Splines and Loess and So Forth

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

.

Outline

Outline

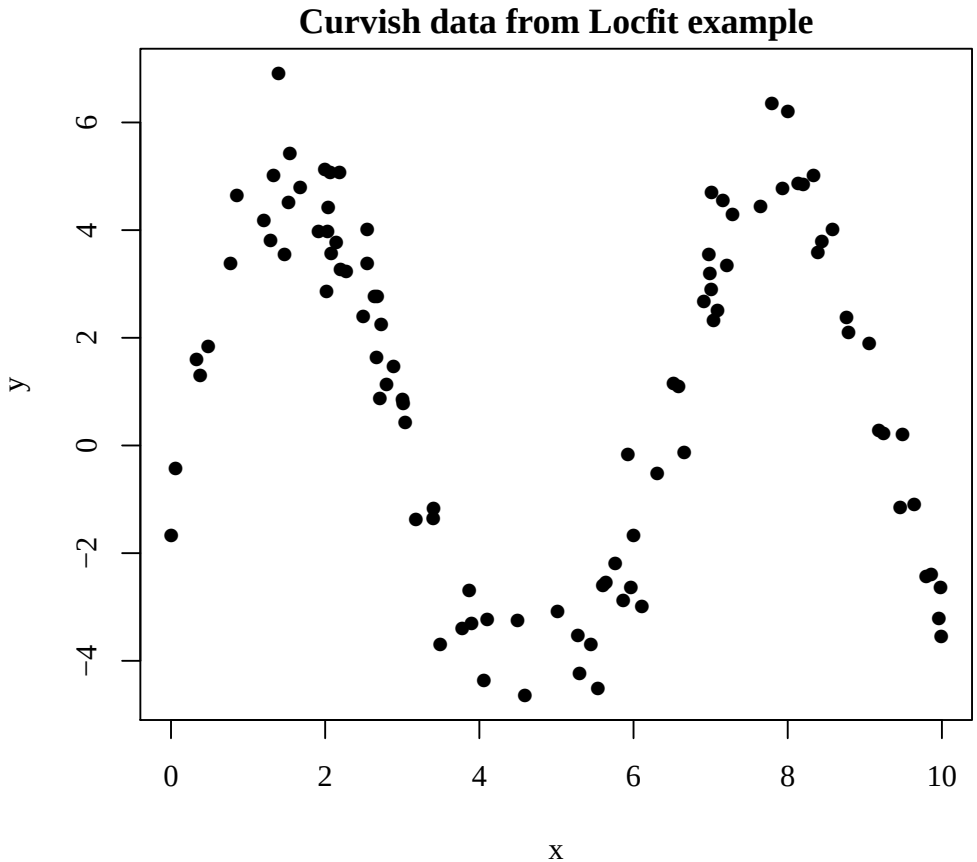
Get a Rubber Ruler

- Previous approach was “guess a formula” and “try to estimate its coefficients”.
- Change gears now, and say “I don’t know what the formula might be, I will use a flexible ruler to get the best predicted value I can get.”
- There are MANY details in this enterprise, TOO MANY different rubber rulers to use. Sorry.

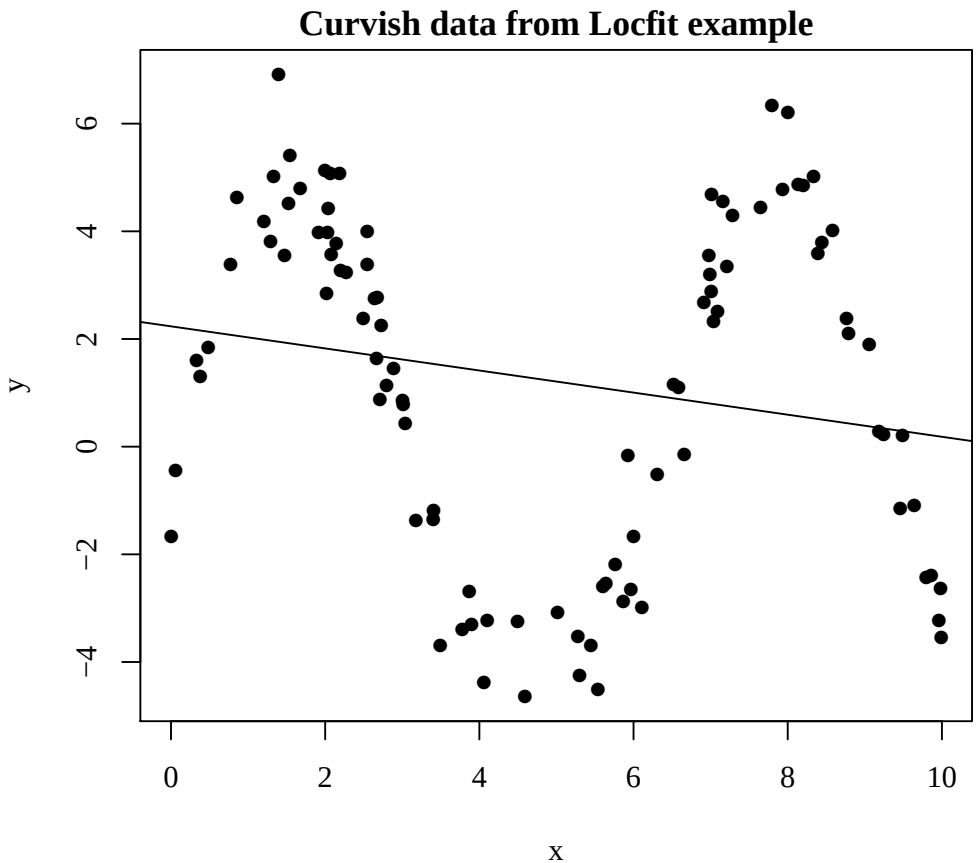
Outline

From the locfit package for R

Suppose the data looks like this (awful!)



Straight Line Not right

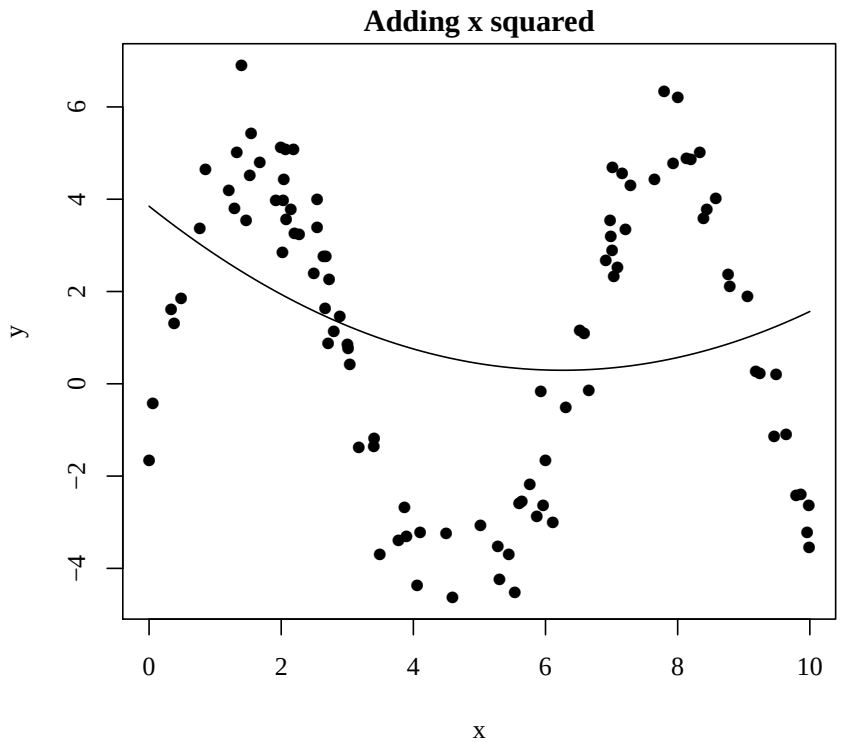


Would a Quadratic Term Help?

- People Say "Throw in a Squared Term"
- People often include x^2 to "check for nonlinearity"

$$\begin{array}{rcccc} \hat{y}_i & = & 3.8 & - & 1.1x_i + 0.09 \cdot x_i^2 \\ & & (0.96) & * & (0.44) * \ (0.042) * \end{array}$$

- 0.09 is "statistically significant" and positive. So?



Does a "Significant" b2 Mean Anything Here?

- What does the star mean to you?

```
Call:
lm(formula = y ~ x + I(x^2), data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-5.5108 -2.6118  0.8003  2.4781  5.8356

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.84765    0.96066   4.005 0.000121 ***
x           -1.13643    0.44710  -2.542 0.012612 *
I(x^2)       0.09083    0.04239   2.142 0.034657 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.047 on 97 degrees of freedom
Multiple R^2:  0.07998,  Adjusted R^2:  0.06101
F-statistic: 4.216 on 2 and 97 DF,  p-value: 0.01755
```

- Unfortunately, to most people, the star is evidence that "the quadratic model is right".

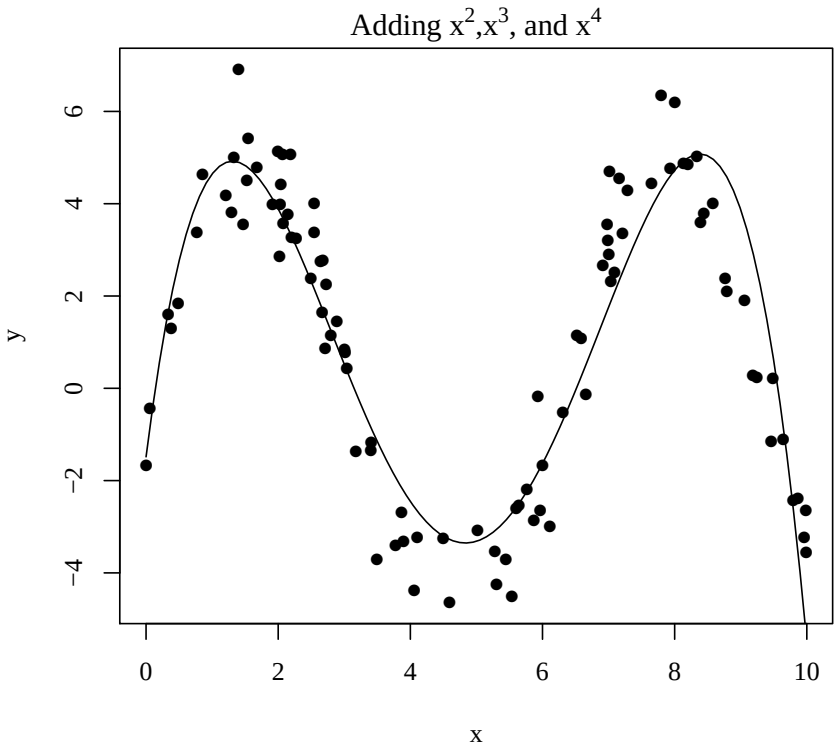
Add Higher Order Terms, Get Closer to the Right Model

Build a more detailed polynomial: include more terms, x_i^3 and x_i^4

- Recall Taylor's Theorem
- Fact: A polynomial has one fewer "hump" than it has terms. Since our graph has 3 "humps", the fitted model would need to include (at least) x_i^4

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i + \hat{b}_2 x_i^2 + \hat{b}_3 x_i^3 + \hat{b}_4 x_i^4 + e_i$$

Predicted Values of 4th Degree Polynomial: OK!



This almost never works in practice.

- Observed data is not as symmetrical, may require even more terms.
- Noisy data obscures the true shape, parameter variances large, standard errors large.
- Unless your x_i data is measured across a very wide range, coefficients for x^3 and x^4 won't be estimated with much precision. (Even if we "mean-center" or use orthogonal polynomials).

Outline

GAM is our LONG TERM GOAL

- Start out with the usual: $y_i = b_o + b_1 x1_i + b_i x2_i + b_3 x3_i + e_i$
- Suppose there's some "wiggle" in the effect of $x3_i$, but we don't know what it is

GAM = Semi-parametric Regression

- Wouldn't it be great to fit a generalized model, where we use some smoother for $x3_i$'s effect?

```
gam(y ~ x1 + x2 + s(x3), data=dat)
```

- s(x3) is "some good smoother function", may depend on parameters
- Generalized Additive Model (GAM): We can estimate some parametric terms, x1 and x2, and then let the rubber ruler fit the others.

Kinds of rubber rulers

- Natural Cubic Splines (recommended by Harrell, Regression Modeling Strategies)
  - Segments of X and limited curvature changes.
- Loess (Locally Weighted Regression): Most intuitive, I think
  - Separately predict each case! Reduce weight on more distant observations
- Smoothing Splines.
  - Allow the predictive line to "wiggle" at any point, but with a penalty

GAM: Generalized Additive Model

Leading R packages that can combine the usual regression with various kinds of "smoothed" predictor functions.

- "gam"
  Hastie, T. and Tibshirani, R. (1990) *Generalized Additive Models*. London: Chapman and Hall. A very famous book by the authors who founded this area of study.
- "mgvc"
  Wood S.N. (2006) *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC Press. This is my favorite regression book of all time and I strongly urge any regression modeler to get a copy! It has a superior introduction to regression, the generalized linear model, and random effects (mixed) models.

Other R packages

- rms: *Regression Modeling Strategies* (Frank Harrell). Another favorite regression book. This is where I learned about natural cubic splines
- polspline: Polynomial Spline routines (recommended by Harrell)
- locfit: loess as evolved out of Bell Labs (C. Loader)
- np: Nonparametric Econometrics (Tristen Hayfield and Jeffrey Racine).

Outline

Nonlinear Spline Overview

- A “piecewise linear spline” connects straight lines.
- A “cubic spline” connects cubic functions.
- A “natural spline” is a cubic spline with some restrictions on the end points.

Linear Splines

- Appears simple, parsimonious
- Divide the x-axis into sections
- Estimate regression lines for each one
- But don't allow "breaks" between segments.

