Nonlinearity and Generalized Additive Models 
Lecture 2

Robert Andersen
McMaster University

http://socserv.mcmaster.ca/andersen

Definition of a Smoother
• A smoother is simply a tool for summarizing the trend of a response variable Y as a function of one or more predictors X’s
  – Since it produces an estimate of the trend that is less variable than the trend itself, it is called a smoother
  – In this respect, even linear regression is a smoother—it is the smoothest of all possible smoothers
  – Some other smoothers are polynomial regression, regression splines, smoothing splines, lowess smooths, running means, and bin smoothers.
• With respect to nonparametric regression, smoothing splines and lowess smooths play the most important role
• Nonparametric smoothing has two main functions:
  1. To describe a trend, typically important in diagnostic plots
  2. To estimate the dependence of Y on X. They are most useful with respect to generalized additive models, which consider several X’s

Why Nonparametric Regression?
• The linear model is desirable because it is simple to fit, results are easy to understand, and there is a wide variety of useful techniques for testing the assumptions involved
  – Nonetheless, there are cases when the linear model should not be applied because of an intrinsic nonlinearity in the data
• Nonparametric regression provides a means for modelling such data
  – Nonparametric regression can be used as a benchmark for linear models against which to test the linearity assumption
• Nonparametric regression also provides a useful way to enhance scatterplots to display underlying structure in the data—as we have already seen, it plays an important role in diagnostic plots

A More General Way to Think of Regression Analysis
• Rather than expect a linear relationship, we can trace the conditional distribution of a dependent variable, Y, as a function of one or more explanatory variables, X’s
  \[ p(y|x_1, ..., x_k) = f(x_1, ..., x_k) \]
• Consider the simple case of a single predictor variable
• With large samples and when the values of X are discrete, it is possible to estimate the regression by determining the mean of Y (could also use the median) at each value of X:
  \[ \mu = E(Y|x) = f(x) \]
• A naïve nonparametric regression line connects the conditional means
Introduction to Nonparametric Regression (2)

- Simple linear regression would work perfectly well for the graph on the left, but not the graph on the right.

Introduction to Nonparametric Regression (3)

- If $X$ is continuous, we may not have enough cases at each value of $X$ to calculate precise conditional means.
- If we have a large sample, we can dissect the range of $X$ into narrow bins that contain many observations, obtaining fairly precise estimates of the conditional mean of $Y$ within them.
  - More advanced nonparametric models fit local regressions to the data in the bins.
  - A line connecting the fitted values of these individual regressions is then graphed.
- **Nonparametric regression has no parameters**
  - Since there are no coefficients, the fitted curve must be plotted to see the relationship.

Types of Nonparametric Regression

- There are several types of nonparametric regression. The most commonly used is the **lowess (or loess)** procedure first developed by Cleveland (1979).
  - Lowess (or loess) is an acronym for **locally weighted scatterplot smoothing**.
  - These models fit local polynomial regressions and join them together.
- Another important set of nonparametric models are **smoothing splines**.
  - These models partition the data and fit separate piecewise regressions to each section, smoothing them together where they join.
- **Generalized additive models** extend these models to multiple regression and to non-quantitative dependent variables (*i.e.*, logit models etc.).
  - These models specify a separate functional form for the relationship between $Y$ and each $X$.

Lowess: Local polynomial regression

How does it work?

- The following example explains the most common type of nonparametric regression—**lowess (locally weighted scatterplot smoothing)** or locally weighted polynomial regression.
- The data are from 27 pre-election polls conducted before the 1997 Canadian election.
  - **Time**: Measured in number of days from January 1st, 1997.
  - **Liberal**: Percentage of respondents in the polls supporting the Liberal Party.
- A scatterplot shows the possibility of a nonlinear relationship, so we’ll try to explore this relationship using lowess.
- In the following slides I show how a loess smooth is fit step-by-step.
Lowess: Local Polynomial Regression

Step 1: Defining the window width

- The first step is to define the \textit{window width} $m$, that encloses the closest neighbours to each data observation (the \textit{window half-width} is labelled $h$)
  - For this example, we use $m=16$ (\textit{i.e.}, for each data point we select the 16 nearest neighbours in terms of their $X$-value)
    - 16 was chosen to represent 60% of the data
    - The researcher typically chooses the window width by trial and error (more on this later)
  - The graph on the following page shows the 16 closest observations to $X_{(10)}=88$. Here we call $X_{(10)}$ our \textit{focal X}
- Although for this example I start at $X_{(10)}$, in the real case we would start with the first observation and move through the data, finding the 16 closest observations to each case

Local Polynomial Regression

Step 1 (R-script)

```r
#Defining the window width
plot(TIME, LIBERAL, xlab="Time (in days)", ylab="Liberal Support",
     type="u", main="Defining the Window Width")
ord <- order(TIME)
time <- TIME[ord]
pre <- LIBERAL[ord]
x0 <- time[10]
diffs <- abs(time - x0)
which.diff <- sort(diffs)[16]
abline(v=c(x0-which.diff, x0+which.diff), lty=2)
abline(v=x0)
points(time[diffs > which.diff], Lib[diffs > which.diff], pch=16, cex=2, col=gray(.75))
points(time[diffs <= which.diff], Lib[diffs <= which.diff], cex=2)
x.s <- time[diffs <= which.diff]
y.s <- Lib[diffs <= which.diff]
text(locator(1), "Window Width")
```
Local Polynomial Regression

Step 2: Weighting the data

- We then choose a kernel weight function to give greatest weight to observations that are closest to the focal \( X \) observation
  - In practice, the tricube weight function is usually used
- Let \( z_i = (x_i - x_0)/h \), which is the scaled distance between the predictor value for the \( i \)th observation and the focal \( x \)
  \[
  W_T(z) = \begin{cases} 
  (1 - |z|^3)^3 & \text{for } |z| < 1 \\
  0 & \text{for } |z| \geq 1 
  \end{cases}
  \]
  Here \( h \) is the half-width of the window centred on \( x_i \)
- Notice that observations more than \( h \) (the half-window or bandwidth of the local regression) away from the focal \( x \) receive a weight of 0

