LSIRM Statistical/Machine Learning
Lecture 4: Linearity Diagnostics

Dave Armstrong

University of Western Ontario
Department of Political Science

e: dave.armstrong@uwo.ca
w: www.quantoid.net/teachwlu/

Outline

- Splines
  - Basis Functions
  - B-splines
  - Importance of number of knots and knot placement
- Worked example

Definition of Splines

Splines are:

> *... piecewise regression functions we constrain to join at points called knots (Keele 2007, 70)*

- In their simplest form, they are dummy regressors that we use to force the regression line to change direction at some value(s) of X.
- These are similar in spirit to LPR models where we use a subset of data to fit local regressions (but the window doesn't move here).
- These are also allowed to take any particular functional form, but they are a bit more constrained than the LPR model.

Splines vs. LPR Models

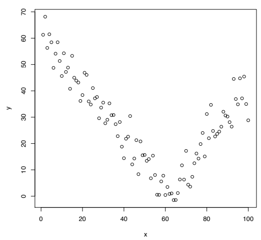
- Splines provide a better MSE fit to the data.
  - Where $MSE\left(\hat{\theta}\right) = \mathrm{Var}\left(\hat{\theta}\right) + \left(\mathrm{Bias}\left(\hat{\theta}, \theta\right)\right)^2$
  - Generally, LPR models will have smaller bias, but much greater variance.
- Splines can be designed to prevent over-fitting (smoothing splines)
- Splines are more easily incorporated in *semi*-parametric models.

Regression Splines

We start with the following familiar model:

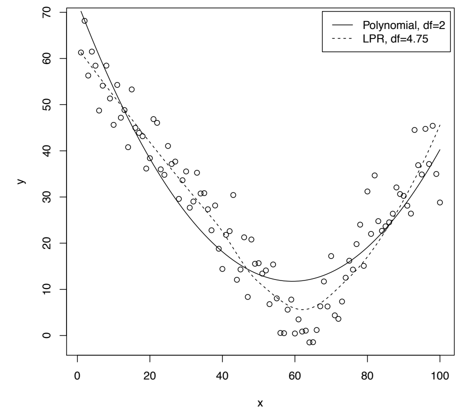
$$y = f(x) + \varepsilon$$

Here, we would like to estimate this with one model rather than a series of local models.



Failure of Polynomials and LPR

Given what we already learned, we could fit a quadratic polynomial or a LPR:



Simple Example

In this simple example, it is easy to figure out what sort of model we want:

- It appears that the relationship between x and y would be well-characterized by two lines.
  - One with a negative slope in the range $x = [0, 60)$
  - One with a positive slope in the range $x = [60, 100]$

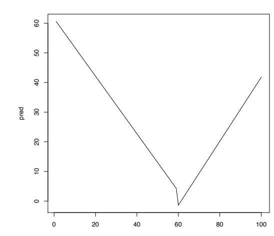
These are all the things we need to know right now to model the relationship.

Dummy Interactions

You might ask, couldn't we just use an interaction between x and a dummy variable coded 1 if $x > 60$ and zero otherwise.

$$y = b_0 + b_1x_1 + b_2d + b_3x \times d + e$$

This seems like a perfectly reasonable thing to do. What can it give you though:



Basis Functions

A basis function is really just a function that transforms the values of X. So, instead of estimating:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

we estimate:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \ldots + \beta_k b_k(x_i) + \varepsilon_i$$

The basis functions $b_k(\cdot)$ are known ahead of time (not estimated by the model).

- We can think of polynomials as basis functions where $b_j(x_i) = x_i^j$

Piecewise Polynomials

One way that we can think about regression splines is as piecewise polynomial functions:

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \varepsilon_i & x_i < c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \varepsilon_i & x_i \geq c \end{cases}$$

Just as above though, these polynomials are unconstrained and can generate a discontinuity at the *knot* location c.

Constraining the Model

To constrain the model, the splines are constructed:

- such that the first and second derivatives of the function continuous.
- Each constraint reduces the number of degrees of freedom we use by one.
- In general, the model uses: Polynomial Degree + # Knots + 1 degrees of freedom

Truncated Power Basis Functions

The easiest set of Spline functions to consider (for knot location k) are called truncated power functions, defined as:

$$h(x, k) = (x - k)_+^3 = \begin{cases} (x - k)^3 & \text{if } x > k \\ 0 & \text{otherwise} \end{cases}$$

When using these basis functions in, we put the full (i.e., global) parametric function in and a truncated power function of degree n for each knot.

Linear Truncated Power Functions

To use the truncated power basis for our problem, we need:

- The global linear model
- One truncated power function for the x values greater than the knot location (60).

$$y = b_0 + b_1 x + b_2 (x - 60)^1_+ + e$$

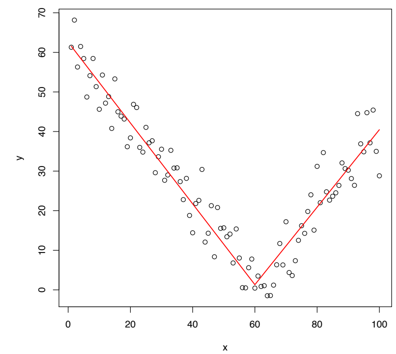
This sets up essentially 2 equations:

$$\begin{aligned} x \le 60 &: y = b_0 + b_1 x \\ x > 60 &: y = b_0 + b_1 x + b_2(x - 60) = (b_0 - 60 b_2) + (b_1 + b_2) x \end{aligned}$$

Notice that here we are only estimating 3 parameters, where the interaction would estimate 4 parameters. Thus, this is a constrained version of the interaction.

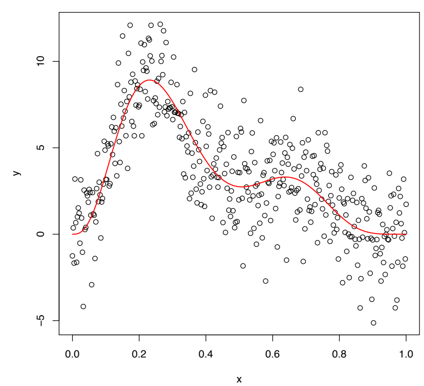
Fixing the Discontinuity

Including x and $(x - 60)_+$ as regressors, which generates the following predictions:
(Note, with piecewise linear functions, we're not constraining the derivatives to be continuous).



Example: Cubic Spline

Consider the following relationship:



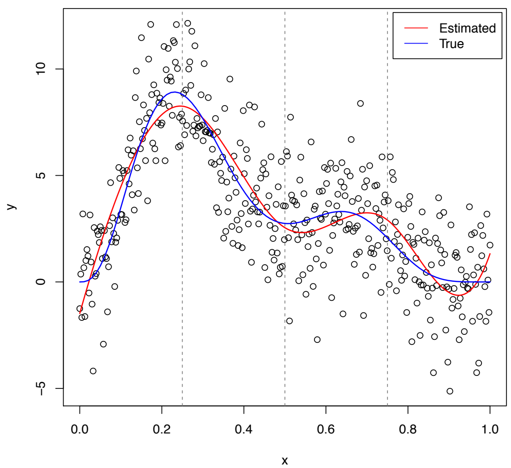
Truncated Power Basis Functions: Cubic Spline

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \sum_{m-1}^{\#\text{ knots}} b_{k+3}(x - k_m)^3_+$$