Getting Different Slopes on Different Sections of x_i

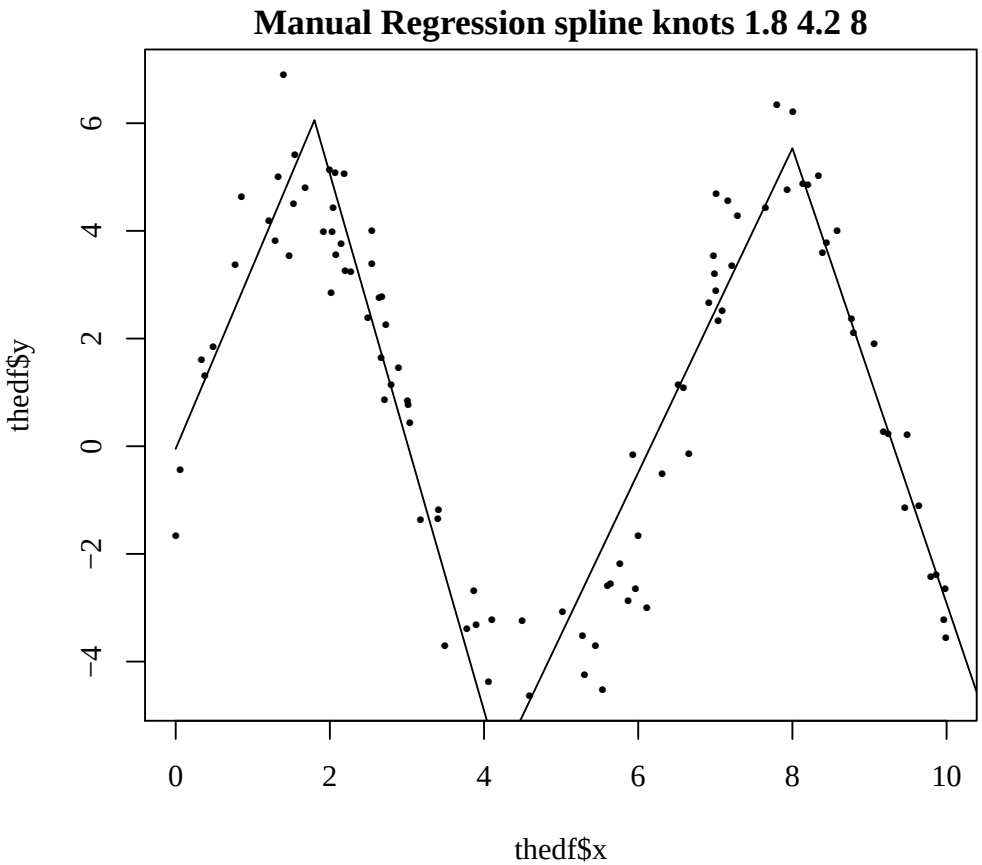
- Many authors use a "plus function" notation

$$(x_i - \tau_1)_+ = 0 \text{ if } x_i < \tau_1 \quad (1)$$
$$x_i - \tau_1 \text{ otherwise} \quad (2)$$

- Literally,
  - up to a knot (or "break point") τ_1, x_i has no effect.
  - Above τ_i, the effect is $(x_i - \tau_1)_+$
- Fitted model with one knot: $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x + \hat{b}_2 (x_i - \tau_1)_+$

Plot of Manual Spline Results

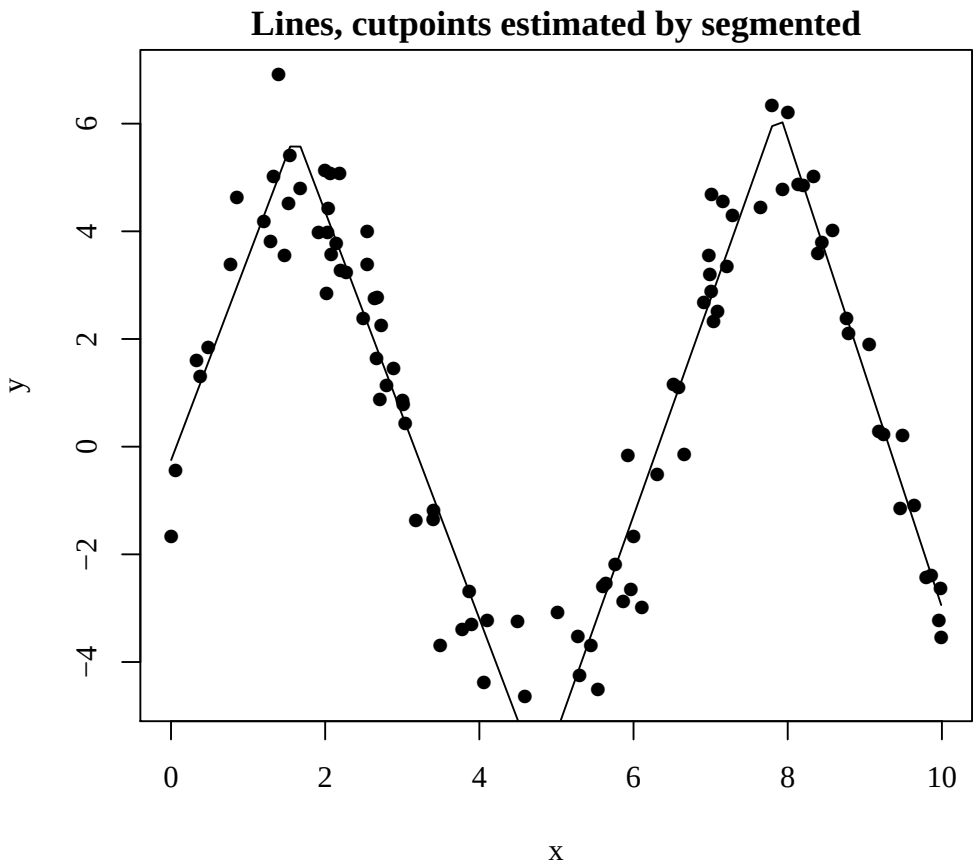
- For the curvy locfit data considered, I guessed 3 breakpoints.
- I manually estimate the model.
- I did not guess the breakpoints for the splines quite right (but close).



Routines to Guess Knots and Splines simultaneously

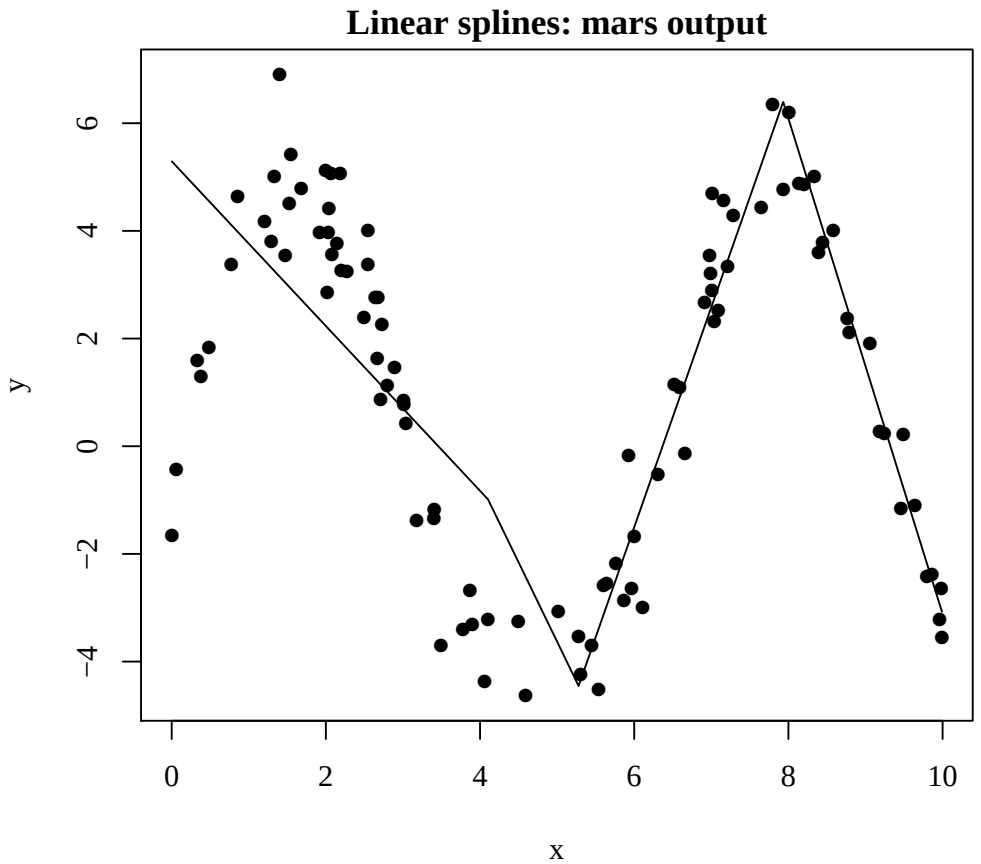
- R package "segmented"
- Nice interface!
  - Fit a regression
  - Give that lm object to segmented() with a request
- Somewhat fragile (my experience). Good initial guesses for the break points (the option psi) needed.
- segmented() provides a test for the significance of the assumption that the line segments connect at the break points.

segmented results



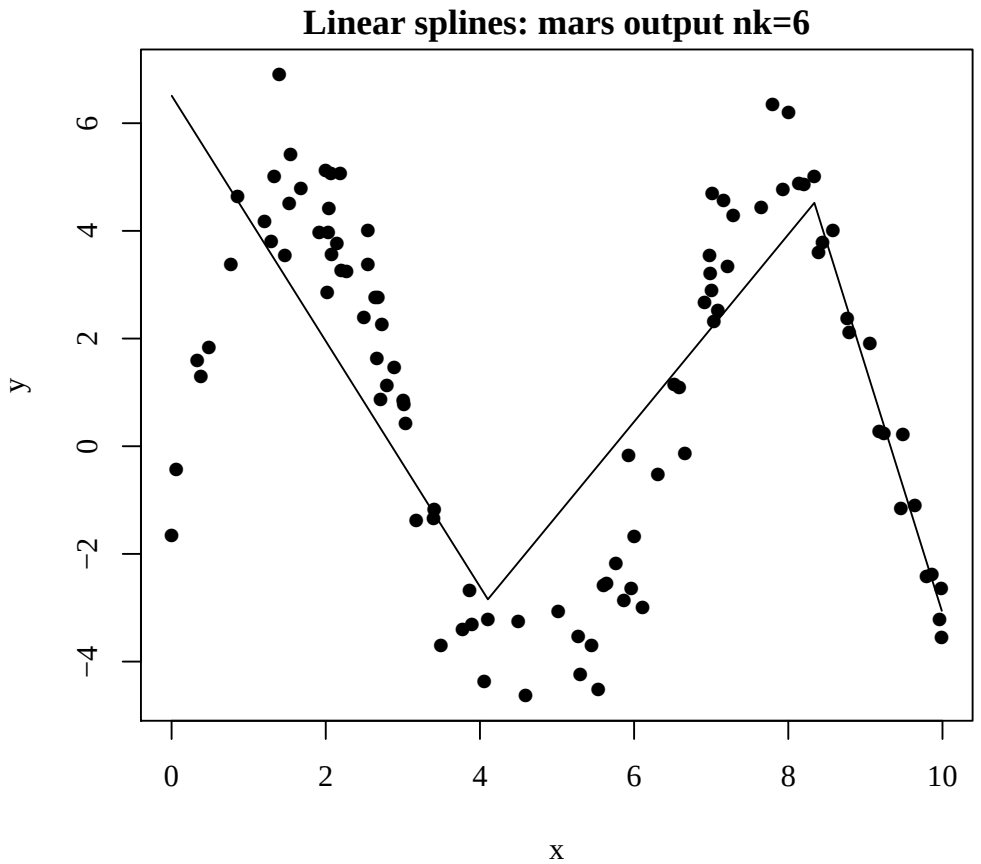
Multivariate Adaptive Regression Splines

- R package "mda" (Trevor Hastie), routine: MARS: Multivariate Adaptive Regression Splines (see Venables & Ripley, MASS, 4ed, p. 235)
- number of knots estimated from the data
- Has options to "curve" the segments as well!



MARS Plot Code With Fewer Breakpoints

Try a restricted number of breakpoints.



Do We Like Linear Splines (on a Theoretical Level)?

- Kinks. Do our theories really have "kinks"?
- Does effect of x remain constant until an instant where it changes from b_2 to b_3?
- Should model have them?
- Big philosophical question.

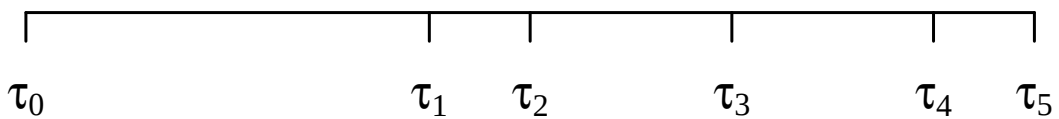
Splines without Kinks

- There are more types of spline models than you can shake a stick at.
- We focus on the natural cubic spline.

Cubic Spline Basics

- input variable x_i
- x_i is subdivided into k segments with end points $(\tau_0, \tau_1, \ldots, \tau_{k+1})$.

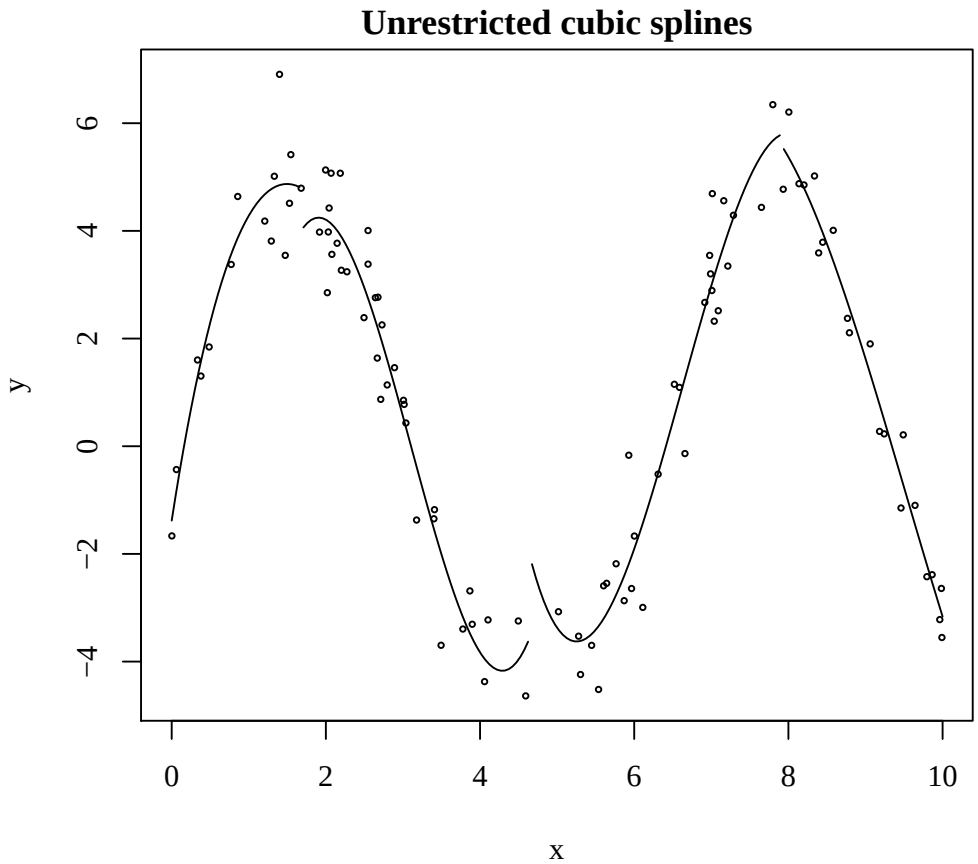
Segments with 4 knots in the interior of the line.



Cubic Spline Basics

- On each interval, we can imagine a cubic model,

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i + \hat{b}_2 x_i^2 + \hat{b}_3 x_i^3$$



First Fix: Smooth Cubic Splines

- Each segment's spline must blend smoothly into the next. So assume:
  1. No gaps at the knots.
  2. Smooth Connections between parts. No pointy kinks at the knots.
- Those 2 assumptions have VERY MUCH power on simplifying the problem. In particular, they mean that the slope and second derivatives at the knot points have to match exactly.

Adapt the "plus function" notation for the Cubic Spine

- squared plus function

$$(x_i - \tau)_+^2 = 0 \text{ if } x_i < \tau \quad (3)$$
$$(x_i - \tau)^2 \text{ otherwise} \quad (4)$$

- Cubic plus function:

$$(x_i - \tau)_+^3 = 0 \text{ if } x_i < \tau \quad (5)$$
$$(x_i - \tau)^3 \text{ otherwise} \quad (6)$$

I believe these are called "truncated power basis" splines. These are the "teaching version" of cubic splines

Simplifying the Cubic Splines

The model would be too overwhelming if we try to estimate a separate cubic equation within every segment.

$$\begin{aligned}\hat{y}_i \quad = \quad & \hat{b}_0 + \hat{b}_1 x_i + \hat{b}_2 x^2 + \hat{b}_3 x_i^3 + \\ & \hat{b}_4 + \hat{b}_5 (x_i - \tau_1)_+ + \hat{b}_6 (x_i - \tau_1)_+^2 + \hat{b}_7 (x_i - \tau_1)_+^3 \ (\textit{after first knot}) \\ & \hat{b}_8 + \hat{b}_9 (x_i - \tau_2)_+ + \hat{b}_{10} (x_i - \tau_2)_+^2 + \hat{b}_{11} (x_i - \tau_2)_+^3 \ (\textit{after second knot})\end{aligned}$$

But we can throw away many of those coefficients

- no gaps: $b_4 = b_8 = 0$.
- no kinks at knots: $b_5 = b_6 = 0$.
- no kinks at knots: $b_6 = b_{10} = 0$.