### Local Polynomial Regression

Step 2: R-script

```r
# The Tricube Weight
tricube <- function(z) {
  ifelse (abs(z) < 1, (1 - (abs(z))^3)^3, 0)
}
plot(range(TIME), c(0,1), xlab="Time (in days)", ylab="Tricube Weight",
     type='n', main="The Tricube Weight")
abline(v=c(x0-which.diff, x0+which.diff), lty=2)
abline(v=x0)
xwts <- seq(x0-which.diff, x0+which.diff, len=250)
lines(xwts, tricube((xwts-x0)/which.diff), lty=1, lwd=2)
points(x.n, tricube((x.n - x0)/which.diff), cex=2)
```

Local Polynomial Regression

Step 3: Locally weighted least squares

- A polynomial regression using weighted least squares (with the kernel weights) is then applied to the focal \( X \), using only the nearest neighbours to minimize the weighted residual sum of squares
  - Typically a local linear regression or a local quadratic regression is used, but higher order polynomials are also possible
  \[
  Y_i = A + B_1(x_i-x_0) + B_2(x_i-x_0)^2 + \cdots + B_p(x_i-x_0)^p + E_i
  \]
- From this regression, we then calculate the fitted value for the focal \( X \) value and plot it on the scatterplot
  - The regression line within the window in the following graph shows the fitted value for the focal \( x_i \) from a local linear regression
Local Polynomial Regression

**Step 3**

Local Linear Regression

![Diagram showing local linear regression with fitted value of Y at focal X](image)

Locally Weighted Averaging

**Step 4: The Nonparametric Curve**

- Steps 1-3 are carried out for each observation in the data
  - There is a separate local regression for each value of X
  - A fitted value from these regressions for each focal X is plotted on the scatterplot
- The fitted values are connected, producing the local polynomial **nonparametric regression curve**
  - As we shall see later, the larger the window width, the smoother the curve
  - Also, it is important to remember that there are no parameter estimates—**The effect of X on Y must be graphed**

```r
plot(TIME, LIBERAL, xlab="Time (in days)", ylab="Liberal Support", main="The Lowess Fit", cex=2)
lines(lowess(TIME, LIBERAL, f=0.6, iter=0), lwd=2)
```

Robustness Weights

**Adjusting for outliers (1)**

- Since we are trying to determine an underlying structure in the data, **we don’t want unusual cases to have extraordinary influence on the curve**
- Following from the linear regression case, M-Estimation for robust regression can be adapted to ensure that the lowess smooth is not unduly affected by outliers
  1. We start by calculating the residuals from the fitted values for the local regressions
     \[
     E_i = Y_i - \hat{Y}_i
     \]
  2. Determine a measure of the scale of the residuals (e.g., the **median absolute deviation** from the **median residual**):
     \[
     \text{MAD} = \text{median}(|E_i - \bar{E}|)
     \]
     where \( \bar{E} = \text{median}(E_i) \)
**Robustness Weights Adjusting for outliers (1)**

3. Calculate resistance weights $v_i$ for each observation using an appropriate weight function to determine the relative size of each residual. Here we use the **Bisquare Weight Function**:

$$v_i = w_H(z) = \begin{cases} 
(1 - z^2)^2 & \text{for } |z| < 1 \\
0 & \text{for } |z| \geq 1 
\end{cases}$$

where $z = \frac{E_i}{t \times \text{MAD}}$

and $t$ is a tuning constant

- $t=6$ MADs corresponds approximately to 4 standard deviations. In other words, we exclude observations that have a probability of being observed of less than 0.0001.

---

**Robustness Weights Adjusting for outliers (3)**

4. We then refit the local polynomial regressions using both local weights ($w_i$) and the resistance weights ($v_i$)

5. From these new regressions, we calculate new fitted values

6. Steps 1-4 are repeated (iterated) until the fitted values stabilize

7. Finally, a curve is drawn to connect the fitted values, giving us the **lowess smooth**

```r
plot(TIME, LIBERAL, xlab="Time (in days)", ylab="Liberal Support", main="The Robust Lowess Fit", cex=2)
lines(lowess(TIME, LIBERAL, f=0.6), lwd=2)
lines(lowess(TIME, LIBERAL, f=0.6, iter=0), lty=2, col="red")
legend(locator(1), lty=c(1:2), lwd=c(2,1), col=c("black", "red"), legend=c('Robust', 'Non-robust'))
```

---

**Robustness Weights Adjusting for outliers (4)**

- In this case the robust fit is nearly identical to the regular lowess fit, indicating that outliers are not problematic

- Most lowess procedures use the robust fit by default

---

**Lowess in R**

- Local polynomial regression (or lowess smooths) can be fit several ways in R, using several different packages:
  - The base package of R contains the `lowess` function
  - The `stats` package contains the `loess` function
  - The `locfit` package contain `locfit` and `locfit.robust` functions
  - The `sm` package contains the `sm.regression` function

- Each package has some comparative strengths and weakness
  - The unfortunate thing is that they all do not specify the models in the same way—i.e., they have different model structure commands, and different ways of specifying the window width $m$ (or window span $S$)
Lowess in R (2)

- Below I have specified the same model using some R functions. Notice the span (0.6—or 60% of the data. More about this later) is specified differently for each function. The model command also differs

```r
# lowess in the base package:
model.lowess<-lowess(TIME,LIBERAL, f=0.6)
# lowess in modreg package:
library(stats)
model.loess<-loess(LIBERAL~TIME, span=0.6, degree=1)
# lowess in locfit
library(locfit)
model.locfit<-locfit(LIBERAL~TIME, alpha=0.6, deg=1)
# nonparametric smooths in sm
library(sm)
model.sm<-sm.regression(TIME, LIBERAL, h=0.3)
```

Lowess in R (3)