Let's consider our example with 3 knots $k = \{.25, .5, .75\}$

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3) + I((x - k[1])^3 * (x >=
##     k[1])) + I((x - k[2])^3 * (x >= k[2])) + I((x - k[3])^3 *
##     (x >= k[3])))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1553 -1.3015  0.0084  1.3594  5.3203
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     -1.4789     0.6386  -2.316 0.021079 *
## x                               68.4874    14.2232   4.815 2.10e-06 ***
## I(x^2)                         -72.4938    83.6956  -0.866 0.386931
## I(x^3)                        -183.0427   139.7292  -1.310 0.190967
## I((x - k[1])^3 * (x >= k[1]))  690.4099   186.6713   3.699 0.000248 ***
## I((x - k[2])^3 * (x >= k[2])) -978.1580   103.0594  -9.491  < 2e-16 ***
## I((x - k[3])^3 * (x >= k[3])) 1334.8591   186.6713   7.151 4.24e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.02 on 393 degrees of freedom
## Multiple R-squared:  0.6416,Adjusted R-squared:  0.6361
## F-statistic: 117.3 on 6 and 393 DF,  p-value: < 2.2e-16
```

Plotting the Curve



Problems with Truncated Power Basis Functions

- Highly collinear and can lead to instability and singularities (i.e., computationally bad stuff) at worst.
- Not as "local" as some other options, the support of the piecewise functions can be over the whole range of the data or nearly the whole range of the data.
- Can produce erratic tail behavior.

Other basis functions, like the B-spline basis functions solve all of these problems:

- Reduces collinearity (though doesn't eliminate it)
- Support of the function is more narrowly bounded.
- Uses knots at the boundaries of x and assumes linearity beyond the knots.

Example: B-spline

```
library(splines)
csmod2 <- lm(y ~ bs(x, knots=c(.25,.5,.75)))
summary(csmod2)
```

```
##
## Call:
## lm(formula = y ~ bs(x, knots = c(0.25, 0.5, 0.75)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1553 -1.3015  0.0084  1.3594  5.3203
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          -1.4789     0.6386  -2.316  0.02108
## bs(x, knots = c(0.25, 0.5, 0.75))1   5.7073     1.1853   4.815  2.1e-06
## bs(x, knots = c(0.25, 0.5, 0.75))2  14.1013     0.7617  18.513  < 2e-16
## bs(x, knots = c(0.25, 0.5, 0.75))3   0.4703     0.9425   0.499  0.61808
## bs(x, knots = c(0.25, 0.5, 0.75))4   8.1830     0.8696   9.410  < 2e-16
## bs(x, knots = c(0.25, 0.5, 0.75))5  -1.8705     0.9846  -1.900  0.05819
## bs(x, knots = c(0.25, 0.5, 0.75))6   2.8050     0.8880   3.159  0.00171
##
## (Intercept)                        *
## bs(x, knots = c(0.25, 0.5, 0.75))1 ***
## bs(x, knots = c(0.25, 0.5, 0.75))2 ***
## bs(x, knots = c(0.25, 0.5, 0.75))3
## bs(x, knots = c(0.25, 0.5, 0.75))4 ***
## bs(x, knots = c(0.25, 0.5, 0.75))5 .
## bs(x, knots = c(0.25, 0.5, 0.75))6 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.02 on 393 degrees of freedom
## Multiple R-squared:  0.6416,Adjusted R-squared:  0.6361
## F-statistic: 117.3 on 6 and 393 DF,  p-value: < 2.2e-16
```


Interpreting Spline Coefficients

So, how do you interpret the spline coefficients?

- You don't.
- Remember that these are all functions of x, so we cannot change the values of one component of the basis function while holding the others constant, the others would have to change, too.

Choices in Spline Models

Degree: the analyst has to choose the degree of the polynomial fit to the subsets of the data.

Number of knots: the analyst has to choose the number of knots

Location of knots: Often, knots are spaced evenly over the support of the data (i.e., the range of x), but that needn't be the case.

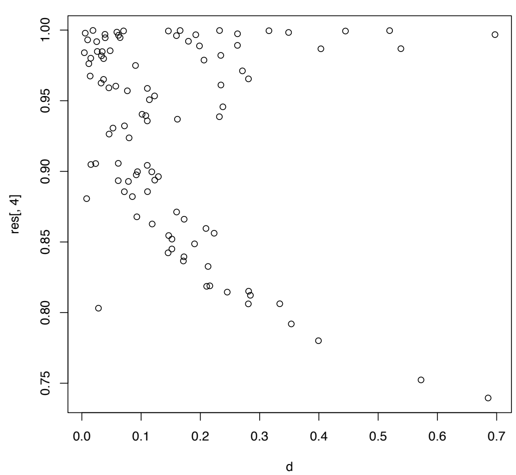
- Knot placement can be guided by theory if possible.
- Otherwise, for the functions we generally need to estimate, a few knots should probably work just fine.

How important is knot placement?

I did a simulation where I did the following:

1. Create y using a B-spline with knot locations at $k = \{.1, .6, .8\}$ plus some random normal error
2. Randomly draw knot values from a uniform distribution $k^* \sim U(0, 1)$ (and ensure they are ordered from smallest to largest).
3. Estimate a model using the randomly drawn knot locations and save predictions $\hat{y}^*$
4. Correlate y and $\hat{y}^*$
5. Plot correlations as a function of the euclidian distance between k and k^* (i.e., $\sum_{i=1}^{3}(k_i - k_i^*)^2$)

Plot



How Important is Knot Placement? II

- So long as the polynomial degree is reasonably high (3 should be high enough for what we do, but 4 might be useful if you have a very complicated function), knot placement is not particularly important.
- Use theory, if it exists, to place knots.
- If theory doesn't exist, knots placed evenly across the range of $\boldsymbol{x}$ will, in general, minimize error.
  - If you think about the knots as random variables (because we don't know their values) and further that they are distributed uniformly (i.e., neither middle or extreme values are more likely), then technically evenly spaced knots minimize distance to the true, but unknown knots.

How Important is Polynomial Degree?

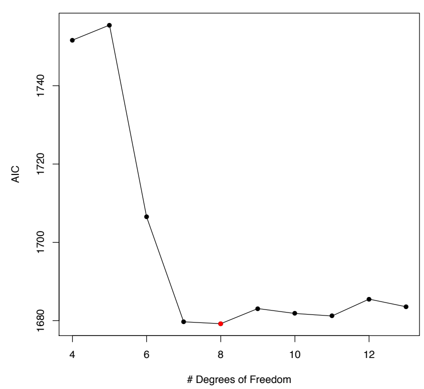
- Pretty important, particularly if we don't know or have a really good sense of where the knots should be.
  - B-splines are more forgiving of knot placement errors the higher the polynomial degree.
- Generally no good reason to use something more restrictive than a cubic spline.
  - We are generally not trying to model particularly complicated functions.
- More knots are more likely to be used than a higher polynomial degree to make the function more flexible.

How Important is the Number of Knots

- Flexibility increases with number of knots and polynomial degree.
- Increasing number of knots can make the function more flexible.
- We can use AIC, BIC or Cross-Validation to choose number of knots.

AIC for Number of Knots

```
library(DAMisc)
tmp <- data.frame(x=x, y=y)
NKnots(y ~ 1, "x", tmp, plot=TRUE, criterion="AIC")
```



Worked Example

