36-617: Applied Linear Models

Nonparametric Regression II

Brian Junker
132E Baker Hall
brian@stat.cmu.edu

Announcements

- Work that is due:
  - HW05 due tonight at 1159pm
- Midterm grades:
  - HW01-04, Quiz01-05, Participation, Take-home Midterm
- Over the fall break:
  - No hw assigned or due
  - No quiz on Mon after break
- Reading:
  - This week: Sheather Appx; ISLR Ch 7
  - Next week: Gelman & Hill Ch 9 & 10 (copies in week07 folder)
    - (read to get a sense rather than to get every detail)

Outline

- Other nonparametric linear smoothing methods
  - Local Regression Smoothers
    - loess()
    - Kernel Smoothed Regression
- A Comparison of Linear Smoothers
- Generalized Additive Models (GAMs)
  - Nonparametric transformations for the Heights data
  - Nonparametric transformations for the Wells data
- And beyond… (nonparametric interactions)
- Advice on nonparametric regression

Local Regression Smoothers (e.g. loess())

- For each pair (x, y) for which we desire $\hat{y}$,
  - Build a regression using the $k = \lceil sn \rceil$ nearest neighbors to x in the data, for some fixed fraction s
  - Downweight neighbors farther from x.
- Slightly more formally (local linear regression):
  - Given data $(x_1, y_1), \ldots, (x_n, y_n)$ and a point x,
  - Obtain $\widehat{\beta_{0x}}, \widehat{\beta_{1x}}$ by minimizing the local RSS

$$RSS_x = \sum_{i=1}^{n} K(x_i, x)(y_i - \beta_{0x} - \beta_{1x} x_i)^2$$

    where $K(x_i, x)$ selects & downweights k nearest nbrs
  - Produce $\hat{y} = \widehat{\beta_{0x}} + \widehat{\beta_{1x}} x$

Local polynomial regression

- We could replace

$$RSS_x = \sum_{i=1}^{n} K(x_i, x)(y_i - \beta_{0x} - \beta_{1x} x_i)^2$$

with an RSS using any polynomial of degree p:

$$RSS_x^p = \sum_{i=1}^{n} K(x_i, x)\left(y_i - (\beta_{0x} + \beta_{1x} x_i + \cdots + \beta_{px} x_i^p)\right)^2$$

- In practice, one only sees
  - p=0 (locally weighted moving average)
  - p=1 (locally weighted linear regression)
  - p=2 (locally weighted quadratic regression)

Choices to make in local regression

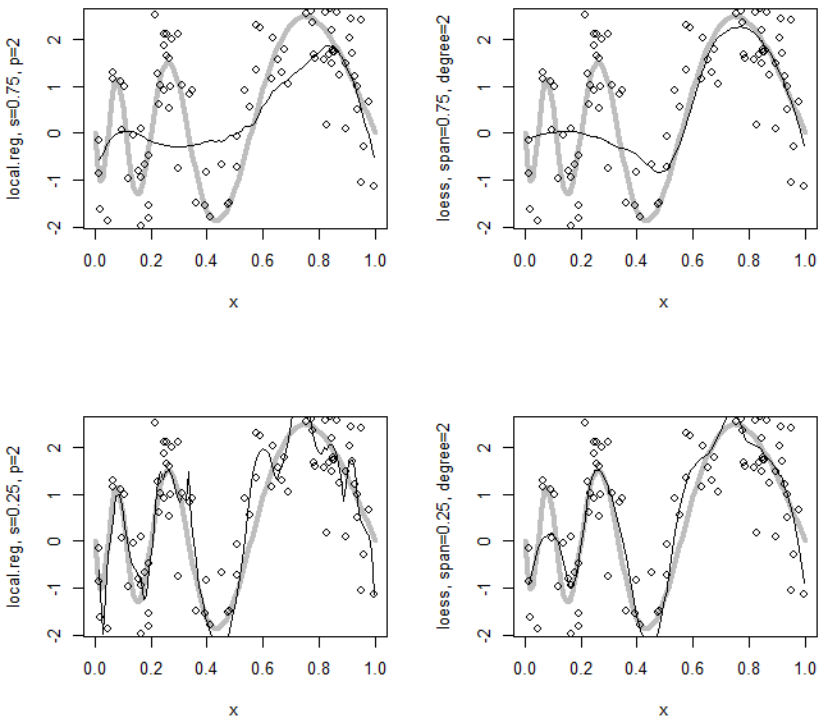
- The degree p of the polynomial
  - loess() defaults to $p = 2$, but any of $p = 0,1,2$ work
  - $p = 0$ is essentially Kernel Smoothing (see later in slides)
- The Kernel function $K(x_i, x)$
  - Traditional loess() choice is $K(x_i, x) = \frac{1}{h} w\left(\frac{x_i - x}{h}\right)$, where the *bandwidth* $h = s \cdot range(x_i \in \lceil ns \rceil$ nearest neighbors) and

$$w(t) = \begin{cases} \left(1 - |x|^3\right)^3, |x| < 1 \\ 0, |x| \geq 1 \end{cases}$$ (The *tricube* kernel)

  - …or any $w(t)$ that is symmetric with a tallest central mode
- The fraction s of data to consider;
  - s=0.75 is loess() default.

Trying local regression on our example…

```
> local.reg <- function(xi,yi,s=0.75, ntest=100) {
    xtest <- seq(min(xi),max(xi),length=ntest)
    model <- formula(yi ~ xi + I(xi^2))
    w <- function(x,h) {
       x <- x/h; result <- ifelse(abs(x)>=1,0,(1-abs(x)^3)^3)/h
       return(result)
    }
    s <- min(s,1); k <- ceiling(s*length(xi))
    yhat <- NULL
    for(x in xtest) {
       deltas <- abs(xi-x);  d.ord <- order(deltas)[1:k]
       loc <- data.frame(xi=xi[d.ord],yi=yi[d.ord])
       h <- s*(max(loc$xi) - min(loc$xi))
       lmfit <- lm(model,weights=w(deltas[d.ord],h),data=loc)
       yhat <- c(yhat, predict(lmfit,data.frame(xi=x)))
    }
    return(list(x=xtest,y=yhat))
}
> par(mfrow=c(2,2))
> setup("local.reg, s=0.75, p=2"); lines(local.reg(x,y,s=0.75))
> setup("loess, span=0.75, degree=2")
> lines(x,predict(loess(y ~ x,span=0.75,degree=2)))
> setup("local.reg, s=0.25, p=2"); lines(local.reg(x,y,s=0.25))
> setup("loess, span=0.25, degree=2")
> lines(x,predict(loess(y ~ x,span=0.25,degree=2)))
```



- local.reg() and loess() don't perfectly match (loess documentation incomplete?)
- It seems the definition of "nearest neighbors" is wider for loess()
- But their qualitative performance is similar