- The separate functions also have different ways of graphing the loess smooth

```r
split.screen(figs=c(2,2))
#lowess
screen(1)
plot(TIME, LIBERAL, main="lowess")
lines(model.lowess)
#loess
screen(2)
plot(TIME, LIBERAL)
lines(loess.smooth(TIME, LIBERAL, main="loess"))
#locfit
screen(3)
plot(TIME,LIBERAL, main="locfit")
lines(model.locfit)
close.screen(all = TRUE)
```

Assumptions under the Lowess Model

- The assumptions under the Lowess model are much less restrictive than the assumptions for the linear model
  - Most importantly, no strong global assumptions are made about \( \mu \)
  - We assume, however, that locally around a point \( x \), \( \mu \) can be approximated by a small class of parametric functions (polynomial regression)
  - Still, the errors \( \varepsilon_i \) are assumed independent and randomly distributed with mean 0
- Nevertheless, a number of choices—particularly window width, type of polynomial and type of weight function—affect the trade-off between bias and variance
**Window Span**

**Trade-off between bias and variance**

- The window width \( m \), is the number of cases in each local regression.
- It is more practical to think in terms of percentage of cases closest to the focal point (i.e., nearest neighbours), which we call the span \( S \).
  - A span that is too small (meaning that insufficient data fall within the window) produces a curve characterised by a lot of noise—i.e., this results in a large variance.
  - If the span is too large, the regression will be over-smoothed and thus the local polynomial may not fit the data well—i.e., it results in loss of important information, and thus large bias.
- We want the smallest span that provides a smooth fit.
  - The default span for loess is \( S = 0.75 \); for lowess it is \( f = 2/3 \).

**Local Polynomial Degree**

- The degree of the polynomial also affects the bias-variance trade-off.
  - A higher degree polynomial will provide a better approximation of the underlying mean than a lower polynomial degree—i.e., a higher degree polynomial will have less bias.
  - Higher degree polynomials also have more coefficients to estimate, however, resulting in higher variability.
- It is usually most effective to choose a low degree polynomial and concentrate instead on choosing the best bandwidth.
  - The most commonly used polynomials are local linear and local quadratic—i.e., the local linear has more bias, but has less variance, especially at the boundaries.
  - Cubic and higher order polynomials tend not to improve the fit by much.
**Weight Function**

- The choice of the weight function has much less effect on the bias-variance trade-off than other elements of the nonparametric specification, but it can affect the visual quality of the fitted regression curve.
- Although there is no restriction on the particular weight function that is used, it is desirable to use a smooth and continuous weight.
- The most commonly used weight function for nonparametric regression models is the tricube weight function.
  - In fact, all of the loess functions in R use this weight function, and this cannot be altered easily. I see no reason why you would want to change this.

**Statistical Inference and Approximate Degrees of Freedom**

- The concept of degrees of freedom for nonparametric models is not as intuitive as for linear models since there are no parameters estimated.
- Nonetheless, the approximate degrees of freedom are a **generalization of the number of parameters in a parametric model**.
  - The analogy to the linear model model is not perfect, but approximate.
- Using the approximate degrees of freedom, we can carry out F-tests to compare different estimates applied to the same dataset:
  - Compare different levels of polynomial fits; Compare the smoothed model to a linear model etc.
- Determining the degrees of freedom is also necessary for constructing confidence envelopes around the fitted curve.

**Statistical Inference and Approximate Degrees of Freedom (2)**

- Although the most obvious way to define the degrees of freedom in parametric regression is by the number of predictors, $k$ (including the constant), the **trace of the hat matrix** $H$—which transforms $Y$ into $Y$-hat—also contains this information.
- Analogous degrees of freedom for nonparametric models are obtained by substituting the smoother matrix $S$, which plays a similar role to the hat matrix $H$—i.e., it transform $Y$ into $Y$-hat.
  - In other words, $df_{MOD} = \text{trace}(S)$
  - Unlike in linear model, the degrees of freedom for nonparametric regression are not necessarily whole numbers.
- The residual degrees of freedom is then $df_{RES} = n - df_{MOD}$, and the **estimated error variance** is $S^2 = \sum e^2 / df_{RES}$.
- This method is used for both lowess and smoothing splines.

**Confidence Envelope for the Regression Curve**

- The estimate variance of the fitted value $Y$-hat at $X=x_i$ is
  $$\hat{V}(\hat{y}_i) = S^2 \sum_{j=1}^{n} w_{ij}^2$$
  where $S^2$ is the estimated error variance and $w_{ij}$ are the weights applied to each $x$.
- Assuming normally distributed errors, an approximate 95% confidence interval the population regression $\mu|X_i$ is
  $$\hat{y}_i \pm 2\sqrt{\hat{V}(\hat{y}_i)}$$
  We then simply join the confidence intervals for each of the $X$-values together to plot the **95% confidence band or envelope** for the regression function.
- Alternatively, bootstrap standard errors can be used.
Nonparametric regression as a test for nonlinearity (1)

- F-tests comparing the residual sums of squares for alternative nested models can be carried out in exactly the same way as for linear models—Remember, however: these tests are only approximate because of the approximation of the df
- A useful test for F-Test for nonlinearity is to contrast a nonparametric regression with a linear regression
  - These models are nested because a linear model is a special case of the general, nonlinear, relationship
  \[ F_0 = \frac{RSS_0 - RSS_1}{\text{trace}(S)/2} \]
  \[ \frac{RSS_1}{n-\text{trace}(S)} \]

Where RSS_0 is the residual sum of squares from the linear model; RSS_1 is from the nonparametric model; trace(S) is the df for the nonparametric model.

