Chapter 8

Transformation

Transformations of the response and predictors can improve the fit and correct violations of model assumptions such as constant error variance. We may also consider adding additional predictors that are functions of the existing predictors like quadratic or crossproduct terms.

8.1 Transforming the response

Let's start with some general considerations about transforming the response.

- Suppose that you are contemplating a logged response in a simple regression situation:

  $$\log y = \beta_0 + \beta_1 x + \varepsilon$$

  In the original scale of the response, this model becomes

  $$y = \exp(\beta_0 + \beta_1 x) \cdot \exp(\varepsilon)$$

  In this model, the errors enter *multiplicatively* and not *additively* as they usually do. So the use of standard regression methods for the logged response model requires that we believe that the errors enter multiplicatively in the original scale. Notice that if we believe the proper model for y to be

  $$y = \exp(\beta_0 + \beta_1 x) + \varepsilon$$

  then we cannot linearize this model and non-linear regression methods would need to be applied.

  As a practical matter, we often do not know how the errors enter the model, additively, multiplicatively or otherwise. The usual approach is to try different transforms and then check the residuals to see whether they satisfy the conditions required for linear regression. Unless you have good information that the error enters in some particular way, this is the simplest and most appropriate way to go.

- Although you may transform the response, you will probably need to express predictions in the original scale. This is simply a matter of back-transforming. For example, in the logged model above, your prediction would be $\exp(\hat{y}_0)$. If your prediction confidence interval in the logged scale was $[l, u]$, then you would use $[\exp l, \exp u]$. This interval will not be symmetric but this may be desirable — remember what happened with the prediction confidence intervals for Galapagos data.

- Regression coefficients will need to be interpreted with respect to the transformed scale. There is no straightforward way of backtransforming them to values that can interpreted in the original scale. You cannot directly compare regression coefficients for models where the response transformation is different. Difficulties of this type may dissuade one from transforming the response even if this requires the use of another type of model such as a generalized linear model.

When you use a log transformation on the response, the regression coefficients have a particular interpretation:

$$\log \hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \ldots + \hat{\beta}_p x_p$$
$$\hat{y} = e^{\hat{\beta}_0} e^{\hat{\beta}_1 x_1} \ldots e^{\hat{\beta}_p x_p}$$

An increase of one in x_1 would multiply the predicted response (in the original scale) by $e^{\hat{\beta}_1}$. Thus when a log scale is used the regression coefficients can be interpreted in a multiplicative rather than the usual additive manner.

The Box-Cox method is a popular way to determine a tranformation on the response. It is designed for strictly positive responses and chooses the transformation to find the best fit to the data. The method transforms the response $y \to t_\lambda(y)$ where the family of transformations indexed by λ is

$$t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases}$$

For fixed $y > 0$, $t_\lambda(y)$ is continuous in λ. Choose λ using maximum likelihood. The profile log-likelihood assuming normality of the errors is

$$L(\lambda) = -\frac{n}{2}\log(\mathrm{RSS}_\lambda/n) + (\lambda - 1)\sum \log y_i$$

where RSS_λ is the RSS when $t_\lambda(y)$ is the response. You can compute $\hat{\lambda}$ exactly to maximize this but usually $L(\lambda)$ is just maximized over a grid of values such as $\{-2, -1, -1/2, 0, 1/2, 1, 2\}$. This ensures that the chosen $\hat{\lambda}$ is more easily interpreted. For example, if $\hat{\lambda} = 0.46$, it would be hard to explain what this new response means, but $\sqrt{y}$ would be easier.

Transforming the response can make the model harder to interpret so we don't want to do it unless it's really necessary. One way to check this is to form a confidence interval for λ. A $100(1-\alpha)\%$ confidence interval for λ is

$$\{\lambda : \quad L(\lambda) > L(\hat{\lambda}) - \frac{1}{2}\chi^{2(1-\alpha)}_1\}$$

This interval can be derived by inverting the likelihood ratio test of the hypothesis that $H_0 : \lambda = \lambda_0$ which uses the statistic $2(L(\hat{\lambda}) - L(\lambda_0))$ having approximate null distribution χ^2_1.