Computational details in the r file accompanying this lecture

Kernel smoothed regression

- Consider local regression with $p = 0$, and always using all the data (instead of $k = sn$ nearest nbrs)

$$RSS_x = \sum_{i=1}^{n} K(x_i, x)(y_i - \beta_{0x})^2 = \sum_{i=1}^{n} \frac{1}{h} w\left(\frac{x_i - x}{h}\right)(y_i - \beta_{0x})^2$$

- Setting derivative with respect to β_0 equal to 0,

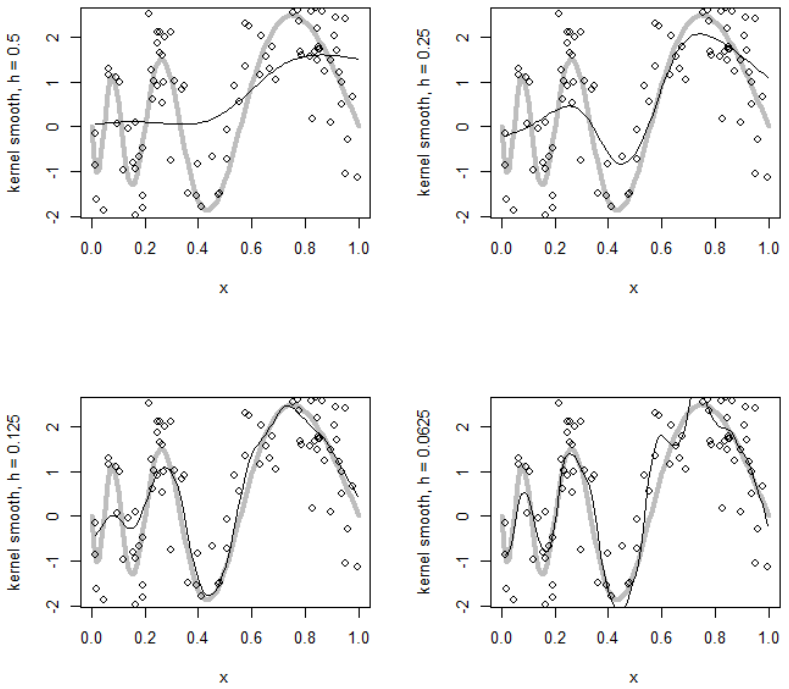
$$-2 \sum_{i=1}^{n} \frac{1}{h} w\left(\frac{x_i - x}{h}\right)(y_i - \beta_{0x}) = 0$$

$$\hat{y} = \widehat{\beta_{0x}} = \frac{\sum_{i=1}^{n} y_i \, w\left(\frac{x_i - x}{h}\right)}{\sum_{i=1}^{n} w\left(\frac{x_i - x}{h}\right)}$$

- *This yields the Nadaraya-Watson kernel smoothed regression*; implemented as ksmooth() in base R

Trying kernel smoothing on our example…

- Here we just demonstrate ksmooth()
- ksmooth() does not implement the tricube kernel; we will use a normal density kernel instead




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

> setup("kernel smooth, h = 0.5")
> lines(ksmooth(x,y,kernel="normal",bandwidth=0.5))

> setup("kernel smooth, h = 0.25")
> lines(ksmooth(x,y,kernel="normal",bandwidth=0.25))

> setup("kernel smooth, h = 0.125")
> lines(ksmooth(x,y,kernel="normal",bandwidth=0.125))

> setup("kernel smooth, h = 0.0625")
> lines(ksmooth(x,y,kernel="normal",bandwidth=0.0625))
```

Is local regression a linear smoother?

- If we apply the N-W Kernel smooth to the original data $(x_1, y_1), \ldots, (x_n, y_n)$, we get

$$\widehat{y_i} = \frac{\sum_{j=1}^{n} y_j \, w\left(\frac{x_j - x_i}{h}\right)}{\sum_{j=1}^{n} w\left(\frac{x_j - x_i}{h}\right)} = \sum_{j=1}^{n} \left(\frac{w\left(\frac{x_j - x_i}{h}\right)}{\sum_{k=1}^{n} w\left(\frac{x_k - x_i}{h}\right)} \right) y_j$$

  So N-W kernel smooth is linear, $\hat{y} = Hy$, where

$$H_{ij} = \frac{w\left(\frac{x_j - x_i}{h}\right)}{\sum_{k=1}^{n} w\left(\frac{x_k - x_i}{h}\right)}$$

- For general local regression, we can get similar expressions for each $\widehat{y_i}$, but h also depends on i.
  - *Local regression and Kernel smoothing **<u>are</u>** linear!*
  - As h (or s) decreases, the effective $df = tr(H)$ goes up

A comparison of linear smoothers

	Regression Spline	Natural Spline	Smoothing Spline	Local Regression	loess	Kernel Regression
Arbitrary or Conventional/Default Choice						
Polynomial Degree	3	3	3	0, 1, or 2	0, 1, or 2	0
Kernel function	- -	- -	- -	$K(x_i, x)$	based on tricube	$w(t)$
Knot locations*	$t_1, \ldots, t_m$	$t_1, \ldots, t_m$	$t_1, \ldots, t_m$	- -	- -	- -
Can be Chosen Algorithmically (LOOCV, etc.)						
Knot count	m	m	m	- -	- -	- -
Smoothing parameter	- -	- -	λ	s (sample fraction)	s (sample fraction)	h (bandwidth)
Features of the Fit						
Form of H	$Z(Z^TZ)^{-1}Z^T$	$Z(Z^TZ)^{-1}Z^T$	$G(G^TG+\lambda M)^{-1}G^T$	iterative calculation	iterative calculation	iterative calculation
df	$tr(H) = m+4$	$tr(H) = m+2$	$2 \leq tr(H_\lambda) \leq m+2$	$tr(H_s)$	$tr(H_s)$	$tr(H_h)$
Data subsets	1 (all data)	1 (all data)	1 (all data)	many (one per test x)	many (one per test x)	1 (all data)
model fits	1	1	1	many (one per test x)	many (one per test x)	many (one per test x)
Fitting method	OLS	OLS	Ridge OLS	iterative calculation	iterative calculation	iterative calculation

- First three columns: Spline-based regression
  - Relatively simple specification (generalizes ordinary regression methods—not so intuitive?)
  - Makes efficient use of the data
  - Easy to incorporate into other linear regression models
- Last three columns: Local regression
  - More complex specification (generalizes moving averages—more intuitive?)
  - Makes intensive/nonefficient use of the data
  - May require *backfitting* or similar to incorporate into other lin. reg. models

*Knot location is somewhat of a dark art, but good defaults are either (a) uniformly across the range of x; or (b) uniform quantiles of x.

Using smoothers to transform variables in regression: ***Generalized Additive Models***

- Regression splines and natural splines (without penalties) just generate columns of the X matrix
- Can incorporate directly into lm() fits – we only need ordinary least-squares to fit these models.
- (Generalized) linear models that have one or more predictors transformed by a nonparametric smooth are called *(generalized) additive models,* or *GAMs*.
- Good places online to start to learn about GAMs:
  - https://www.mainard.co.uk/post/why-mgcv-is-awesome/
  - https://multithreaded.stitchfix.com/blog/2015/07/30/gam/

Both blog posts have links to other really useful resources.

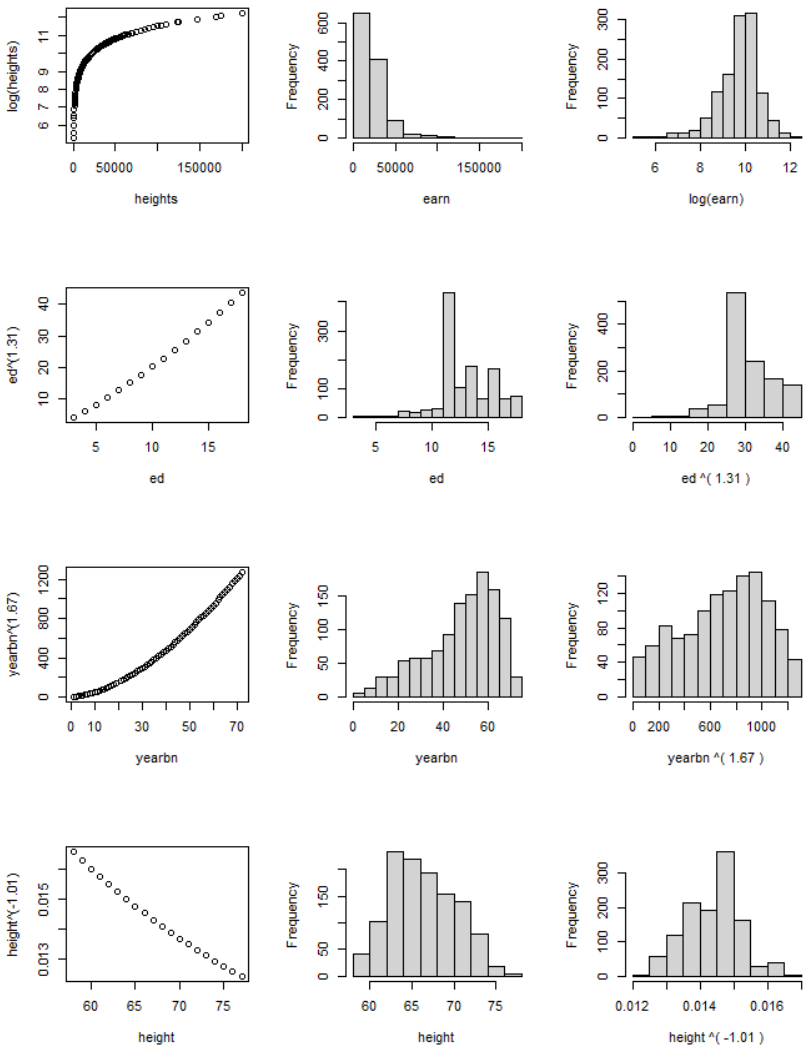
Example #1: The Heights data

- Recall the “heights.dta” data, in which we wanted to predict earnings from several demographic variables:

```
> str(heights)
'data.frame':   1186 obs. of  7 variables:
 $ earn  : num  50000 60000 30000 51000 9000 29000 32000 2000
27000 6530 ...
 $ sex   : num  0 = Female, 1 = Male
 $ race  : Factor w/ 4 levels "1","2","3","4"
 $ hisp  : num  0 = not Hispanic, 1 = hispanic
 $ ed    : num  Number of years of education
 $ yearbn: num  e.g. 45 means 1945, 32 means 1932, etc.
 $ height: num  Height, in inches
```

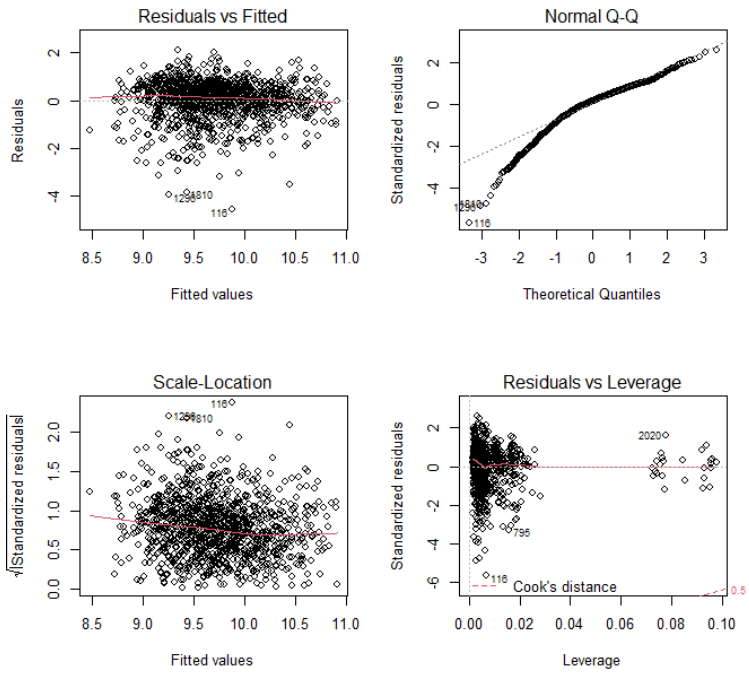
Transforming the variables

- log(earn)
- Box-Cox transforms:
  - $(ed)^{1.31}$
  - $(yearbn)^{1.67}$
  - $(height)^{-1.01}$
- These transforms are intended to *improve normality* of the variables, *not their predictive power*.



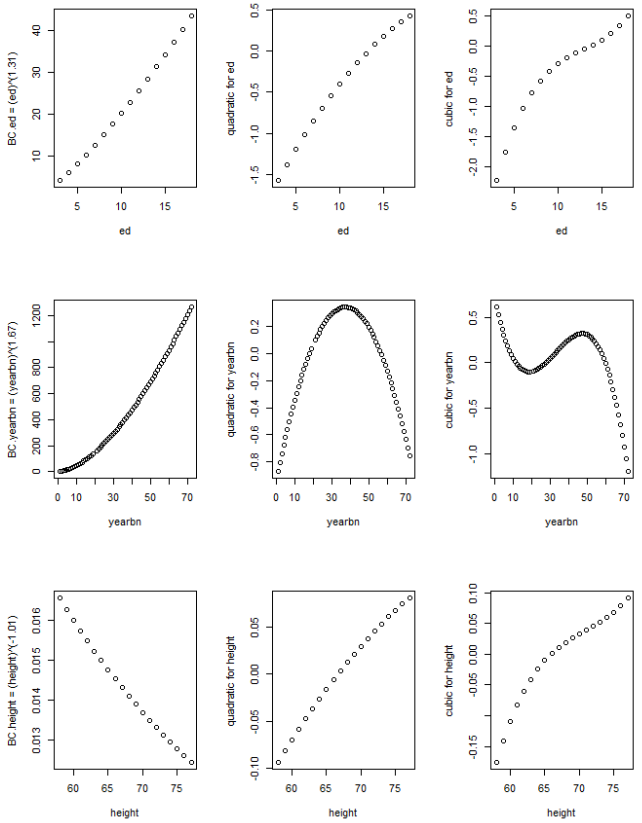
Results for Box-Cox transforms…

