CSSS 569: Visualizing Data

Exploratory Data Analysis: The Border between Exploration and Modeling

Christopher Adolph

Department of Political Science
and
Center for Statistics and the Social Sciences
University of Washington, Seattle

Exploratory Data Analysis

Exploratory Data Analysis refers to a approach pioneered by statistician John Tukey

Emphasis on

- letting the data speak for itself
- non-parametric models
- tools for exploring high-dimensional datasets

Relatively little used in social science, where we prefer parametric models

Dangers: sometimes parametric models are fragile, and EDA could help show this

Without preliminary EDA, finding the right parametric specification may be harder

Basic Exploratory Data Analysis

We've covered some of the basic tools of EDA:

- scatterplot matrices
- conditional plots (lattice/trellis)
- histogram-like plots

Today, the border between EDA & modeling

Next time, (harder) EDA for high dimensional data

Why is this a topic for visualization?

Simple answer: only way to understand some fits is visually

Deeper answer: visual EDA complements and supports better statistical modeling

Henceforth, our goal will be to use VDQIs to improve our statistical modeling and inference

Complement to your other coursework

The border between EDA and modeling

Models make simplifying assumptions

The precision of model estimates comes from these assumptions

Wanted: Assumptions "pretty close" to the behavior of the data

How do we check? non-parametric & semi-parametric EDA

Approach: partially relax modeling assumptions, and see if data support simplification

E.g., let the line wiggle if it wants; then check for approximate linearity

A framework for probability models of data

Introducing graphical techniques for a wide variety of statistical methods

$\rightarrow$ We need a language to refer to diverse probability models

Most models have a stochatic component:

$$\mathbf{y} \sim f_{\mathcal{D}}(\mu, \alpha)$$

and a systematic component

$$\mu = g(\mathbf{X}, \beta)$$

$\mathbf{y}$ is the data vector of interest. $\mathbf{X}$ is a matrix of covariates

$f_{\mathcal{D}}$ is a probability density function for distribution $\mathcal{D}$

μ is (usually) the expected value. α is a "nuisance" parameter vector

β is a parameter vector associated with the covariates

A framework for probability models of data

$$\mathbf{y} \sim f_{\mathcal{D}}(\mu, \alpha)$$
$$\mu = g(\mathbf{X}, \beta)$$

nests most (if not all) models you know.

Linear regression:	$\mathbf{y}$	$\sim$	$f_{\text{Normal}}(\mu, \sigma^2)$
(continuous data)	μ	$=$	$\mathbf{X}\beta$
Logit:	$\mathbf{y}$	$\sim$	$f_{\text{Bernoulli}}(\mu)$
(binary data)	μ	$=$	$[1 - \exp(-\mathbf{X}\beta)]^{-1}$
Poisson:	$\mathbf{y}$	$\sim$	$f_{\text{Poisson}}(\lambda)$
(count data)	λ	$=$	$\exp(\mathbf{X}\beta)$

and so on.

A framework for probability models of data

Note the parallel to the notation of Generalized Linear Models (GLMs)

For example, we can write logit equivalently

$$\mathbf{y} \sim f_{\text{Bernoulli}}(\mu) \qquad \mathbf{y} \sim f_{\text{Bernoulli}}(\mu)$$

$$\mu = g(\mathbf{X}\beta) \qquad \mathrm{g}^{-1}(\mu) = \mathbf{X}\beta$$

$$\mu = [1 - \exp(-\mathbf{X}\beta)]^{-1} \qquad \log[\mu/(1-\mu)] = \mathbf{X}\beta$$

The framework on the left applies to just about any distribution you will encounter

It is equivalent to the form on the right, which is customary for GLMs.
$g(\cdot)$ is called the link function in the GLM context

GLMs are a class of models for which $f_{\mathcal{D}}$ is a member of the exponential family (which includes the Normal, Binomial, Gamma, and Poisson; see `family()`)

A framework for probability models of data

Nice aspects of the framework:

- Very general. Works for most any model, and most any estimation method (Maximum likelihood, Bayesian inference, etc.).
- Focuses on the data of interest, y
- Reduces attention to β, which is just a cog in the machine that turns X into y
- For different models, β has different, usually non-obvious interpretations, but
- In each case y, $\mathrm{E}(y|\mathbf{X})$, & $\mathrm{E}(y_c - y_d|\mathbf{X}_c, \mathbf{X}_d)$ have simple, substantively interesting interpretations

Under this framework, our problem is simply to explain how y, or an intresting function of y, varies as we change $\mathbf{X}$.

That's what modeling is for.

That's what we want to visualize.

Unpacking the framework

Let's focus on

$$\mu = g(X, \beta)$$

Here are some typical specifications of the RHS of this equation in LS models

$$\mu = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

$$\mu = \beta_0 + \beta_1 X_1^2 + \beta_2 X_2 + \dots$$

$$\mu = \beta_0 + \beta_1 X_1 + \beta_2 \log(X_2) + \dots$$

A general form for these transformations:

$$\mu = \eta + h_1(\beta_1, x_1) + h_2(\beta_2, x_2) \dots$$

Unpacking the framework

A general form for these transformations:

$$\mu = \eta + h_1(\beta_1, x_1) + h_2(\beta_2, x_2) \ldots$$

which we can write compactly as:

$$\mu = \eta + \sum_{k=1}^{p} h_k(\beta_k, x_k)$$

$h_k(\cdot)$ could be any arbitrary function.

We could multiply β by the square of x_k

We could multiply β by the log of x_k

We could multiply β by the average of the nearest t neighboring x's

Unpacking the framework

$$\mu = \eta + \sum_{k=1}^{p} h_k(x_k)$$

What if we don't even need a β_k for some of our $h_k(\cdots)$?

E.g., use a running average of y over the last 3 values of x

Then we get a nonparametric specification

Note that nonparametric does not mean choiceless—
we still had to choose 3 rather than 5 values to average

But there is no parameter being *estimated*

Non-parametric models: smooths

Non-parametric "smooths", like the moving average, fall on the border between plots of the data and traditional regression models

Scatterplots leave model "fitting" up to the viewer's eyes

Regression models, such as this linear regression on polynomials:

$$y_i = \beta_0 + \beta X_{1i} + \beta_1 X_{1i}^2 + \varepsilon_i$$

or this probit regression on linear terms

$$\Pr(y_i) = F_{\text{Normal}}(\beta_0 + \beta X_{1i}, 1)$$

make assumptions the viewer must take on faith, esp. the *specification*

Choices about whether terms should enter as linear, cubed, logged, etc are often arbitrary

Smooths reduce the influence of the model on functional form, in favor of the data

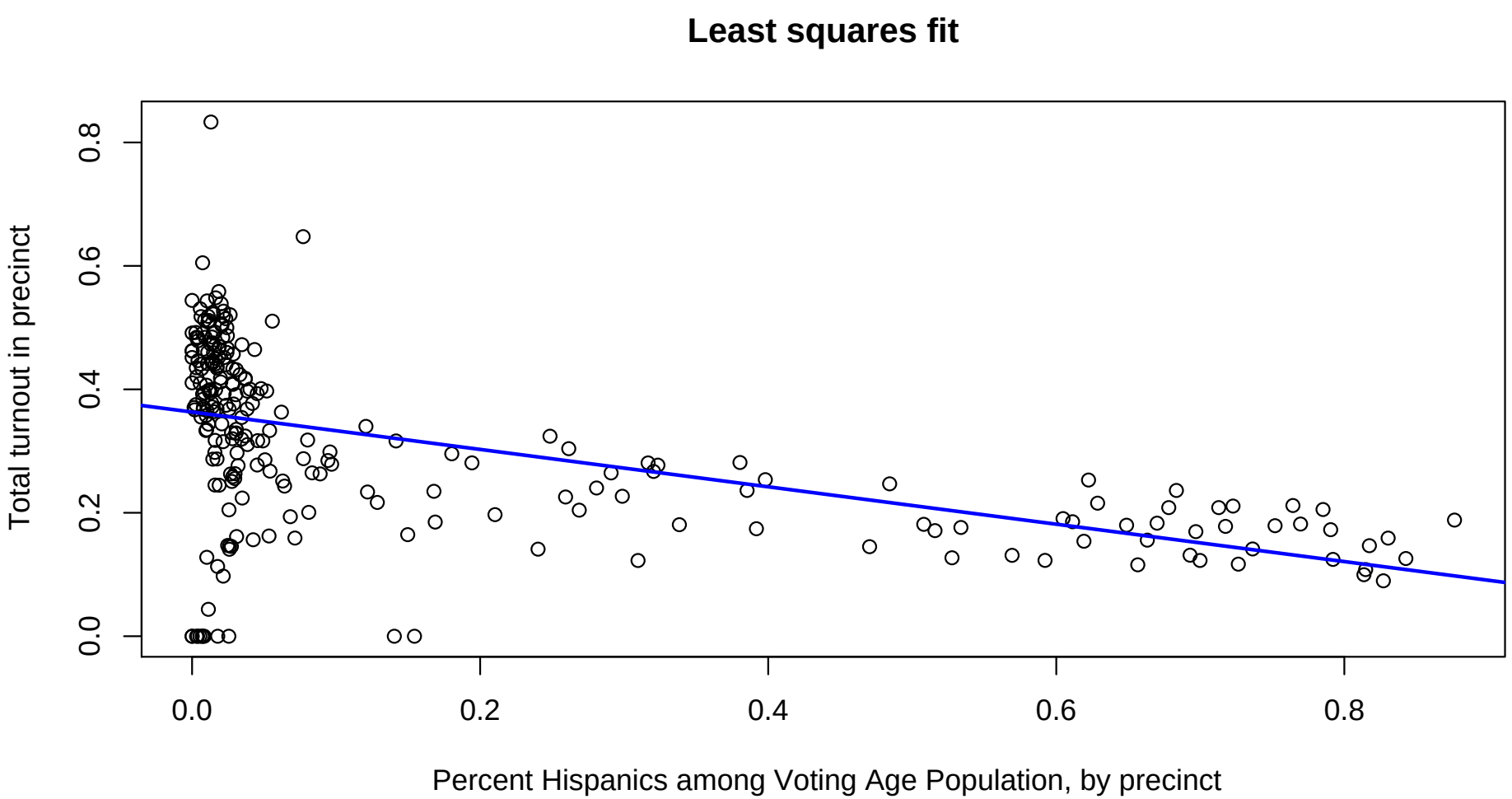
Non-parametric models: smooths

Some commonly used smoothers:

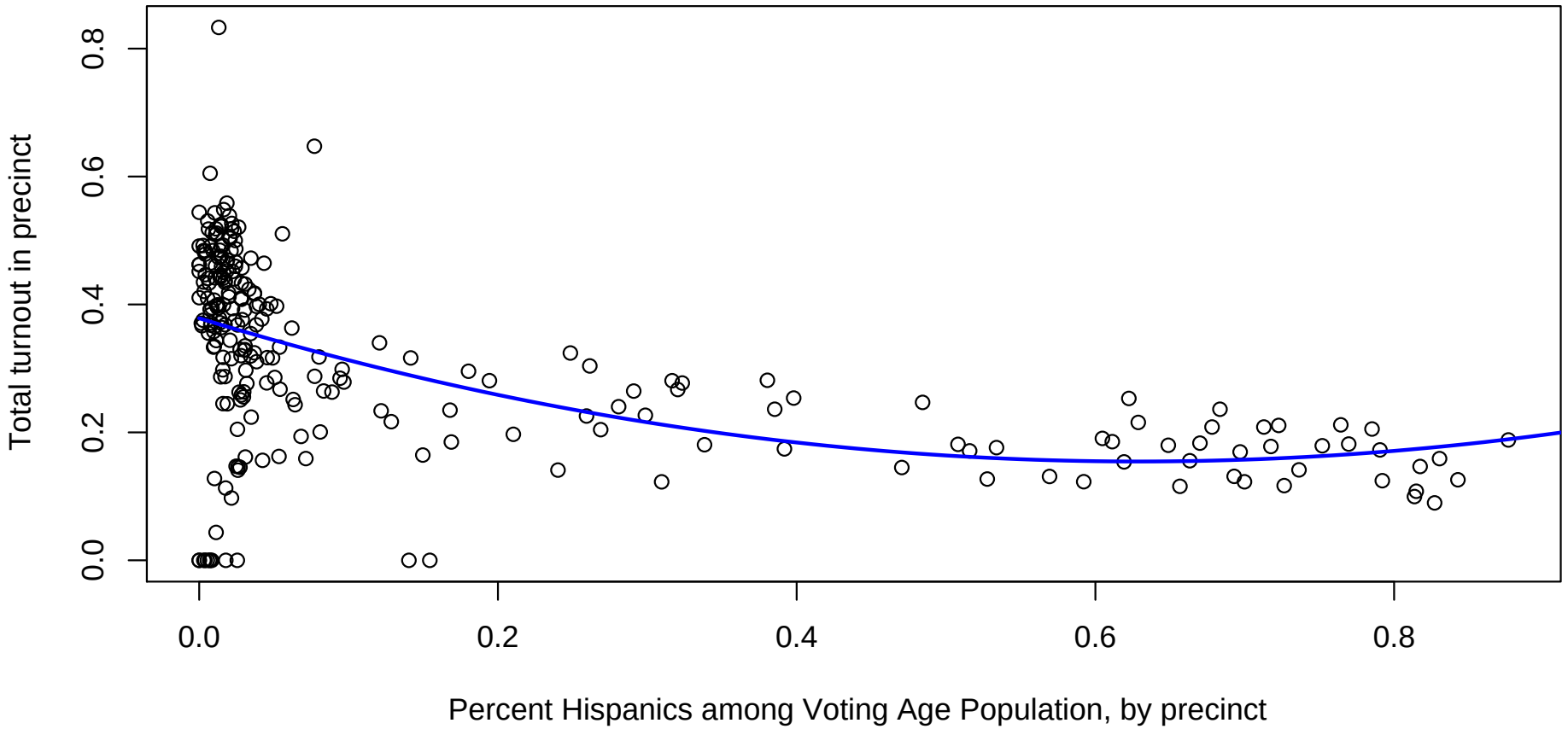
- running averages
- running medians
- loess
- splines
- kernel density (mainly useful for smoothing histograms)

Let's look at how a variety of smoothers deal with a simple dataset:

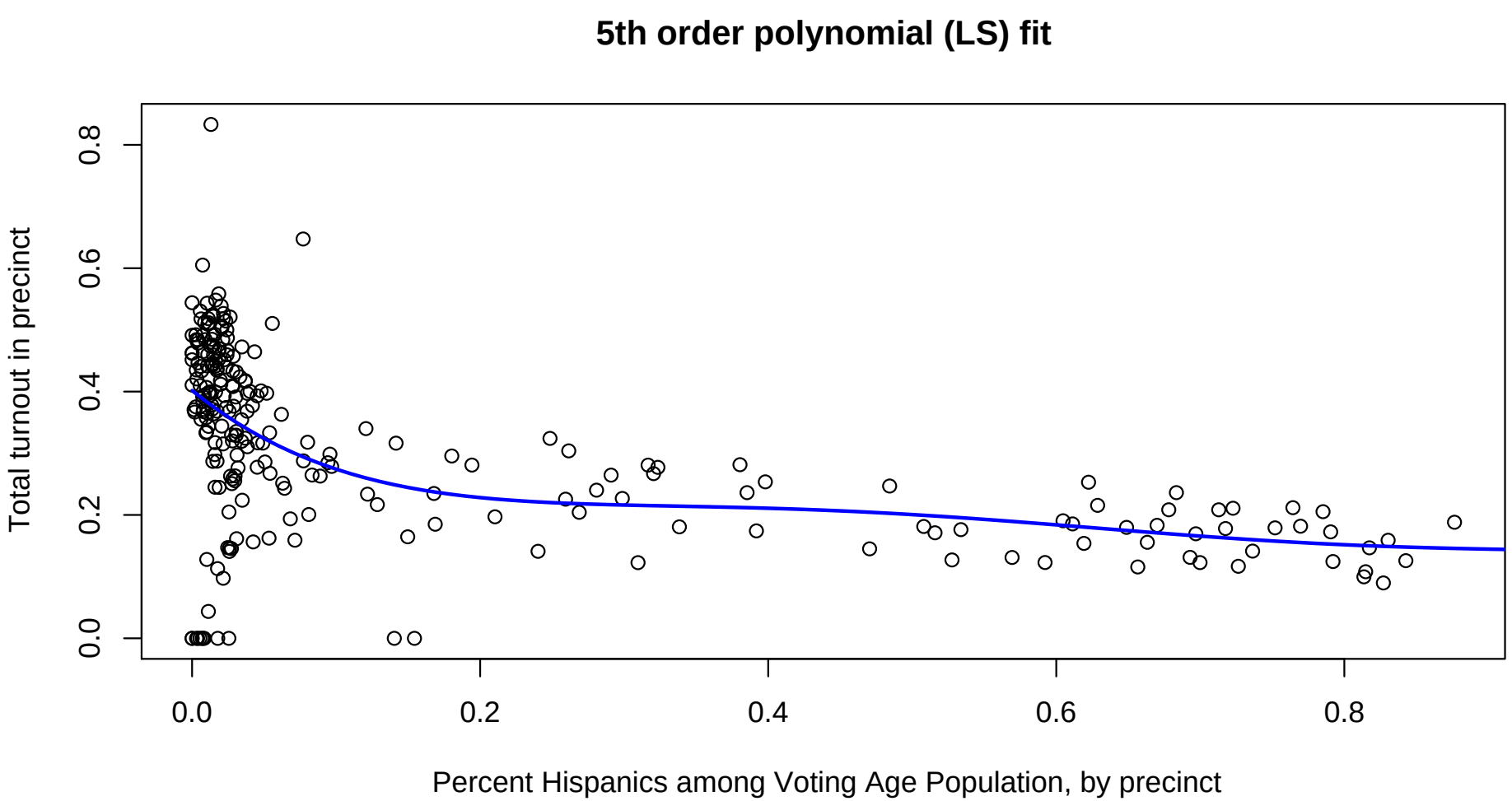
Turnout as a function of %Hispanic voters in a Pennsylvania State Senate election




```
res.lm <- lm(t~x,na.action="na.omit")
abline(res.lm$coefficients[1],res.lm$coefficients[2],col="blue",lwd=2)
```