```
library(car)
library(foreign)
dat <- read.dta(
   "http://www.quantoid.net/files/reg3/jacob.dta")
dat$perotnorm <- bcnPower(dat$perotvote, 1.55, gamma=33)
rawlm <- lm(chal_vote ~ perotnorm + chal_spend +
    exp_chal, data=dat)
```

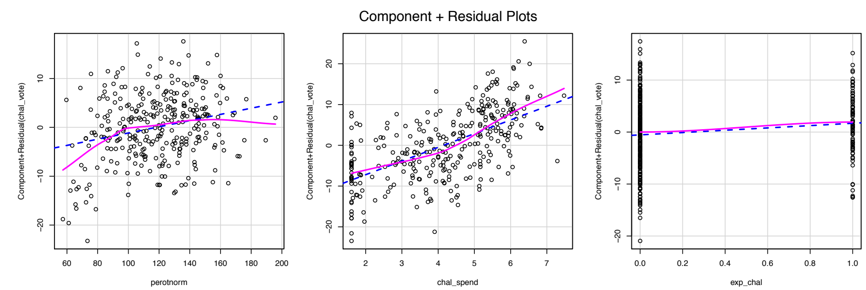
Raw Model

```
summary(rawlm)
```

```
##
## Call:
## lm(formula = chal_vote ~ perotnorm + chal_spend + exp_chal, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.3971  -4.5456   0.3237   4.4299  17.9832
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.44642    1.94949   7.410 1.22e-12 ***
## perotnorm    0.06341    0.01451   4.369 1.71e-05 ***
## chal_spend   3.35775    0.28007  11.989  < 2e-16 ***
## exp_chal     2.22782    0.99170   2.246   0.0254 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.781 on 308 degrees of freedom
## Multiple R-squared:  0.4487,Adjusted R-squared:  0.4433
## F-statistic: 83.55 on 3 and 308 DF,  p-value: < 2.2e-16
```


GDP

```
crPlots(rawlm, layout=c(1,3))
```



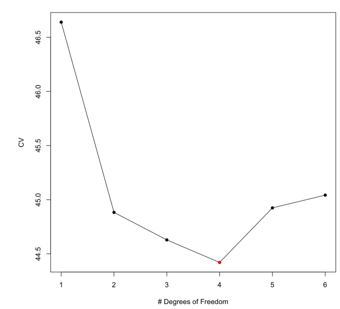
Perot Vote Polynomial

```
mod <- lm(chal_vote ~ poly(perotnorm, 2) + chal_spend + exp_chal, data=dat)
summary(mod)
```

```
##
## Call:
## lm(formula = chal_vote ~ poly(perotnorm, 2) + chal_spend + exp_chal,
##     data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.3537  -4.6612   0.4819   4.1575  17.2497
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          22.2604     1.1263  19.765  < 2e-16 ***
## poly(perotnorm, 2)1  30.1626     6.6995   4.502 9.56e-06 ***
## poly(perotnorm, 2)2 -25.2270     6.6594  -3.788 0.000183 ***
## chal_spend            3.2799     0.2750  11.929  < 2e-16 ***
## exp_chal              2.2539     0.9709   2.321 0.020919 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.638 on 307 degrees of freedom
## Multiple R-squared:  0.4733,Adjusted R-squared:  0.4664
## F-statistic: 68.96 on 4 and 307 DF,  p-value: < 2.2e-16
```

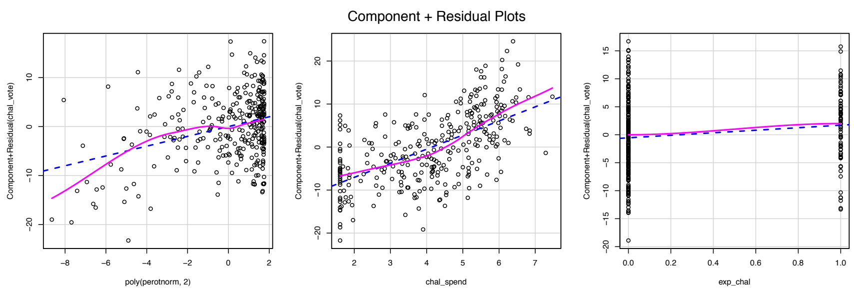

Is 2 DF Enough for Perot Vote?

```
library(DAMisc)
NKnots(chal_vote ~  chal_spend + exp_chal,
  "perotnorm", max.knots=3, data=dat, includePoly=T,
  criterion="CV", plot=T, cviter=10)
```



More CR Plots

```r
crPlots(mod, layout=c(1,3))
```



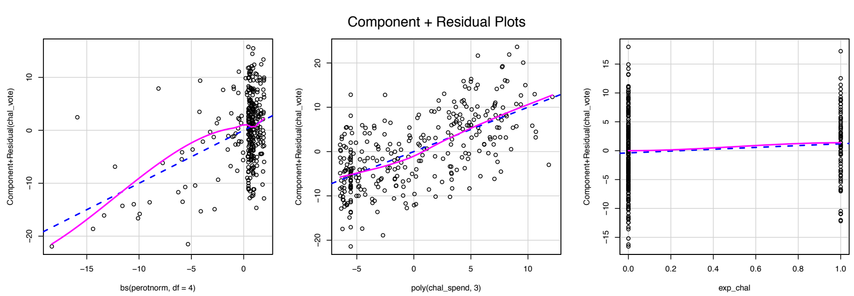
Challenger Spending

```r
mod2 <- lm(chal_vote ~ bs(perotnorm, df=4) + poly(chal_spend,3) + exp_chal, data=dat)
summary(mod2)
```

```
##
## Call:
## lm(formula = chal_vote ~ bs(perotnorm, df = 4) + poly(chal_spend,
##     3) + exp_chal, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.1918  -4.3469   0.2869   3.9500  18.3635
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              17.6907     2.7834   6.356 7.60e-10 ***
## bs(perotnorm, df = 4)1  23.3641     4.6557   5.018 8.90e-07 ***
## bs(perotnorm, df = 4)2  13.9058     3.1374   4.432 1.30e-05 ***
## bs(perotnorm, df = 4)3  23.9656     4.7769   5.017 8.96e-07 ***
## bs(perotnorm, df = 4)4  15.7760     5.1871   3.041 0.002561 **
## poly(chal_spend, 3)1    86.9915     7.2014  12.080  < 2e-16 ***
## poly(chal_spend, 3)2    25.6738     6.5240   3.935 0.000103 ***
## poly(chal_spend, 3)3   -12.4961     6.5378  -1.911 0.056903 .
## exp_chal                 1.5835     0.9532   1.661 0.097712 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.38 on 303 degrees of freedom
## Multiple R-squared:  0.5198,Adjusted R-squared:  0.5072
## F-statistic: 41.01 on 8 and 303 DF,  p-value: < 2.2e-16
```

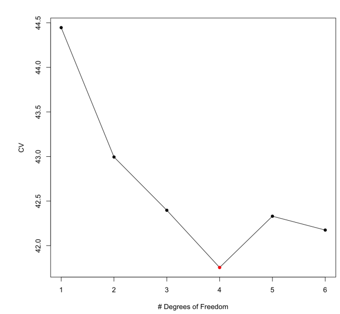

CR Plots

