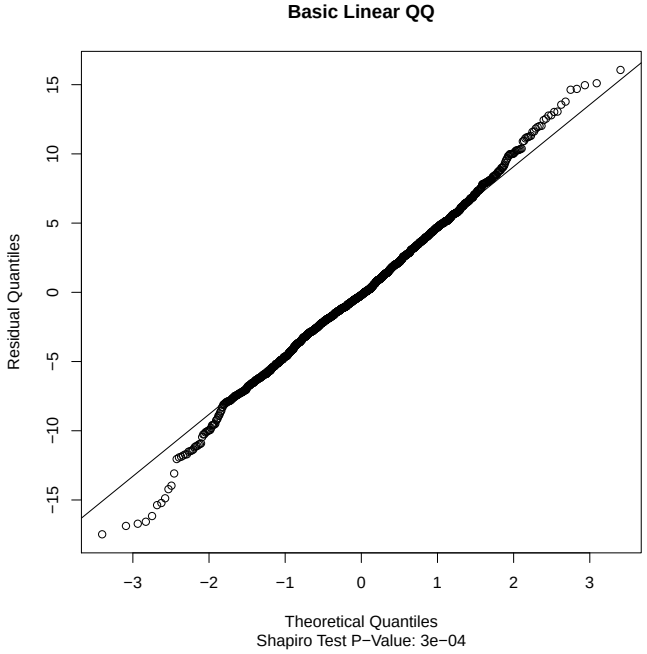
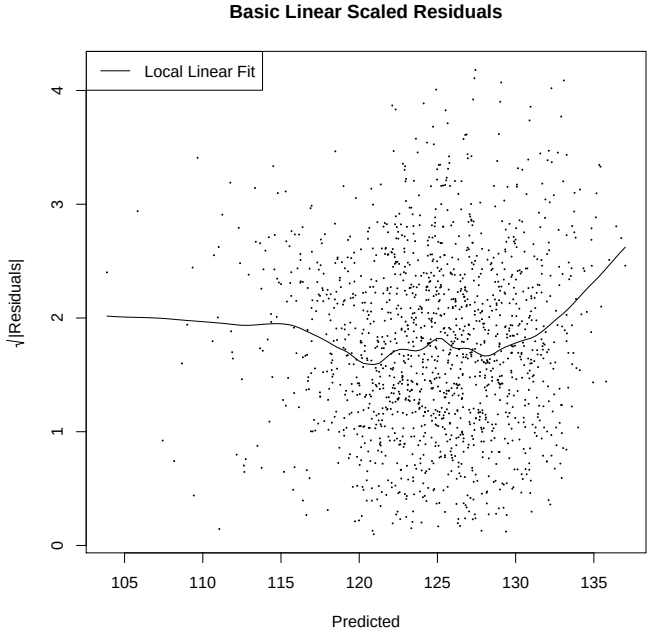
1. a) **Summary:** Fitting a simple linear model, I see that the constant variance assumptions don't hold, which would suggest transformation. The normally distributed error also isn't quite true, but the difference is fairly minor. The first choice for transformation the Box-Cox series of transformations, but the 95% confidence interval for the maximum likelihood includes 1, suggesting that such a transformation isn't worth while. Instead, what ended up working was to normalize all the variables so that their smallest value was 0 and the largest 1, and then using the square root of the variables, or the square root of one minus the variable, as predictors. This results in improvement, giving residuals with approximately constant variance.

Work: I ran the simple linear model:

```
lm(formula = pre ~ ., data = af)

Residuals:
     Min       1Q   Median       3Q      Max
-17.4788  -2.8845  -0.2094   3.1512  16.0645

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.328e+02  5.450e-01 243.720   <2e-16 ***
freq        -1.283e-03  4.213e-05 -30.440   <2e-16 ***
ang         -4.221e-01  3.891e-02 -10.847   <2e-16 ***
cl          -3.567e+01  1.632e+00 -21.865   <2e-16 ***
vel          9.995e-02  8.141e-03  12.278   <2e-16 ***
thi         -1.473e+02  1.502e+01  -9.807   <2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1

Residual standard error: 4.81 on 1496 degrees of freedom
Multiple R-squared:  0.5157,    Adjusted R-squared:  0.5141
F-statistic: 318.6 on 5 and 1496 DF,  p-value: < 2.2e-16
```

Considering possible problems other than nonlinearity, I looked at a QQ plot of the residuals and a plot of the square root of the absolute values of the residuals versus the predicted values:

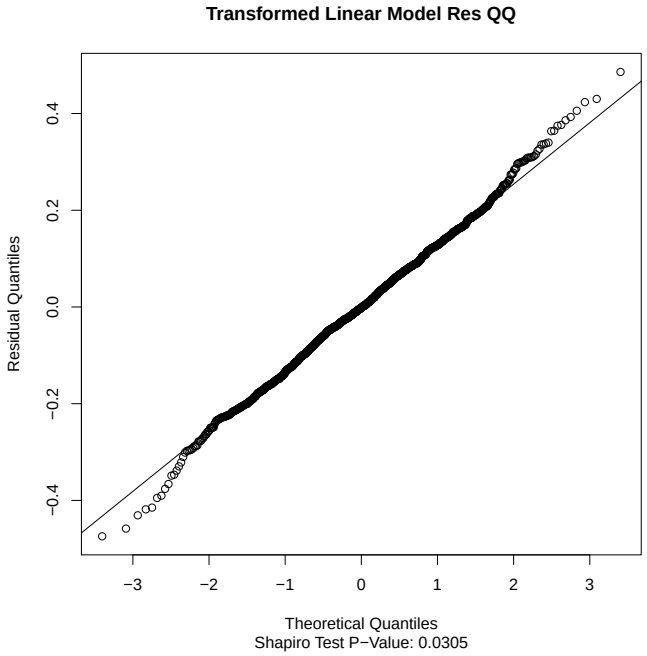


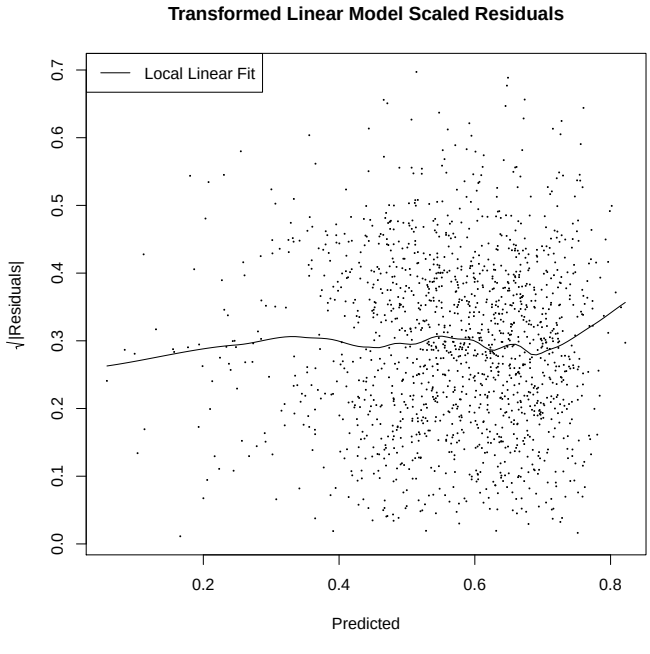


In the QQ plot, we see that the distribution of the residuals has much somewhat thicker tails than a normal distribution, though stays pretty close otherwise. The Shapiro-Wilkes test, with a p-value of .0003 has us reject normality, but the predictions are still unbiased if the normality assumption doesn't hold, and the deviation from normality is pretty small. The variation in the variance of the residuals doesn't look terrible, but the local linear regression fit seems to show some meaningful variation over the range of predictions. Thus, I ran a new model where, after I scaled the variables so they all go from 0 to 1, I transformed the predictors, taking the square root of one minus angle, chord length, and thickness, and the square root of velocity. This resulted in the following model:

```
lm(formula = pre ~ freq + sqrt(1 - ang) + sqrt(1 - cl) + sqrt(vel) +
    sqrt(1 - thi), data = af.s)

Residuals:
     Min       1Q   Median       3Q      Max
-0.47417 -0.08582 -0.00194  0.08558  0.48602

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.115386   0.025066  -4.603 4.51e-06 ***
freq          -0.622280   0.022562 -27.581  < 2e-16 ***
sqrt(1 - ang)  0.223197   0.023702   9.417  < 2e-16 ***
sqrt(1 - cl)   0.234694   0.012334  19.028  < 2e-16 ***
sqrt(vel)      0.100698   0.009657  10.427  < 2e-16 ***
sqrt(1 - thi)  0.409452   0.025718  15.921  < 2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1

Residual standard error: 0.1327 on 1496 degrees of freedom
Multiple R-squared:  0.479,     Adjusted R-squared:  0.4773
F-statistic: 275.1 on 5 and 1496 DF,  p-value: < 2.2e-16
```