```
> lm.0 <- lm(log(earn) ~ sex + race + hisp +
+ I(ed^1.31) + I(yearbn^1.67) + I(height^-1.01),
+ data=heights)
> round(summary(lm.0)$coef,2)
                Est  S.E.   tval pval
(Int)         10.05  0.61 16.61 0.00
sex           -0.45  0.07 -6.72 0.00
race2         -0.04  0.08 -0.45 0.65
race3          0.15  0.22  0.68 0.50
race4         -0.37  0.25 -1.52 0.13
hisp           0.14  0.10  1.37 0.17
ed^1.31        0.04  0.00 12.26 0.00
ybn^1.67       0.00  0.00 -7.66 0.00
hgt^-1.01    -75.50 40.84 -1.85 0.06
> par(mfrow=c(2,2))
> plot(lm.0)
```



What transformations would we get from polynomial regression?

```
> lm.poly2 <- lm(log(earn) ~ sex +
+ race + hisp +
+ poly(ed,2) +
+ poly(yearbn,2) +
+ poly(height,2),
+ data=heights)
> lm.poly3 <- lm(log(earn) ~ sex +
+ race + hisp +
+ poly(ed,3) +
+ poly(yearbn,3) +
+ poly(height,3),
+ data=heights)
>
> ## On the right we compare the Box-Cox
> ## transforms with these 2 and 3
> ## degree polynomial tranforms
> ## (the code to do this is in
> ## the R file for this lecture)
```



Digression: Partial Residual Plots

- Recall our discussion of *added variable plots*
  - The terms in a regression do not directly predict y
  - They predict the residual after regressing y on all the other terms – these are called *partial residuals*
- In the fitted model $\hat{y} = \widehat{\beta_0} + \widehat{\beta_1}X_1 + \cdots + \widehat{\beta_p}X_p$, $\widehat{\beta_j}$ is the coefficient on X_j when we regress

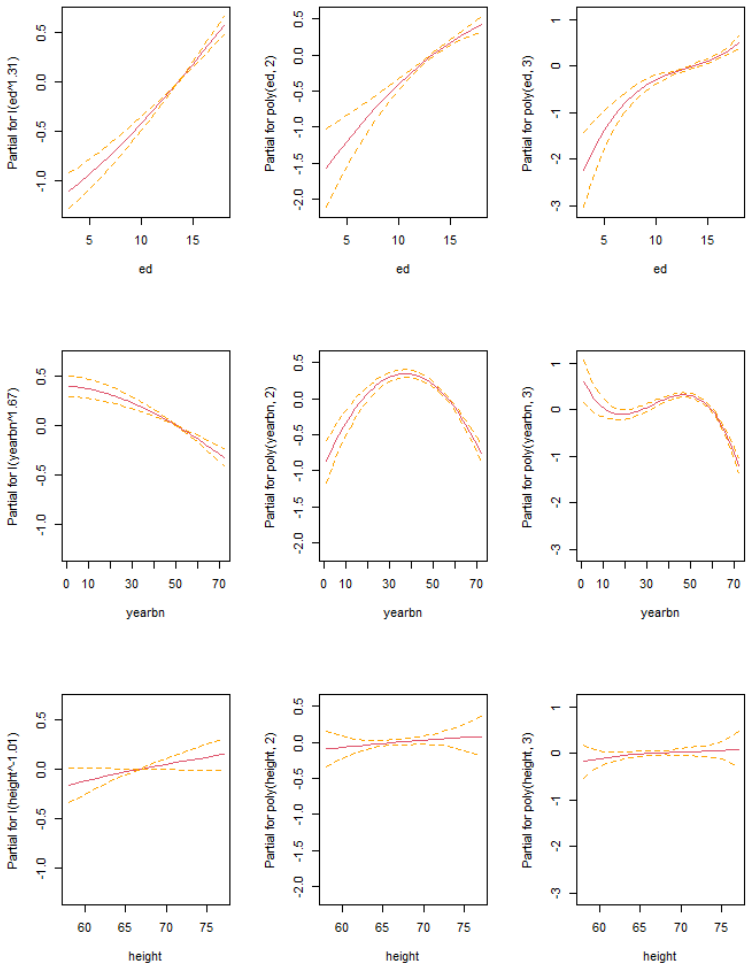
  $r_j = y - (\widehat{\beta_0} + \widehat{\beta_1}X_1 + \cdots \widehat{\beta_{j-1}}X_{j-1} + \widehat{\beta_{j+1}}X_{j+1} + \cdots + \widehat{\beta_p}X_p)$

  on X_j: $r_j = (const) + \beta_j X_j + \varepsilon$.
- Plotting a transform against its partial residual tells us what it is actually predicting in the model!

Comparing Partial Residual Plots…

```
par(mfcol=c(3,3))
termplot(lm.0,se=T,terms=4:6)
termplot(lm.poly2,se=T,terms=4:6)
termplot(lm.poly3,se=T,terms=4:6)
```

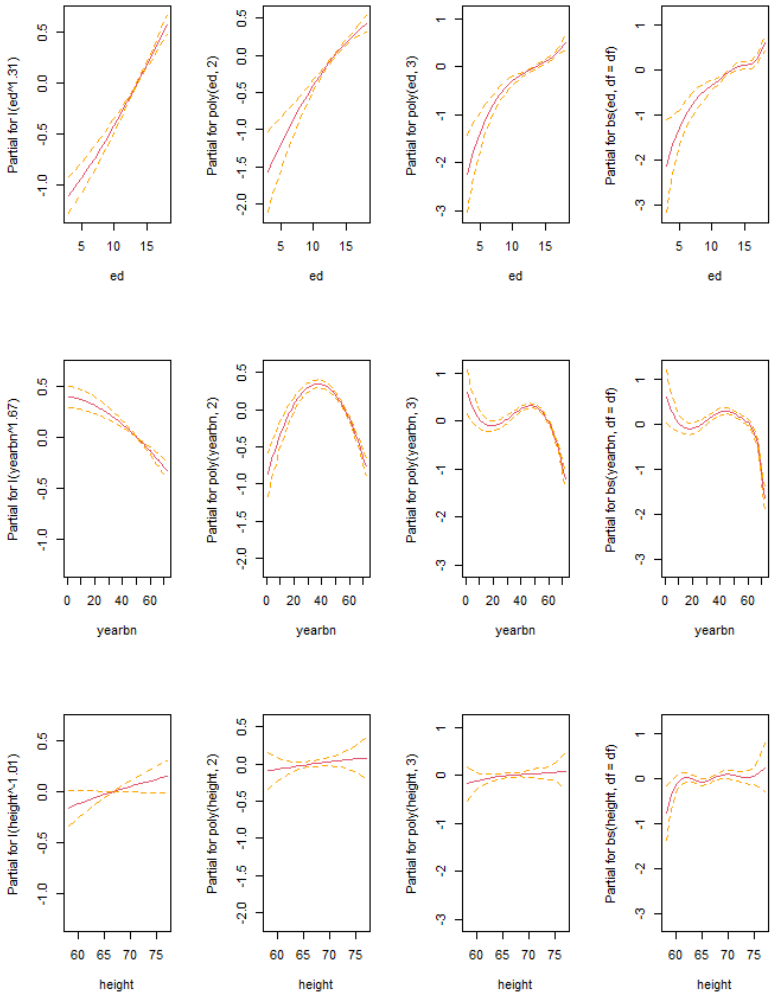
- Box-Cox vs polynomials:
  - Agree(?) on ed, height
  - Disagree on yearbn!
- The part of log(earn) not explained by the other variables:
  - BC: Goes up with age
  - Poly2: Goes up then down
  - Poly3: up, plateau, up more?



Which is tracking the data better?

