Lectures 2, 3 and 4:
Splines, Smoothers, and generalized additive models

Data Science 2
CS 109b, Stat 121b, AC 209b, E-109b

Mark Glickman Pavlos Protopapas

Reading: James et al., chapter 7.

Outline:

- Polynomial regression and basis functions
- Regression splines
- Smoothers
- Additive and Generalized Additive Models

From linear to non-linear effects:

You have seen in CS 109a models in which the contribution of a predictor is included linearly.

For a quantitative response Y and quantitative predictor x, we can assume

$$Y = \beta_0 + \beta_1 x + \varepsilon$$

with $\varepsilon \sim \mathrm{N}(0, \sigma^2)$.

We have a number of options for including non-linear effects of x if we believe the relationship is not linear.

Some methods you have already seen in CS 109a (e.g., random forests, SVM).

Review – polynomial models in one variable:

(there will be an ultimate point to this review – be patient!)

Create new variables $x^2, x^3, \ldots, x^d$ for some specified d. Then assume

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \varepsilon.$$

Modeling a polynomial function of a predictor is useful

- when the researcher knows based on scientific theory that the true mean function is curvilinear with respect to the predictor.

- as an approximation to a possibly complex non-linear mean function of the predictor.

Comments on polynomial functions of predictors:

- The model
$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \varepsilon$$
is actually a linear regression model. This is because the model is linear in the unknown parameters β_j.
- Can use the same polynomial approach for logistic regression:
$$\text{logit } \Pr(Y = 1) = \log\left(\frac{\Pr(Y = 1)}{\Pr(Y = 0)}\right) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d.$$
- Can use standard model fitting methods (least-squares, maximum likelihood) for polynomial models because of the linearity of the β_j.

Why polynomial models make sense: Taylor's theorem

Suppose $f(x)$ is the true mean function of x, so that
$$\mathrm{E}(Y|x) = f(x).$$

Taylor's theorem says (in essence) that we can approximate $f(x)$ to any degree of accuracy we like with a polynomial approximation, with better accuracy for higher order polynomials.

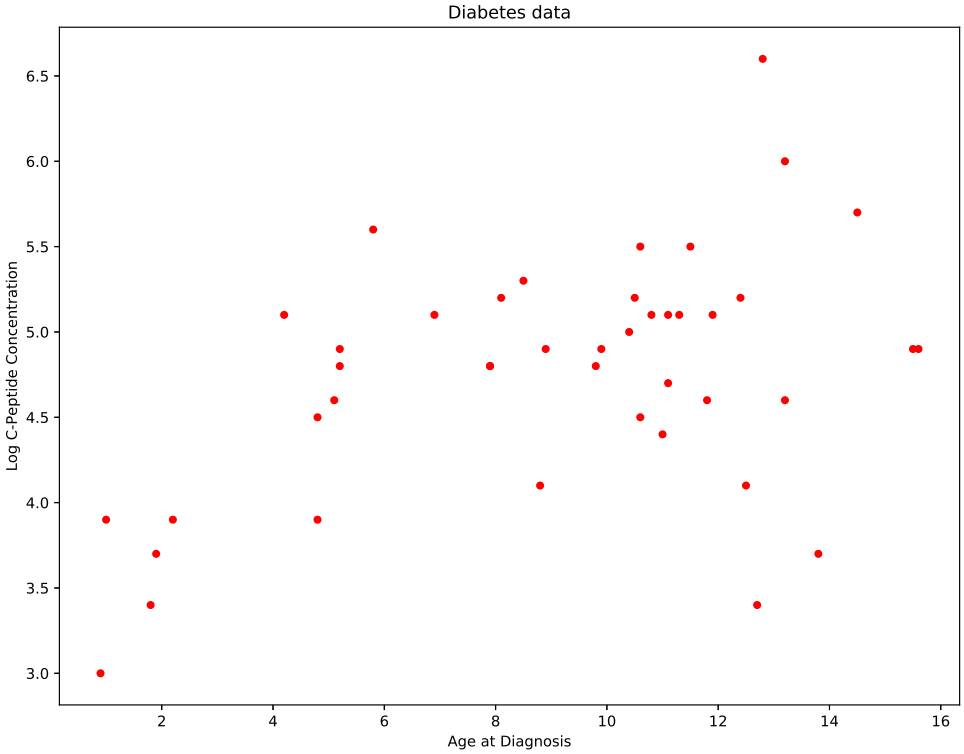
$$\begin{aligned} f(x) &= f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \cdots + \frac{f^{(d)}(0)}{d!}x^d + \cdots \\ &= \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \cdots \end{aligned}$$

Simple motivating example: Diabetes in children

A study from 1987 investigated factors affecting diabetes in children as measured through serum C-peptide levels.

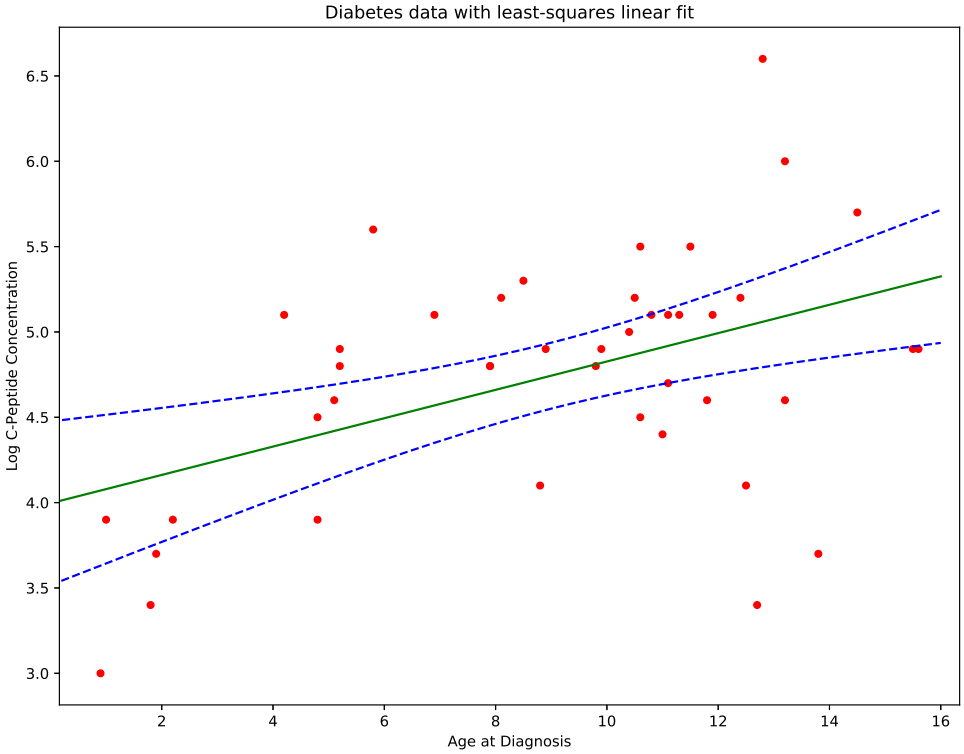
- Response: $Y =$ Log of serum C-peptide level (pmol/ml) in a child
- Predictor: $x =$ Age the child was diagnosed with diabetes

Want to examine the relationship between these two variables on the 43 children in the study.



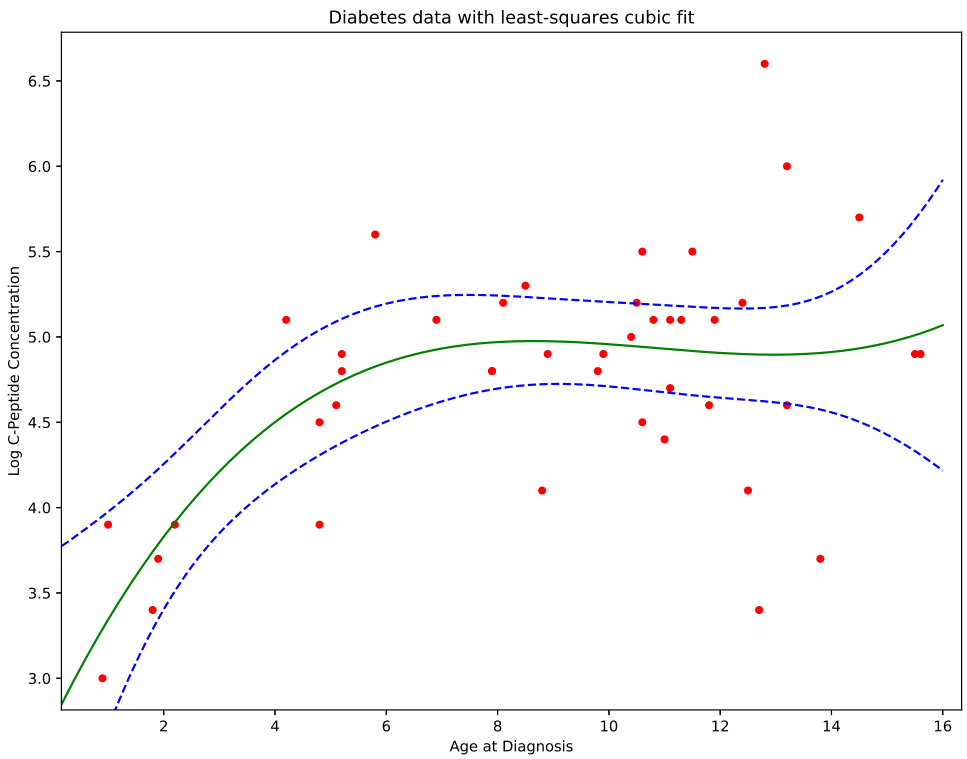
First try: Least squares regression line

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



Second try: Least squares regression cubic polynomial

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \varepsilon_i$$



Can fit logistic regression models as well. Define

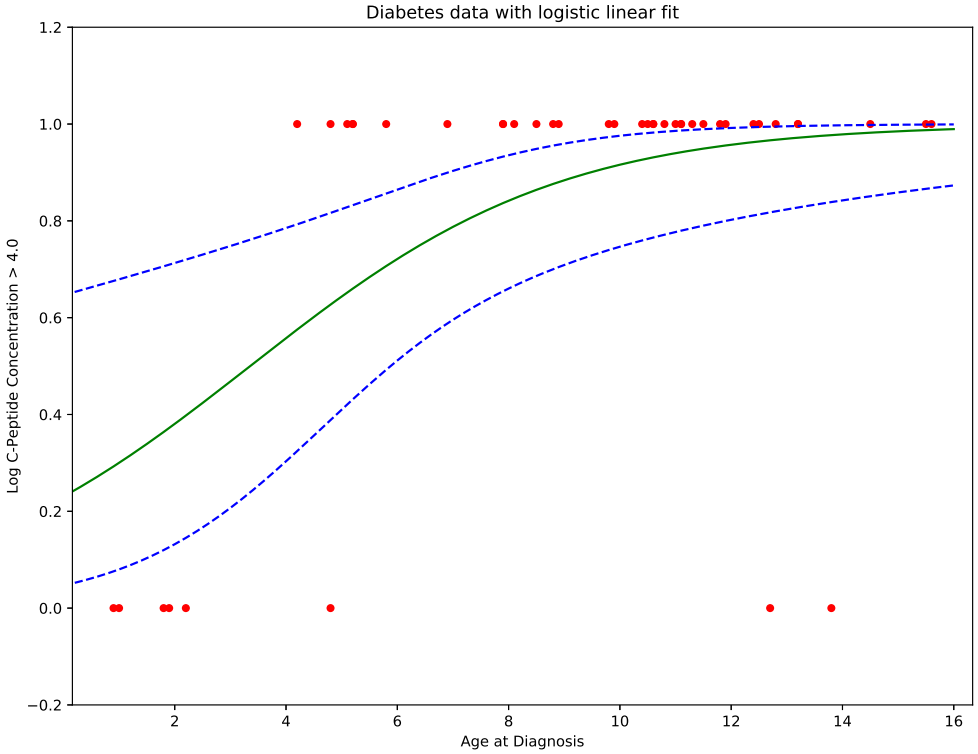
$$Y_i^* = \begin{cases} 1 & \text{if log C-Peptide concentration} > 4.0 \\ 0 & \text{if log C-Peptide concentration} \leq 4.0 \end{cases}$$

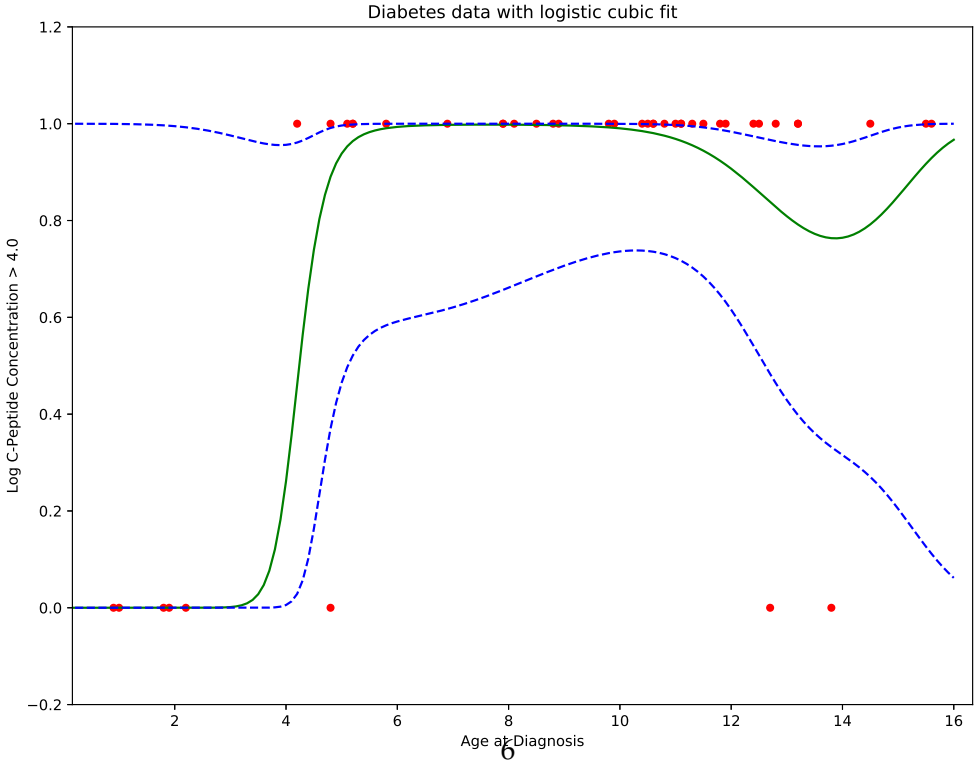
Linear age:

$$\text{logit } \Pr(Y_i^* = 1) = \beta_0 + \beta_1 x_i.$$

Cubic age:

$$\text{logit } \Pr(Y_i^* = 1) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3.$$