Checking the fit again, I see some mild improvement:



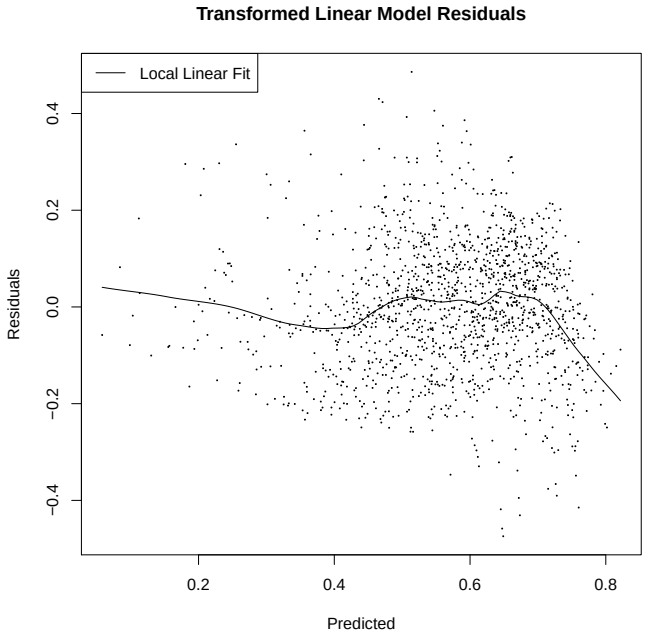


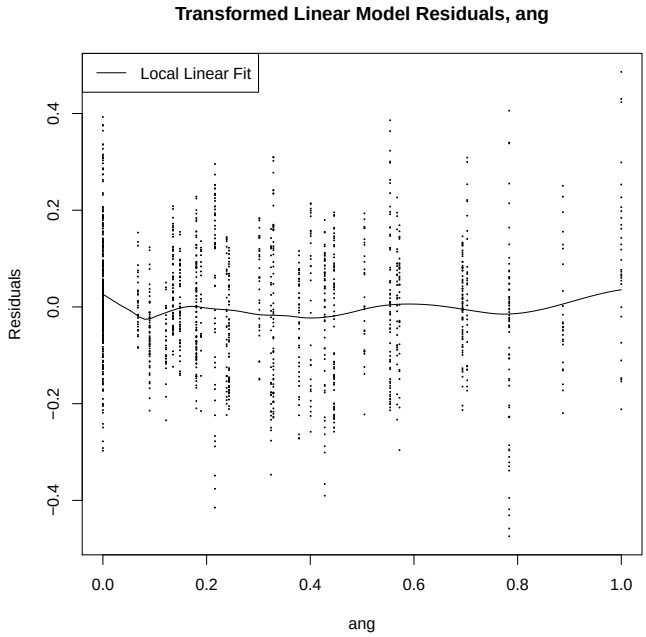
The Residuals are a tad closer to being normally distributed (the lower tail deviates a little less), but more importantly, as one can see from the linear fit, the variance appears to vary much less over the

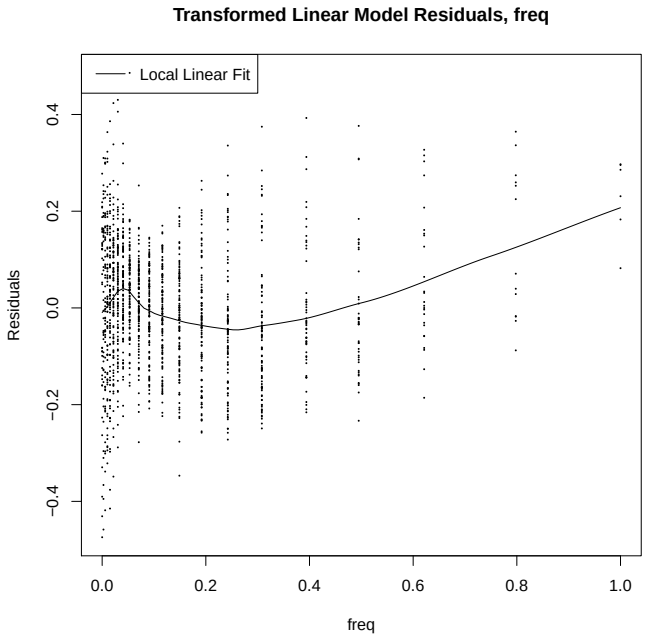
range of predictions, and looks close to being constant, except maybe at the end.

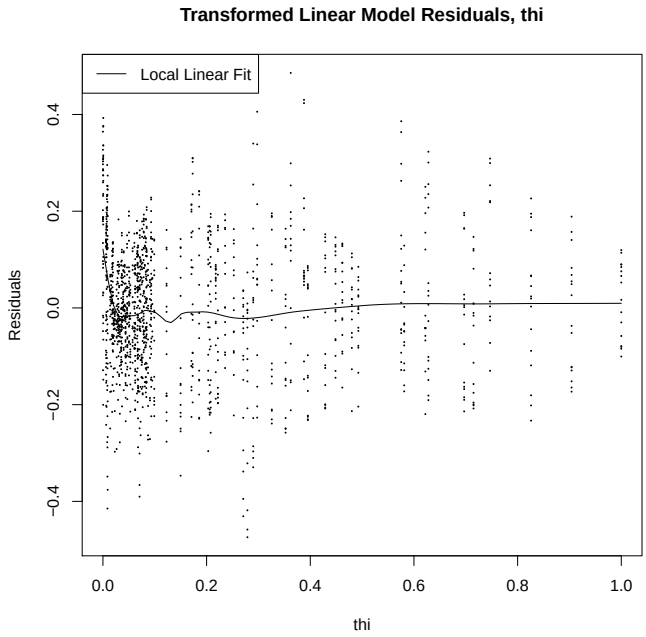
b) **Summary:** We seem to have some pretty meaningful nonlinearity in the model from part a. To improve upon it, I transform the outcome, pressure, by taking it square root, and replace the terms for frequency, angle, and chord length with a fourth degree polynomial of those three variables together. This reduced the non-linearity and vastly improved the overall fit, as judged by adjusted R-squared.

Work: I checked for non-linearity in the model from part a by plotting the residuals by predicted value and a few of the variables (the other two only have a few unique values and looked fine):