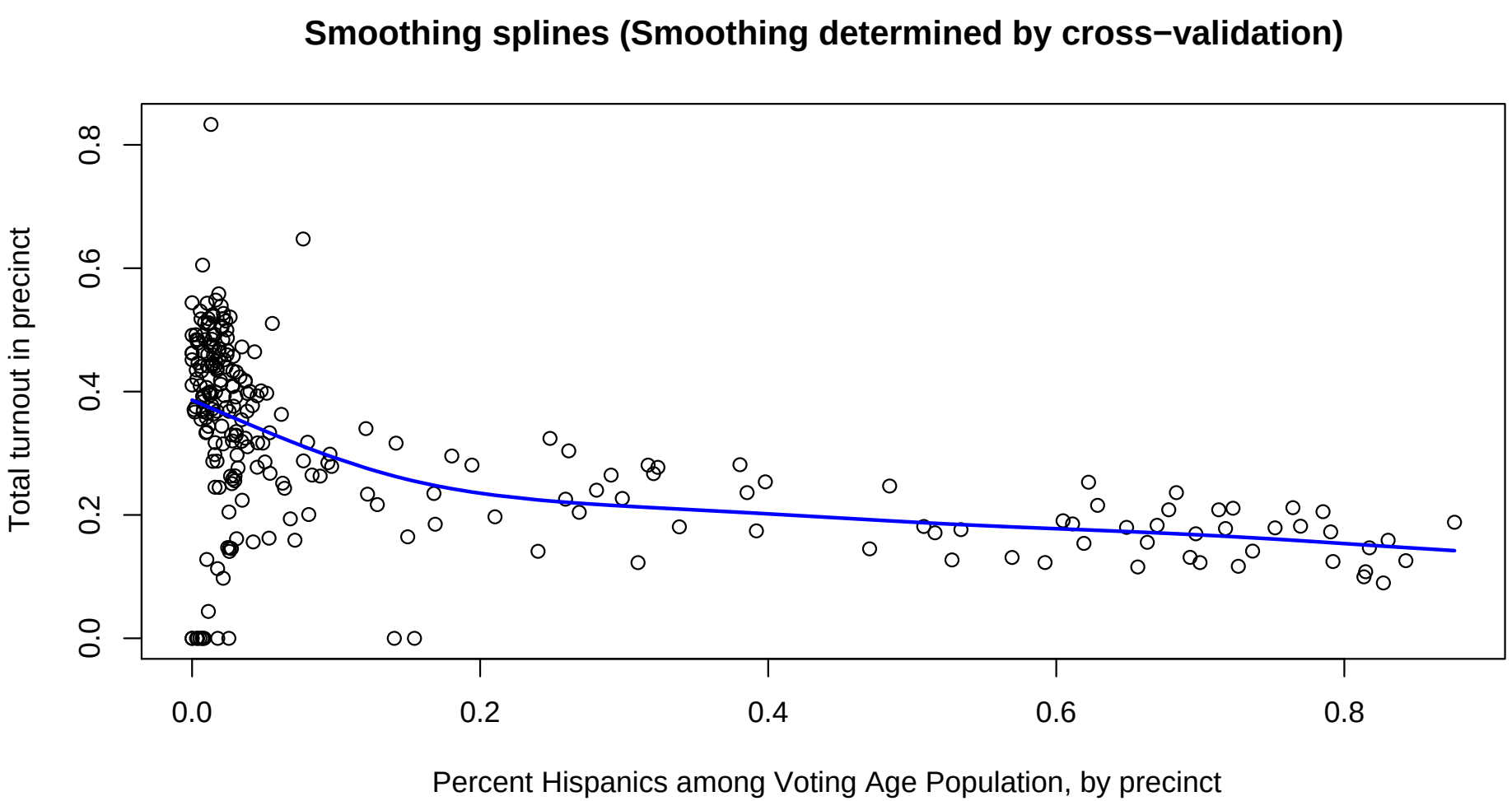



```
res.q <- lm(t~x+I(x^2),na.action="na.omit")
pred.q <- allx <- seq(0,1,0.01)
for (i in 1:length(allx)) {
  pred.q[i] <- res.q$coefficients[1] + res.q$coefficients[2]*allx[i]
               + res.q$coefficients[3]*allx[i]^2
}
```




```
res.q <- lm(t~x+I(x^2)+I(x^3)+I(x^4)+I(x^5),na.action="na.omit")
```

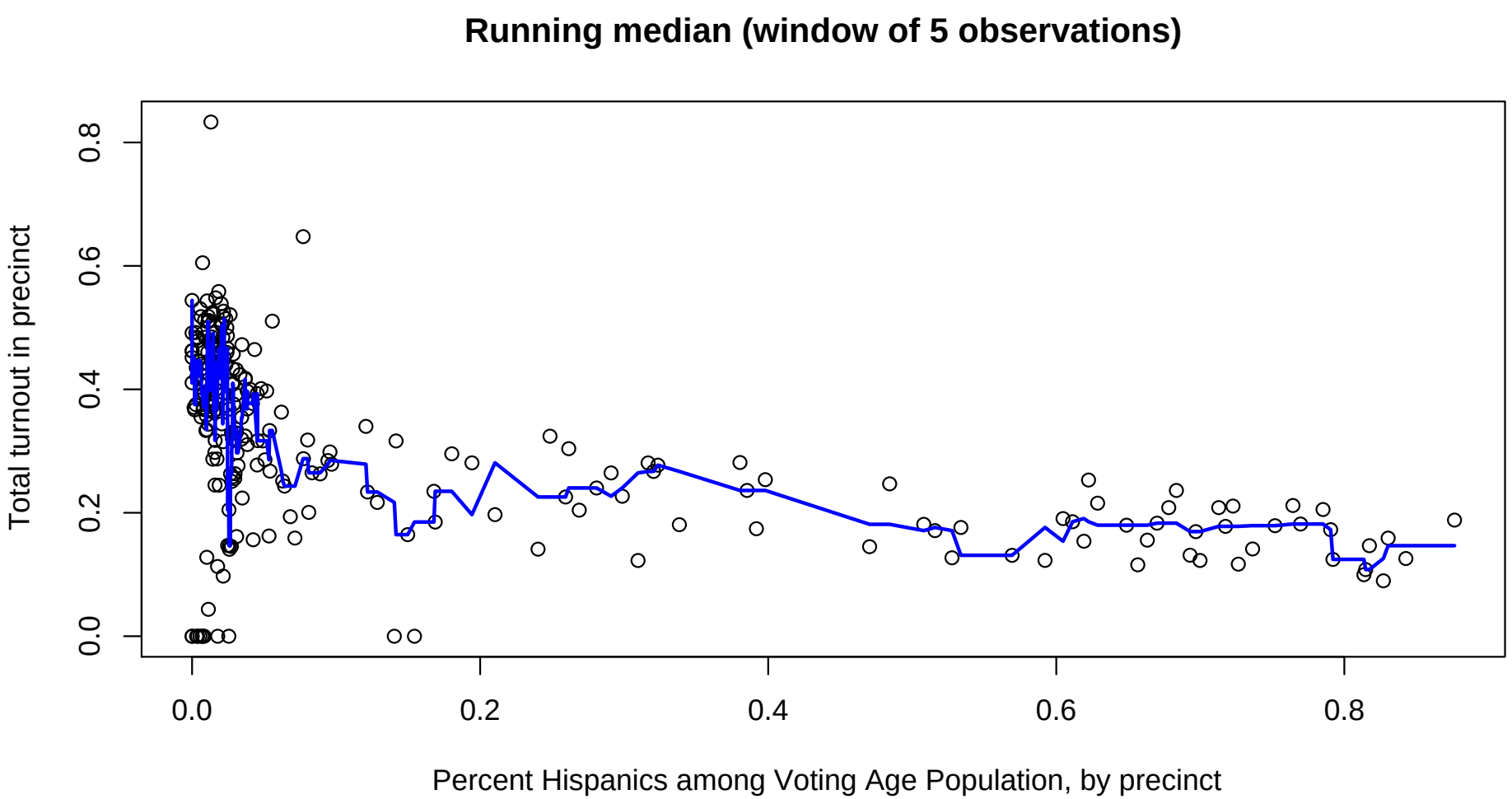
etc. Danger of overfitting with any polynomial fit.




```
lines(smooth.spline(y=t,x=x),col="blue",lwd=2)
```

We can approximate high order polynomial fits with smoothing splines

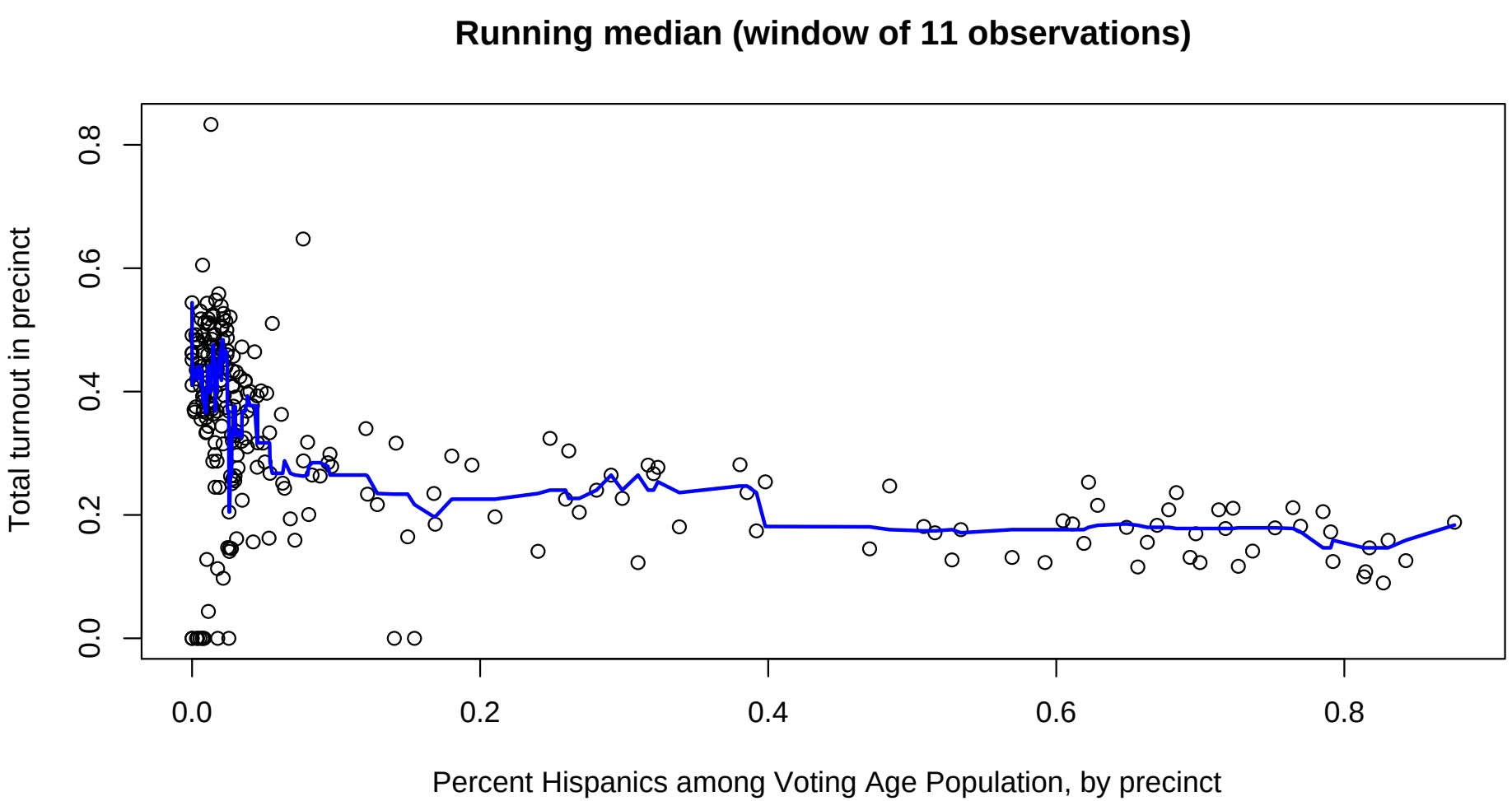
This function tries to avoid overfitting by selecting the "wiggliness" based on cross-validation



This works only if `t` and `x` both sorted by `x` *first*:

```
lines(x=x,y=runmed(t,k=5),lwd=2,col="blue")
```

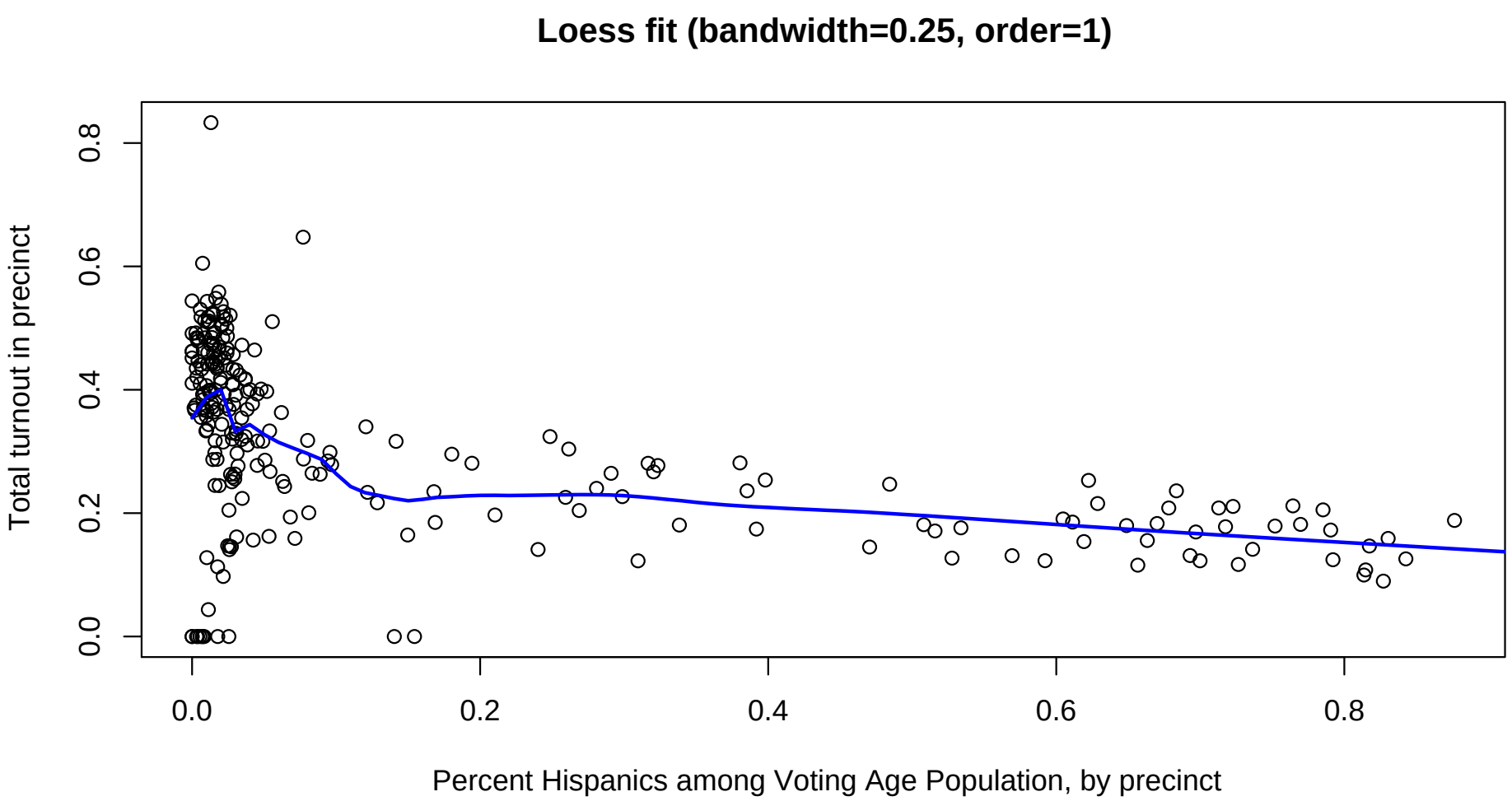
Maximum robustness. Not very smooth though.