And so Restricted Cubic Spline Model Boils Down To

- Renumbering the coefficients in a sane way, we are left with

$$\begin{aligned} \hat{y}_i &= \hat{b}_0 + \hat{b}_1 x_i + \hat{b}_2 x^2 + \hat{b}_3 x_i^3 + \\ &\quad + \hat{b}_4 (x_i - \tau_1)_+^3 \ (\textit{after first knot}) \qquad (7) \\ &\quad + \hat{b}_5 (x_i - \tau_2)_+^3 \ (\textit{after second knot}) \\ &\quad \textit{and so forth} \qquad (8) \end{aligned}$$

- After the first interior knot, add $\hat{b}_4 (x_i - \tau_1)_+^3$.
- After the second knot, we have BOTH $\hat{b}_4 (x_i - \tau_1)_+^3 + \hat{b}_5 (x_i - \tau_2)_+^3$.
- Emphasize: a change in the coefficient for one segment may have a "ripple effect" across all of the others. If one segment's coefficient gets tuned "high," then the others have to adjust to compensate.

Natural Splines: One more restriction

- The "outside" segments are "unteathered" on the edges.
- To stabilize the fit there, a **restricted (or natural) cubic spline** allows only a linear relationship on those segments. (see Harrell's *Regression Modeling Strategies*, p. 20).
- The "theoretical" view, then, is just a linear equation supplemented by a bunch of cubic "plus" functions.

$$\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i + \hat{b}_2 (x_i - \tau_1)_+^3 + \hat{b}_3 (x_i - \tau_2)_+^3 + \ldots$$

More Sophisticated Computer Tricks Behind the Scenes

- We could manually create the power variables $(x_i - \tau_j)_+^3$ and fit with OLS.
- However, that is not the "numerically most stable" approach. "The truncated power basis is attractive because the plus-function terms are intuitive and maybe entered as covariates in standard regression packages. However, the number of plus-functions requiring evaluation increase with the number of breakpoints, and these terms often become collinear, just as terms in a standard polynomial regression do." (Lynn A Sleeper and David P. Harrington, "Regression Splines in a Cox Model with Application to Covariate Effects in Liver Disease", *Journal of the American Statistical Association*, 85(412) December 1990, p.943 (941-949)).
- The "b-spline" encoding is more numerically stable approach. Harrell states that the difference is not usually substantial. On the other hand, Wood emphasizes the b-spline quite a bit.

Examples? Let's use Frank Harrell's rms Package

- Harrell's rcs() function creates the "truncated power basis natural spline".

```
rcs(x, parms=5)
```

- By default, rcs will position knots so that the data is divided into equally sized subgroups (quintiles, etc.).
- R package: splines. Function "ns" creates the "numerically more stable" b-spline representation for a natural cubic splines

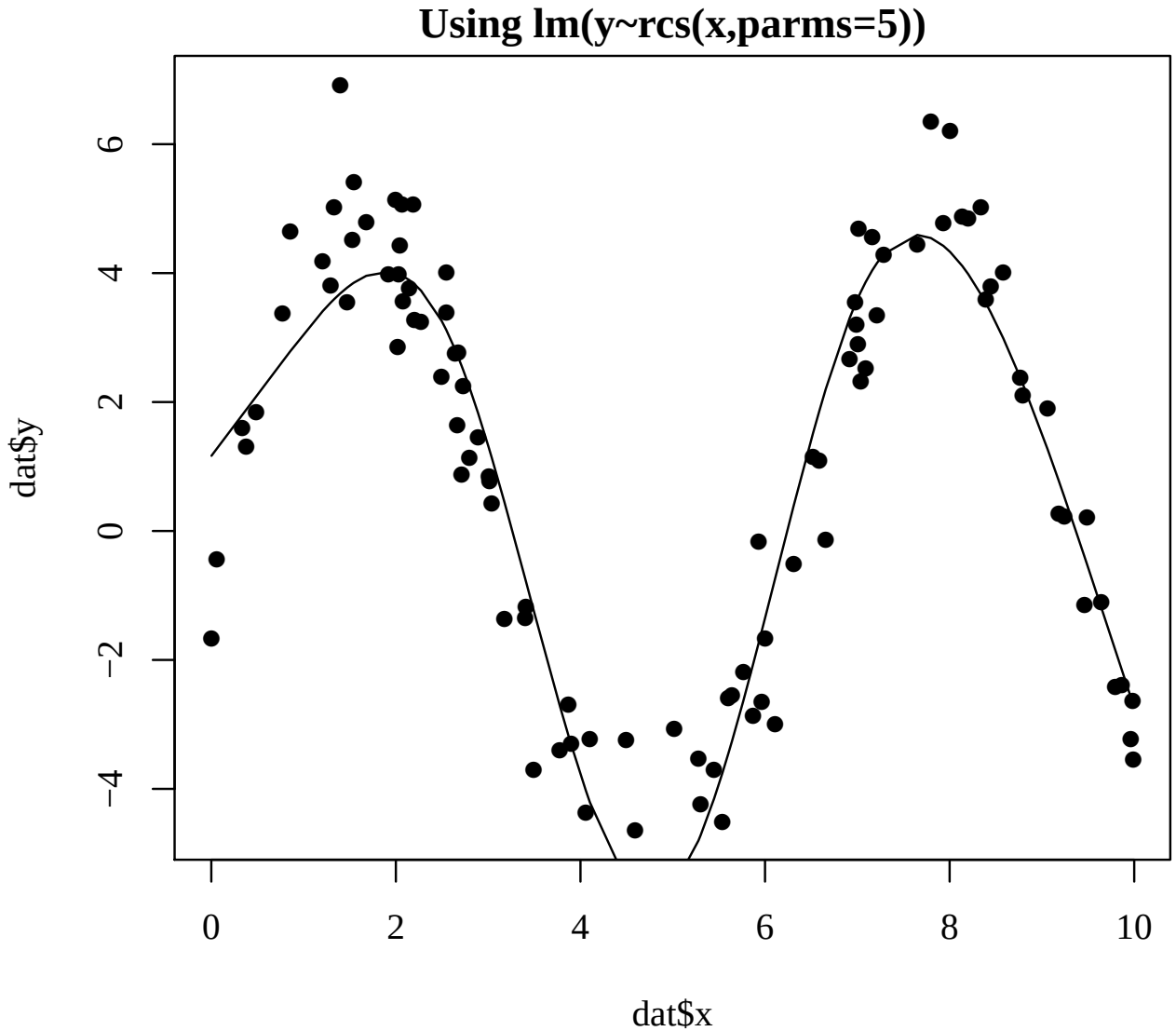
```
ns(x, df=4)
```

Fit a Regression with a Restricted Natural Cubic Spline

- We can use either $rcs(x, nk = knots)$ or $ns(x, df = k - 1)$ as regression model inputs:

```
m5 <- lm(y ~ rcs(x,5), data=dat)
```

Fit a Regression with a Restricted Natural Cubic Spline ...

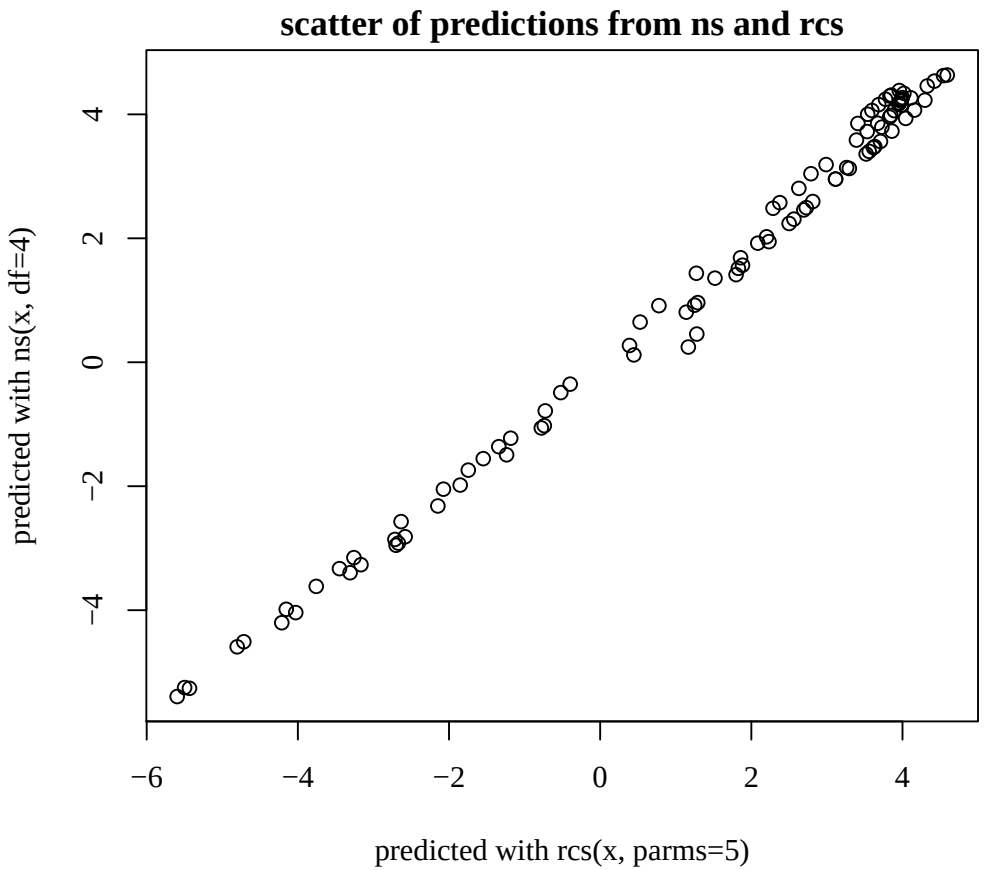


Model Summary Output Difficult to Understand, Though

```
Call:
lm(formula = y ~ rcs(x, parms = 5), data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-2.8319 -0.6288 -0.0866  0.7363  3.2213

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)              1.1623     0.4637   2.507   0.0139 *
rcs(x, parms = 5)x       1.8992     0.2710   7.009 3.46e-10 ***
rcs(x, parms = 5)x'    -39.8111     2.1626 -18.409  < 2e-16 ***
rcs(x, parms = 5)x''   119.3189     5.5632  21.448  < 2e-16 ***
rcs(x, parms = 5)x''' -161.6405     6.5280 -24.761  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.084 on 95 degrees of freedom
Multiple R^2:  0.8859,  Adjusted R^2:  0.8811
F-statistic: 184.4 on 4 and 95 DF,  p-value: < 2.2e-16
```

Predictions from b-spline "ns" and Truncated "rcs" Very Similar

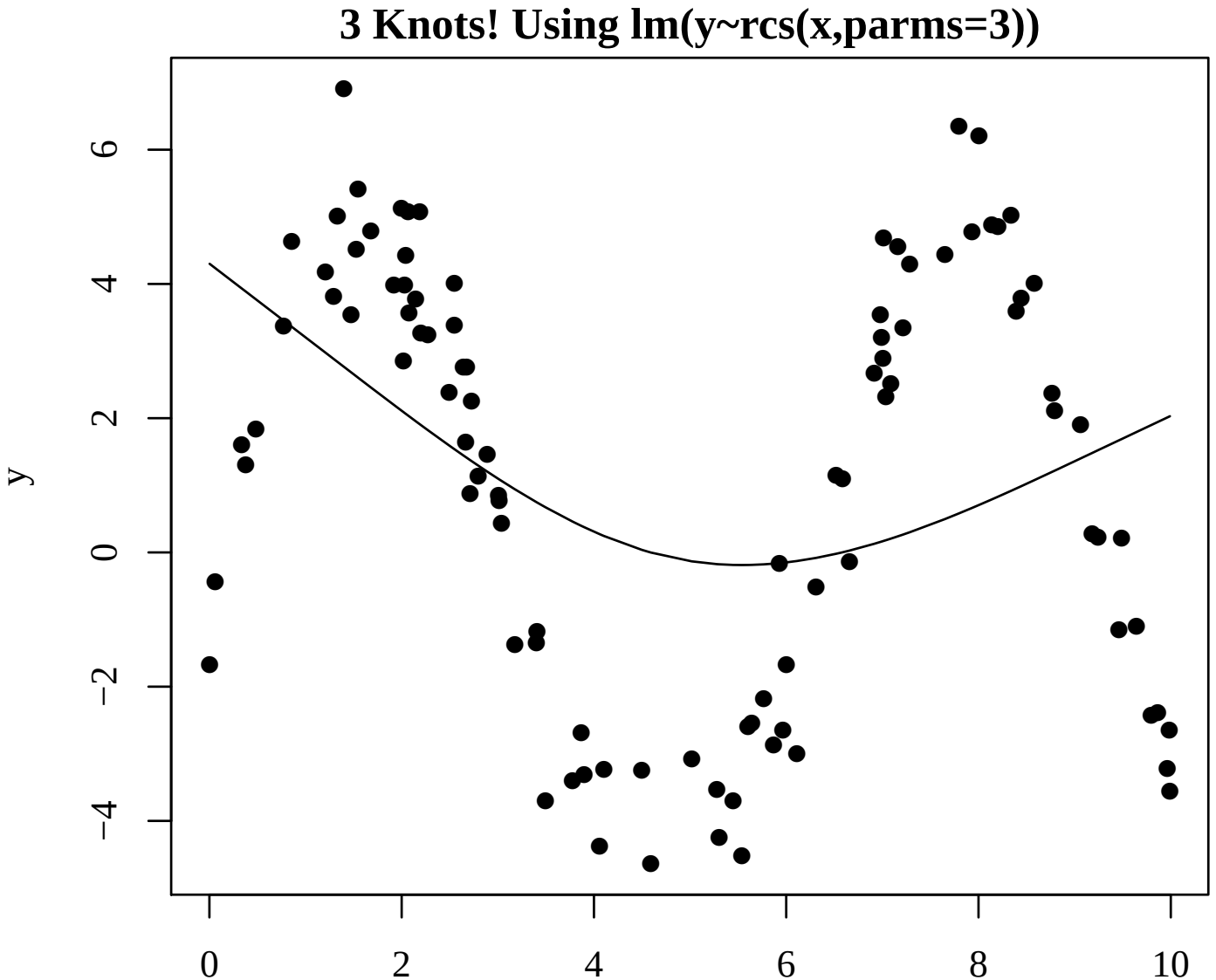
```
m5 <- lm(y ~ ns(x, df=4), data=dat)
```