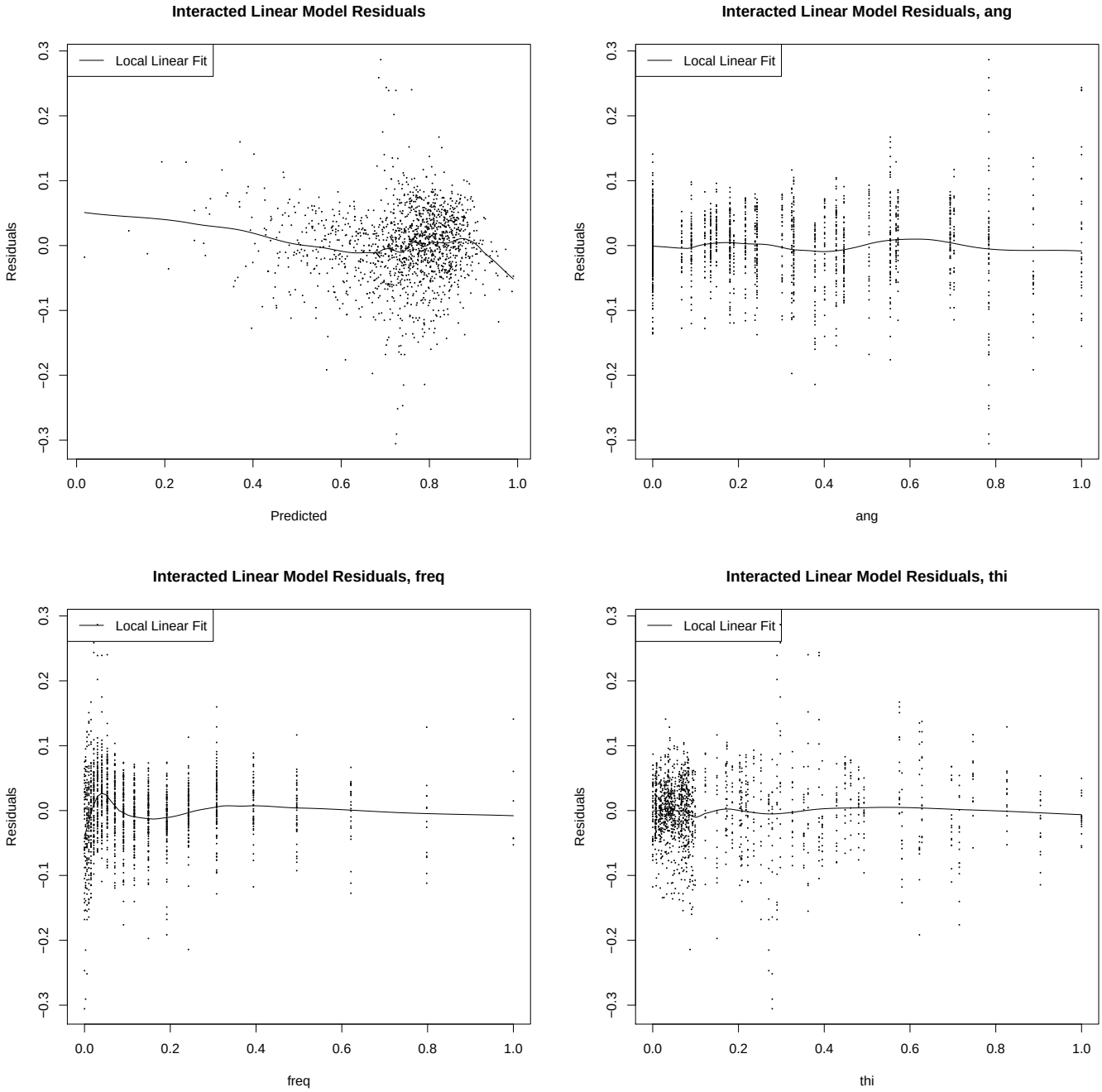




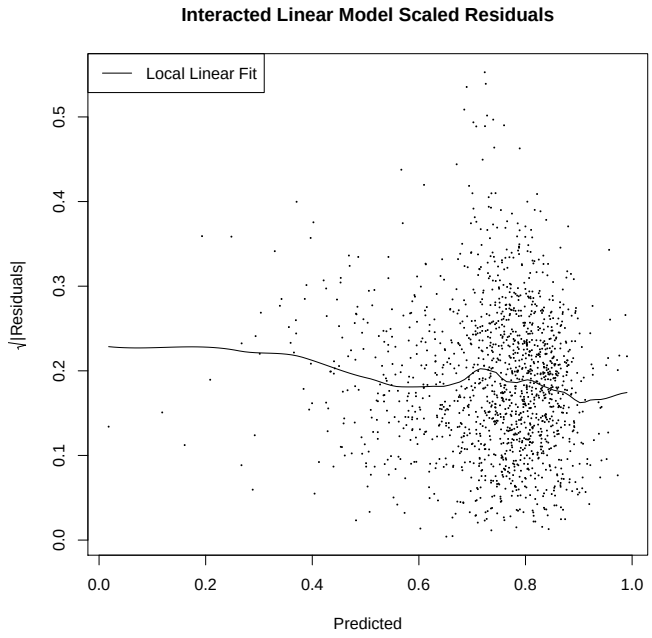
As you can see, the local linear fits pretty strongly show that the residuals don't have a mean of 0 over the range of predicted values nor range of frequency, even if it looks OK for angle and thickness. I couldn't find any way to remedy this without interactions; I tried a number of models which had every predictor as a factor and some kind of transformation of the outcome, since this would admit every possible transformation of each predictor, but every model still failed to have residuals with mean 0 over the range of predicted values. Thus, I introduced a set of orthogonal polynomials of degree up to 4 in terms of frequency, angle, and chord length together, as well as took the square root of pressure. This is the resulting model summary:

```
lm(formula = sqrt(pre) ~ polym(freq, ang, cl, degree = 4) + sqrt(vel) +
    sqrt(1 - thi), data = af.s)

Residuals:
      Min        1Q    Median        3Q       Max
-0.305558 -0.030814  0.002862  0.035226  0.286641

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                              4.283e-01  2.814e-02  15.220  < 2e-16 ***
polym(freq, ang, cl, degree = 4)1.0.0   -3.739e+00  4.900e-01  -7.631 4.18e-14 ***
polym(freq, ang, cl, degree = 4)2.0.0    1.323e-01  3.131e-01   0.423  0.67263
polym(freq, ang, cl, degree = 4)3.0.0    1.802e-01  1.303e-01   1.383  0.16688
polym(freq, ang, cl, degree = 4)4.0.0   -5.347e-01  7.095e-02  -7.536 8.42e-14 ***
polym(freq, ang, cl, degree = 4)0.1.0    2.124e+00  1.757e+00   1.209  0.22692
polym(freq, ang, cl, degree = 4)1.1.0   -8.541e+01  3.024e+01  -2.824  0.00480 **
polym(freq, ang, cl, degree = 4)2.1.0    3.550e+01  1.737e+01   2.044  0.04114 *
polym(freq, ang, cl, degree = 4)3.1.0   -2.411e+01  6.316e+00  -3.818  0.00014 ***
polym(freq, ang, cl, degree = 4)0.2.0    2.471e+00  1.269e+00   1.947  0.05167 .
polym(freq, ang, cl, degree = 4)1.2.0    1.716e+01  1.790e+01   0.959  0.33777
polym(freq, ang, cl, degree = 4)2.2.0   -1.797e+01  8.552e+00  -2.101  0.03579 *
polym(freq, ang, cl, degree = 4)0.3.0    1.210e+00  4.503e-01   2.688  0.00727 **
polym(freq, ang, cl, degree = 4)1.3.0    1.176e+01  5.186e+00   2.268  0.02346 *
polym(freq, ang, cl, degree = 4)0.4.0    4.760e-01  1.137e-01   4.187 3.00e-05 ***
polym(freq, ang, cl, degree = 4)0.0.1    8.880e-01  1.350e+00   0.658  0.51080
polym(freq, ang, cl, degree = 4)1.0.1   -6.173e+01  1.976e+01  -3.123  0.00182 **
polym(freq, ang, cl, degree = 4)2.0.1    5.885e+01  6.699e+00   8.785  < 2e-16 ***
polym(freq, ang, cl, degree = 4)3.0.1   -2.991e+01  2.963e+00 -10.094  < 2e-16 ***
polym(freq, ang, cl, degree = 4)0.1.1    1.515e+02  8.692e+01   1.742  0.08164 .
polym(freq, ang, cl, degree = 4)1.1.1    7.440e+02  1.097e+03   0.678  0.49773
polym(freq, ang, cl, degree = 4)2.1.1    9.364e+02  2.979e+02   3.143  0.00171 **
polym(freq, ang, cl, degree = 4)0.2.1    1.184e+02  5.738e+01   2.063  0.03930 *
polym(freq, ang, cl, degree = 4)1.2.1    3.799e+02  5.323e+02   0.714  0.47552
polym(freq, ang, cl, degree = 4)0.3.1    4.140e+01  1.741e+01   2.378  0.01753 *
polym(freq, ang, cl, degree = 4)0.0.2    1.092e+00  6.873e-01   1.589  0.11230
polym(freq, ang, cl, degree = 4)1.0.2    4.258e+01  7.726e+00   5.511 4.20e-08 ***
polym(freq, ang, cl, degree = 4)2.0.2   -2.766e+01  3.583e+00  -7.720 2.14e-14 ***
polym(freq, ang, cl, degree = 4)0.1.2    7.802e+01  3.986e+01   1.957  0.05050 .
polym(freq, ang, cl, degree = 4)1.1.2    6.053e+02  3.337e+02   1.814  0.06988 .
polym(freq, ang, cl, degree = 4)0.2.2    2.902e+01  2.005e+01   1.448  0.14792
polym(freq, ang, cl, degree = 4)0.0.3    4.024e-01  1.885e-01   2.134  0.03298 *
polym(freq, ang, cl, degree = 4)1.0.3   -3.240e+01  3.012e+00 -10.756  < 2e-16 ***
polym(freq, ang, cl, degree = 4)0.1.3    4.022e+00  8.862e+00   0.454  0.65002
polym(freq, ang, cl, degree = 4)0.0.4   -2.848e-01  6.914e-02  -4.119 4.03e-05 ***
sqrt(vel)                                6.358e-02  4.366e-03  14.562  < 2e-16 ***
sqrt(1 - thi)                            3.357e-01  2.612e-02  12.853  < 2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1

Residual standard error: 0.05817 on 1465 degrees of freedom
Multiple R-squared:  0.8175,    Adjusted R-squared:  0.813
F-statistic: 182.2 on 36 and 1465 DF,  p-value: < 2.2e-16
```