Nonparametric Regression as a test for nonlinearity (2)

- Returning to the Prestige data, I regress prestige on income
- We see here that there is a clear departure from linearity in these data—the linear line (in red) strays outside the 95% confidence envelope for the loess smooth
- An incremental F-test gives a formal test for nonlinearity

Nonparametric Regression as a test for nonlinearity (3)

```r
### R script for plot
#linear model
mod1<-lm(prestige~income)
#loess smooth
mod2<-loess(prestige~income)

### plotting the 95% confidence bands for the smooth
inc<-seq(min(income), max(income), length=500)
fit<-predict(mod2, data.frame(income=inc), se=TRUE)
plot(income, prestige, ylim=c(15,95))
lines(income, fit$fit, lwd=3)
lines(income, fit$fit+2*fit$se.fit, lty=2)
lines(income, fit$fit-2*fit$se.fit, lty=2)
#linear regression line
abline(lm(prestige~income), col="red", lwd=2)
```

Nonparametric Regression as a test for nonlinearity (4)

- The F-test below returns a P-value=.0022, indicating that there is a significant departure from linearity

```r
mod1<-lm(prestige~income)
summary(mod1)# to get N (102) and df (2)
mod2<-loess(prestige~income)
summary(mod2)#to get df (trace=7.20)

# residual sum of squares
rss1<-sum(residuals(mod1)^2)
rss2<-sum(residuals(mod2)^2)

# F-Test
Ftest<-(rss1-rss2)/(7.28-2)/(rss2/(102-7.28))
pf(Ftest, 7.28-2, 102-7.28, lower.tail=FALSE)
```
Diagnostics

- Fit in nonparametric regression depends not only on variable selection but also the choice of smoothing parameter.
- As with the linear model, the most important diagnostic component is the residuals.
- Several plots can be useful:
  - **Residuals vs predictor values** (useful for detecting lack of fit, such as trimmed peaks, and nonconstant error variance)
  - **Quantile comparison plots of the residuals** (to detect departures from normality in the residual distribution)
  - **Absolute residuals vs predictors** (to detect dependence of the residuals on predictors)
- These plots are easy to construct in R.

Quantile Comparison Plot of Residuals: Looking for outliers

```
> library(car)
> qq.plot(residuals(mod2))
```

Residual Plot: Looking for nonconstant error variance and poor fit

```
> plot(mod2$fit, residuals(mod2),
    xlab="Fitted Values", ylab="Residuals")
> abline(h=0,lty=2)
```

Absolute Residuals versus Predictors: Looking for dependence

```
> plot(income, abs(residuals(mod2)), xlab="Income",
    ylab="Absoulte Residuals", cex=1.5)
> lines(lowess(income, abs(residuals(mod2))))
```
Nonparametric Regression for Non-Continuous Dependent Variables

- The `sm` package has functions for fitting local regressions to binary and Poisson data
- Below is an example of a nonparametric logistic regression using the 1999 Indian Election data

```r
data <- read.table('c:/teaching/icpsr/data/India99.dat', header=T)
attach(data)

# Predicting vote for RJP (1=yes, 0=no)
# from liberal attitudes scale
model <- sm.binomial(LIBERAL, BJPVOTE, h=.5, display="se")
detach(data)
```

Nonparametric Poisson Regression (1)

- The example below uses nonparametric Poisson regression to predict number of association memberships (ASSOC) from AGE
- The data are a subset of the 1981-91 World Values Survey (n=4803), using only respondents from Canada, Great Britain, United Kingdom and the United States
- The nonparametric regression is specified using the `sm.poisson` function in the `sm` package

```r
library(sm)
sm.poisson(AGE, ASSOC, h=5, display="se")
```
Smoothing Splines

- Nonparametric regression that extends cubic splines by minimizing the \textit{penalized residual sum of squares}

\[ RSS(f, \lambda) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int_{x_{\min}}^{x_{\max}} f''(x)^2 \, dx \]

- The first term measures the closeness to the data; the second term penalizes curvature in the function
- Here \( \lambda \) is a fixed \textit{smoothing parameter} with respect to the unknown regression function \( f(x) \) that is found on the basis of the data \((x_i, y_i)\). The rate of change of the slope of the function \( f \) is given by \( f'' \)
  - In other words, \( \lambda \) (which must be positive) establishes a tradeoff between the closeness of fit and the penalty
  - If \( \lambda = 0 \), \( f \) is the function that interpolates the data; If \( \lambda \to \infty \), the second derivative is constrained to 0, and thus we have a constant slope (least squares fit)

Smoothing Splines (2)

- The smoothing spline estimator solves the problem of allowing fits with variable slope, but it creates a new problem
  - \textit{How do we determine the appropriate value for the smoothing parameter \( \lambda \) for a given data set?}
- The same value of \( \lambda \) is unlikely to work equally well with every dataset
- 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 \( \lambda \) 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 \( \lambda \)

Smoothing Splines (3)

Cross-Validation for choosing \( \lambda \)

- Cross-validation resamples the original sample
- The data are split into \( k \) subsets and then our model is fit \( k \) times, each time trying to predict using the left-out subset
- \textit{Prediction error} for each subset is then calculated as the sum of the 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)

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

- \textit{Generalized cross validation} (GCV) is the most commonly used method to choose the smoothing parameter \( \lambda \) for smoothing spline models

Smoothing Splines (5)

Cross-Validation for choosing \( \lambda \)

- \textit{Generalized Cross-Validation} 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 by

\[ GCV(\lambda) = \frac{n^{-1} RSS(f_\lambda)}{(1 - n^{-1} \sum_{i=1}^{n} h_i(x_i))^2} \]

- Simply put, GVC compares the fit of all models based on all possible values of \( \lambda \), choosing the one that fits best
- GVC choice of \( \lambda \) is typically the default method in software programs, including the packages in \textbf{R}
Smoothing Splines (6)
Fixing the Degrees of Freedom

• Aside from variable selection, model selection for smoothing splines depends on the degree of the polynomial, the number of knots, and the placement of the knots
  – By definition, the knots are at all unique values of \( X \)
  – Cubic degree is almost always used in practice
  – This means, then, that we need only select the smoothing parameter \( \lambda \)

• Since \( df = \text{trace}(S_\lambda) \) is monotone in \( \lambda \), we can invert the relationship and specify \( \lambda \) by fixing the degrees of freedom
  – A large \( df \) results in a small penalty \( \lambda \), resulting in a jagged curve (larger variance); A small \( df \) results in a large penalty \( \lambda \), and thus a jagged curve (larger bias);
  – This method of using the \( df \) provides a uniform approach to compare many different smoothing methods