Diabetes data with logistic linear fit
Log C-Peptide Concentration > 4.0
Age at Diagnosis
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
2
4
6
8
10
12
14
16


Diabetes data with logistic cubic fit
Log C-Peptide Concentration > 4.0
Age at Diagnosis
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
2
4
6
8
10
12
14
16

Statistical significance of polynomial predictor: Likelihood ratio test

Typical setup for least-squares models

$\mathbf{H}_o$: $Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

$\mathbf{H}_a$: $Y_i = \beta_0 + \beta_1 x_i + \cdots + \beta_d x_i^d + \varepsilon_i$

Typical setup for logistic regression models

$\mathbf{H}_o$: logit $\Pr(Y_i = 1) = \beta_0 + \beta_1 x_i$

$\mathbf{H}_a$: logit $\Pr(Y_i = 1) = \beta_0 + \beta_1 x_i + \cdots + \beta_d x_i^d$

Application to diabetes data:

Least squares (after fitting models):

```
from statsmodels.stats.anova import anova_lm
print(anova_lm(fit1_lm,fit2_lm,test="F"))
```

Output:

```
   df_resid         ssr  df_diff   ss_diff         F   Pr(>F)
0      41.0  17.115527      0.0       NaN       NaN      NaN
1      39.0  13.852548      2.0  3.262979  4.593241  0.01617
```

Significant non-linearity.

