**4 - ACE & AVAS Procedures – Alternating Conditional Expections (ACE) and Additivity and VAriance Stabilization (AVAS)  
“Automatic” Predictor & Response Transformations**

**4.1 – ACE/AVAS and Additive Models**

Again the general regression problem can be formulated as:

An additive model assumes that the p-dimensional function can be approximated additively, i.e.

assuming no interaction or joint effects. I prefer the formulation of the general OLS regression model that uses the concept of terms ( where these terms can take a variety of function forms.

Here *k* could certainly be larger or smaller than *p*. Again in terms of review and to set the table for the types of models that we will be examining in the next few weeks, here are some potential terms which remember are just functions of the potential predictors,.

etc…

The ACE and AVAS algorithms find optimal transformations “nonparametrically” for the response and all the potential predictors to minimize the RSS or equivalently to maximize the . Although ACE and AVAS are pretty old (1988-ish), they can be useful in finding transformations that can then be “parameterized” based upon the visualizations of the nonparametric smooth estimates. They also use the “backfitting algorithm” which is used in other more *modern* statistical learning methods.

The basic form the ACE and AVAS models is,

and the transformations are found by minimizing the following:

which is the fraction of the variation of not explained by the regression on

We assume without loss of generality that and ’s are standardized to have mean 0 and variance 1. Many of the algorithms we will be considering in the remainder of the course require that all numeric variables have been ***standardized***. For the standardized predictors we have,

and for the standardized response we have,

where,

Thus the following will hold,

.

The data-based version of the proportion of unexplained variation is

and the expectation constraints give,

To help clarify the general ACE algorithm set-up, consider the above constraints applied to an OLS regression,

and

with the OLS regression coefficients from the regression of the mean centered response values on the mean centered predictors. The fraction of unexplained variation is then given by

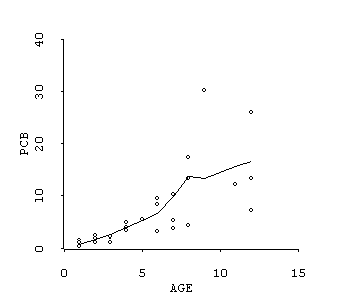
and the usual .

**4.2 - Alternating Conditional Expectations (ACE)**This is an algorithm developed by Breiman and Friedman (1985) for finding the “optimal” transformations that minimize as defined above. To get a feel for the algorithm used to find these optimal transformations, we first consider the case where *p = 1*.

Then reduces to because we are going to assume the transformed response has been standardized to have variance 1, i.e.

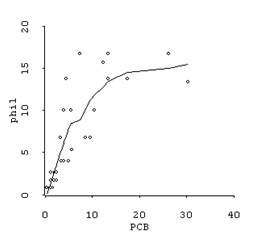
Consider the case where , then . We are then looking for the function with the smallest mean square for predicting . The solution is to take the conditional expectation of given . In other words, is the function *most correlated* with . If we restrict to be a straight line, the solution becomes the OLS estimator,

If we don’t want to impose this much structure on we can estimate it by constructing a scatterplot of and use a scatterplot smoother (e.g. lowess) to estimate the conditional expectation,



Now to estimate the functional form for the response , we consider choosing to minimize,

we choose which can be estimated by smoothing a scatterplot of

  
To make the variance equal to one, simply divide by its standard deviation,

Now you repeat the process of alternating these two conditional expectation procedures until essentially fails to decrease sufficiently. This will yield a sequence of estimates: , , The entire process is demonstrated through the example below.

**Example 4.0** - This example uses data from a study of PCB concentrations found in rainbow trout from the Finger Lakes in NY. Here we seek to relate the concentration of Y = PCB (ppb) found in the tissues of a trout as a function of X = Age (yrs.). I will “manually” perform the ACE algorithm to these data using R. For the scatterplot smoothing necessary to estimate and I will be using the command lowess(x,y,f=0.5). In the end we seek to find the optimal functional forms of both x = Age and y = PCB for the model, .

To begin we set = standardized PCB and = standardized Age. Our first estimate of obtained via a loess smooth.

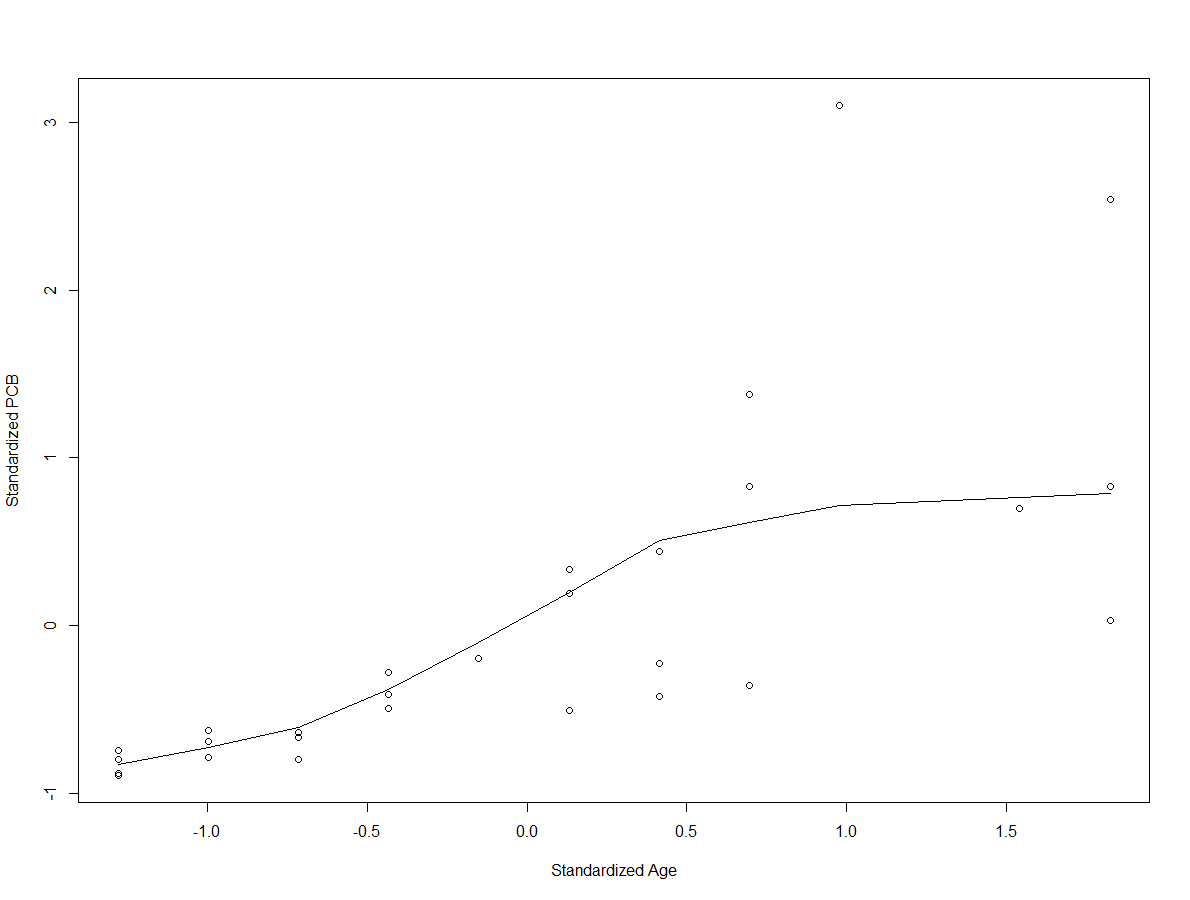
> x = scale(PCBtrout$Age)

> y = scale(PCBtrout$PCB)

> plot(x,y,xlab="Standardized Age",ylab="Standardized PCB")

> phi1 = lowess(x,y,f=0.5)

> lines(phi1)

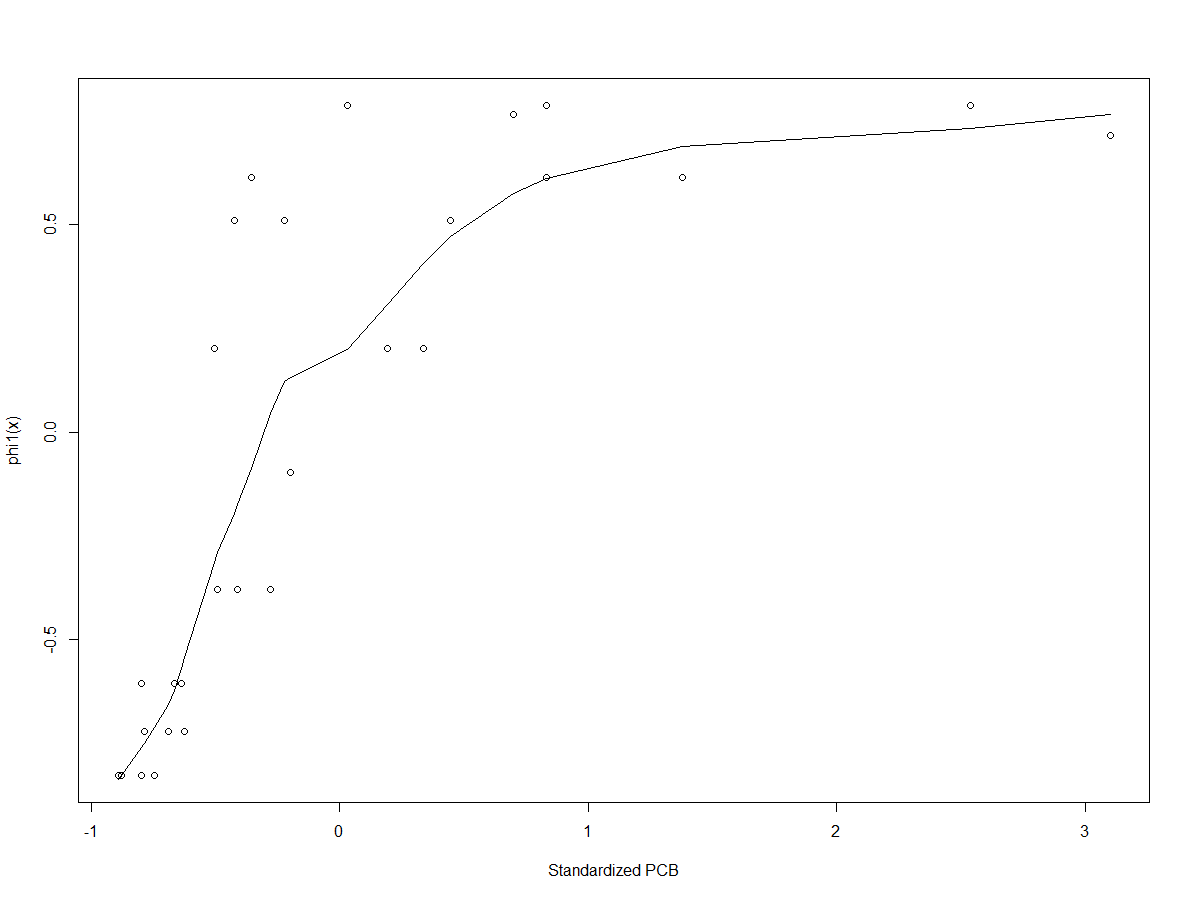


Save the current estimate of .

> phi1 = phi1$y 🡨 this extracts the result of the lowess smoother  
  
We then estimate by again using a lowess smooth of the plot, .  
  
> plot(y,phi1$y,xlab="Standardized PCB",ylab="phi1(x)")

> theta1 = lowess(y,phi1$y,f=0.5)

> lines(theta1)



After saving the current estimate of we can then fit the SLR regression model using our current estimates

> theta1 = theta1$y

> first.lm = lm(theta1~phi1)

> summary(first.lm)

Call:

lm(formula = theta1 ~ phi1)

Residuals:

Min 1Q Median 3Q Max

-0.24424 -0.05797 0.01428 0.06239 0.19686

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.06480 0.01929 -3.36 0.00242 \*\*

phi1 0.86324 0.03101 27.84 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.102 on 26 degrees of freedom

Multiple R-squared: 0.9675, Adjusted R-squared: 0.9663

F-statistic: 774.8 on 1 and 26 DF, p-value: < 2.2e-16

Now repeat the process by alternating the conditional expectations,

and saving the resulting estimate in phi2.

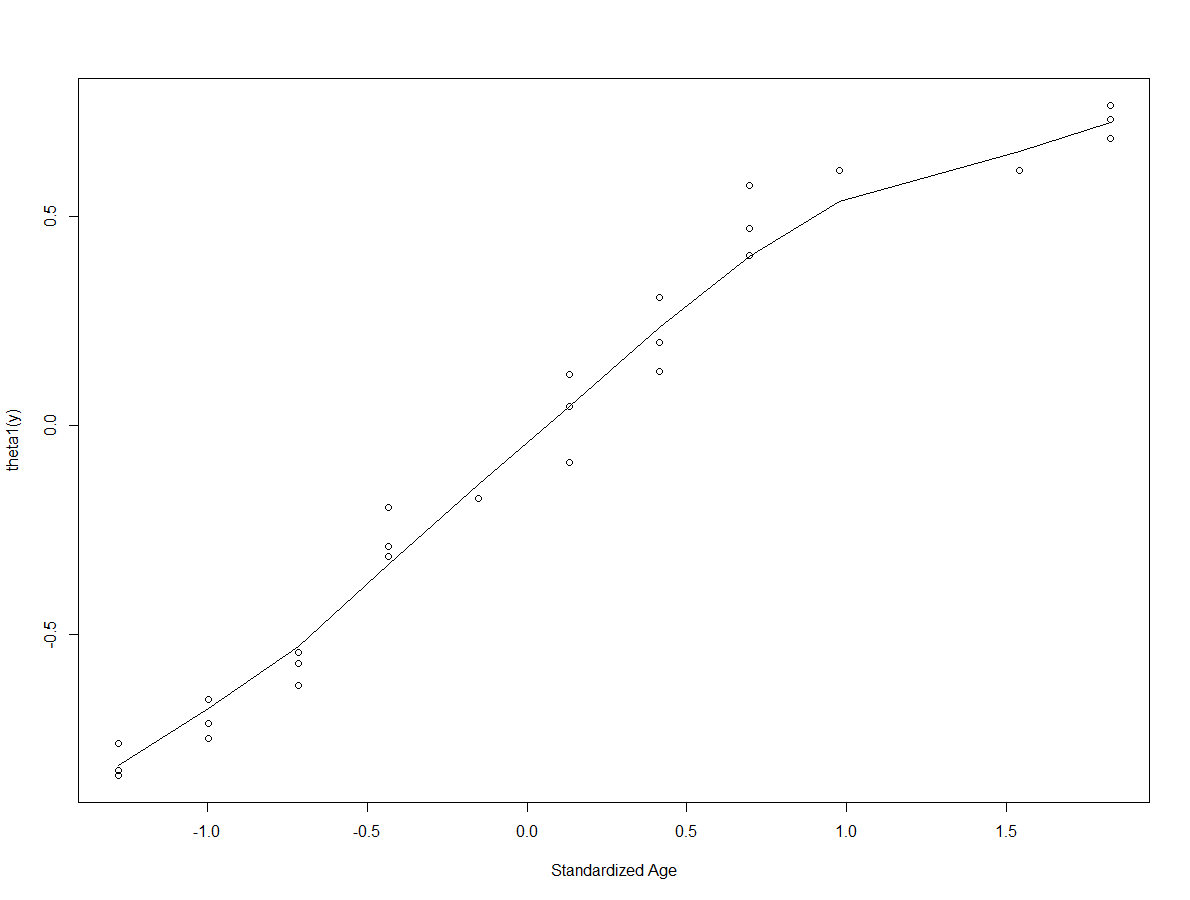
**Second iteration**

> plot(x,theta1,xlab="Standardized Age",ylab="theta1(y)")

> phi2 = lowess(x,theta1,f=0.5)

> lines(phi2)

> phi2 = phi2$y



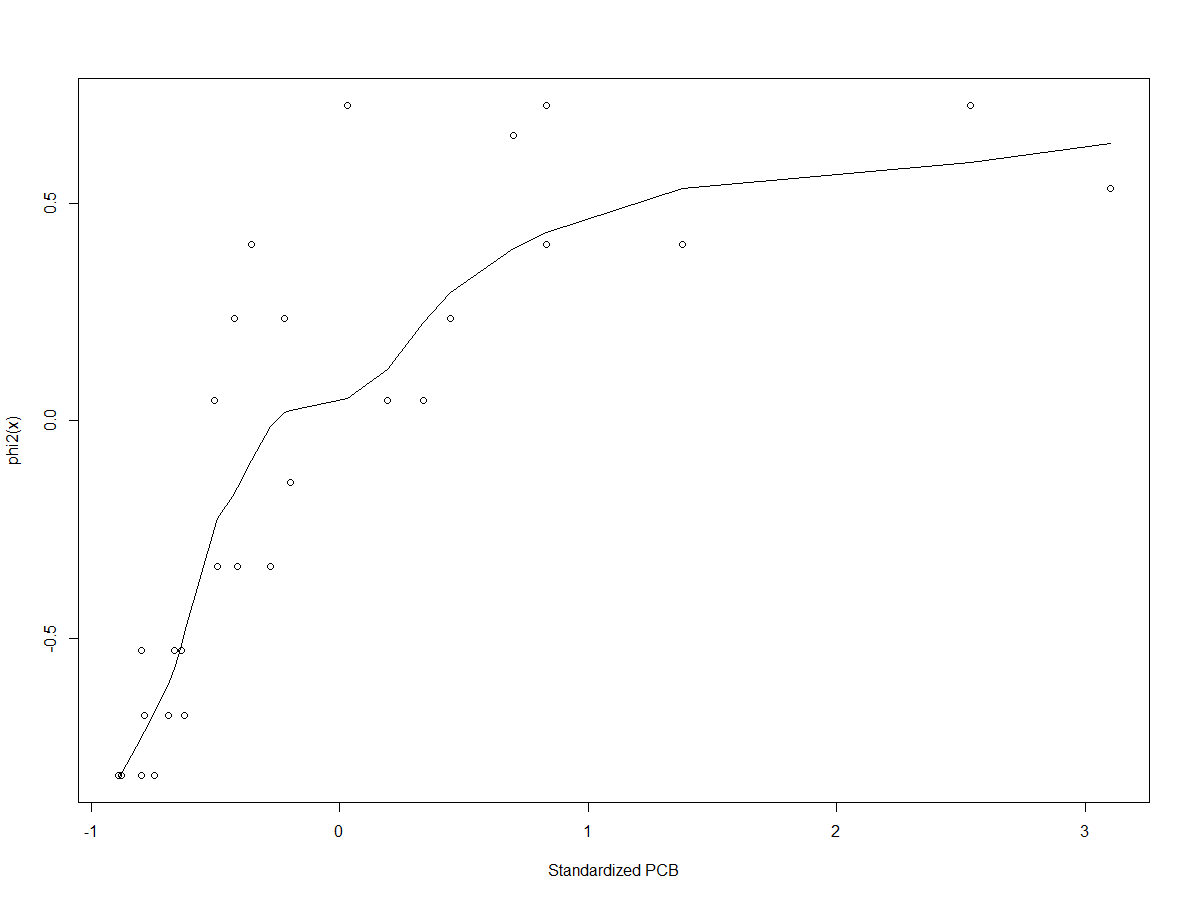
Now we estimate and save it as theta2.

