by our pollution model and about 19.1% of pollution days will be missed. Specificity: 57.0% of truly not polluted days are expected to be marked as legitimate by our model and about 43% of truly unpolluted days will be falsely marked as polluted. In our case, it's more important to detect all polluted days and warm people about the pollution, which makes sensitivity more important. And because of the relatively high sensitivity, we conclude that our model performs very well. However, the overall accuracy of our model 72.1%, only about 10% more than the NIR. This suggests that our model is improving, is doing something, but not too much. Also, the SensSD, SpecSD and AccuracySD, are 0.009444763, 0.01292559, 0.005391899 respectively.

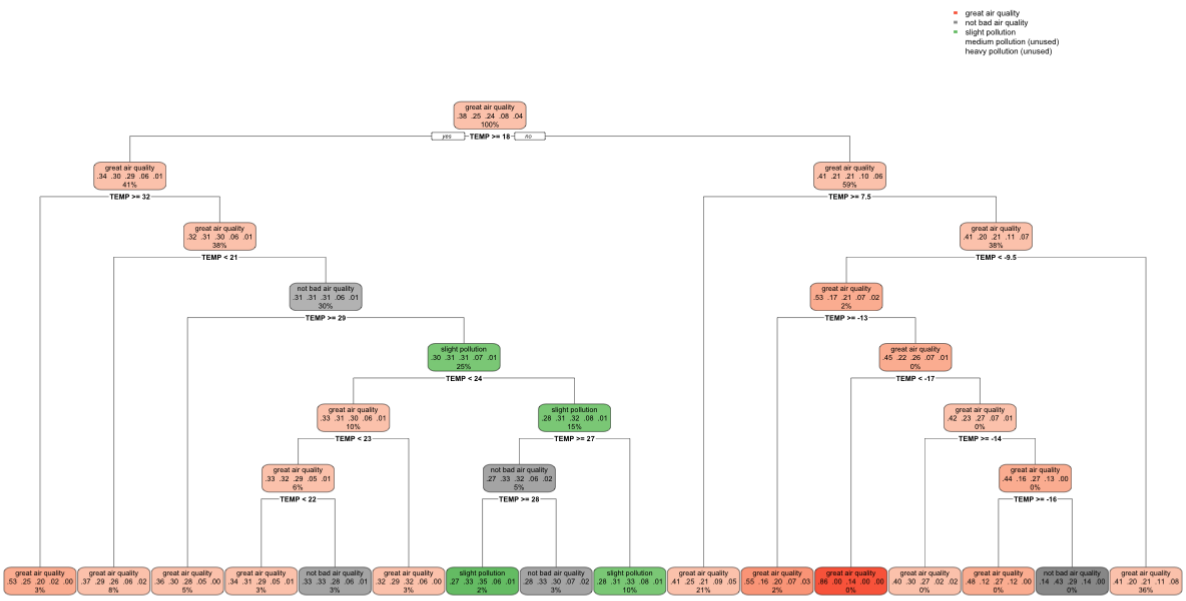
Besies logistic regression models, we also use Trees in our classifications:

Here are some simple tree models:

