Modeling Precipitation with Multiple Linear Regression

John Lomas

11/8/2018

Set Up

We'll begin by preparing the dataframe containing the weaather data from which we will construct our precipitation models.

```r
# Load libraries necessary for model diagnostics
library(car)
library(MASS)
library(dplyr)

# Load the data and re-order the Months factor for visually appealing plotting
weather <- read.csv("weather_df.csv")
weather$Month = factor(weather$Month, levels(weather$Month)[c(5, 4, 8, 1,
                                                              9, 7, 6, 2,
                                                              12, 11, 10, 3)])

# Remove samples where data is missing from one or more variables
weather.complete <- weather[complete.cases(weather),]

# Transform temperature from celusius to kelvin
weather.complete <- mutate(weather.complete, Tavg_K = Tavg + 273.15)
weather.complete <- mutate(weather.complete, Tmax_K = Tmax + 273.15)
weather.complete <- mutate(weather.complete, Tmin_K = Tmin + 273.15)
head(weather.complete)
```

```
##      Date Year Day_of_Year Day_of_Run Solar_Rad Wind_Speed Wind_Direction
## 1 12/1/05 2005         335          1     0.097       2.99              2
## 2 12/2/05 2005         336          2     0.841       2.02            295
## 3 12/3/05 2005         337          3     2.794       2.22            126
## 4 12/4/05 2005         338          4     2.295       0.82             90
## 5 12/5/05 2005         339          5     2.322       0.46            148
## 6 12/6/05 2005         340          6     2.230       0.55            198
##   Wind_Gust Tavg Tmax  Tmin Havg Hmax Hmin Pressure Snow_Depth
## 1     19.89  1.9  3.0  -0.1   98   99   95      796      244.8
## 2     13.88 -4.7 -0.2  -7.3   90   98   79      799      250.8
## 3     18.19 -6.5 -3.5 -10.0   72   94   51      806      453.6
## 4     13.00 -5.9  0.6 -10.7   68   88   49      811      253.2
## 5      6.47 -3.1  3.7  -7.4   51   78   25      810      253.6
## 6      6.92 -2.1  4.9  -7.0   45   63   21      808      251.2
##   Total_Precip    Month Tavg_K Tmax_K Tmin_K
## 1        99.06 December 275.05 276.15 273.05
## 2         3.05 December 268.45 272.95 265.85
## 3         1.52 December 266.65 269.65 263.15
## 4         0.00 December 267.25 273.75 262.45
## 5         0.25 December 270.05 276.85 265.75
## 6         0.00 December 271.05 278.05 266.15
```

Generate the full model

We'll start by fitting a linear model using all of the potentially relevant variables from the weather dataset: Temperature, Pressure, Humidity, and Solar Radiation.

```
# Generate the full model with Total Precipitation as the response variable
fullModel <- lm(Total_Precip ~ Pressure + Tmin_K + Tmax_K + Tavg_K +
                                  Hmin + Hmax + Havg +
                                  Solar_Rad, data = weather.complete)

# Print summary of model fitting
summary(fullModel)
```

```
##
## Call:
## lm(formula = Total_Precip ~ Pressure + Tmin_K + Tmax_K + Tavg_K +
##     Hmin + Hmax + Havg + Solar_Rad, data = weather.complete)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.996  -2.723  -0.423   1.431 154.203
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.90081   27.24852   0.730 0.465231
## Pressure    -0.14275    0.03573  -3.995  6.6e-05 ***
## Tmin_K      -0.09507    0.11617  -0.818 0.413208
## Tmax_K      -0.13257    0.08034  -1.650 0.098999 .
## Tavg_K       0.56883    0.16334   3.482 0.000503 ***
## Hmin         0.04985    0.01959   2.544 0.010996 *
## Hmax        -0.02838    0.02270  -1.250 0.211302
## Havg         0.10970    0.03281   3.343 0.000836 ***
## Solar_Rad   -1.90687    0.15785 -12.080  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.858 on 3346 degrees of freedom
## Multiple R-squared:  0.2496, Adjusted R-squared:  0.2478
## F-statistic: 139.1 on 8 and 3346 DF,  p-value: < 2.2e-16
```

Clearly, the full model is already probelmatic. There are multiple insignificant coefficients and the Adjsusted R-squared value is very low; indicating that the model represents littl of the total variation in the data set. Next, we gain an even clearer idea of the problems associated with the model by checking the assumptions of linear regression.