> plot(y,phi2,xlab="Standardized PCB",ylab="phi2(x)")

> theta2 = lowess(y,phi2,f=0.5)

> lines(theta2)

> theta2 = theta2$y



Again we can fit the SLR model using our current estimates.

> second.lm = lm(theta2~phi2)

> summary(second.lm)

Call:

lm(formula = theta2 ~ phi2)

Residuals:

Min 1Q Median 3Q Max

-0.121906 -0.062112 0.004751 0.030263 0.174116

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.05622 0.01384 -4.061 0.000398 \*\*\*

phi2 0.85461 0.02561 33.374 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.07217 on 26 degrees of freedom

Multiple R-squared: 0.9772, Adjusted R-squared: 0.9763

F-statistic: 1114 on 1 and 26 DF, p-value: < 2.2e-16

We can then repeat this process until we obtain convergence (i.e. minimal changes in the ).

**Third iteration**  
> plot(x,theta2,xlab="Standardized Age",ylab="theta2(y)")

> phi3 = lowess(x,theta2,f=0.5)

> lines(phi3)

> phi3 = phi3$y

> plot(y,phi3,xlab="Standardized PCB",ylab="phi3(x)")

> theta3 = lowess(y,phi3,f=0.5)

> lines(theta3)

> theta3 = theta3$y

> third.lm = lm(theta3~phi3)

> summary(third.lm)

Call:

lm(formula = theta3 ~ phi3)

Residuals:

Min 1Q Median 3Q Max

-0.091701 -0.044778 0.002676 0.030486 0.137198

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.04631 0.01182 -3.918 0.000578 \*\*\*

phi3 0.86368 0.02489 34.702 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.05973 on 26 degrees of freedom

Multiple R-squared: 0.9789, Adjusted R-squared: 0.9781

F-statistic: 1204 on 1 and 26 DF, p-value: < 2.2e-16

**Fourth iteration**

> plot(x,theta3,xlab="Standardized Age",ylab="theta3(y)")

> phi4 = lowess(x,theta3,f=0.5)

> lines(phi4)

> phi4 = phi4$y

> plot(y,phi4,xlab="Standardized PcB",ylab="phi4(x)")

> theta4 = lowess(y,phi4,f=0.5)

> lines(theta4)

> theta4 = theta4$y

> fourth.lm = lm(theta4~phi4)

> summary(fourth.lm)

Call:

lm(formula = theta4 ~ phi4)

Residuals:

Min 1Q Median 3Q Max

-0.073198 -0.041479 0.000875 0.033786 0.115178

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.04584 0.01042 -4.398 0.000165 \*\*\*

phi4 0.87287 0.02438 35.805 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0504 on 26 degrees of freedom

Multiple R-squared: 0.9801, Adjusted R-squared: 0.9794

F-statistic: 1282 on 1 and 26 DF, p-value: < 2.2e-16

**Fifth iteration**

> plot(x,theta4,xlab="Standardized Age",ylab="theta4(y)")

> phi5 = lowess(x,theta4,f=0.50)

> lines(phi5)

> phi5 = phi5$y

> plot(y,phi5,xlab="Standardized PCB",ylab="phi5(x)")

> theta5 = lowess(y,phi5,f=0.50)

> lines(theta5)

> theta5 = theta5$y

> fifth.lm = lm(theta5~phi5)

> summary(fifth.lm)

Call:

lm(formula = theta5 ~ phi5)

Residuals:

Min 1Q Median 3Q Max

-0.06467 -0.03415 -0.00100 0.03232 0.09836

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.046321 0.009511 -4.87 4.73e-05 \*\*\*

phi5 0.878757 0.024000 36.61 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.04318 on 26 degrees of freedom

Multiple R-squared: 0.981, Adjusted R-squared: 0.9802

F-statistic: 1341 on 1 and 26 DF, p-value: < 2.2e-16

**Sixth iteration**

> plot(x,theta5,xlab="Standardized Age",ylab="theta5(y)")

> phi6 = lowess(x,theta5,f=0.5)

> lines(phi6)

> phi6 = phi6$y

> plot(y,phi6,xlab="Standarized PCB",ylab="phi6(x)")

> theta6 = lowess(y,phi6,f=0.5)

> lines(theta6)

> theta6 = theta6$y

> sixth.lm = lm(theta6~phi6)

> summary(sixth.lm)

Call:

lm(formula = theta6 ~ phi6)

Residuals:

Min 1Q Median 3Q Max

-0.058167 -0.028804 -0.001593 0.028828 0.085349

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -0.045941 0.009009 -5.10 2.58e-05 \*\*\*

phi6 0.882547 0.023907 36.91 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

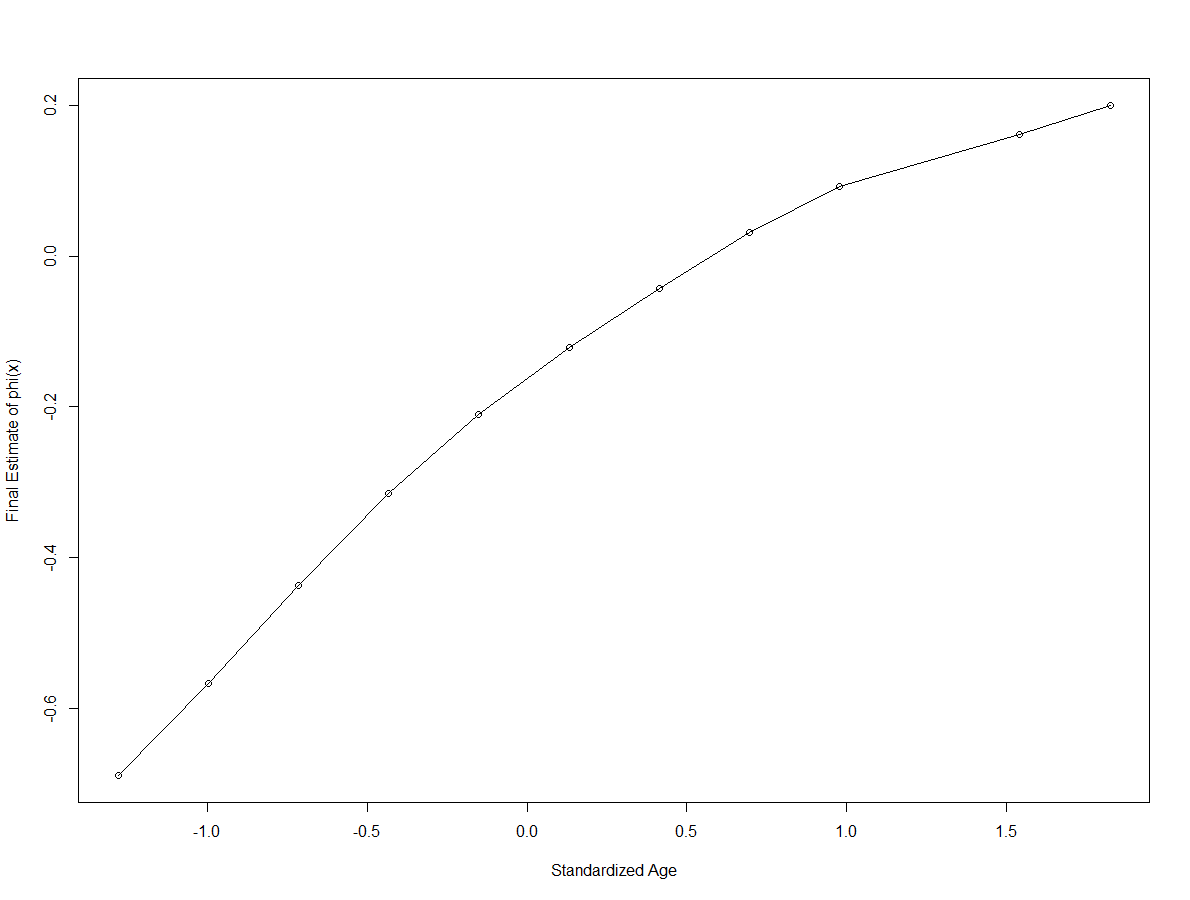
Residual standard error: 0.03767 on 26 degrees of freedom

Multiple R-squared: 0.9813, Adjusted R-squared: 0.9806

F-statistic: 1363 on 1 and 26 DF, p-value: < 2.2e-16

Final Estimate of

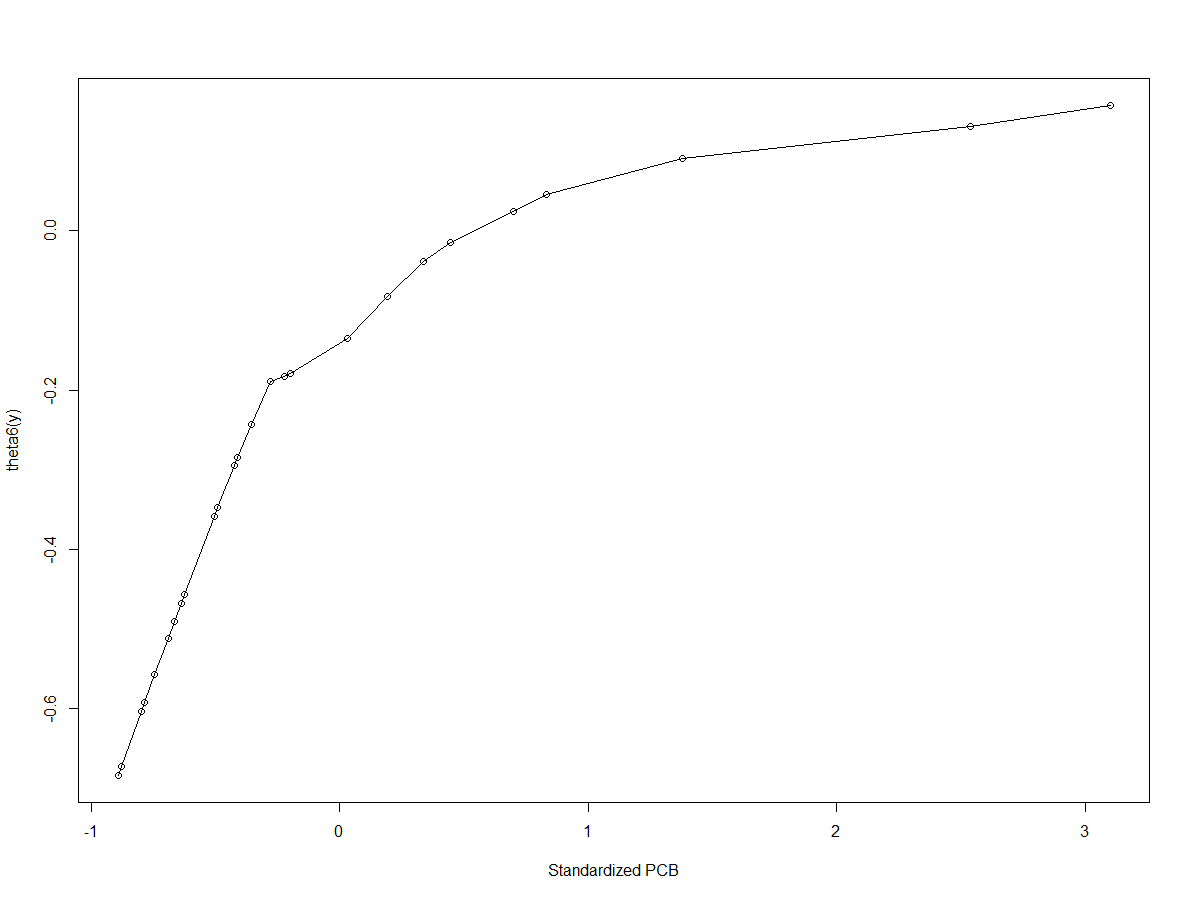
> plot(sort(x),sort(phi6),xlab="Standardized Age",ylab="Final Estimate of phi(x)")

> lines(sort(x),sort(phi6))  


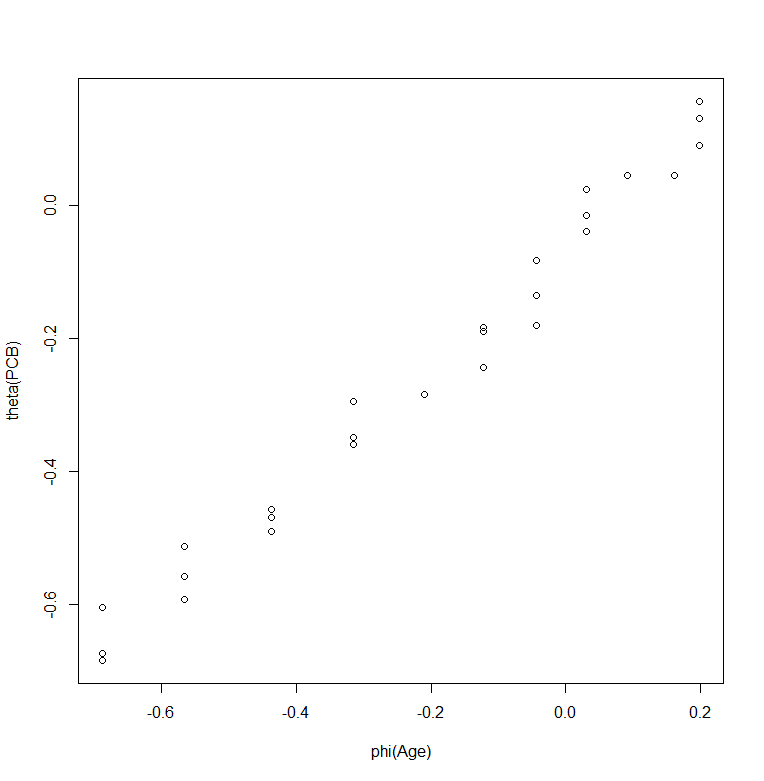
Final Estimate of

> plot(sort(y),sort(theta6$y),xlab="Standardized PCB",ylab="theta6(y)")

> lines(sort(y),sort(theta6$y))



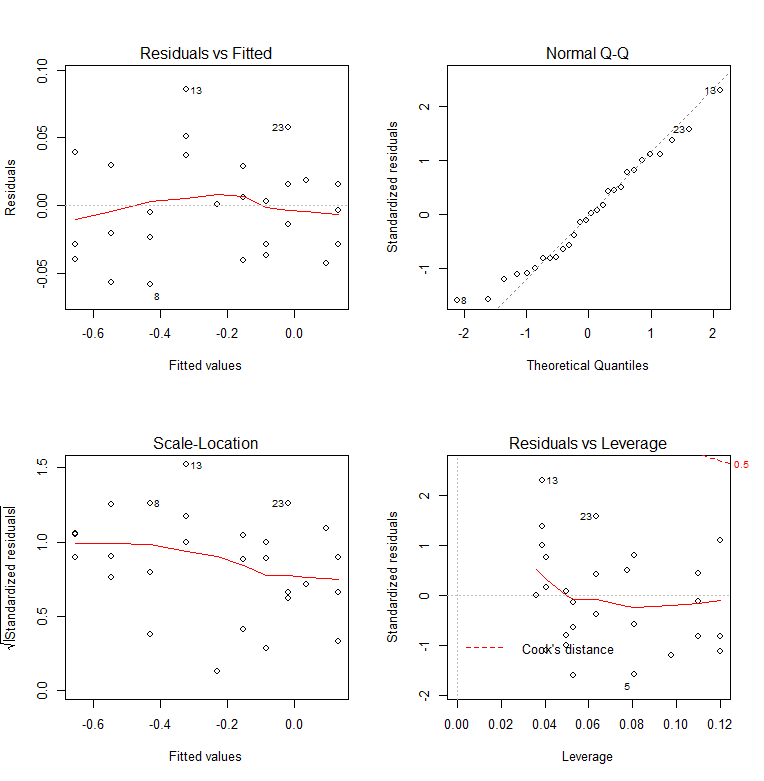
Plot of Final -



> sixth.lm = lm(theta6~phi6)

> par(mfrow=c(2,2))

> plot(sixth.lm)



The quality of the resulting fit is very good. However other than the visualization of the final estimated functional forms of both PCB concentration (ppb) and the Age (yrs.) of the trout we have no parametric form for them such as the logarithm or square root. However if you consider the nonparametric estimates of and you could decide that and . Fitting both of these models however does not achieve the lofty obtained by using the nonparametric estimates with .

**4.3 – ACE and AVAS with Multiple Predictors**

When we have multiple numeric predictors ( the ACE algorithm can also be used fit the additive model . The process for predictors is generalized in the algorithm below.

**ACE Algorithm - uses what is known as the *Backfitting Algorithm***

**Step 1** Set

**Step 2** For *k = 1* to *p* set

and replace with . After cycling through each predictor, check if

decreases (i.e. increases), if yes repeat the loop. If no, go to **Step 3**.

**Step 3** Set

Repeat **Step 2** with and **Step 3**. Check if decreases (i.e. increases), if   
 yes, loop again. If no, go to **Step 4**.

**Step 4** The current versions of are the final solutions

Notice that each step in the ACE algorithm requires estimation of a conditional expectation. How is this done? We generally use a locally weighted smoother, e.g. a lowess smoother as in the example above. One problem with these smoothers is that they tend not to robust to outliers, thus outliers and influential points can create problems for the ACE algorithm. To estimate the variance smooth we smooth the squared residuals vs. y.

**AVAS**

I will not present the details of AVAS here, but it has an additional step in the process of estimating the transformation of , , so that it also stabilizes the variation, i.e.