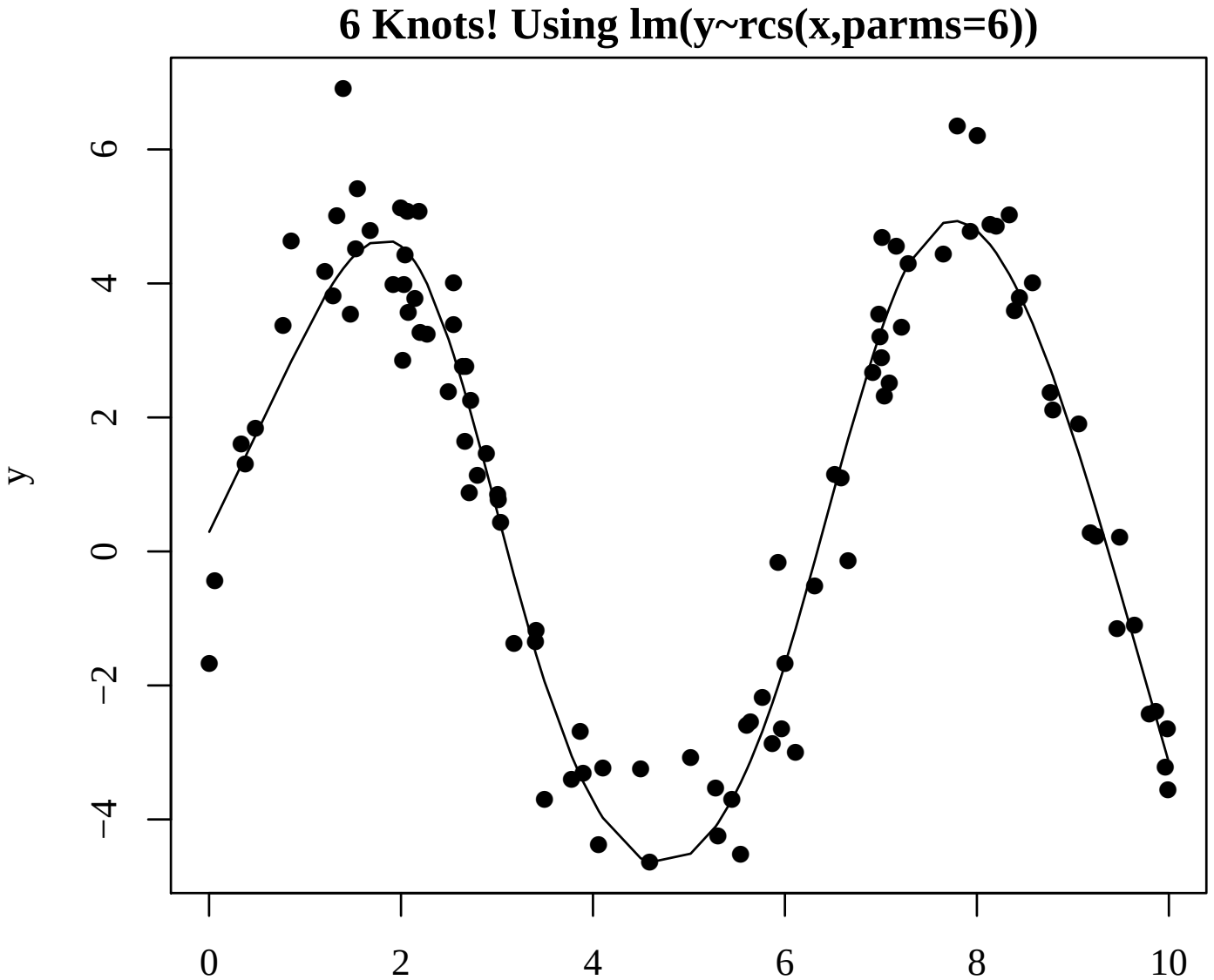
Choosing the number of knots

- Harrell writes that the conventional wisdom, following studies by Stone, is that the positioning of the knots is not very influential. If the knots section off the data into evenly sized groups, then the fit is "pretty good."
- But the number of knots is important.
  - Usually need at least 4 knots
  - Usually more than 7 knots not helpful.

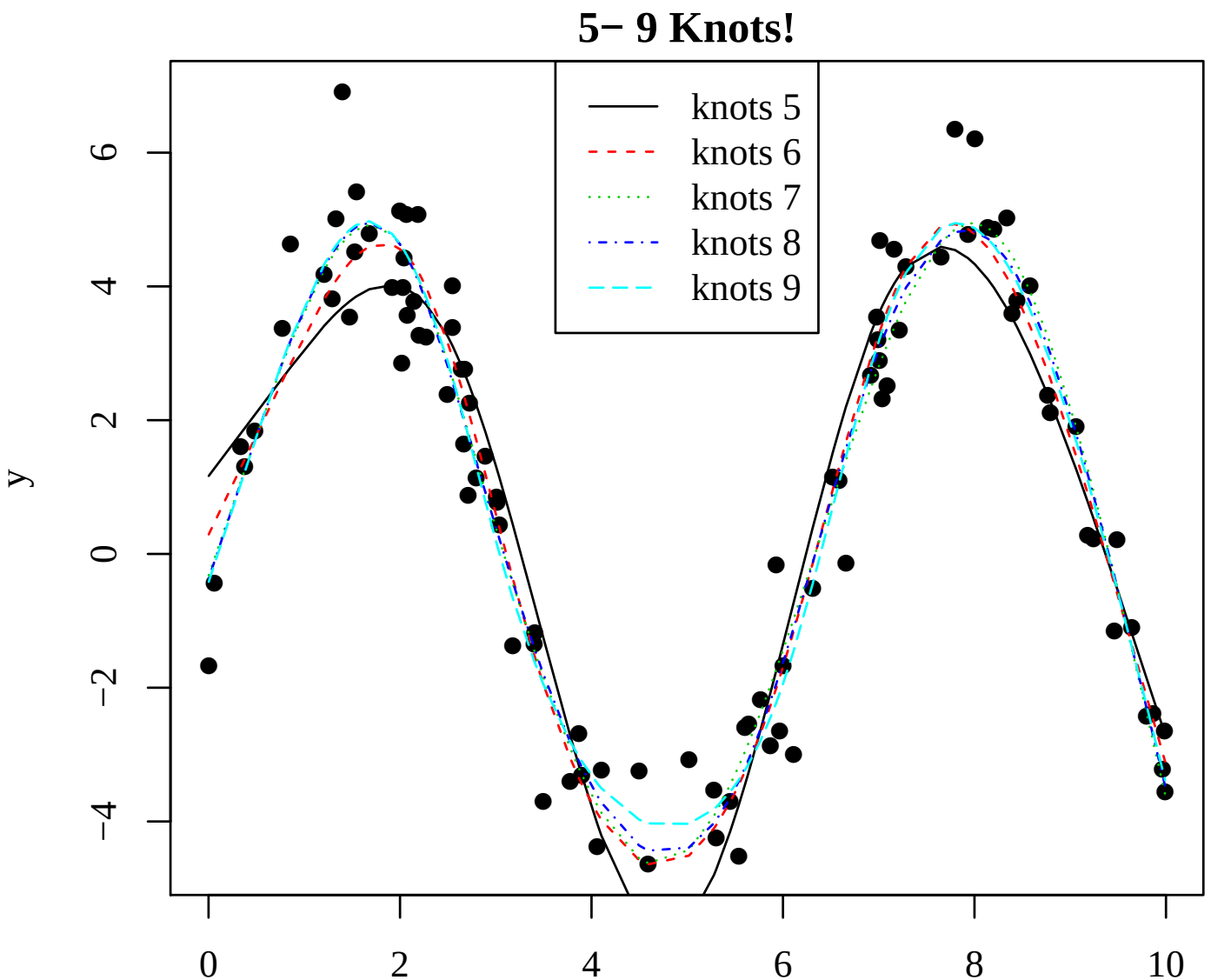
Clearly, 3 knots is not Enough



6 Knots Not So Bad



Several different values for knots



Danger of Over-Fitting

- Over-fitting: customizing the model to quirks of one random sample.
- Theoretically, want a manageable model: get rid of many knots as possible.
- Should penalize use of knots, somehow
- Cross Validation is a concept that can be used as a guide in deciding on the appropriate number of knots

Cross Validation: Choosing the number of knots

- Basic Idea: Choose your modeling approach, including number of knots.
- Estimate model for a subset
- Check if fitted model works well for other cases that were not used in fitting
- For evaluating "out of sample" predictions, we use a "misfit indicator"*
  - Such as "mean square error" or Akaike's Information Criterion, Deviance, etc (for more advanced applications).

"Leave One Out" Cross Validation

- Remove the $i'th$ observation, Re-calculate the predictive curve on $N - \{i\}$.
- Calculate a "leave-one-out prediction" $\check{y}_i$. Meaning: use the model estimated on $N - \{i\}$ to predict for the $i'th$ observation.
- How bad was that prediction? Easy: $(y_i - \check{y}_i)^2$
- Repeat procedure for all observations. Calculate the average of squared errors.

$$CV = \frac{1}{N} \sum_{i=1}^{N} (y_i - \check{y}_i)^2$$

When the Dust Clears

- Calculate CV for models with 4 knots, then 5 knots, and so forth
- Choose number of knots that minimizes the CV measure
- Many Variants, "Generalized Cross Validation"

Outline

Sometimes Called "Nonparametric" Regression

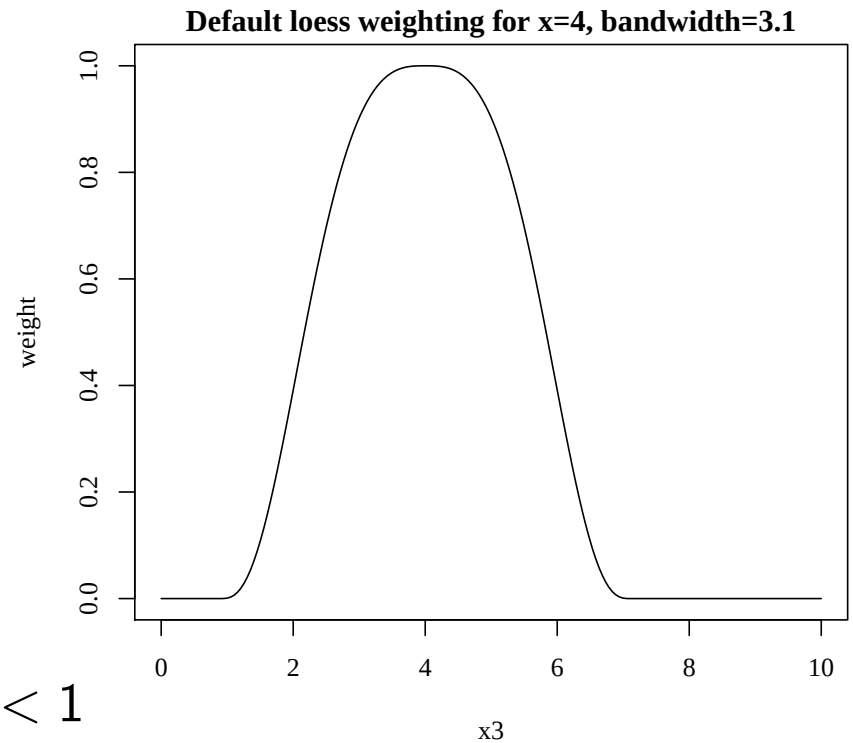
- LOESS: Locally Weighted Error Sum of Squares regression. Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association* 83, 596–610.
- Fit a separate predictive model for each observation!
- Model is "nonparametric" in the sense that we do not emphasize estimation of "**the slope**" for a particular "coefficient".
- Now we estimate "the slope" for 100s of separate points. Loess is not "nonparametric" in my view. It is mega-parametric!

LOESS

1. Make a separate regression model for each value of the input variable.
2. Construct each regression so it puts most "weight" on observed scores that are close.
3. Output is a column of predictions, one for each observation.

Weights for Points Close to x'

- Fit one regression for each value of x_i. Concentrate on one now x'.
- Points outside the span of h have 0 Weight.
- Closer points inside the span have more weight
- Typical:



$$Weight\,(x_i) = \begin{cases} (1 - |\frac{x_i - x'}{h}|^3)^3 & if\ |\frac{x_i - x'}{h}| < 1 \\ 0 & otherwise \end{cases} \quad (9)$$

How Weights Affect Predictions

- In "ordinary least squares," there are no weights. Minimize this:

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

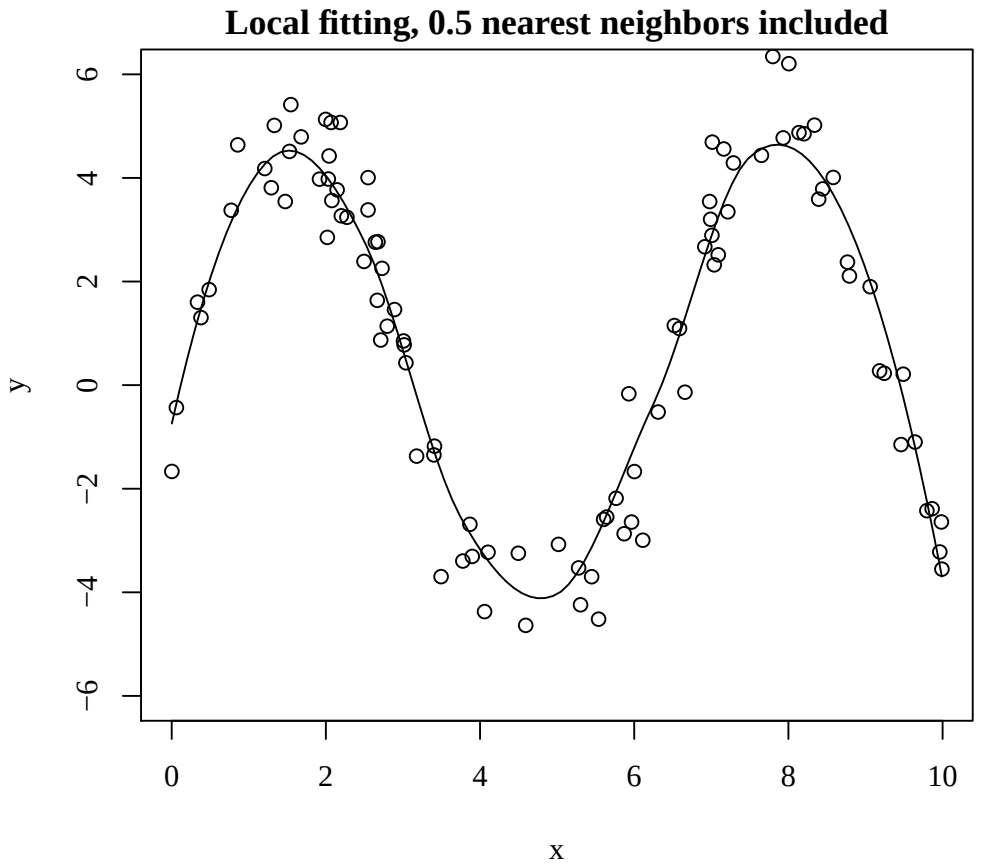
- Loess uses weighted local regression to de-emphasize the far-away values.

$$\sum_{i=1}^{n} W(x_i)[y_i - \hat{y}_i]^2$$

- The predictor can be
  - linear model, $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i$,
  - quadratic model, $\hat{y}_i = \hat{b}_0 + \hat{b}_1 x_i + \hat{b}_2 x_i^2$.

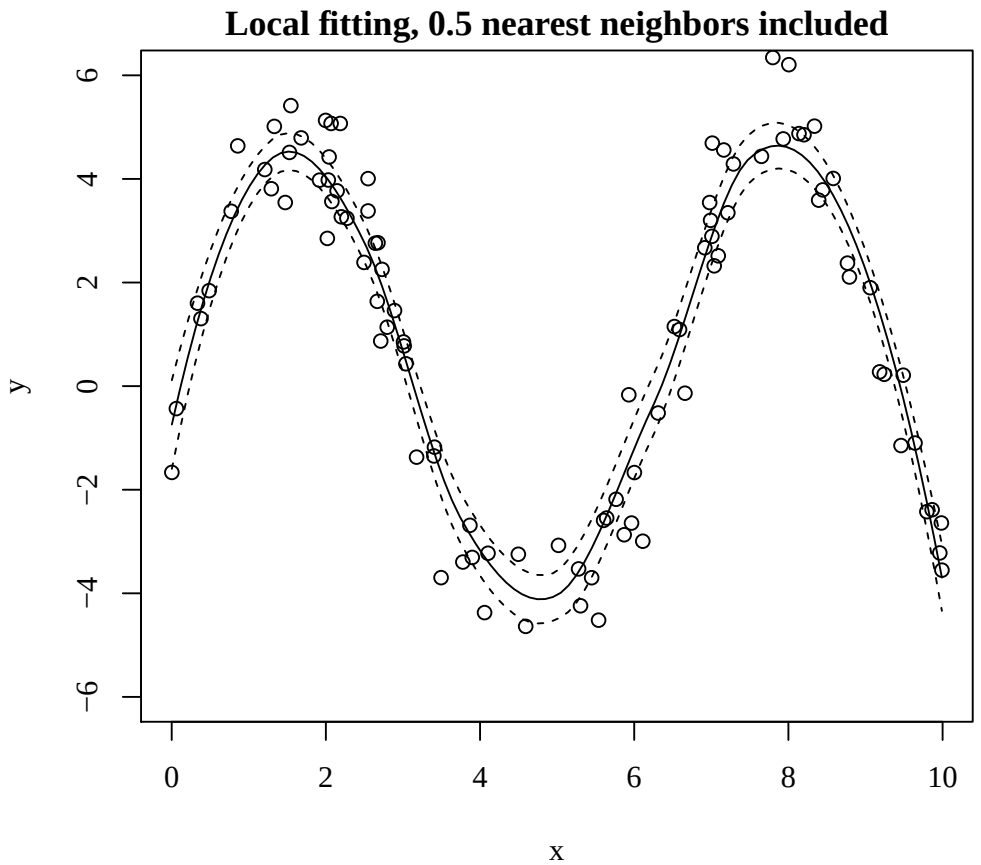
Using locfit's local polynomial estimator

- locfit is a generalized framework for loess with linear and generalized regression models.
- Author: C. Loader early developer of code for loess at AT&T
- The plotter makes the points look like a smooth line.