There are a lot of variables added, but not that many considering the size of the data set, and we see a big overall improvement in fit, as shown by the adjusted R^2 increasing to .813 from .477. Looking at plots of the residuals, we also seem to have mostly mitigated the issues with nonlinearity:

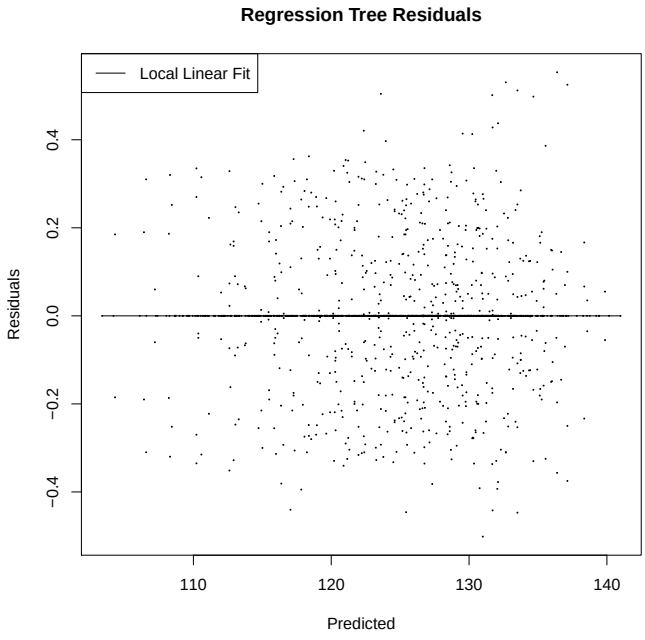


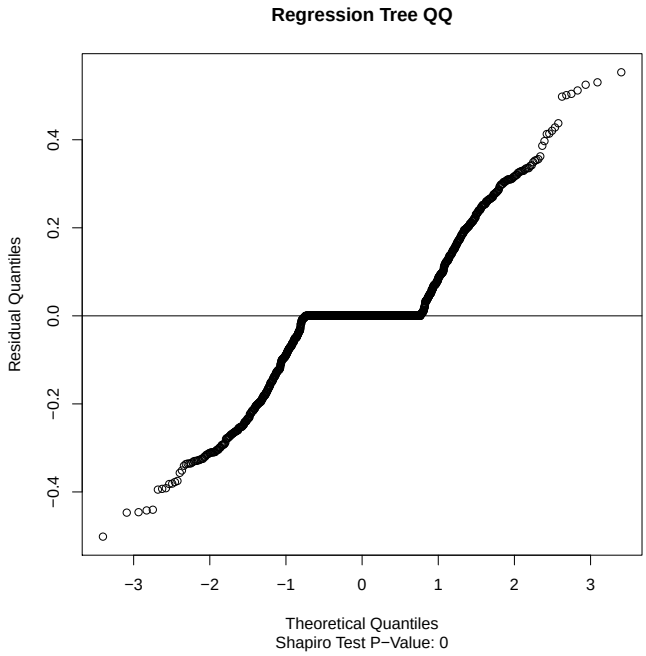
While there is still some nonlinearity, the mean is consistently right around 0, except in a few spots with sparse data. We do seem to have meaningful non-zero means at lower frequencies, and at the higher end of the predicted values, but these biases are small, and much better than with the previous model. Double checking the variance, we seem to still have close to constant variance, with possibly a little less consistency than we had before:



c) Of these two, each has its own advantages. The model from part a has the advantage of simplicity,, while only mildly violating the linear model assumptions, making it suitable for explanation. The confidence intervals obviously won't be perfect, since the predictions are biased and the normality assumption doesn't hold. However, the confidence level is high enough that we can still fairly safely say that the predicted positive or negative correlations between the predictors and outcome hold true, and the estimates of how much this is by serves a sufficient ballpark guess. On the other hand, the model from b does seem to fulfill the assumptions of a linear regression model mostly (except possibly normality) and has high R^2, so it is good for prediction, even if the confidence intervals of the prediction might be slightly off. It is just convoluted enough that it is hard to interpret what effect angle, frequency, and chord length have on pressure without making a visual, since there are no single coefficients on these predictors..

d) Fitting a tree model, I pruned the tree at a complexity parameter of $3.31 * 10^{-6}$. Any split which didn't decrease total R^2 by at least that much was discarded. This was chosen because it was the points at which 10 fold cross validation estimated there would be the lowest error. This gave a tree with 1121 splits for 1502 data points, which seems like a lot, but when one calculates the adjusted R^2, the result is still an impressive .999, having reduced the RSS to 27.467 from it original value of 71480.51. Between this and the cross validation, we are reasonably assured that this is not an over fit. We can look at the distribution of the residuals:





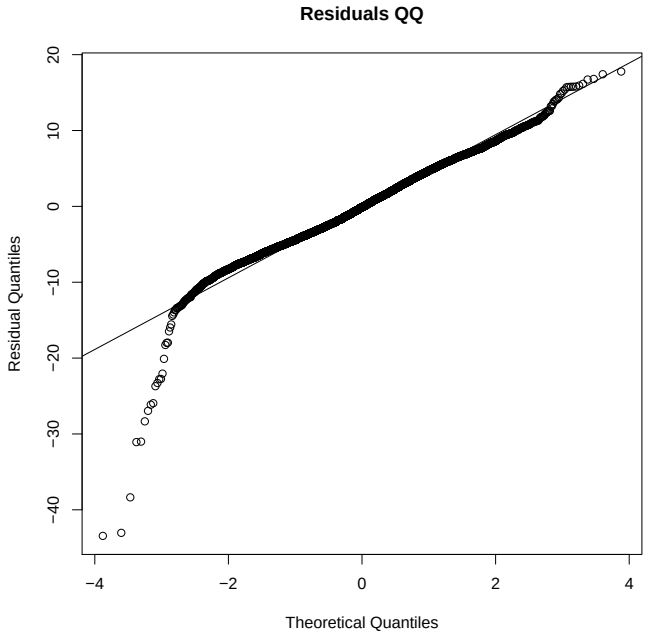
As we would expect, by construction, the residuals definitely have mean 0, with a large number of them, in fact, being 0. Additionally, the residuals are no where close to normally distributed, but this is to be expected with so many of them already at 0.

e) As we we have seen here, in data that doesn't naturally fit the assumptions necessary for linear regression, its hard to transform it to the point where all the assumptions hold well. One can add an arbitrary amount of flexibility to a linear model to make it work, through splines, polynomials, other spanning function bases, and the like, but this has to be done manually. By the point enough flexibility has been added, often there is no easy interpretation of coefficients. Here in lies the advantage of a tree model; it has arbitrary amounts of flexibility built in by default, making it appropriate and easier for prediction with this kind of messy data. If you happen to stumble on a data set that actually has a linear relationship between two variables though, a tree model will never predict quite as well nor be as interpretable as the linear regression.

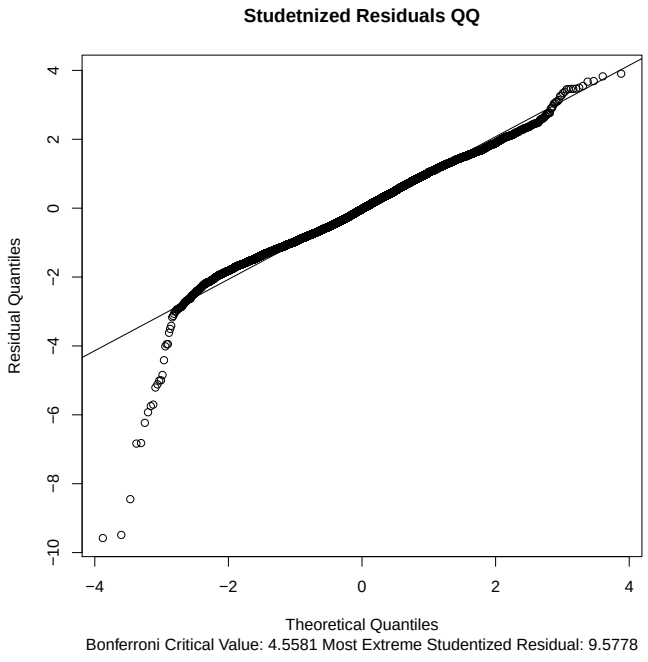
2. a) This is the summary from the linear model:

```
lm(formula = PE ~ ., data = cch)

Residuals:
    Min      1Q  Median      3Q     Max
-43.435  -3.166  -0.118   3.201  17.778

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) 454.609274   9.748512   46.634  < 2e-16 ***
AT           -1.977513   0.015289 -129.342  < 2e-16 ***
V            -0.233916   0.007282  -32.122  < 2e-16 ***
AP            0.062083   0.009458    6.564 5.51e-11 ***
RH           -0.158054   0.004168  -37.918  < 2e-16 ***
---
Signif. codes:  0  ***  0.001  **  0.01  *  0.05  .  0.1

Residual standard error: 4.558 on 9563 degrees of freedom
Multiple R-squared:  0.9287,    Adjusted R-squared:  0.9287
F-statistic: 3.114e+04 on 4 and 9563 DF,  p-value: < 2.2e-16
```

