Chapter 16
Transform-Both-Sides Regression

16.1 Background

Fitting multiple regression models by the method of least squares is one of the most commonly used methods in statistics. There are a number of challenges to the use of least squares, even when it is only used for estimation and not inference, including the following.

1. How should continuous predictors be transformed so as to get a good fit?
2. Is it better to transform the response variable? How does one find a good transformation that simplifies the right-hand side of the equation?
3. What if Y needs to be transformed non-monotonically (e.g., $|Y - 100|$) before it will have any correlation with X?

When one is trying to draw an inference about population effects using confidence limits or hypothesis tests, the most common approach is to assume that the residuals have a normal distribution. This is equivalent to assuming that the conditional distribution of the response Y given the set of predictors X is normal with mean depending on X and variance that is (one hopes) a constant independent of X. The need for a distributional assumption to enable us to draw inferences creates a number of other challenges such as the following.

1. If for the untransformed original scale of the response Y the distribution of the residuals is not normal with constant spread, ordinary methods will not yield correct inferences (e.g., confidence intervals will not have the desired coverage probability and the intervals will need to be asymmetric).
2. Quite often there is a transformation of Y that will yield well-behaving residuals. How do you find this transformation? Can you find a transformation for the Xs at the same time?

F.E. Harrell, Jr., *Regression Modeling Strategies*, Springer Series in Statistics, DOI 10.1007/978-3-319-19425-7_16

3. All classical statistical inferential methods assume that the full model was pre-specified, that is, the model was not modified after examining the data. How does one correct confidence limits, for example, for data-based model and transformation selection?

16.2 Generalized Additive Models

Hastie and Tibshirani[275] have developed *generalized additive models* (GAMs) for a variety of distributions for Y. There are semiparametric GAMs, but most GAMs for continuous Y assume that the conditional distribution of Y is from a specific distribution family. GAMs nicely estimate the transformation each continuous X requires so as to optimize a fitting criterion such as sum of squared errors or log likelihood, subject to the degrees of freedom the analyst desires to spend on each predictor. However, GAMs assume that Y has already been transformed to fit the specified distribution family.

There is excellent software available for fitting a wide variety of GAMs, such as the `R` packages `gam`, `mgcv`, and `robustgam`.

16.3 Nonparametric Estimation of Y-Transformation

When the model's left-hand side also needs transformation, either to improve R^2 or to achieve constant variance of the residuals (which increases the chances of satisfying a normality assumption), there are a few approaches available. One approach is Breiman and Friedman's *alternating conditional expectation* (ACE) method.[68] ACE simultaneously transforms both Y and each of the Xs so as to maximize the multiple R^2 between the transformed Y and the transformed Xs. The model is given by

$$g(Y) = f_1(X_1) + f_2(X_2) + \ldots + f_p(X_p). \tag{16.1}$$

ACE allows the analyst to impose restrictions on the transformations such as monotonicity. It allows for categorical predictors, whose categories will automatically be given numeric scores. The transformation for Y is allowed to be non-monotonic. One feature of ACE is its ability to estimate the *maximal correlation* between an X and the response Y. Unlike the ordinary correlation coefficient (which assumes linearity) or Spearman's rank correlation (which assumes monotonicity), the maximal correlation has the property that it is zero if and only if X and Y are statistically independent. This property holds because ACE allows for non-monotonic transformations of all variables. The "super smoother" (see the S `supsmu` function) is the basis for the nonparametric estimation of transformations for continuous Xs.

Tibshirani developed a different algorithm for nonparametric additive regression based on least squares, *additivity and variance stabilization* (AVAS).[607] Unlike ACE, AVAS forces $g(Y)$ to be monotonic. AVAS's fitting criterion is to maximize R^2 while forcing the transformation for Y to result in nearly constant variance of residuals. The model specification is the same as for ACE (Equation 16.3).

ACE and AVAS are powerful fitting algorithms, but they can result in overfitting (R^2 can be greatly inflated when one fits many predictors), and they provide no statistical inferential measures. As discussed earlier, the process of estimating transformations (especially those for Y) can result in significant variance under-estimation, especially for small sample sizes. The bootstrap can be used to correct the apparent R^2 (R^2_{app}) for overfitting. As before, it estimates the optimism (bias) in R^2_{app} and subtracts this optimism from R^2_{app} to get a more trustworthy estimate. The bootstrap can also be used to compute confidence limits for all estimated transformations, and confidence limits for estimated predictor effects that take fully into account the uncertainty associated with the transformations. To do this, all steps involved in fitting the additive models must be repeated fresh for each re-sample.