```r
crPlots(mod2, layout=c(1,3))
```



Knots for Challenger Spending

```r
NKnots(chal_vote ~  bs(perotnorm, 4) +  exp_chal,
  "chal_spend", max.knots=3, data=dat, includePoly=T,
  criterion="CV", plot=T, cviter=10)
```



Spline for Challenger Spending

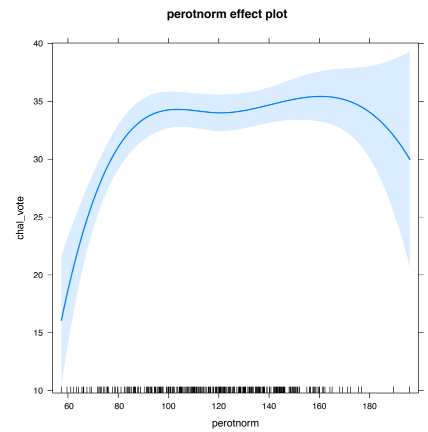
```
mod3 <- lm(chal_vote ~ bs(perotnorm, df=4) + bs(chal_spend,df=4) + exp_chal, data=dat)
summary(mod3)
```

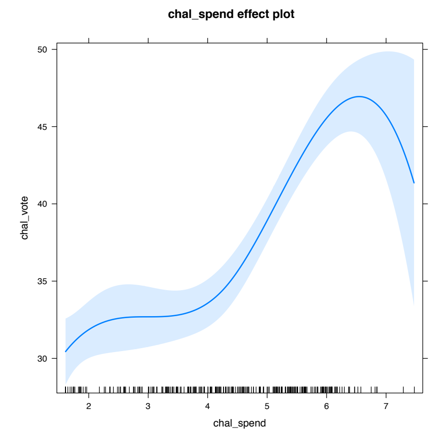
```
##
## Call:
## lm(formula = chal_vote ~ bs(perotnorm, df = 4) + bs(chal_spend,
##     df = 4) + exp_chal, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.4429  -4.5351   0.1465   4.1241  18.5350
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              12.1453     2.6481   4.586 6.62e-06 ***
## bs(perotnorm, df = 4)1   21.8213     4.6148   4.729 3.47e-06 ***
## bs(perotnorm, df = 4)2   13.5091     3.0948   4.365 1.75e-05 ***
## bs(perotnorm, df = 4)3   23.1377     4.7155   4.907 1.52e-06 ***
## bs(perotnorm, df = 4)4   14.0190     5.1429   2.726  0.00679 **
## bs(chal_spend, df = 4)1   4.4007     2.7002   1.630  0.10419
## bs(chal_spend, df = 4)2  -4.7143     3.1110  -1.515  0.13073
## bs(chal_spend, df = 4)3  24.9053     3.0987   8.037 2.09e-14 ***
## bs(chal_spend, df = 4)4  10.8618     4.1055   2.646  0.00858 **
## exp_chal                  1.2397     0.9458   1.311  0.19095
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.288 on 302 degrees of freedom
## Multiple R-squared:  0.5351,Adjusted R-squared:  0.5212
## F-statistic: 38.62 on 9 and 302 DF,  p-value: < 2.2e-16
```


Effects

```
library(effects)
plot(effect("bs(perotnorm, df=4)", mod3, xlevels=100))
```

```
plot(effect("bs(chal_spend, df=4)", mod3, xlevels=100))
```





Testing Functional Form Hypotheses

```
NKnotsTest(chal_vote ~  bs(chal_spend,df=4) + exp_chal,
  "perotnorm", data=dat, targetdf=4, adjust="none")
```