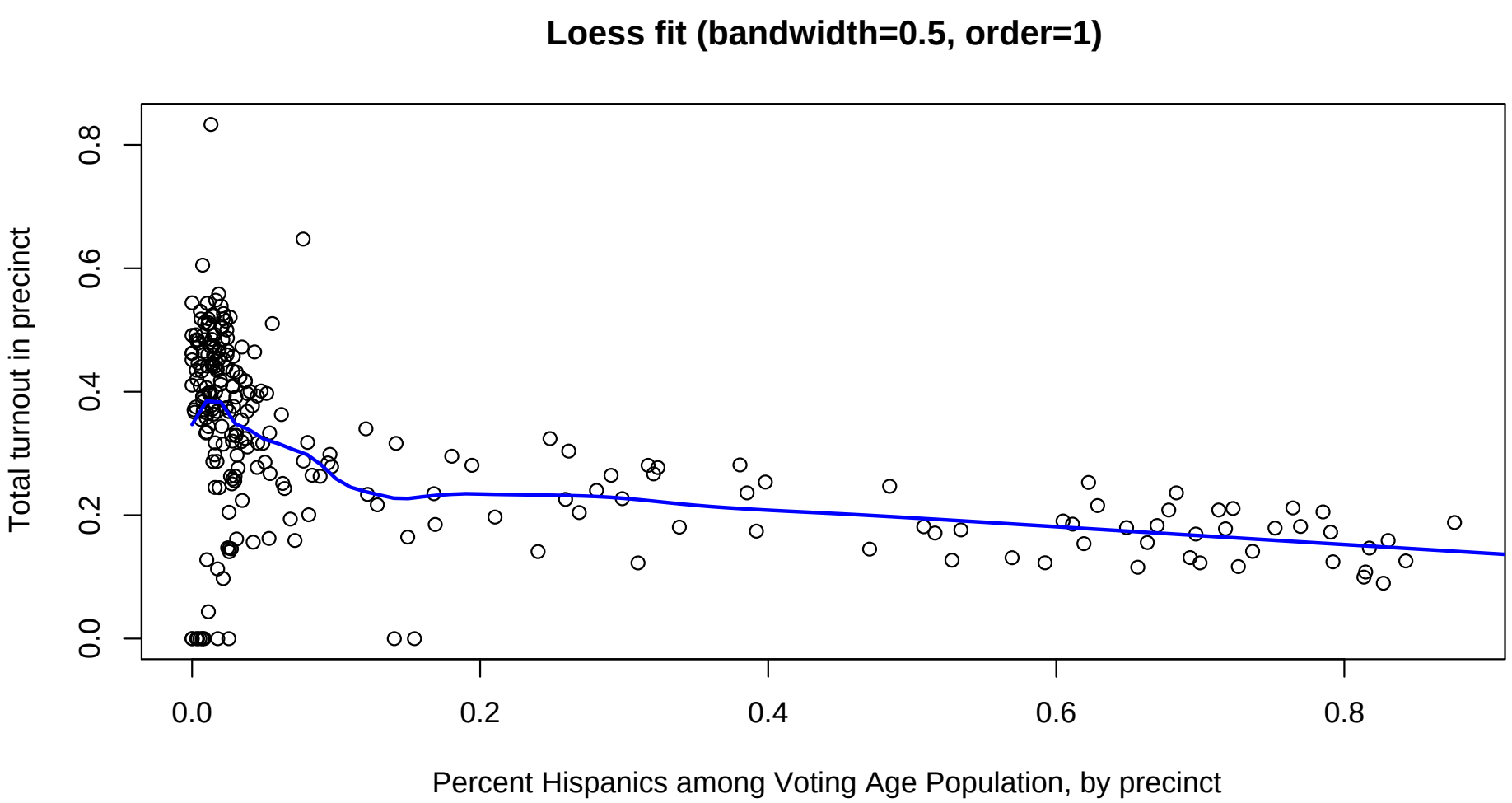
This works only if `t` and `x` both sorted by `x` *first*:

```
lines(x=x,y=runmed(t,k=11),lwd=2,col="blue")
```

Maximum robustness. A bit smoother, but we'd like another alternative

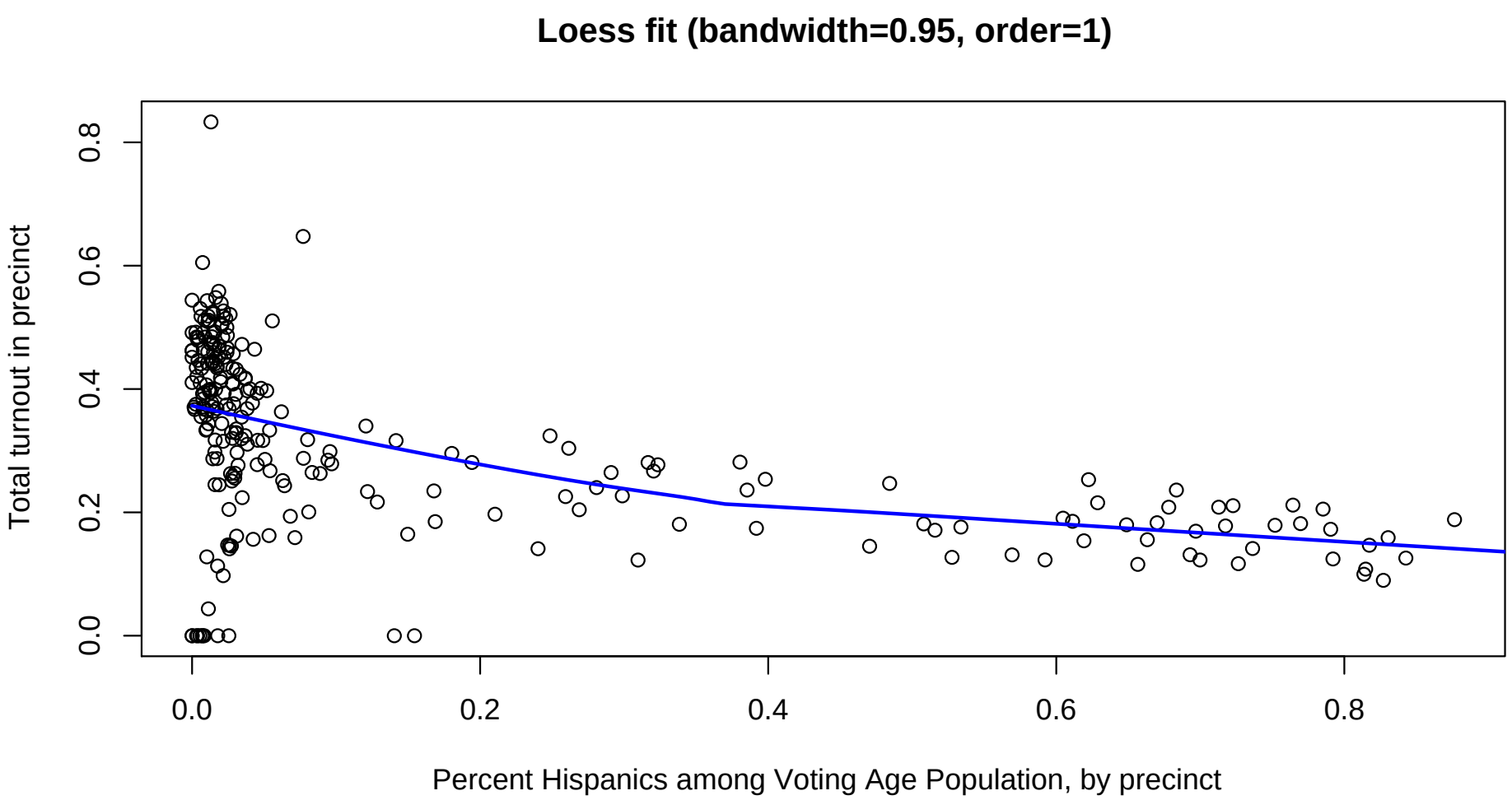



```
res.loess25 <- loess(t~x,
                     surface="direct",
                     degree=1,
                     span=0.25)
pred.loess25 <- predict(res.loess25,newdata=allx)
lines(x=allx,y=pred.loess25,col="blue",lwd=2)
```




```
res.loess50 <- loess(t~x,
                     surface="direct",
                     degree=1,
                     span=0.5)
```

Higher bandwidth $\rightarrow$ a smoother plot




```
res.loess95 <- loess(t~x,
                     surface="direct",
                     degree=1,
                     span=0.95)
```

This is about as high as we go. Note that we find a kink.

Non-parametric models: smooths

Lots of mathematical details here.

But key issue is how much flexibility to allow

In each model, there is a choice of how flexible the fit should be

- for loess and kernel, the bandwidth;
- for splines, the degree of smoothing and number of knots
- for running averages, the number and period of the averages

Loess

How do we perform this fit? And why is there no β?

$$\hat{y} = f_{\text{loess}}(x)$$

Interactive version of the following:

`demonstrations.wolfram.com/HowLoessWorks`

Loess is locally weighted polynomial regression

To fit n datapoints y given x using loess:

1. Choose a sequence x^* at which to calculate loess fits y^*

Loess is locally weighted polynomial regression

To fit n datapoints y given x using loess:

1. Choose a sequence x^* at which to calculate loess fits y^*

2. Choose a smoothing parameter α

Loess is locally weighted polynomial regression

To fit n datapoints y given x using loess:

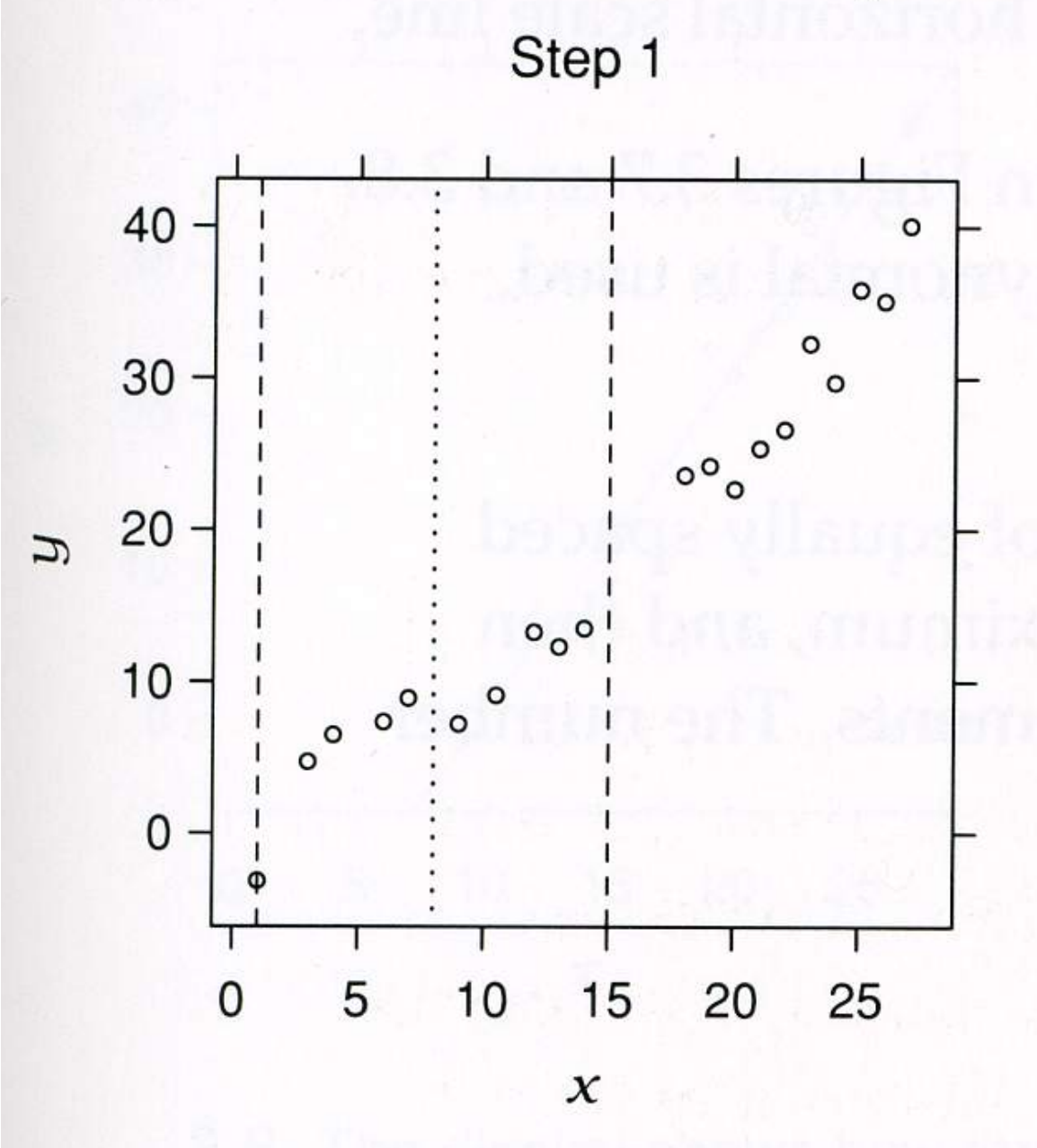
1. Choose a sequence x^* at which to calculate loess fits y^*
2. Choose a smoothing parameter α
3. Choose a polynomial order λ; usually 1 or 2

Loess is locally weighted polynomial regression

To fit n datapoints y given x using loess:

1. Choose a sequence x^* at which to calculate loess fits y^*
2. Choose a smoothing parameter α
3. Choose a polynomial order λ; usually 1 or 2
4. For each x^*,
   (a) Select the $\alpha n/2$ nearest neighbors to x^*,

Select the $\alpha n/2$ nearest neighbors to x^*



Loess is locally weighted polynomial regression

To fit n datapoints y given x using loess:

1. Choose a sequence x^* at which to calculate loess fits y^*

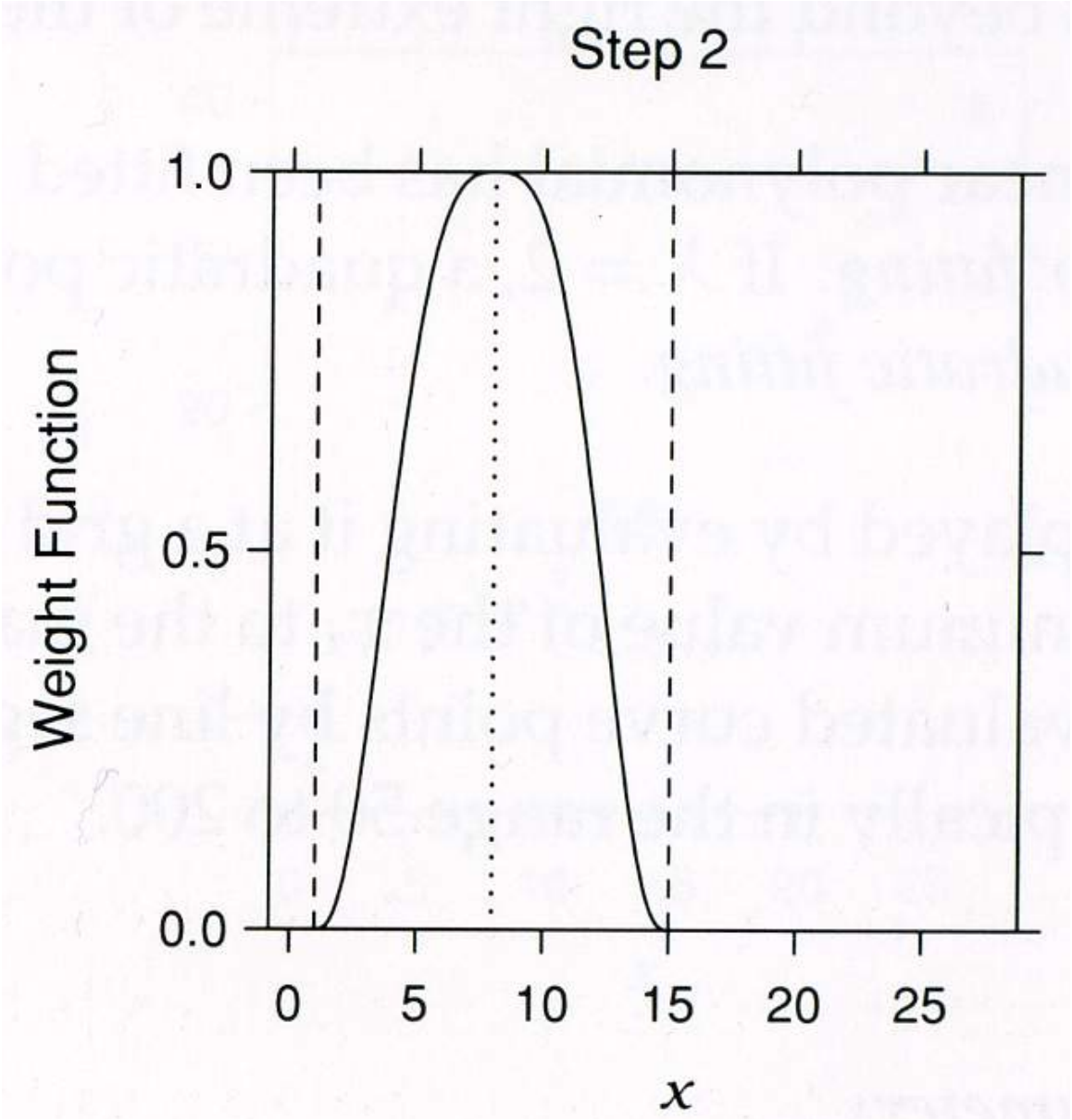
2. Choose a smoothing parameter α

3. Choose a polynomial order λ; usually 1 or 2

4. For each x^*

   (a) Select the $\alpha n/2$ nearest neighbors to x^*,
   (b) Apply weights descending in distance from x^*

Apply weights descending in distance from x^*



Loess is locally weighted polynomial regression

To fit n datapoints y given x using loess:

1. Choose a sequence x^* at which to calculate loess fits y^*

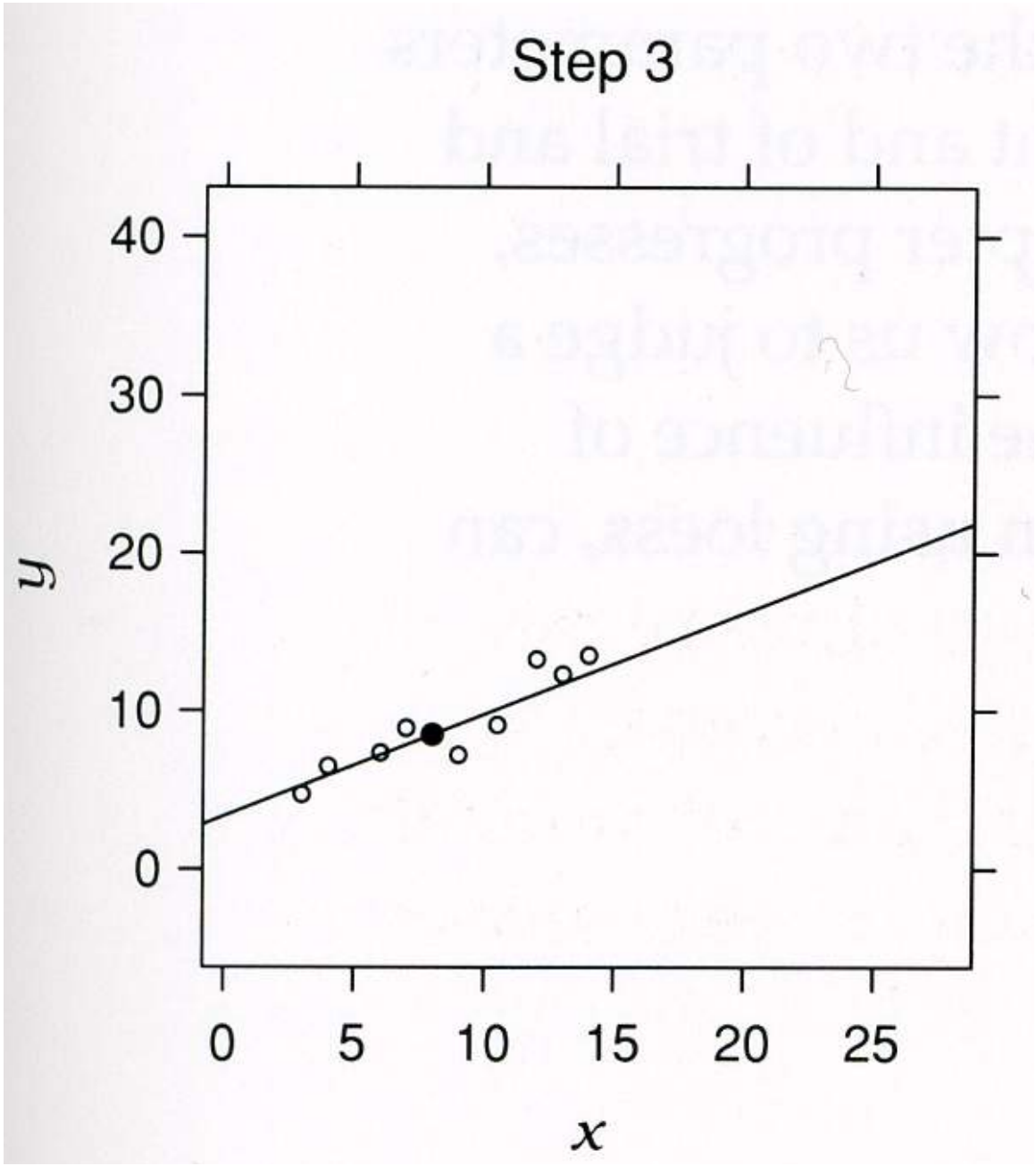
2. Choose a smoothing parameter α

3. Choose a polynomial order λ; usually 1 or 2

4. For each x^*

   (a) Select the $\alpha n/2$ nearest neighbors to x^*,
   (b) Apply weights descending in distance from x^*
   (c) Fit by WLS y_{selected} on the λth order polynomial of x_{selected}

WLS of y_{selected} on the λth order polynomial of x_{selected}



Loess is locally weighted polynomial regression

To fit n datapoints y given x using loess:

1. Choose a sequence x^* at which to calculate loess fits y^*

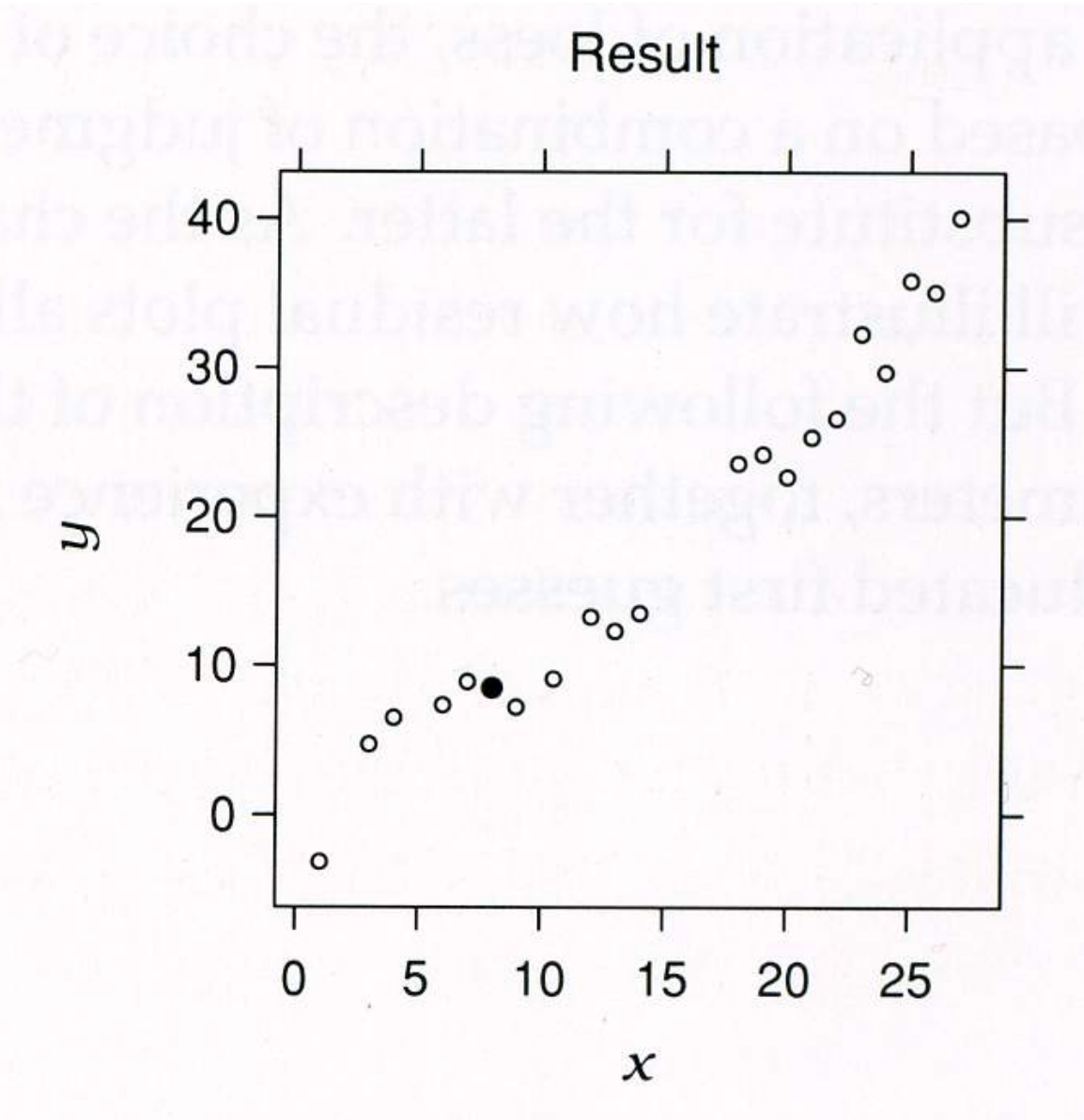
2. Choose a smoothing parameter α

3. Choose a polynomial order λ; usually 1 or 2

4. For each x^*

   (a) Select the $\alpha n/2$ nearest neighbors to x^*,
   (b) Apply weights descending in distance from x^*
   (c) Fit by WLS y_{selected} on the λth order polynomial of $_{\text{selected}}$
   (d) Record only the WLS prediction of y^* (discard the rest of the line)

Record only the WLS prediction of y^*



Loess is locally weighted polynomial regression

1. Choose a sequence x^* at which to calculate loess fits y^*

2. Choose a smoothing parameter α

3. Choose a polynomial order λ; usually 1 or 2

4. For each x^*

   (a) Select the $\alpha n/2$ nearest neighbors to x^*,
   (b) Apply weights descending in distance from x^*
   (c) Fit by WLS y_{selected} on the λth order polynomial of $_{\text{selected}}$
   (d) Record only the WLS prediction of y^* (discard the rest of the line)

5. Connect the fitted y^*'s with a line.

Key issue here is the bandwidth parameter, α. Lower values make a jerkier line; higher values a smoother one

If your data seem to curve, use a quadratic fit within loess ($\lambda = 2$) to get a better fit

What the heck is a "spline"?

Splines are a concept from carpentry, of all places

Splines let us summarize very complex curves with a few numbers

Basic idea:

- Imagine a flexible piece of wood.
- We pick it up and bend in many places;
- then tack it down (at "knots") to a board.

The idea behind splines

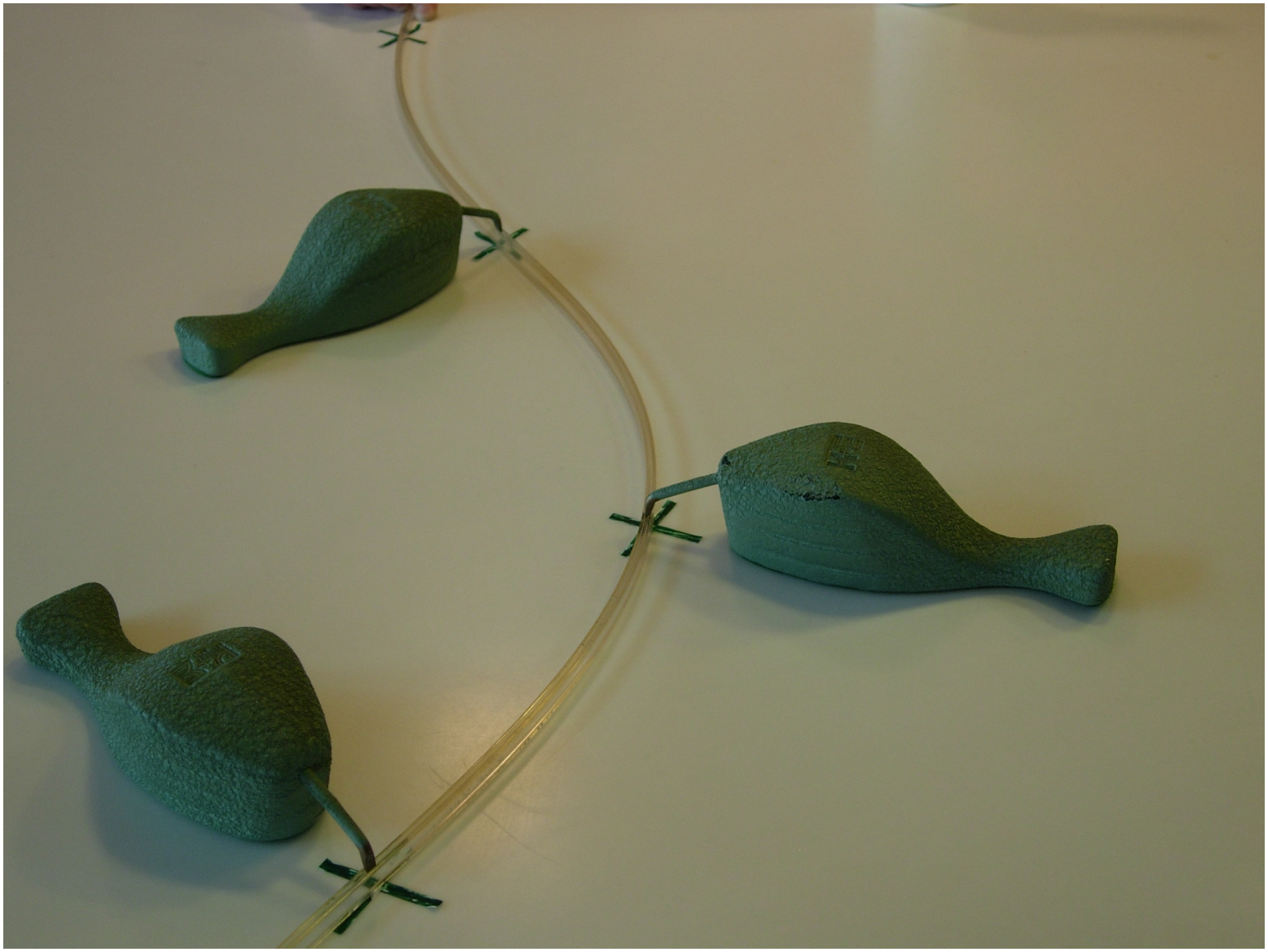


Photo of draftman's spline from Carl de Boor
www.cs.wisc.edu/d̃eboor/draftspline.html

Splines in statistics

Many similar shapes can be approximated by local cubic polynomials, which we will call cubic splines

(Note: there are *many* kinds of splines.)

Even more than loess,
Cubic splines rely on simple local structure to create global flexibility

1. Start with a few data points.

2. Connect every point to its nearest neighbor with a (twice differentiable) cubic polynomial

3. To make a "smoothing" or approximate spline,
   take the weighted average of the spline and the least squares fit

4. Choose knots & amt of smoothing subject to a penalized likelihood criterion, e.g.,

$$\max(2 \times \text{ log-likelihood } - \text{ trade-off } \times \text{ smoothness penalty })$$

Spline examples

Using demonstration software

`http://www.particleincell.com/blog/2012/bezier-splines/`

More complex example: Democracy & Development

The relationship between democracy, dictatorship, and economic development is well-explored in political science

Key recent work: Przeworksi, Alvarez, Cheibub & Limongi. *Democracy and Development: Political Insitutions and Well-being in the World, 1950–1990*

We'll borrow their data, but run *much* cruder models

In particular, PACL investigate selection bias and the difference between creation and sustenance of democracies.

We'll ignore these issues, and consider covariates of

the degree of civil liberties (CIVLIB: 1-7 scale)

the presence of democracy (REG: 0-1 binary)

More complex example: Democracy & Development

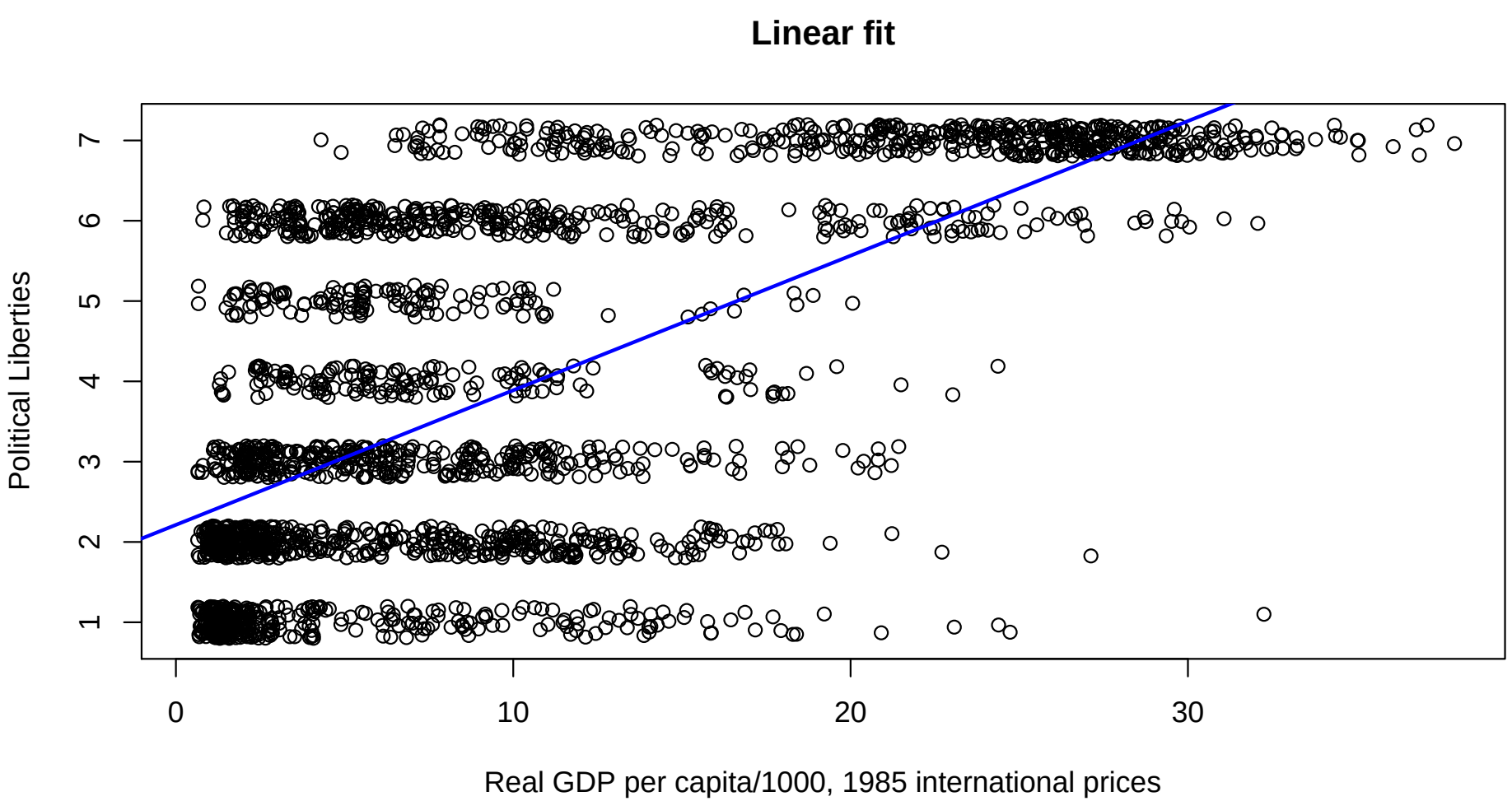
Our candidate covariates

GDPW	GDP per capita in real international prices
EDT	average years of education
ELF60	ethnolinguistic fractionalization
MOSLEM	percentage of Muslims in country
OIL	whether oil accounts for 50+% of exports
STRA	count of recent regime transitions
NEWC	whether county was created after 1945
BRITCOL	whether country was a British colony