Limited testing has shown that the sample size needs to exceed 100 for ACE and AVAS to provide stable estimates. In small sample sizes the bootstrap bias-corrected estimate of R^2 will be zero because the sample information did not support simultaneous estimation of all transformations.

16.4 Obtaining Estimates on the Original Scale

A common practice in least squares fitting is to attempt to rectify lack of fit by taking parametric transformations of Y before fitting; the logarithm is the most common transformation.[a] If after transformation the model's residuals have a population median of zero, the inverse transformation of a predicted transformed value estimates the population median of Y given X. This is because unlike means, quantiles are transformation-preserving. Many analysts make the mistake of not reporting which population parameter is being estimated when inverse transforming $X\hat{\beta}$, and sometimes they even report that the mean is being estimated.

How would one go about estimating the population mean or other parameter on the untransformed scale? If the residuals are assumed to be normally distributed and if $\log(Y)$ is the transformation, the mean of the log-normal distribution, a function of both the mean and the variance of the residuals, can be used to derive the desired quantity. However, if the residuals are not normally distributed, this procedure will not result in the correct estimator.

[a] A disadvantage of transform-both-sides regression is this difficulty of interpreting estimates on the original scale. Sometimes the use of a special generalized linear model can allow for a good fit without transforming Y.

Duan[165] developed a "smearing" estimator for more nonparametrically obtaining estimates of parameters on the original scale. In the simple one-sample case without predictors in which one has computed $\hat{\theta} = \sum_{i=1}^{n} \log(Y_i)/n$, the residuals from this fitted value are given by $e_i = \log(Y_i) - \hat{\theta}$. The smearing estimator of the population mean is $\sum \exp[\hat{\theta} + e_i]/n$. In this simple case the result is the ordinary sample mean $\overline{Y}$.

The worth of Duan's smearing estimator is in regression modeling. Suppose that the regression was run on $g(Y)$ from which estimated values $\hat{g}(Y_i) = X_i\hat{\beta}$ and residuals on the transformed scale $e_i = \hat{g}(Y_i) - X_i\hat{\beta}$ were obtained. Instead of restricting ourselves to estimating the population mean, let $W(y_1, y_2, \ldots, y_n)$ denote any function of a vector of untransformed response values. To estimate the population mean in the homogeneous one-sample case, W is the simple average of all of its arguments. To estimate the population 0.25 quantile, W is the sample 0.25 quantile of $y_1, \ldots, y_n$. Then the smearing estimator of the population parameter estimated by W given X is $W(g^{-1}(a + e_1), g^{-1}(a + e_2), \ldots, g^{-1}(a + e_n))$, where g^{-1} is the inverse of the g transformation and $a = X\hat{\beta}$.

When using the AVAS algorithm, the monotonic transformation g is estimated from the data, and the predicted value of $\hat{g}(Y)$ is given by Equation 16.3. So we extend the smearing estimator as $W(\hat{g}^{-1}(a+e_1), \ldots, \hat{g}^{-1}(a+e_n))$, where a is the predicted transformed response given X. As $\hat{g}$ is nonparametric (i.e., a table look-up), the `areg.boot` function described below computes $\hat{g}^{-1}$ using reverse linear interpolation.

If residuals from $\hat{g}(Y)$ are assumed to be symmetrically distributed, their population median is zero and we can estimate the median on the untransformed scale by computing $\hat{g}^{-1}(X\hat{\beta})$. To be safe, `areg.boot` adds the median residual to $X\hat{\beta}$ when estimating the population median (the median residual can be ignored by specifying `statistic='fitted'` to functions that operate on objects created by `areg.boot`).

When quantiles of Y are of major interest, a more direct way to obtain estimates is through the use of quantile regression[357]. An excellent case study including comparisons with other methods such as Cox regression can be found in Austin et al.[38].