Logistic regression: (see python code in notebook)

```
Analysis of Deviance Table

Model 1: r_ybin ~ r_age
Model 2: r_ybin ~ poly(r_age, 3, raw = T)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        41     30.443
2        39     16.684  2   13.759 0.001029 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Very significant non-linearity!

Comments:

- Can include polynomial terms for multiple variables simultaneously in multiple least-squares or logistic regression.
- Choice of d (order of polynomial) - either by scientific considerations, or by cross-validation.
- Actually rare to model unknown curvature using polynomial functions
  - Usually low-order polynomials provide a poor fit
  - Larger order polynomials do not capture local behavior well unless d is very large.
  - Large d can be difficult to interpret.

Deeper look into nonlinear regression:

Goal: : Model $\mathrm{E}(Y|x)$ as a function of a predictor, x, based on a flexible set of choices of functions.

The true model may be

$$\mathrm{E}(Y|x) = f(x)$$

where $f(x)$ is a highly nonlinear function of predictor variable x.

We may be willing to consider as best approximations:

$$\mathrm{E}(Y|x) = \beta_0 + \beta_1 x$$

or

$$\mathrm{E}(Y|x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d.$$

In each case, we are considering linear combinations of component functions.

For linear regression, we have two functions of x:

$$\begin{aligned} f_1(x) &= 1 \\ f_2(x) &= x \end{aligned}$$

such that an approximation to $f(x)$ must be a linear combination of these two functions:

$$\mathrm{E}(Y|x) = c_1 f_1(x) + c_2 f_2(x).$$

Similarly, for polynomial regression, we have

$$\begin{aligned} f_1(x) &= 1 \\ f_2(x) &= x \\ &\vdots \quad\quad \vdots \\ f_{d+1}(x) &= x^d \end{aligned}$$

so that $f(x)$ is approximated by the linear combination

$$\mathrm{E}(Y|x) = c_1 f_1(x) + \cdots + c_d f_d(x).$$

The set of component functions that are part of the approximating linear combination are called a basis, and the functions themselves are the corresponding basis functions.

The span of a basis is the set of all possible functions that can be formed from the linear combination of basis functions.

Examples:

The set $S = \{1, x\}$ is a basis for all linear functions of x.

The set $S = \{1, x, x^2, \ldots, x^d\}$ is a basis for all d-th order polynomials of x.

The point of the discussion of polynomial regression:

- Polynomial regression is arguably the simplest example of creating a basis to approximate a non-linear function.
- Takes advantage of a linear combination representation

$$\mathrm{E}(Y|x) = c_1 f_1(x) + \cdots + c_d f_d(x).$$

  where each of the $f_j(x)$ are individual basis functions, namely $f_j(x) = x^{j-1}$.

Slightly more interesting examples:

The (infinite) set $S = \{1, x, x^2, x^3, \ldots\}$ is a basis for all differentiable functions of x. (by Taylor's theorem)

The (infinite) set

$$S = \{1, \sin(2\pi x), \sin(4\pi x), \sin(6\pi x), \ldots, \cos(2\pi x), \cos(4\pi x), \cos(6\pi x), \ldots\}$$

is a basis for all square-integrable functions of x, for $0 < x < 1$.

Orthogonal polynomial basis:

We have already seen the use of a polynomial basis $\{1, x, x^2, \ldots, x^d\}$.

One issue that can arise in a linear models setting is that terms like x and x^2 can be highly correlated for observed predictor data.

Example: Suppose we have a data set of $n = 5$ values, and for predictor x we observe

$$x = (0.98, 0.99, 1.0, 1.01, 1.02).$$

Then the values of x^2 are

$$x^2 = (0.9604, 0.9801, 1.0, 1.0201, 1.0404).$$

The correlation between the vectors x and x^2 is 0.9999825.

Problem: Having highly correlated predictors can

- result in numerical instability in fitting the model
- produce meaningless (highly inflated) coefficient estimates

A solution: Instead of using $\{1, x, x^2\}$ as the basis, use a basis of polynomials that are uncorrelated on the data.

For the 5-observation example, let

$$S = \{1,\ (x-1)/0.03152,\ -0.5345 + 2672.61(x-1)^2\}$$

This basis is an example of an orthogonal polynomial basis.

Notice

- This basis is comprised of a constant, a linear function of x, and a quadratic function of x. It forms a basis for all quadratic functions just like $\{1, x, x^2\}$.
- The linear and quadratic basis functions for the five observations evaluates to

  $$(x-1)/0.03152 = (-0.632, -0.316, 0.000, 0.316, 0.632)$$

  and

  $$-0.5345 + 2672.61(x-1)^2 = (0.535, -0.267, -0.535, -0.267, 0.535)$$

The correlation between these two variables is 0. Also, the correlation between each and a column of 1s is also 0.

The values are normalized so that the sum of the squared elements is 1.

Back to polynomial regression:

Polynomial regression is actually rarely used in practice unless a scientific theory dictates the use of polynomials.

- Low-order polynomials are an inferior solution to other existing ways to acknowledge non-linearity.
- Increasing the order of the polynomial usually does not help particularly because of odd behavior near the extremes of the data.

A more flexible approach is to use piecewise polynomials.

In particular, use connected piecewise polynomials, also known as splines.

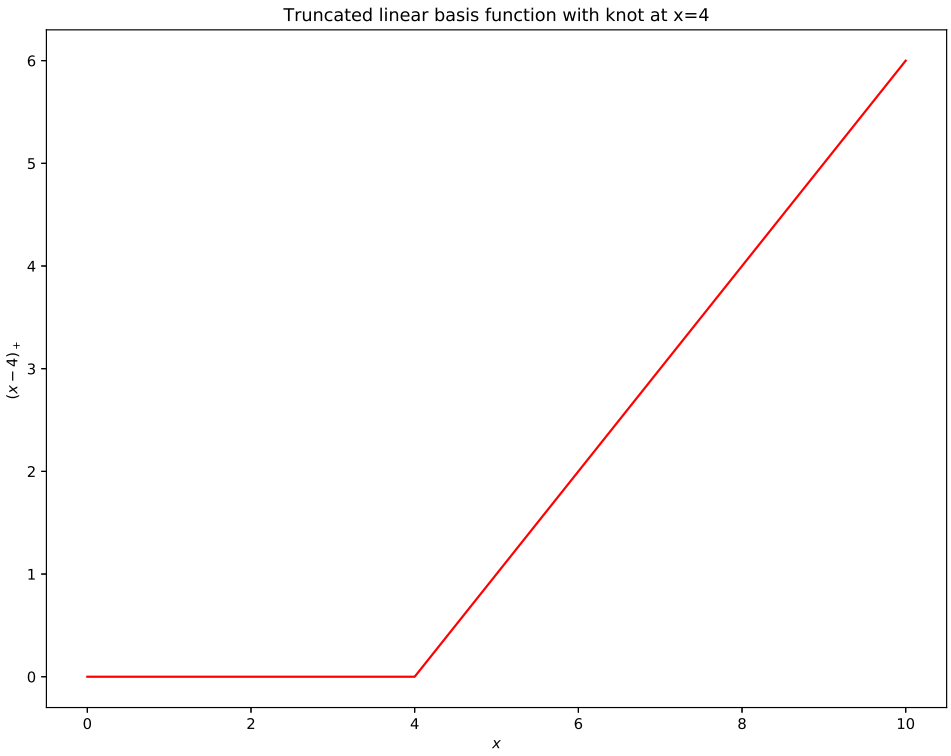
Example: Piecewise linear spline

Define the function

$$(x-\xi)_+ = \begin{cases} x-\xi & \text{if } x > \xi \\ 0 & \text{otherwise.} \end{cases}$$

This function is flat to the left of ξ, and linear (with slope 1) to the right. The value ξ is called a "knot" of the function.

The function $(x-\xi)_+$, a *truncated linear function*, will be a building block for constructing splines.



Connection to ReLUs:

ReLU is short for "Rectified Linear Unit."

A ReLU is a truncated linear function evaluated at 0, that is,

$$x_+ = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