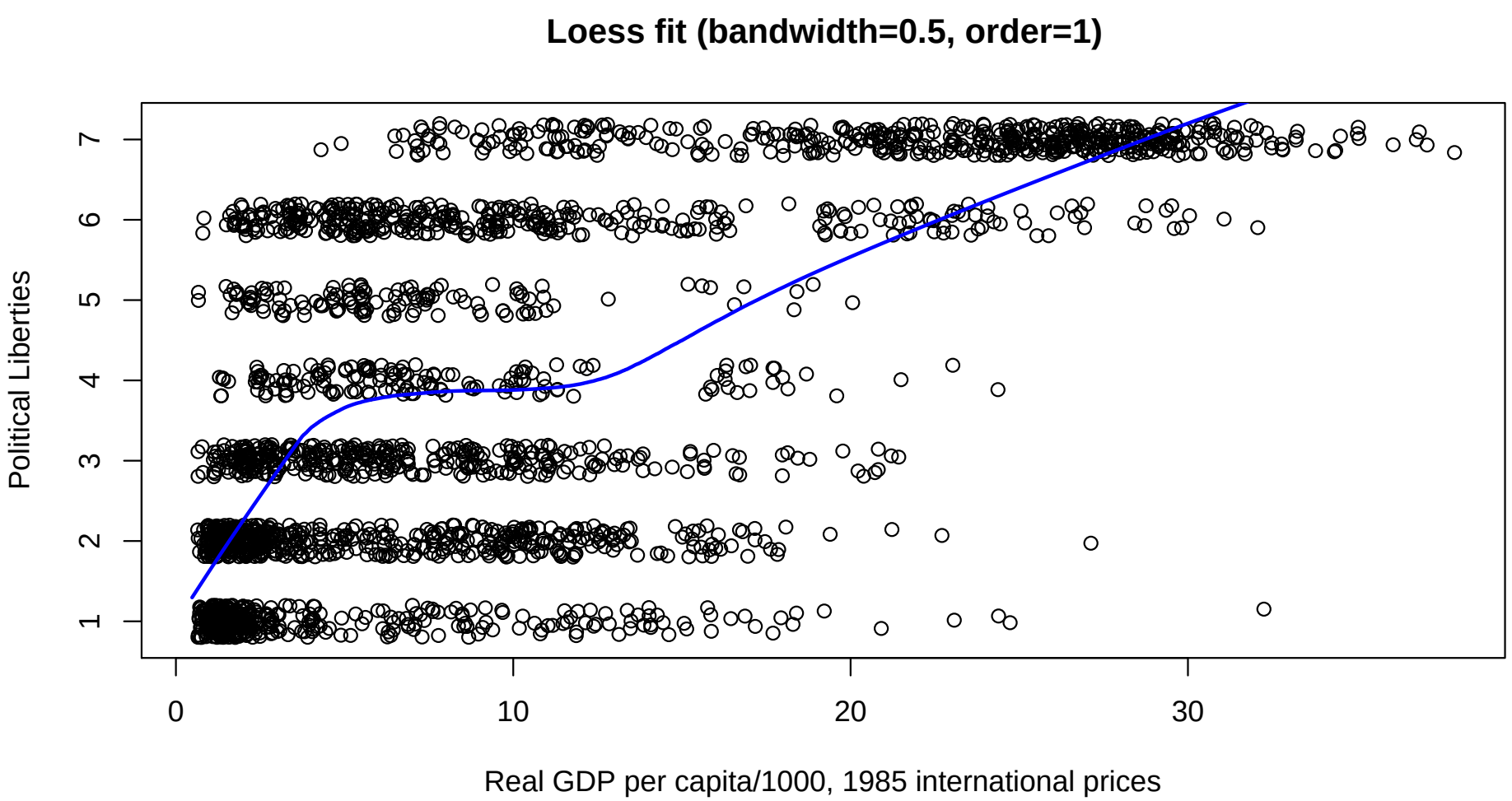
Let's start with the simplest model we can run.

Is there a relationship between civil liberties and economic development?

Let's pretend linear models are appropriate for this categorical variable



Leaving aside the categorical nature of the data,
does this fit look "right"?



Loess reveals "thresholds".

A problem: We haven't controlled for *anything*

Smoothers won't be much use if we're restricted to bivariate models

Fortunately, there is a generalization to the multivariate case. . .

Generalized Additive Models (GAMs)

Generalized Additive Models are a generalization of GLMs

incorporate smooths into multiple regression, and into logit, probit, etc.

GAMs take the following form:

$$g^{-1}(\mu) = \alpha + \sum_{j=1}^{p} \beta_j X_j \quad + \sum_{k=1}^{q} f_k(Z_k)$$

where y is a response, X_j and Z_k are covariates, and f_k is a smoothing function

Note we can estimate parametric relations for some covariates, and non-parametric for others

f_k are often splines or loess fits

GAM for Democracy

Let's control for the rest of our variables while letting the degree of political liberties remain a smooth function of the level of development

Sample code:

```
library(mgcv)
res.gam1 <- gam(POLLIB~s(GDPW)+EDT+NEWC+BRITCOL+STRA+ELF60+OIL
                  +MOSLEM)
summary(res.gam1)
```

Notes:

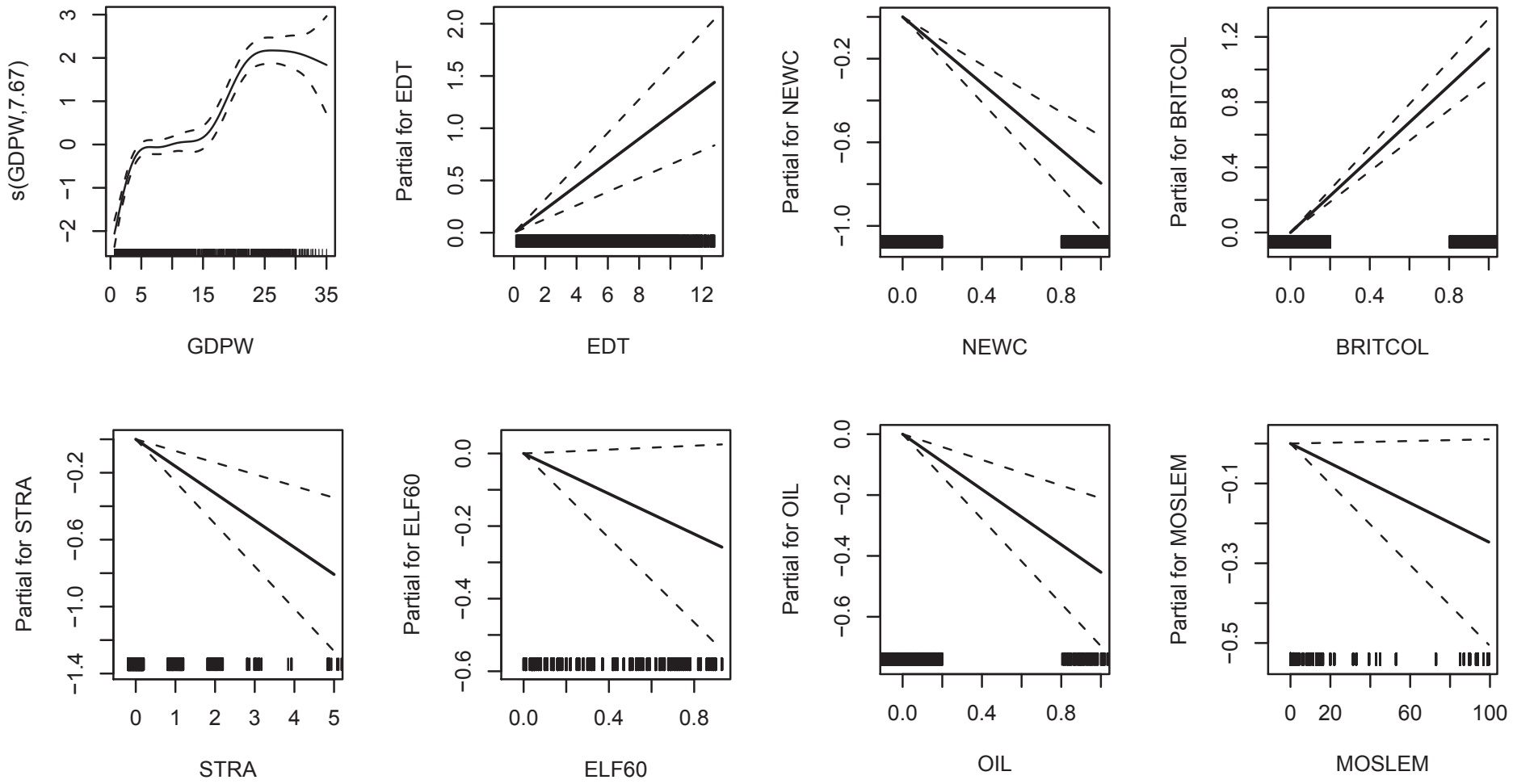
- This is the `gam` function in `mgcv`.
  There is another `gam` in library `gam`; slightly different
- Without specifying a distribution `family`,
  gam defaults to a linear Normal model
- We could specify more details of the smooth;
  this code just let's R find the best smoothing spline by cross-validation

Output of summary(res.gam1)

```
Family: gaussian
Link function: identity

Formula:
POLLIB ~ s(GDPW) + EDT + NEWC + BRITCOL + STRA + ELF60 + OIL +
    MOSLEM

Parametric coefficients:
             Estimate  std. err.    t ratio    Pr(>|t|)
(Intercept)    3.8475     0.1552       24.8    < 2.22e-16
        EDT   0.11236    0.02354      4.773    1.9840e-06
       NEWC  -0.79545     0.1125      -7.07    2.3322e-12
    BRITCOL    1.1263    0.09354      12.04    < 2.22e-16
       STRA  -0.16153    0.04582     -3.525    0.00043583
      ELF60   -0.2774     0.1521     -1.824    0.068387
        OIL  -0.45368     0.1214     -3.737    0.00019278
     MOSLEM -0.0024823   0.001295     -1.917    0.055408

Approximate significance of smooth terms:
               edf       chi.sq     p-value
s(GDPW)      7.666       328.44     < 2.22e-16
```

```
R-sq.(adj) =  0.638   Deviance explained = 64.1%
GCV score = 1.8114   Scale est. = 1.7934    n = 1580
```

`mgcv`'s `gam` has preset graphics



Above made with: `plot(res.gam1,pages=1,all.terms=T)`
plus editing in Illustrator

We can smooth over two dimensions

Suppose we wanted to smooth over both development & education:

One way is to let each smooth be additive to the response:

```
library(mgcv)
res.gam2 <- gam(POLLIB~s(GDPW)+s(EDT)+NEWC+BRITCOL+STRA+ELF60+OIL
                  +MOSLEM)
summary(res.gam2)
```

which lets more of the above plots be smooths

Another way is to let the smooths "interact." Sample code:

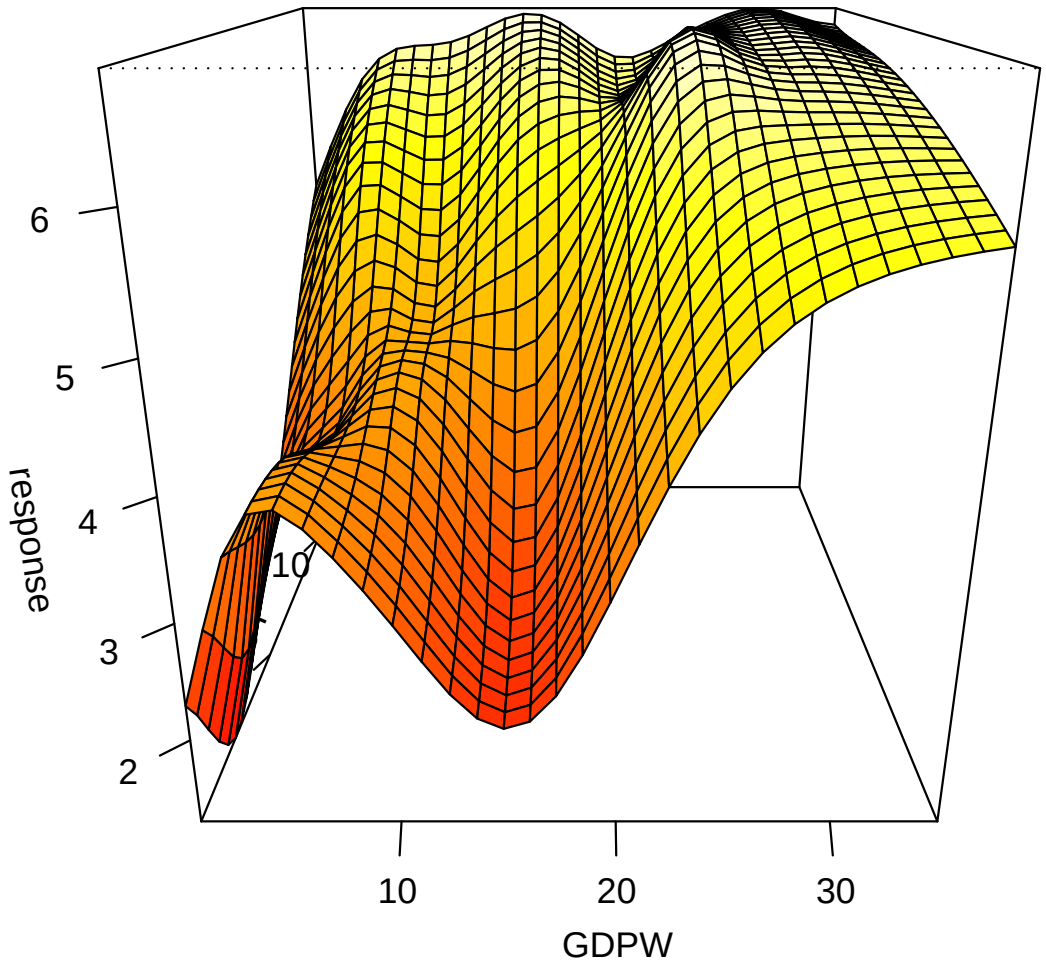
```
library(mgcv)
res.gam2 <- gam(POLLIB~s(GDPW,EDT)+NEWC+BRITCOL+STRA+ELF60+OIL+MOSLEM)
summary(res.gam2)
```

Now there is a smooth *surface* relating the joint levels of GDPW & EDT to POLLIB

We can visualize this with

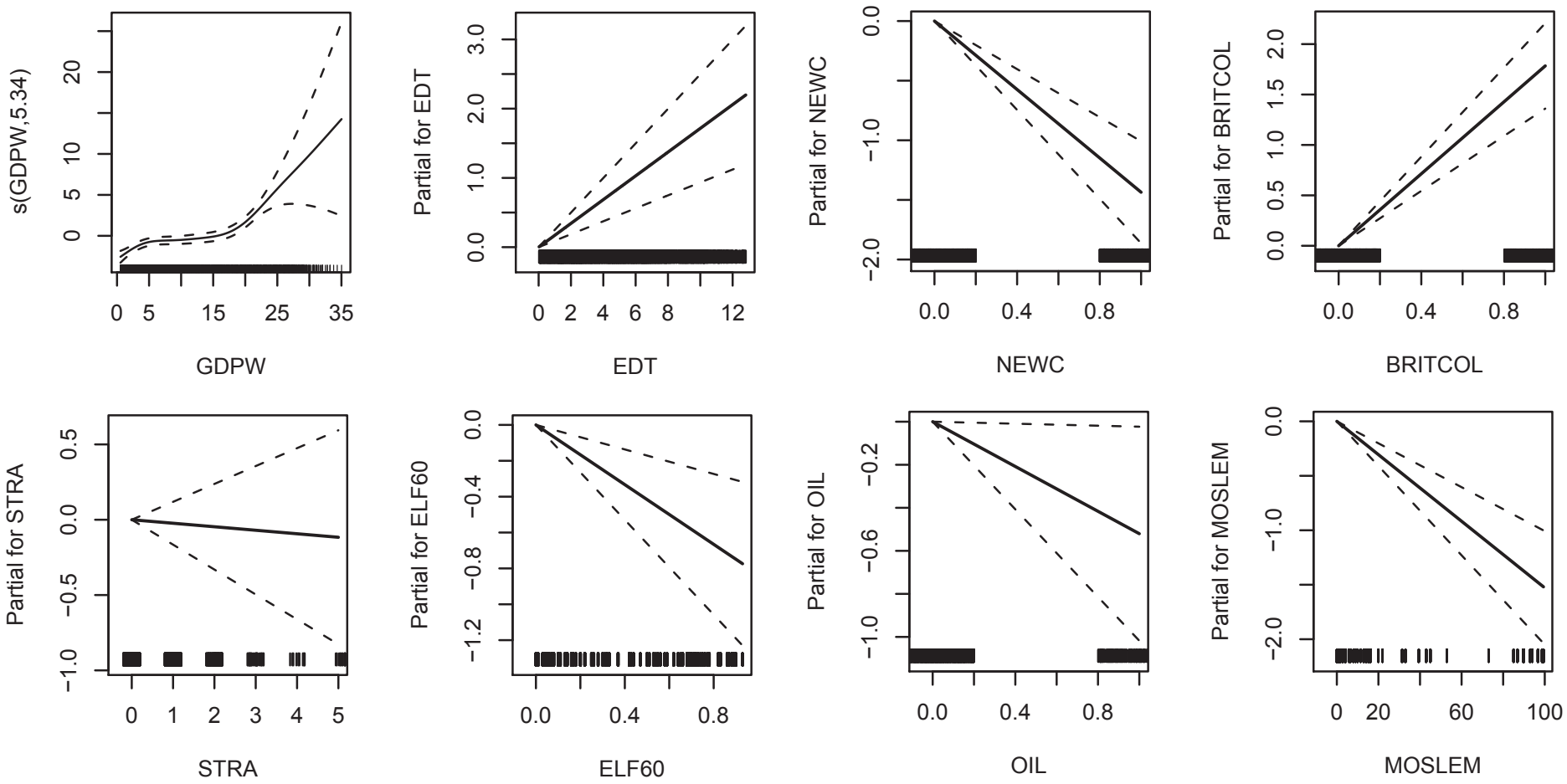
```
vis.gam(res.gam2, view=c("GDPW","EDT"), type="response",
        ticktype="detailed", theta=0, phi=20)
```