```
##                 F       DF1 DF2 p(F)  Clarke Pr(Better) p(Clarke) Delta_AIC
## DF=4 vs. DF=1  10.451* 3   302 0.000 168    0.538      0.193     24.818
## DF=4 vs. DF=2  6.615*  2   302 0.002 149    0.478      0.462     9.377
## DF=4 vs. DF=3  7.356*  1   302 0.007 163    0.522      0.462     5.509
##    Target
## DF=4 vs. DF=5  NANA    1   301 NA    239*   0.766      0.000 (T) 2.168
## DF=4 vs. DF=6  0.250   2   300 0.779 243*   0.779      0.000 (T) 3.479
## DF=4 vs. DF=7  0.645   3   299 0.587 227*   0.728      0.000 (T) 3.988
## DF=4 vs. DF=8  0.920   4   298 0.453 233*   0.747      0.000 (T) 4.172
## DF=4 vs. DF=9  0.874   5   297 0.499 249*   0.798      0.000 (T) 5.442
## DF=4 vs. DF=10 0.789   6   296 0.579 244*   0.782      0.000 (T) 7.048
## DF=4 vs. DF=11 0.707   7   295 0.667 242*   0.776      0.000 (T) 8.812
## DF=4 vs. DF=12 0.716   8   294 0.677 241*   0.772      0.000 (T) 9.979
## DF=4 vs. DF=13 0.674   9   293 0.732 240*   0.769      0.000 (T) 11.606
##                Delta_AICc Delta_BIC
## DF=4 vs. DF=1  24.413     13.589
## DF=4 vs. DF=2  9.093      1.891
## DF=4 vs. DF=3  5.360      1.766
##    Target
## DF=4 vs. DF=5  2.332      5.911
## DF=4 vs. DF=6  3.821      10.965
## DF=4 vs. DF=7  4.523      15.217
## DF=4 vs. DF=8  4.913      19.144
## DF=4 vs. DF=9  6.406      24.157
## DF=4 vs. DF=10 8.250      29.506
## DF=4 vs. DF=11 10.267     35.014
## DF=4 vs. DF=12 11.701     39.923
## DF=4 vs. DF=13 13.612     45.293
```


Smoothing Splines and GAMs
- Smoothing Splines
- Generalized Additive Models
  - Example: Canadian Occupational Prestige
  - Interpretation
  - RSS and Degrees of Freedom
  - Model Testing

Smoothing Splines

A common criticism of both LPR and Cubic Spline models in the social sciences is that they are *too* flexible.

- A model that is overfit has: "too many parameters relative to the amount of data and cause random variation in the data to appear as a systematic effect" (Keele 2007, 90)

Spline models, since they are estimated with OLS, minimize the following:

$$SS(f) = \sum_{i=1}^{n} [y - f(x)]^2$$

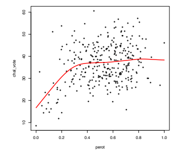
Smoothing splines penalize extra parameters to place extra weight on parsimony; they minimize the following:

$$SS(f) = \sum_{i=1}^{n} [y - f(x)]^2 + \lambda \int_{x_1}^{x_n} [f''(x)]^2 dx$$

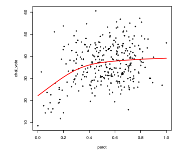
The second term imposes a "roughness penalty".

Choosing the Smoothing

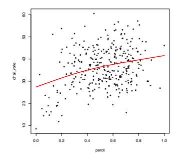
You can make either one of two choices to govern how smooth the curve looks - the degrees of freedom or λ (actually in **R**, λ is a function of the `spar` argument to the command).



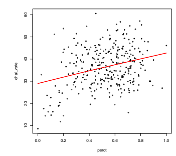
(a) spar = 0



(b) spar = 0.1



(c) spar = 1



(d) spar = 10

Choosing λ

- How do we determine the appropriate value for the smoothing parameter λ given a data set
- The same value of λ is unlikely to work equally well with every data set
- The "best" choice of smoothing parameter is one that minimizes the mean squared error:

$$L(\lambda) = n^{-1} \sum_{i=1}^{n} (f(x_i) - f_\lambda(x_i))^2$$

- In other words, the choice of λ depends on the unknown true regression curve and the inherent variability of the smoothing estimator
- We must estimate $L(\lambda)$ in order to get a data driven choice for λ

Cross-validation for choosing λ

- Cross-validation re-samples the original sample
- The data are split into k subsets and then our model is fit k times, each trying to predict using the left-out subset
- Prediction error for each subset is then calculated as the sum of squared errors:

$$RSS = \sum (Y_i - \hat{Y}_i)^2$$

- We do this with several possible models, choosing the one with the smallest average error (i.e., mean squared error)

$$MSE = \frac{\sum (Y_i - \hat{Y})^2}{n}$$

- Generalized Cross Validation (GCV) is the most commonly used method for choosing the smoothing parameter λ for smoothing spline models

Cross-validation for choosing λ (2)

- GCV uses n subsets of the data
- Each subset removes one observation from the dataset
  - That is, there is one subset corresponding to each observation that is removed
- The GCV criterion is then defined as:

$$GCV(\lambda) = \frac{\sum_{i=1}^{n} (y_i - \hat{f}_\lambda(x_i))^2}{\left(1 - n^{-1} tr(S)\right)^2}$$

- Simply put, GCV compares the fit of all models based on all possible values of λ, choosing the one that fits best
- GCV choice of λ is typically the default method in software programs, including the packages in **R**

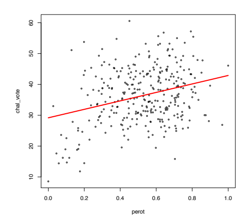
Degrees of Freedom

- As with lowess smoothing, the df for smoothing splines are an approximate generalization of the number of parameters in the parametric model
  - In exactly the same way, df for nonparametric spline models are obtained from the diagonal of the smoother matrix $\boldsymbol{S}$, which plays a similar role to the hat matrix $\boldsymbol{H}$ in linear regression, it transforms Y into $\hat{Y}$
  - The approximate or effective degrees of freedom are defined by: $df_\lambda = \text{trace}(\boldsymbol{S}_\lambda)$
  - The df_λ specifies, the approximate number of parameters used to fit the spline
- Using the effective degrees of freedom, we can carry out F-tests to compare different estimates and models applied to the same dataset, especially to compare the nonparametric smooth model to a linear model

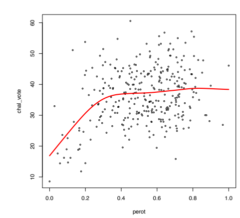
Smoothing Splines in **R**

- Two packages in **R** can be used to fit smoothing splines:
  - the `smoothing.spline` function in the `splines` package
  - the `sm.spline` function in the `pspline` package
- Since $df_\lambda = \text{trace}(\boldsymbol{S}_\lambda)$ we can either specify λ directly or invert the relationship and specify degrees of freedom instead
  - The latter method is much easier and somewhat more intuitive
  - By default, GCV is used by both the `smooth.spline` and `sm.spline` functions to choose λ
- Remember, like lowess models, this is a nonparametric model, so the effects must be graphed

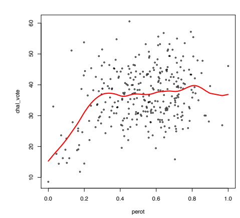
Comparing Degrees of Freedom



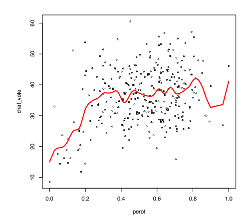
(e) df = 2



(f) df = 5



(g) df = 10



(h) df = 20

Additive Regression Models

- Additive regression models essentially apply local regression to low-dimensional projections of the data
  - That is, they estimate the regression surface by a combination of a collection of one-dimensional functions
- The nonparametric additive regression is:

$$Y = A + f_1(X_1) + f_2(X_2) + \cdots + f_k(X_k) + \varepsilon$$

  where the f_j are arbitrary functions estimated from the data, the ε are assumed to have constant variance and mean 0
- The estimated functions f_j are the analogues of the coefficients in linear regression

Additive Regression Models (2)

- The assumption that the contribution of each covariate is additive is analogous to the assumption in linear regression that each component is estimated separately
- Recall that the linear regression model is

$$Y = A + \sum_{j=1}^{k} B_j X_j + \varepsilon$$

  where B_j represent linear effects
- For the additive model, we model Y as an additive combination of arbitrary functions of the X's:

$$Y = A + \sum_{j=1}^{k} f_j(X_j) + \varepsilon$$

- The f_j represent arbitrary trends that can be estimated by lowess or smoothing splines

Additive Regression Models (3)

- Now comes the question: How do we find these arbitrary trends?
- If the X's are completely independent (which they won't be) we could just estimate each functional form using nonparametric regression of Y on each of the X's independently
  - Similarly in linear regression when the X's are completely uncorrelated, the partial regression slopes are identical to the marginal regression slopes
- Since the X's are related, however, we need to proceed in another way, in essence removing the effects of other predictors which are unknown before we begin
- We use a procedure called backfitting to find each curve, controlling for the effects of the others

Estimation and Backfitting

- Suppose that we have a two predictor additive model:

$$Y_i = \alpha + f_1(x_{i1}) + f_2(x_{i2}) + \varepsilon_i$$

- If we unrealistically know the partial regression function f_2, but not f_1, we could rearrange the equation in order to solve for f_1:

$$Y_i - f_2(x_{i2}) = \alpha + f_1(x_{i1}) + \varepsilon_i$$

- In other words, smoothing $Y_i - f_2(x_{i2})$ against x_{i1} produces an estimate of $\alpha + f_1(x_{i1})$
- Simply put, knowing one function allows us to find the other - in the real world, however, we don't know either so we must proceed initially with preliminary estimates

Estimation and Backfitting (2)

1. Start by expressing the variables in mean deviation form so that the partial regressions sum to zero, thus eliminating the individual intercepts
2. Take the preliminary estimates of each function from a least squares regression of Y on the X's:

$$\begin{aligned} y_i - \bar{y} &= b_1(x_{i1} - \bar{x}_1) + b_2(x_{i2} - \bar{x}_2) + \varepsilon_i \\ y_i^* &= b_1 x_{i1}^* + b_2 x_{i2}^* + \varepsilon_i \end{aligned}$$

3. The preliminary estimates are used as step (0) in an iterative estimation process

$$\begin{aligned} f_1^{(0)} &= b_1 x_{i1}^* \\ f_2^{(0)} &= b_2 x_{i2}^* \end{aligned}$$

4. Find the partial residuals for X_1 (recall the partial residuals remove Y from its linear relationship to X_2 while retaining the relationship with X_1

Estimation and Backfitting (3)

The partial residuals for X_1 are then

$$\begin{aligned} e_{i[1]}^{(1)} &= y_i^* - b_2(x_{i2}^*) \\ &= e_i + b_1 x_{i1}^* \end{aligned}$$

5. The same procedure in step 4 is done for x_2^*
6. Next, we smooth these partial residuals against their respective X's, providing a new estimate of f

$$\begin{aligned} f_k^{(1)} &= \text{smooth}\left[e_{i[k]}^{(1)} \text{ on } x_{ik}^*\right] \\ &= S_k\left\{Y_i - \left[f_1^{(1)}(x_{i1}^*) + f_2^{(1)}(x_{i2})^*\right]\right\} \end{aligned}$$

where $\boldsymbol{S}$ is the $n \times n$ smoother transformation matrix for X_j that depends only on the configuration of X_{ij} for the j^{th} predictor

Estimation and Backfitting (4)

- either loess or smoothing splines can be used to find the regression curves
- If local polynomial regression is used, a decision must be made about the span that is used
- If a smoothing spline is used, the degrees of freedom can be specified beforehand or using cross-validation with the goal of minimizing the penalized residual sum of squares

$$RSS(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int_{x_1}^{x_n} f''(x)^2 dx$$

- The first term measures the closeness to the data; the second term penalizes the curvature of the function

Estimation and Backfitting (5)

- The process of finding new estimates of the functions by smoothing the partial residuals is reiterated until the partial functions converge
  - That is, when the estimates of the smooth functions stabilize from one iteration to the next, we stop
- When this process is done, we obtain estimates of $s_j(X_{ij})$ for every value of X_j
- More importantly, we will have reduced a multiple regression to a series of two-dimensional partial regression problems, making interpretation easy:
  - Since each partial regression is only two-dimensional, the functional forms can be plotted in two dimensions showing the partial effect of each X_j on Y
  - In other words, perspective plots are not necessary, unless we include an interaction between two smoothed terms

Different Smoothers Available

- Thin plate regression splines: low-rank, isotonic smoother for any number of covariates. Bestfor single-term smooths and multiple term smooths where both terms are measured in the same units. Specify with `bs='tp'`. This is the default in `mgcv`.
- Thin plate spline + shrinkage: Penalized so the whole term *could* shrink to zero. Specify with `bs='ts'`.
- Cubic Regression Splines (like a b-spline), specify with `bs='cr'`.
- Cubic Regression Spline + shrinkage: A shrinkage version of cubic regression splines, addes an additional shrinkage penalty. Specify with `bs='cs'`.
- Tensor product smooth: Best for multiple-term smooths, particularly when the multiple terms are measured in different units. Specify with the `te()` function for a multiple-terms smooth or with `ti(x)`, `ti(y)` and `ti(x,y)` for a test of additivity. Can also specify any of the `bs=X` from above in the tensor product function.

Additive Regression Models in **R** : Canadian Prestige (2)

- The `summary` function returns tests for each smooth, the degrees of freedom for each smooth and an adjusted R^2 for the model
- The deviance can be obtained from the command `deviance(model)`