Smoothing Splines in R

• Two packages in R can be used to fit smoothing splines:
  – The `smooth.spline` function in the `splines` package
  – The `sm.spline` function in the `pspline` package

• Since \( df = \text{trace}(S_\lambda) \) we can specify either \( \lambda \) directly, or invert the relationship and specify the degrees of freedom instead
  – The latter method is much easier and somewhat more intuitive
  – By default, GCV is used by both the `smooth.spline` function and the `sm.spline` functions to choose \( \lambda \)

• Remember, like lowess models, this is a nonparametric model so the effects must be graphed

Fit According to Degrees of Freedom

<table>
<thead>
<tr>
<th></th>
<th>CV=2.9 df</th>
<th>5 df</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Coffee Break
**Multiple Nonparametric Regression**
- Formally, local polynomial regression and smoothing splines extend easily to multiple regression
- In practice, however, the *Curse of Dimensionality* gets in the way
- For example, as the number of predictors increases, the number of points in a local neighbourhood for a particular focal point declines
  - In order to include a fixed number of cases in each local regression the span necessarily gets larger, and thus the regression becomes less local—as a result the bias of the estimate increases
- Moreover, because nonparametric regression has no parameter estimates, the model becomes difficult to interpret the more explanatory variables we include
  - Graphing the fitted values helps, but with more than two predictors we can’t see the complete regression surface
- Despite these limitations, multiple nonparametric regression with two explanatory variables can be useful

**Local Polynomial Multiple Regression: Example: Prestige data**
- Using the **Prestige** data in the **car** package, we fit two models:
  1. *prestige* regressed on *income* only
  2. *prestige* on *income* and *education*

  ```r
  library(stats)
  mod.loess1<-loess(prestige~income)
  mod.loess2<-loess(prestige~income+education)
  ```
- The general tests used in the simple local polynomial case can also be used here, and can be extended to the incremental F-test for terms in the model
  - That is, we can test the impact of a term by comparing a model with it to a model without it
- First, let’s explore the regression surface for `mod.loess2`

**Local Polynomial Multiple Regression: Example: Prestige data (2)**
- The regression surface is clearly nonlinear, both for income and education
- The next step is a type II incremental F-test for the terms in the model
- We could also test for nonlinearity using an F-test, comparing the RSS of this model with the RSS of the linear model

```r
#perspective plot
#Setting up "newdata" points for fitted values
income2<-seq(min(income), max(income), len=25)
education2<-seq(min(education), max(education), len=25)
data<-expand.grid(income=income2, education=education2)
#Fitted values for each model for the newdata
fit.mod<-matrix(predict(mod.loess2, data), 25,25)
persp (income2, education2, fit.mod, 
theta=35, ph=20, ticktype='detailed',
xlab='Income', ylab='Education', expand=2/3, 
shade=.75, col="yellow", 
zlab='Fitted Prestige')
```
### Statistical Significance Test for Terms

- The `anova` function allows comparison of two models fit with the `loess` function.
- As we see below, there is a statistically significant difference between the model with education and the model without it. We could further proceed to test if a linear trend was appropriate.

```r
> anova(mod.loess1, mod.loess2)
Model 1: loess(formula = prestige ~ income)
Model 2: loess(formula = prestige ~ income + education)

Analysis of Variance: denominator df 89.11

     Df Sum Sq Mean Sq F value Pr(>F)
1     1  5.512 5.5124  21.30 3.8e-07 ***
2     2 42.087 21.044  99.90 < 2e-16 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

### Additive Regression Models

- Limitations of the general nonparametric model motivate the use of **additive models**
  - These models essentially apply local regression to low dimensional projections of the data
- The nonparametric additive regression model is
  \[ Y_i = \alpha + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_k(x_{ik}) + \varepsilon_i \]
- The \( f_i \) in this equation are functions that we will estimate from the data; we assume that the errors \( \varepsilon_i \) have constant error variance and have a mean of 0
- Additive models create an estimate of the regression surface by a combination of a collection of one-dimensional functions
  - In effect, then, GAMs restrict the nonparametric model by excluding interactions between the predictors

### Estimation and backfitting (1)

- The assumption that the **contribution of each covariate is additive** is analogous to the assumption in linear regression that each component is estimated separately
  - If the X's were completely independent—which will not be the case—we could simply estimate each functional form using a nonparametric regression of Y on each of the X's separately
  - 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 will be related, we need to proceed in another way, in effect removing the effects of other predictors—which are unknown before we begin
- We must use what is called **backfitting**, which essentially reduces the multiple regression to a series of two-dimensional partial regression problems to find the curve
- Suppose that we had a two predictor additive model:
  \[ Y_i = \alpha + f_1(x_{i1}) + f_2(x_{i2}) + \varepsilon_i \]
- If we unrealistically knew 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, we see here that smoothing \( Y_i-f_2(x_{i2}) \) against \( x_{i1} \) produces an estimate of \( \alpha+f_1(x_{i1}) \)
- So, **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 estimates
Estimation and backfitting (2)

1. We start by expressing the variables in mean deviation form so that the partial regressions sum to zero.
2. We then take preliminary estimates of each function from a least-squares regression of $Y$ on the $X$'s:
   \[ y_i - \bar{y} = b_1(x_{i1} - \bar{x}_1) + b_2(x_{i2} - \bar{x}_2) + \varepsilon_i \]
3. These estimates are then used as step (0) in the iterative process of estimation:
   \[ f_1^{(0)} = b_1(x_{i1} - \bar{x}_1) \]
   \[ f_2^{(0)} = b_2(x_{i2} - \bar{x}_2) \]
4. Next we find the partial residuals, in order to remove $Y$ from its linear relationship to $X_i$.

Estimation and backfitting (3)

The partial residuals are then:
\[ e^{(1)}_{i1} = e_i + b_1(x_{i1} - \bar{x}_1) \]
\[ = y_i - \bar{y} - b_2(x_{i2} - \bar{x}_2) \]
5. Smoothing these partial residuals against their respective $X$'s provides a new estimate of $f$:
\[ \hat{f}^{(l)}_k = \text{lowess}[Y^{(l)}_{(k)i} \text{ on } x_{ik}] \]
\[ = S_k \{ Y_i - [f_{1(i)}(x_{i1}) + f_{2(i)}(x_{i2})] \} \]
where $S$ is the $(n \times n)$ lowess transformation matrix that depends only on the configuration of $X_j$ for the $j$th predictor.

Estimation and backfitting (4)

- We continually reiterate the process of finding new estimates of the functions by smoothing the partial residuals until the partial functions converge.
- This reduces a multiple regression to a series of two-dimensional partial regression problems, making interpretation easy:
  - Because each partial regression is two-dimensional, we can employ scatterplot smoothing to find the functional form of the relationship between $Y$ and each $X_j$.
  - We can then plot the partial effects for each individual bivariate relationship on separate two-dimensional plots, making them easy to understand.

Approximate Degrees of freedom

- In the nonparametric case the trace of the $S$ smoother matrix gives the $df$, but when the effects are additive we must adjust the degrees of freedom accordingly.
- We start by taking the degrees of freedom for each predictor:
  \[ df_j = \text{trace}(s) - 1 \]
  - 1 is subtracted from each degrees of freedom reflecting the constraint that the partial regression function sums to zero (the $X$'s are in mean-deviation form).
- These individual degrees of freedom are combined for a single measure:
  \[ df_{res} = n = \sum_{j=1}^{k} df_j - 1 \]
  - 1 is subtracted from the final degrees of freedom to account for the constant in the model.
Additive Regression Models in R

- Two packages in R that can be used to fit generalized additive models are the mgcv and the gam package.
- Both specify models in a similar way, using a gam function but there are some differences:
  - mgcv fits smoothing splines only, choosing the smoothing parameter by generalized cross-validation.
  - The gam package uses either loess and/or smoothing splines, and also allows. You can also choose the span or degrees of freedom.
- For both packages, the formula takes the same form as the glm function except now we have the option of having smoothed and parametric terms (e.g., s(variable) will fit a smoothing spline or lo(variable) for loess in the gam package).