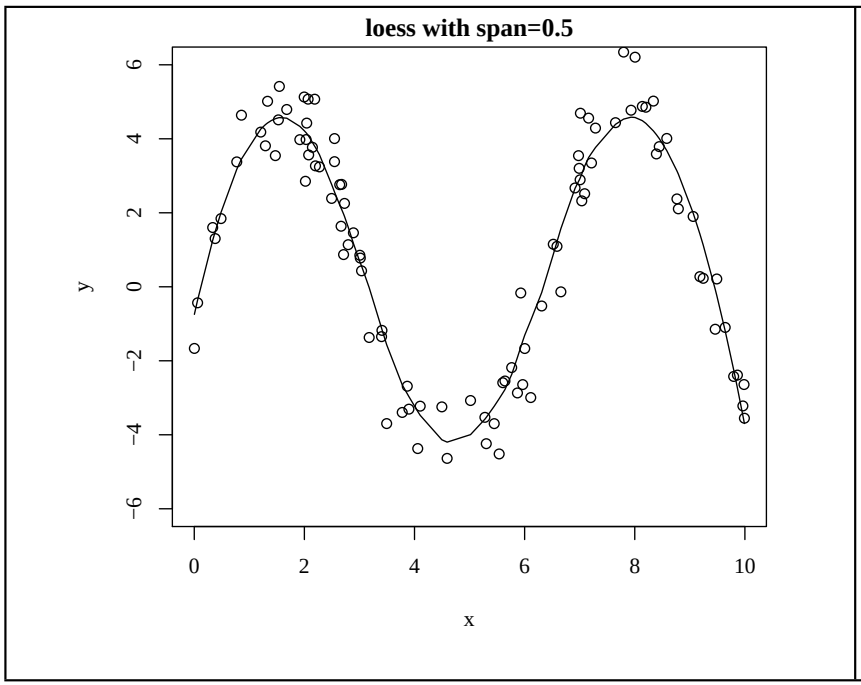


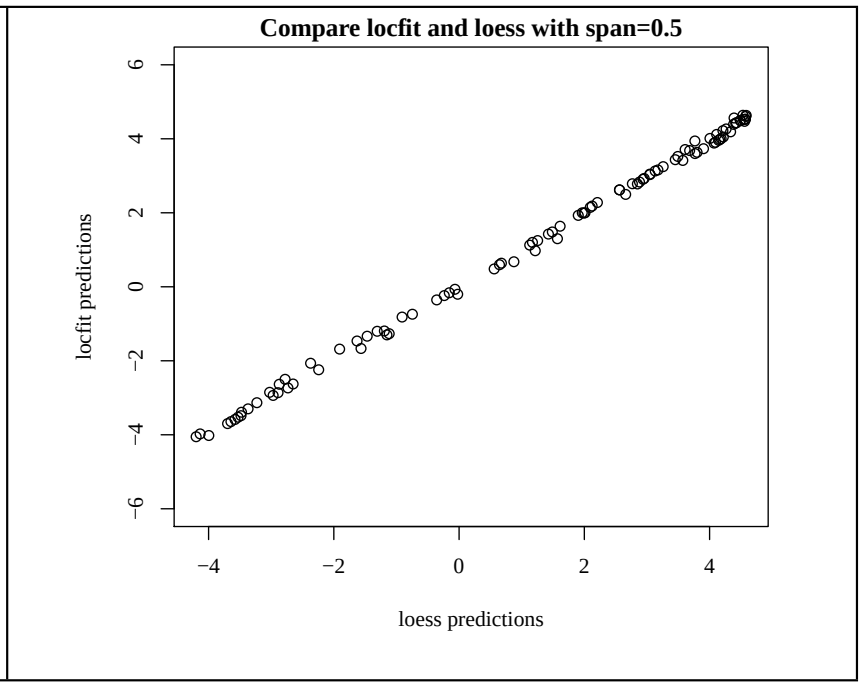
Add 95% Local Confidence Interval

- The 95% band can be calculated in a variety of ways
- This uses the local estimate of the variance to customize the 95% confidence interval



Compare Against loess Function output





Extension to several input variables

- Second Generation loess tools include the ability to add more predictor variables.
- The transition from one to several variables is very simple in theory. The weights simply depend on distance, which can be measured in various ways.
- "Curse of dimensionality" makes fitting these "nonparametric" curves very costly if there are more than 3 predictors.

Outline

Smoothing Splines

- Smoothing Splines: marriage of Splines and Loess
- Every x_i becomes a knot.
- Build a natural spline model that "wiggles" between every pair of points.
- We penalize wiggliness.

Here is the objective function

- Goal is to minimize a Sum of Squared Errors Plus a penalty for "wiggliness" (a roughness penalty).

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

- A theorem in Wood demonstrates that $f(x_i)$ will be cubic.

λ is a Smoothing Parameter

- If λ is too small, f offers no simplification, it fits the data exactly.
- If $\lambda = \infty$, a harsh penalty leads to a straight line model (no curves)
- May manually set λ
  - or use some algorithm to estimate models for various λ and compare results by Cross Validation.

One Way To Summarize "degrees of freedom" used

- Suppose you have the "smoother matrix" that maps from observed y_i to the predicted y_i. For each i:

$$\hat{y}_i = h_{i1}y_1 + h_{i2}y_2 + h_{i3}y_3 + \ldots + h_{iN}y_N \tag{10}$$

  - If $h_{ii} = 1$, and $h_{ij} = 0$, then this just "reproduces" y_i.
  - But, if $h_{ii} = 0$, it means that case i is just "receiving" its prediction from the other cases. We use no "unique information" in predicting i.

- Thus, the sum of the "smoother coefficients" (the diagonal elements if you view $[h_{ii}]$ as an $N \times N$ matrix)

$$\sum h_{ii} \tag{11}$$

  serves as an indicator of the "customization" needed to make a set of predictions.

That sum is "effective degrees of freedom."

Sorry. Necessary to Read Manual on Parameters and Settings

- Models can be fit by various methods
  - Possible to set value of "effective degrees of freedom".
    - In that case, the smoothing parameter λ is adjusted so the final model uses the desired df.
  - User may instead specify a smoothing parameter, λ (or the like), and then the "effective degrees of freedom" used will come out as a logical consequence.
  - Software may use Cross Validation to choose a value of λ, possibly beginning calculations at the user's best initial guess for λ.

Many Routines Available

- smooth.spline in R core (thanks to Brian Ripley and Martin Maechler).
- package "pspline" has smooth.Pspline (defaults to natural cubic smoothing spline)

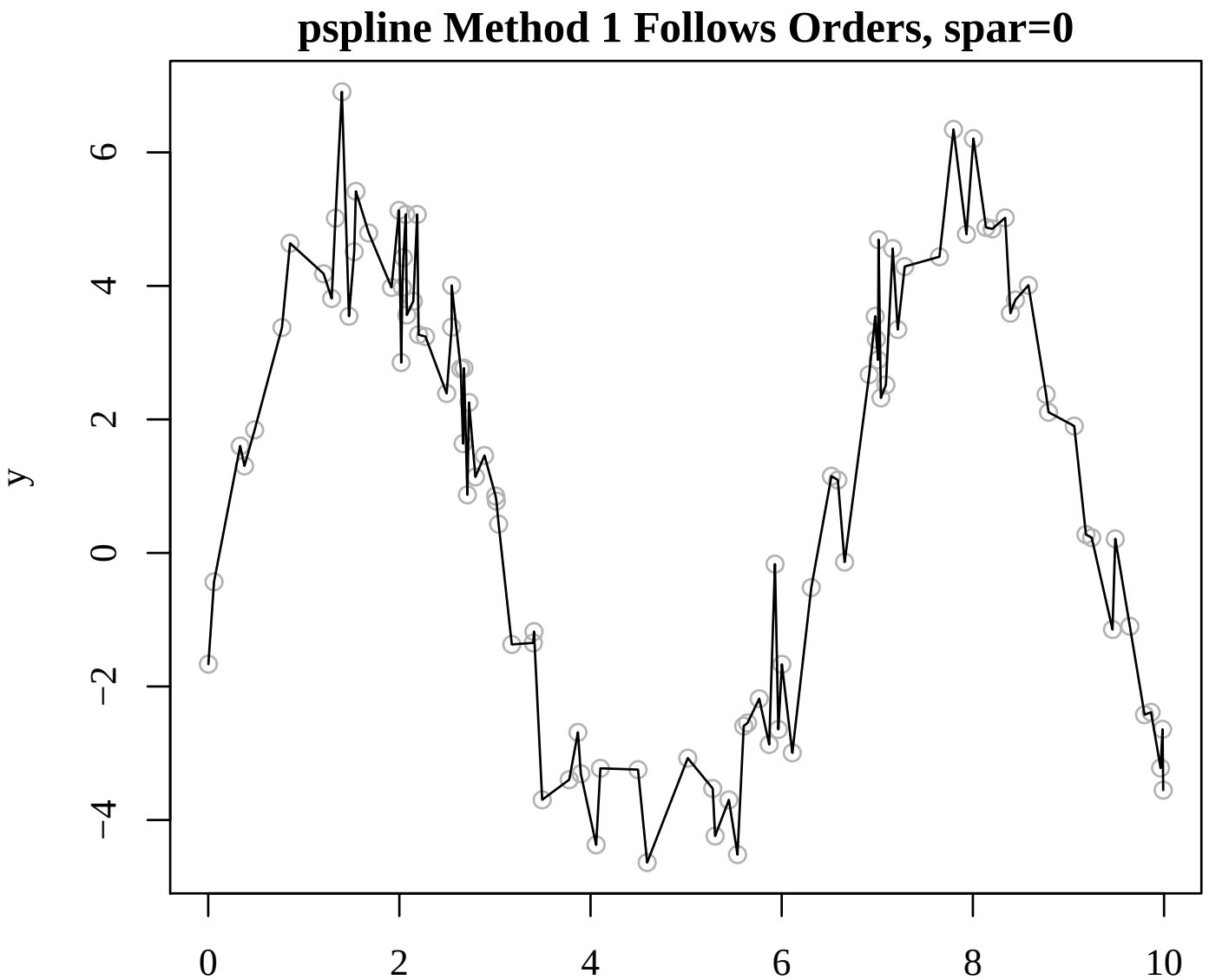
I Find the pspline Manual Most Understandable

method=1 forces the use of the designated spar, which defaults to 0 (no smoothing)

```
psp10 <- smooth.Pspline(x, y)
psp10
```

```
Call:
smooth.Pspline(x = x, y = y)

Smoothing Parameter (Spar): 0
Equivalent Degrees of Freedom (Df): 100
GCV Criterion: NaN
CV  Criterion: NaN
```

Pspline method=1, spar=0

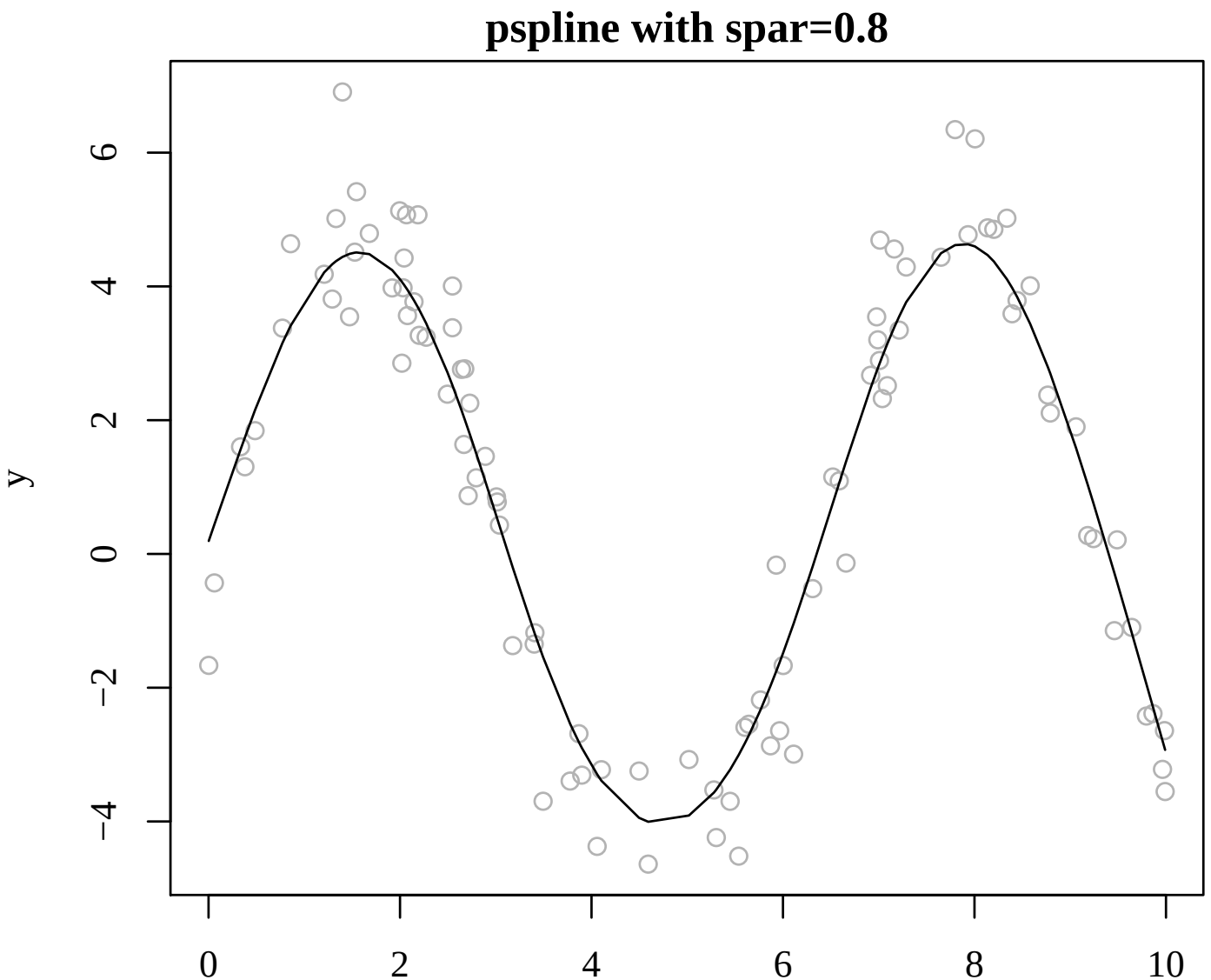


Pspline method 1, with More Smooothing (spar=0.8)

```
psp15 <- smooth.Pspline(x, y, spar=0.8, method=1)
psp15
```

```
Call:
smooth.Pspline(x = x, y = y, spar = 0.8, method = 1)

Smoothing Parameter (Spar): 0.8
Equivalent Degrees of Freedom (Df): 7.611132
GCV Criterion: 0.7944122
CV  Criterion: 0.8109217
```