```
library(mgcv)
library(car)
data(Prestige)
prestige.gam <- gam(prestige ~ s(income) + s(education), data=Prestige)
summary(prestige.gam)


##
## Family: gaussian
## Link function: identity
##
## Formula:
## prestige ~ s(income) + s(education)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  46.8333     0.6889   67.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df     F  p-value
## s(income)    3.118  3.877 14.61 1.53e-09 ***
## s(education) 3.177  3.952 38.78  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.836   Deviance explained = 84.7%
```

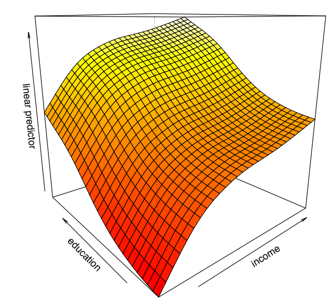

Additive Regression Models in **R** : Canadian Prestige (3)

- Again, as with other nonparametric models, we have no slope parameters to investigate (we do have an intercept, however)
- A plot of the regression surface is necessary

```
vis.gam(prestige.gam, theta=-40)
```

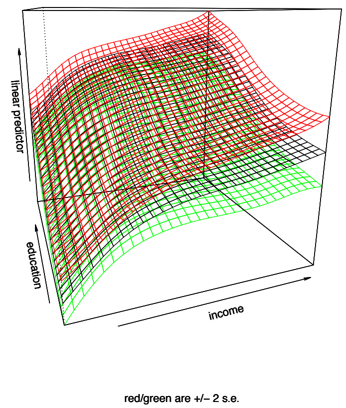

Additive Regression Models in **R** : Canadian Prestige (4)

- We can see the nonlinear relationship for both education and income with prestige, but there is no real interaction (i.e., the slope for income is the same at every value of education)



Additive Regression Models in **R** : Canadian Prestige (5)

The `vis.gam()` function can also make perspective plots with confidence regions if you use the `se=2` argument



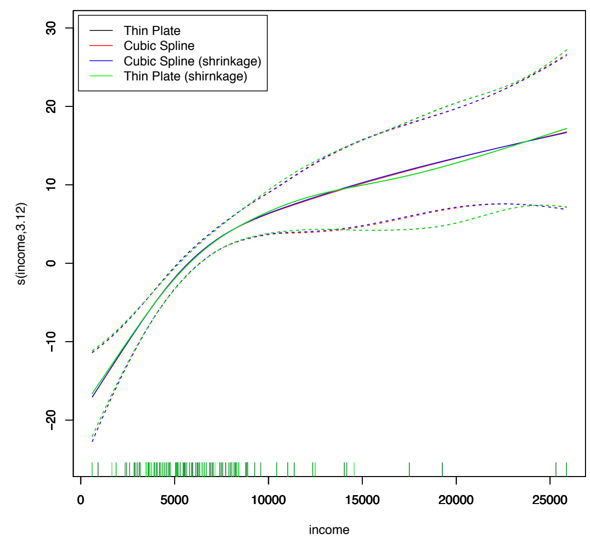
Additive Regression Models in **R** : Canadian Prestige (6)

- Since the slices of the additive regression in the direction of one predictor (holding the other constant) are parallel, we can graph each partial regression function separately
- This is the benefit of the additive model
  - We can graph as many plots as there are variables, and allowing us to easily visualize the relationships
- In other words, a multidimensional regression has been reduced to a series of two-dimensional partial-regression plots

Other Smoothers

```
gam1 <- gam(prestige ~ s(income, bs="tp") +
  s(education, bs="tp"), data=Prestige)
gam2 <- gam(prestige ~ s(income, bs="cr") +
  s(education, bs="cr"), data=Prestige)
gam3 <- gam(prestige ~ s(income, bs="cs") +
  s(education, bs="cs"), data=Prestige)
gam4 <- gam(prestige ~ s(income, bs="ts") +
  s(education, bs="ts"), data=Prestige)


plot(gam1, select=1, ylim=c(-25,30))
par(new=T)
plot(gam2, select=1, col="red",
  ylim=c(-25,30), xlab="", ylab="")
par(new=T)
plot(gam3, select=1, col="blue",
  ylim=c(-25,30), xlab="", ylab="")
par(new=T)
plot(gam4, select=1, col="green",
  ylim=c(-25,30), xlab="", ylab="")
legend("topleft", c("Thin Plate", "Cubic Spline",
  "Cubic Spline (shrinkage)", "Thin Plate (shirnkage)"),
  lty=c(1,1,1,1), col=c("black", "red", "blue", "green"), inset=.01)
```

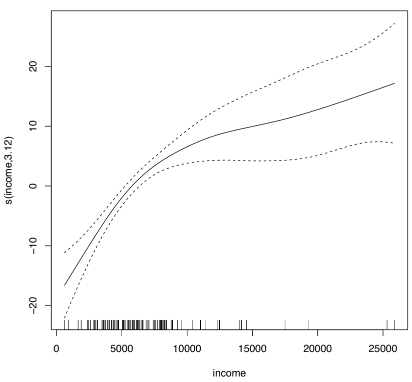

Plot

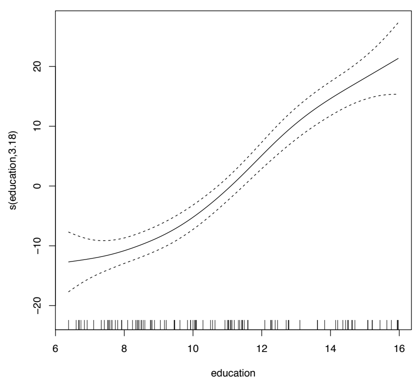


Effect of Education and Income