```
> library(splines)
> df <- 5
> lm.bs <- lm(log(earn) ~ sex +
+ race + hisp +
+ bs(ed,df=df) +
+ bs(yearbn,df=df) +
+ bs(height,df=df),
+ data=heights)
> par(mfcol=c(3,4))
> termplot(lm.0,se=T,terms=4:6)
> termplot(lm.poly2,se=T,terms=4:6)
> termplot(lm.poly3,se=T,terms=4:6)
> termplot(lm.bs,se=T,terms=4:6)
```

- The bspline with 5 df requested seems to confirm the cubic fit
- We only guessed the df; we will use smoothing splines to check.



Adding smoothing splines to a linear model

- Smoothing splines no longer simply add columns to the X matrix. *They also shrink the coefficients*.
- We can no longer fit using simple least-squares.
- A *Backfitting* algorithm is used instead:
  - Choose starting values $\widehat{\beta_1}, \ldots, \widehat{\beta_p}$
  - Regress partial residuals r_1 onto X_1 to update $\widehat{\beta_1}$
  - …
  - Regress partial residuals r_p onto X_p to update $\widehat{\beta_p}$
  - Repeat updates until no more changes.

R packages that use backfitting to fit GAMs

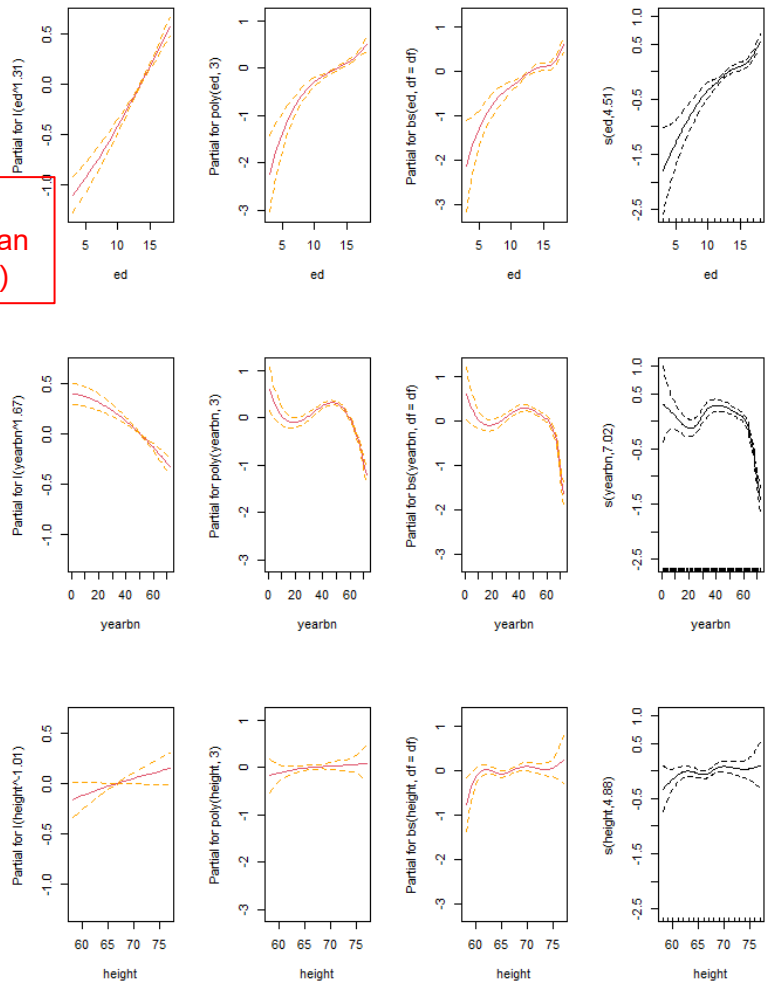
- library(gam)
  - Built & maintained by Trevor Hastie, since ca. 1990
  - Uses smooth.spline and loess as smoothers
  - Has methods for stepwise and for imputing NA's
- library(mgcv)
  - Built & maintained by Simon Wood, since mid-2000's
  - Can use cubic smoothing splines (like smooth.spline) but is built around "thin-plate splines".
  - Modern methods for choosing λ, for assessing model fit, for nonparametric interactions, and for big data sets.

These libraries are ***very incompatible***. To switch from one to the other, you have to quit and restart R.

Comparing our models with smooth spline model

```
> library(mgcv)
> gam.ss <- gam(log(earn) ~ sex +
+ race + hisp +
+ s(ed,bs="cr") +
+ s(yearbn,bs="cr") +
+ s(height,bs="cr"),
+ data=heights,method="GCV.Cp")
> par(mfcol=c(3,4))
> termplot(lm.0,se=T,terms=4:6)
> termplot(lm.poly3,se=T,terms=4:6)
> termplot(lm.bs,se=T,terms=4:6)
> plot(gam.ss) # mgcv's "termplot"
```

Use cubic regression splines ("cr") rather than thin-plate splines ("tp")



- bs="cr" chooses cubic regression splines.
- By default, s() chooses an optimal lambda for smoothing
- The df for each smooth is shown on the y label; 5 df wasn't too far off…
- Generally the smooth-spline solution is the same as the bspline and cubic solution
- Next we examine which model provides the best fit…

Comparing model fits…

```
> ## By default, anova() for mgcv uses
> ## F tests that are hard to interpret.
> ## If we set test="Cp" we can compare
> ## models with Mallows' Cp, which is
> ## similar to AIC.