With any multiple linear regression, the modeler assumes that: * The errors are normal distributed * The errors have a constant variance * The the errors are statistically independent * The underlying function is linear in nature * The independent variables are statistically independent

Model Diagnostics

First, we'll define functions that we'll use during for model diagnostics.

```
# Compute variance inflation factor
VIF_func <- function(model) car::vif(model)
```

```
# qq plot for studentized residuals
QQ_func <- function(model) qqPlot(model, main="QQ Plot")

# Residual histogram plot
ResHist_func <- function(model) {
                      # Compute the studentized residuals
                      sresid <- studres(model)

                      # Generate a histogram plot of the studentized residuals
                      hist(sresid, freq=FALSE,
                           main="Distribution of Studentized Residuals",
                           xlim = c(-5,5),
                           ylim = c(0, 0.6),
                           breaks = 20)

                      # Generate line to aid in viewing the distribution
                      xfit<-seq(min(sresid),max(sresid),length=40)
                      yfit<-dnorm(xfit)
                      lines(xfit, yfit)
}

# component + residual plot
CRP_func <- function(model) crPlots(model, ylab = "Total_Precip")
```

Multicollinearity

The first thing to check for is issues with multicollinearity. Multicollinearity occurs when the independent variables are coorelated. This is a problem because severe multicollinearity makes it difficult to accurately predict regression coefficients.

Here, we'll compute the Variance Inflation Factor (VIF) for each variable and look for those variables that have a VIF less than or equal to 10.

```
all_vifs <- VIF_func(fullModel)
print(all_vifs)
```

```
##  Pressure    Tmin_K    Tmax_K    Tavg_K      Hmin      Hmax      Havg
##  1.749368 30.683658 27.266180 81.922870 12.366527  7.410510 25.955696
## Solar_Rad
##  3.148090
```

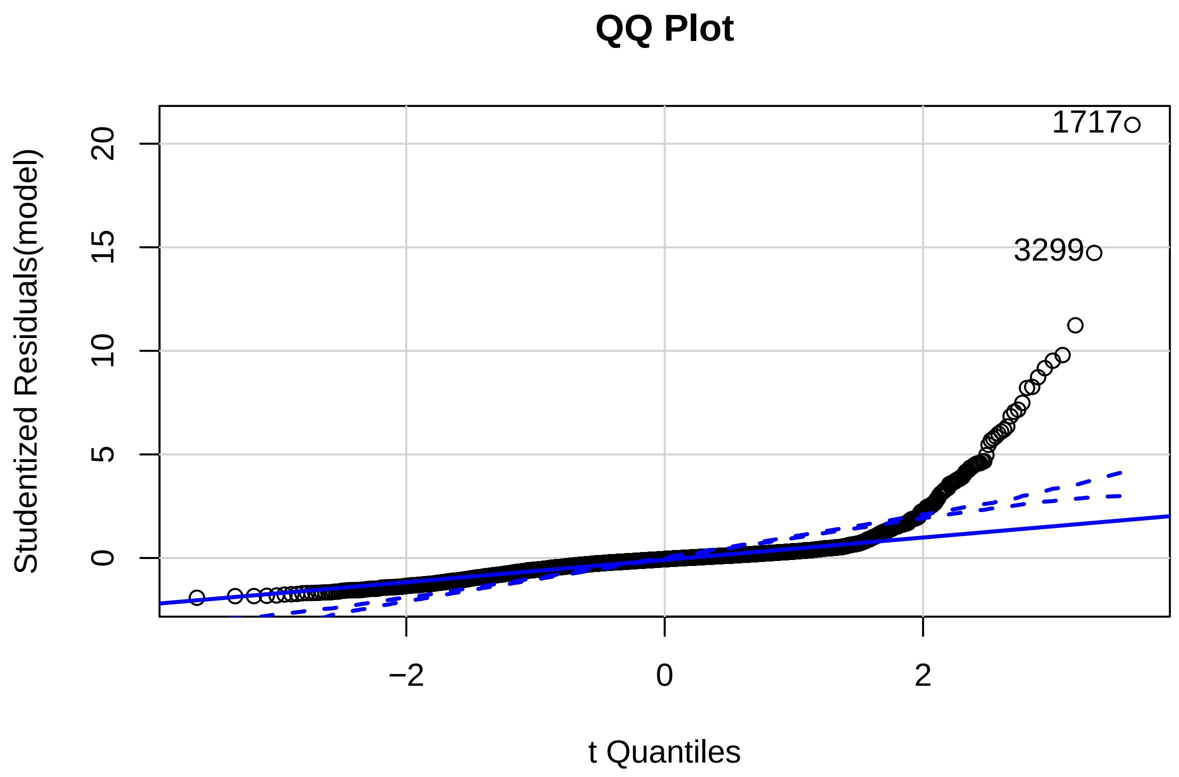
Multicollinearity is a problem in the model and it will have to be delt with as we move forward.