```r
> library(gam)
> Prestige.gam<-gam(prestige~s(income)+s(education))
```

Additive regression models: Canadian Prestige example (1)

- The summary function returns tests for each smooth, the degrees of freedom for each smooth, and the AIC for the model. It also gives the deviance.
- Below we see that both income and education have significant effects on prestige.

```r
> summary(Prestige.gam)
Call: gam(formula = Prestige ~ s(income) + s(education), data = Prestige)

Deviance Residuals:
        Min          1Q      Median          3Q         Max
-10.152767  -1.983016   0.711206   1.871136   2.404626

Coefficients:
                      Value Std.Err  z-value Pr(>|z|)
(Intercept)       -6.623  1.3467  -4.9478  0.0000
s(income)         0.030  0.0050   6.1097  0.0000 ***
s(education)      0.051  0.0060   8.4950  0.0000 ***

Approximate significance of smooth terms:
                      edf Ref.df F-value p-value
s(income)          1.008 1.000  8.3447  0.0000 ***
s(education)       1.001 1.000  9.2293  0.0000 ***
```

Additive regression models: Canadian Prestige example (2)

- Again, as with other nonparametric models, we have no parameters to investigate.
- A plot of the regression surface is then necessary.

```r
> inc<-seq(min(income), max(income), len=25)
> ed<-seq(min(education), max(education), len=25)
> #dividing the range of income and education
> #into 25 values each
> data<-expand.grid(income=inc, education=ed)
> fit.prestige<-matrix(predict(Prestige.gam, data), 25, 25)
> persp(income, education, fit.prestige, theta=45, phi=30, ticktype='detailed',
+ xlab='Income', ylab='Education', zlab='Prestige', expand=2/3,
+ shade=.5)
```

Additive regression models: Canadian Prestige example (3)

- GAM for Prestige model:
- We can see the nonlinear relationship for both education and Income with Prestige but there is no interaction between them—i.e., the slope for income is the same at every value of education.
- We can compare this model to the general nonparametric regression model.
Additive regression models: Canadian Prestige example (4)

- General nonparametric model for Prestige:
- The two models seem to be quite similar, although there are some nuances—particularly in the mid-range of income—that are not picked up by the additive model because they X’s do not interact.

Additive regression models in R: Canadian Prestige data (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, 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.

```
> plot(Prestige.gam, se=T, ask=T)
Make a plot selection (or 0 to exit):
1: plot: s(income)
2: plot: s(education)
```

Interpreting the Effects

- A plot of 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 \( X \).
- Interpreting the scale of the graphs then becomes easy:
  - 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 \( 45+4=49 \).
  - 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=47.3. Therefore the fitted value for an occupation with average income of $10000/year and 10 years of education on average is $47.3+6-5=48.3$

**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 degrees of freedom* are equal to the trace of the smoother matrix (the matrix that projects Y onto Y-hat).
- We extend this to the additive model:

\[
df_j = \text{trace}(S) - 1
\]

1 is subtracted from each $df$ reflecting the constraint that each partial regression function sums to zero (the individual intercept 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:

\[
df_{res} = \sum_{j=1}^{k} df_j + 1
\]

1 is added to the final degrees of freedom to account for the overall constant in the model.
Specifying Degrees of Freedom

- As was the case the degrees of freedom or alternatively the smoothing parameter $\lambda$ can be specified by the researcher.
- Also like smoothing splines, however, **generalized cross-validation** can be used to specify the degrees of freedom.
  - Recall that this finds the smoothing parameter that gives the lowest average mean squared error from the cross-validation samples.
- Cross-validation is implemented using the `mgcv` package in R.