Pspline method 1, spar=0.8

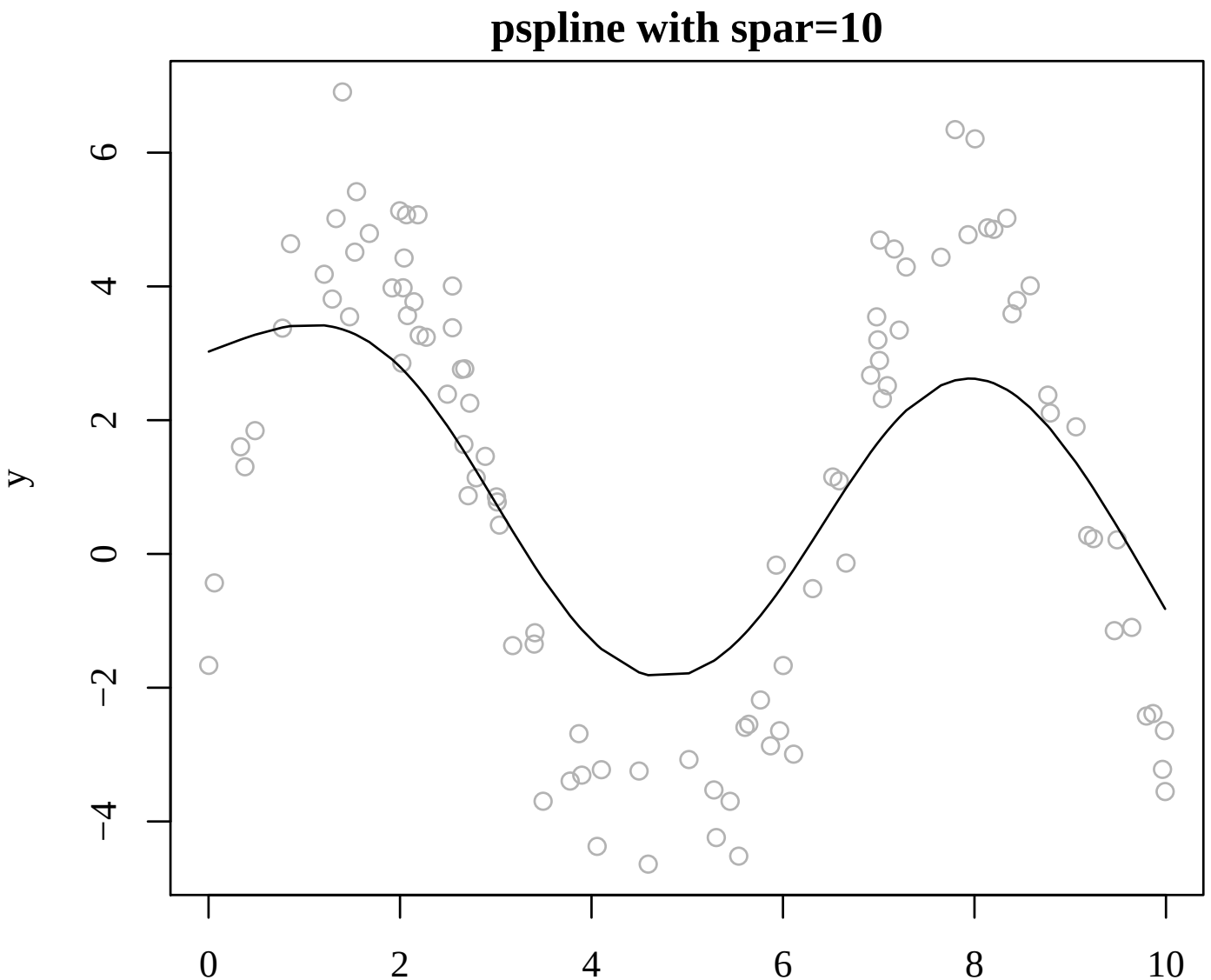


Pspline method=1, spar=10

```
psp17 <- smooth.Pspline(x, y, spar=10, method=1)
psp17
```

```
Call:
smooth.Pspline(x = x, y = y, spar = 10, method = 1)

Smoothing Parameter (Spar): 10
Equivalent Degrees of Freedom (Df): 4.555432
GCV Criterion: 3.67737
CV  Criterion: 3.791645
```

Pspline method=1, spar=10

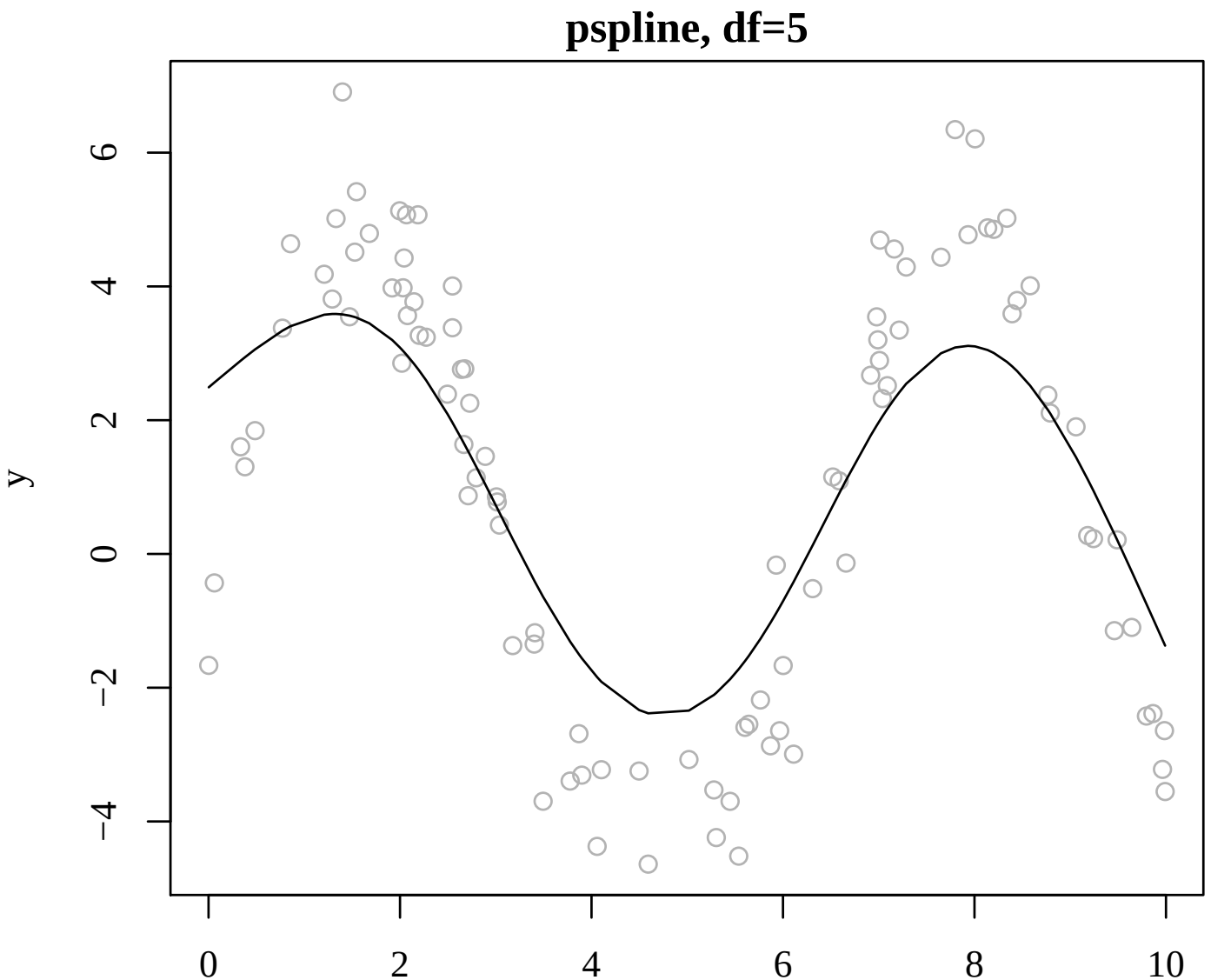


Pspline method=2, df=5 (calculates spar)

```
psp25 <- smooth.Pspline(x, y, df=5, method=2)
psp25
```

```
Call:
smooth.Pspline(x = x, y = y, df = 5, method = 2)

Smoothing Parameter (Spar): 6.236906
Equivalent Degrees of Freedom (Df): 4.995585
GCV Criterion: 2.648365
CV  Criterion: 2.745691
```

Pspline method=2, df=5

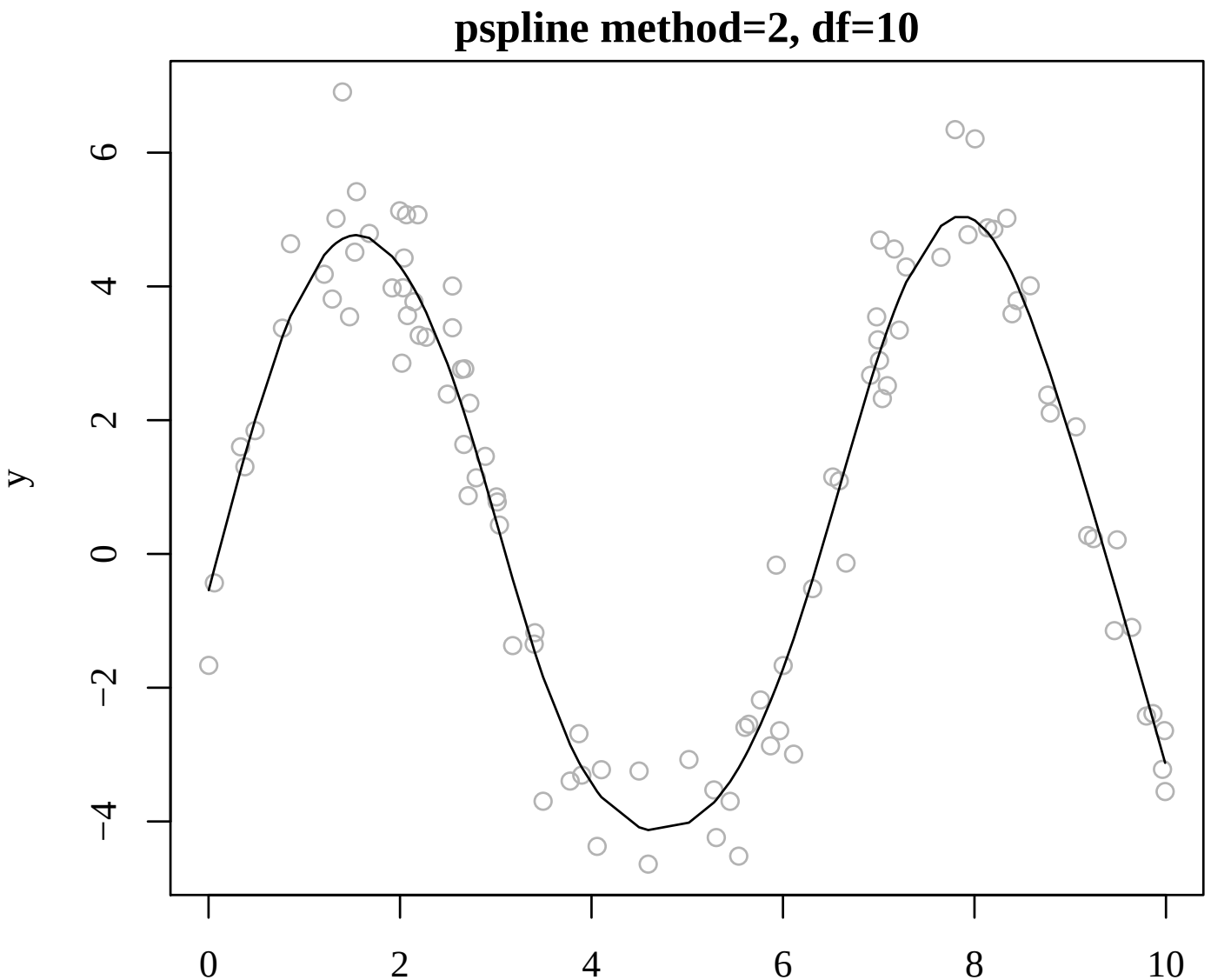


Pspline method=2, df=10 (calculates spar)

```
psp26 <- smooth.Pspline(x, y, df=10, method=2)
psp26
```

```
Call:
smooth.Pspline(x = x, y = y, df = 10, method = 2)

Smoothing Parameter (Spar): 0.2260648
Equivalent Degrees of Freedom (Df): 9.990066
GCV Criterion: 0.6836645
CV  Criterion: 0.6840962
```