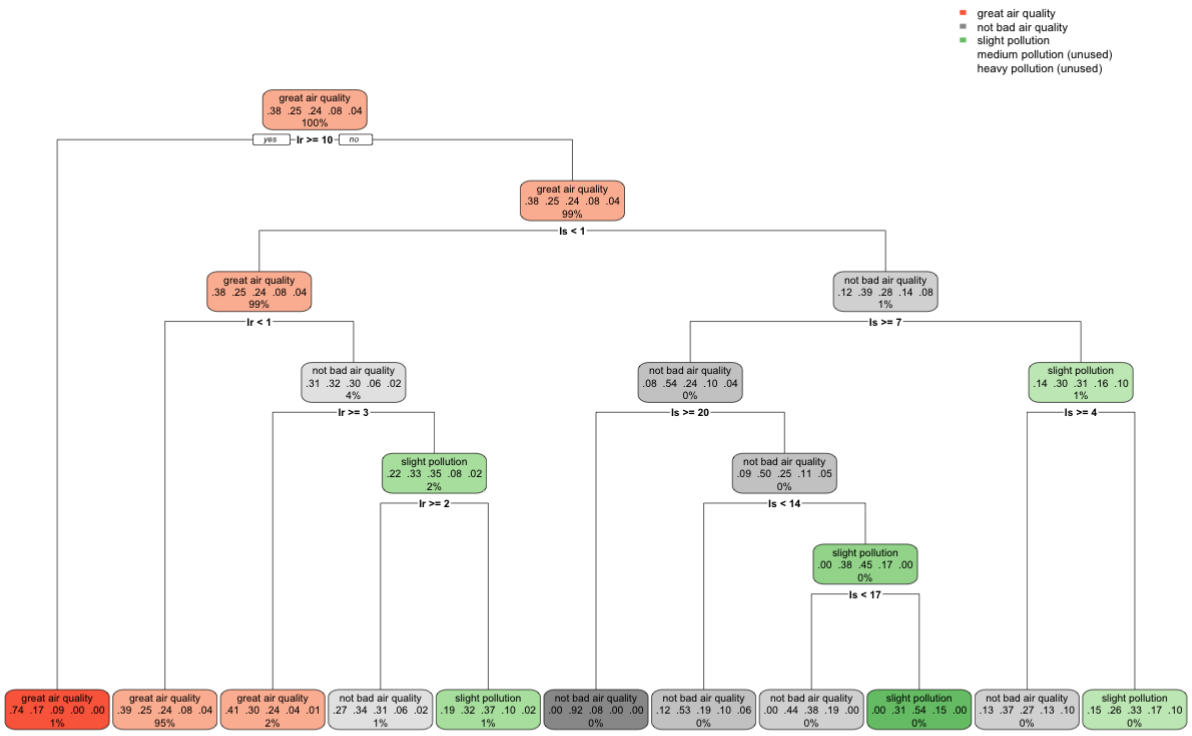
```
set.seed(253)
tree_mod_lowcp1 <- train(
    pm2.5Level ~ season,
    data = PM2.5,
    method = "rpart",
    metric = "Accuracy",
    tuneGrid = data.frame(cp = 0),
    trControl = trainControl(method = "cv", number = 10),
    na.action = na.omit
)

rpart.plot(tree_mod_lowcp1$finalModel)
```




```
set.seed(253)
tree_mod_lowcp2 <- train(
    pm2.5Level ~ TEMP,
    data = PM2.5,
    method = "rpart",
    metric = "Accuracy",
    tuneGrid = data.frame(cp = 0),
    trControl = trainControl(method = "cv", number = 10),
    na.action = na.omit
)

rpart.plot(tree_mod_lowcp2$finalModel)
```

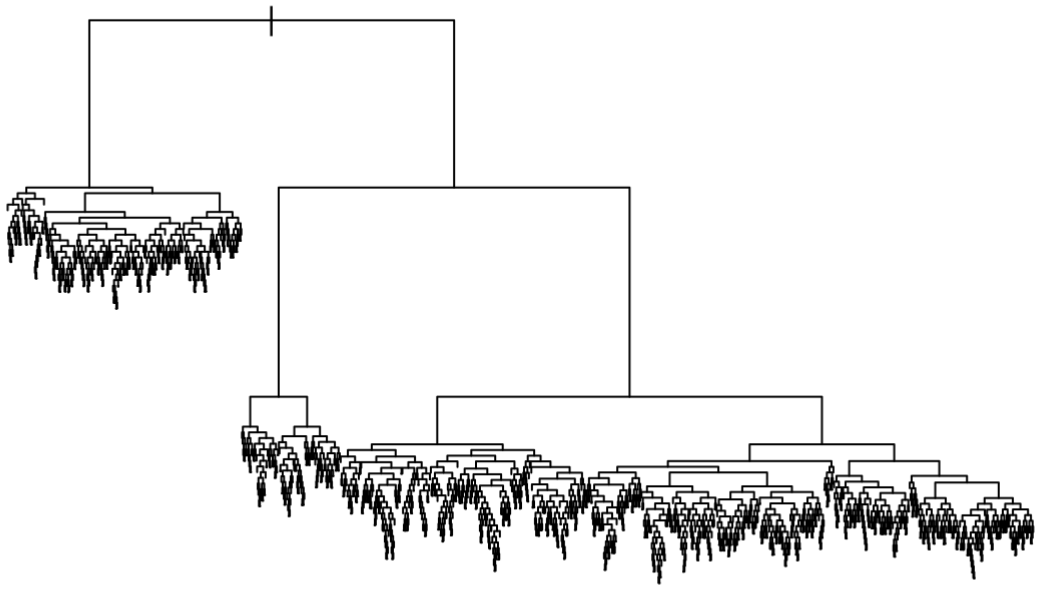



```
set.seed(253)
tree_mod_lowcp3 <- train(
    pm2.5Level ~ Is+Ir,
    data = PM2.5,
    method = "rpart",
    metric = "Accuracy",
    tuneGrid = data.frame(cp = 0),
    trControl = trainControl(method = "cv", number = 10),
    na.action = na.omit
)

rpart.plot(tree_mod_lowcp3$finalModel)
```