Does the response in the savings data need transformation? You'll need a function from the MASS library:

```
> library(MASS)
```

Try it out on the savings dataset and plot the results.

```
> data(savings)
> g <- lm(sr ~ pop15+pop75+dpi+ddpi,savings)
> boxcox(g,plotit=T)
> boxcox(g,plotit=T,lambda=seq(0.5,1.5,by=0.1))
```

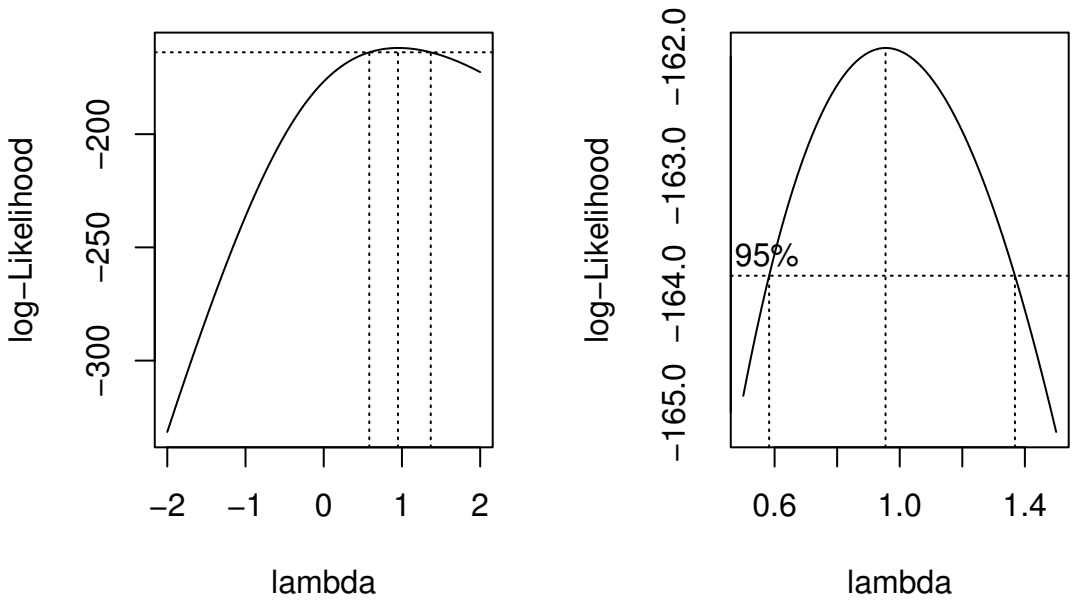

Figure 8.1: Log-likelihood plots for the Box-Cox transformation of the savings data

The first plot shown in Figure 8.1 is too broad. I narrowed the range of λ in the second plot so that we can read off the confidence interval more easily.

The confidence interval for λ runs from about 0.6 to about 1.4. We can see that there is no good reason to transform.

Now consider the Galápagos Islands dataset analyzed earlier:

```
> data(gala)
> g <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent,gala)
> boxcox(g,plotit=T)
> boxcox(g,lambda=seq(0.0,1.0,by=0.05),plotit=T)
```

The plots are shown in Figure 8.2. We see that perhaps a cube-root transformation might be best here. A square root is also a possibility as this falls just within the confidence intervals. Certainly there is a strong need to transform.

Notes

1. The Box-Cox method gets upset by outliers - if you find $\hat{\lambda} = 5$ then this is probably the reason — there can be little justification for actually making such an extreme transformation.

2. What if some $y_i < 0$? Sometimes adding a constant to all y can work provided that constant is small.

3. If $\max_i y_i / \min_i y_i$ is small then the Box-Cox won't do anything because power transforms are well approximated by linear transformations over short intervals.

4. Should the estimation of λ count as an extra parameter to be taken account of in the degrees of freedom? This is a difficult question since λ is not a linear parameter and its estimation is not part of the least squares fit.