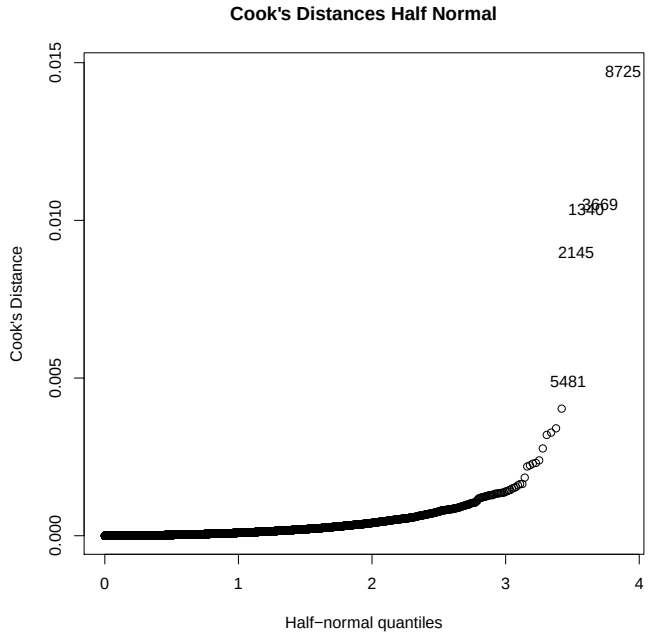
i. This residuals are pretty close to normal for the most part, but is different enough that we can say with a high degree of certainty that weren't actually drawn from normal distribution. Looking at the QQ plot of the residuals, the quantiles of residuals line up pretty well with normal quantiles except at the lower tail (perfect correspondence is falling along the line). At the lower tail though, there is immense deviation from the normal quantiles. With such a huge data set, such a large deviation for so many points is extremely unlikely. The end result is that we can expect the estimated confidence intervals (from the normal assumption) will be somewhat off.



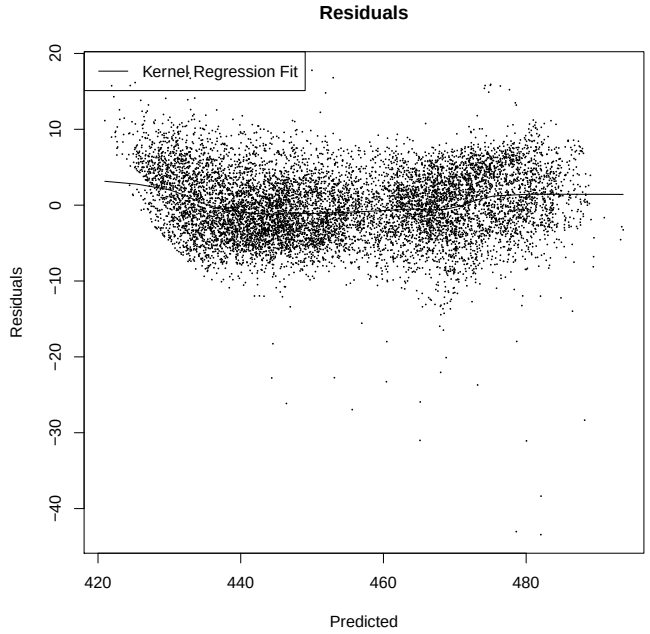
ii. There are unquestionably pretty significant outliers. We can see this by looking at the Cook distance and the studentized residuals. I have plotted a quantile plot of the studentized residuals below. The Bonferroni Critical value for the absolute value of studentized residuals at an alpha of .05, which is the $\frac{1-.05/2}{n}$ quantile of a t distribution with $n-p$ degrees of freedom, is 4.5581. By comparison, looking at the plot, we can see that there are a number of studentized residuals with an absolute value greater than 4.5581, going as high as 9.5778. Since this critical value is conservative, we definitely have a number of outliers.



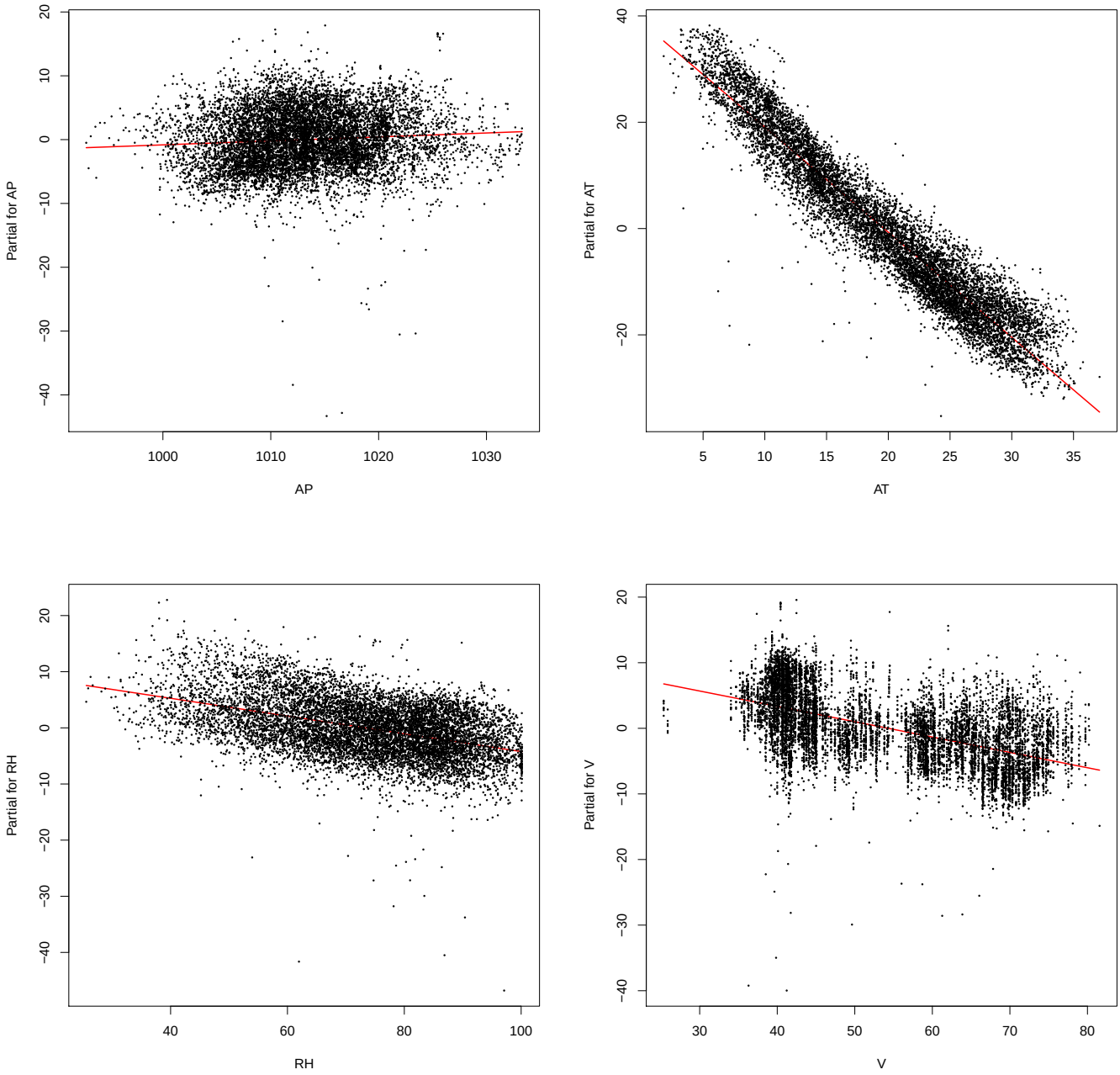
Looking at this another way, we consider the Cook distances, plotted against the half-normal quantiles. We certainly seem to have some Cook distances which stand out; this further supports the notion we're looking at outliers.



iii. The linearity assumption seems to hold up for the most part. The first check is that the residuals really do have mean 0 throughout the range of predictions. Plotting the residuals against the predicted values and fitting a flexible kernel regression model, we do seem to have very close to mean 0, except at the lowest end of the predicted range; the local regression is very close to 0 everywhere else.