A smooth over two covariates together



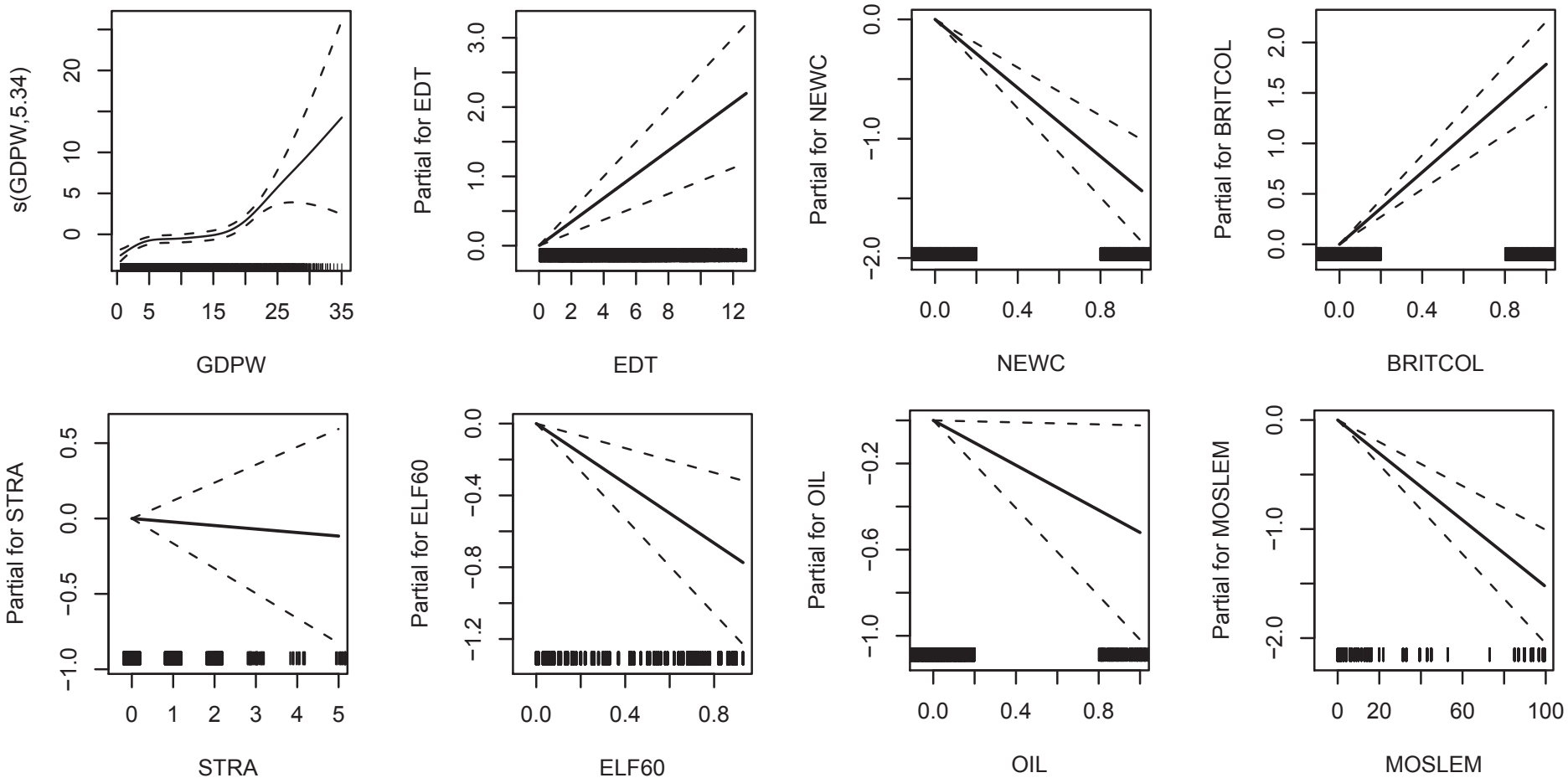
Interpretation is challenging, but this may be occassionally enlightening

A logit example using GAM



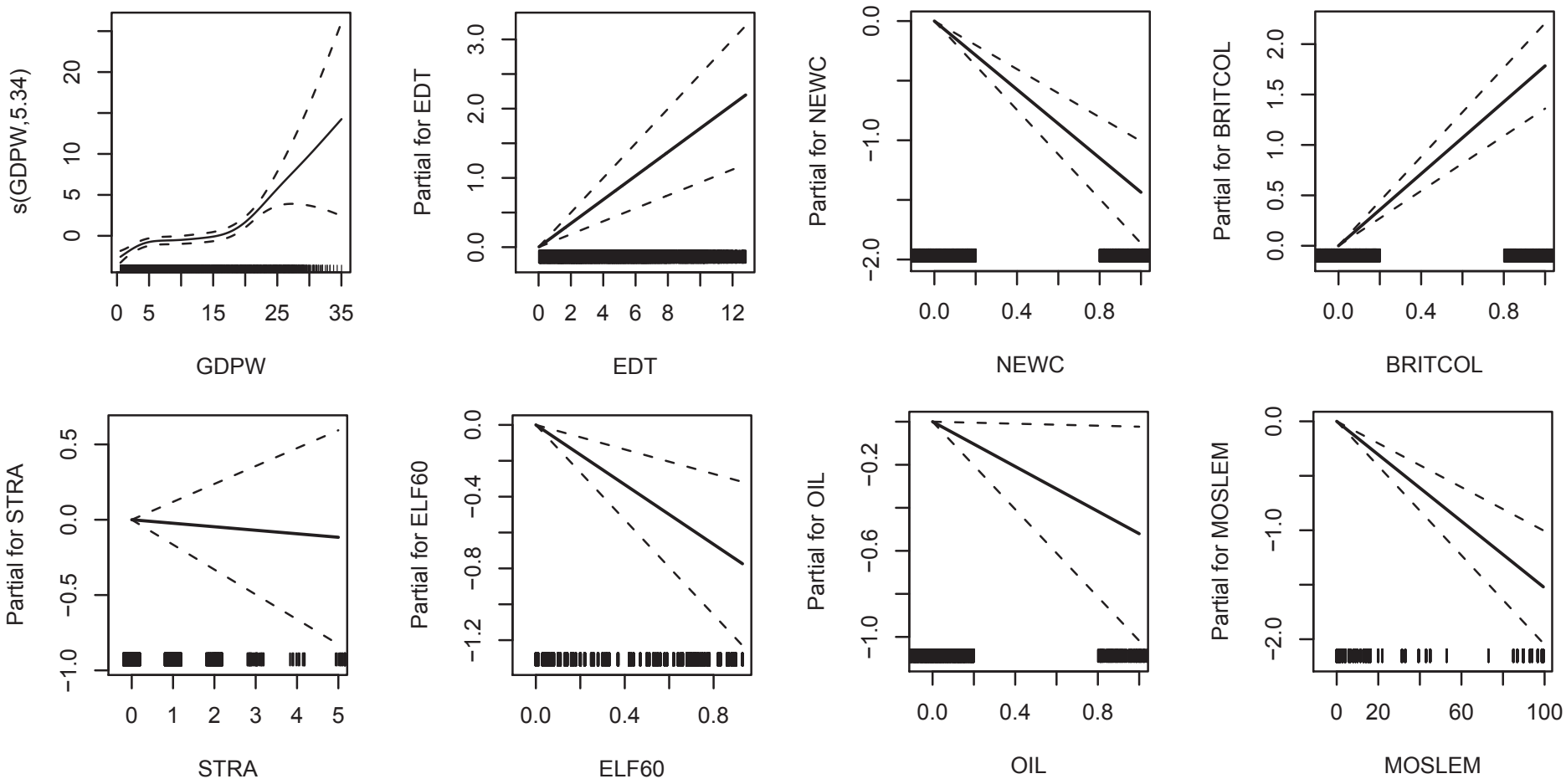
We can use GAM to fit GLM type models. What are the correlates of Democracy?

A logit example using GAM




```
res.gam4 <- gam(REG~s(GDPW)+EDT+NEWC+BRITCOL+STRA+ELF60+OIL
                    +MOSLEM, family=binomial())
summary(res.gam4)
plot(res.gam4, pages=1, all.terms=TRUE)
```

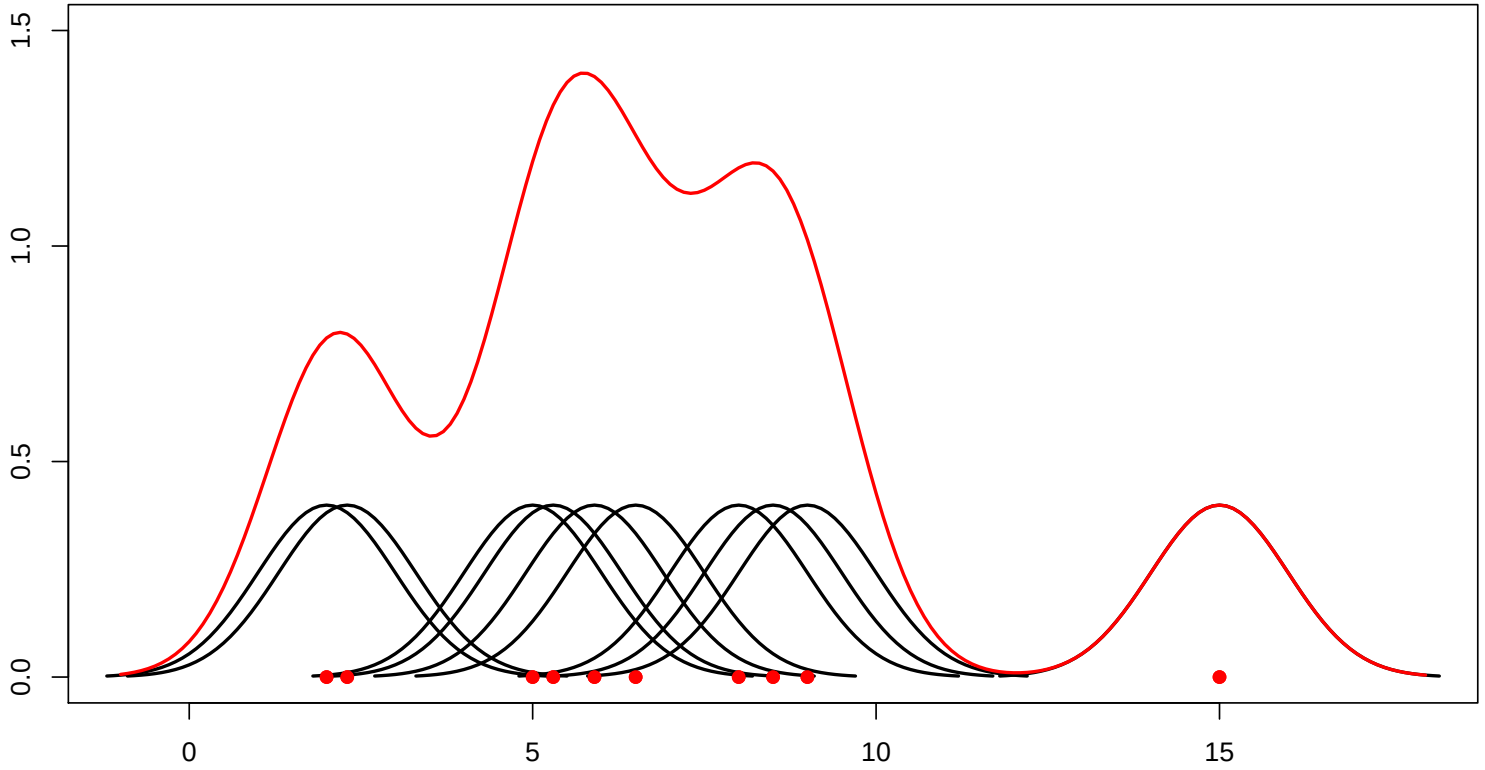
A logit example using GAM



When the GAM is a nonlinear model (e.g., logit), the scale is difficult to interpret

It would be better if the y-axes were in the units of Pr(REG)

One last smooth: kernel density estimation



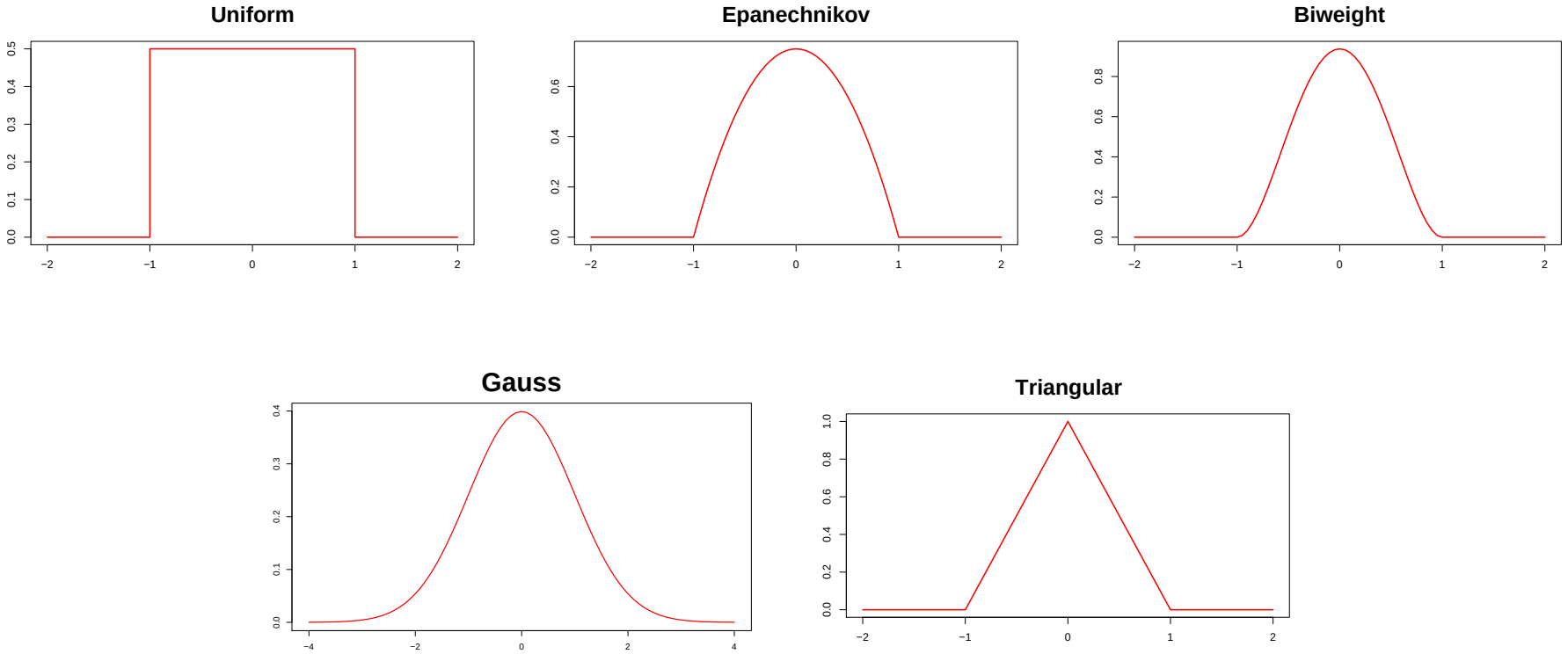
(Source for KDE diagrams: Stefanie Scheid - Introduction to Kernel Smoothing)

Another popular smoother is kernel density estimation (KDE)

KDE treats each data point as the center of a "kernel", and adds those kernels up