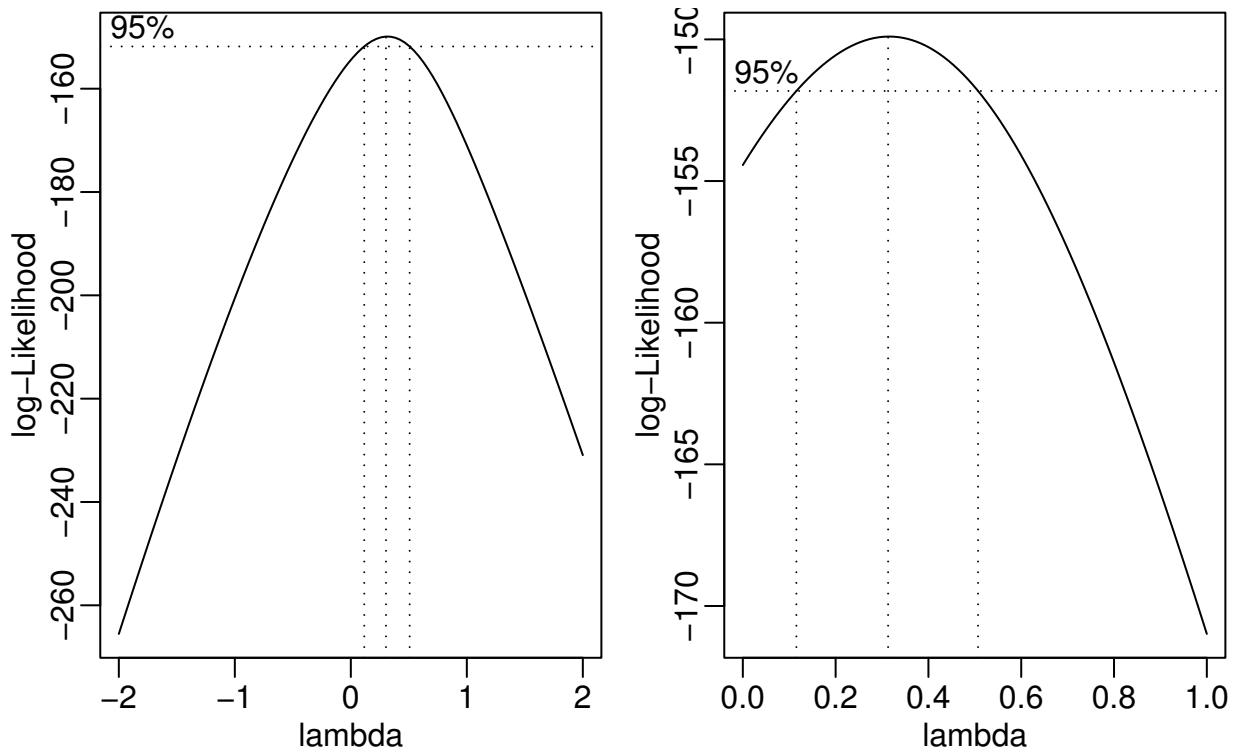

Figure 8.2: Log-likelihood plots for the Box-Cox transformation of the Galápagos data

The Box-Cox method is not the only way of transforming the predictors. For responses, that are proportions (or percentages), the logit transformation, $\log(y/(1-y))$ is often used, while for responses that are correlations, Fisher's z transform, $y = 0.5\log((1+y)/(1-y))$ is worth considering.

8.2 Transforming the predictors

You can take a Box-Cox style approach for each of the predictors, choosing the transformation to minimize the RSS. However, this takes time and furthermore the correct transformation for each predictor may depend on getting the others right too. Partial residuals are a good way of finding suggestions for transforming the predictors

8.2.1 Broken Stick Regression

Sometimes we have reason to believe that different linear regression models apply in different regions of the data. For example, in the analysis of the savings data, we observed that there were two groups in the data and we might want to fit a different model to the two parts. Suppose we focus attention on just the `pop15` predictor for ease of presentation. We fit the two regression models depending on whether `pop15` is greater or less than 35%. The two fits are shown in Figure 8.3.

```
> g1 <- lm(sr ~ pop15, savings, subset=(pop15 < 35))
> g2 <- lm(sr ~ pop15, savings, subset=(pop15 > 35))
> plot(savings$pop15,savings$sr,xlab="Pop'n under 15",ylab="Savings Rate")
```

```
> abline(v=35,lty=5)
> segments(20,g1$coef[1]+g1$coef[2]*20,35,g1$coef[1]+g1$coef[2]*35)
> segments(48,g2$coef[1]+g2$coef[2]*48,35,g2$coef[1]+g2$coef[2]*35)
```