.

**4.4 – Examples of ACE and AVAS**

ACE and AVAS routines are contained in the acepack library from CRAN. Both ACE and AVAS take a single predictor x or a matrix of predictors X if we have numeric predictors and the response y as arguments.

> ace.mod = ace(X,y)

> avas.mod = avas(X,y)

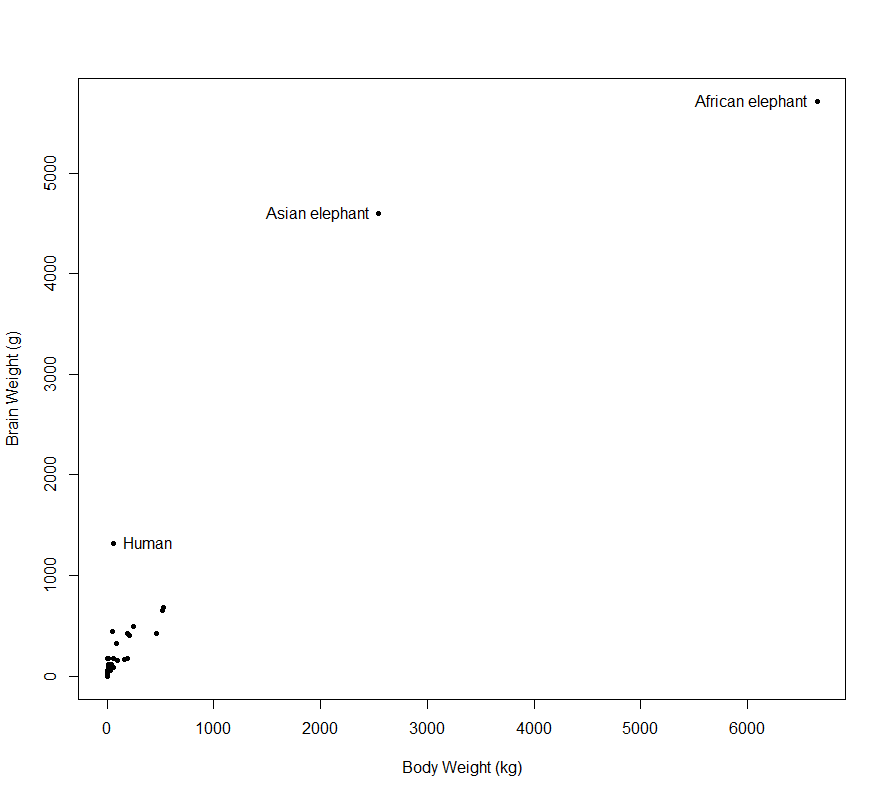
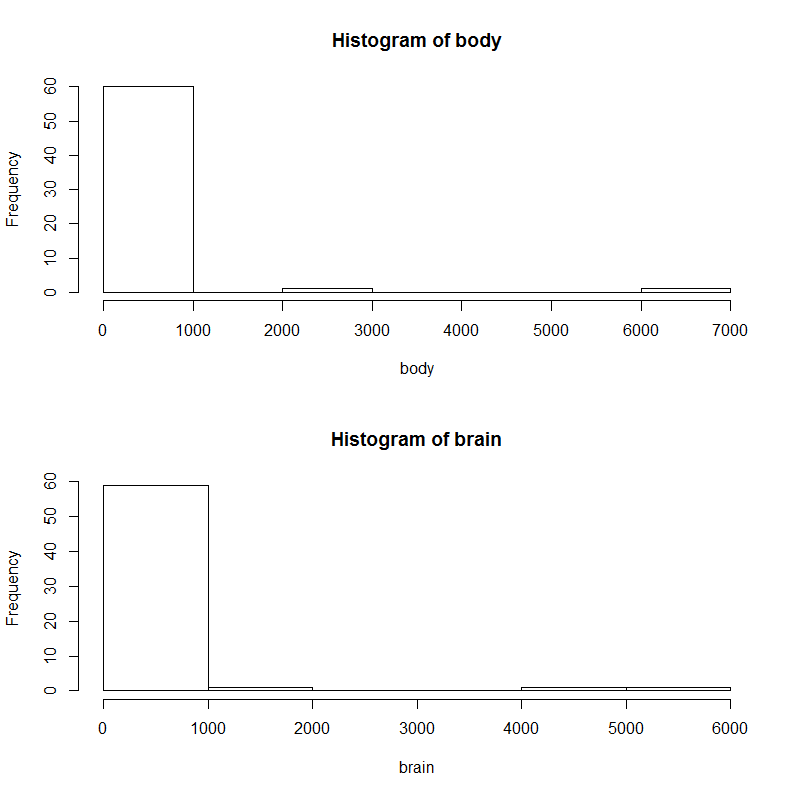
**Example 4.1 – Brain Weight and Body Weight of Mammals (p = 1)**

This is a simple example to illustrate the use ACE and AVAS on a small data set in which we are interested in modeling the brain weight of a variety of mammals as a function of their typical body weight. These data are contained in the data frame mammals in the data set R directory I have shared with you.

> attach(Mammals)  
> layout(matrix(c(1,2,1,3),2,2,byrow=T))

> plot(body,brain,xlab="Body Weight (kg)",ylab="Brain Weight (g)",pch=20)

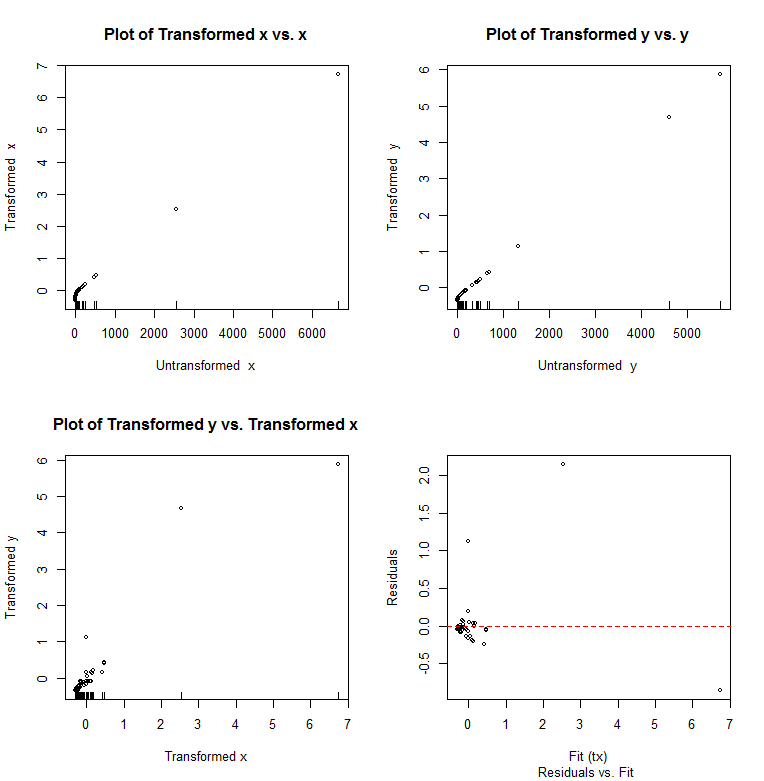
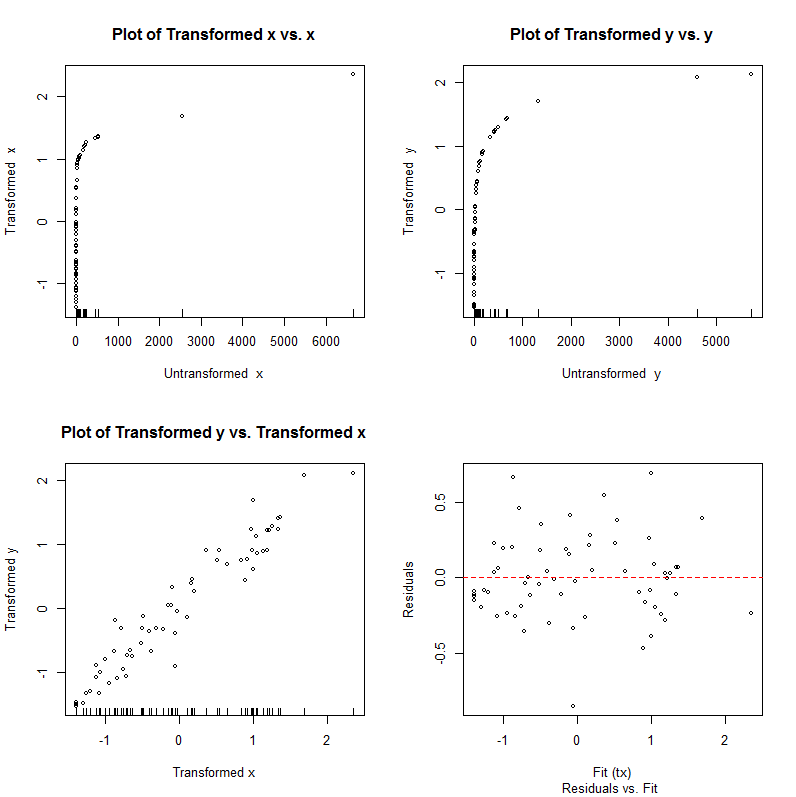
> identify(body,brain,row.names(Mammals))  
> hist(body)  
> hist(brain)

> ace.brain = ace(body,brain)

> saceplot(body,brain,ace.brain) 🡨 function I wrote to display the results of ACE/AVAS

ACE AVAS

> avas.brain = avas(body,brain)

> saceplot(body,brain,avas.brain)

Clearly both variables have markedly skewed right distributions, thus using transformation guidelines we might consider taking the logarithm of both variables.

> log.body = log(body)

> log.brain = log(brain)

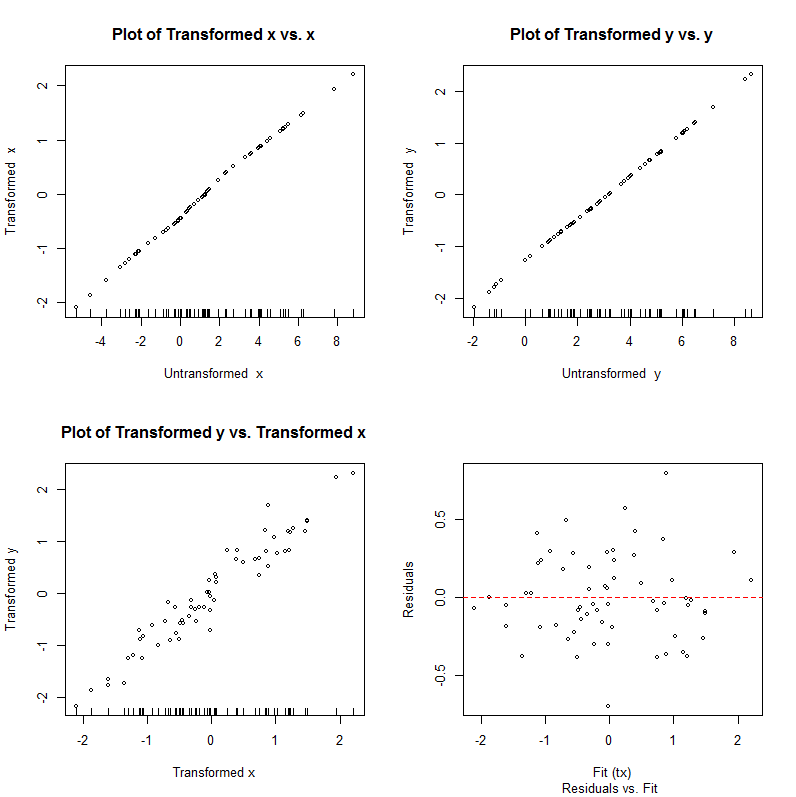
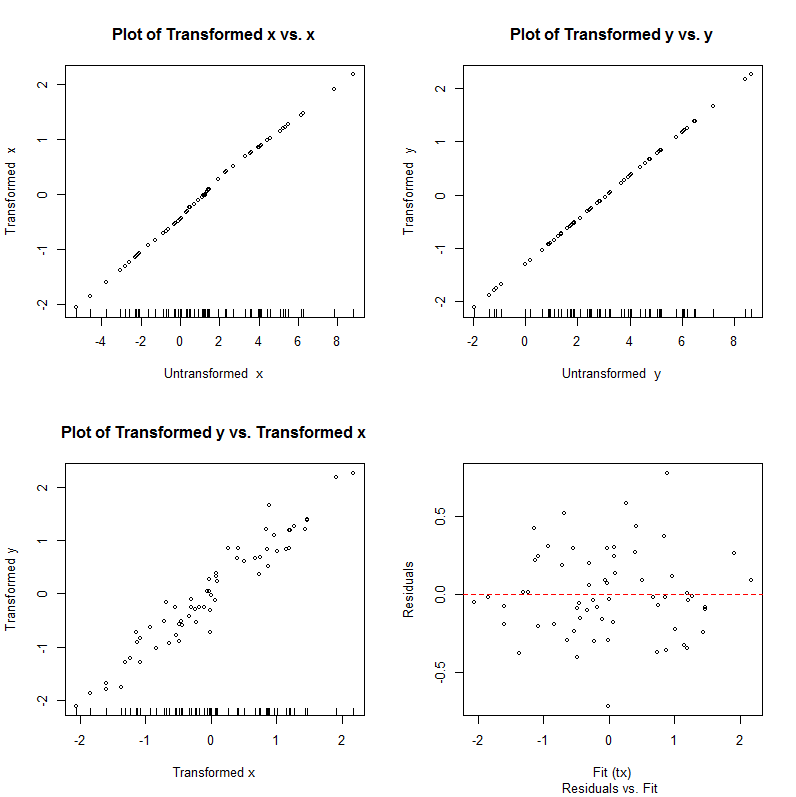
> plot(log.body,log.brain)

> ace.log = ace(log.body,log.brain)

> saceplot(log.body,log.brain,ace.log)

> avas.log = avas(log.body,log.brain)

> saceplot(log.body,log.brain,avas.log)

ACE AVAS  
 

Both ACE and AVAS leave the logarithms of body and brain untransformed, confirming that these parametric transformations are essentially optimal. This example illustrates two important things: (1) AVAS generally is more adept than ACE at finding the optimal transformations and is more resistant to outliers/influential points and (2) these methods are not a substitute for carefully consideration of the distributions (marginal or joint) of the response and the potential predictors.

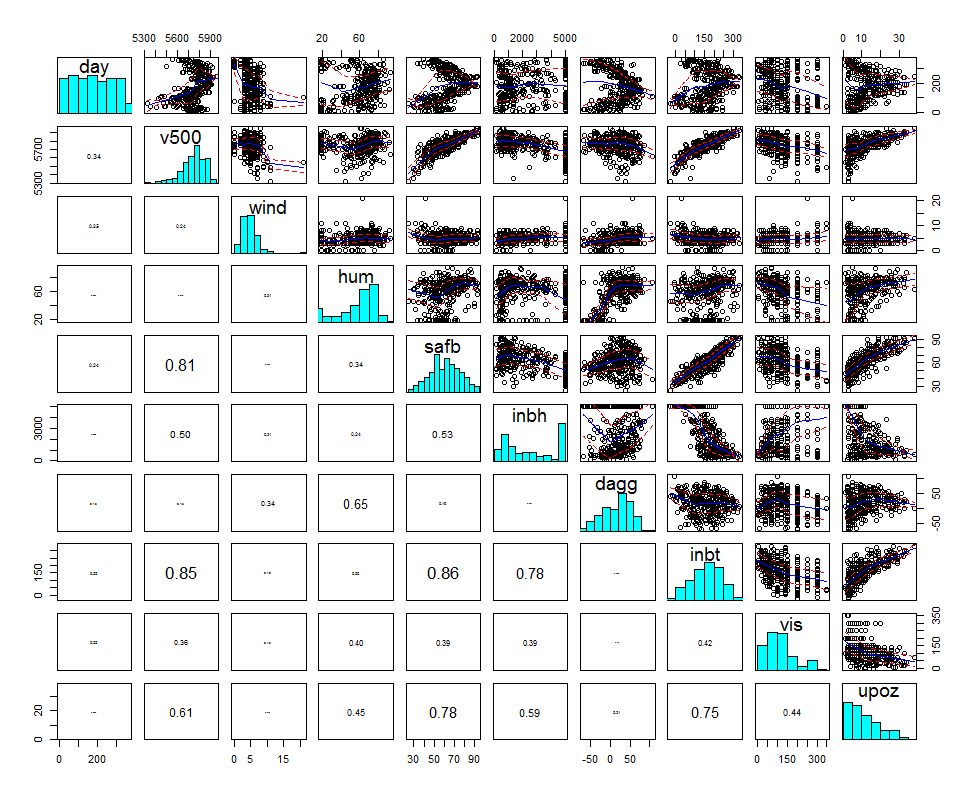
**Example 4.2 – Upper Ozone Concentration in Los Angeles Basin (p > 1)**

Ozone is one of the nasty constituents of photochemical smog, and its concentration is the standard indicator of the severity of such smog. If its level is high, a smog alert is called. Ozone is not emitted directly into the atmosphere. Rather, it is a product of chemical reactions that require solar radiation and emissions of primary pollutants from smoke stacks and automobile exhaust. When the ventilation of the atmosphere is low, the chemical reactions bring ozone to high levels. Low ventilation occurs when wind speeds are low and temperature is high because on hot, calm days in the summer the atmosphere cannot cleanse itself. The goal in the analysis of these data is to determine how ozone depends on the other variables. A brief description of the variables and their abbreviations is given below:

* **safb** – Sandburg Air Force Base temperature (Co)
* **inbh** – Inversion base height (ft.)
* **dagg** – Daggett pressure gradient (mmHg)
* **vis** – Visibility (miles)
* **v500** – Vandenburg 500 millibar height (m)
* **hum** – Humidity (%)
* **inbt** – Inversion base temperature (Fo)
* **wind** – Wind speed (mph)
* **upoz** – Upper ozone concentration (ppm) 🡨 response (Y)

We begin our analysis by examining a scatterplot matrix of the data. It appears that upper ozone concentration is related to several of the predictors, although not linearly. We also see that several of the variables are correlated with one another and some of the relationships between the predictors are nonlinear in appearance.

> pairs.plus(Ozdata)



We begin by fitting a OLS regression model using all available predictors.