Normality of Error Terms

Next we'll evaluate whether the errors are normally distributed about the regression model. Here, we'll use a Q-Q PLot and look for the data to follow a linear trend.

```
# qq plot for studentized residuals

QQ_func(fullModel)
```

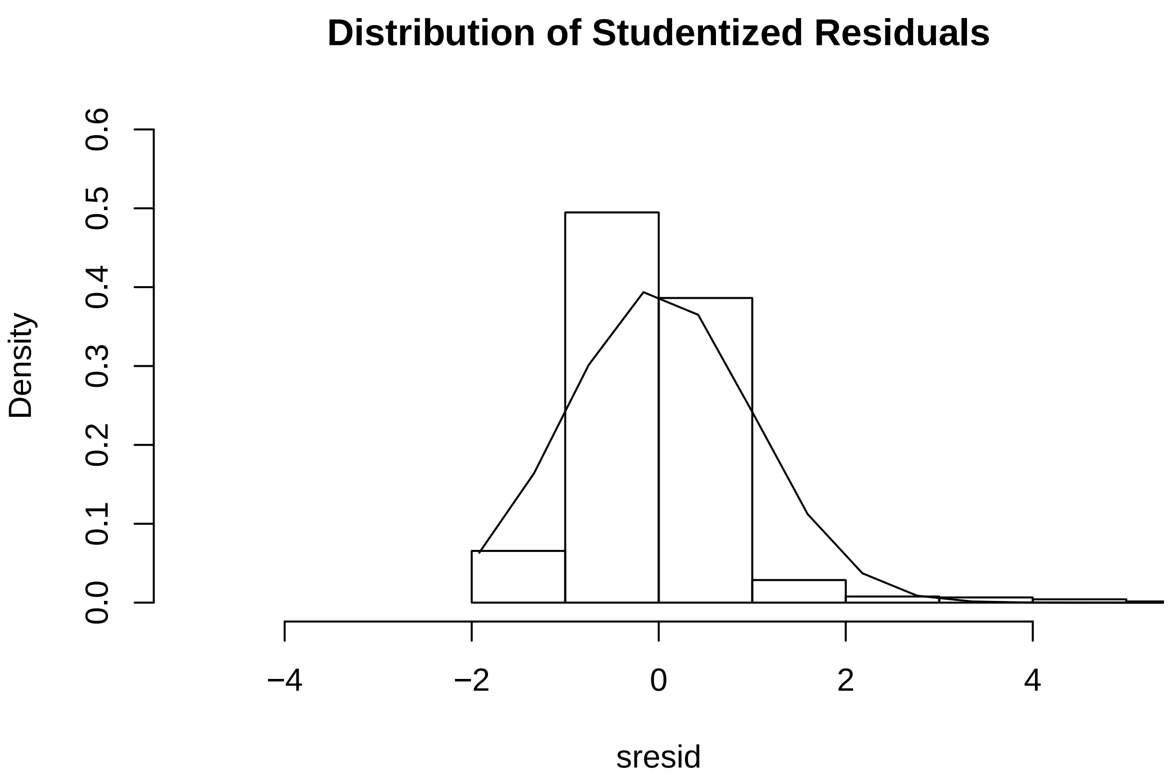


```
## [1] 1717 3299
```

Normality is preserved until high precipitation days. This means that the model is likely performing poorly in the region where precipitation actually occurs.