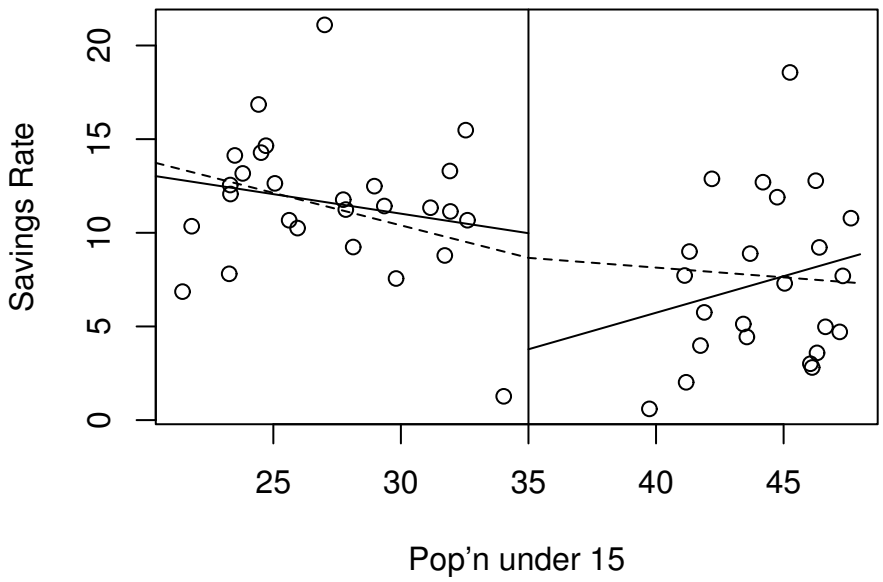

Figure 8.3: Subset regression fit is shown with the solid line while the broken stick regression is shown with the dotted line

A possible objection to this subsetted regression fit is that the two parts of the fit do not meet at the join. If we believe the fit should be continuous as the predictor varies, then this is unsatisfactory. One solution to this problem is the broken stick regression fit. Define two *basis functions*:

$$B_l(x) = \begin{cases} c-x & \texttt{if } x < c \\ 0 & \texttt{otherwise} \end{cases}$$

and

$$B_r(x) = \begin{cases} x-c & \texttt{if } x > c \\ 0 & \texttt{otherwise} \end{cases}$$

where c marks the division between the two groups. B_l and B_r form a first-order spline basis with a knotpoint at c. Sometimes B_l and B_r are called hockey-stick functions because of their shape. We can now fit a model of the form

$$y = \beta_0 + \beta_1 B_l(x) + \beta_2 B_r(x) + \varepsilon$$

using standard regression methods. The two linear parts are guaranteed to meet at c. Notice that this model uses only three parameters in contrast to the four total parameters used in the subsetted regression illustrated above. A parameter has been saved by insisting on the continuity of the fit at c.

We define the two hockey stick functions, compute and display the fit:

```
> lhs <- function(x) ifelse(x < 35,35-x,0)
> rhs <- function(x) ifelse(x < 35,0,x-35)
> gb <- lm(sr ~ lhs(pop15) + rhs(pop15), savings)
```

```
> x <- seq(20,48,by=1)
> py <- gb$coef[1]+gb$coef[2]*lhs(x)+gb$coef[3]*rhs(x)
> lines(x,py,lty=2)
```

The two (dotted) lines now meet at 35 as shown in Figure 8.3. The intercept of this model is the value of the response at the join.

We might question which fit is preferable in this particular instance. For the high `pop15` countries, we see that the imposition of continuity causes a change in sign for the slope of the fit. We might argue that the two groups of countries are so different and that there are so few countries in the middle region, that we might not want to impose continuity at all.

We can have more than one knotpoint simply by defining more pairs of basis functions with different knotpoints. Broken stick regression is sometimes called *segmented regression*. Allowing the knotpoints to be parameters is worth considering but this will result in a nonlinear model.

8.2.2 Polynomials

Another way of generalizing the $X\beta$ part of the model is to add polynomial terms. In the one-predictor case, we have

$$y = \beta_0 + \beta_1 x + \ldots + \beta_d x^d + \varepsilon$$

which allows for a more flexible relationship although we usually don't believe it exactly represents any underlying reality.

There are two ways to choose d:

1. Keep adding terms until the added term is not statistically significant.

2. Start with a large d — eliminate not statistically significant terms starting with the highest order term.

Warning: Do not eliminate lower order terms from the model even if they are not statistically significant. An additive change in scale would change the t-statistic of all but the highest order term. We would not want the conclusions of our study to be so brittle to such changes in the scale which ought to be inconsequential.