And we add more predictors for each split and we also use different complexity parameter values so that any split that does not increase node purity by cp not attempted

```
set.seed(253)
tree_mod <- train(
    pm2.5Level ~ season+PRES+cbwd+Iws+Is+Ir,
    data = PM2.5,
    method = "rpart",
    tuneGrid = data.frame(cp = seq(0, 0.5, length.out = 1)),
    trControl = trainControl(method = "cv", number = 10, selectionFunction = "oneSE"),
    metric = "Accuracy",
    na.action = na.omit
)

plot(tree_mod$finalModel)
```



Pick a case to make prediction:

```
# Pick out training case 2000 to make a prediction
test_case <- PM2.5[2000,]
# Show only the needed predictors
test_case %>% select(TEMP,cbwd,season)
```

```
##      TEMP cbwd season
## 2095    3   cv Spring
```

Based on prediction, we see this particular day as being slightly polluted. It's better to warn people about this situation based on our tree model prediction.

```
predict(tree_mod, newdata = test_case, type = "prob")
```

```
##      great air quality not bad air quality slight pollution medium pollution
## 2095         0.2432432           0.1891892        0.3513514       0.05405405
##      heavy pollution
## 2095       0.1621622
```

```
predict(tree_mod, newdata = test_case, type = "raw")
```

```
## [1] slight pollution
## 5 Levels: great air quality not bad air quality ... heavy pollution
```

Check for tree model variable importance:

```
tree_mod$finalModel$variable.importance
```

```
##          Iws         PRES       cbwdNW       cbwdNE seasonWinter   seasonFall
##   3168.33861   1570.30797   1445.73238    313.80127    296.94422    236.74497
##           Ir seasonSummer       cbwdSE           Is
##    174.04417    170.41216    140.18607     86.96029
```

From the tree models, we see that wind speed is the most significant predictor in our model. As wind speed increases, the pollutants will be blown away, causing air quality to improve significantly. We also see that northwestern wind is also significant. Wind from the northwest is usually associated with sand and dust storms. These storms bring a lot of pollutants into the city of Beijing and causing the air quality to drop. The other significant predictor is PRES. After some research, I find out that temperature inversion could happen when the atmosphere pressure is high. The high warm air covers the lower cold air, causing air convection to stop. Without this air convection, the pollutants in the air are trapped, leading to serious pollution. The pollution caused by temperature inversion are also seen in London and Los Angeles in the last century.

Conclusion:

We analyze the pm2.5 data in Beijing by using two supervised learning methods: regression and classification. For regression, we build linear and nonlinear models and conclude that the GAM model with all predictors is actually the best one, with the smallest MAE error metrics value. From the model, we see that wind from northwest and northeast are linearly and positively associated with pm2.5. Also, pressure, temperature and wind speed is generally negatively associated with pm2.5 concentration while dew point is positively related. For classification, we build logistic regression and trees models. For the logictic regression with almost all predictors, we achieved a sensitivity of over 80% and on overall accuracy over 72%. This allows us to catch the real polluted days rather accurately and warn people beforehand. From the tree model, we build some simple trees with only one predictor and gradually to a large tree with almost every predictor. The tree model shows us that the most important variables are wind speed, pressure, wind direction north and season winter. These importance levels allow us to delve into why these variable are particularly important and how we can do to address them. As a policy indicator, we know that wind from the north brings a lot of pollutants into the city of Beijing. So it's possible to plant trees to absorb that sand in order to improve air quality. Also, winter in Beijing is more likely to be polluted. The exhaust from cars and factories are trapped in the air because of low temperature, low wind speed and high pressure that is usually associated with winter. It's possible to limit the amount of exhaust emission by limiting the amount of cars on the road.