The non-normailty can also be observed using a residual histogram plot. The histogram below shows that the distribution of the residuals is skewed to the left; an indication that the assumption of normality is not being met.

```
ResHist_func(fullModel)
```



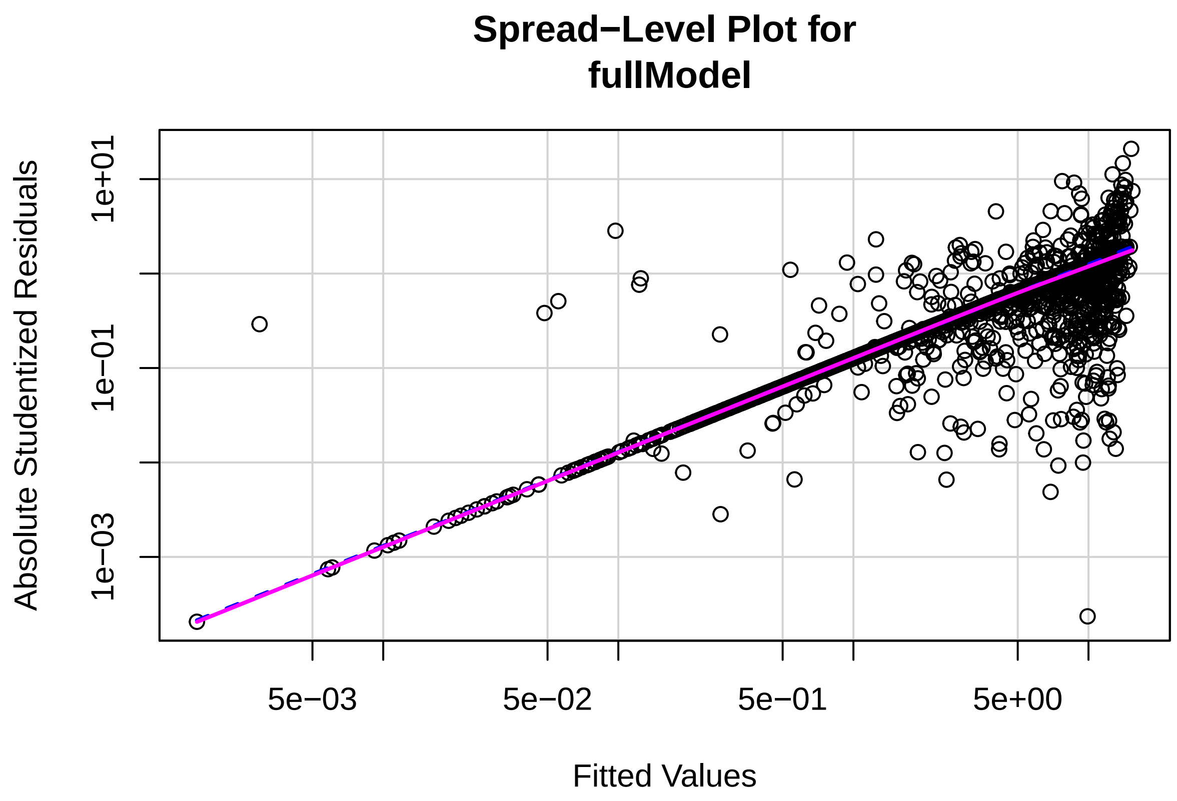
Constancy of error variance

Next we'll check the assumption of homoskedasticity (constancy of error variance) using the Breusch-Pagan test, which tests the null hypothesis of homoskedasticity. We can also view the error variance using a spread-level plot to see how the variance changes.

```r
# Breusch-Pagan test
ncvTest(fullModel)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 6125.165, Df = 1, p = < 2.22e-16
```

```r
# plot studentized residuals vs. fitted values
spreadLevelPlot(fullModel)
```