Let's see if we can use polynomial regression on the `ddpi` variable in the savings data. First fit a linear model:

```
> summary(lm(sr ~ ddpi,savings))
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    7.883      1.011    7.80  4.5e-10
ddpi           0.476      0.215    2.22    0.031

Residual standard error: 4.31 on 48 degrees of freedom
Multiple R-Squared: 0.0929,     Adjusted R-squared: 0.074
F-statistic: 4.92 on 1 and 48 degrees of freedom,       p-value: 0.0314
```

p-value of `ddpi` is significant so move on to a quadratic term:

```
> summary(lm(sr ~ ddpi+I(ddpi^2),savings))
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.1304     1.4347    3.58  0.00082
```

```
ddpi          1.7575     0.5377    3.27  0.00203
I(ddpi^2)    -0.0930     0.0361   -2.57  0.01326

Residual standard error: 4.08 on 47 degrees of freedom
Multiple R-Squared: 0.205,      Adjusted R-squared: 0.171
F-statistic: 6.06 on 2 and 47 degrees of freedom,       p-value: 0.00456
```

Again the p-value of ddpi^2 is significant so move on to a cubic term:

```
> summary(lm(sr ~ ddpi+I(ddpi^2)+I(ddpi^3),savings))
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.145360   2.198606    2.34    0.024
ddpi         1.746017   1.380455    1.26    0.212
I(ddpi^2)   -0.090967   0.225598   -0.40    0.689
I(ddpi^3)   -0.000085   0.009374   -0.01    0.993

Residual standard error: 4.12 on 46 degrees of freedom
Multiple R-Squared: 0.205,      Adjusted R-squared: 0.153
F-statistic: 3.95 on 3 and 46 degrees of freedom,       p-value: 0.0137
```

p-value of ddpi^3 is not significant so stick with the quadratic. What do you notice about the other p-values? Why do we find a quadratic model when the previous analysis on transforming predictors found that the ddpi variable did not need transformation? Check that starting from a large model (including the fourth power) and working downwards gives the same result.

To illustrate the point about the significance of lower order terms, suppose we transform ddpi by subtracting 10 and refit the quadratic model:

```
> savings <- data.frame(savings,mddpi=savings$ddpi-10)
> summary(lm(sr ~ mddpi+I(mddpi^2),savings))
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  13.4070     1.4240    9.41  2.2e-12
mddpi        -0.1022     0.3027   -0.34    0.737
I(mddpi^2)   -0.0930     0.0361   -2.57    0.013

Residual standard error: 4.08 on 47 degrees of freedom
Multiple R-Squared: 0.205,      Adjusted R-squared: 0.171
F-statistic: 6.06 on 2 and 47 degrees of freedom,       p-value: 0.00456
```

We see that the quadratic term remains unchanged but the linear term is now insignificant. Since there is often no necessary importance to zero on a scale of measurement, there is no good reason to remove the linear term in this model but not in the previous version. No advantage would be gained.

You have to refit the model each time a term is removed and for large d there can be problem with numerical stability. Orthogonal polynomials get round this problem by defining

$$\begin{aligned} z_1 &= a_1 + b_1 x \\ z_2 &= a_2 + b_2 x + c_2 x^2 \\ z_3 &= a_3 + b_3 x + c_3 x^2 + d_3 x^3 \end{aligned}$$

etc. where the coefficients $a, b, c, \ldots$ are chosen so that $z_i^T z_j = 0$ when $i \neq j$. The z are called orthogonal polynomials. The value of orthogonal polynomials has declined with advances in computing speeds although they are still worth knowing about because of their numerical stability and ease of use. The `poly()` function constructs Orthogonal polynomials.

```
> g <- lm(sr ~ poly(ddpi,4),savings)
> summary(g)
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      9.6710     0.5846   16.54   <2e-16
poly(ddpi, 4)1   9.5590     4.1338    2.31    0.025
poly(ddpi, 4)2 -10.4999     4.1338   -2.54    0.015
poly(ddpi, 4)3  -0.0374     4.1338   -0.01    0.993
poly(ddpi, 4)4   3.6120     4.1338    0.87    0.387

Residual standard error: 4.13 on 45 degrees of freedom
Multiple R-Squared: 0.218,      Adjusted R-squared: 0.149
F-statistic: 3.14 on 4 and 45 degrees of freedom,       p-value: 0.0232
```