These functions are an essential building block for deep neural nets, as you started seeing last semester.

Strategy for a piecewise linear spline construction:

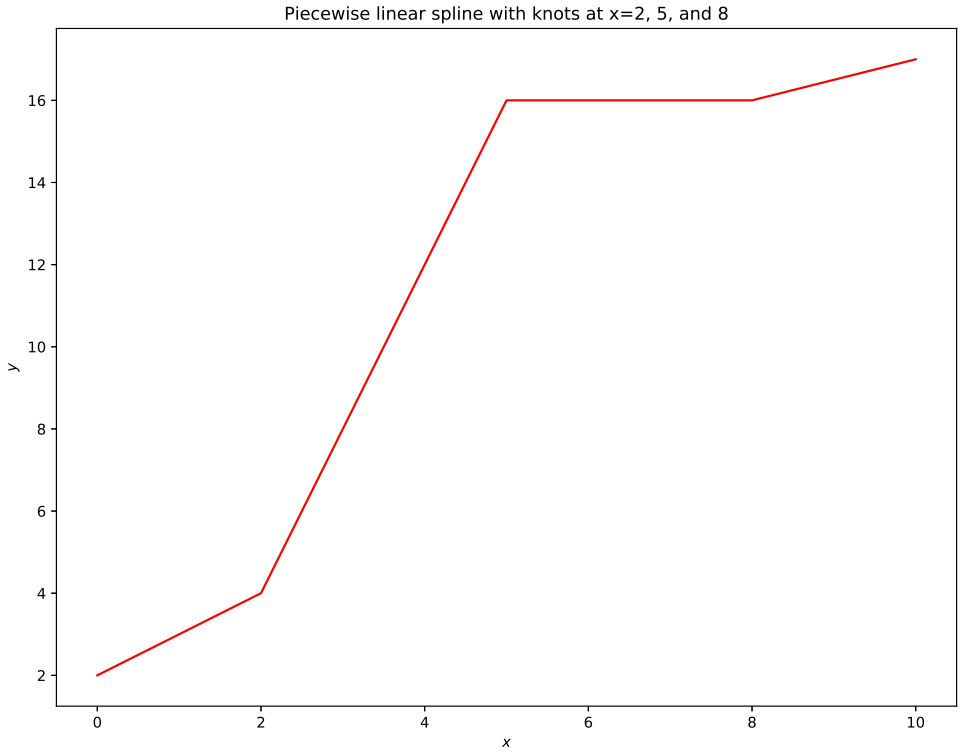
Choose knots $\xi_1 < \xi_2, \cdots < \xi_K$

Now let

$$\mathrm{E}(Y|x) = (\alpha_0 + \alpha_1 x) + \{\beta_1(x-\xi_1)_+ + \beta_2(x-\xi_2)_+ + \cdots + \beta_K(x-\xi_K)_+\}\,.$$

Can think of this construction as

- Start with a linear function $(\alpha_0 + \alpha_1 x)$ to the left of ξ_1.
- At each knot ξ_k, change the slope of the line by an additive amount β_k.



Spline basis functions:

Given knots $\xi_1, \ldots, \xi_K$, any linear spline can be composed from the following basis functions:

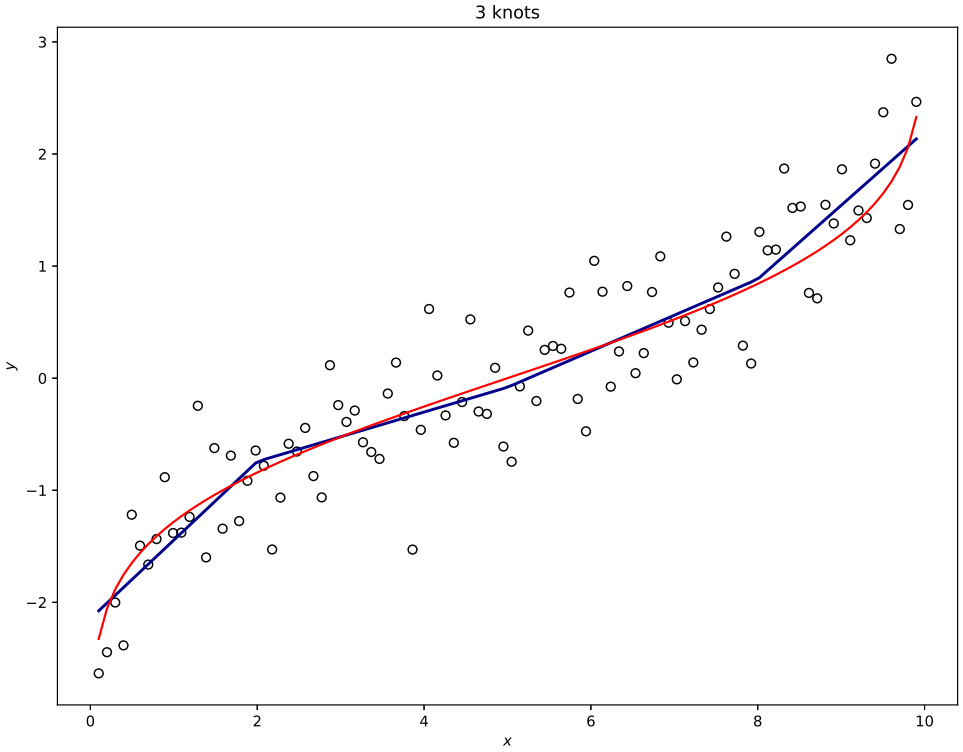
$$S = \{1, x,\ (x-\xi_1)_+, (x-\xi_2)_+, \ldots, (x-\xi_K)_+\}$$

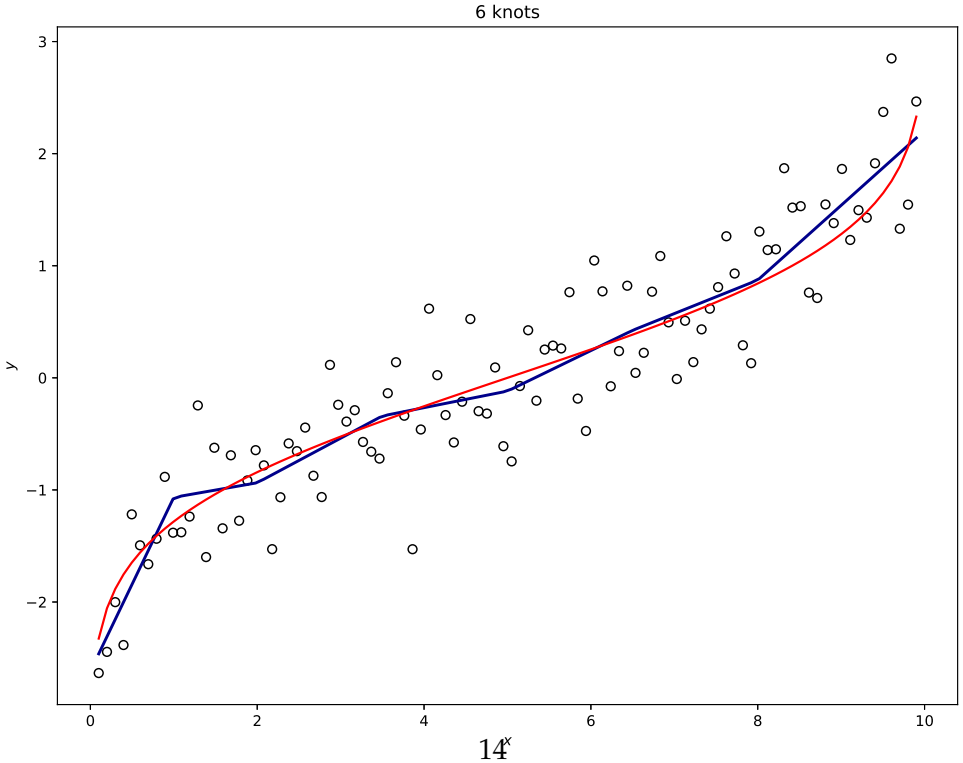
That is, S forms a basis for linear splines.

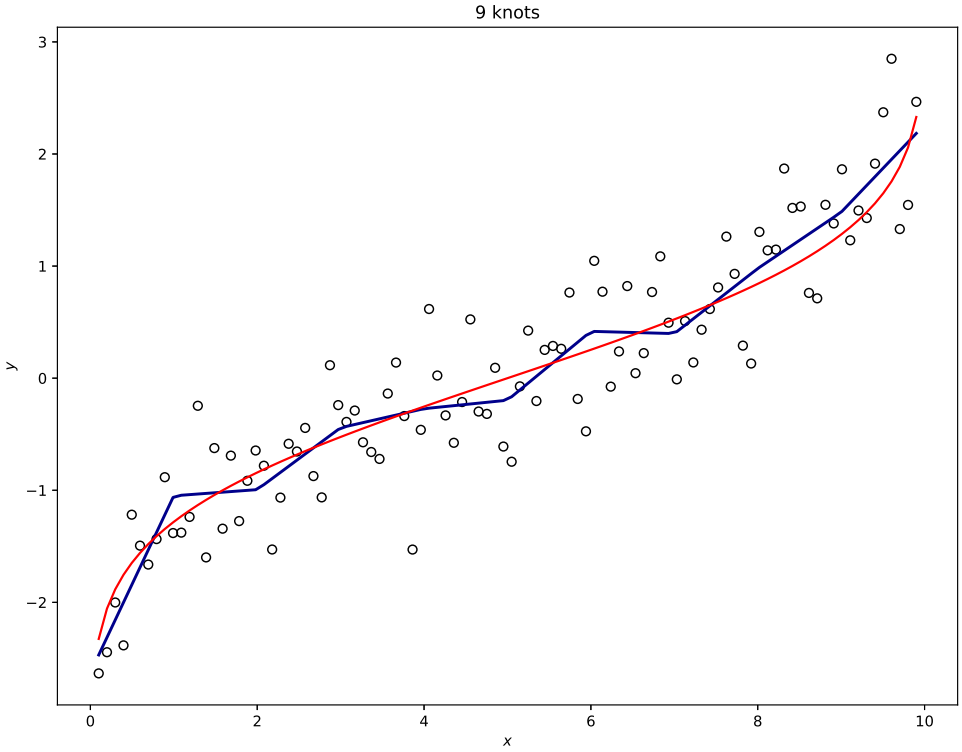
Increasing the number of knots

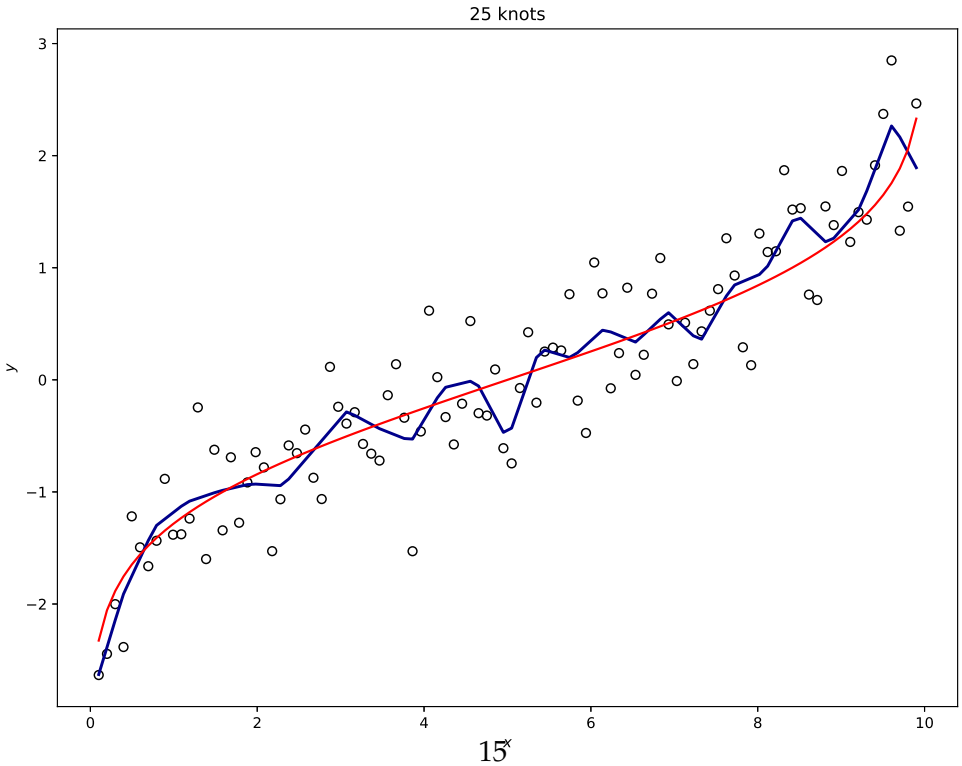
- increases the number of basis function components, and

- increases the scope of functions representable by a linear spline


3 knots
y
x
3
2
1
0
−1
−2
0
2
4
6
8
10


6 knots
y
x
3
2
1
0
−1
−2
0
2
4
6
8
10


9 knots
y
x
3
2
1
0
−1
−2
0
2
4
6
8
10


25 knots
y
x
3
2
1
0
−1
−2
0
2
4
6
8
10

Cubic splines:

Linear splines are okay, but

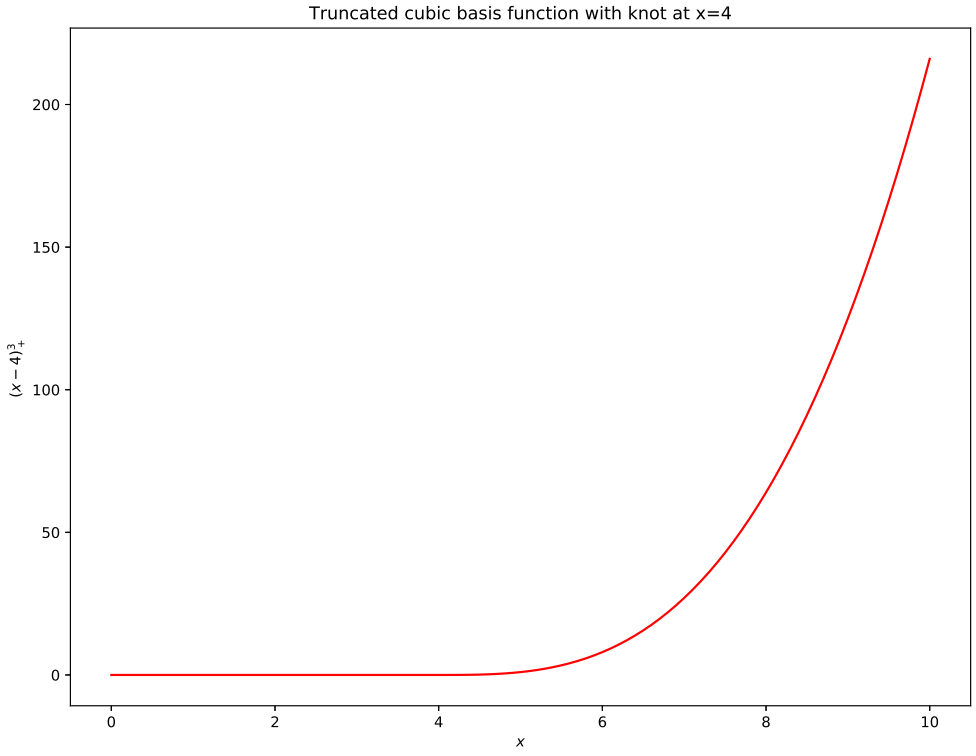
- they are not smooth
- they do not perform well detecting highly curved relationships (unless using a lot of knots)

Much more common in practice to use cubic splines.

The basis function that will help to develop cubic splines is

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

This is called a truncated cubic function.



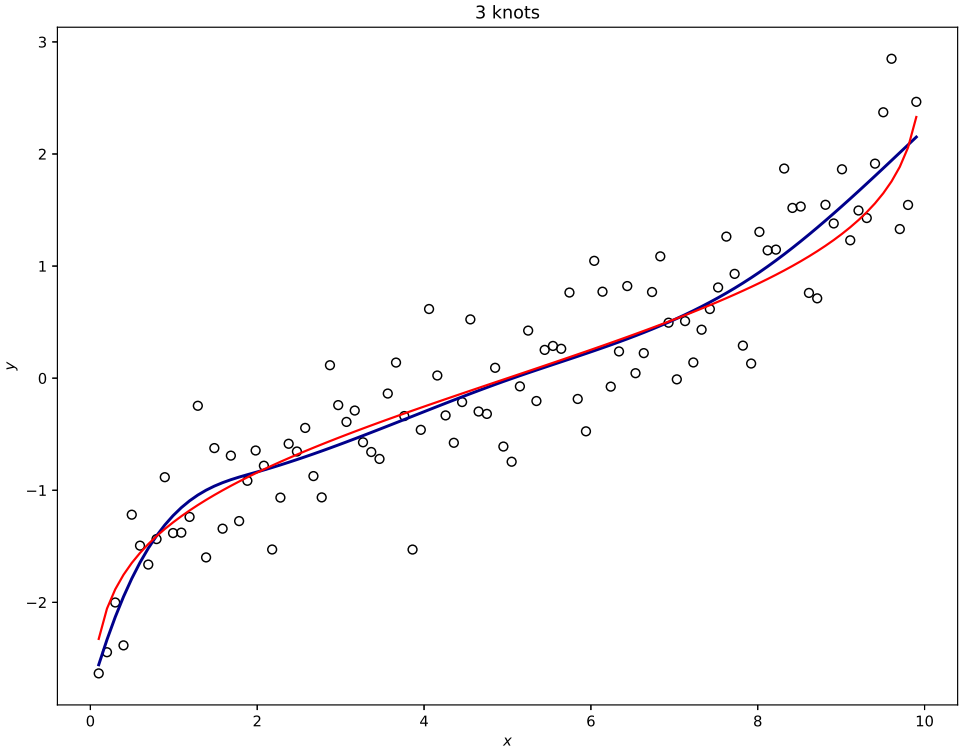
A cubic spline with K knots $\xi_1, \ldots, \xi_K$ to express the mean function of Y is given by

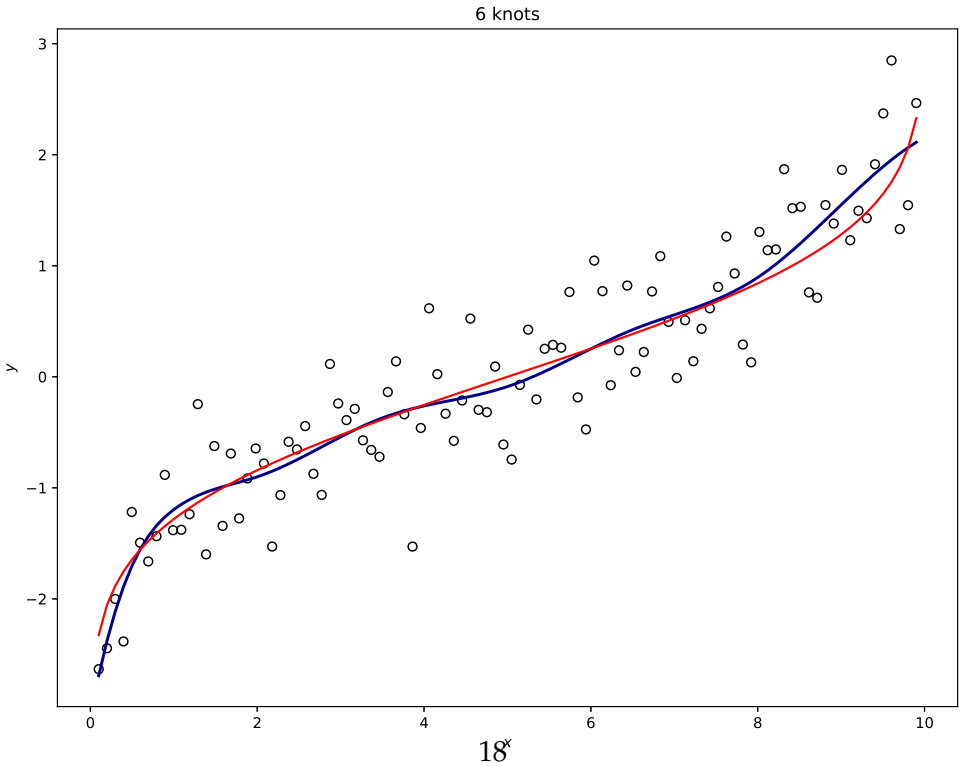
$$f_{cs}(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 + \\ \{\beta_1 (x-\xi_1)_+^3 + \beta_2 (x-\xi_2)_+^3 + \cdots + \beta_K (x-\xi_K)_+^3\}$$

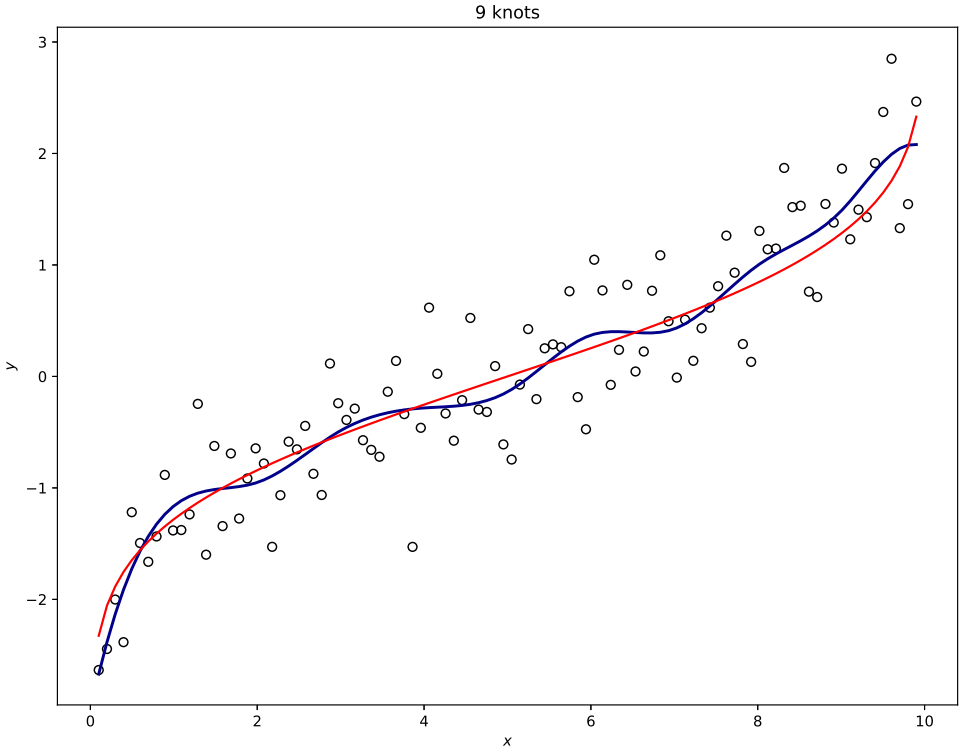
Cubic spline basis: $S = \{1, x, x^2, x^3, (x - \xi_1)^3_+, \ldots, (x - \xi_K)^3_+\}$

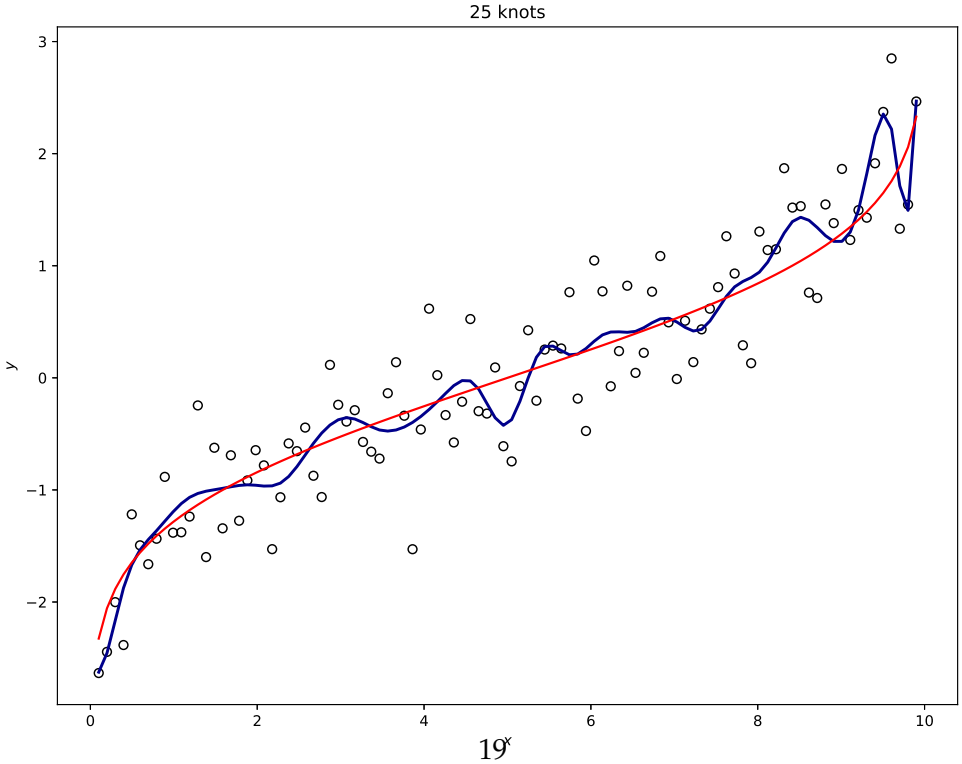
For quantitative outcomes, can perform least-squares regression on

$$y_i = f_{cs}(x_i) + \varepsilon_i$$


3 knots
y
x
3
2
1
0
−1
−2
0
2
4
6
8
10


6 knots
y
x
3
2
1
0
−1
−2
0
2
4
6
8
10


9 knots
3
2
1
0
−1
−2
y
0
2
4
6
8
10
x


25 knots
3
2
1
0
−1
−2
y
0
2
4
6
8
10
x

Important features of cubic spline:

- In between knots ξ_k and ξ_{k+1} the spline as a function of x is a cubic polynomial. This is because the spline value is a sum of cubic polynomials (the "base" polynomial plus the truncated cubic terms to the left of ξ_k).
- At each knot, the spline is continuous, has a continuous first derivative, and has a continuous second derivative. This means that the spline is smooth (to the eye).
- A cubic spline, without any extra constraints, has $K+4$ parameters that need to be estimated.

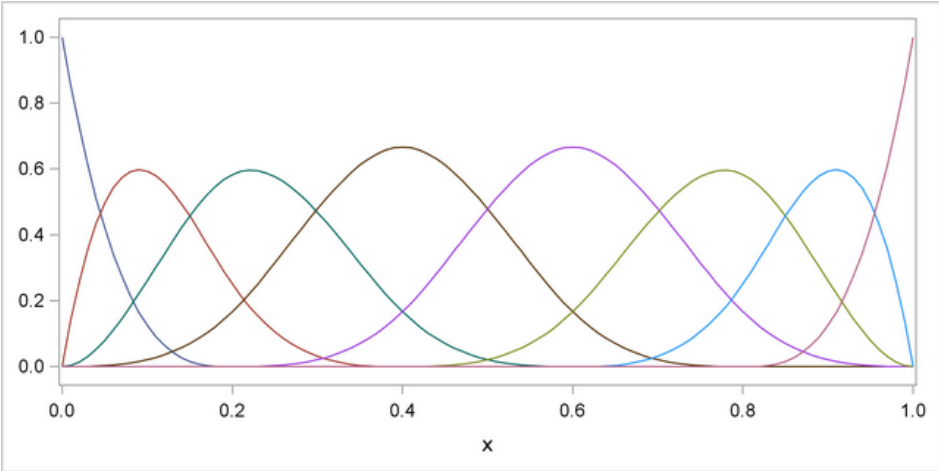
Related comments to cubic splines:

- Instead of using the truncated cubic basis functions, it is more common in software packages to use a "B-spline" basis.

  This basis spans the same as the truncated cubic basis (and has $K+4$ basis functions), but is more efficient in estimation because at most five basis functions are involved in contributing towards any function value. The basis functions are defined recursively given the knots - difficult to write out.
- Because the behavior of cubic splines near boundaries can behave strangely, a variant is "natural" cubic splines. This basis enforces that the function is linear outside the extreme knots.

  This linearity requirement creates 4 constraints (changing a cubic polynomial to linear at each extreme), so with K knots the total number of basis functions is K.

B-spline basis:

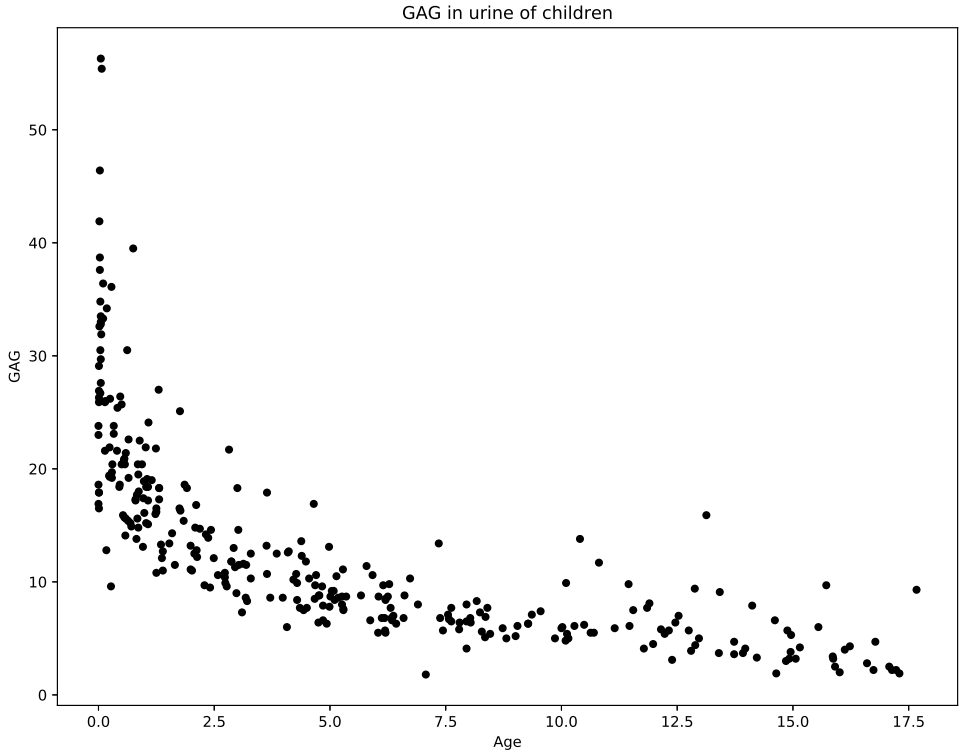


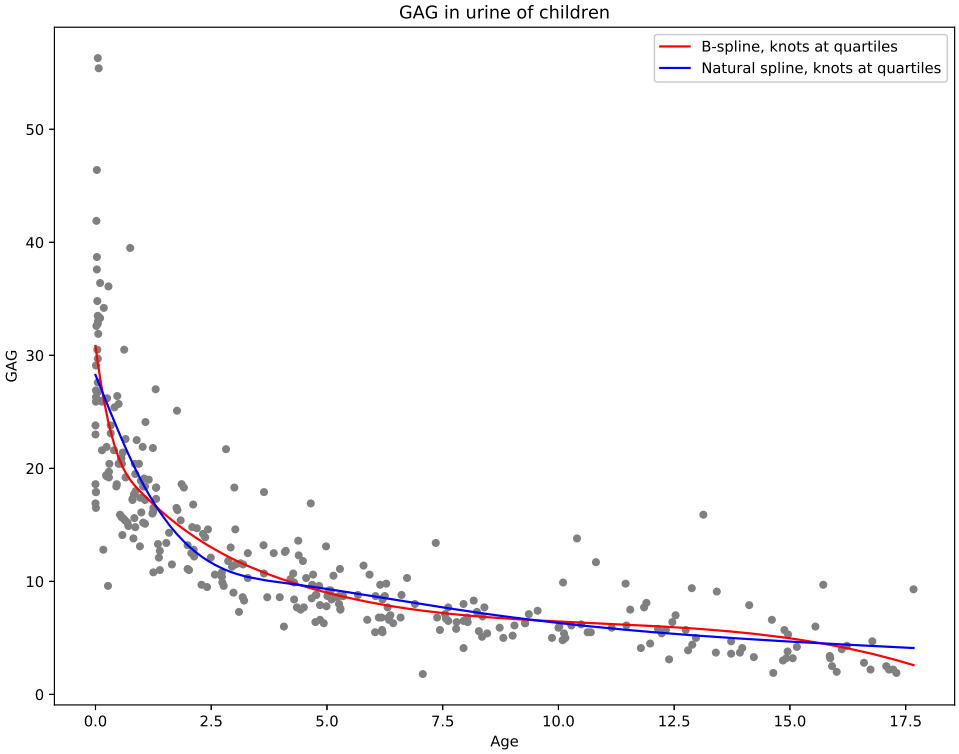
Example:

Data were collected on the concentration of a chemical GAG in the urine of 314 children aged 0 to 17.7 years old.

The response variable is GAG which ranges from 1.8 to 56.3.

As we will see, the B-spline and natural spline are similar.





Choosing the number and position of knots:

- Can place knots at quantiles of the predictor variable, or at regularly spaced intervals
- Choosing the number of knots seems to be more crucial in obtaining a good non-linear fit
- Good idea to place more knots in parts of the predictor data where one might expect the mean function to change rapidly.

Regression splines and multiple predictors:

Because a single predictor transformed as a polynomial term or a spline involves a linear combination of basis functions, it is actually straightforward to extend this approach to multiple predictors.

- Transform each predictor individually to a linear combination of basis functions
- Include all terms simultaneously into the model
  - Fit least-squares multiple regression (if outcome is quantitative)
  - Fit multiple logistic regression (if outcome is binary)

Smoothers and additive models:

So far, we have explored incorporating non-linear functions of predictors through polynomial terms or splines.

Notation:

Let η_i denote the simultaneous contribution of the J predictors $x_{i1}, x_{i2}, \ldots, x_{iJ}$ to the model.

For least-squares regression, we have $\eta_i = \mathrm{E}(Y_i|x)$, and for logistic regression we have $\eta_i = \text{logit}\,\Pr(Y_i = 1)$.

Example simultaneous contribution of predictors:

$$\eta_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_J x_{iJ}$$

for models where each predictor enters linearly.

Another example:

$$\eta_i = \beta_0 + f_1(x_{i1}) + \cdots + f_J(x_{iJ})$$