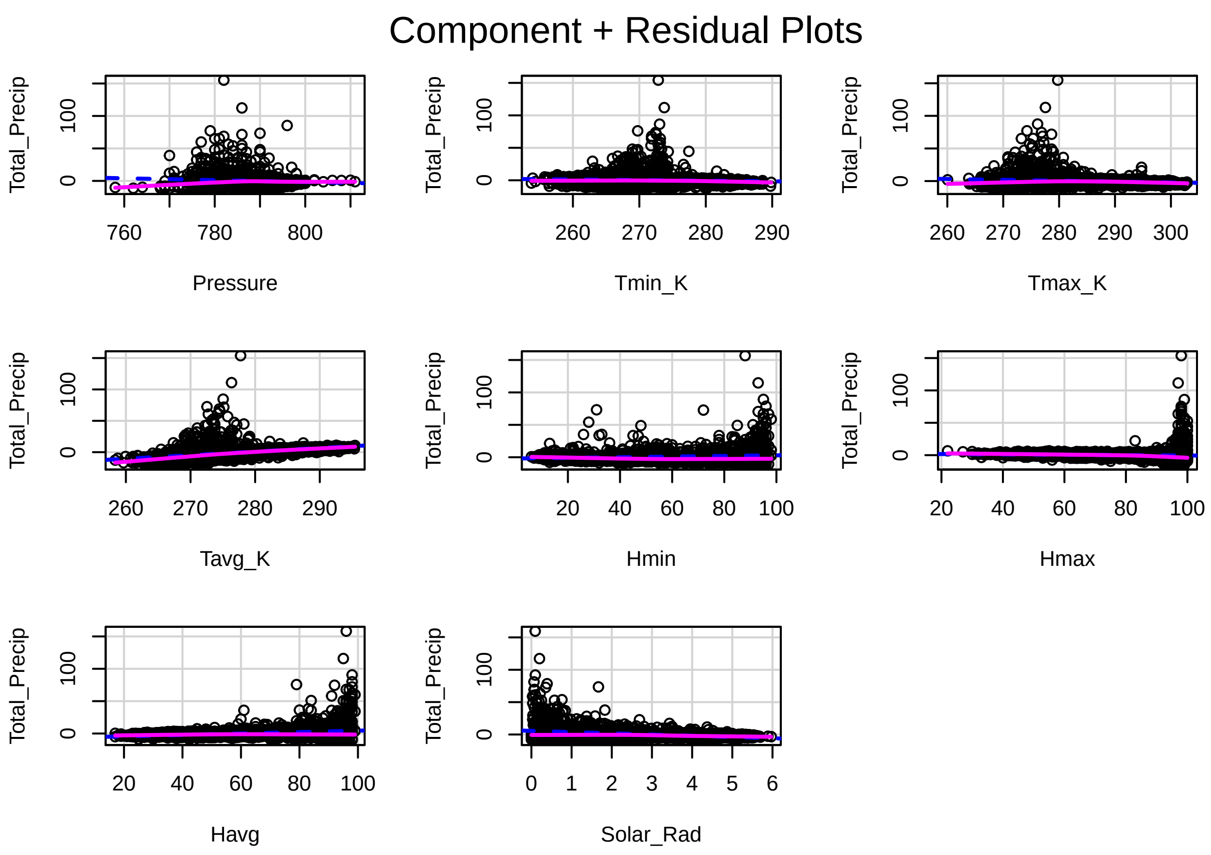
```
##
## Suggested power transformation:  0.006348529
```

Clearly, the variances are not constant. The null hypothesis of homoskedasticity was soundly rejected by the Breusch-Pagan test and the variance clearly increases towards the right of the Spread-Level Plot. This is another problem that we will deal with going forward.

Non-linearity

The last assumption we will look into is the assumpition that the predictor variables are linearly related to the response variable. For this we will look at the component residual plots. We will look for the data to generally follow a linear line of best fit.

```
CRP_func(fullModel)
```



As we saw while looking at these distributions in the previous section, many of the variables do not vary linearly with respect to the total precipitation. An adequate transformation will be needed to obtain linearity.

So far, we have identified many problems associated with the full model. These problems are: * The data is non-linear according to the Component Residual Plots * The error variances are not constant according to the Breusch-Pagan Test * The errors are not normally distributed via the Q-Q Plot * Severe Multicollinearity exists between the predictor variables