16.5 R Functions

The R `acepack` package's `ace` function implements all the features of the ACE algorithm, and its `avas` function does likewise for AVAS. The bootstrap and smearing capabilities mentioned above are offered for these estimation functions by the `areg.boot` ("additive regression using the bootstrap") function in the `Hmisc` package. Unlike the `ace` and `avas` functions, `areg.boot` uses the R modeling language, making it easier for the analyst to specify the predictor variables and what is assumed about their relationships with the transformed Y. `areg.boot` also implements a parametric transform-both-sides approach using restricted cubic splines and canonical variates, and offers various estimation options with and without smearing. It can estimate the effect of changing one predictor, holding others constant, using the ordinary bootstrap to estimate the standard deviation of difference in two possibly transformed estimates (for two values of X), assuming normality of such differences. Normality is assumed to avoid generating a large number of bootstrap replications of time-consuming model fits. It would not be very difficult to add nonparametric bootstrap confidence limit capabilities to the software. `areg.boot` re-samples every aspect of the modeling process it uses, just as Faraway[186] did for parametric least squares modeling.

`areg.boot` implements a variety of methods as shown in the simple example below. The `monotone` function restricts a variable's transformation to be monotonic, while the `I` function restricts it to be linear.

```
f ← areg.boot(Y ~ monotone(age) +
                  sex + weight + I(blood.pressure))

plot(f)         #show transformations, CLs
Function(f)     #generate S functions
                #defining transformations
predict(f)      #get predictions, smearing estimates
summary(f)      #compute CLs on effects of each X
smearingEst()   #generalized smearing estimators
Mean(f)         #derive S function to
                #compute smearing mean Y
Quantile(f)     #derive function to compute smearing quantile
```

The methods are best described in a case study.

16.6 Case Study

Consider simulated data where the conditional distribution of Y is log-normal given X, but where transform-both-sides regression methods use unlogged Y. Predictor X_1 is linearly related to log Y, X_2 is related by $|X_2 - \frac{1}{2}|$, and categorical X_3 has reference group a effect of zero, group b effect of 0.3, and group c effect of 0.5.

```
require(rms)
```

```
set.seed(7)
n ← 400
x1 ← runif(n)
x2 ← runif(n)
x3 ← factor(sample(c('a','b','c'), n, TRUE))
y  ← exp(x1 + 2*abs(x2 - .5) + .3*(x3=='b') + .5*(x3=='c') +
         .5*rnorm(n))
```

```
# For reference fit appropriate OLS model
print(ols(log(y) ∼ x1 + rcs(x2, 5) + x3), coefs=FALSE,
      latex=TRUE)
```

Linear Regression Model

```
ols(formula = log(y) ~ x1 + rcs(x2, 5) + x3)
```

	Model Likelihood Ratio Test		Discrimination Indexes	
Obs 400	LR χ^2	236.87	R^2	0.447
σ 0.4722	d.f.	7	R^2_{adj}	0.437
d.f. 392	Pr$(>\chi^2)$	0.0000	g	0.482

Residuals

Min	1Q	Median	3Q	Max
−1.346	−0.3075	−0.0134	0.327	1.527

Now fit the avas model. We use 300 bootstrap repetitions but only plot the first 20 estimates to see clearly how the bootstrap re-estimates of transformations vary. Had we wanted to restrict transformations to be linear, we would have specified the identity function, for example, `I(x1)`.

```
f ← areg.boot(y ∼ x1 + x2 + x3, method='avas', B=300)
```

```
f
```

```
avas Additive Regression Model

areg.boot(x = y ∼ x1 + x2 + x3, B = 300, method = "avas")


Predictor Types

   type
x1    s
x2    s
x3    c

y type: s

n= 400   p= 3

Apparent R2 on transformed Y scale: 0.444
Bootstrap validated R2            : 0.42

Coefficients of standardized transformations:

    Intercept            x1            x2            x3
-3.443111e-16  9.702960e-01  1.224320e+00  9.881150e-01


Residuals on transformed scale:
```

```
          Min            1Q        Median            3Q           Max
-1.877152e+00 -5.252194e-01 -3.732200e-02  5.339122e-01  2.172680e+00
         Mean          S.D.
 8.673617e-19  7.420788e-01
```

Note that the coefficients above do not mean very much as the scale of the transformations is arbitrary. We see that the model was very slightly overfitted (R^2 dropped from 0.44 to 0.42), and the R^2 are in agreement with the OLS model fit above.