> anova(lm.0, lm.poly2, lm.poly3, lm.bs,
+ gam.ss, test="Cp")

  Res.Df    RSS     Df Sum of Sq     Cp
1 1177.0 781.18                   791.36
2 1174.0 718.27 3.0000    62.906 731.85
3 1171.0 675.78 3.0000    42.490 692.76
4 1165.0 659.23 6.0000    16.555 683.00
5 1163.6 658.52 1.4062     0.702 683.89


> ## Or we can compare with AIC
> ## directly...
```

```
> models <- list(lm.0, lm.poly2,
+ lm.poly3, lm.bs, gam.ss)
> names(models) <- strsplit(
+ "lm.0,lm.poly2,lm.poly3,lm.bs,gam.ss",
+ ",")[[1]]

> my.anova <- data.frame(
+ logLik=sapply(models,logLik),
+ df=sapply(models,
+   function(x) attributes(logLik(x))$df),
+ AIC=sapply(models,AIC),
+ BIC=sapply(models,BIC))

> rownames(my.anova) <- names(models)
> round(my.anova,2)
           logLik    df     AIC     BIC
lm.0     -1435.26 10.00 2890.52 2941.30
lm.poly2 -1385.47 13.00 2796.95 2862.97
lm.poly3 -1349.31 16.00 2730.63 2811.88
lm.bs    -1334.61 22.00 2713.21 2824.94
gam.ss   -1333.98 23.41 2714.76 2833.63
>
```

Aside: Comparing mgcv's summary() with base-R's summary()…

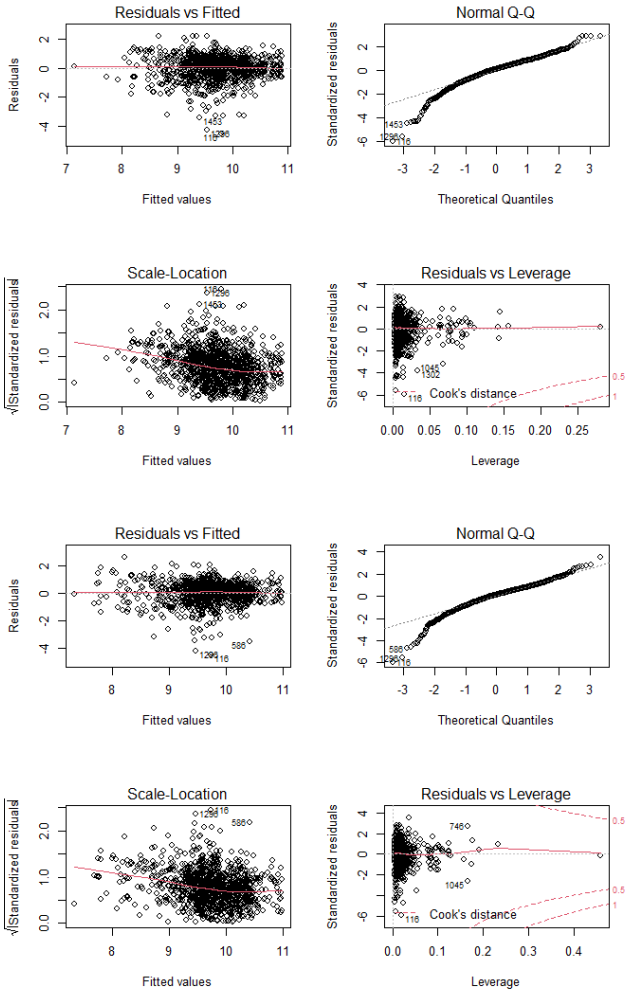
```
> summary(gam.ss)
[…]
Parametric coefficients:
        Est          SE    tval     pval
(Int)  9.97    0.09978  99.883  < 2e-16
sex   -0.51    0.06584  -7.700 2.89e-14
race2 -0.05    0.07544  -0.663   0.5074
race3  0.06    0.20646   0.278   0.7811
race4 -0.40    0.23152  -1.742   0.0817
hisp   0.05    0.09710   0.506   0.6126

Approximate significance of smooth terms:
            edf Ref.df     F p-value
s(ed)     4.505  5.437 23.27  <2e-16
s(yearbn) 7.021  8.002 31.96  <2e-16
s(height) 4.880  5.976  1.19   0.317

> ## Interpret the F stat and its pval
> ## like we do the t stat and its pval:
> ## We are testing whether to add this
> ## term in the model, *after* adding
> ## other terms.
```

```
> round(summary(lm.bs)$coef,2)
                 Est   SE   tval   pval
(Int)           7.67 0.64  11.94   0.00
sex            -0.50 0.07  -7.56   0.00
race2          -0.04 0.08  -0.59   0.56
race3           0.04 0.21   0.19   0.85
race4          -0.40 0.23  -1.75   0.08
hisp            0.06 0.10   0.57   0.57
bs(ed)1         1.57 0.80   1.97   0.05
bs(ed)2         1.66 0.49   3.38   0.00
bs(ed)3         2.38 0.54   4.40   0.00
bs(ed)4         2.08 0.51   4.04   0.00
bs(ed)5         2.72 0.52   5.26   0.00
bs(yearbn)1    -1.41 0.51  -2.80   0.01
bs(yearbn)2     0.07 0.26   0.27   0.78
bs(yearbn)3    -0.57 0.34  -1.69   0.09
bs(yearbn)4    -0.66 0.30  -2.23   0.03
bs(yearbn)5    -2.24 0.33  -6.77   0.00
bs(height)1     1.15 0.45   2.58   0.01
bs(height)2     0.45 0.27   1.65   0.10
bs(height)3     1.07 0.37   2.88   0.00
bs(height)4     0.58 0.33   1.74   0.08
bs(height)5     1.01 0.42   2.41   0.02
```

Residual plots for the best two models: lm.poly3 and lm.bs

```
> par(mfrow=c(2,2))
> plot(lm.poly3)
> plot(lm.bs)
```

- The residuals for lm.bs look just slightly better than for lm.poly3 and lm.0
- All the models have a problem with extreme low outliers in the residuals
- There may be a predictor we don't have in the data set



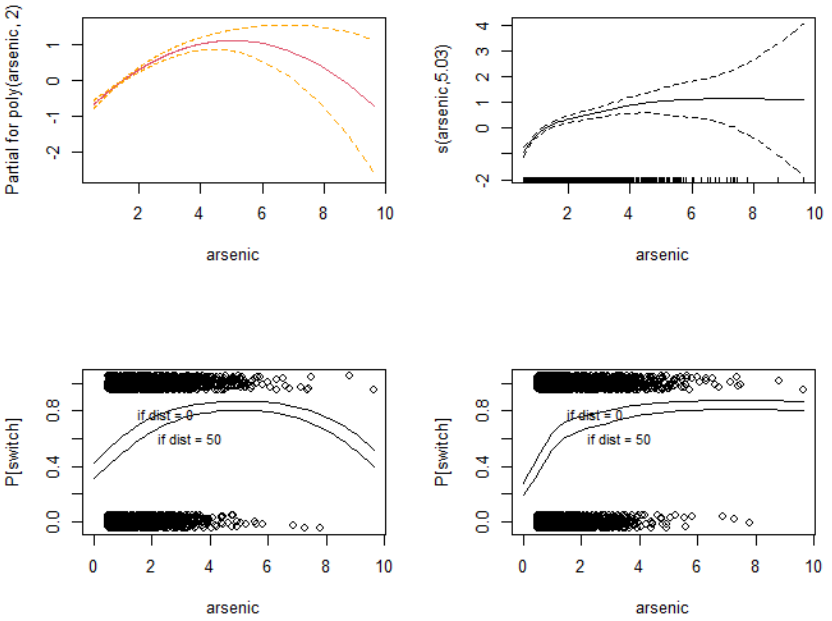
Example #2: The Wells data…

- After educating families in Bangladesh about the harmful effects of arsenic, do they switch water wells?