Now we will attempt to remedy these issues…

Generate an improved model

To address the multicollinearity problem, we can simply remove variables that are obviously realted and re-generate the model. Thus, instead of having all data related to temperateratature and humidity in the model, we will use only the daily average values

```
reducedModel <- lm(Total_Precip ~ Pressure + Tavg_K + Havg + Solar_Rad, data = weather.complete)
summary(reducedModel)
```

```
##
## Call:
## lm(formula = Total_Precip ~ Pressure + Tavg_K + Havg + Solar_Rad,
##     data = weather.complete)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.768  -3.057  -0.487   1.506 154.819
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 38.58299   26.87731   1.436    0.151
## Pressure    -0.18036    0.03500  -5.154  2.7e-07 ***
## Tavg_K       0.37431    0.02759  13.568  < 2e-16 ***
## Havg         0.13830    0.01004  13.774  < 2e-16 ***
## Solar_Rad   -2.10944    0.13611 -15.498  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.889 on 3350 degrees of freedom
## Multiple R-squared:  0.2428, Adjusted R-squared:  0.2419
## F-statistic: 268.6 on 4 and 3350 DF,  p-value: < 2.2e-16
```

```r
red_vifs <- car::vif(reducedModel)
print(red_vifs)
```

```
##  Pressure    Tavg_K      Havg Solar_Rad
##  1.665228  2.318672  2.411942  2.322414
```

Simply by removing clearly related variables, we now have a model with limited multicollinearity isues and a full set of significant regression coefficients. The Adjusted R-squared is about the same so there was no improvement in the model's predictive ability.

Next, we'll perform the Yeo-Johnson transformation which might help our heteroskedasticity issue as well as our non-linearity issue. Afterwards, we will diagnose the model once more.

```r
# run the yeo-Johnson transformation
bc <- boxCox(reducedModel, family="yjPower", plotit = FALSE)

# Get the power for the transformation
lambda <- bc$x[which.max(bc$y)]

#Transform the depenedent variable
TP.transformed <- yjPower(weather$Total_Precip, lambda)

# Re-generate the model
YJmodel <- lm(TP.transformed ~ Pressure + Tavg + Havg + Solar_Rad, data = weather)
summary(YJmodel)
```

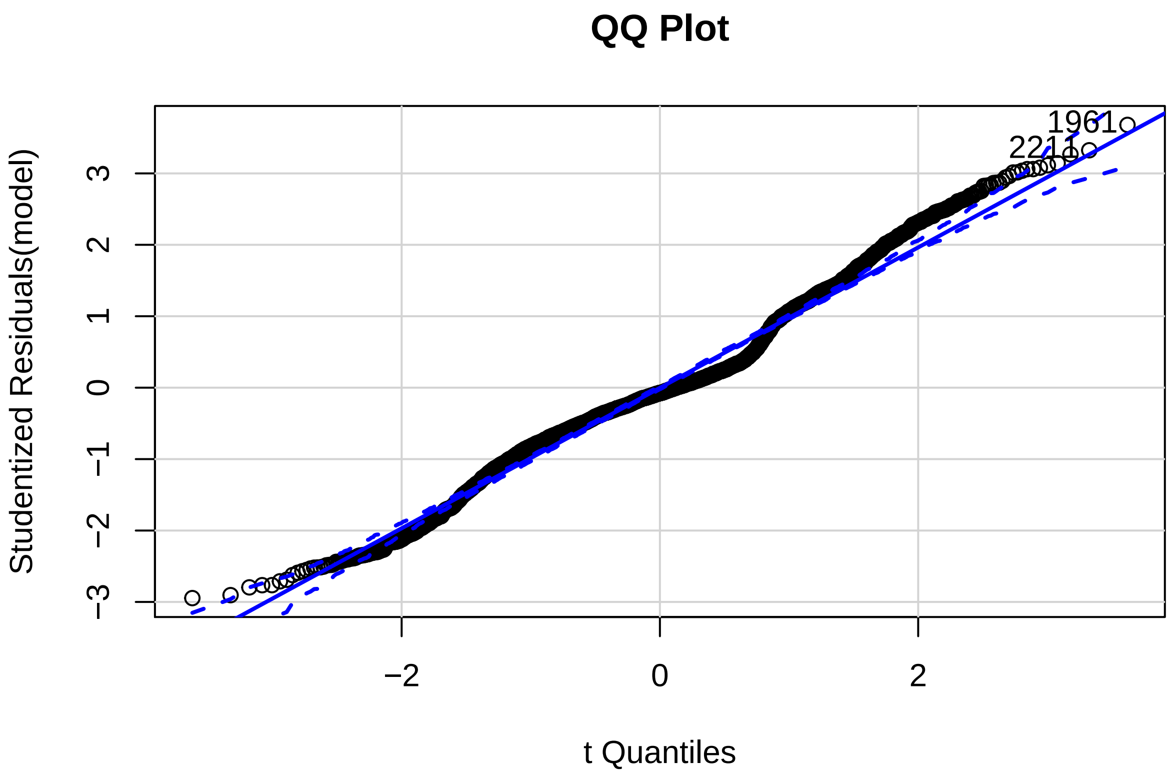
```
##
## Call:
## lm(formula = TP.transformed ~ Pressure + Tavg + Havg + Solar_Rad,
##     data = weather)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44173 -0.08323 -0.01116  0.06378  0.55341
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.0991878  0.5324527   5.821 6.42e-09 ***
## Pressure    -0.0040411  0.0006694  -6.037 1.74e-09 ***
## Tavg         0.0052565  0.0005277   9.961  < 2e-16 ***
## Havg         0.0046791  0.0001921  24.363  < 2e-16 ***
## Solar_Rad   -0.0378151  0.0026035 -14.525  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1509 on 3350 degrees of freedom
##   (694 observations deleted due to missingness)
## Multiple R-squared:  0.4275, Adjusted R-squared:  0.4268
## F-statistic: 625.4 on 4 and 3350 DF,  p-value: < 2.2e-16
```