where $f_j(x_{ij})$ might be a cubic spline for variable x_j.

A model that includes the additive contribution of nonlinearly transformed predictors is called an additive model.

Actually, to be more precise, if the outcome variable is assumed to be normally distributed, then the above is an additive model. If the outcome variable is not normally distributed (e.g., binary), then the above model is called a generalized additive model.

Beyond polynomials and splines: Smoothers

Our approach thus far of identifying a non-linear function of a predictor has relied on

- Polynomial functions of the predictor
- Splines, i.e., piecewise polynomials that are connected at knots

Is there a more general way to view non-linear functions of a predictor?

Smoothers:

A smoother $S(x)$ is a function that is used to estimate some feature of a response as a function of one or more predictors x that is typically less variable than the response.

By convention, we'll denote $s(x)$ to be an estimate of $S(x)$, called a "smooth," based on observed data.

A "scatterplot smoother" is a smoother of just one predictor variable.

The basic intuition is that given a particular values of x, what principles can we apply to estimate the mean of Y?

Usually two decisions to make when choosing/estimating a smoother:

- The type of local averaging of y-values in the neighborhood of x to obtain $S(x)$.
- The size of the neighborhood of values involved in the local average.

We will look at these two issues one at a time.

A few types of smoothers: Different types of local averages

Running-mean (moving average) and running-line smoothers:

- Choose a window length.
- Then for each x, compute the mean of the y_i within the window around x (running-mean), or compute the least-squares estimate at x based on data within the window (running-line).

Kernel smoothers:

Rather than compute averages of y_i within windows around a given x, computed a weighted average of y_i values depending on how far each x_i is from x.

- Specify a "kernel" function, $K(x_i, x)$. This function is non-negative values. For any x, this gives the weight that should be used for point (x_i, y_i) when computing the weighted average of the y_i.
- A kernel function is chosen so that
  - $K(x_i, x)$ is a maximum when $x = x_i$
  - $K(x_i, x)$ goes to 0 as $|x - x_i|$ increases

  Typical choice of K is a normal density (Gaussian) function.
- Compute the weighted average of all y_i using the kernel weights.

Lowess (or loess):

Abbreviation for "locally-weighted scatterplot smoother."

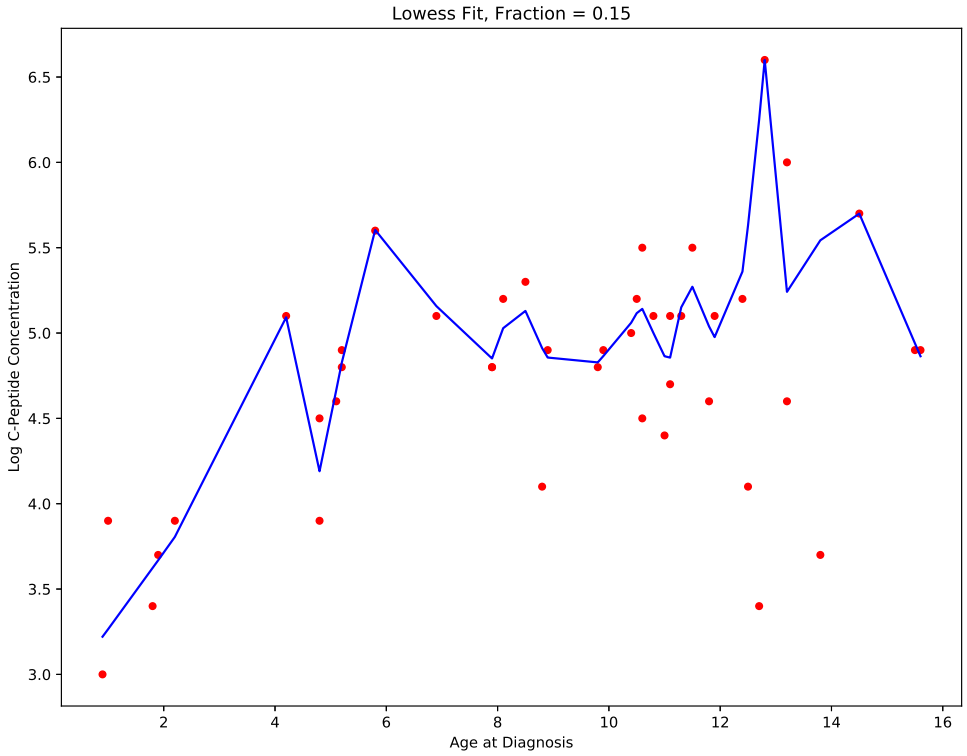
Combines the ideas for running-line smoothers and kernel smoothers.

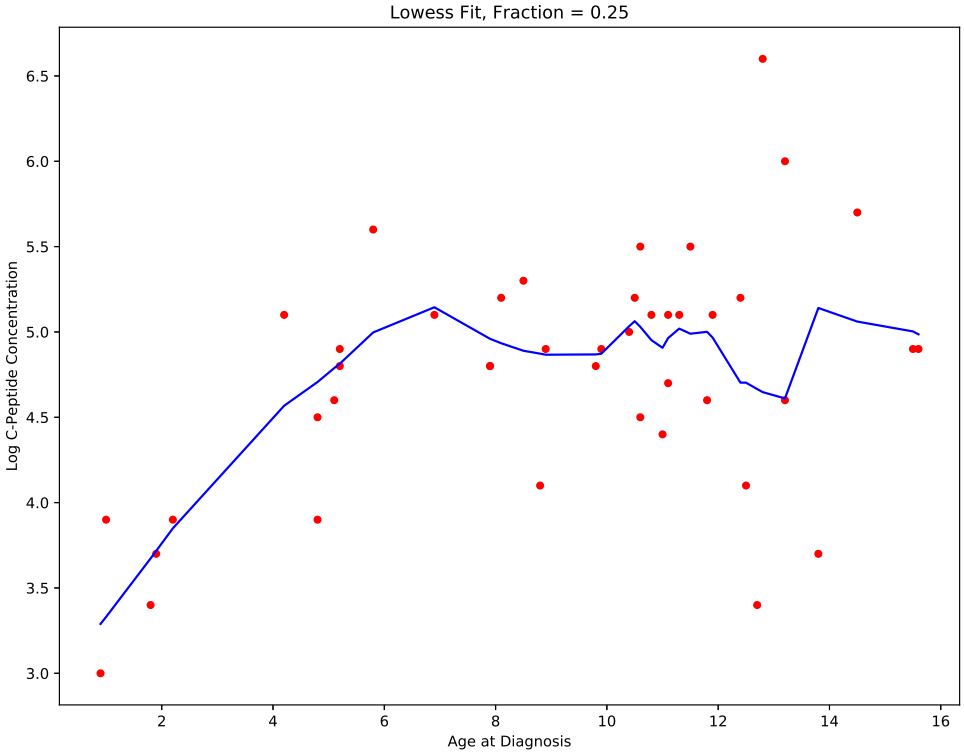
- Specify a kernel function.
- At each x, perform a weighted least-squares regression of all points with kernel weights defined by $K(x_i, x)$.
- The smooth is determined as the fitted value of weighted least-squares regression at each location x.

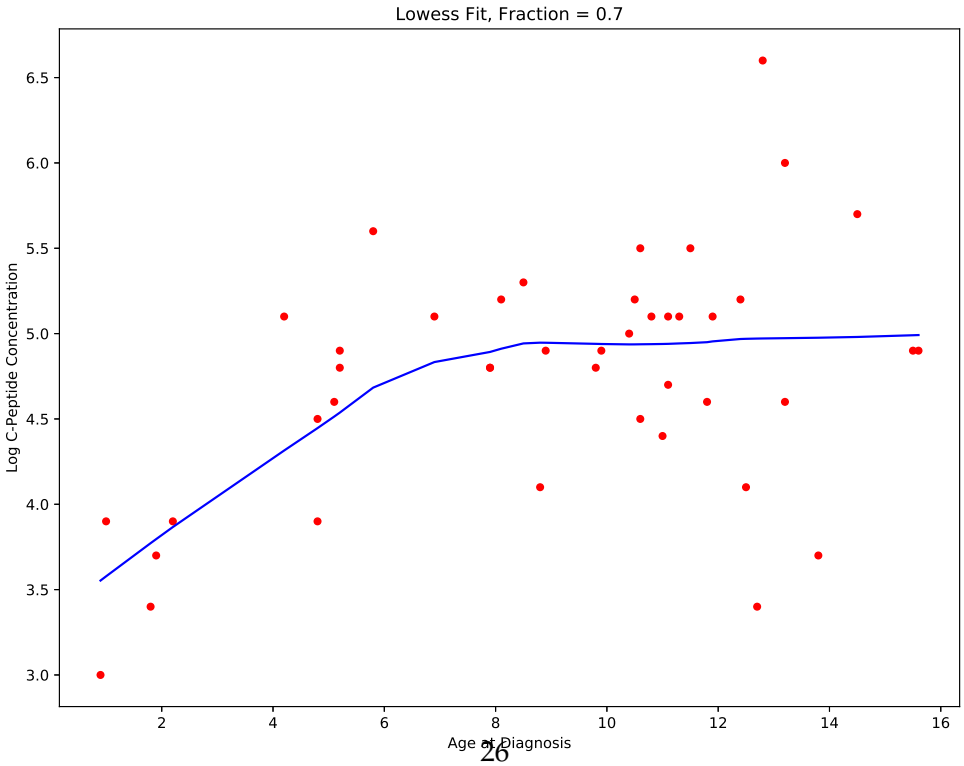
The role of the neighborhood size: Smoothness

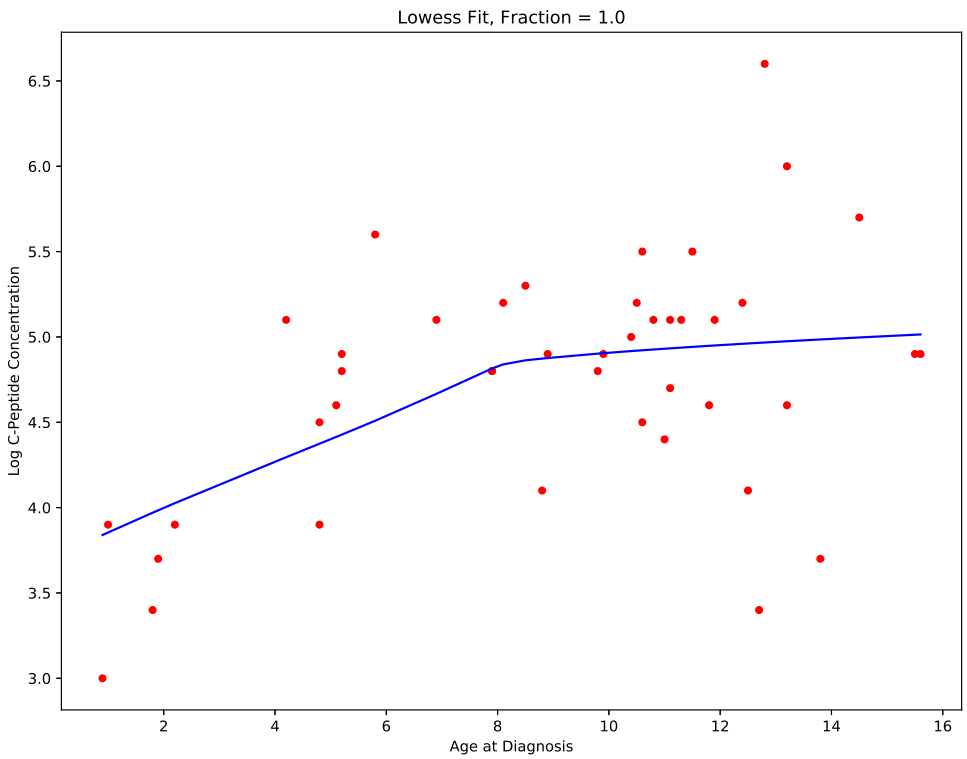
Can see the differences in the effects of neighborhood sizes in different lowess smooths on the diabetes data.

- The larger the neighborhood over which averages of y_i are taken, the smoother the estimated function; the smaller the neighborhood, the more jagged the estimated function.
- How should we choose the neighborhood size/smoothness of a smoother? More on this shortly.




Lowess Fit, Fraction = 0.25
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16


Lowess Fit, Fraction = 0.7
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16



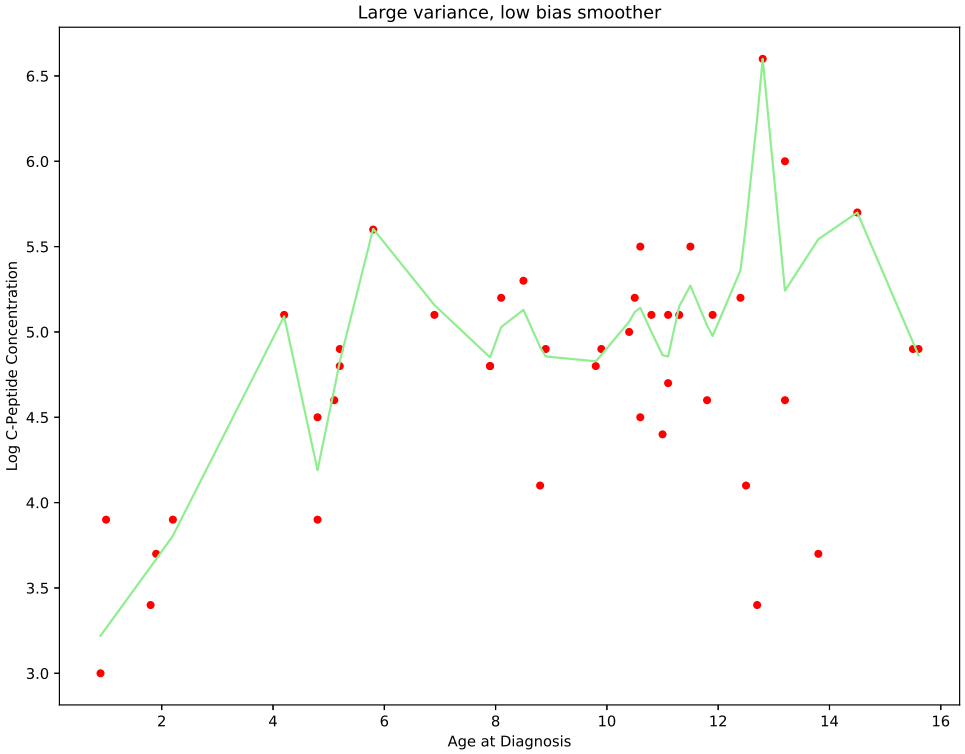
Fundamental issue with choosing smoothness: Bias-Variance Tradeoff

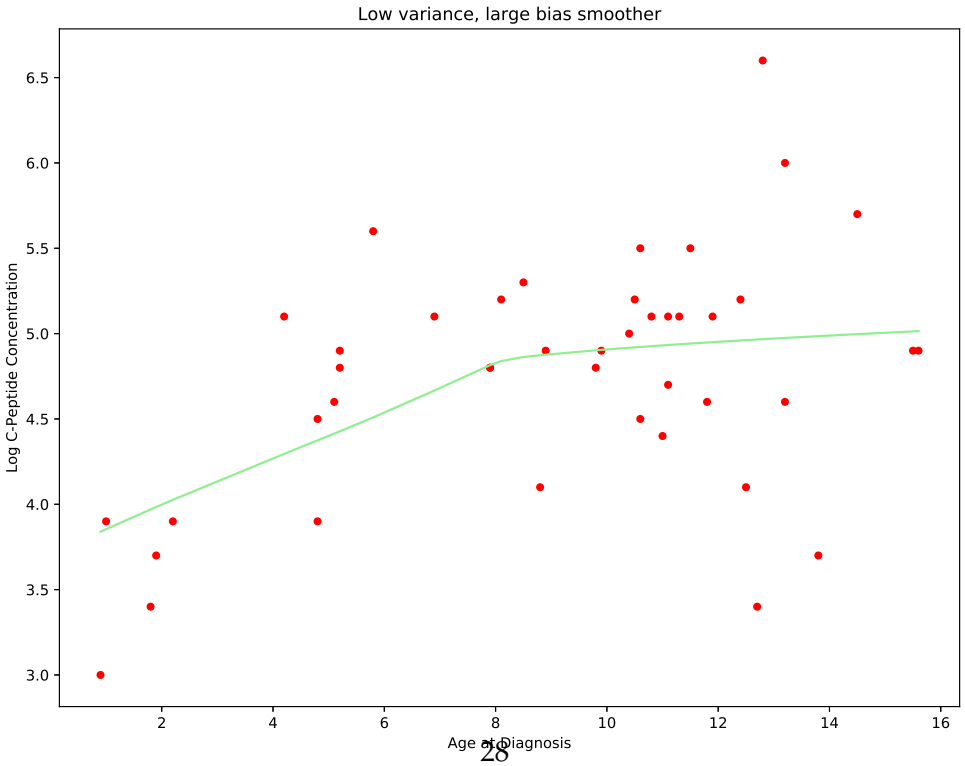
If $\hat{\mu}$ is an estimator of $\mathrm{E}(Y)$, μ, using a particular smoother, then

- Bias: $\mathrm{E}(\hat{\mu}) - \mu$
- Variance: $\mathrm{Var}(\hat{\mu})$

If we use a smoother that is too smooth, there is a risk of large bias for particular $\boldsymbol{x}$ values.

If we use a smoother that is too rough, there is a risk of large variance of the smooth (very sensitive to the data used).


Large variance, low bias smoother
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16


Low variance, large bias smoother
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16

Our main smoother:

Choice of a smoother as an optimization problem:

Among all functions f that are twice-differentiable, find the one that minimizes the penalized sum of squares

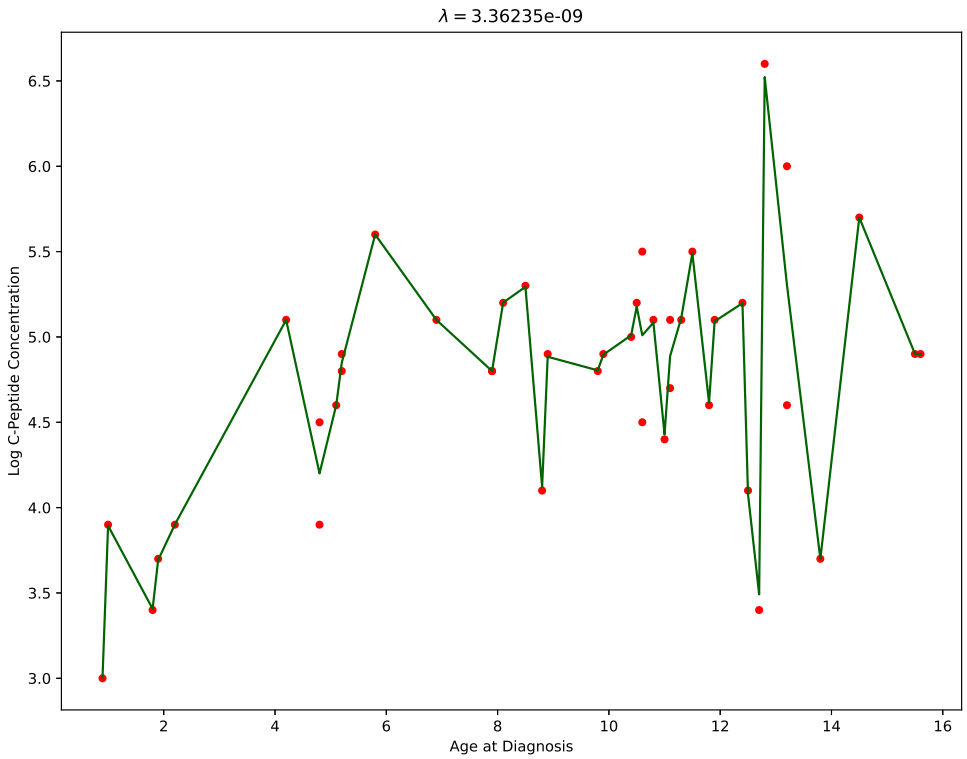
$$\text{PSS}(f \mid \boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_a^b (f''(t))^2 dt$$

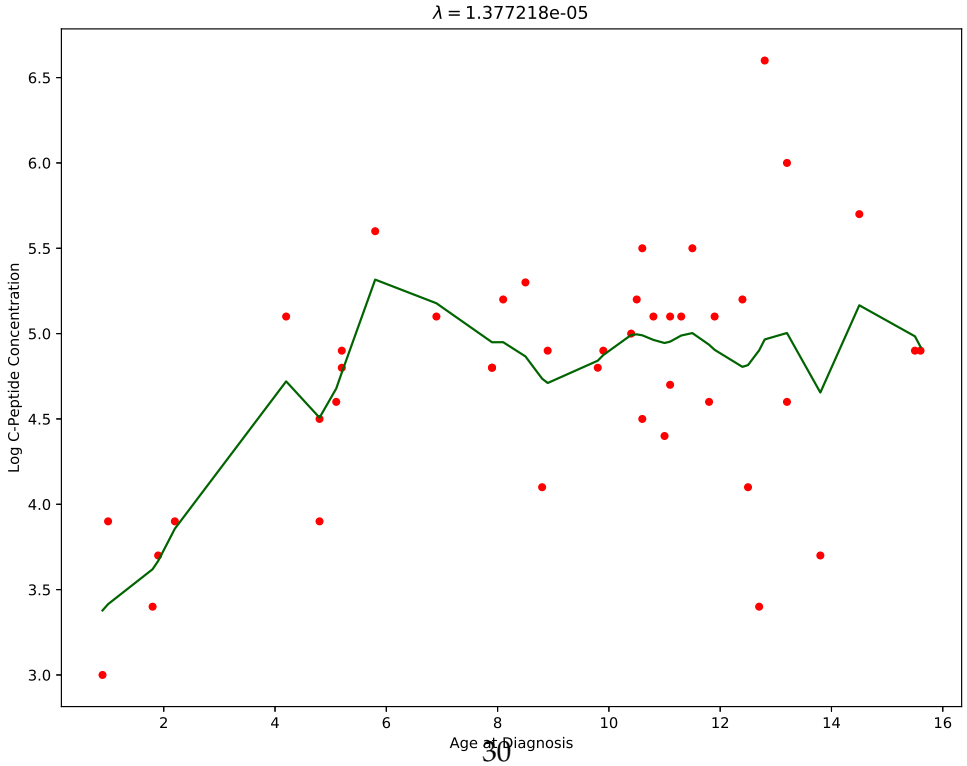
for a given value of $\lambda \geq 0$, where

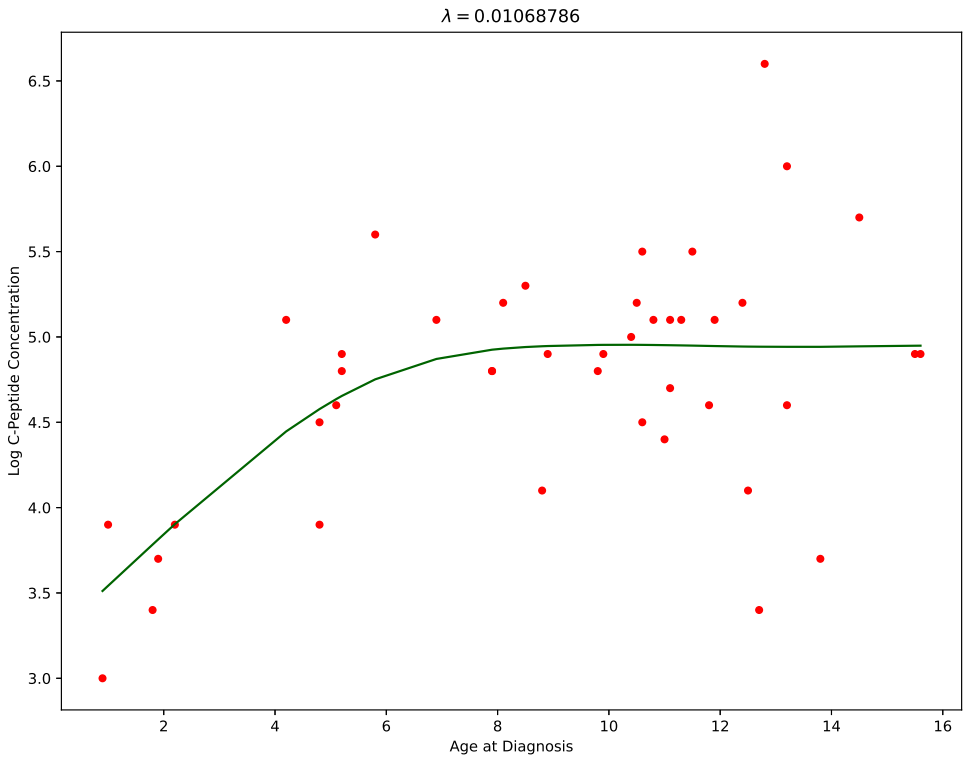
$$a \leq x_1 \leq x_2 \leq \cdots \leq x_n \leq b.$$

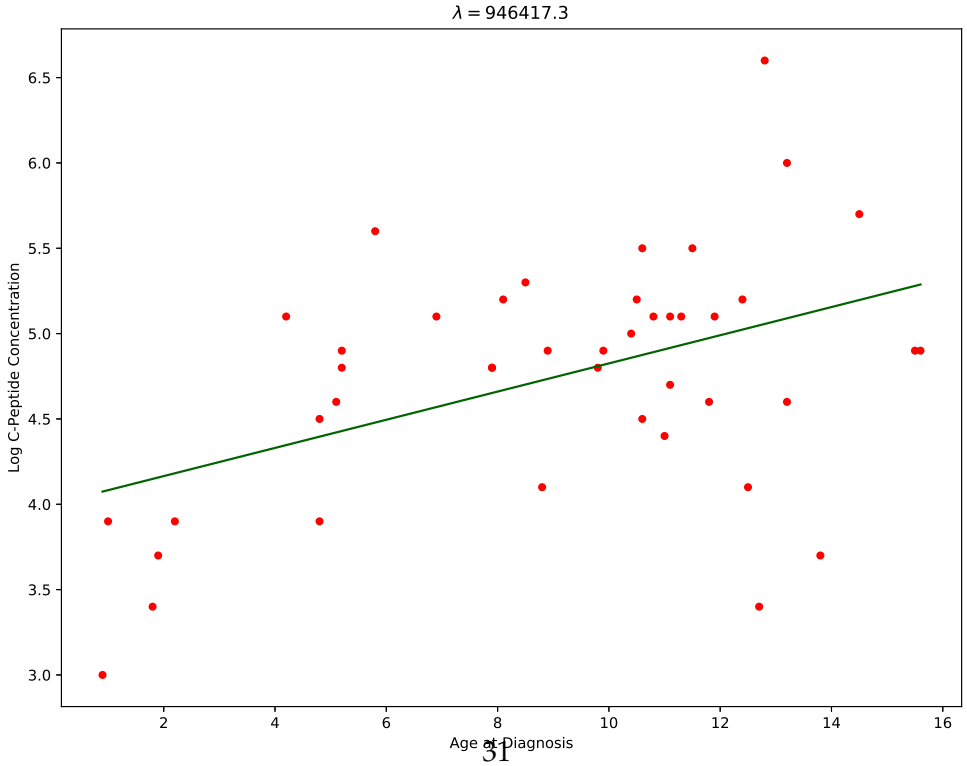
Why is this a reasonable criterion for f:

- The first term measures the lack of fit having chosen f.
- The second term is a penalty for the wiggliness of f.
- The value λ is a "tuning parameter" that balances the trade-off between lack of fit and wiggliness.
  - Small values of λ result in a "connect-the-dots" fit.
  - Large values of λ result in a fit closer to a line.


λ = 3.36235e-09
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16


λ = 1.377218e-05
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16


$\lambda = 0.01068786$
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16


$\lambda = 946417.3$
Log C-Peptide Concentration
Age at Diagnosis
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
2
4
6
8
10
12
14
16

Solution to optimization problem:

For a given λ, the unique f that minimizes the penalized sum of squares turns out to be a cubic spline with knots at each x_i. As a reminder:

- A (different) cubic polynomial is fit between each x_i and x_{i+1} (not necessarily connecting to (x_i, y_i) or (x_{i+1}, y_{i+1})).
- The smoothing spline is continuous at each x_i.
- Not only do the cubic polynomials connect at each x_i, but the first and second derivatives of the cubic polynomials connecting at each x_i are equal. This is equivalent to saying that the smoothing spline is twice differentiable (even at the x_i).
- Conventionally we use assume a natural cubic spline: The smoothing spline is linear to the left of x_1 and to the right of x_n.

Actually, this is a special type of cubic spline called a "cubic smoothing spline."

- Without further discussion, the cubic spline is over-parameterized. If the knots are at every x_i, there is no information to choose from among different cubic polynomials between x_i and x_{i+1}.
- The parameter λ is what makes the connecting cubic polynomial choices unique.

Once λ is specified, determining the smooth is just fitting a cubic spline.

Choosing the smoothing parameter/neighborhood size: Cross-validation (CV)

As you saw in CS109a, this is a very important concept in machine learning, and non-model based estimation.

Review: Choose a smoothing parameter that produces the best predictions on data not analyzed.

For our setup when we observed pairs (x_i, y_i), a reasonable CV criterion is

$$\text{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - s_\lambda^{(i)}(x_i))^2$$

where $s_\lambda^{(i)}$ is the smooth estimated from all the data not including observation i. Choose λ to minimize this expression.

Why this idea is appealing:

- The procedure automates the choice of λ (or of the neighborhood size).
- CV essentially prevents "overfitting" or "oversmoothing."
- The mean of $\text{CV}(\lambda)$ is expected to be close to the true average predictive squared error.

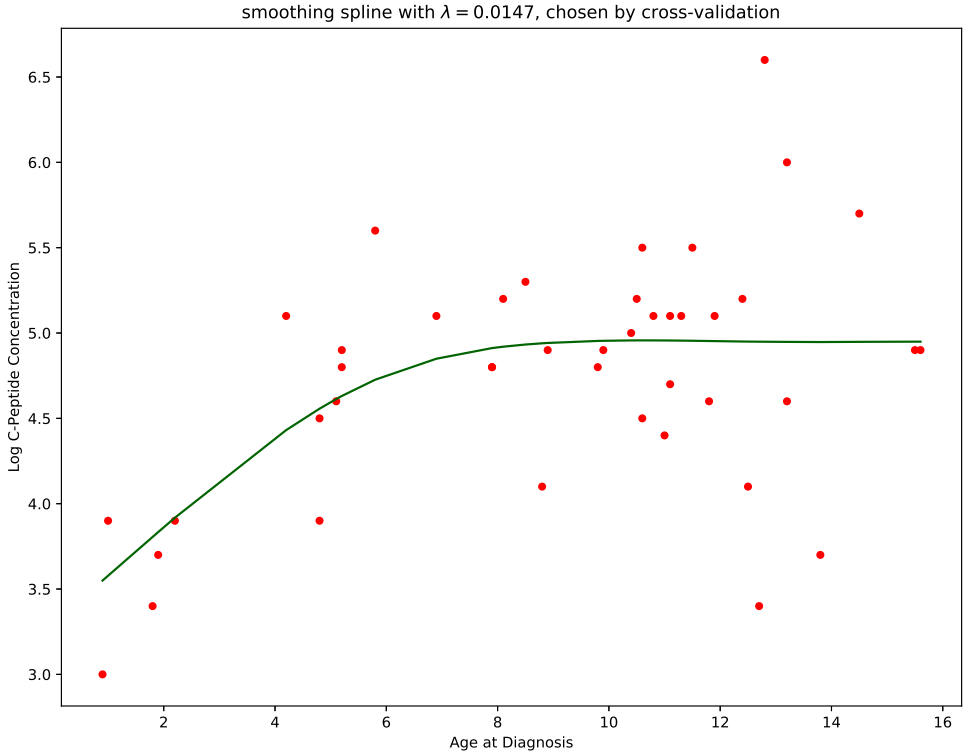
Worth pointing out that other CV strategies exist that leave out more than one observation at a time, such as K-fold CV and Monte Carlo CV.

A fortunate result:

For leave-one-out CV, it appears that we need to refit the smooth n times (once per left-out observation). And that's for each candidate choice of λ!

Fortunately, CV(λ) can be mathematically shown to be a function of terms that involve the fit of the full sample of data (see page 279 in the textbook).

Thus for each λ, we only need to fit the cubic smoothing spline once to compute a cross-validation measure of lack of fit.



Extending smoothers to multiple predictors:

- Additive models: $y_i = \beta_0 + S_1(x_{i1}) + \cdots + S_J(x_{iJ}) + \varepsilon_i$
- Generalized additive models (GAMs): Assume the Y_i have a distribution such as binomial, with $\mu_i = \mathrm{E}(Y_i|x)$, with the additive predictor

$$\text{logit}\,\Pr(Y_i = 1) = \eta_i = \beta_0 + S_1(x_{i1}) + \cdots + S_J(x_{iJ})$$

Additive models are a special case of GAMs. Also, logistic regression is a special case of GAMs.

Worth noting that GAMs can include linear terms like $x_j\beta_j$ in addition to the smoother terms ($x_j\beta_j$ is a particular type of smoother).

Fitting a GAM:

Start by picking arbitrary $\hat{\beta}_0$, $s_1(x_1), \ldots, s_J(x_J)$. From this information, we can compute the $\hat{\eta}_i$, $\hat{\mu}_i$, etc.

- Define the i-th working response

$$z_i = \begin{cases} \hat{\eta}_i + (y_i - \hat{\mu}_i)(\hat{\mu}_i(1-\hat{\mu}_i)) & \text{for logistic regression binary response models} \\ y_i & \text{for least-squares regression models} \end{cases}$$

- For fixed $\boldsymbol{z} = (z_1, \ldots, z_n)$, run "backfitting algorithm." That is, smooth

$$\boldsymbol{z} - \hat{\beta}_0 - s_1(\boldsymbol{x}_1) - \cdots - s_{j-1}(\boldsymbol{x}_{j-1}) - s_{j+1}(\boldsymbol{x}_{j+1}) - \cdots - s_J(\boldsymbol{x}_J)$$

  on $\boldsymbol{x}_j$ to get new estimated $s_j(\boldsymbol{x}_j)$. Iterate 3-4 times to get improved estimates of the $s_j(\boldsymbol{x}_j)$ for all j.

Repeat the above two steps until convergence.

One difference between additive models and GAM for binary responses: Cross-validation criterion for smoothing parameter

Instead of choosing λ to minimize

$$\text{CV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - s_\lambda^{(i)}(x_i))^2,$$

minimize the sum

$$\text{CV}(\lambda) = -\sum_{i=1}^{n}\left(y_i \log p_\lambda^{(i)} + (1-y_i)\log(1-p_\lambda^{(i)})\right).$$

Interesting to note that because of the similarities to linear models, some properties are inherited.

- Can compute a log-likelihood statistic.
- Likelihood ratio tests are (approximately) valid.
- "Non-parametric" degrees of freedom. Is calculated by summing the diagonal elements of the smoother equivalent of the "hat" matrix.
- So-called "score tests" for the significance of individual smoothers can be performed as chi-squared tests.

Example: Kyphosis data

Data set consists of 81 children who have had corrective spinal surgery.

The variables:

`Kyphosis`: a binary factor indicating if a kyphosis (type of deformation) was present after the operation

`Age`: age in months

`Number`: the number of vertebrae involved

`Start`: the number of the topmost vertebra operated on

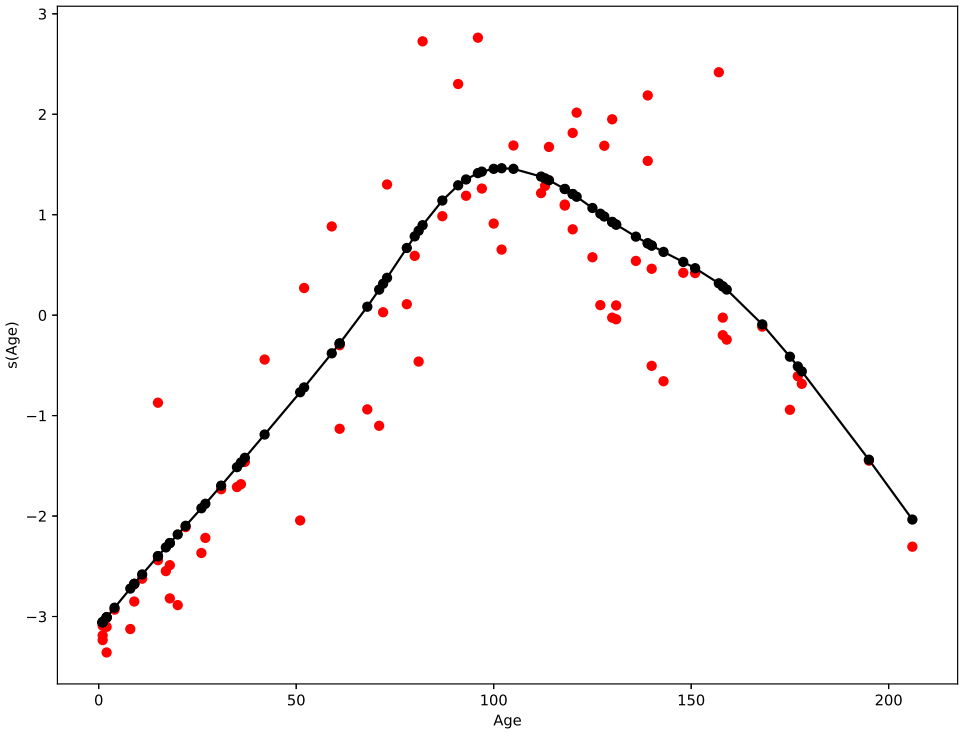
Data summaries:

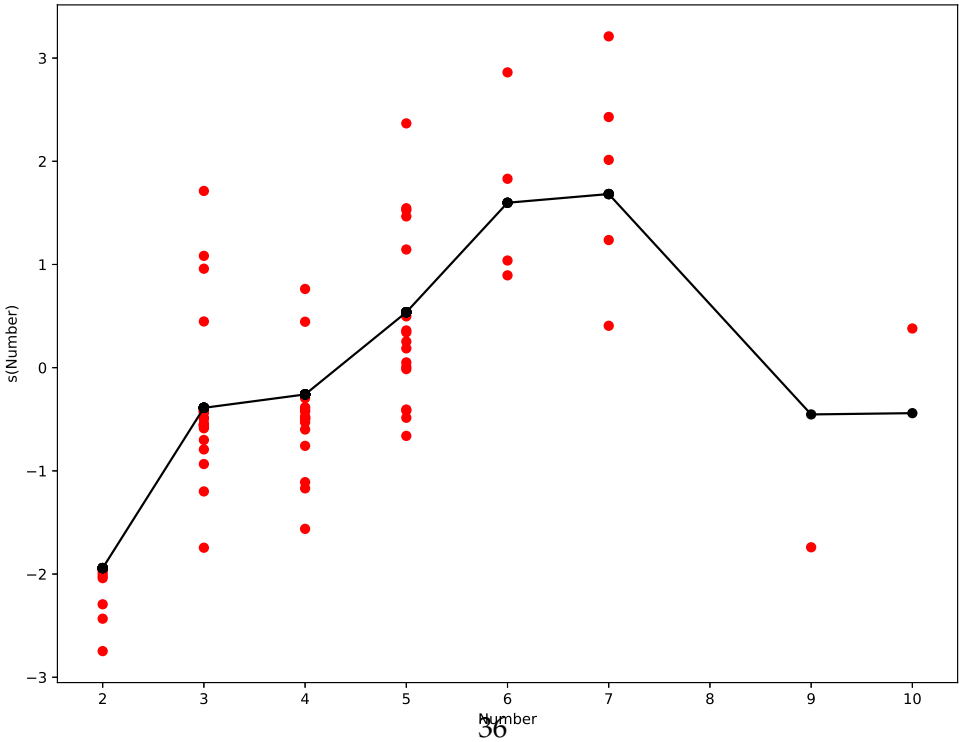
```
              Age      Number       Start
count  81.000000  81.000000  81.000000
mean   83.654321   4.049383  11.493827
std    58.104251   1.619423   4.883962
min     1.000000   2.000000   1.000000
25%    26.000000   3.000000   9.000000
50%    87.000000   4.000000  13.000000
75%   130.000000   5.000000  16.000000
max   206.000000  10.000000  18.000000
```

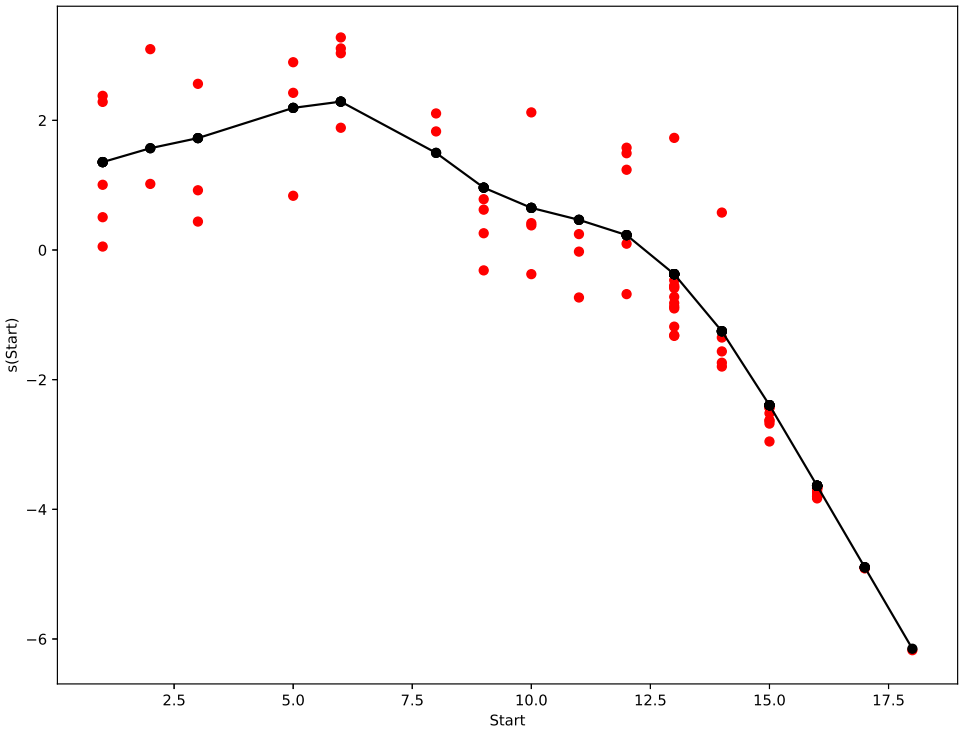
GAM in action:

See python code below.