Can you see how we come to the same conclusion as above with just this summary? We can verify the orthogonality of the design matrix when using orthogonal polynomials:

```
> x <- model.matrix(g)
> dimnames(x) <- list(NULL,c("Int","power1","power2","power3","power4"))
> round(t(x) %*% x,3)
       Int power1 power2 power3 power4
Int     50      0      0      0      0
power1   0      1      0      0      0
power2   0      0      1      0      0
power3   0      0      0      1      0
power4   0      0      0      0      1
```

You can have more than two predictors as can be seen in this *response surface* model:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2$$

8.3 Regression Splines

Polynomials have the advantage of smoothness but the disadvantage that each data point affects the fit globally. This is because the power functions used for the polynomials take non-zero values across the whole range of the predictor. In contrast, the broken stick regression method localizes the influence of each data point to its particular segment which is good but we do not have the same smoothness as with the polynomials. There is a way we can combine the beneficial aspects of both these methods — smoothness and local influence — by using *B-spline* basis functions.

We may define a cubic B-spline basis on the interval $[a, b]$ by the following requirements on the interior basis functions with knot-points at $t_1, \ldots, t_k$.

1. A given basis function is non-zero on interval defined by four successive knots and zero elsewhere. This property ensures the local influence property.

2. The basis function is a cubic polynomial for each sub-interval between successive knots

3. The basis function is continuous and continuous in its first and second derivatives at each knot point. This property ensures the smoothness of the fit.

4. The basis function integrates to one over its support

The basis functions at the ends of the interval are defined a little differently to ensure continuity in derivatives at the edge of the interval. A full definition of B-splines and more details about their properties may be found in "A practical guide to splines" by Carl De Boor.

Let's see how the competing methods do on a constructed example. Suppose we know the true model is

$$y = \sin^3(2\pi x^3) + \varepsilon, \qquad \varepsilon \sim N(0, (0.1)^2)$$

The advantage of using simulated data is that we can see how close our methods come to the truth. We generate the data and display it in Figure 8.3.

```
> funky <- function(x) sin(2*pi*x^3)^3
> x <- seq(0,1,by=0.01)
> y <- funky(x) + 0.1*rnorm(101)
> matplot(x,cbind(y,funky(x)),type="pl",ylab="y",pch=18,lty=1,
            main="True Model")
```

We see how an orthogonal polynomial bases of orders 4 and 12 do in fitting this data:

```
> g4 <- lm(y ~ poly(x,4))
> g12 <- lm(y ~ poly(x,12))
> matplot(x,cbind(y,g4$fit,g12$fit),type="pll",ylab="y",pch=18,
                  lty=c(1,2),main="Orthogonal Polynomials")
```

The two fits are shown in the second panel of Figure 8.3. We see that order 4 is a clear underfit. Order 12 is much better although the fit is too wiggly in the first section and misses the point of inflection.

We now create the B-spline basis. You need to have three additional knots at the start and end to get the right basis. I have chosen to the knot locations to put more in regions of greater curvature. I have used 12 basis functions for comparability to the orthogonal polynomial fit.

```
> library(splines)
> knots <- c(0,0,0,0,0.2,0.4,0.5,0.6,0.7,0.8,0.85,0.9,1,1,1,1)
> bx <- splineDesign(knots,x)
> gs <- lm(y ~ bx)
> matplot(x,bx,type="l",main="B-spline basis functions")
> matplot(x,cbind(y,gs$fit),type="pl",ylab="y",pch=18,lty=1,
  main="Spline fit")
```

The basis functions themselves are shown in the third panel of Figure 8.3 while the fit itself appears in the fourth panel. We see that the fit comes very close to the truth.

Regression splines are useful for fitting functions with some flexibility provided we have enough data. We can form basis functions for all the predictors in our model but we need to be careful not to use up too many degrees of freedom.