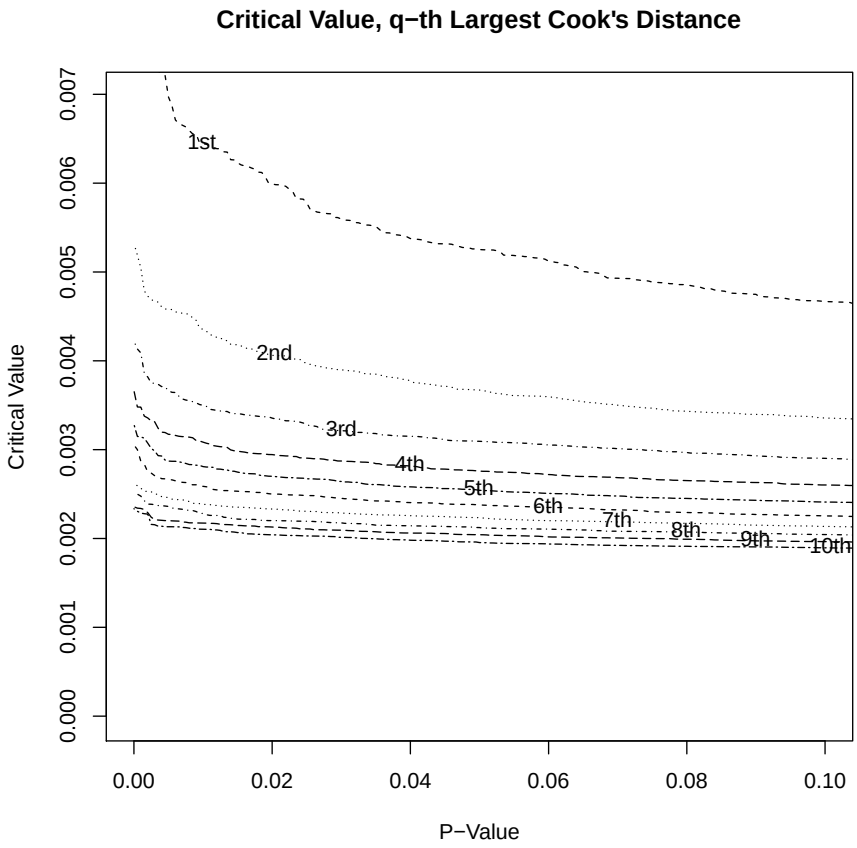


Additionally, checking the partial residual plot of each variable in turn, all of them the linear fits seem to be extremely good. If there isn't a linear relationship between one of the variables and the outcome, the difference isn't much.



b) For both of these statistics, I used a parametric bootstrap methodology to come up with critical values. For each boot, I drew a new set of outcomes Y^*, each one drawn from a normal distribution whose mean was the prediction from the original linear model for the given observation, and whose variance was the estimated variance from the original linear model. Using Y^* and the original X, I fit a new linear model, and then calculated the statistic (Cook Distance or absolute studetnized residual) from this model for all the observations, keeping the q largest values. After this, the critical value for the qth largest value at a p-value of $alpha$ is the $1-\alpha$ percentile of the qth largest values from all the boots. In other words, it was chosen so it is larger than qth largest value from exactly α of the boots.

i. **Summary:** Using the method described above, run for 2000 total boots, I generated critical values for the 1st through 10th largest Cook Distances. We reject the ten largest cook distances as outliers, since they fall above these values.
Work:I have plotted the critical values less than .1 below. Obviously the critical values should asymptotically approach infinity as they go to 0. This is only a finite simulation though, so little should be read into the estimates extremely close to 0.



Comparing the 1st-10th largest Cook Distances from the original model:

```
     1       2       3       4       5       6       7       8       9      10
0.0147  0.0105  0.0103  0.0090  0.0049  0.0040  0.0034  0.0033  0.0032  0.0028
```

To the respective critical values at p-values of .01, .05, and .1:

```
          1       2       3       4       5       6       7       8       9      10
.01  0.0064  0.0044  0.0035  0.0031  0.0028  0.0026  0.0024  0.0023  0.0022  0.0021
.05  0.0053  0.0037  0.0031  0.0028  0.0025  0.0024  0.0022  0.0021  0.0020  0.0020
.1   0.0047  0.0034  0.0029  0.0026  0.0024  0.0023  0.0021  0.0020  0.0020  0.0019
```

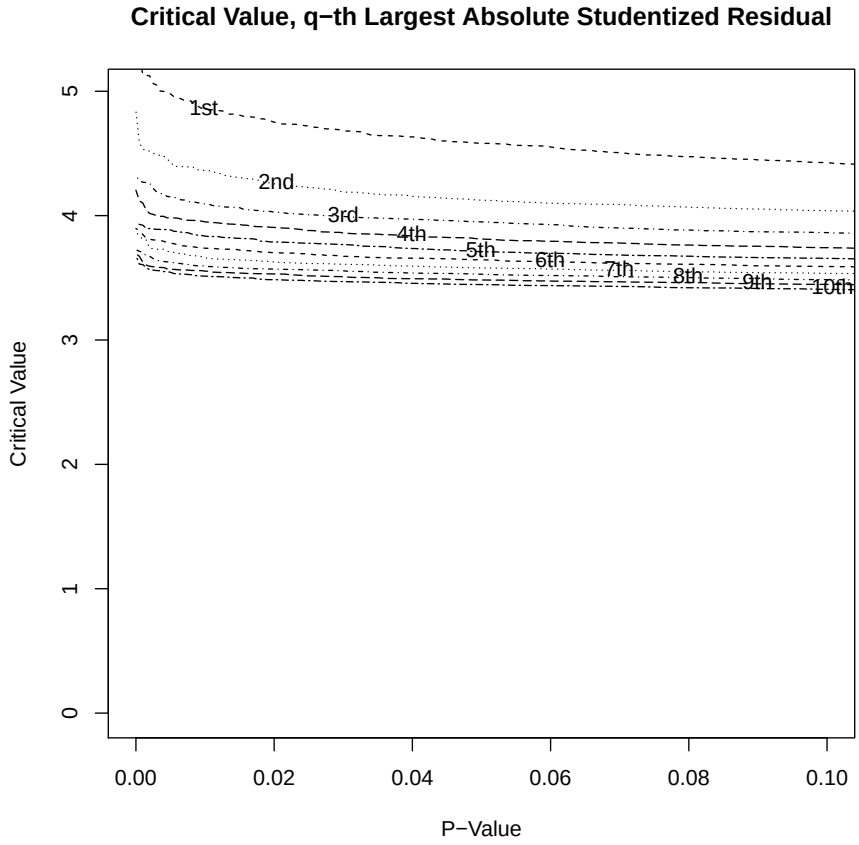
We see that they all fall outside of the critical values for each confidence level, so we would reject them all as outliers for each confidence level.

My R Code:

```
q <- 10
b <- 2000
sigma <- summary(cch.lm)$sigma
Yhat <- predict(cch.lm)
X <- as.matrix(cch[1:4])

# i. using cook distance
cook.boot  <- function(sigma,Yhat,X,q) {
    cooks <- sort(cooks.distance(lm(rnorm(length(Yhat),Yhat,sigma)~
       X)),decreasing=T)
    return(cooks[1:q])
}
cook.mat <- t(replicate(b,cook.boot(sigma,Yhat,X,q)))
```

```
sort.cook.mat <- apply(cook.mat,2,sort)
x <- 1-(1:b)/b
ymax <- sort.cook.mat[ceiling(.995*b),1]
pdf('final/2bi_plot.pdf')
matplot(y=sort.cook.mat,x=x,lty=rep(2:6,2),col=rep(1,10),type="l",
    xlim=c(0,.1),ylim=c(0,ymax),xlab="P-Value",ylab="Critical Value
    ",main="Critical Value, q-th Largest Cook's Distance")
xseq <- seq(.01,.1,.01)
yseq <- rep(NULL,10)
for (i in 1:length(xseq)) {
     yseq[i] <- sort.cook.mat[ceiling((1-xseq[i])*b),i]
}
text(x=xseq,y=yseq,pos=rep(4,10),labels=c("1st","2nd","3rd",paste
    (4:10,"th",sep="")),offset=-.6)
dev.off()
```

ii. **Summary:** Using the method described above, once again for 2000 total boots, but this time for studentized residuals, I have once generated the critical values. We again reject all 10 largest studentized residuals as outliers, since they fall outside these values.
Work: The critical values are plotted below. The same issue for critical values extremely close to 0 exists.