Next we plot the transformations, 0.95 confidence bands, and a sample of the bootstrap estimates.

```
plot(f, boot=20)  # Figure 16.1
```

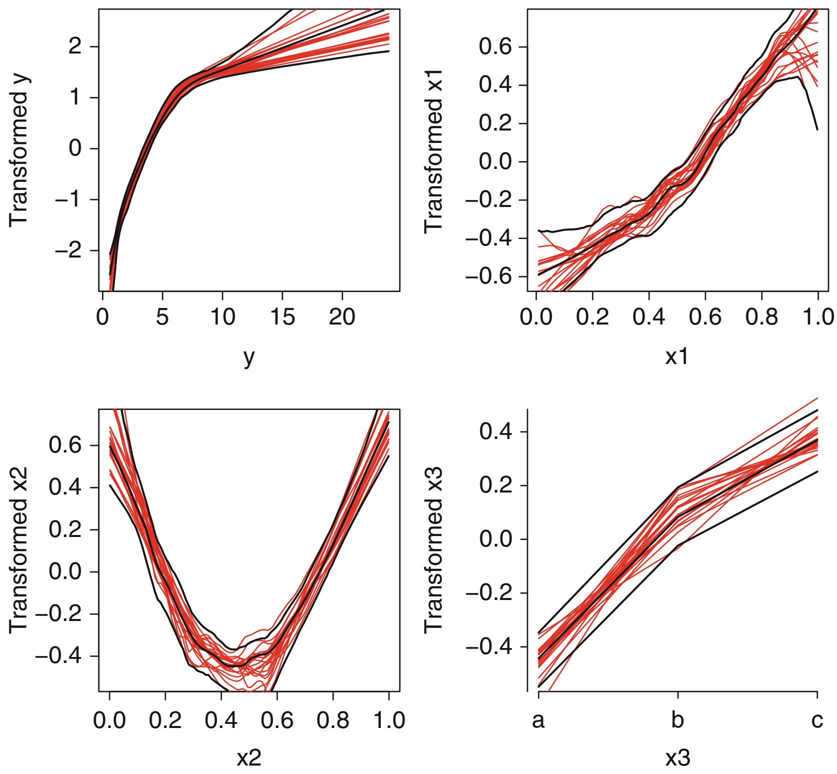

Fig. 16.1 `avas` transformations: overall estimates, pointwise 0.95 confidence bands, and 20 bootstrap estimates (red lines).

The plot is shown in Figure 16.1. The nonparametrically estimated transformation of `x1` is almost linear, and the transformation of `x2` is close to $|x2-0.5|$. We know that the true transformation of `y` is $\log(y)$, so variance stabilization and normality of residuals will be achieved if the estimated `y`-transformation is close to $\log(y)$.

```
ys ← seq(.8, 20, length=200)
ytrans ← Function(f)$y   # Function outputs all transforms
plot(log(ys), ytrans(ys), type='l')   # Figure 16.2
abline(lm(ytrans(ys) ∼ log(ys)), col=gray(.8))
```

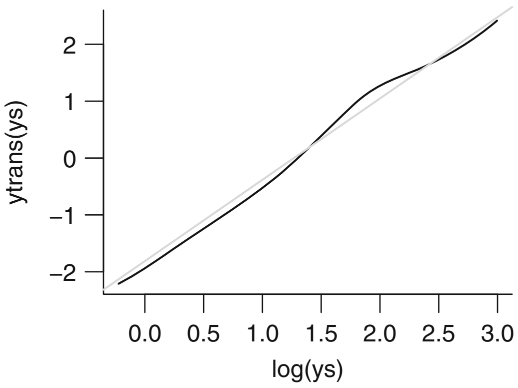

Fig. 16.2 Checking estimated against optimal transformation

Approximate linearity indicates that the estimated transformation is very log-like.[b]

Now let us obtain approximate tests of effects of each predictor. `summary` does this by setting all other predictors to reference values (e.g., medians), and comparing predicted responses for a given level of the predictor X with predictions for the lowest setting of X. The default predicted response for `summary` is the median, which is used here. Therefore tests are for differences in medians.

```
summary(f, values=list(x1=c(.2, .8), x2=c(.1, .5)))
```

```
summary.areg.boot(object = f, values = list(x1 = c(0.2, 0.8),
    x2 = c(0.1, 0.5)))

Estimates based on 300 resamples



Values to which predictors are set when estimating
effects of other predictors:

       y        x1        x2        x3
3.728843  0.500000  0.300000  2.000000
```