```
r_gam_lib = importr('gam')
r_gam = r_gam_lib.gam
r_kyph = robjects.FactorVector(kyphosis[["Kyphosis"]].values.flatten())
r_Age = robjects.FloatVector(kyphosis[["Age"]].values.flatten())
r_Number = robjects.FloatVector(kyphosis[["Number"]].values.flatten())
r_Start = robjects.FloatVector(kyphosis[["Start"]].values.flatten())
kyph1_fmla = robjects.Formula("Kyphosis ~ s(Age) + s(Number) + s(Start)")
kyph1_fmla.environment['Kyphosis']=r_kyph
kyph1_fmla.environment['Age']=r_Age
kyph1_fmla.environment['Number']=r_Number
kyph1_fmla.environment['Start']=r_Start
kyph1_gam = r_gam(kyph1_fmla,family="binomial")
```


s(Age)
Age
3
2
1
0
−1
−2
−3
0
50
100
150
200


s(Number)
Number
3
2
1
0
−1
−2
−3
2
3
4
5
6
7
8
9
10



Is incorporating the three variables better than nothing?

Fit a model with only an intercept, and perform a likelihood ratio test against the model with all three terms.

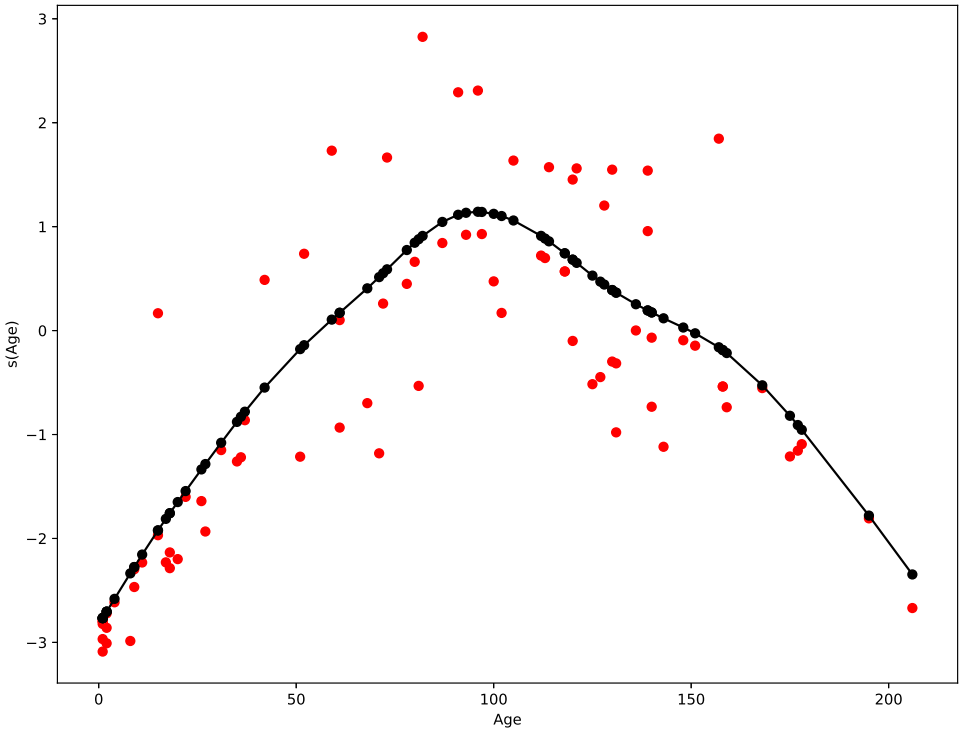
```
Analysis of Deviance Table