```r
plot(prestige.gam, select=1)
```

```r
plot(prestige.gam, select=2)
```

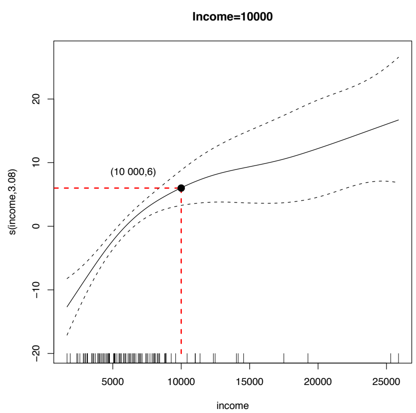


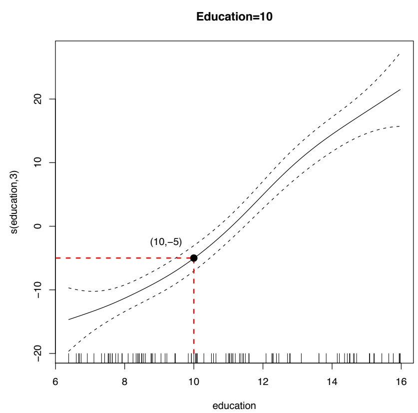


Interpreting the Effects

- A plot of X_j versus $s_j(X_j)$ shows the relationship between X_j and Y holding constant the other variables in the model
- Since Y is expressed in mean deviation form, the smooth term $s_j(X_j)$ is also centered and thus each plot represents how Y changes relative to its mean with changes in X
  - The value of 0 on the Y-axis is the mean of Y
  - As the line moves away from 0 in a negative direction we subtract the distance from the mean when determining the fitted value. For example, if the mean is 45 and for a particular X-value (say x=15) the curve is at $s_j(X_j) = 4$, this means the fitted value of Y controlling for all other explanatory variables is 45+4=9
  - If there are several nonparametric relationships, we can add together the effects on the two graphs for any particular observation to find its fitted value of Y

Interpreting the Effects (2)





- The mean of prestige is 47.3. Therefore, the fitted value for an occupation with average income of \$10000/year and 10 years of education is on average: 47.3+6-5=48.3

Residual Sum of Squares

- As was the case for smoothing splines and lowess smooths, statistical inference and hypothesis testing is based on the residual sum of squares (or deviance in the case of generalized additive models) and the degrees of freedom
- the RSS for an additive model is easily defined in the usual manner:

$$\text{RSS} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- the approximate degrees of freedom, however, need to be adjusted from the regular nonparametric case, however, because we are no longer specifying a jointly-conditional functional form

Degrees of Freedom

- Recall that for nonparametric regression, the approximate df are equal to the trace of the smoother matrix (the matrix that projects Y on to $\hat{Y}$
- We extend this to additive models:

$$df_j = \text{trace}(\boldsymbol{S}) - 1$$

  1 is subtracted from each df reflecting the constant that each partial regression function sums to zero (the individual intercepts have been removed)
- Parametric terms entered in the model each occupy a single degree of freedom as in the linear regression case
- the individual degrees of freedom are then combined for a single measure:

$$df_{mod} = \sum_{j=1}^{k} df_j + 1$$

  1 is added to the final degrees of freedom to account for the overall constant in the model

Specifying Degrees of Freedom

- We can set either the degrees of freedom or the smoothing parameter λ
- Also, like with smoothing splines, generalized cross-validation can be used to specify degrees of freedom
  - Recall that this finds the smoothing parameter that gives the lowest average MSE from the cross-validation samples
- Cross-validation is implemented using the `mgcv` package in **R**

```
> Prestige.gam2 <- gam(prestige ~ te(income, k=7,
    fx=TRUE) + te(education), data=Prestige)
```

- We specify the number of degrees of freedom with `k` and specify `fx=TRUE` else GCV will be used

Cautions about Statistical tests when λ is chosen using GCV

- If the smoothing parameters λ's (or equivalently, the degrees of freedom) are chosen using GCV, caution must be used when employing analysis of deviance
- If a variable is added or removed from the model, the smoothing parameter λ that yields the smallest MSE will likely also change
  - By implication, the degrees of freedom also change implying that the equivalent number of parameters used for the model is different
  - In other words, the test will only be approximate because the otherwise nested models have different degrees of freedom associated with λ
- As a result, it is advisable to fix the degrees of freedom when computing models

Testing for Linearity

- We can compare the linear model of prestige regressed on income and education with the additive model by carrying out an incremental F-test

```
prestige.ols<-gam(prestige~income+education, data=Prestige)
anova(prestige.ols, prestige.gam, test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: prestige ~ income + education
## Model 2: prestige ~ s(income) + s(education)
##   Resid. Df Resid. Dev     Df Deviance      F    Pr(>F)
## 1    99.000     6038.9
## 2    93.171     4585.0 5.8293   1453.9 5.1516 0.0001513 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The difference between the models is statistically significant - the additive model describes the relationship between prestige and education and income much better

Testing for Additivity of Smooth

- We can test for additivity of a smooth by using the `ti` function to make the smooths:

```
prestige.add <- gam(prestige ~ ti(income) + ti(education), data=Prestige)
prestige.mult <- gam(prestige ~ ti(income) + ti(education) +
    ti(income, education), data=Prestige)
anova(prestige.add, prestige.mult, test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: prestige ~ ti(income) + ti(education)
## Model 2: prestige ~ ti(income) + ti(education) + ti(income, education)
##   Resid. Df Resid. Dev     Df Deviance      F  Pr(>F)
## 1    94.324     4569.4
## 2    92.994     4369.3 1.3302   200.16 3.2446 0.06277 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


Back to the GDP data

```
dat <- read.dta(
   "http://www.quantoid.net/files/reg3/gdp_data_2000.dta")
rawlm <- lm(log(rgdpna_pc) ~ polity2 + primsch_enroll_pc +
   pop_c100k_pc, data=dat)
summary(rawlm)
```

```
##
## Call:
## lm(formula = log(rgdpna_pc) ~ polity2 + primsch_enroll_pc + pop_c100k_pc,
##     data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.66283 -0.62070  0.03893  0.58882  2.98972
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        9.0550272  0.2970106  30.487  < 2e-16 ***
## polity2            0.0415854  0.0142745   2.913  0.00416 **
## primsch_enroll_pc -0.0004229  0.0000958  -4.414 2.01e-05 ***
## pop_c100k_pc       0.0021669  0.0004590   4.721 5.64e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.06 on 140 degrees of freedom
## Multiple R-squared:  0.3198,Adjusted R-squared:  0.3052
## F-statistic: 21.94 on 3 and 140 DF,  p-value: 1.054e-11
```


GAM of GDP

```
library(mgcv)
gam.gdp <- gam(log(rgdpna_pc) ~ s(polity2, bs="cs") + primsch_enroll_pc +
   s(pop_c100k_pc, bs="cs"), data=dat)
summary(gam.gdp)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(rgdpna_pc) ~ s(polity2, bs = "cs") + primsch_enroll_pc +
##     s(pop_c100k_pc, bs = "cs")
##
## Parametric coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        9.345e+00  1.913e-01   48.85  < 2e-16 ***
## primsch_enroll_pc -2.121e-04  7.657e-05   -2.77  0.00642 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F  p-value
## s(polity2)      5.506      9 14.244  < 2e-16 ***
## s(pop_c100k_pc) 6.226      9  4.825 3.31e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.635   Deviance explained = 66.7%
## GCV = 0.65305  Scale est. = 0.59078   n = 144
```


Test of models