> lm.oz1 = lm(upoz~.,data=Ozdata)

> summary(lm.oz1)

Call:

lm(formula = upoz ~ ., data = Ozdata)

Residuals:

Min 1Q Median 3Q Max

-12.2407 -2.8832 -0.3353 2.7409 13.3523

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 18.3792938 29.5045242 0.623 0.53377

day -0.0088490 0.0027199 -3.253 0.00126 \*\*

v500 -0.0051340 0.0053950 -0.952 0.34200

wind -0.0198304 0.1238829 -0.160 0.87292

hum 0.0804923 0.0188345 4.274 2.54e-05 \*\*\*

safb 0.2743349 0.0497361 5.516 7.17e-08 \*\*\*

inbh -0.0002497 0.0002950 -0.846 0.39798

dagg -0.0036968 0.0112925 -0.327 0.74360

inbt 0.0292640 0.0136115 2.150 0.03231 \*

vis -0.0080742 0.0037565 -2.149 0.03235 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

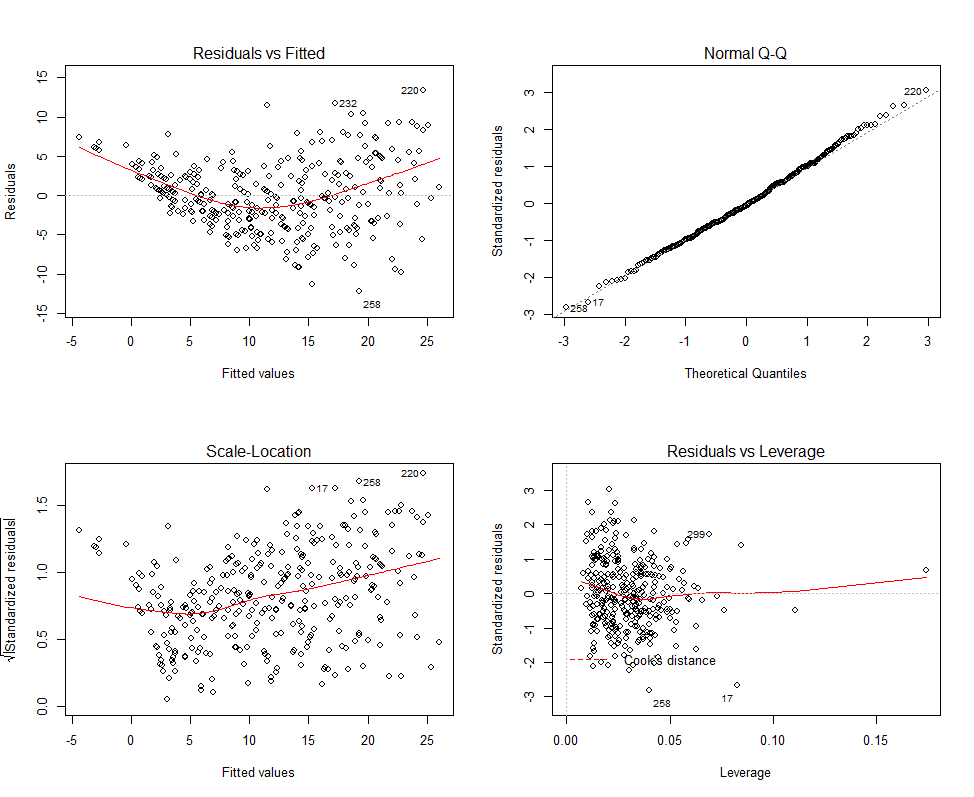
Residual standard error: 4.441 on 320 degrees of freedom

Multiple R-squared: 0.7011, Adjusted R-squared: 0.6927

F-statistic: 83.4 on 9 and 320 DF, p-value: < 2.2e-16

> par(mfrow=c(2,2))

> plot(lm.oz1)



> par(mfrow=c(1,1))

> X = model.matrix(upoz~.,data=Ozdata)[,-1]

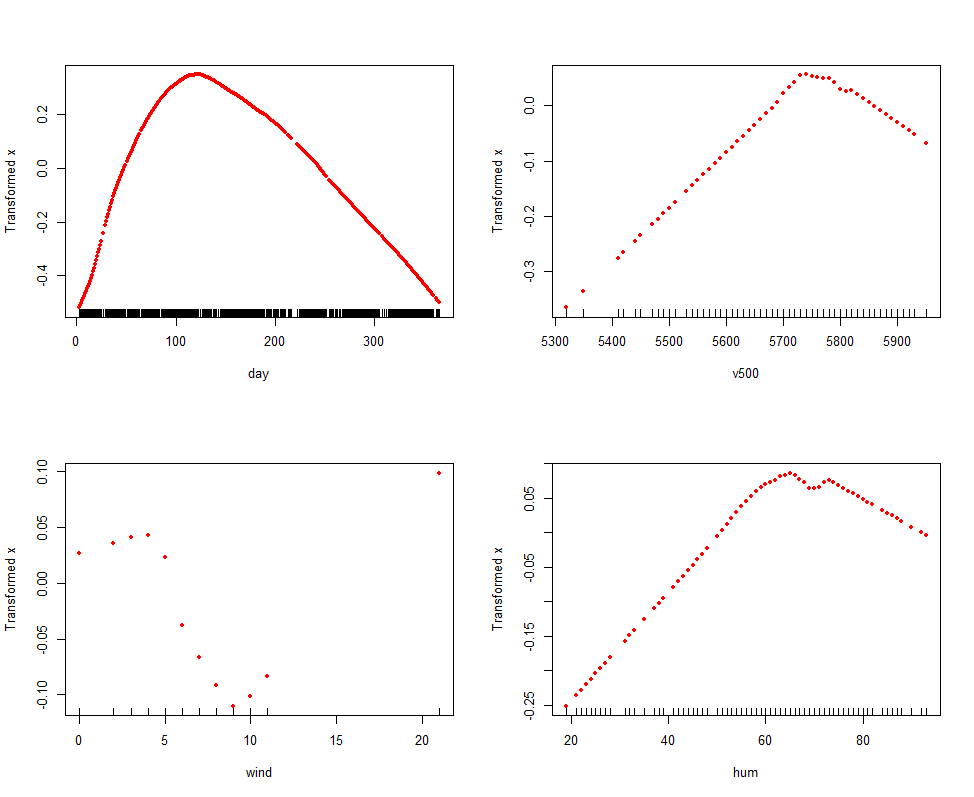
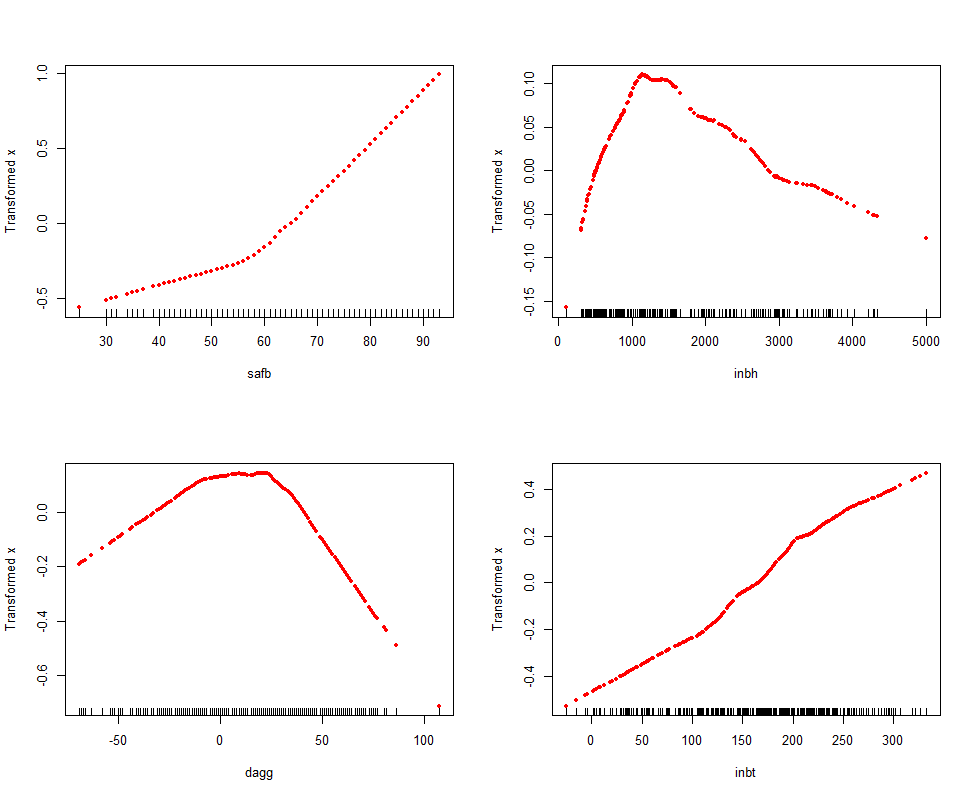
> y = Ozdata$upoz

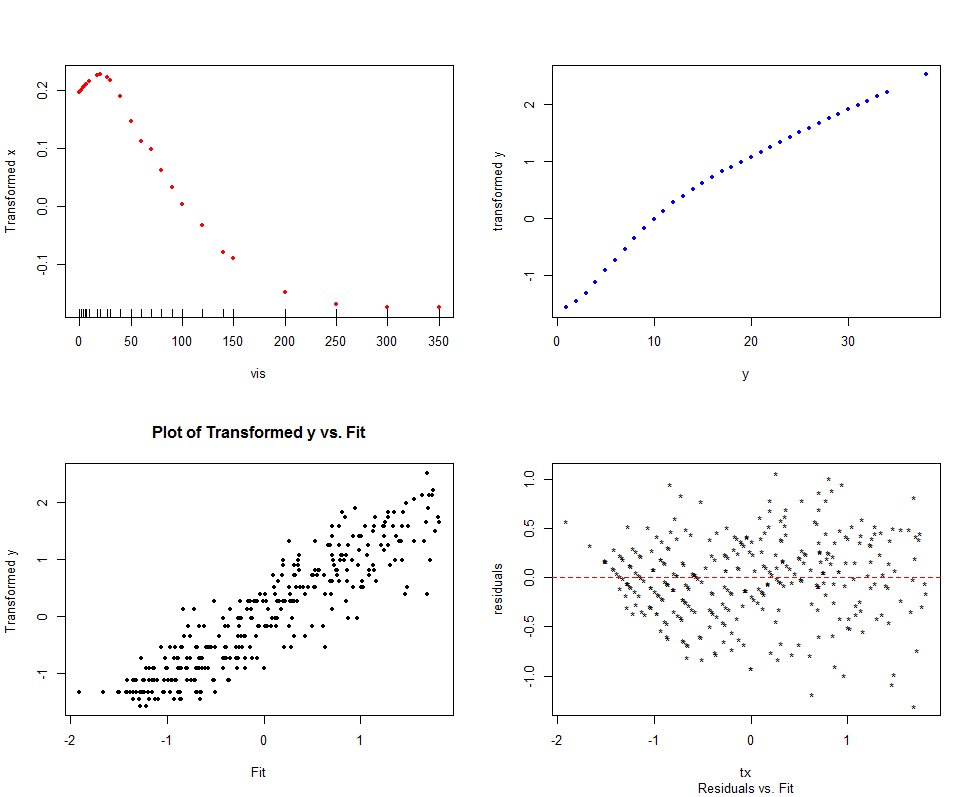
> ace.oz1 = ace(X,y)

> ace.oz1$rsq

[1] 0.8330531

> maceplot(X,y,ace.oz1) 🡨 function I wrote to view ACE/AVAS results (p > 1)



> lm.oz2 = lm(upoz^.333~hum+wind+v500+poly(safb,2)+poly(dagg,2)+poly(inbh,2)+poly(vis,2)+poly(day,3))

> summary(lm.oz2)

Call:

lm(formula = upoz^0.333 ~ hum + wind + v500 + poly(safb, 2) +

poly(dagg, 2) + poly(inbh, 2) + poly(vis, 2) + poly(day,3))

Residuals:

Min 1Q Median 3Q Max

-0.68579 -0.14443 0.01969 0.14317 0.53943

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -4.1369810 1.5897082 -2.602 0.009696 \*\*

hum 0.0021265 0.0010620 2.002 0.046092 \*

wind -0.0169285 0.0064344 -2.631 0.008934 \*\*

v500 0.0010868 0.0002741 3.965 9.10e-05 \*\*\*

poly(safb, 2)1 2.6338054 0.5457456 4.826 2.17e-06 \*\*\*

poly(safb, 2)2 0.6124786 0.2345044 2.612 0.009438 \*\*

poly(dagg, 2)1 -1.4198366 0.4239539 -3.349 0.000909 \*\*\*

poly(dagg, 2)2 -1.8937203 0.2600519 -7.282 2.65e-12 \*\*\*

poly(inbh, 2)1 -1.1140532 0.3174524 -3.509 0.000515 \*\*\*

poly(inbh, 2)2 -0.8988258 0.2447373 -3.673 0.000282 \*\*\*

poly(vis, 2)1 -1.3954589 0.2761436 -5.053 7.37e-07 \*\*\*

poly(vis, 2)2 0.8477931 0.2382359 3.559 0.000430 \*\*\*

poly(day, 3)1 -1.7636597 0.2696610 -6.540 2.48e-10 \*\*\*

poly(day, 3)2 -3.7200745 0.4213109 -8.830 < 2e-16 \*\*\*

poly(day, 3)3 1.1611770 0.2724163 4.263 2.67e-05 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

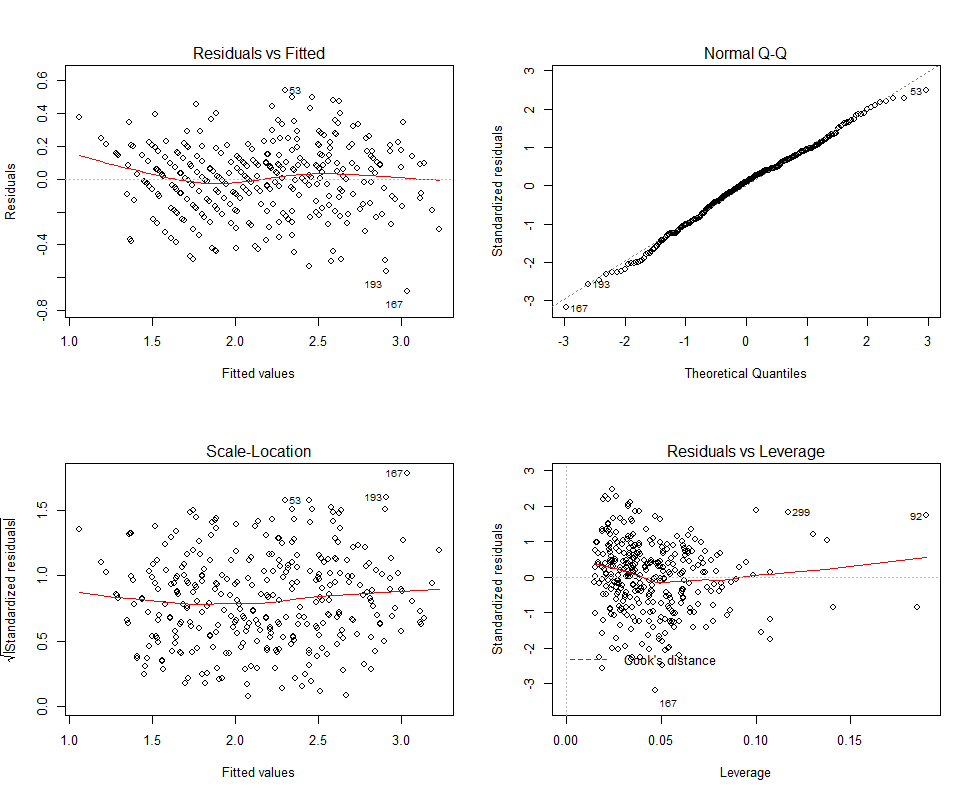
Residual standard error: 0.2204 on 315 degrees of freedom

Multiple R-squared: 0.8276, Adjusted R-squared: 0.82

F-statistic: 108 on 14 and 315 DF, p-value: < 2.2e-16

> par(mfrow=c(2,2))

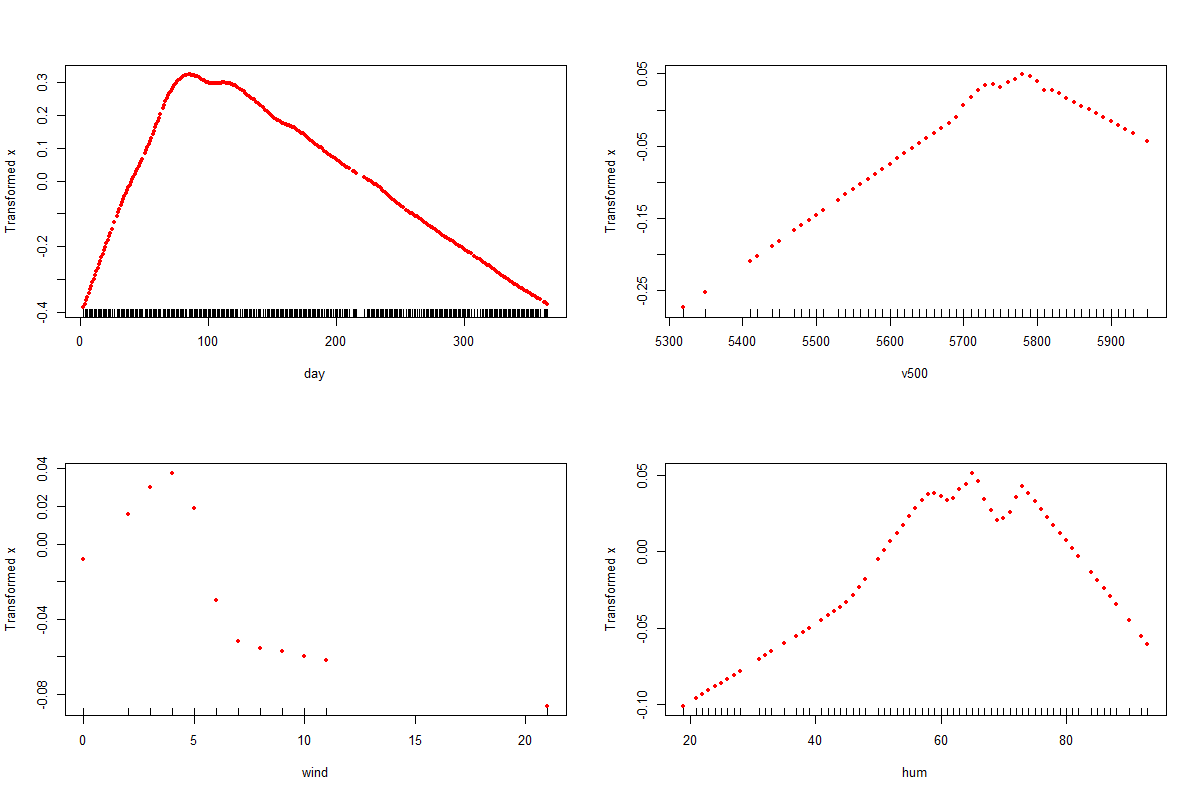
> plot(lm.oz2)