[b] Beware that use of a data–derived transformation in an ordinary model, as this will result in standard errors that are too small. This is because model selection is not taken into account.[186]

```
Estimates of differences of effects on Median Y (from first X
value), and bootstrap standard errors of these differences.
Settings for X are shown as row headings.


Predictor: x1

x     Differences       S.E Lower 0.95 Upper 0.95        Z      Pr(|Z|)
  0.2    0.000000        NA         NA         NA       NA           NA
  0.8    1.546992 0.2099959   1.135408   1.958577 7.366773 1.747491e-13


Predictor: x2

x     Differences       S.E Lower 0.95 Upper 0.95         Z      Pr(|Z|)
  0.1    0.000000        NA         NA         NA        NA           NA
  0.5   -1.658961 0.3163361  -2.278968  -1.038953 -5.244298 1.568786e-07


Predictor: x3

x   Differences       S.E Lower 0.95 Upper 0.95        Z      Pr(|Z|)
  a   0.0000000        NA         NA         NA       NA           NA
  b   0.8447422 0.1768244  0.4981728   1.191312 4.777295 1.776692e-06
  c   1.3526151 0.2206395  0.9201697   1.785061 6.130431 8.764127e-10
```

For example, when `x1` increases from 0.2 to 0.8 we predict an increase in median `y` by 1.55 with bootstrap standard error 0.21, when all other predictors are held to constants. Setting them to other constants will yield different estimates of the `x1` effect, as the transformation of `y` is nonlinear.

Next depict the fitted model by plotting predicted values, with `x2` varying on the x-axis, and three curves corresponding to three values of `x3`. `x1` is set to 0.5. Figure 16.3 shows estimates of both the median and the mean `y`.

```
newdat ← expand.grid(x2=seq(.05, .95, length=200),
                     x3=c('a','b','c'), x1=.5,
                     statistic=c('median','mean'))
yhat ← c(predict(f, subset(newdat, statistic=='median'),
                 statistic='median'),
         predict(f, subset(newdat, statistic=='mean'),
                 statistic='mean'))
newdat ←
  upData(newdat,
         lp = x1 + 2*abs(x2 - .5) + .3*(x3=='b') +
              .5*(x3=='c'),
         ytrue = ifelse(statistic=='median', exp(lp),
           exp(lp + 0.5*(0.5^2))), pr=FALSE)
```

```
Input object size:       45472 bytes;    4 variables
Added variable           lp
Added variable           ytrue
Added variable           pr
```

```
New object size:         69800 bytes;    7 variables
```

```
# Use Hmisc function xYplot to produce Figure 16.3
xYplot(yhat ~ x2 | statistic, groups=x3,
       data=newdat, type='l', col=1,
       ylab=expression(hat(y)),
       panel=function(...) {
         panel.xYplot(...)
         dat ← subset(newdat,
           statistic==c('median','mean')[current.column()])
         for(w in c('a','b','c'))
           with(subset(dat, x3==w),
                llines(x2, ytrue, col=gray(.7), lwd=1.5))
       }
     )
```

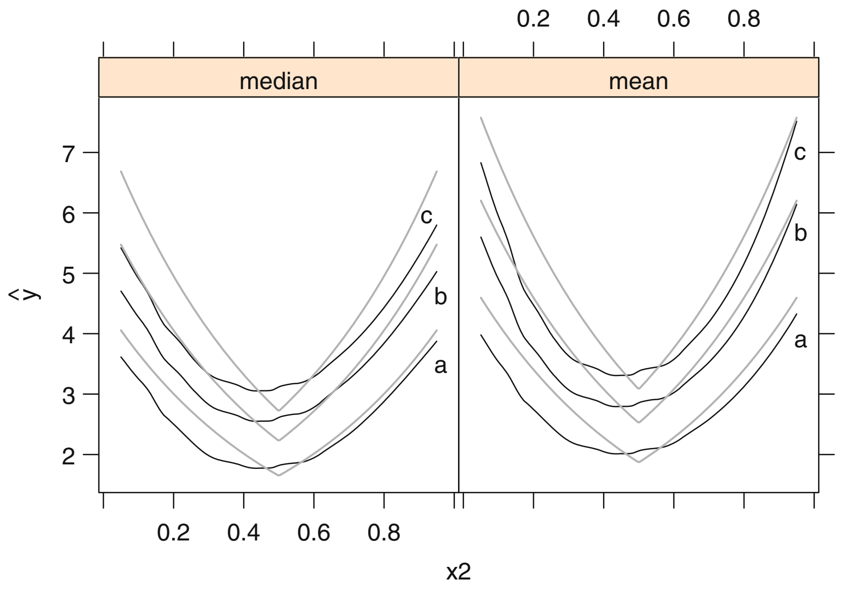

Fig. 16.3 Predicted median (left panel) and mean (right panel) y as a function of x2 and x3. True population values are shown in gray.