Notice that with our transformation, the Adjusted R-Squared has almost doubled (although the predictive ability of the model is still quite poor).

Now lets check our normality, linearity, and variances once more. . . .

Normality

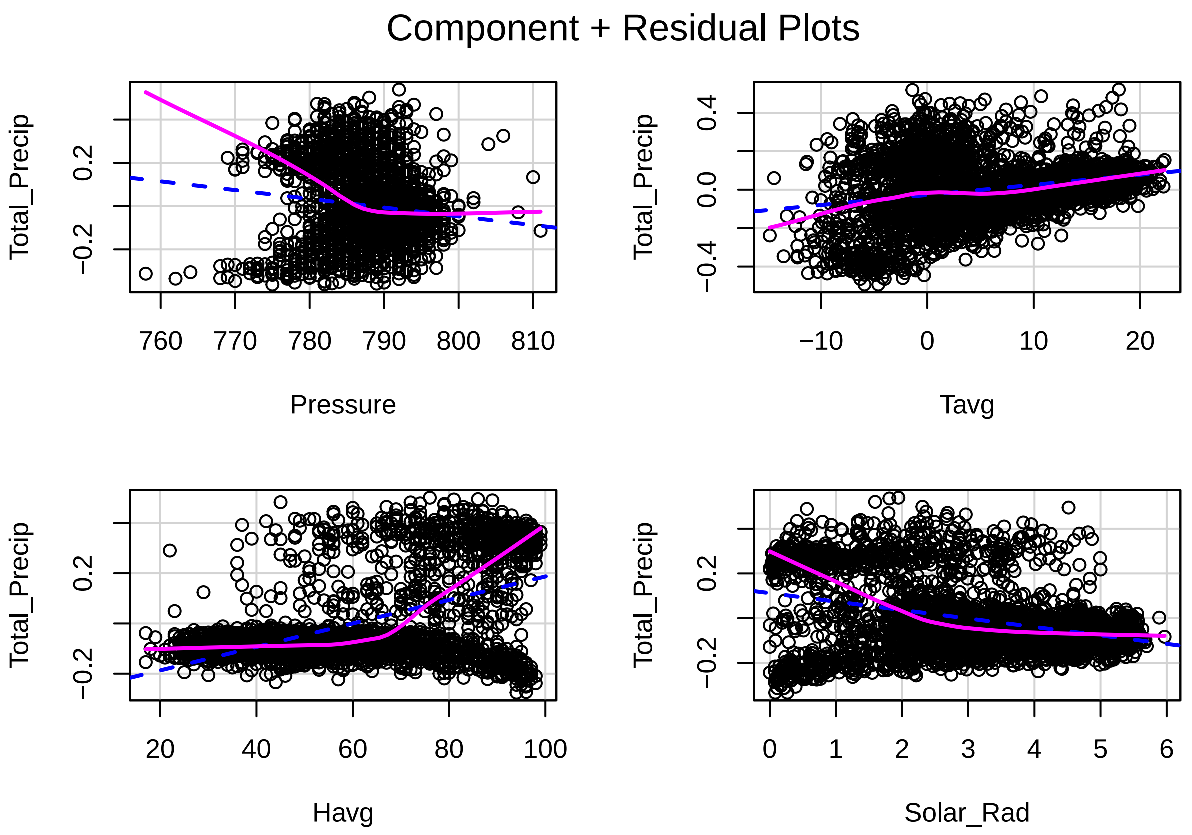
```
# qq plot for studentized residuals
QQ_func(YJmodel)
```



```
## [1] 1961 2211
```