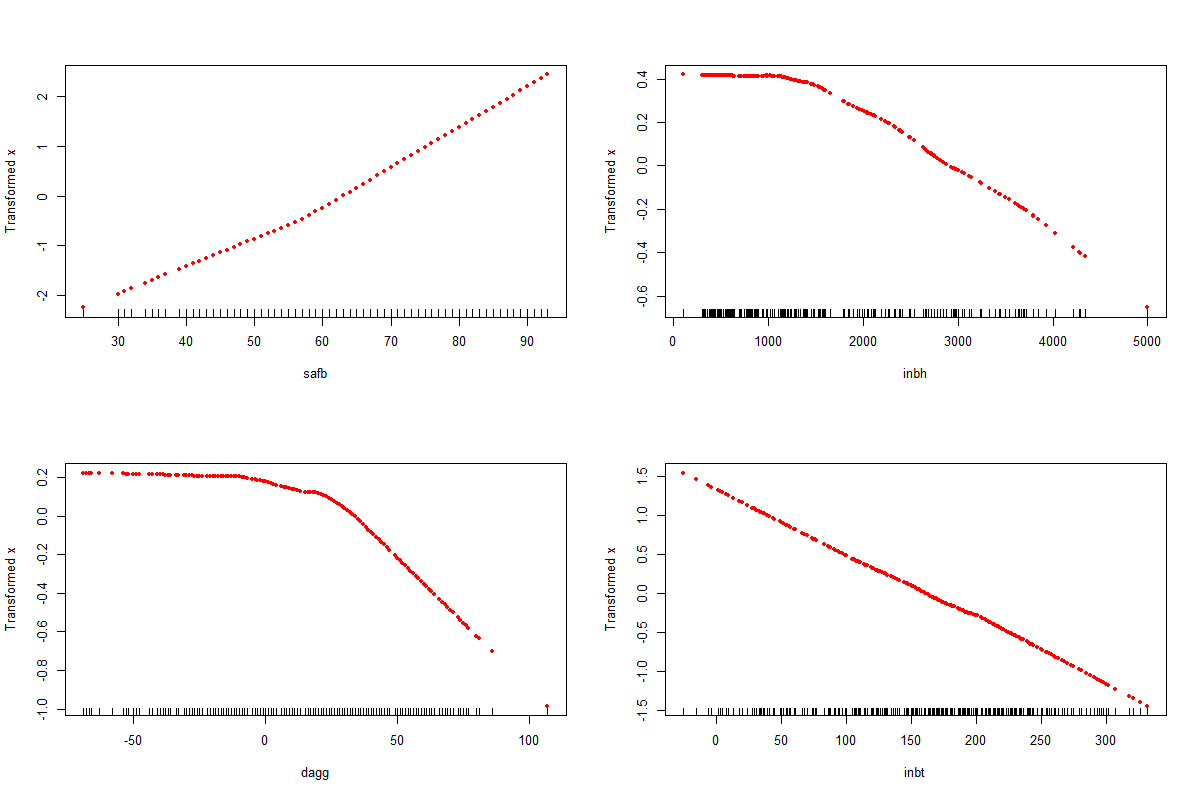


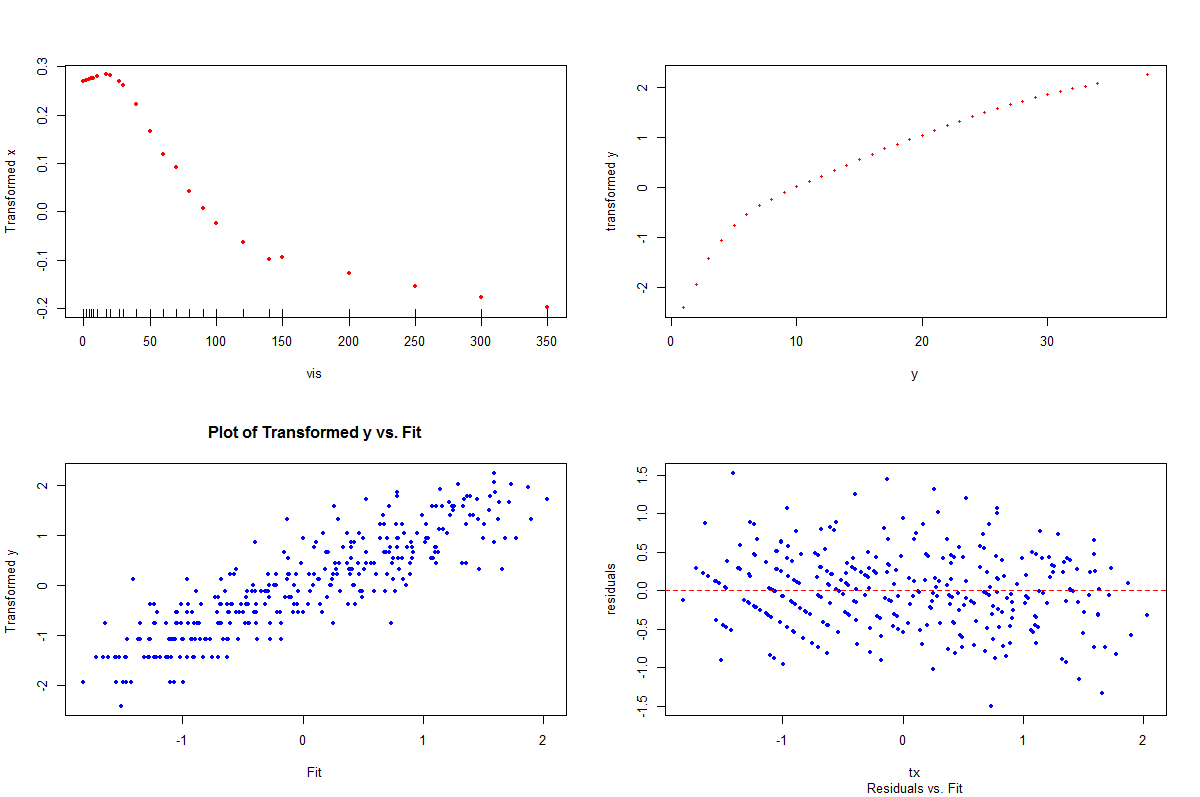
> par(mfrow=c(1,1))

Next we will consider running AVAS to find “optimal” transformations for Y and the predictors.

> avas.oz = avas(X,y)

> maceplot(X,y,avas.oz)  






Using these plots we can again choose “parametric” forms of the transformations found by AVAS to fit an OLS model with a transformed response and terms to capture the nonlinear structure in these data.

**Example 4.3 – Boston Housing Data (p > 1)**

The Boston Housing data set was the basis for a 1978 paper by Harrison and Rubinfeld, which discussed approaches for using housing market data to estimate the willingness to pay for clean air. The authors employed a hedonic price model, based on the premise that the price of the property is determined by structural attributes (such as size, age, condition) as well as neighborhood attributes (such as crime rate, accessibility, environmental factors). This type of approach is often used to quantify the effects of environmental factors that affect the price of a property.

Data were gathered for 506 census tracts in the Boston Standard Metropolitan Statistical Area (SMSA) in 1970, collected from a number of sources including the 1970 US Census and the Boston Metropolitan Area Planning Committee. The variables used to develop the Harrison Rubinfeld housing value equation are listed in the table below. (Boston.working)

**Variables Used in the Harrison-Rubinfeld Housing Value Equation**

|  |  |  |  |
| --- | --- | --- | --- |
| **variable** | **type** | **definition** | **source** |
| CMEDV | Dependent Variable (Y) | Median value of homes in thousands of dollars | 1970 U.S. Census |
| RM | Structural | Average number of rooms | 1970 U.S. Census |
| AGE | % of units built prior to 1940 | 1970 U.S. Census |
| B | Neighborhood | Black % of population | 1970 U.S. Census |
| LSTAT | % of population that is lower socioeconomic status | 1970 U.S. Census |
| CRIM | Crime rate | FBI (1970) |
| ZN | % of residential land zoned for lots > than 25,000 sq. ft. | Metro Area Planning Commission (1972) |
| INDUS | % of non-retail business acres (proxy for industry) | Mass. Dept. of Commerce & Development (1965) |
| TAX | Property tax rate | Mass. Taxpayers Foundation (1970) |
| PTRATIO | Pupil-Teacher ratio | Mass. Dept. of Ed (’71-‘72) |
| CHAS | Dummy variable indicating proximity to Charles River (1 = on river) | 1970 U.S. Census Tract maps |
| DIS | Accessibility | Weighted distances to major employment centers in area | Schnare dissertation (Unpublished, 1973) |
| RAD | Index of accessibility to radial highways | MIT Boston Project |
| NOX | Air Pollution | Nitrogen oxide concentrations (pphm) | TASSIM |

**Reference**

Harrison, D., and Rubinfeld, D. L., “Hedonic Housing Prices and the Demand for Clean Air,” *Journal of Environmental Economics and Managem*ent, **5** (1978), 81-102.

We begin by constructing an OLS regression model for CMEDV using all predictors as given.

> X = model.matrix(CMEDV~.,data=Boston.working)[,-1]

> y = Boston.working$CMEDV

> bos.lm = lm(CMEDV~.,data=Boston.working)

> summary(bos.lm)

Call:

lm(formula = CMEDV ~ ., data = Boston.working)

Residuals:

Min 1Q Median 3Q Max

-15574.2 -2693.6 -520.6 1839.4 26166.2

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.632e+04 5.058e+03 7.181 2.57e-12 \*\*\*

CRIM -1.062e+02 3.257e+01 -3.261 0.001187 \*\*

ZN 4.781e+01 1.361e+01 3.513 0.000484 \*\*\*

INDUS 2.328e+01 6.095e+01 0.382 0.702713

CHAS 2.696e+03 8.540e+02 3.157 0.001693 \*\*

RM 3.794e+03 4.142e+02 9.160 < 2e-16 \*\*\*

AGE 4.964e-01 1.309e+01 0.038 0.969773

DIS -1.502e+03 1.977e+02 -7.598 1.53e-13 \*\*\*

RAD 3.032e+02 6.577e+01 4.610 5.14e-06 \*\*\*

TAX -1.269e+01 3.728e+00 -3.403 0.000720 \*\*\*

PTRATIO -9.236e+02 1.297e+02 -7.122 3.79e-12 \*\*\*

B 9.254e+03 2.662e+03 3.476 0.000553 \*\*\*

LSTAT -5.296e+04 5.025e+03 -10.538 < 2e-16 \*\*\*

NOX -1.774e+03 3.786e+02 -4.687 3.60e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

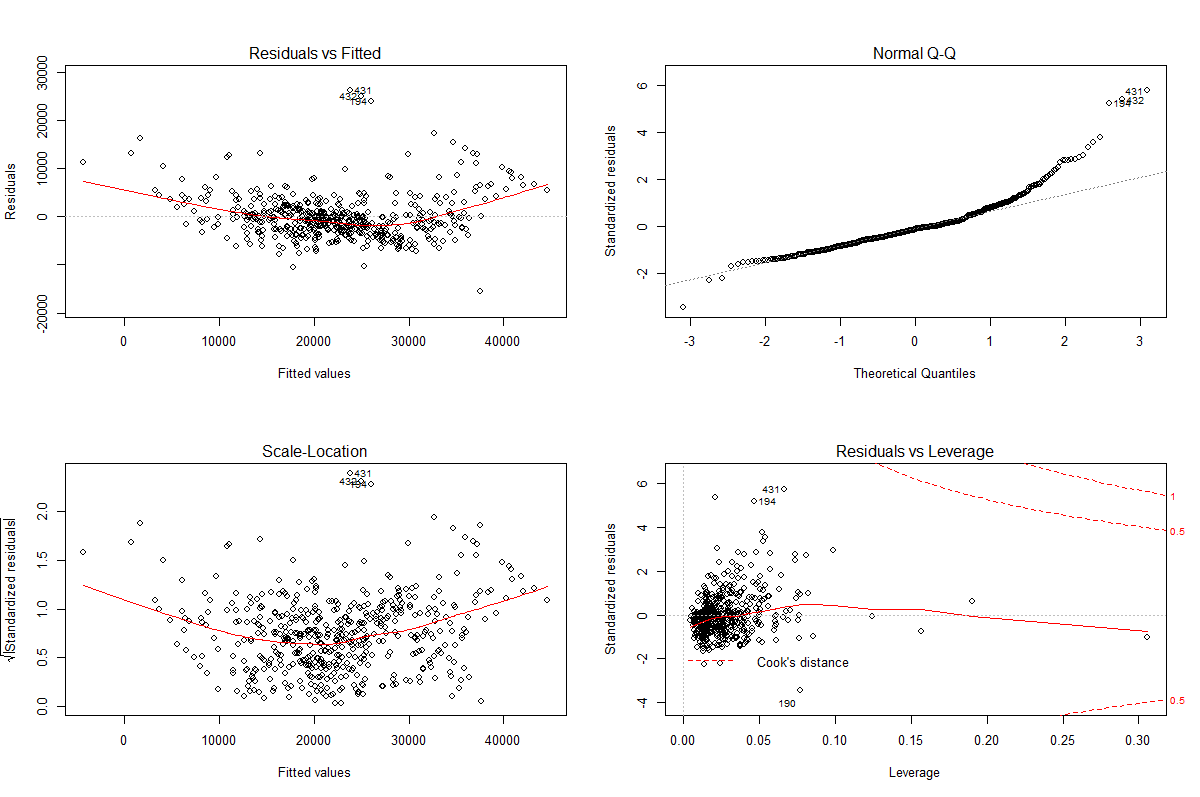
Residual standard error: 4704 on 492 degrees of freedom

Multiple R-squared: 0.7444, Adjusted R-squared: 0.7376

F-statistic: 110.2 on 13 and 492 DF, p-value: < 2.2e-16

> par(mfrow=c(2,2)

> plot(bos.lm)

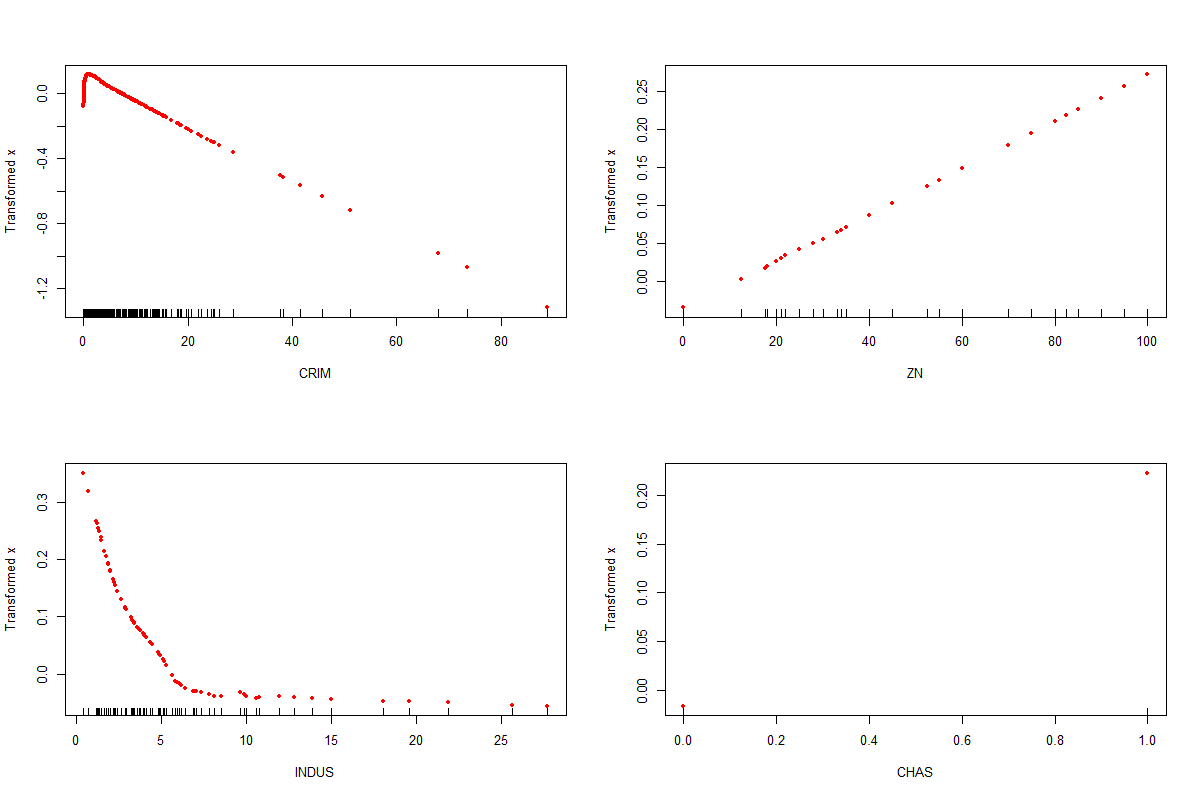


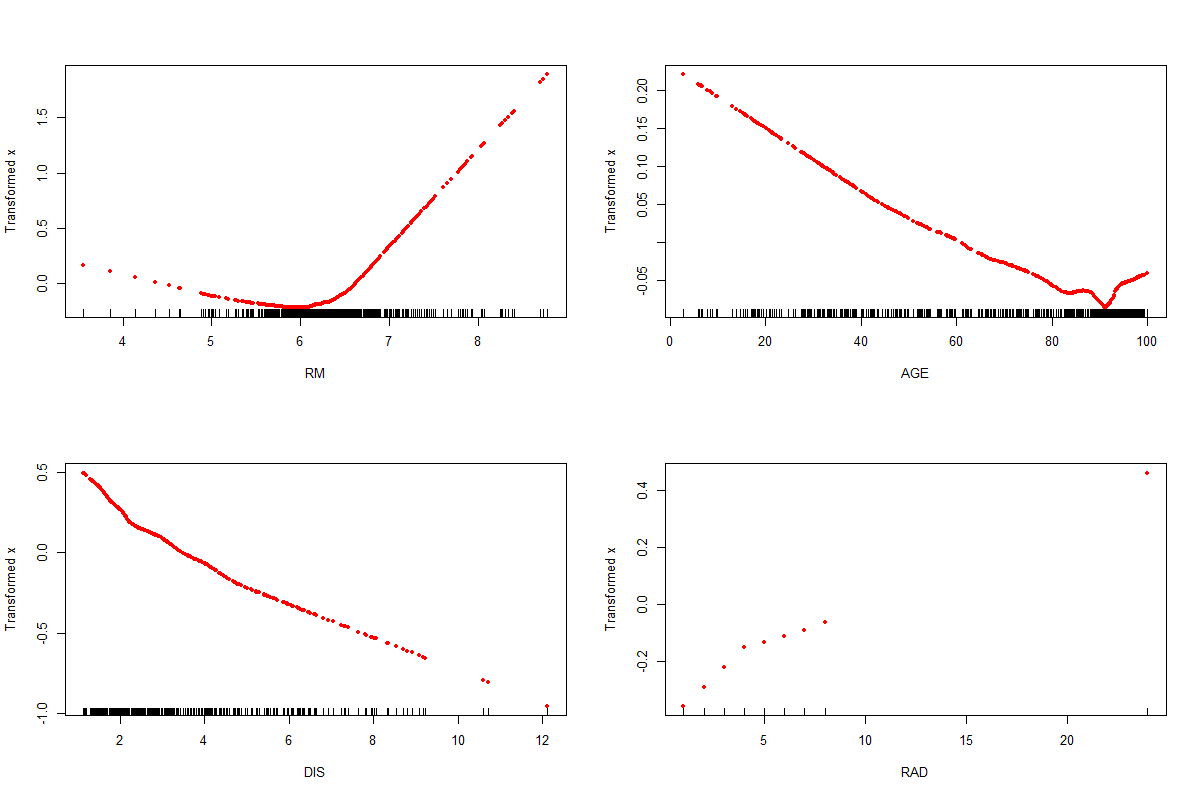
> par(mfrow=c(1,1))

There is definitely evidence of both nonlinearity and heteroscedasticity (non-constant variance) in the plot of the residuals vs. the fitted values from the OLS fit.