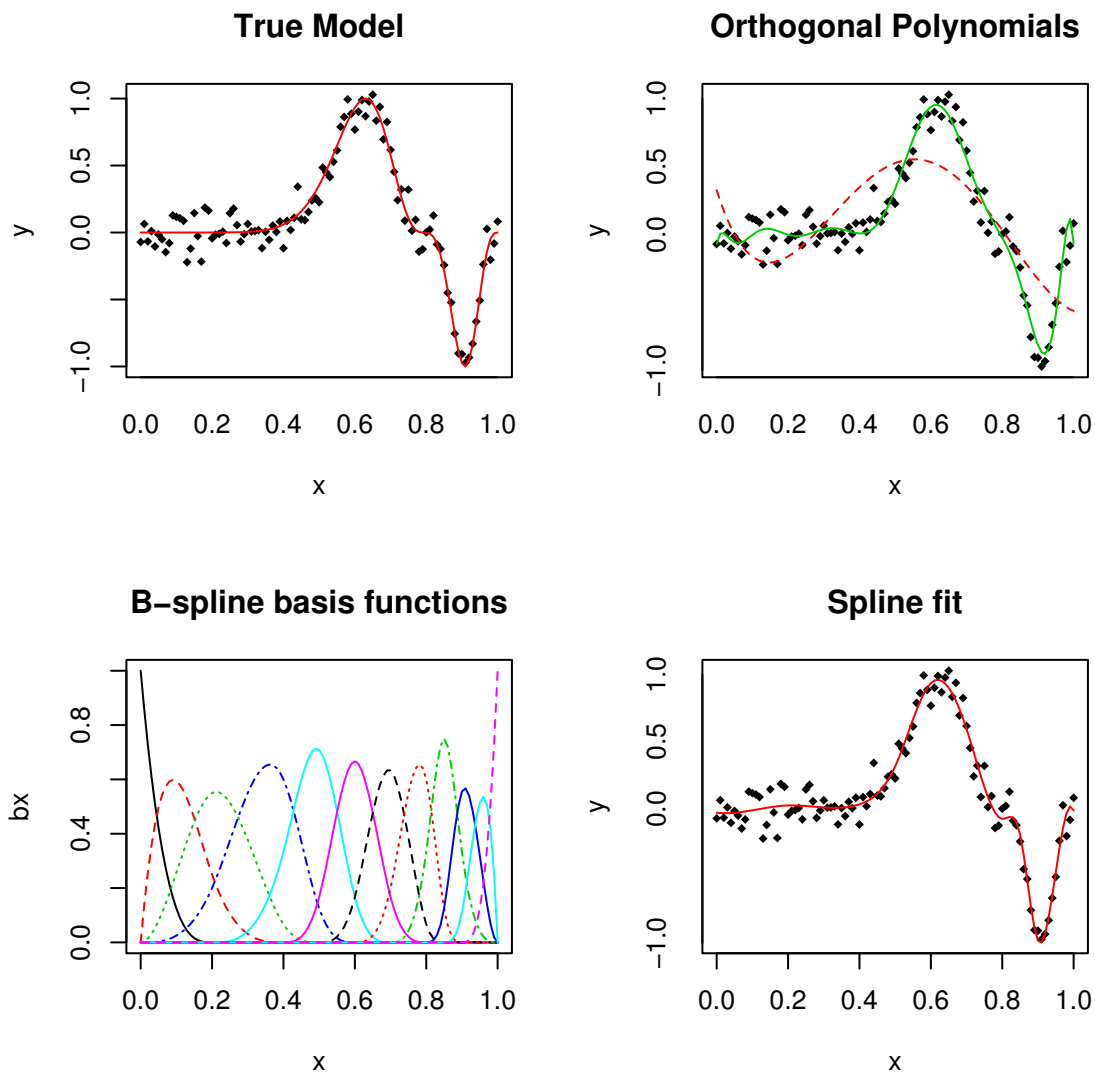

Figure 8.4: Orthogonal Splines compared to B-splines

8.4 Modern Methods

The methods described above are somewhat awkward to apply exhaustively and even then they may miss important structure because of the problem of trying to find good transformations on several variables simultaneously. One recent approach is the additive model:

$$y = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p) + \varepsilon$$

where nonparametric regression techniques are used to estimated the f_i's. Alternatively, you could implement this using the regression spline bases for each predictor variable. Other techniques are ACE, AVAS, Regression Trees, MARS and neural networks.

It is important to realize the strengths and weaknesses of regression analysis. For larger data sets with relatively little noise, more recently developed complex models will be able to fit the data better while keeping the number of parameters under control. For smaller data sets or where the noise level is high (as

is typically found in the social sciences), more complex models are not justified and standard regression is most effective. One relative advantage of regression is that the models are easier to interpret in contrast to techniques like neural networks which are usually only good for predictive purposes.