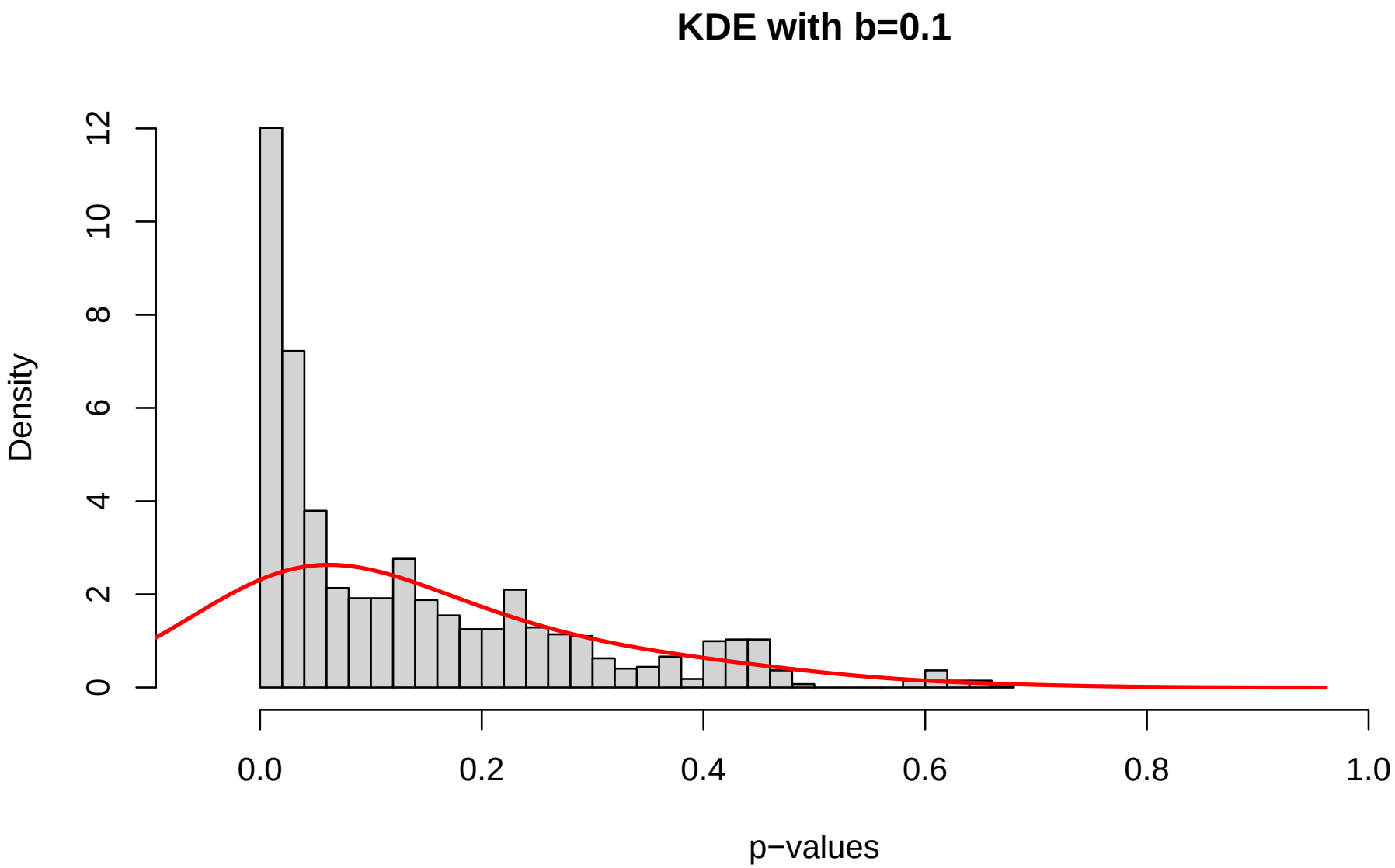
This let's the effect of each datapoint "smooth out".
Shape of smoothing out is the kernel; degree of smoothing is the bandwidth

Choices of kernels to place around datapoints



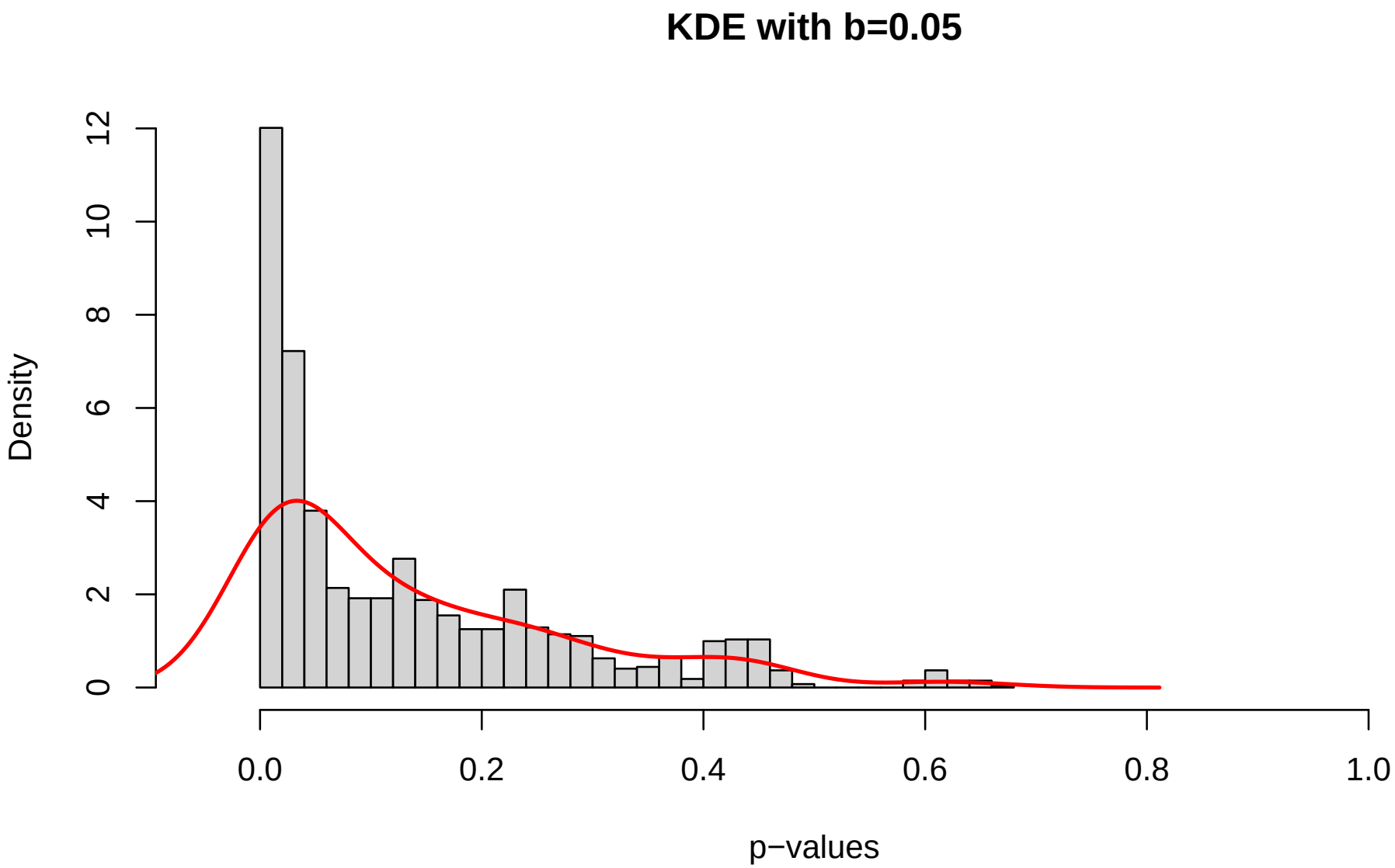
Epanechnikov minimizes error. But not usually important which you choose.

Smoothing a histogram



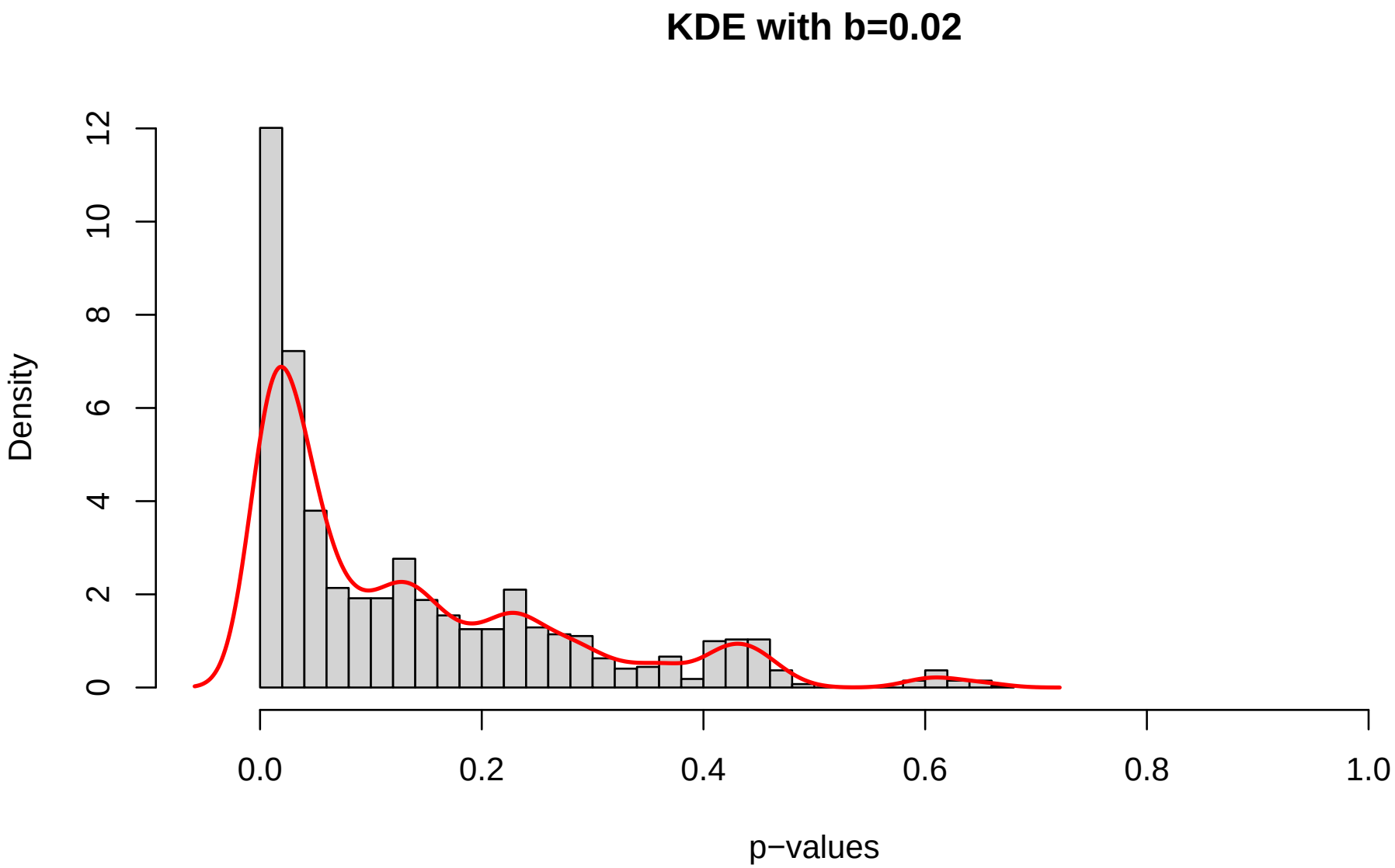
Bandwith *is* an important choice, as with any smoother.

Smoothing a histogram



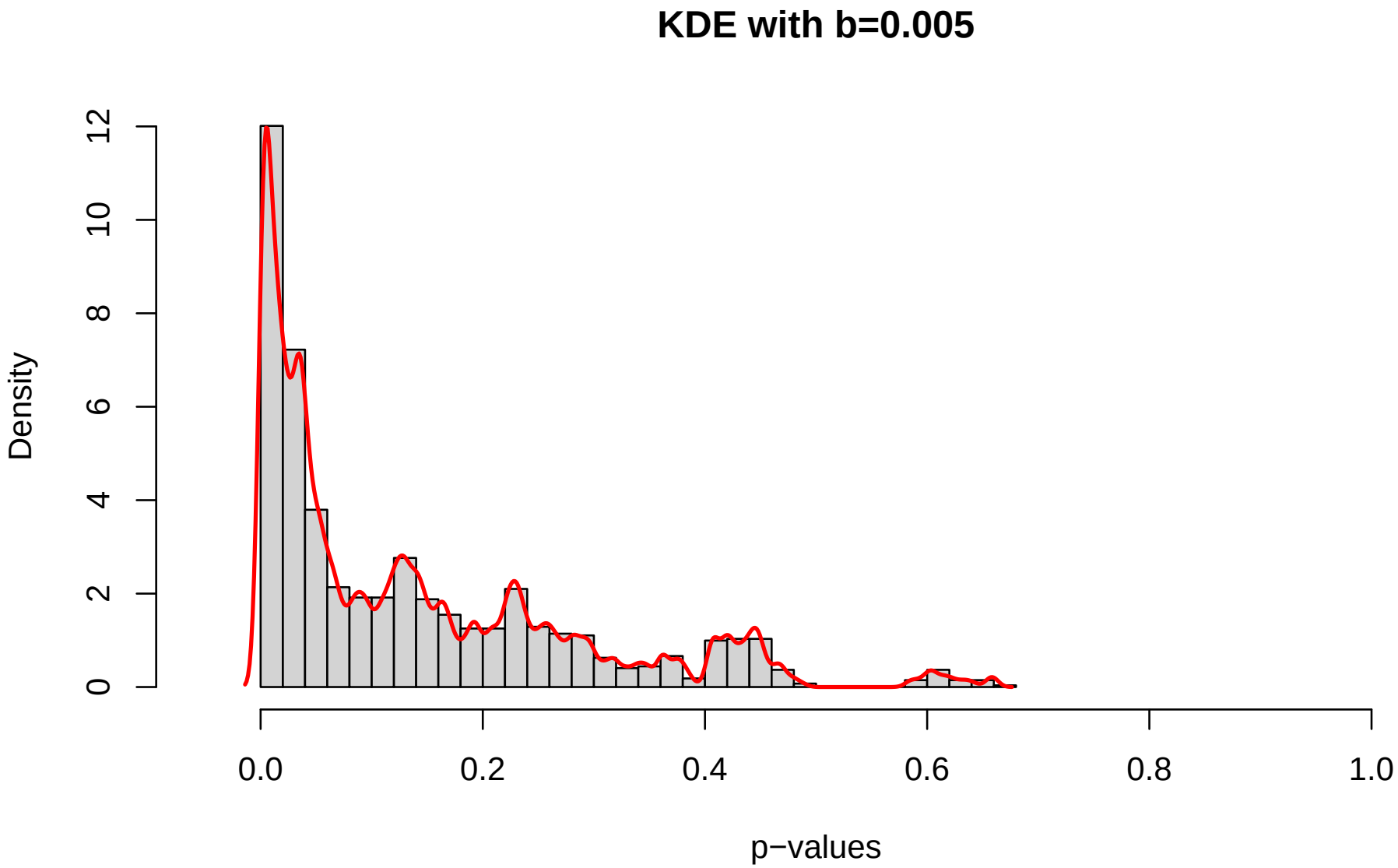
Bandwith *is* an important choice, as with any smoother.

Smoothing a histogram



Bandwith *is* an important choice, as with any smoother.

Smoothing a histogram



Bandwith *is* an important choice, as with any smoother.

Uses of KDE

KDE is very useful for smoothing histograms of all kinds

If you draw from the predictive or posterior distribution of a model,
you can smooth with KDE

This works even in two dimensions
(e.g., you want the joint confidence region of two parameters)

Generally, if you have a 2D histogram, and you want contours, consider KDE

In `R`, see the `density()` command for one dimensional KDE

See `kde2d` in the `MASS` library for 2D version

The curse of dimensionality

The biggest problem in visual display is that humans can only perceive 3 dimensions

Most data, esp in social sciences, have many variables; potentially many dimensions

Paper & computer screens are even more limiting: 2D

Three approachs to fighting the curse of dimensionality:

- Cleverly display all the data.
- Use models to reduce the number of dimensions
- Use time & motion

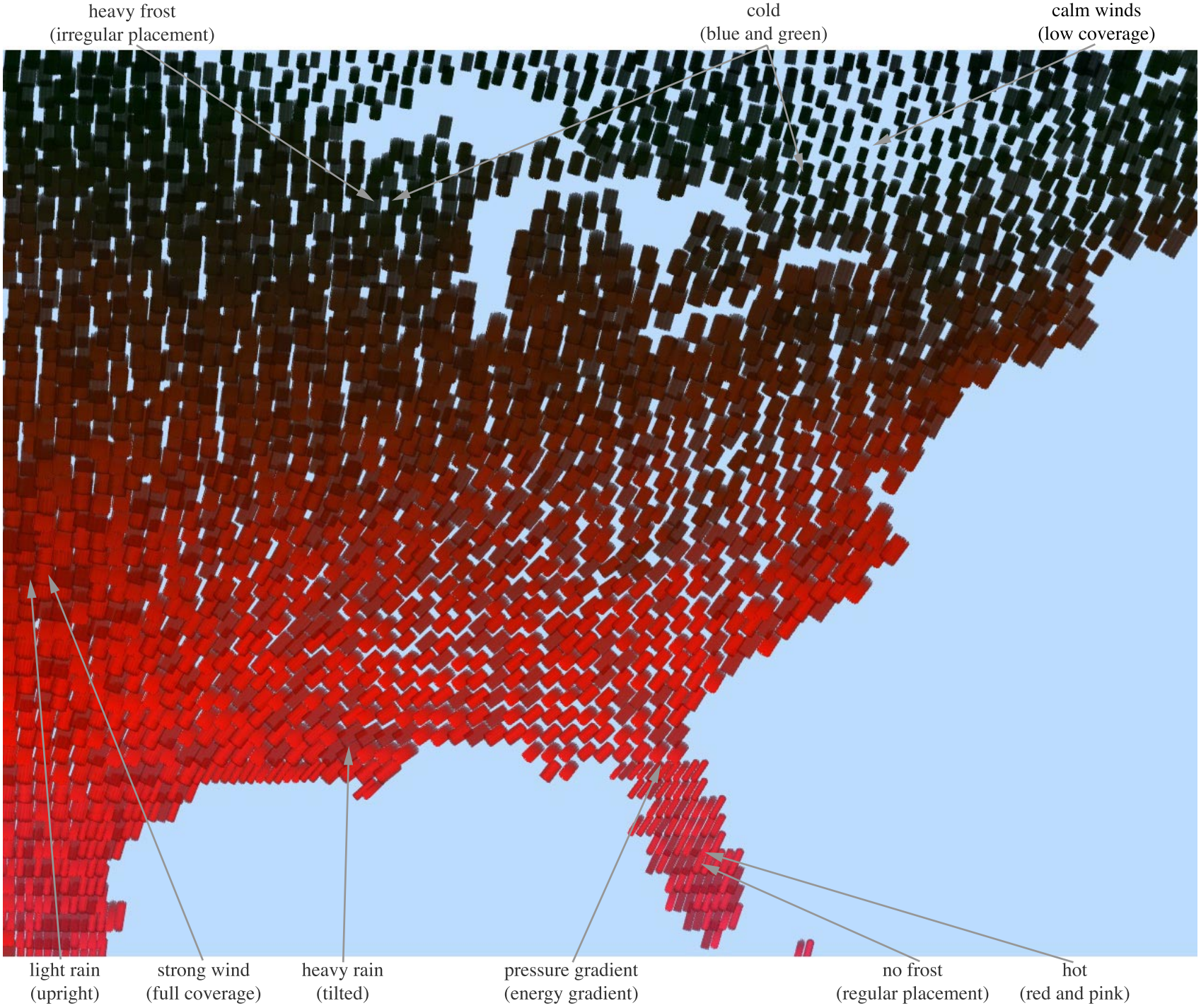
Clever display of the whole dataset

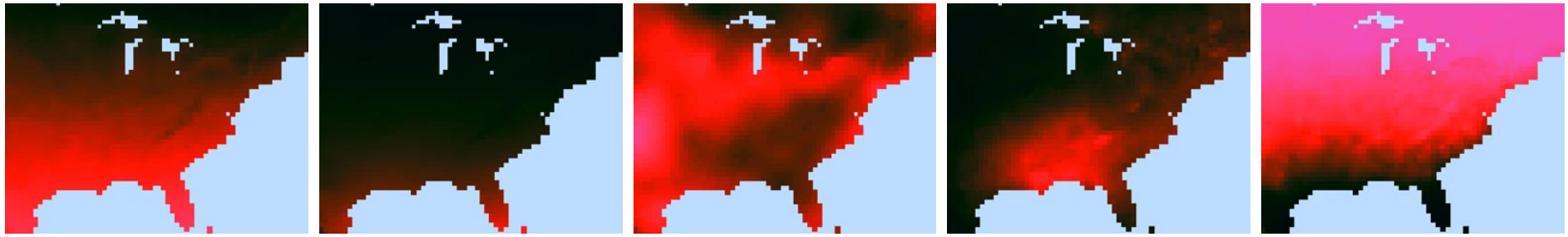
This is what we've done all quarter:

1. Small multiples
2. Glyphs
3. Layer pre-attentive elements

We've seen lots of examples.