```
'data.frame':   3020 obs. of  5 variables:
 $ switch : int  1 = switched to a new well; 0 = didn't
 $ arsenic: num  arsenic level in the old well
 $ dist   : num  distance to nearest safe well
 $ assoc  : int  1 = head of household is comm. leader
 $ educ   : int  years of schooling of head of household
```

Let's review our last model for this data, and see if gam can improve it

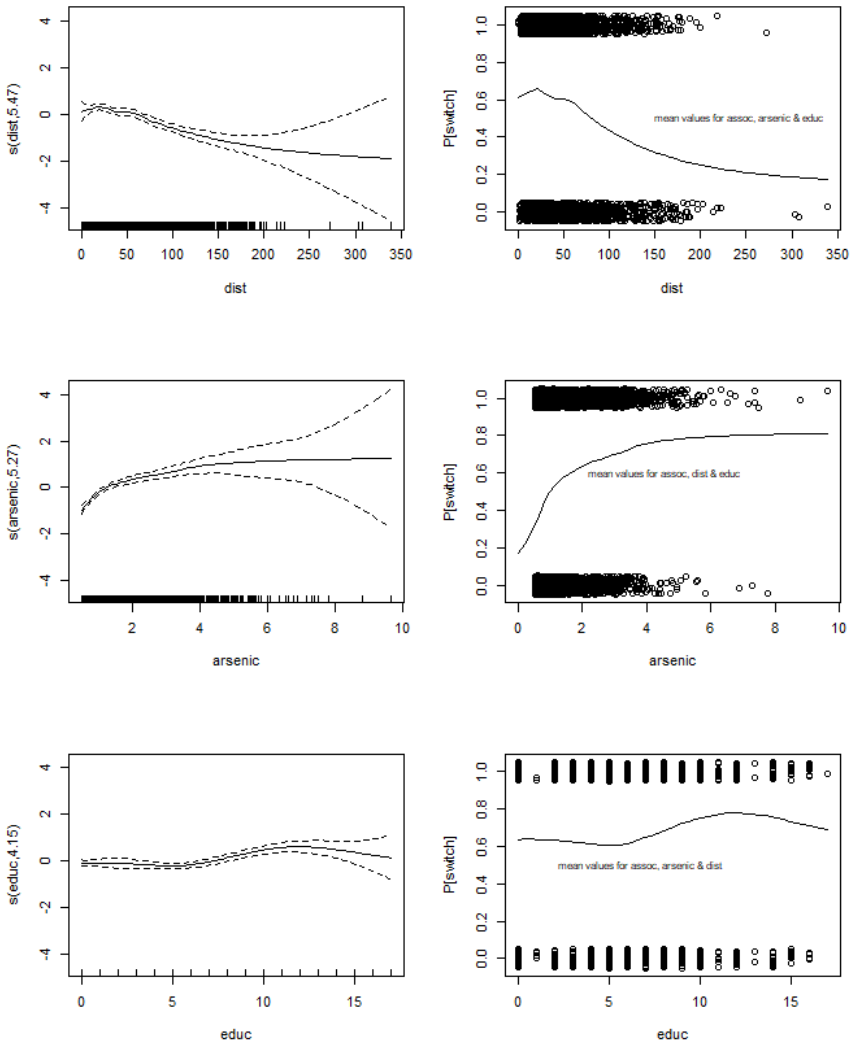
```
> ## The model we liked before was:
> glm.3 <- glm(switch ~ dist +
+ poly(arsenic,2),data=wells,
+ family=binomial)

> ## let's try smooth spline for arsenic...
> gam.3 <- gam(switch ~ dist +
+ s(arsenic,bs="cr"), data=wells,
+ family=binomial)

> par(mfrow=c(2,2))
> termplot(glm.3,se=T,terms=2)
> plot(gam.3)

> ## Now let's see how the predict P[switch}
>
> ## Please see the r file for this lecture
> ## for the details of building these plots…
```



- The smooth spline has a better interpretation than the quadratic
- Increasing arsenic increases the prob of switching, up to a point, and then it levels off.

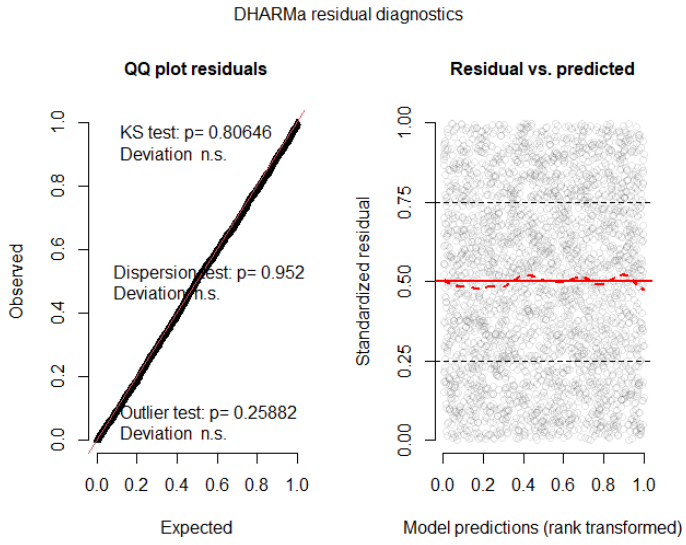
Let's see what happens if we use all the predictors…

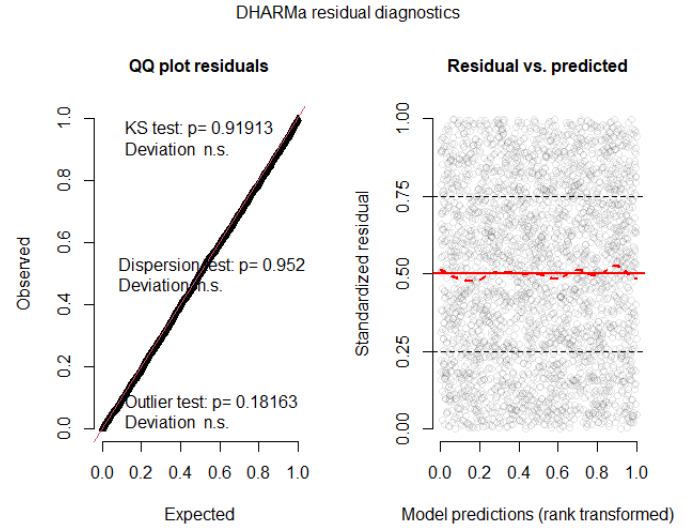
```
> glm.all <- glm(switch ~ assoc +
+ dist +
+ arsenic +
+ educ,
+ data=wells,family=binomial)
> gam.all <- gam(switch ~ assoc +
+ s(dist,bs="cr") +
+ s(arsenic,bs="cr") +
+ s(educ,bs="cr"),
+ data=wells,family=binomial)
> par(mfcol=c(3,2))
> plot(gam.all,se=T,ask=F)
> attach(wells)
>
> ## code for generating the rest
> ## of the plot is in the r file
> ## for this lecture…
```



Model comparisons…