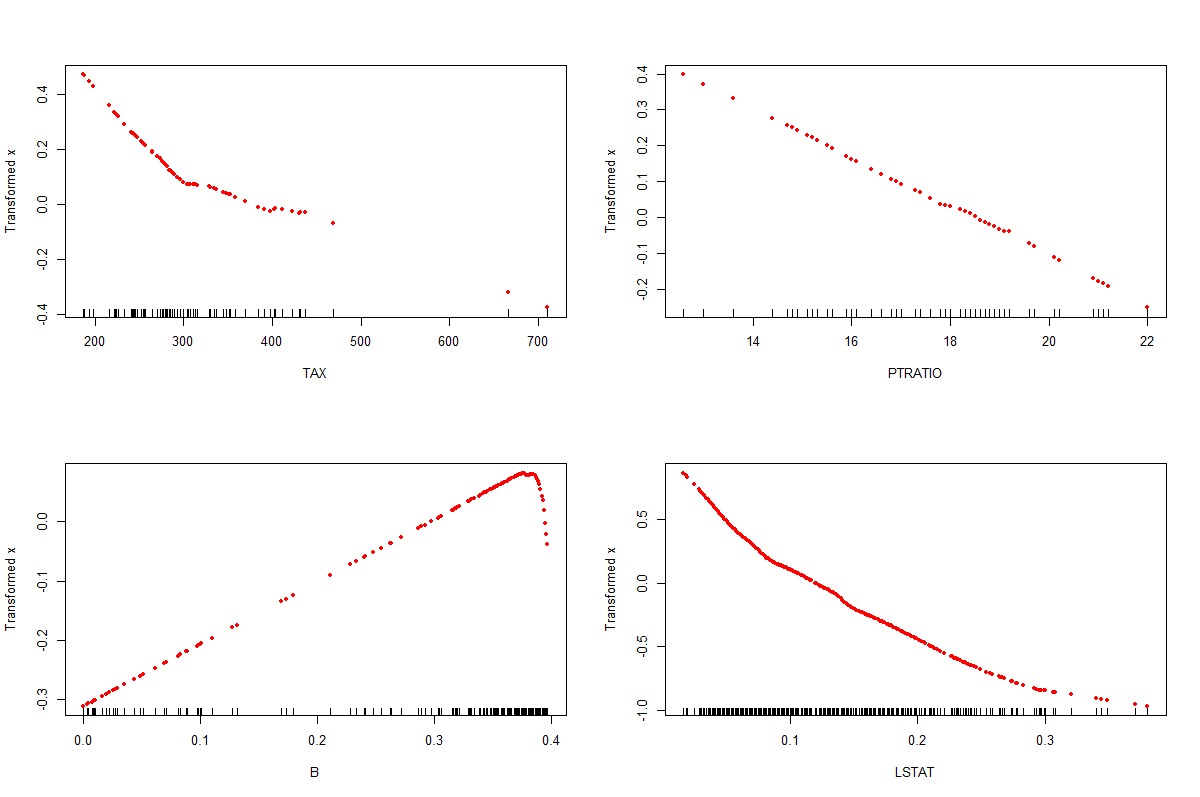
Let’s see what ACE and AVAS give for “optimal” transformations for the response (Y) and the potential predictors in X.

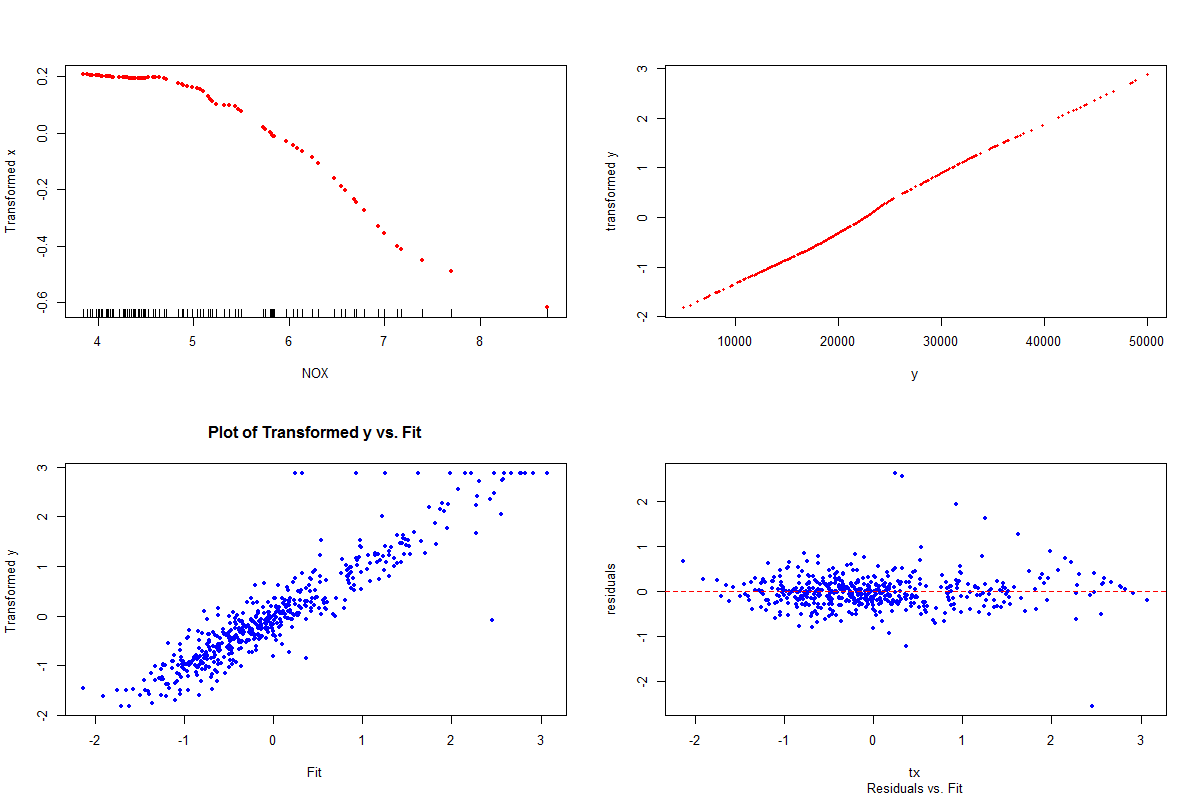
> bos.ace = ace(X,y); maceplot(X,y,bos.ace)





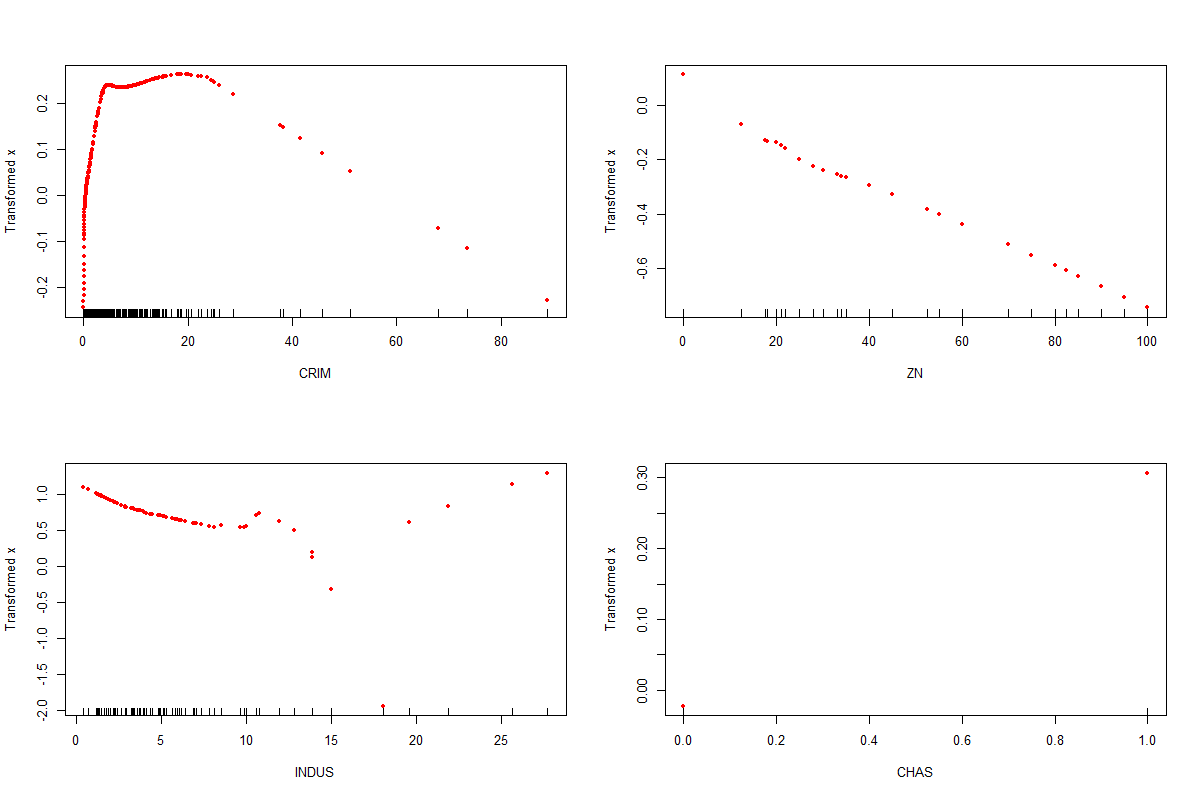
**ACE results (cont’d)**

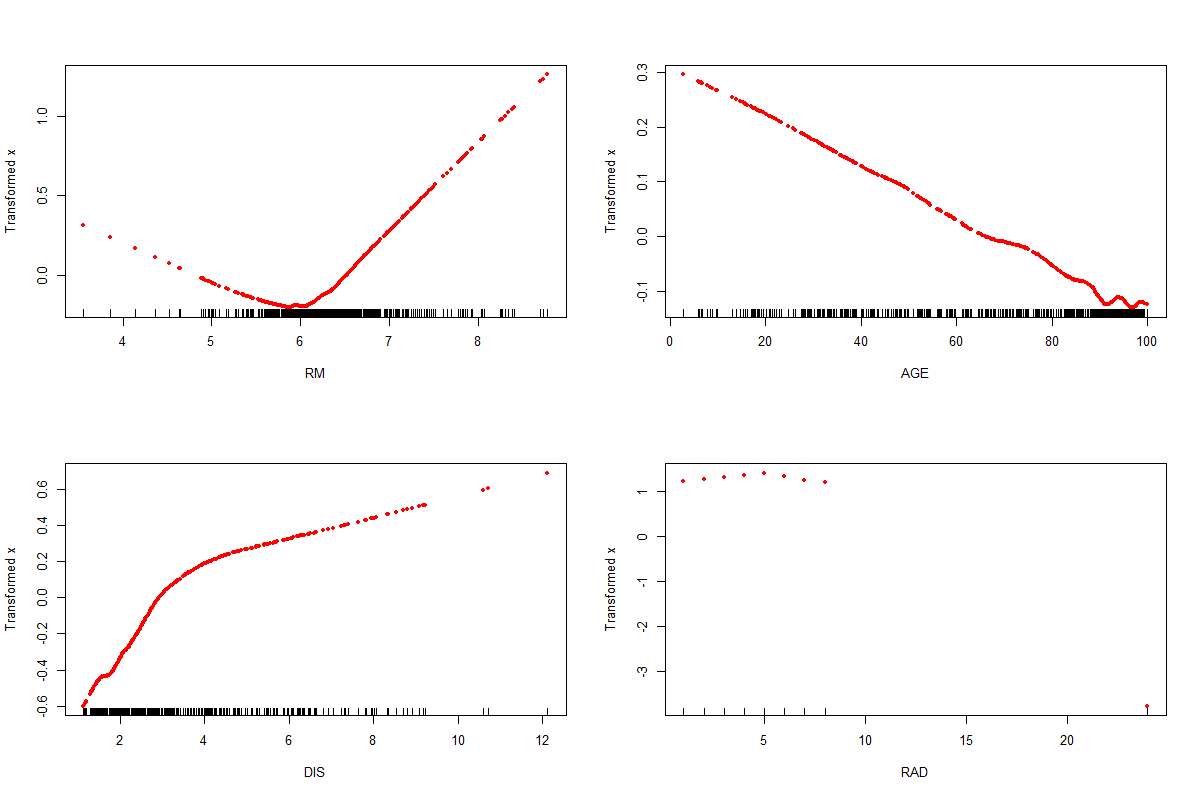




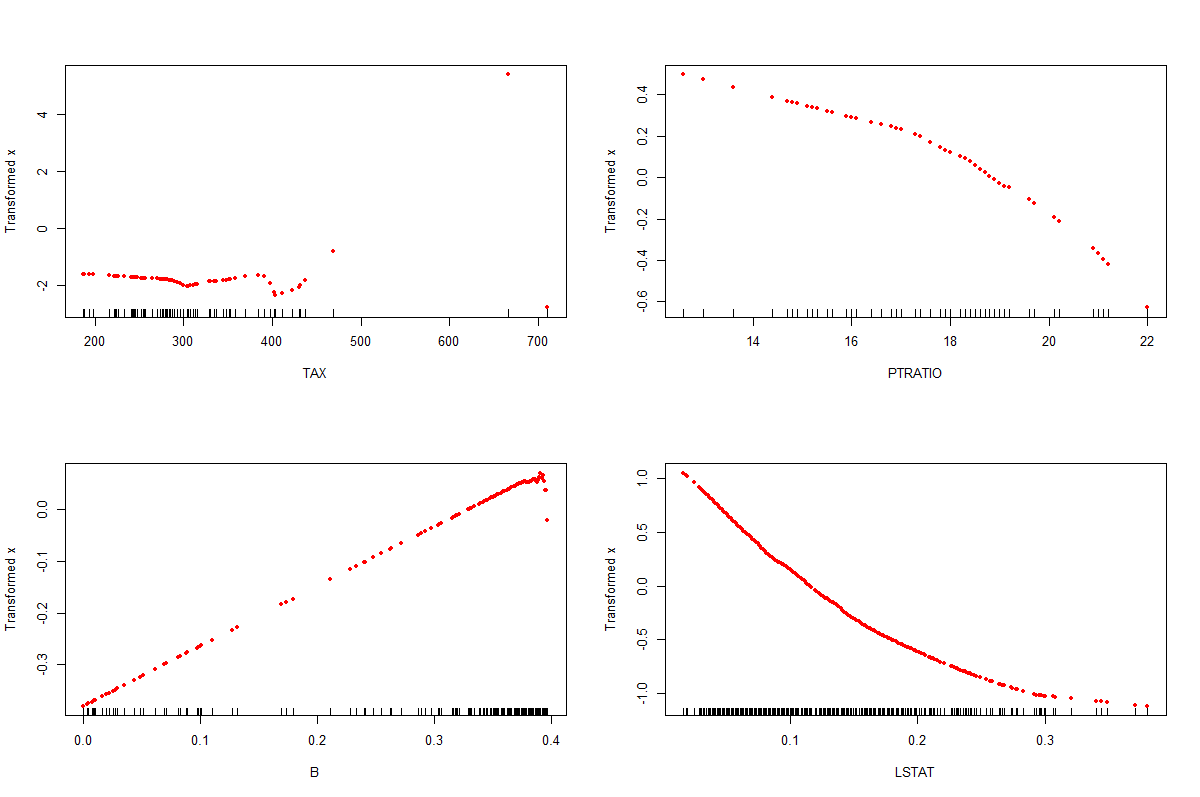
> bos.avas = avas(X,y)