```
anova(rawlm, gam.gdp)
```

```
## Analysis of Variance Table
##
## Model 1: log(rgdpna_pc) ~ polity2 + primsch_enroll_pc + pop_c100k_pc
## Model 2: log(rgdpna_pc) ~ s(polity2, bs = "cs") + primsch_enroll_pc +
##     s(pop_c100k_pc, bs = "cs")
##   Res.Df    RSS     Df Sum of Sq      F  Pr(>F)
## 1 140.00 157.25
## 2 130.27  76.96 9.7313    80.293 13.966 2.9e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Testing Urban Population

```
library(splines)
lm.mod <- lm(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   bs(pop_c100k_pc, df=8), data=dat)
gam.mod <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(pop_c100k_pc, bs="cs", k=9), data=dat)
anova(lm.mod, gam.mod)
```

```
## Analysis of Variance Table
##
## Model 1: log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc + bs(pop_c100k_pc,
##     df = 8)
## Model 2: log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc + s(pop_c100k_pc,
##     bs = "cs", k = 9)
##   Res.Df    RSS      Df Sum of Sq      F  Pr(>F)
## 1 132.00 78.409
## 2 133.67 82.811 -1.6664   -4.4025 4.4475 0.01882 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


Testing Polity2

```
gam.mod2 <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(pop_c100k_pc, k=9, fx=T), data=dat)
gam.mod2a <- gam(log(rgdpna_pc) ~ s(polity2, bs="cs") + primsch_enroll_pc +
   s(pop_c100k_pc, k=9, fx=T), data=dat)
anova(gam.mod2, gam.mod2a, test='F')
```

```
## Analysis of Deviance Table
##
## Model 1: log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc + s(pop_c100k_pc,
##     k = 9, fx = T)
## Model 2: log(rgdpna_pc) ~ s(polity2, bs = "cs") + primsch_enroll_pc +
##     s(pop_c100k_pc, k = 9, fx = T)
##   Resid. Df Resid. Dev     Df Deviance      F Pr(>F)
## 1    132.00     80.942
## 2    127.29     75.960 4.7136   4.9825 1.7858 0.1245
```


Problems with Data Sparsity

The urban population variable exhibits a couple of different problems with sparsity in the data. First, there are 8 countries with values of 0, then the next smallest value is 31. There are also 6 values tha range from 726-996.

- Splines (and any flexible function) still have a chance to overfit data in sparse regions.
- Normalizing transformations could help pull in big outliers on the high end.

Let's try out a couple of other things with the population variable.

Different Spline Functions

```
gam.pop1 <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(pop_c100k_pc, k=9, bs="cs"), data=dat)
gam.pop2 <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(pop_c100k_pc, k=9, bs="ts"), data=dat)
```

Now, transforming urban population with a Y-J normalization

```
pt1 <- powerTransform(pop_c100k_pc ~ 1, data=dat, family="yjPower")
gam.pop1a <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(VGAM:::yeo.johnson(pop_c100k_pc, 0.5), k=9, bs="cs"), data=dat)
gam.pop2a <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(VGAM:::yeo.johnson(pop_c100k_pc, 0.5), k=9, bs="ts"), data=dat)
```

Finally, transforming urban population with a BCn normalizatiopn

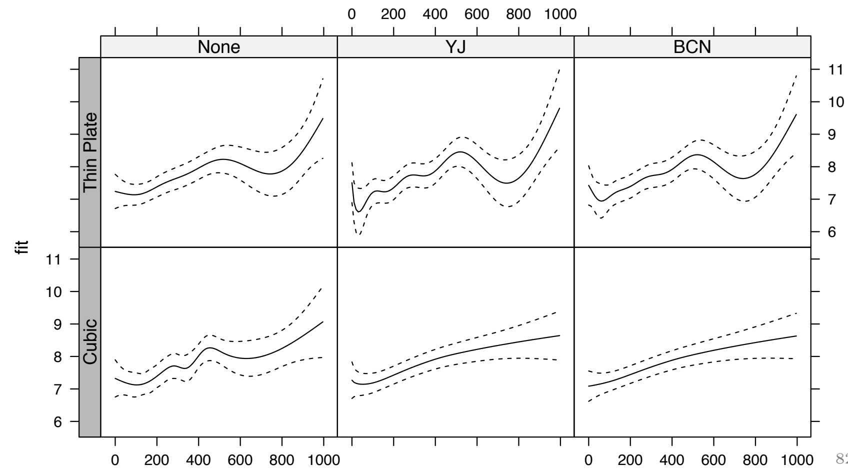
```
pt2 <- powerTransform(pop_c100k_pc ~ 1, data=dat, family="bcnPower")
gam.pop1b <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(car:::bcnPower(pop_c100k_pc, lambda=0.5, gamma=49.8),
      k=9, bs="cs"), data=dat)
gam.pop2b <- gam(log(rgdpna_pc) ~ poly(polity2, 2) + primsch_enroll_pc +
   s(car:::bcnPower(pop_c100k_pc, lambda=0.5, gamma=49.8),
      k=9, bs="ts"), data=dat)
```

Plots of Different Smoothers

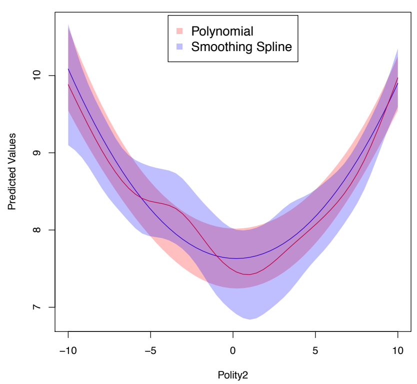
```r
newdat <- data.frame(
  pop_c100k_pc = seq(0,996, length=100),
  primsch_enroll_pc = mean(dat$primsch_enroll_pc, na.rm=T),
  polity2 = 0)
pred1 <- predict(gam.pop1, newdata=newdat, se.fit=T)
pred1a <- predict(gam.pop1a, newdata=newdat, se.fit=T)
pred1b <- predict(gam.pop1b, newdata=newdat, se.fit=T)
pred2 <- predict(gam.pop2, newdata=newdat, se.fit=T)
pred2a <- predict(gam.pop2a, newdata=newdat, se.fit=T)
pred2b <- predict(gam.pop2b, newdata=newdat, se.fit=T)

plot.dat <- data.frame(
  urbanpop = rep(newdat$pop_c100k_pc, 6),
  fit = c(pred1$fit, pred1a$fit, pred1b$fit,
          pred2$fit, pred2a$fit, pred2b$fit),
  se = c(pred1$se.fit, pred1a$se.fit, pred1b$se.fit,
          pred2$se.fit, pred2a$se.fit, pred2b$se.fit),
  spline = as.factor(rep(c("Cubic", "Thin Plate"), each=300)),
  transform = factor(rep(c("None", "YJ", "BCN"), each=100),
          levels=c("None", "YJ", "BCN"))
)
plot.dat$lower <- plot.dat$fit - 1.96*plot.dat$se
plot.dat$upper <- plot.dat$fit + 1.96*plot.dat$se
```

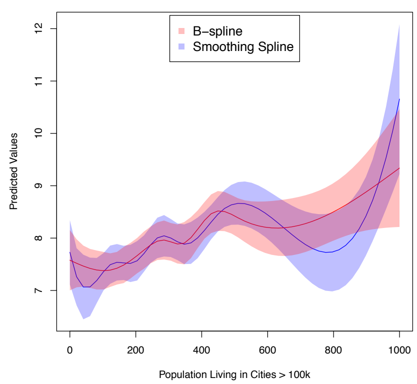

Plots of Different Smoothers

```r
library(DAMisc)
library(latticeExtra)
useOuterStrips(xyplot(fit ~ urbanpop | transform + spline, data=plot.dat,
  panel=panel.ci, prepanel=prepanel.ci, zl=0, lower=plot.dat$lower,
  upper=plot.dat$upper))
```



Plots of Predictions





Conclusions

- Splines (smoothing and otherwise) can be good at finding arbitrary structure in data.
- Smoothing splines (and by extension, GAMs) prevent overfitting through “regularization”.
- Interpreting spline models must be done graphically.