Pspline method=2, df=10

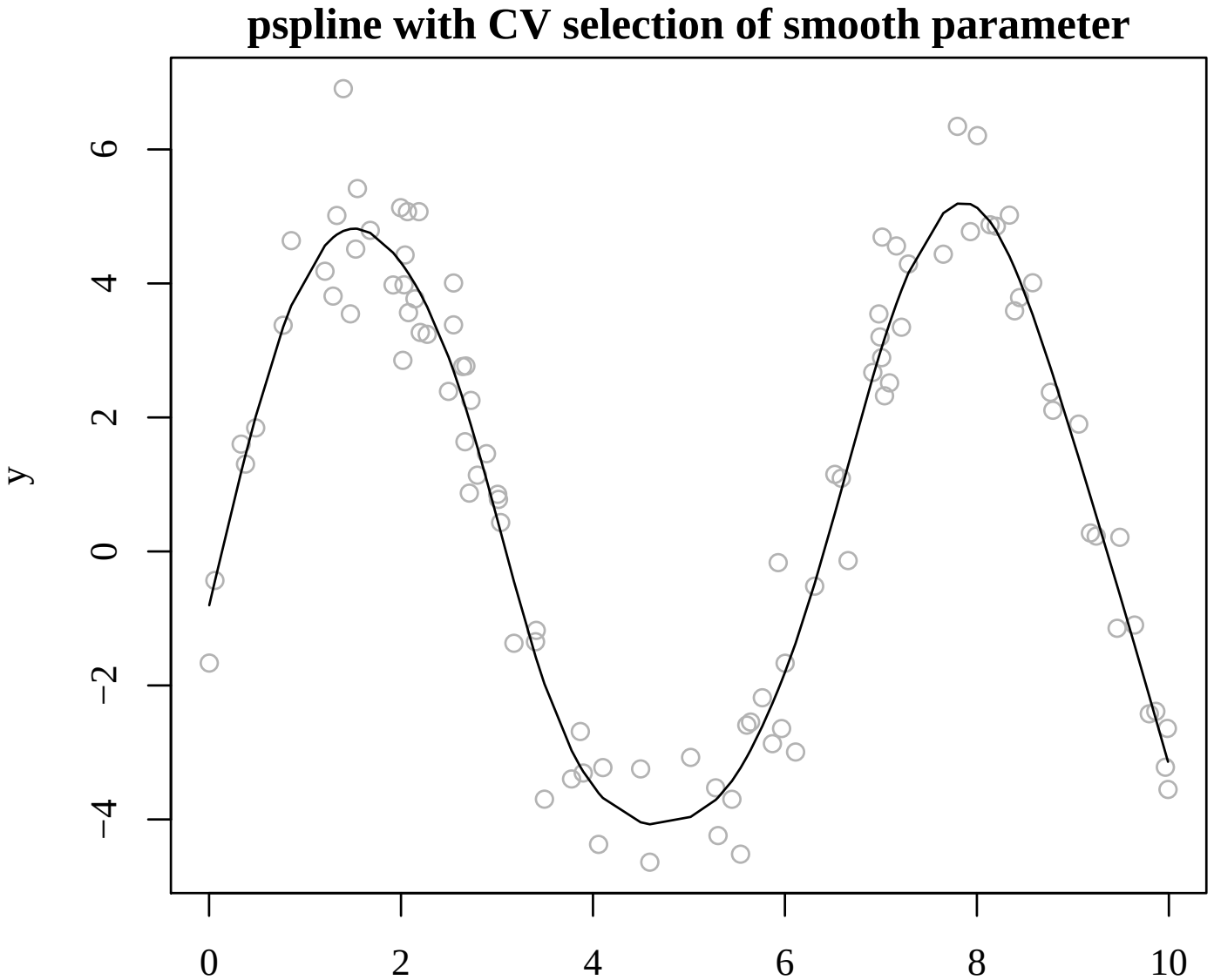


Pspline method=4; let CV decide df and spar

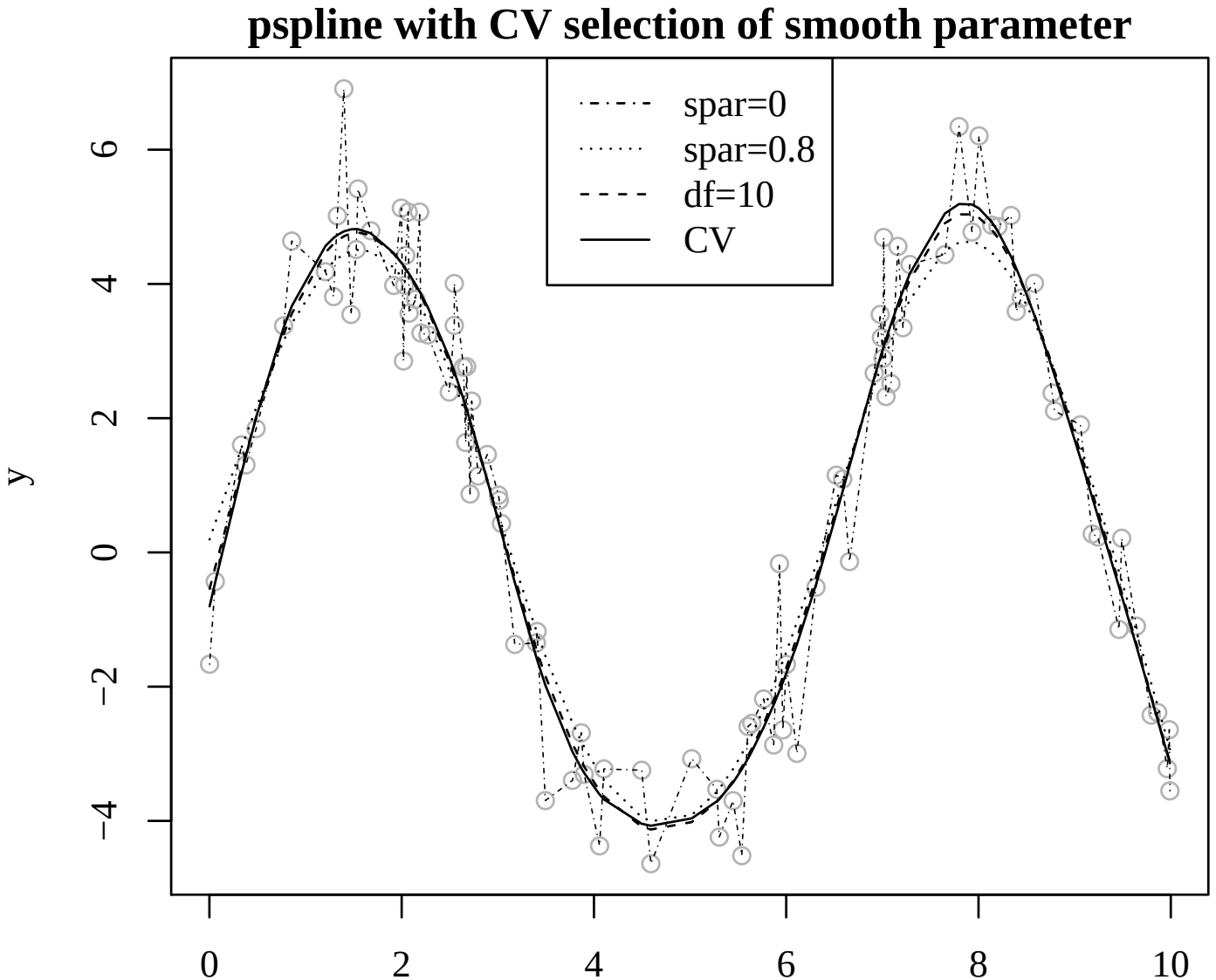
```
psp35 <- smooth.Pspline(x, y, method=4)
psp35
```

```
Call:
smooth.Pspline(x = x, y = y, method = 4)

Smoothing Parameter (Spar): 0.108699
Equivalent Degrees of Freedom (Df): 11.73974
GCV Criterion: 0.6774351
CV  Criterion: 0.673015
```

Pspline method 4, let CV decide spar (df)



Look at the Psplines side by side



Wish I could demonstrate the smooth.spline function

But I keep running into weird little mismatches between what I expect and what it does, so I'm leaving this reminder here.

```
Description:

     Fits a cubic smoothing spline to the supplied data.

Usage:

smooth.spline(x, y = NULL, w = NULL, df, spar = NULL,
cv = FALSE, all.knots = FALSE, nknots = NULL,
keep.data = TRUE, df.offset = 0, penalty = 1,
control.spar = list())

df: the desired equivalent number of degrees of freedom (trace of
    the smoother matrix).

spar: smoothing parameter, typically (but not necessarily) in (0,1].
     The coefficient lambda of the integral of the squared second
     derivative in the fit (penalized log likelihood) criterion is a
     monotone function of 'spar', see the details below.
```

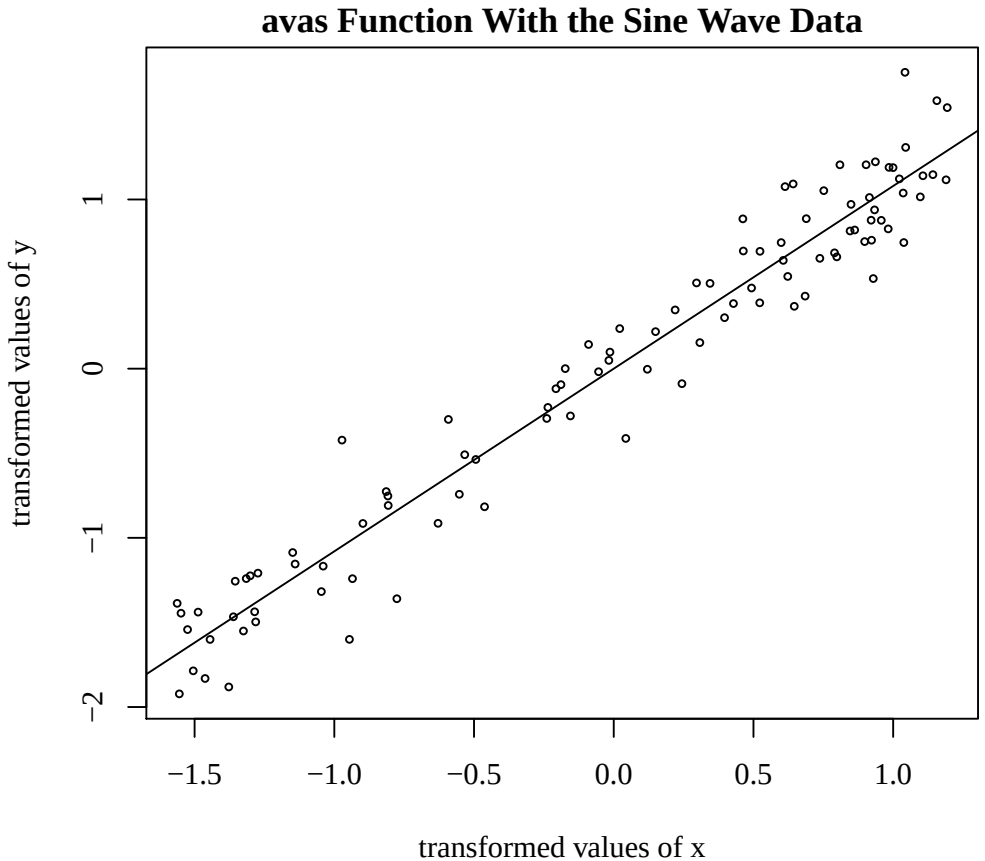
Outline

Bend the Line or Re-Number the Data. Same Thing?

Tibshirani, Rob (1987), "Estimating Optimal Transformations for Regression". Journal of the American Statistical Association 83, 394.
Outlines a transformation process that stretches and squishes the data into a scatterplot suitable for a linear regression with homogeneous variance.
R package: acepack .

Bend the Line or Re-Number the Data. Same Thing?

The transformed x and y are quite pleasantly linear and the linear model fits very nicely.

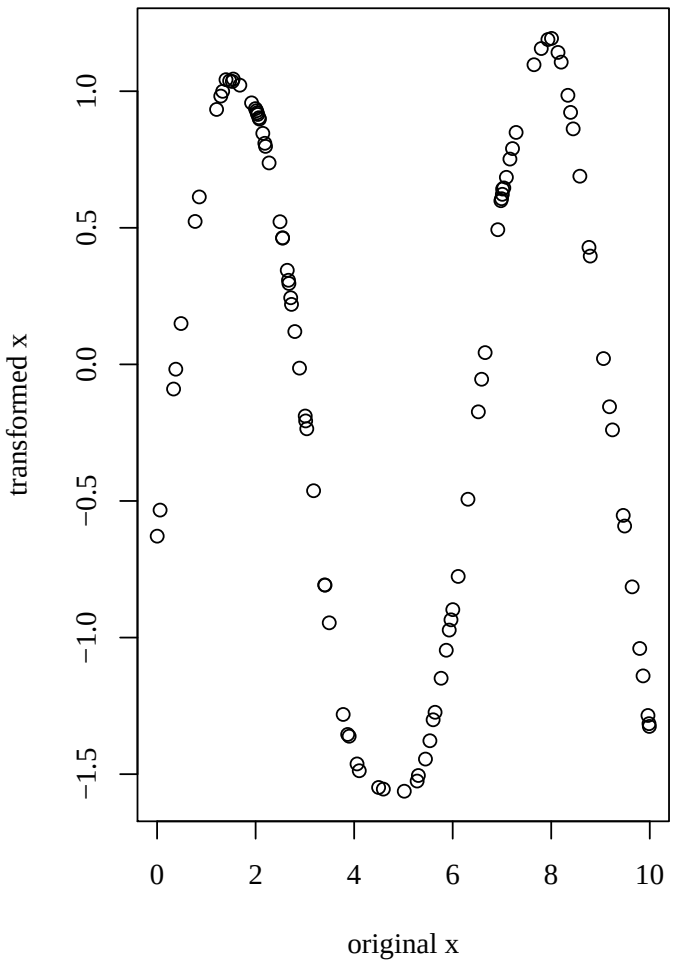


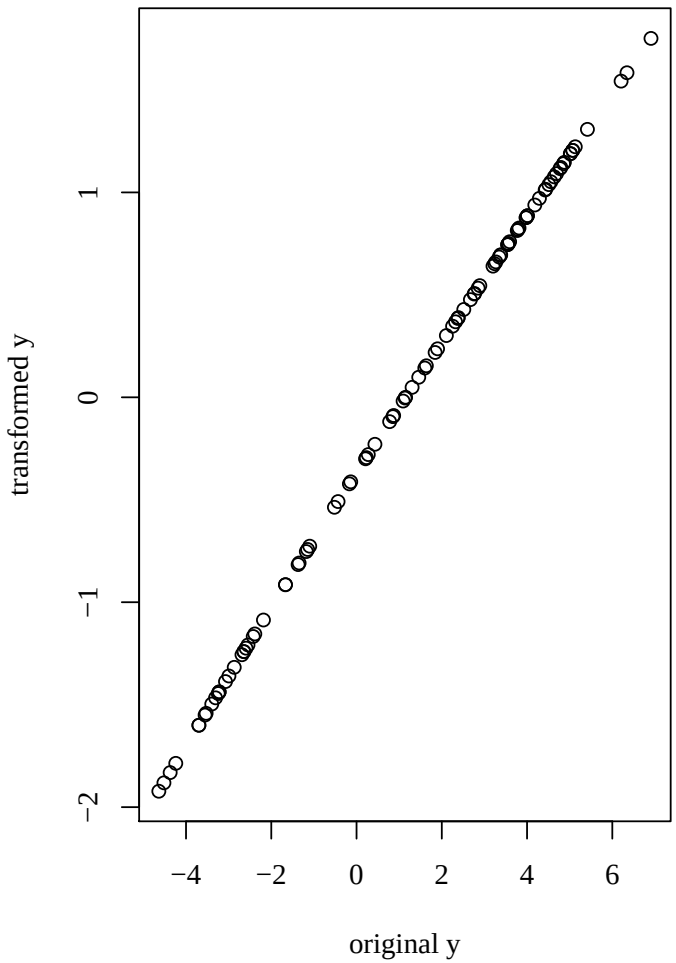
W

Why Does That Work?

What do we have to do to x and y in order to achieve such a beautiful finding?

Why Does That Work? ...





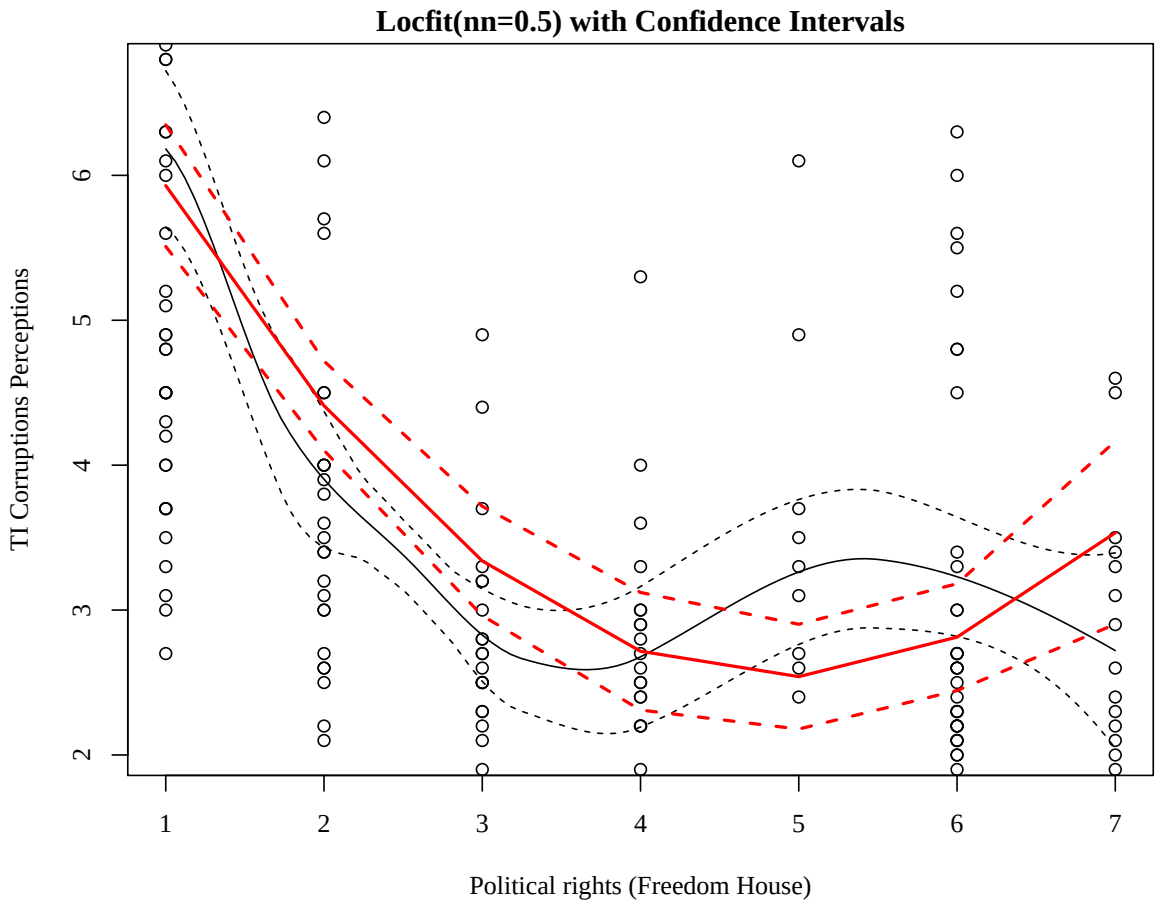
Would you rather bend the line, or bend the data, or both?