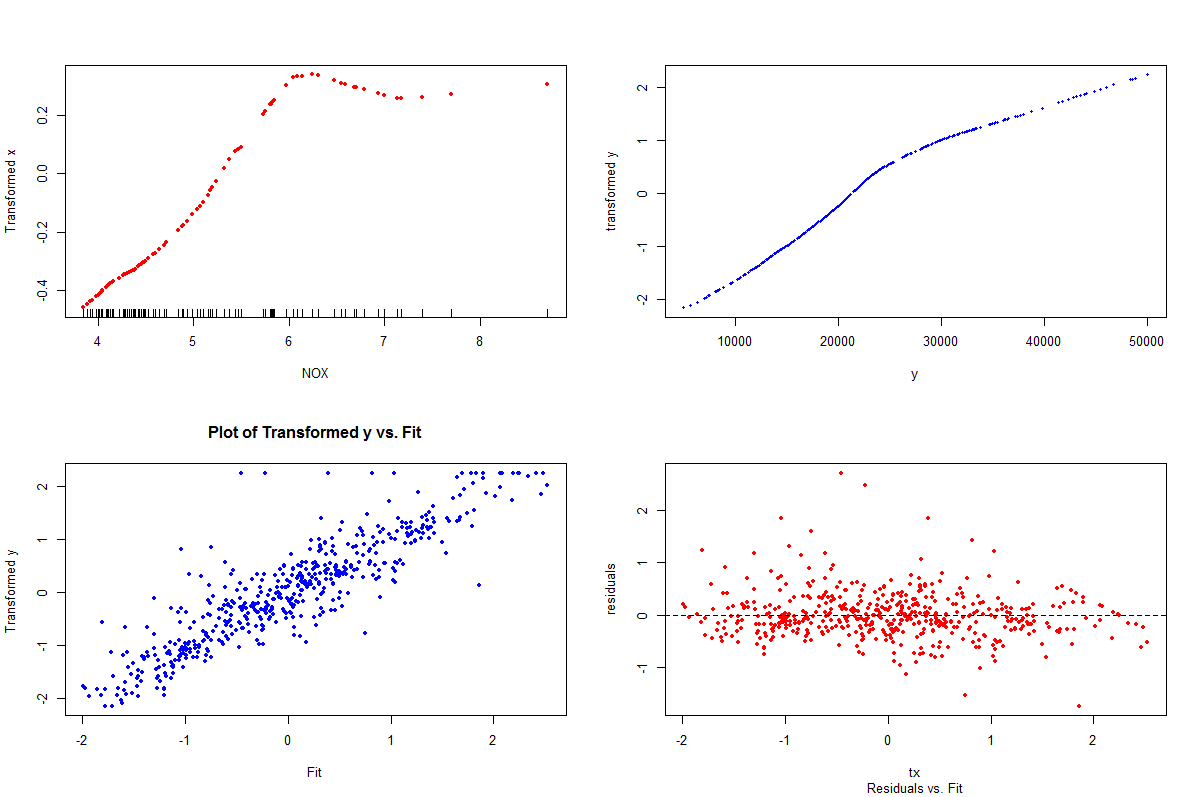
> maceplot(X,y,bos.avas)





**AVAS results (cont’d)**



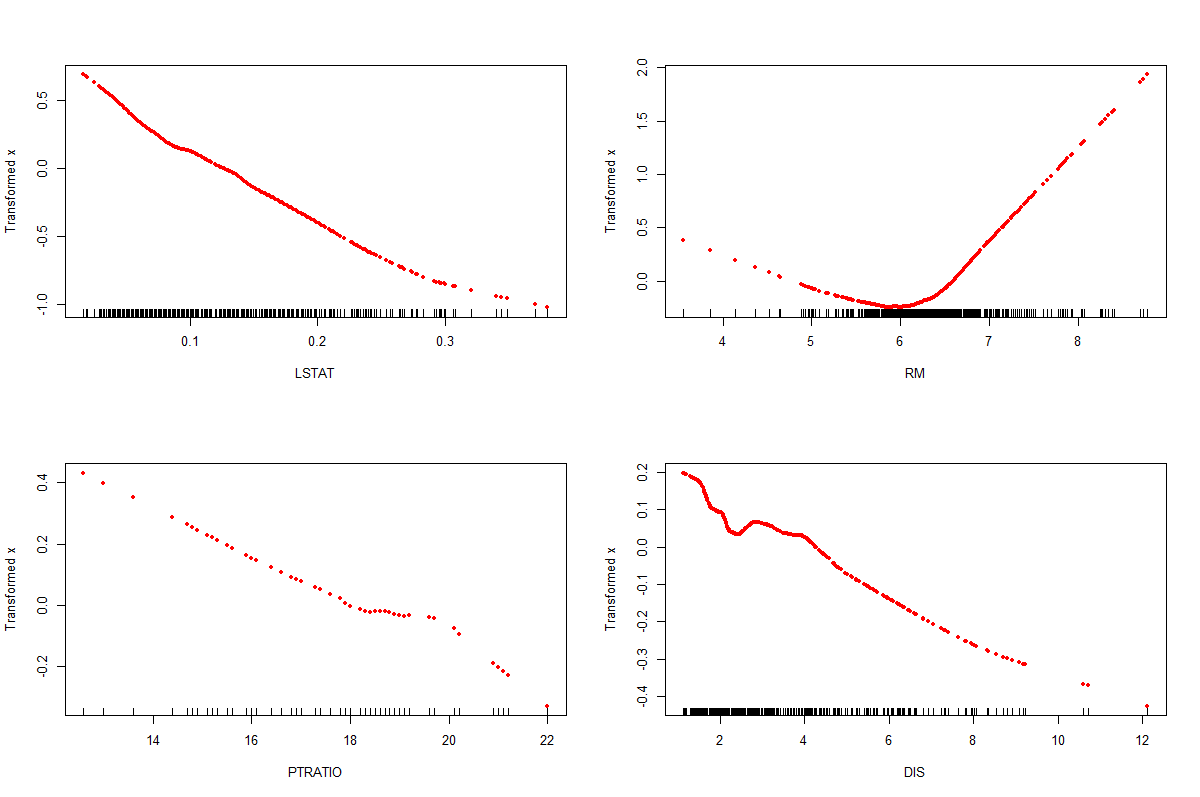


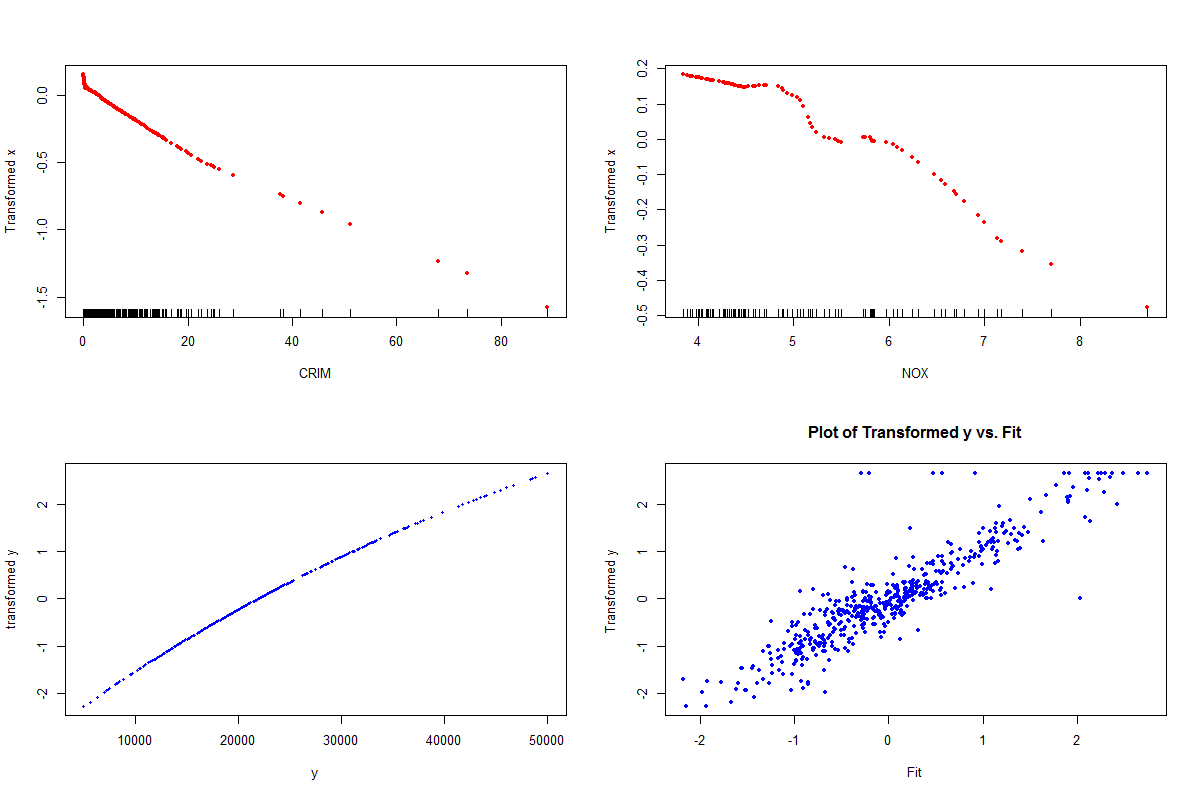
Choosing terms based on these plots could prove challenging. Several of the plots suggest piecewise linear transformations. Also it is important to look at the **scale of the vertical axes** in the vs. *x* plots, as many of the noisy looking transformations represent predictors that are not that important. Choosing a subset of these predictors as being the most useful, we can rerun ACE/AVAS on a smaller subset of these predictors.

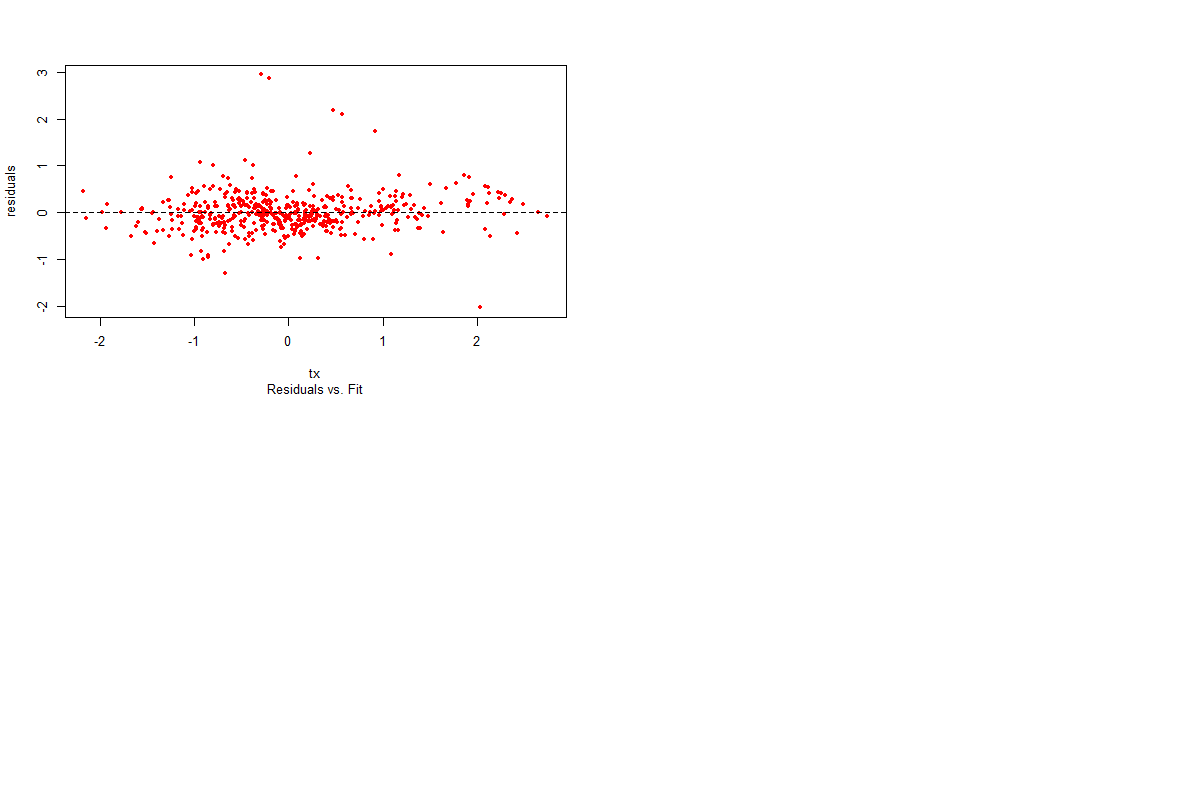
> attach(Boston.working)

> X = cbind(LSTAT,RM,PTRATIO,DIS,CRIM,NOX)

> y = CMEDV







Using a smaller set of variables in the AVAS algorithm gives a clearer impression of the transformations on the important variables. Using these plots to guide the choice of parametric terms the following OLS model seems reasonable.

lm.bos.fin = lm(sqrt(CMEDV)~log(LSTAT)+RM+I(RM^2)+PTRATIO+DIS+CRIM+I(NOX^2))

> summary(lm.bos.fin)

Call:

lm(formula = sqrt(CMEDV) ~ log(LSTAT) + RM + I(RM^2) + PTRATIO +

DIS + CRIM + I(NOX^2))

Residuals:

Min 1Q Median 3Q Max

-58.714 -6.977 -0.915 6.146 79.276

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 306.54371 26.75727 11.456 < 2e-16 \*\*\*

log(LSTAT) -27.58638 1.57208 -17.548 < 2e-16 \*\*\*

RM -56.56838 8.07192 -7.008 7.89e-12 \*\*\*

I(RM^2) 5.03867 0.63180 7.975 1.05e-14 \*\*\*

PTRATIO -2.20143 0.30302 -7.265 1.45e-12 \*\*\*

DIS -3.03396 0.40801 -7.436 4.57e-13 \*\*\*

CRIM -0.61740 0.07621 -8.101 4.23e-15 \*\*\*

I(NOX^2) -0.39288 0.06436 -6.104 2.08e-09 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

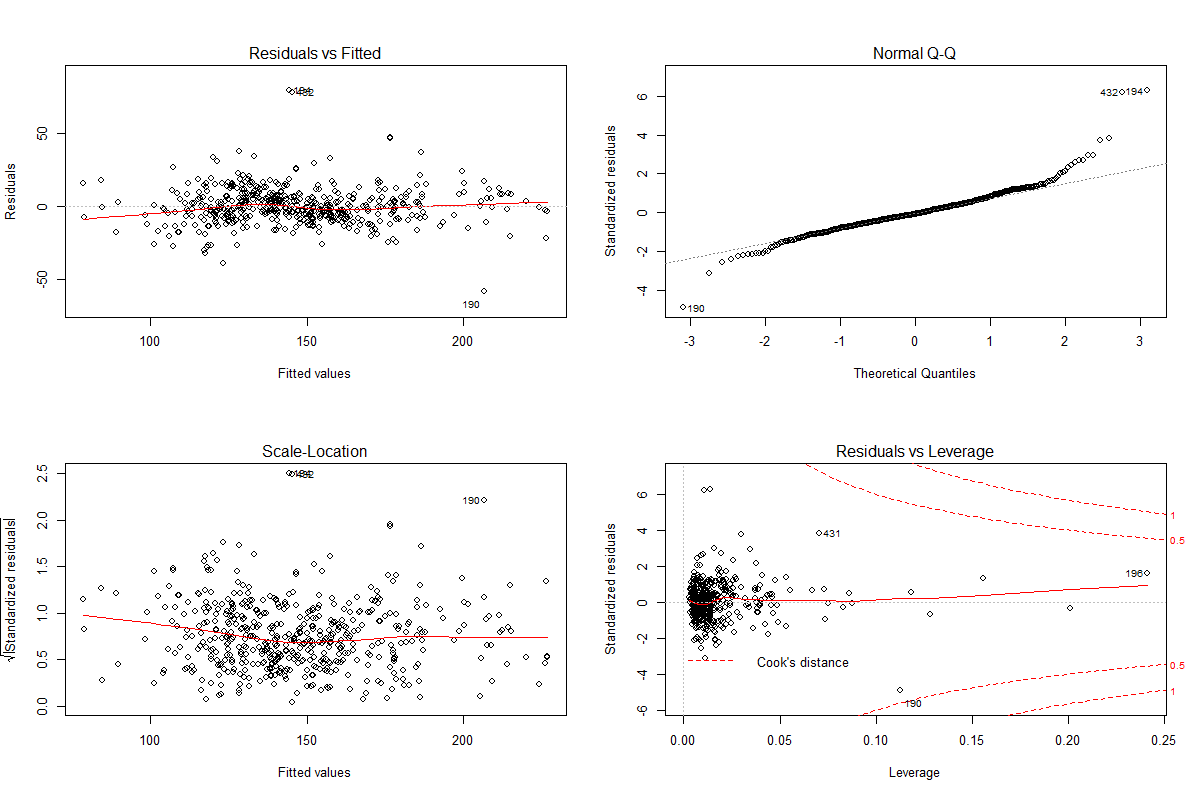
Residual standard error: 12.73 on 498 degrees of freedom

**Multiple R-squared: 0.8171, Adjusted R-squared: 0.8145**

F-statistic: 317.8 on 7 and 498 DF, p-value: < 2.2e-16

> par(mfrow=c(2,2))

> plot(lm.bos.fin)



> par(mfrow=c(1,1))

This model seems to fit quite well and ACE/AVAS aided greatly in the selection of the terms to use in an OLS model. Although the results of ACE or AVAS should not be trusted blindly, these methods certainly belong in a data analyst’s toolbox of methods for use in the model development process.

The next example shows situations where the ACE/AVAS and more importantly the Backfitting Algorithm (which is used by ACE/AVAS and other modern/flexible regression methods) can fail! It turns out in situations where the relationships between the predictors themselves are nonlinear, these nonlinearities can bleed into the scatterplot smoothing processes used to estimate the functional forms of the predictors ).