```
> models <-
+ list(glm.3,glm.all,gam.3,gam.all)
> names(models) <-
strsplit("glm.3,glm.all,gam.3,gam.all",
+ ",")[[1]]


> my.anova <-
+ data.frame(logLik=sapply(models,logLik),
+ df=sapply(models,
+   function(x) attributes(logLik(x))$df),
+ AIC=sapply(models,AIC),
+ BIC=sapply(models,BIC))
> rownames(my.anova) <- names(models)
> round(my.anova,2)

          logLik    df      AIC      BIC
glm.3    -1955.68  4.00 3919.35 3943.40
glm.all  -1953.91  5.00 3917.83 3947.89
gam.3    -1947.70  7.03 3909.46 3951.76
gam.all  -1918.10 16.90 3870.01 3971.65

> library(DHARMa)
> invisible(simulateResiduals(glm.3,plot=T))
> invisible(simulateResiduals(gam.3,plot=T))
```





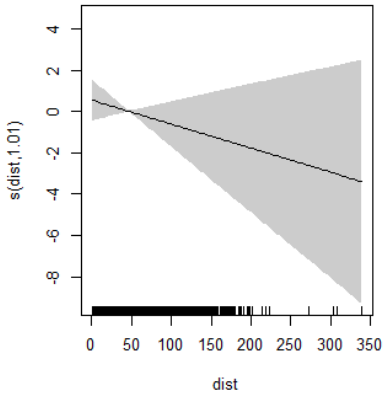
Let's look at the model summaries…

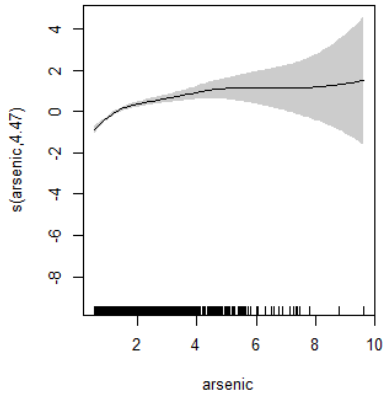
```
> summary(gam.3)
[…]
Parametric coefficients:
    Estimate      SE z-value   Pr(>|z|)
(Int)  0.798   0.065  12.286     <2e-16
dist  -0.010   0.001  -9.231     <2e-16

[…]

Approximate significance of smooth terms:
             edf Ref.df Chi.sq p-value
s(arsenic) 5.035  6.053  165.9  <2e-16

[…]

R-sq.(adj) =   0.07
Deviance explained = 5.41%
UBRE = 0.29452  Scale est. = 1
n = 3020
```

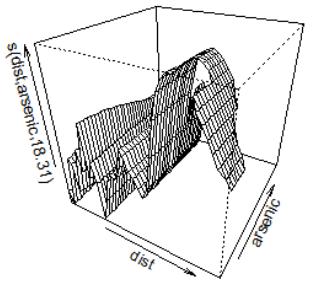
```
> summary(glm.3)
[…]
Coefficients:
                    Estimate SE   z value
Pr(>|z|)
(Intercept)  0.792 0.065  12.204  < 2e-16
dist        -0.010 0.001  -9.079  < 2e-16
poly(ars)1  26.21  2.330  11.249  < 2e-16
poly(ars)2 -10.50  2.204  -4.765 1.89e-06

[…]

    Null deviance: 4118.1  on 3019  df
Residual deviance: 3911.4  on 3016  df
AIC: 3919.4

Number of Fisher Scoring iterations: 4
```

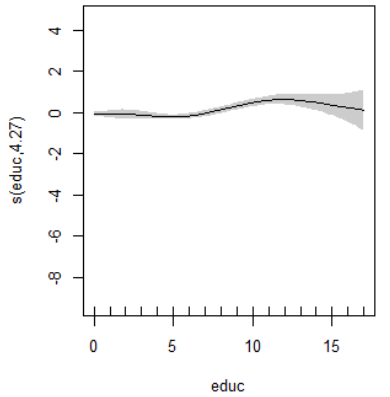
Just a quick taste of nonparametric interactions…

```
> gam.int <- gam(switch ~ assoc +
+ s(dist) +
+ s(arsenic) +
+ s(dist,arsenic) +
+ s(educ),
+ data=wells,family=binomial)
> summary(gam.int)
[…]
Parametric coefficients:
         Est    SE      z-val    p-val
(Int)  0.373    0.051   7.250 4.15e-13
assoc -0.094    0.079  -1.191    0.233
[…]
Approximate significance of smooth terms:
               edf Ref.df  Chi.sq p-value
s(dist)      1.005  1.005   1.284  0.2618
s(arsenic)   4.466  5.488 111.843  <2e-16
s(dist,ars) 18.308 27.000  30.965  0.0125
s(educ)      4.270  5.166  42.773  <2e-16
[…]
> plot(gam.int,scheme=1,pages=1)
```









Since I did not specify bs="cr" in the s() functions, the splines are all thin plate splines.

Advice on nonparametric regression

- Although there are many technical details, nonparametric methods are flexible & play well with lm's & glm's → GAMs!
- There are usually "tuning parameters":
  - **Functional forms**: degree of spline, endpoint conditions, location of knots, degree of local regression, choice of kernel function – *choose these based on defaults/experience/need*
  - **df-driving parameters**: m (number of knots), λ, s, h – *choose these based on guess-and-check, K-fold CV, LOOCV, etc.*
- In GAM's:
  - For terms that have not been smoothed, interpret coefficients as usual
  - For smoothed terms, interpret graph of transformation, or of transformation predicting partial residuals, instead.
  - Start simple (e.g. no smoothed terms) and build up; take the simplest model that does what you & your client/collaborator need!

Summary

- Other nonparametric linear smoothing methods
  - Local Regression Smoothers
    - loess()
    - Kernel Smoothed Regression
- A Comparison of Linear Smoothers
- Generalized Additive Models (GAMs)
  - Nonparametric transformations for the Heights data
  - Nonparametric transformations for the Wells data
- And beyond… (nonparametric interactions)
- Advice on nonparametric regression