Model 1: Kyphosis ~ 1
Model 2: Kyphosis ~ s(Age) + s(Number) + s(Start)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        80     83.234
2        68     40.526 12   42.709 2.53e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

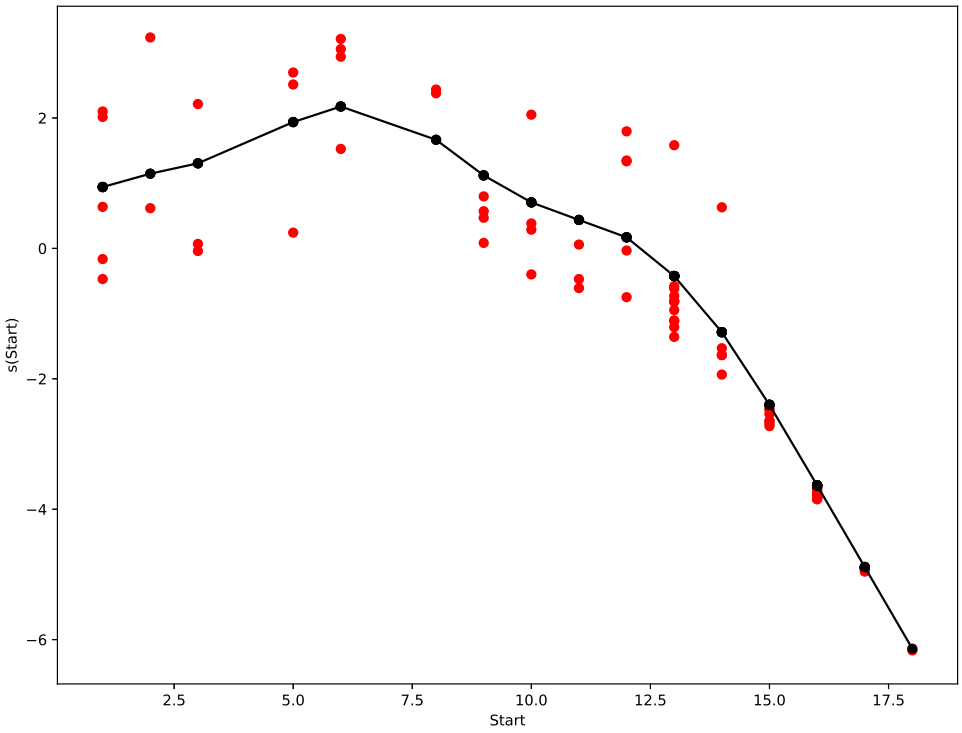
Very significant! However, the individual effects are not significant in the presence of others.

Try removing `Number` (the predictor with the highest p-value).

It turns out that `Start` is more significant, though not at the $\alpha = 0.05$ level.

Can plot individual smooths with “working response” values.





Predictions and mean estimation:

Syntax for prediction:

```
import rpy2.robjects as robjects
r_predict = robjects.r['predict']
kyph_new = robjects.DataFrame({'Age': robjects.IntVector((84,85,86)),
   'Start': robjects.IntVector((7,8,9))})
print(r_predict(kyph2_gam,kyph_new,type="response"))
```

Which results in:

```
        1         2         3
0.8528016 0.7985301 0.7017976
```