**Example 4.4 - Backfitting algorithm fails when predictors are nonlinearly related**

> x3 = runif(100,min=1,max=26)

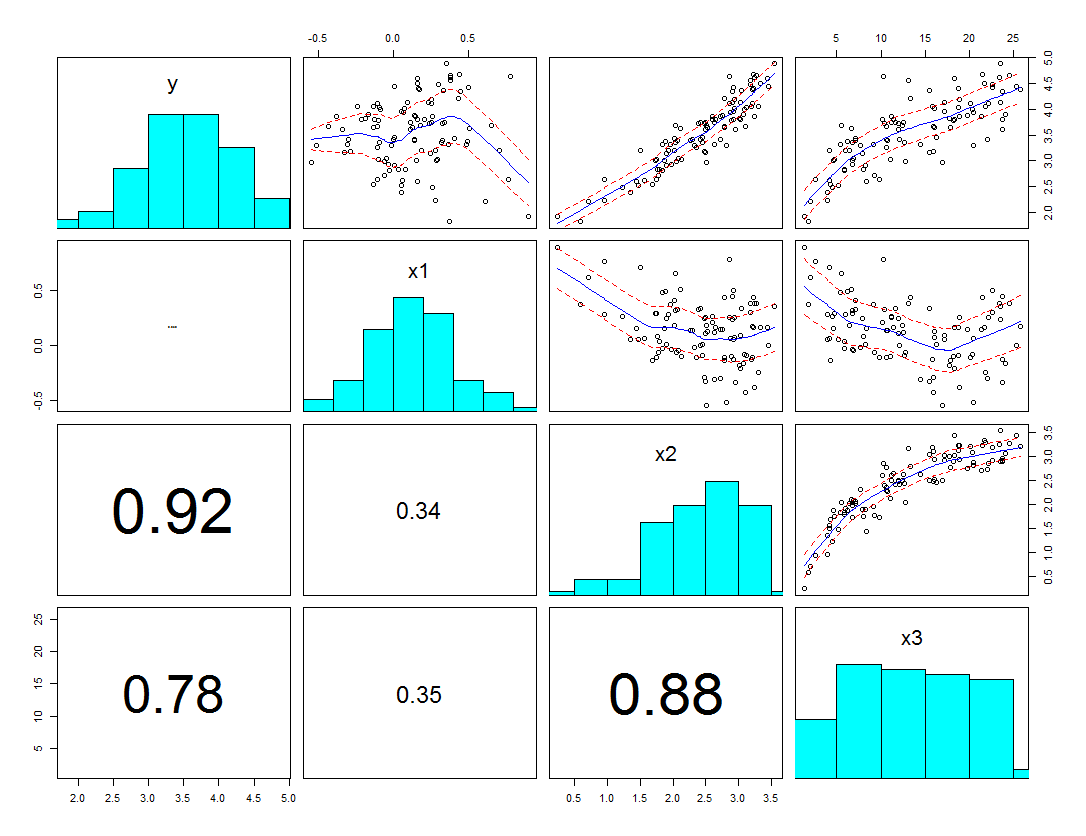
> x1 = 1/x3 + rnorm(100,mean=0,sd=.1)

> x2 = log(x3) + rnorm(100,mean=0,sd=.25)

> y = x1 + x2 + 1/(1+exp(-x3))

> Counterexample = data.frame(y,x1,x2,x3)

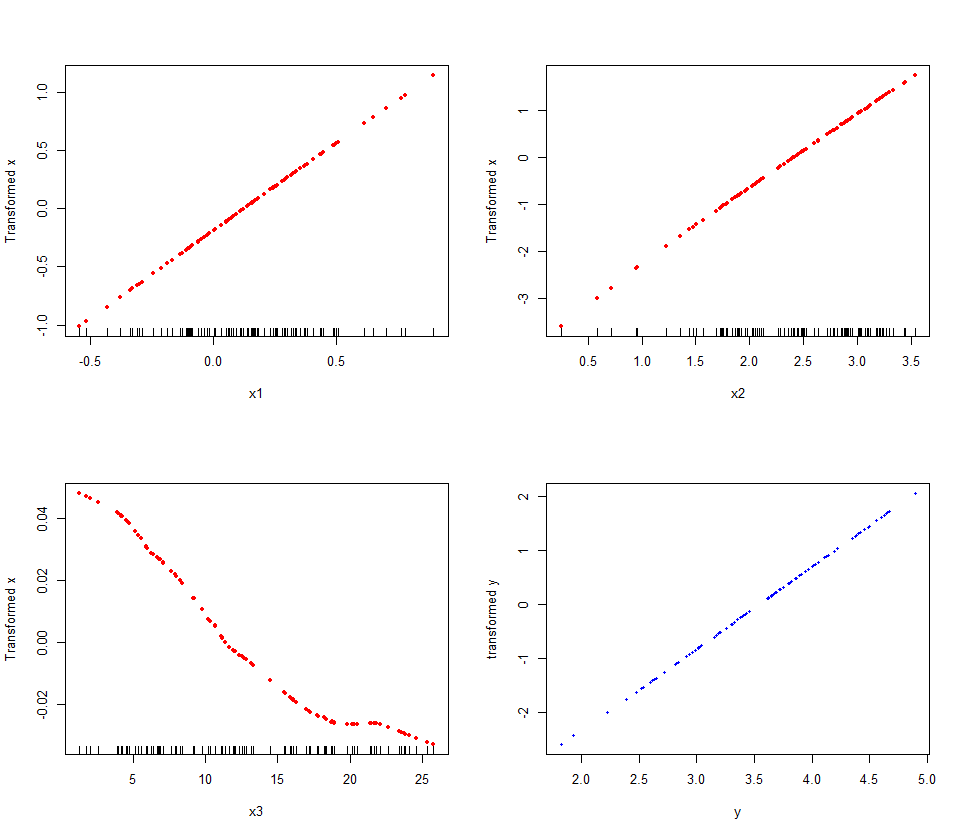
> pairs.plus(Counterexample)

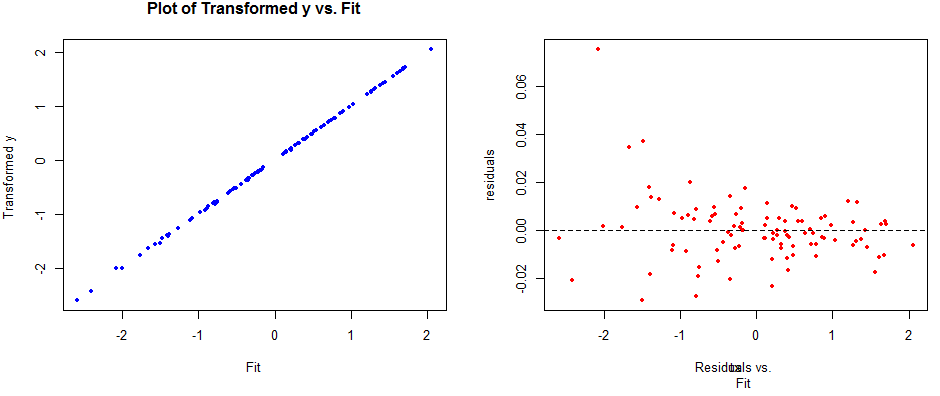


> X = cbind(x1,x2,x3)

> ace.CE = ace(X,y)

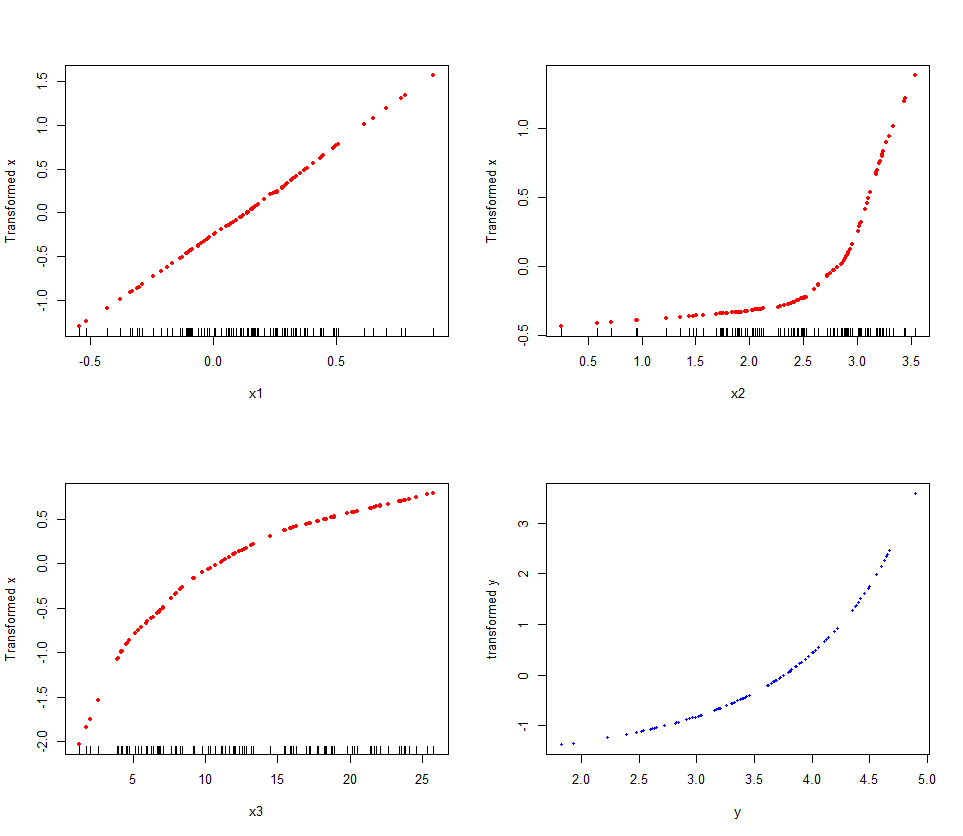
> maceplot(X,y,ace.CE)

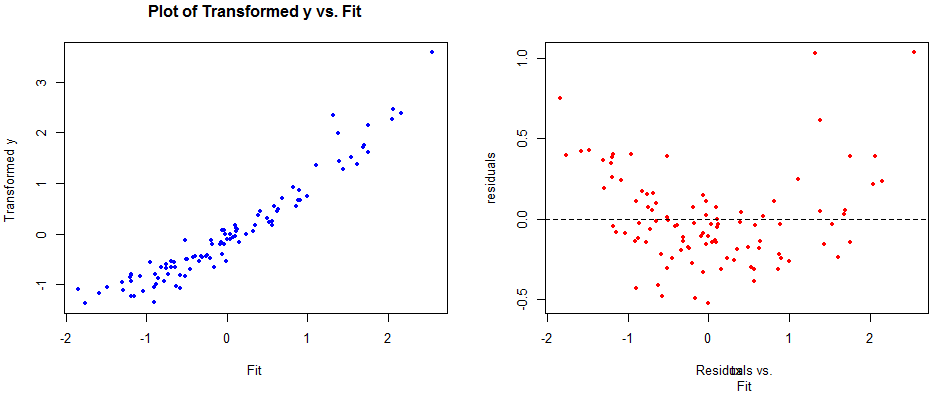




> avas.CE = avas(X,y)

> maceplot(X,y,avas.CE)





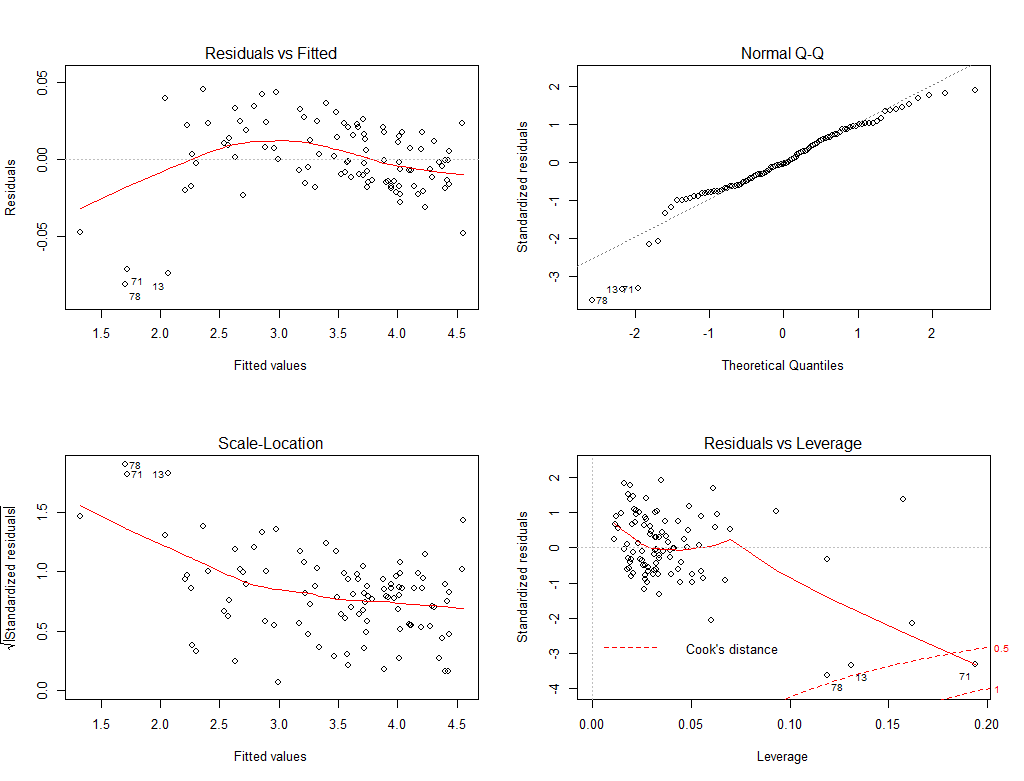
Both ACE and AVAS fail to find the proper functional forms for the and surprisingly AVAS fails horribly at achieving a reasonable fit to these data.

We now consider CERES plots, which in theory should do a better job of identifying the functional form of the predictors in this model.

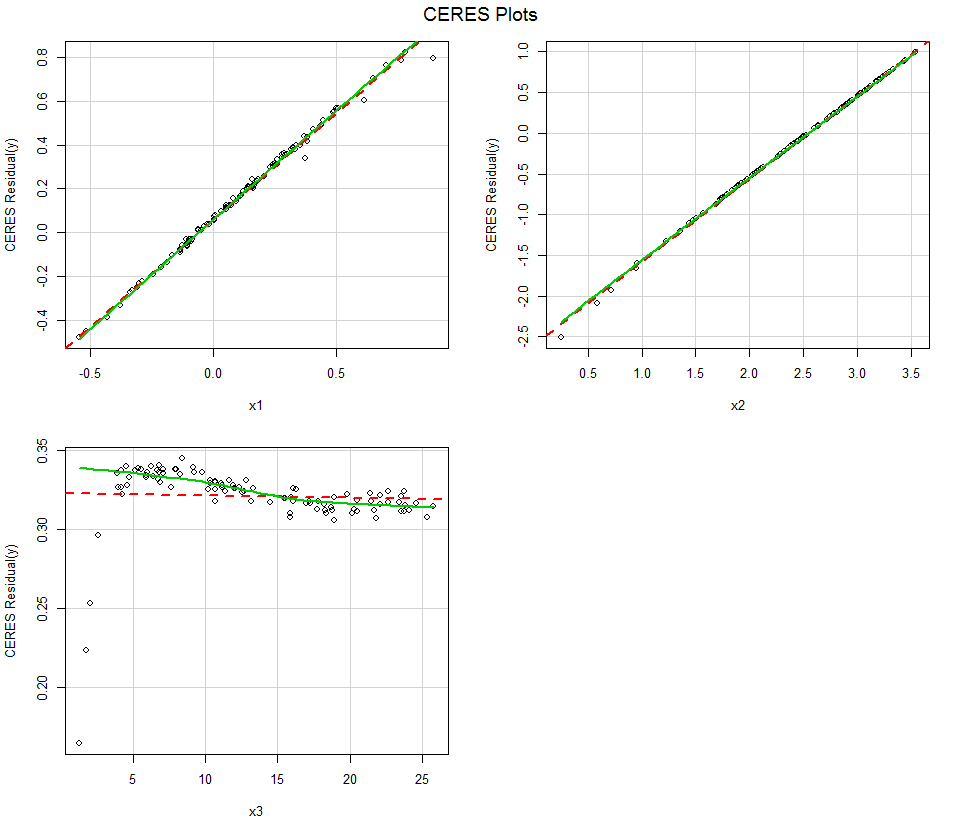
> lm1 = lm(y~x1+x2+x3)

> par(mfrow=c(2,2))

> plot(lm1)



> library(car)  
> ceresPlots(lm1)



Correct Functional form for

The key points to take away from this example are:

1. When predictors are nonlinearly related, strange things can and do happen.
2. The backfitting algorithm, which relies essentially on the component + residual plots (C+R plots) to estimate the functional form of the predictors does not work well when predictors are nonlinearly related.
3. The failure of AVAS is due to the fact that when the functional form of the mean function is wrong, it can appear as though there is heteroscedasticity when in fact the variance is constant. Cook & Weisberg provide another example of this in Chapter 14 of their text when discussing methods for diagnosing nonconstant error variance. See the example on pg. 349 (Applied Regression Including Computing and Graphics).

**4.5 – Appendix of R Functions for ACE and AVAS plotting**

Code for both saceplot and maceplot

saceplot = function (x, y, a, xname = "x", yname = "y") {

par(mfrow = c(2, 2), pty = "m")

plot(x, a$tx, xlab = paste("Untransformed ", xname), ylab =   
 paste("Transformed ",xname), main = "Plot of Transformed x vs. x",  
 cex = 0.6,pch=20)

rug(x)

plot(y, a$ty, xlab = paste("Untransformed ", yname), ylab =  
 paste("Transformed ", yname), main = "Plot of Transformed y vs. y", cex =   
 0.6,pch=20)

rug(y)

plot(a$tx, a$ty, main = "Plot of Transformed y vs. Transformed x",

xlab = "Transformed x", ylab = "Transformed y", cex = 0.6,pch=20)

r <- a$ty - a$tx

rug(a$tx)

plot(a$tx, r, xlab = "Fit (tx)", ylab = "Residuals",   
 sub = "Residuals vs. Fit ", cex = 0.6,pch=20)

abline(h=0,lty=2,col=”red”)

par(mfrow = c(1, 1))

invisible()

}

maceplot = function (xmat, y, x, nrow = 2, ncol = 2) {

par(mfrow = c(nrow, ncol), ask = T)

for (i in 1:ncol(xmat)) {

plot(xmat[, i], x$tx[, i], xlab = dimnames(xmat)[[2]][i],

ylab = "Transformed x", pch = 20,col="red")

rug(xmat[, i])

}

plot(y, x$ty, xlab = "y", ylab = "transformed y", cex = 0.7,

pch = 20,col="blue")

fit <- rep(0, length(y))

for (k in 1:ncol(xmat)) {

fit <- fit + x$tx[, k]

}

plot(fit, x$ty, main = "Plot of Transformed y vs. Fit",

xlab = "Fit", ylab = "Transformed y ", pch = 20,col="blue")

r <- x$ty - fit

plot(fit, r, xlab = "tx", ylab = "residuals", sub = "Residuals vs.   
 Fit",pch = 20,col="red")

abline(h = 0, col = "black", lty = 2)

par(mfrow = c(1, 1), ask = F)

invisible()

}