Linearity

```
# component + residual plot
CRP_func(YJmodel)
```

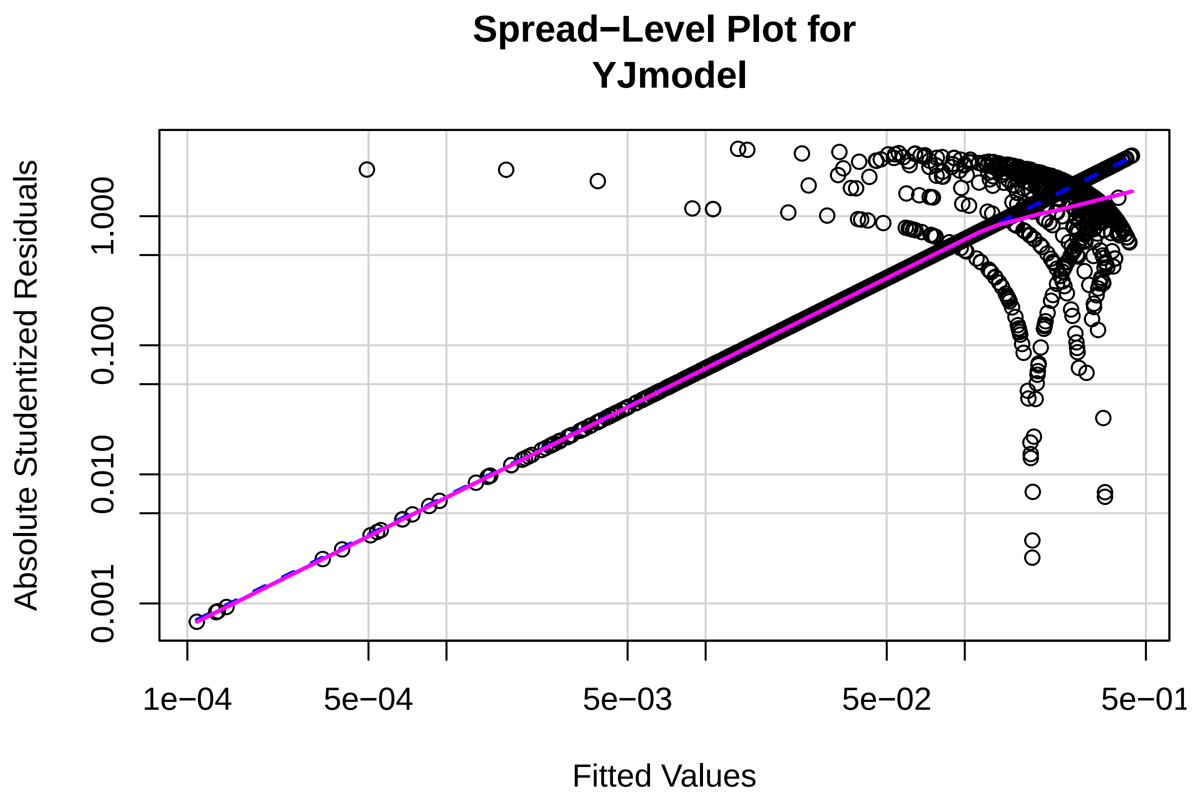


Error Variance

```r
# Breusch-Pagan test
ncvTest(YJmodel)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1086.701, Df = 1, p = < 2.22e-16
```

```r
# plot studentized residuals vs. fitted values
spreadLevelPlot(YJmodel)
```



```
##
## Suggested power transformation:  0.005825722
```

Conclusions

Removing some model terms and introducing the Yeo-Johnson Transformation eliminated the multicollinearity issue and helped to normalize the error distribution. However, these measures had a limited affect on the problems with heteroskedasticity and non-linearty. In order to effictively model precipitation events, other model types may need to be considered.