```r
> library(mgcv)
> Prestige.gam2<-gam(prestige~s(income, df=7, fx=TRUE) + s(education))
#We specify the number of df with df=7
#We also need to specify fx=TRUE
#for "fixed" df, otherwise GCV will be used
```

Cautions about Statistical Tests when the $\lambda$ are chosen using GCV

- If the smoothing parameters $\lambda$'s (or equivalently, the degrees of freedom) are chosen using generalized cross-validation (GCV), caution must be used when using an analysis of deviance.
  - By implication, the degrees of freedom also changes 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 $\lambda$.
- As a result, it is advisable to fix the degrees of freedom when comparing 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.

```r
> #must specify "test="F" for F-tests
> anova(Prestige.ola, 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 95.0000 5272.4
2 90.9158 4258.7 4.0842 1013.7 5.2986 0.0006414 ***
```

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

Diagnostic Plots

The `gam.check` function returns four diagnostic plots:

1. A quantile-comparison plot of the residuals allows us to look for outliers and heavy tails.
2. Residuals versus linear predictors (simply observed $y$ for continuous variables) helps detect nonconstant error variance.
3. Histogram of the residuals are good for detecting nonnormality.
4. Response versus fitted values.

```r
> gam.check(Prestige.gam)
Smoothing parameter selection converged after 4 iterations.
The RMS GCV score gradient at convergence was 3.180205e-06.
The Hessian was positive definite.
The estimated model rank was 19 (maximum possible: 19)
```
Interactions between Smoothed Terms

- The \texttt{gam} function in the \texttt{mgcv} package allows you to specify an interaction term between two or more terms.

- In the case of an interaction between two terms, and when no other variables are included in the model, we essentially have a multiple nonparametric regression

\[ Y = f_1(X_2, X_3) \]

- Once again we need to graph the relationship in a \textit{perspective plot}.

- While it is possible to fit a higher order interaction, once we get past the two-way interaction the graph no longer can be interpreted.

```r
# In R:
gam(prestige~s(income, education))
```

Generalized Additive Models (1)

- \textit{Generalized nonparametric models} and \textit{generalized additive models (GAM)} follow the GLM by replacing $Y$ with a linear predictor $\eta$ and by including the appropriate link function.

- In other words, like GLM’s these models can be fit to data when the outcome variable is not continuous.

  - logit and probit for binomial response; Poisson for count data; Gamma models, etc.

- These models are then simply extensions of the General additive model to outcomes with several different types of distributions.

- These are fit in R using the same \texttt{gam} function, but now the family must be specified (exactly as in the \texttt{glm} case).

GAM using R: Logit model

Tilley’s political generations data (1)


  - I use a subset of the data, examining the probability of voting Conservative in 1992 based on age and cohort (generation).

- The R-script is as follows. Notice that since it is a logit model, I specify \texttt{family=binomial}. I also use loess smooths (\texttt{lo()}) rather than smoothing splines:

```r
Tilley<-read.table(‘c:/teaching/Haifa/Tilley.txt’, header=T) attach(Tilley)
CONID2.gam <- gam(CONID2 ~ lo(AGE) + lo(GEN),family = binomial)
```
Semi-parametric models (1)

- Semi-parametric models are hybrids of the general additive model and linear regression—some terms enter nonparametrically while others enter linearly.
- The semiparametric regression model can be written

\[ Y_i = \alpha + \beta_1 X_{i1} + \cdots + \beta_r X_{i r} + \cdots + f_k(X_{ik}) + \varepsilon_i \]

- Why restrict the functional form of the relationship to be linear when we are using a model that allows it to take on any form?
  - Dummy regressors: smoothing makes sense only when the explanatory variable is quantitative.
  - Test for linearity: Compare the semiparametric model with an additive model to test for a particular linear effect.

Semi-parametric Logit model in R

Tilley’s political generations data (1)

- We could also fit a semiparametric additive model to the Tilley data.
- Here I add `sex` as a predictor.
  - Since it is measured as a dummy regressor (it is not a quantitative variable) it makes no sense to look at a smooth of `sex`.
- For this example, I use the `mgcv` package rather than the `gam` package. The model is fit in the same way with the `gam` function, however:

```r
detach("package:gam")
library(mgcv)
sex <- as.factor(SEX)
CONID2.gam2 <- gam(CONID2 ~ s(AGE) + s(GEN) + sex, 
                   family = binomial)
```

Semi-parametric Logit model in R

Tilley’s political generations data (2)

- The output will now contain both parametric (the coefficient for `sex`) and nonparametric estimates (the degrees of freedom for each of age and generation).

```r
summary(CONID2.gam2)
```

```
Family: binomial  Link function: logit
Formula: CONID2 ~ s(AGE) + s(GEN) + sex

Parametric coefficients:  Estimate  Std. Error   t value  Pr(>|t|)
(Intercept)    -0.4504       0.01952   -23.07 < 2.22e-10
sex2           0.1011       0.02677     3.777   0.0001585

Approximate significance of smooth terms:  edf chisq     p-value
s(AGE)        5.091 27.384   5.251e-05
s(GEN)        8.048 62.993  1.2674e-10
```
Semi-parametric Logit model in R
Tilley’s political generations data (3)

- Although we have a coefficient for sex, we still need to plot the smooths for the other predictors

![Graph showing smooths for different predictors](image)

Vector Generalized Additive Models

- Yee and Wild (1996) extend the class of GAM models to handle multivariate (vector) regression problems
- Suppose that for each observation we have a q-dimensional response vector \( \mathbf{y}=(y_1, \ldots, y_q)^T \) and k independent variables, \( \mathbf{X} \)
  - In other words, we have several \( y \)-values for each observation that we want to estimate simultaneously
- A VGAM is any model for which the conditional distribution of \( \mathbf{y} \) given \( \mathbf{X} \) is of the form:
  \[
  f(y|X) = f^*(y, \eta_1(X), \ldots, \eta_q(X))
  \]
  for some function \( f^* \), where
  \[
  \eta_j(X) = g_{(j)0} + g_{(j)1}(x_1) + \cdots + g_{(j)k}(x_k)
  \]
- Included in this set of models are multinomial logit models and ordered logit models

Vector Generalized Additive Models (2)

- The \texttt{vgam} function in the \texttt{VGAM} package for \texttt{R} fits multivariate models using simultaneous smoothing splines
  - The \texttt{vgam} function will also fit all the models that \texttt{glm} and \texttt{gam} can fit plus many others
- The \texttt{vgam} function takes arguments very similar to the \texttt{gam} function with the exception that the family names are slightly different to avoid conflicting with \texttt{gam}
- The \texttt{VGAM} package has many convenient features, some of which have application outside of generalized additive models
  - For example, it allows you to fit cumulative logit models with or without the proportional odds assumption

(Vector Generalized Additive Models (2) continued)

VGAM Ordered Logit Models
Example: British identity

- The following data are from surveys of British voters asking how “British” they felt on a 5-point Moreno scale
- The goal is to explore the influence of age, class and education on British identity
- If the political generations theory holds, we might expect a curvilinear relationship between identity and age since the Conservative Party is typically more British