Why Does That Work? ...

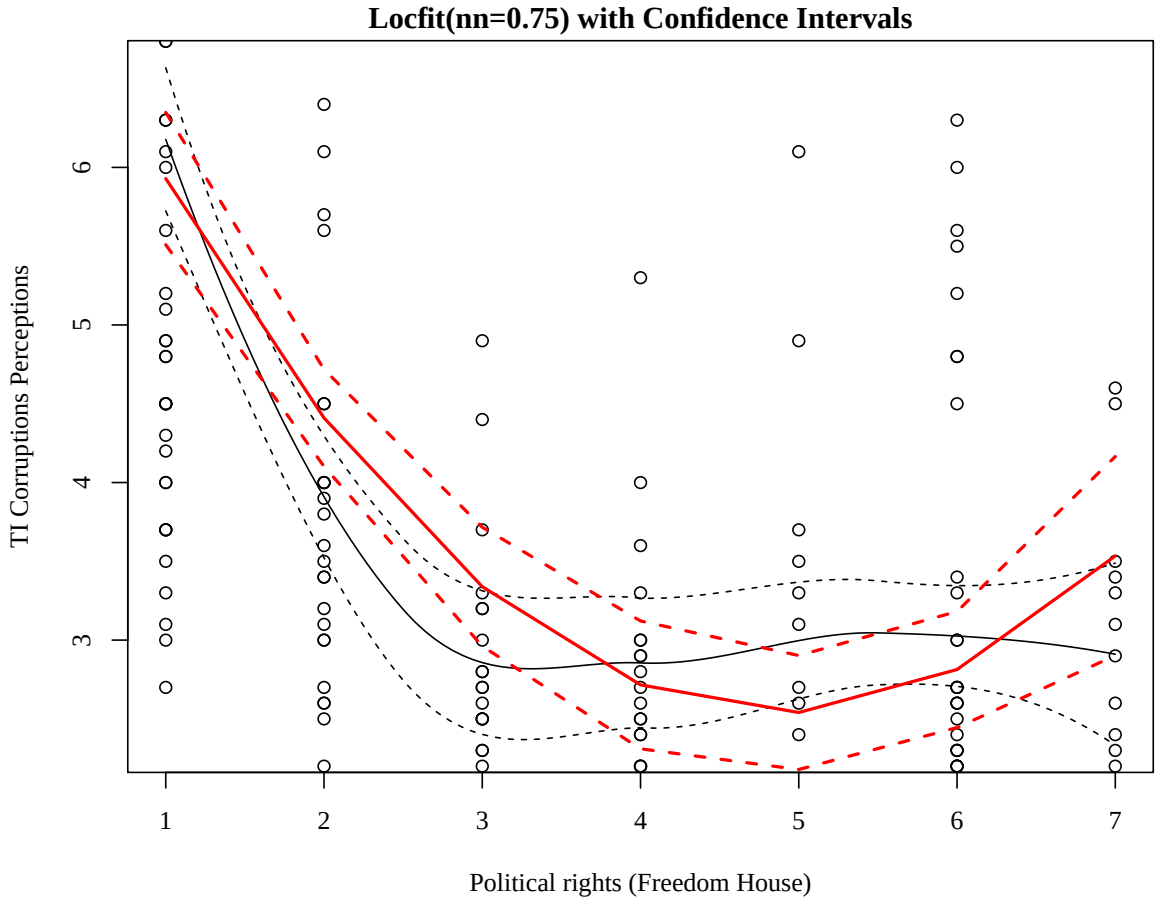
The avas procedure can be applied to many independent variables, and the user can require the program to leave some untransformed or subject some only to monotonic transformations.

Outline

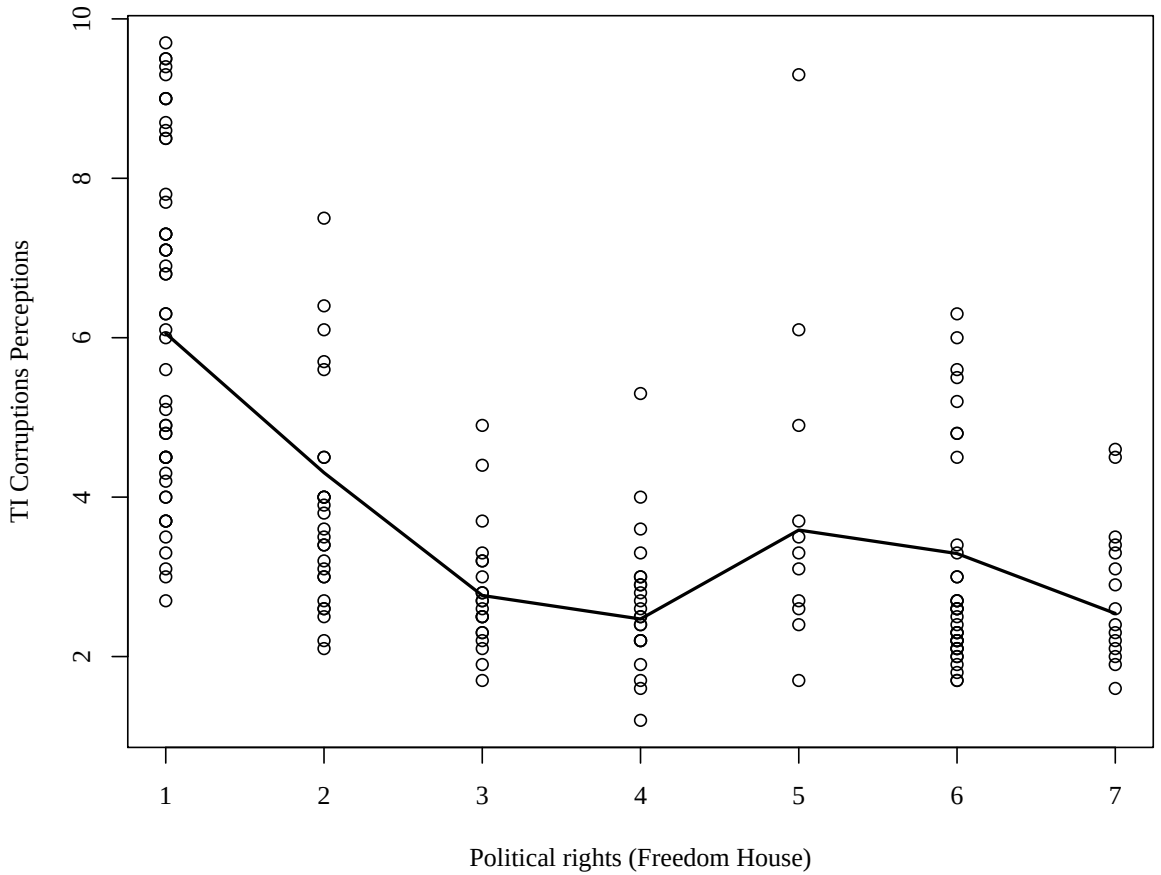
Quadratic vs Loess



Corruption and political freedom: Quadratic vs Loess



Corruption and political freedom: rcs

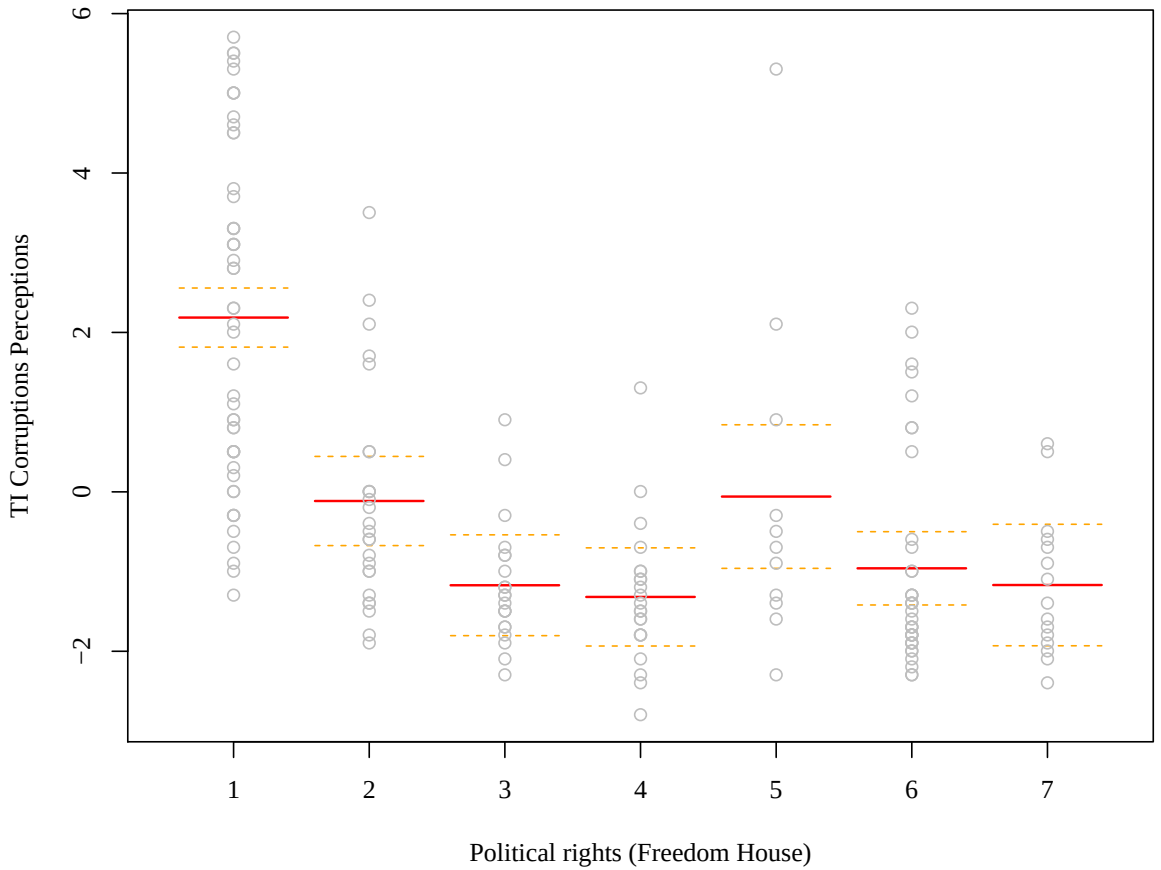


Corruption and political freedom: rcs ...

```
Call:
lm(formula = ti_cpi ~ rcs(fh_pr, 4), data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3540 -1.0921 -0.2710  0.8591  5.7132

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            7.8017     0.3501  22.285  < 2e-16 ***
rcs(fh_pr, 4)fh_pr    -1.7477     0.1764  -9.906  < 2e-16 ***
rcs(fh_pr, 4)fh_pr'    3.3199     0.5635   5.892 1.88e-08 ***
rcs(fh_pr, 4)fh_pr''  -17.2623     3.7109  -4.652 6.42e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.549 on 177 degrees of freedom
  (13 observations deleted due to missingness)
Multiple R^2:  0.4508, Adjusted R^2:  0.4415
F-statistic: 48.43 on 3 and 177 DF,  p-value: < 2.2e-16
```

Corruption and political freedom: Feeling Silly now

Freedom House political rights is a categorical variable (ordinal factor at best)



Corruption and political freedom: Feeling Silly now ...

```
Call:
lm(formula = ti_cpi ~ fh_prf, data = dat)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4820 -0.9361 -0.2808  0.6733  5.3636

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.1820     0.2179  28.365  < 2e-16 ***
fh_prf2      -2.3012     0.3726  -6.176 4.51e-09 ***
fh_prf3      -3.3582     0.4007  -8.380 1.72e-14 ***
fh_prf4      -3.5047     0.3943  -8.889 7.63e-16 ***
fh_prf5      -2.2456     0.5132  -4.375 2.08e-05 ***
fh_prf6      -3.1459     0.3369  -9.339  < 2e-16 ***
fh_prf7      -3.3553     0.4537  -7.396 5.71e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.541 on 174 degrees of freedom
  (13 observations deleted due to missingness)
Multiple R^2:  0.4656,  Adjusted R^2:  0.4472
F-statistic: 25.27 on 6 and 174 DF,  p-value: < 2.2e-16
```

Outline

Problems

1. 'Get the "cystfibr" dataset from the DataSets folder. Let's predict "weight" from "height". (I'm not a medical doctor, I don't know that height and weight really should be related. Its just some data I have.)
    1. Fit an OLS model to the linear relationship, make a standard plot.

```
dat <- read.table("cystfibr.txt", header=T)
plot(weight ~ height, data = dat)
mod1 <- lm(weight ~ height, data = dat)
summary(mod1)
abline(mod1)
```

    2. Fit a loess curve to the height-weight data. You can try loess or locfit for that. Either way, don't forget you have to decide on the "span" and whether your local regressions are linear or quadratic. Ordinarily, I'd run a series of commands like

Problems ...

```
lfit <- loess(weight ~ height, data = dat,
    span = 0.5, degree = 2, family = "
    gaussian")
dat <- dat[order(dat$height) ,]
lopred <- predict(lfit, newdata = dat)
plot(weight ~ height, data = dat)
lines(dat$height, lopred)
abline(mod1, lwd = 0.7, lty=2)
legend("topleft", legend = c("loess","OLS")
    , lwd = c(1,0.7), lty = 1:2)
```

I recently learned there is a short-cut to create the scatter with loess line, you might try it out.

```
with(dat, scatter.smooth( height, weight))
abline(mod1, lwd = 0.7, lty = 2)
legend("topleft", legend = c("loess","OLS")
    , lwd = c(1,0.7), lty = 1:2)
```

Problems ...

I warn you, scatter.smooth does not use the same settings as loess by default, so you do need to read the help page if you want the 2 loess curves to match. I'm not thrilled about that. You might be smarter just to use loess by itself.
After all that work, here's my simple question. Which would you advocate. The OLS fit or the loess fit? What are the best arguments you can make for the one you prefer? I don't know that there is a "right" answer for this question, it is open for argument. While I was experimenting with this, I found the output of summary(lfit) and summary(mod1) to be informative.

3. Let's try a natural spline predictive curve. Here's the way I coded it.

Problems ...

```
mod4 <- lm( weight ~ ns(height, df = 4),
    data = dat)
summary(mod4)
#dang. Should have sorted dat by height
    first.
dat <- dat[order(dat$height), ]
mod4pred <- predict(mod4, newdata = dat)
plot(weight ~ height, data = dat)
lines(dat$height, mod4pred, col = green,
    lty = 4, lwd = 2)
```

2. We might as well waste a little more time on the cystfibr height and weight data.
    1. Fit the quadratic model . If you can plot the predicted values from that on the same graph with loess, I bet you'd have something worth debating. Would you make an argument in favor of loess or the quadratic model? Why?

Problems ...

```
dat$heightsq <- dat$height * dat$height
mod2 <- lm(weight~height + heightsq, data =
    dat)
summary(mod2)
heightseq <- seq(min(dat$height), max(dat$
    height), length.out = 100)
weightpred <- predict(mod2, newdata =
    data.frame(height = heightseq, heightsq
    = heightseq*heightseq))
plot(weight ~ height, data = dat)
lines(heightseq, weightpred, lty = 4, col =
    "red", lwd = 2)
```

2. I wonder how the quadratic regression changes if you center "height". Replace it with this.

Problems ...

```
dat$heightc <- dat$height - mean(dat$height
    , na.rm = TRUE)
## same as dat$heightc <- scale(dat$height,
     scale = FALSE)
```

I wonder 1) how the regression estimates change, when we replace height with heightc, 2) whether the plot changes, and 3) whether you think there is a meaningful difference in the 2 fits.

3. In a nutshell, here is a big question. Why would somebody rather have a set of predictions from a "loess smooth" than a "natural cubic spline" or a "smoothing spline."?
    1. All smoothers use "degrees of freedom" to calculate predictions, in the sense that they use up some of the information.
4. I will keep thinking hard for more interesting examples.