Comparing the absolute value of 1st-10th largest studentized residuals from the original model:

1	2	3	4	5	6	7	8	9	10
9.5778	9.4874	8.4487	6.8353	6.8207	6.2335	5.9248	5.7459	5.7025	5.2078

To the respective critical values at p-values of .01, .05, and .1:

	1	2	3	4	5	6	7	8	9	10
.01	4.8538	4.3641	4.0935	3.9498	3.8352	3.7377	3.6629	3.5958	3.5537	3.5135
.05	4.5815	4.1244	3.9495	3.8115	3.7102	3.6448	3.5793	3.5292	3.4824	3.4456
.1	4.4265	4.0395	3.8636	3.7423	3.6565	3.5908	3.5365	3.4851	3.4471	3.4075

We see that they all fall outside of the critical values for each confidence level, so we would reject them all as outliers for each confidence level. This fits with the simpler Bonferroni Critical value we used before; since it is conservative, and each of the studentized residuals falls above it, it stands to reason a sharper test would reject them to.

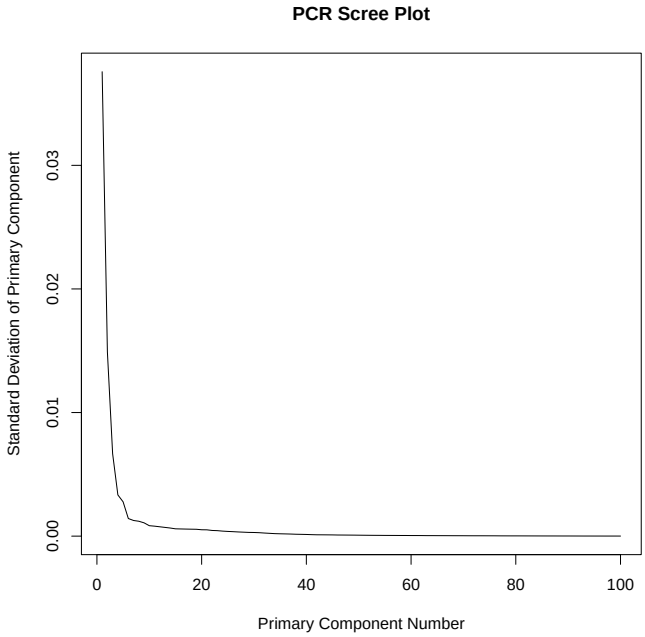
My R Code:

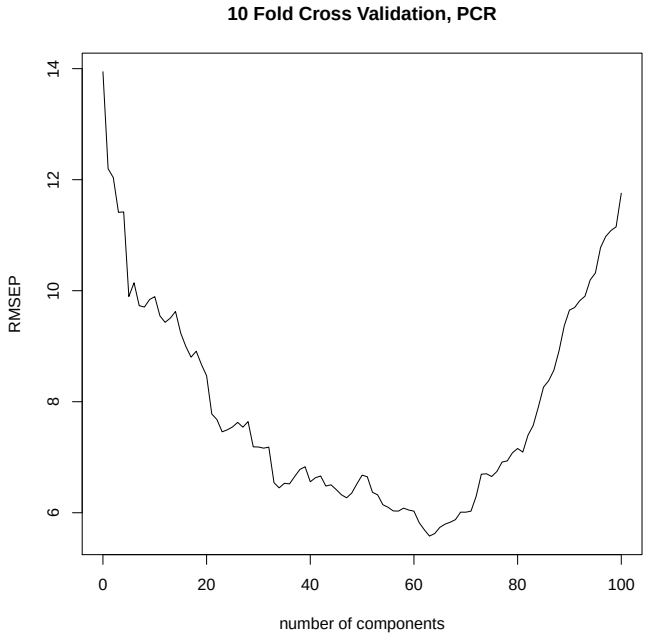
```
rstud.boot <- function(sigma,Yhat,X,q) {
    rstuds <- sort(abs(rstudent(lm(rnorm(length(Yhat),Yhat,sigma)~X
        ))),decreasing=T)
    return(rstuds[1:q])
}
rstud.mat <- t(replicate(b,rstud.boot(sigma,Yhat,X,q)))
sort.rstud.mat <- apply(rstud.mat,2,sort)
x <- 1-(1:b)/b
ymax <- sort.rstud.mat[ceiling(.995*b),1]
pdf('final/2bii_plot.pdf')
matplot(y=sort.rstud.mat,x=x,lty=rep(2:6,2),col=rep(1,10),type="l",
    xlim=c(0,.1),ylim=c(0,ymax),xlab="P-Value",ylab="Critical Value
    ",main="Critical Value, q-th Largest Absolute Studentized
    Residual")
xseq <- seq(.01,.1,.01)
yseq <- rep(NULL,10)
for (i in 1:length(xseq)) {
    yseq[i] <- sort.rstud.mat[ceiling((1-xseq[i])*b),i]
}
text(x=xseq,y=yseq,pos=rep(4,10),labels=c("1st","2nd","3rd",paste
    (4:10,"th",sep="")),offset=-.6)
dev.off()
```

c) From what we've seen, these few extreme points clearly don't fit the assumptions for the simple linear model on the variables we have, which in general seems appropriate for the vast majority of the data. In that sense they are certainly outliers to that model. However, they are not necessarily outliers due to mismeasurement or some other factor such that they should be thrown out as invalid or impossible to include in a linear model. For instance, it may be the case that they represent periods during which their was a mechanical breakdown of some sort. If this were true, we could include them in the model if we had information about mechanical failures at the power plant to include in the model. Then they might well fit the assumptions for a standard linear model.

3. a) Of the shrinkage methods we have discussed, PLS and Lasso are particularly suitable for $n < p$, while PCR is still ok but not as good as the other two. PLS and Lasso are best because they attempt to pick a sparse representation of the predictor space which predicts the outcome well. PLS does this by selecting components of a orthonormal basis in order of their predictiveness, while LASSO, depending at which point it is cut off, adds the original predictors back in as they increase predictive power, keeping the coefficients for the rest at 0. Both of these processes result in a smaller, predictive subspace, effectively reducing p. PCR also shrinks the predictor space, but since it selects the orthogonal components which represent the largest variation in the predictor space, rather than are the most predictive of the outcome, it will likely give a subspace of larger dimension. For instance, in an extreme case, if the largest n primary components don't predict the outcome at all, PCR will fail. Ridge regression is the least suitable option, since it only rarely will out right eliminate a predictor, instead constraining the size of $\|\beta\|_2$ over all predictors.

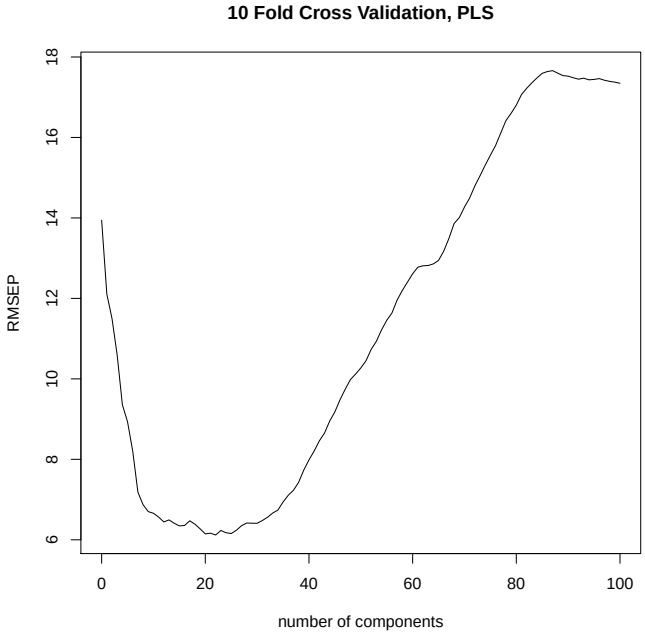
b) I fit a PCR model to the permuted training set, using "pcr" in R, and then selected the number of components from the Scree plot and 10 fold cross validation. I chose 10 fold cross validation for a few reasons. First, with a relatively small number of observations in the test data set, using a smaller k would result in a data set that is small enough to have meaningfully different error properties (since reducing a data set by a set portion increases the prediction error the most when the data set is small). Larger k on the other hand would increasingly look like LOO cross validation. Besides, that though, the choice is somewhat arbitrary, so I went with common wisdom on the subject. The Scree plot and plotted cross validation scores are below:





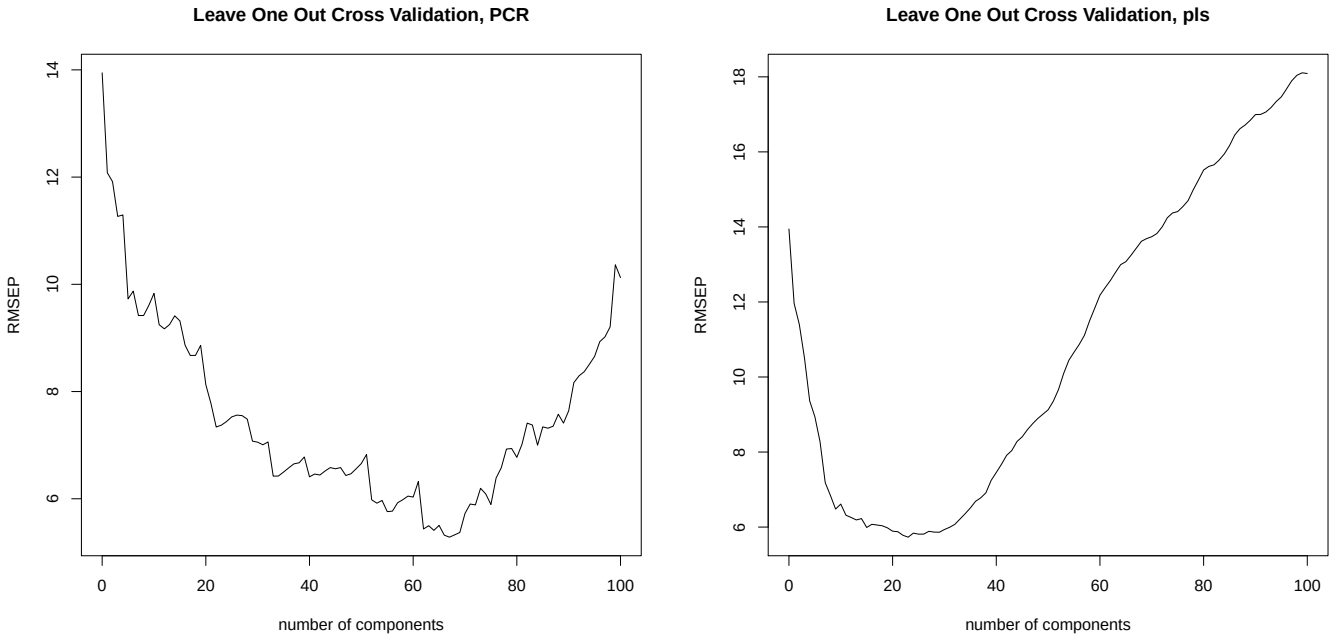
The Scree plot vaguely seemed to have an elbow at 10, where the marginal fall off of the standard deviations for each principal component slowed down, so I tried a model with 10 components. On the test set, the predictions from this model gave a RMSE of 8.69. Going by 10 fold cross validation, including 64 components resulted in the lowest predicted RMSE of 5.58. The actual RMSE of the test data set was 5.65 for this model.

c) I fit a PLS model, using "plsr" in R. I plotted out the 10 fold cross validation scores:



The predicted RMSE was minimized by 23 components, a much smaller number than with PCR, predicting a RMSE of 6.12. In fact, it turned out the actual RMSE on the test data set was 5.15.

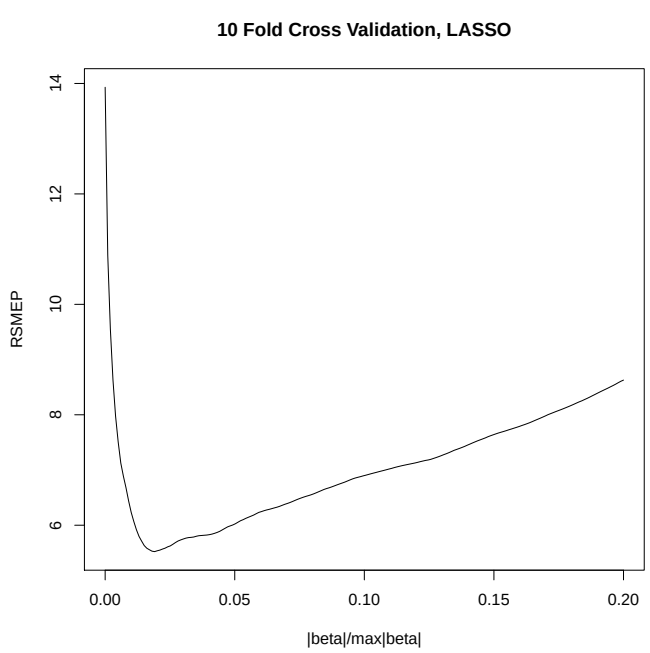
d) Fitting both of these models again, but this time fitting the number of components by LOO cross validation, we get these predictions for RMSE:

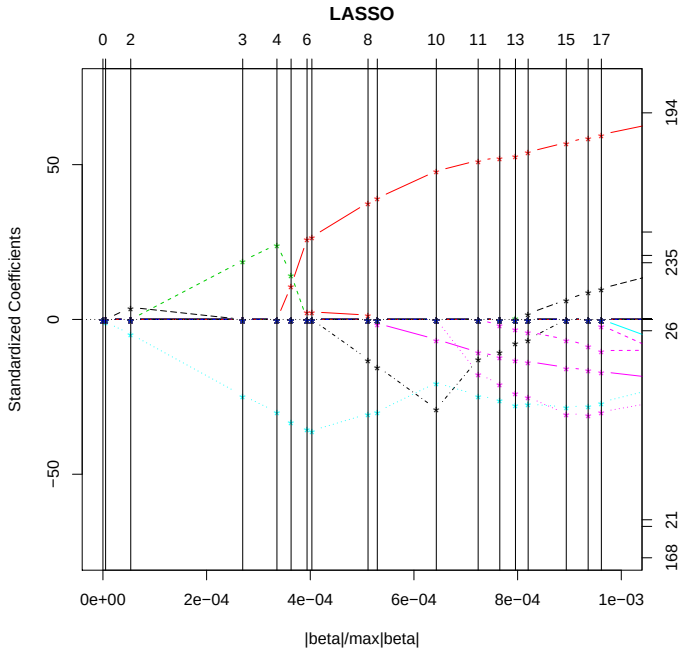


As you can see, they are very similar, though not quite identical, to the results from 10 fold cross validation. For PCR, the minimum is now found at 68 components, with a predicted RMSE of 5.28; this results in an actual RMSE on the test data set of 5.64. For PLS, the minimum is at 24 components, predicting a $RMSE$ of 5.57 compared to an actual RMSE on the test set of 5.26.

Both methods result in pretty similar models with similar performance on the test data set; the PLS model chosen from LOO was a bit worse from the one chosen from 10 fold, but the PCR model was a hair better. The differences are small enough to be regarded as mere noise though. The only really distinction between the methods is that LOO predicted the actual RMSE for PLS far better than 10 fold, though the models end up being almost identical. This is about what I expected; the error in fitting a model fit on $n-1$ data points should be more similar to one fit on n data points than one fit on $\frac{9n}{10}$ data points.

e) I fit a LASSO model using the "lars" command in R. In the same package, there is "cv.lars", which calculates cross validation scores. It subsets the data into K parts and fits a full LASSO model on each combination of $K-1$ of the K subsets. Then, for a set of complexity parameters given by the user, calculates the RMSE of each model on the subset of the data it wasn't fit over. The average of the RMSE calculations from the K models is then the predicted out of sample RMSE. I made use of this, using 10 folds, to choose the complexity parameter. Below is the graph of the RMSE at different complexity parameters, and a small portion of the LASSO plot:





The best complexity parameter according to 10 fold cross validation was $.019 = \frac{\|\beta\|_1}{\max \|\beta\|_1}$, with an estimated RMSE of 5.52. This gave 38 variables with positive coefficients. On the test data set, this model's predictions had a RMSE of 5.33.

Of these three methods, I like PLS the best. It seems better than PCR, since it gives only the components that are actually predictive, so includes far fewer. LASSO results in more predictors, but uses the actual original variables, so it certainly has its value too; I find the pleasant mathematically properties of orthogonal predictors appealing though, even if they are harder to interpret.

f) The biggest advantage of K-fold CV is that, assuming there isn't a simple closed form for LOO CV as with linear smoothers, it is much quicker to run; the model only has to be run K times, instead of n times, saving a lot of computation. It also has the advantage of less variance than LOO. Since each model for LOO is fit on a nearly identical data set, the prediction errors will be highly correlated. On the other hand, each of the models for K fold are for more distinct, making the prediction errors less correlated, and the resulting mean less noisy. LOO CV will produce a less biased estimate of RMSE though; models fit on less data will tend to have somewhat higher error. Thus, identical models fit on n and $n-1$ data points, as with LOO CV, will have a more similar error distribution than identical models fit on n and $n\frac{k-1}{k}$ data points, as with K-fold CV (unless $k = n$).