```r
> library(VGAM)
> mod.vgam<-vgam(BRITMRE~s(AGE)+CLASS+EDUCAT, cumulative (parallel=T))
> mod.vgam2<-vgam(BRITMRE~AGE+CLASS+EDUCAT, cumulative (parallel=T))
#Test for linearity on logit scale
> 1-pchisq(deviance(mod.vgam2)-deviance(mod.vgam), 3)
[1] 0.03557512
> plot(mod.vgam, which.term=c("s(AGE)"), se=TRUE, lcol="red")
```
VGAM Ordered Logit Models
Example: British identity (2)

- The proportional odds assumption has not been met. No loss, however, because we can simply allow the slopes to be different from one category to the next.

VGAM Ordered Logit Models
Example: British identity (3)

- Also of interest here is whether the model meets the proportional odds (parallel lines) assumption.
- We proceed to test this by fitting a model relaxing the assumption and contrasting it with the original model using an analysis of deviance.

```
> mod.vgam2<-vgam(BRITMORE~s(AGE)+CLASS+EDUCAT,
                   cumulative(parallel=T))
> mod.vgam3<-vgam(BRITMORE~s(AGE)+CLASS+EDUCAT,
                   cumulative(parallel=F))
> #Test for parallel lines is statistically significant
> 1-pchisq(deviance(mod.vgam2)-deviance(mod.vgam3),
          df.residual(mod.vgam2)-df.residual(mod.vgam3))
[1] 0.002072171
```

VGAM Poisson Regression Models

- Recall that Poisson models are appropriate for count data.
- Recall also, that the link function for a Poisson model is the log link:

\[ g(\mu) = \log \mu = \beta_0 + \beta_1 X \]

- This means, then, that a unit increase in X has a multiplicative impact of \( e^B \) on \( \mu \).
  - The mean of Y at x+1 is equal to the mean of Y at x multiplied by \( e^B \).
  - Moreover, if \( B=0 \), the multiplicative effect is 1:
    \[ e^B = e^0 = 1 \]
- Like the ordered logit, \textit{GAM Poisson models} are easily fit in R using the \texttt{VGAM} package.
VGAM Poisson Regression Model Example: Voluntary organizations

Earlier in the course we fit a Poisson model to a subset of the 1981-91 World Values Survey (n=4803). The variables are as follows:
- ASSOC: Number of voluntary associations to which the respondent belongs (ranges from 0-15)
- SEX, AGE
- SES: unskilled, skilled, middle upper
- COUNTRY: Great Britain, Canada, USA

Previously the goal was to assess differences in membership by SES and COUNTRY and whether SES effects are different across countries (i.e., an interaction between SES and COUNTRY)

In the present case, we are interested just in AGE effects controlling for SES and COUNTRY

VGAM Poisson Regression Model Example: Voluntary organizations (2)

We see here that age has no apparent linear effect $e^{-0.0001663} = 1.00016$.

VGAM Poisson Regression Model Example: Voluntary organizations (3)

It seems very likely the effects of age on membership in voluntary organizations is not linear
- In other words, we may have missed an important feature of the data by constraining it to be linear

We thus proceed to fit a VGAM Poisson model with a smooth term for AGE

VGAM Poisson Regression Model Example: Voluntary organizations (3)

Df Npar Df Npar Chisq P(Chi)
s(AGE)  1  3  88.618  0

VGAM Poisson Regression Model Example: Voluntary organizations (3)

> plot(vgam.poiss, which.term=c("s(AGE)"), lcol="red", se=TRUE)
Cautions about GAMs (1)

Concurvity

- The generalized additive model analogue to collinearity in linear models
- **Two possible problems** can arise:
  1. A point or group of points that are common outliers in two or more X’s could cause wild tail behavior
  2. If two X’s are too highly correlated, backfitting may be unable to find a unique curve. In these cases, the initial linear coefficient will be all that is returned
- The graph on previous page is a good example—Here type and income are too closely related—i.e., professional jobs are high paying, blue collar jobs pay less, and thus we find only linear fits where the lines cross
- As is the case with collinearity, there is no solution to concurvity other than reformulating the research question

Cautions about GAMs (2)

Missing Data

- Missing data can be seriously problematic for any model if the missing cases have a **systematic relationship to the response or the X’s**
  - Especially problematic are data that are not **missing at random** (i.e., the pattern of missingness is a function of the response)
  - This problem is even more serious for generalized additive models than for linear regression
- The backfitting algorithm **omits all missing observations—and thus their fitted values are set to 0**—when the partial residuals are smoothed against the predictor
  - Since the fitted curves have a mean of 0, **this amounts to assigning the average fitted value to the missing observations**
  - In other words, it the same as using mean imputation in linear models, thus resulting in bias estimates

Summary and Conclusions

- Although social science theories typically specify some direction of causation, they seldom require that the relationship is linear
  - If our goal is to model that social process as closely as possible (within reason, of course) then why impose the linearity restriction?
- GAMs free us from **linearity by default**, allowing the functional form of a relationship to take on an unspecified shape
  - At the very least, GAMs allow us to test whether linearity is a reasonable assumption—in other words, they are a good diagnostic tool
- The limitation of GAM models, however is is that we need an X that takes on many different values in order for the smoothing to work well