Here is a final example that pushes the envelope


heavy frost
(irregular placement)
cold
(blue and green)
calm winds
(low coverage)
light rain
(upright)
strong wind
(full coverage)
heavy rain
(tilted)
pressure gradient
(energy gradient)
no frost
(regular placement)
hot
(red and pink)



temp *pressure* *wind* *precip* *frost*

Source: Christopher G. Healey, "Combining Perception and Impressionist Techniques for Nonphotorealistic Visualization of Multidimensional Data", *SIGGRAPH 2001 Course 32: Nonphotorealistic Rendering in Scientific Visualization* 2001, `http://www.csc.ncsu.edu/faculty/healey/download/sig-course.01.pdf`

Does the 5-dimensional version work "better" than the small multiples?

For look-up, no. But for gestalt impressions, perhaps worth looking at both.

I've experimented with the same 5 dimensions independently. Probably near the limit for a single diagram. Challenging to process.

So let's try to reduce dimensions while keeping most of the data

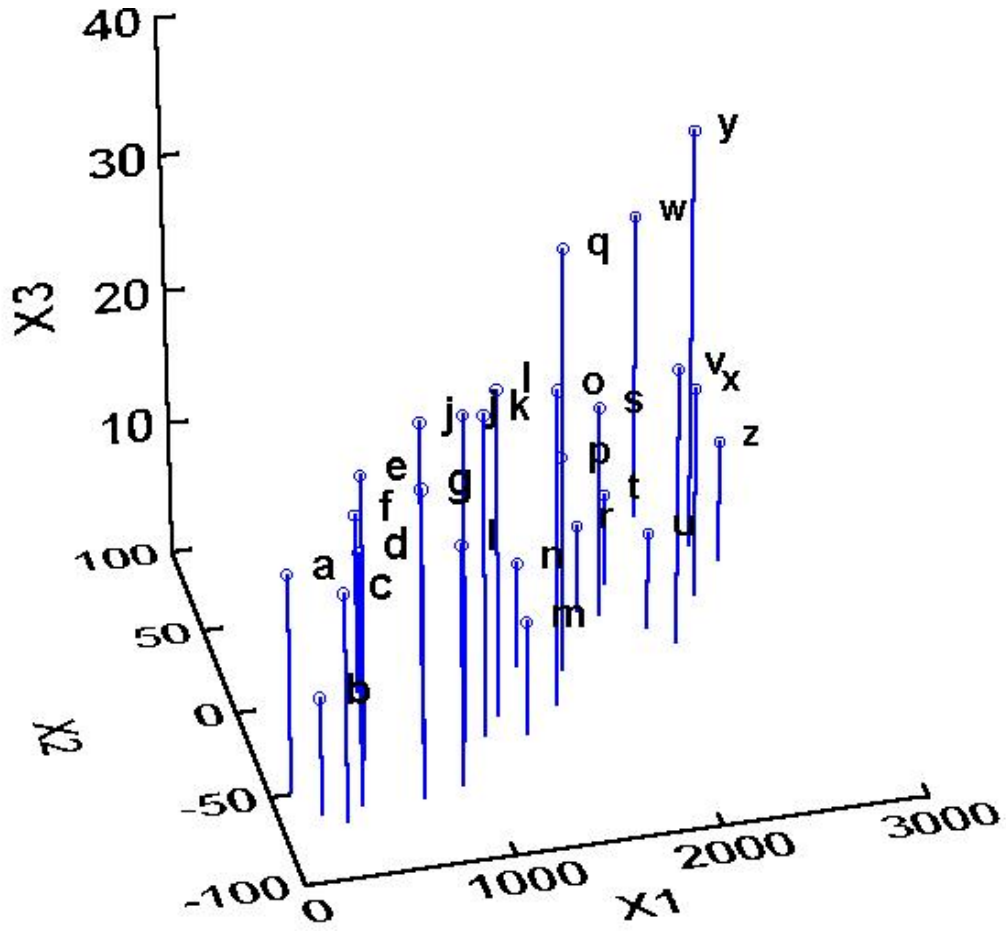
Principal Components Analysis (PCA)

PCA is related to factor analysis and multidimensional scaling

Details involve a lot of linear algebra; what follows is a conceptual summary

Imagine a dataset with k variables plotted in k-space

A candidate for PCA

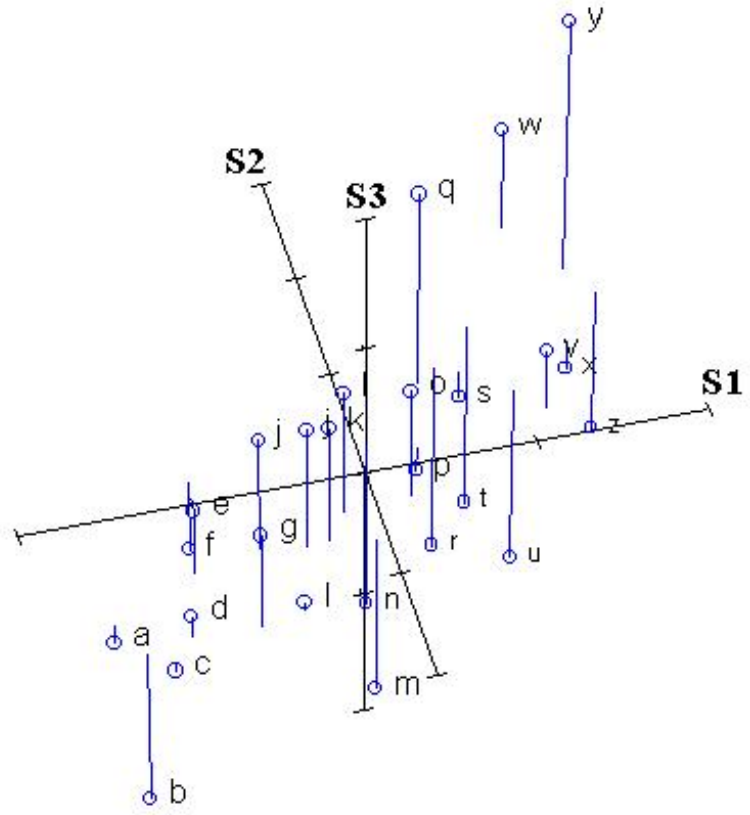


(Source: Michael Palmer, `ordination.okstate.edu/PCA.htm`)

For example, $k = 3$. Note the different scales.

It will help to normalize these variables to have mean 0 and unit variance.

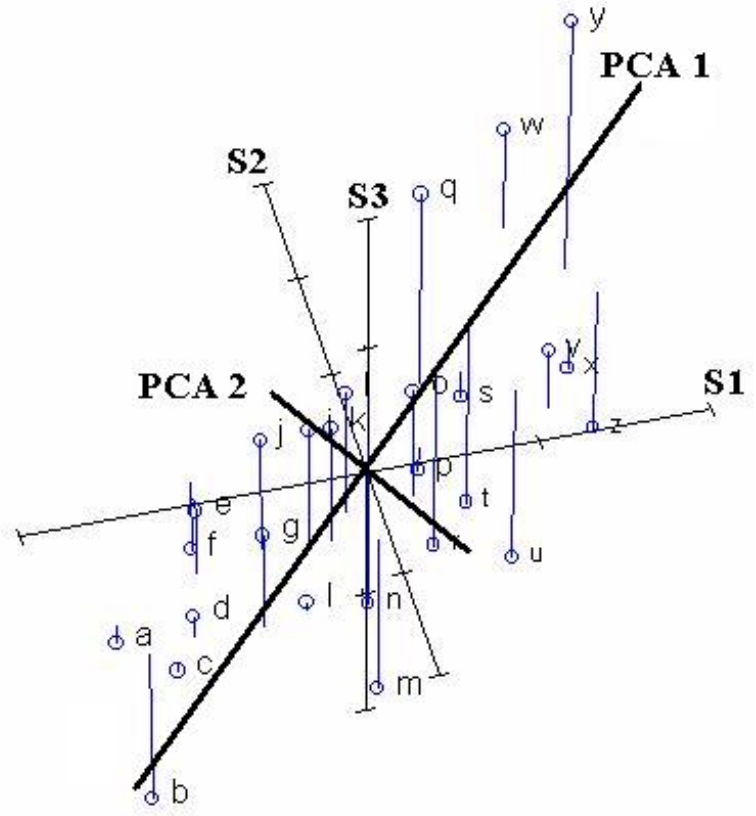
The revised data



We're going to come up with a new set of dimensions, $\leq k$, that best explain and separate the variance in these data

We search for new axes (components) sequentially

The principal components

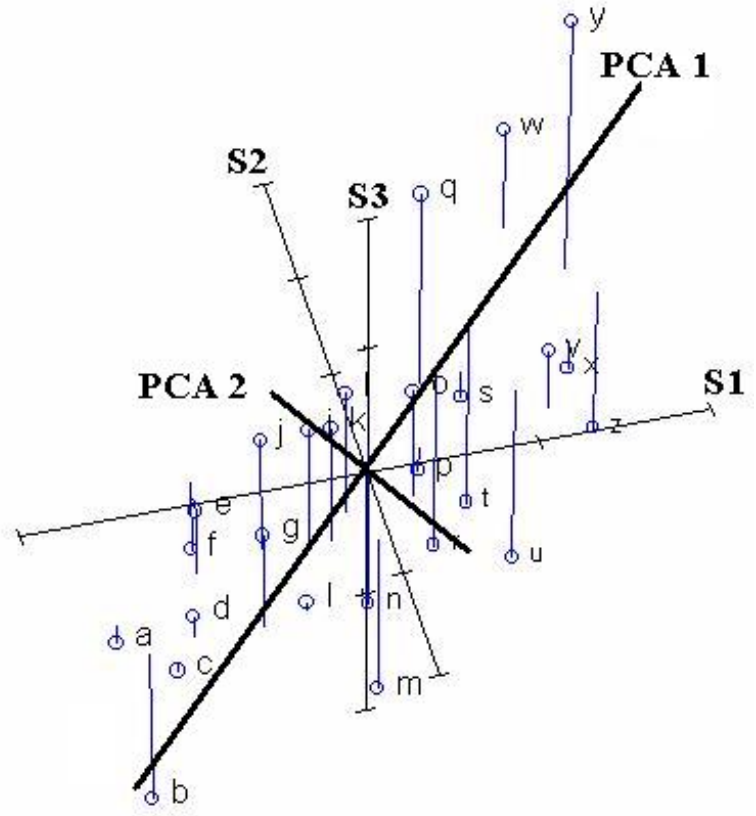


The first principal component PC1 is the axis that minimizes remaining variation in the data

The second pc is the axis, orthogonal to PC1, that explains the greatest part of the remaining variation

etc

The principal components



Two principal components are often (but not always) a nearly complete replacement for the k original dimensions

Each principal component can be seen as a linear combination of the original k axes

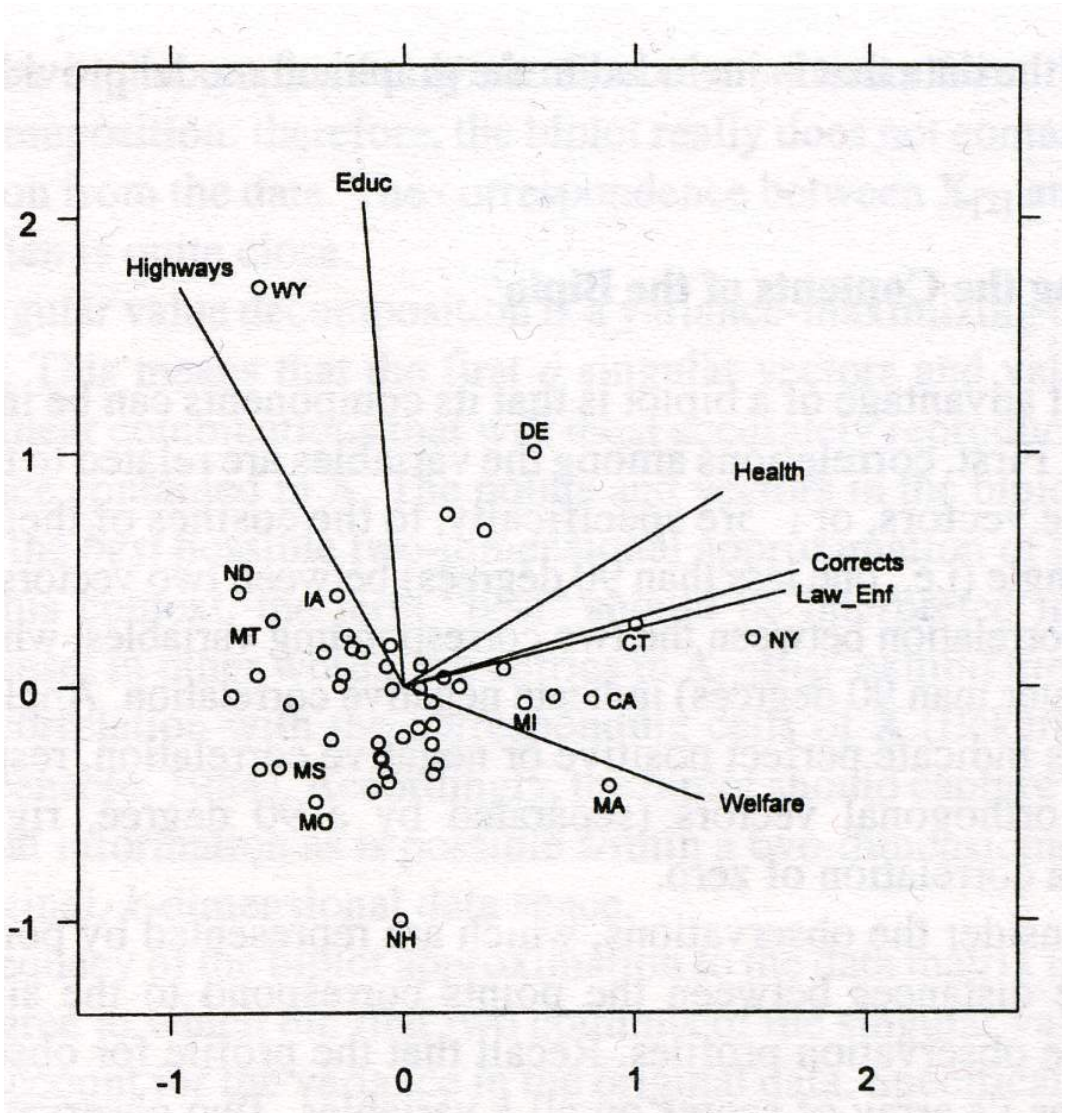
From PCA to biplots

Next we switch to the PC space, and plot in that space the locations of both the data and the original dimensions

This simultaneous plot of dimensions and data is called the biplot

Let's use some US state budget data as an example.

1992 state spending in 6 policy areas



Source: William G. Jacoby, *Statistical Graphics for Visualizing Multivariate Data*, Sage Paper 07-120.

Reading biplots

Reading biplots takes practice. Some tips:

Start with the dimensions.

If two dimensions are overlapping, they are perfectly positively correlated

If 2 d's form a 180 degree line, they are perfectly inversely correlated

If 2 d's form an acute angle, they are somewhat positively correlated

If 2 d's form an obtuse angle, they are somewhat negatively correlated

If 2 d's form a 90 degree angle, they are orthogonal

The dimensions meet at the origin (0,0). (Why?)

Data far from the origin are outliers

Distances between points are Mahalanobis distances
(b/c we standardized the variables to have unit variances)

Democracy & Development Redux

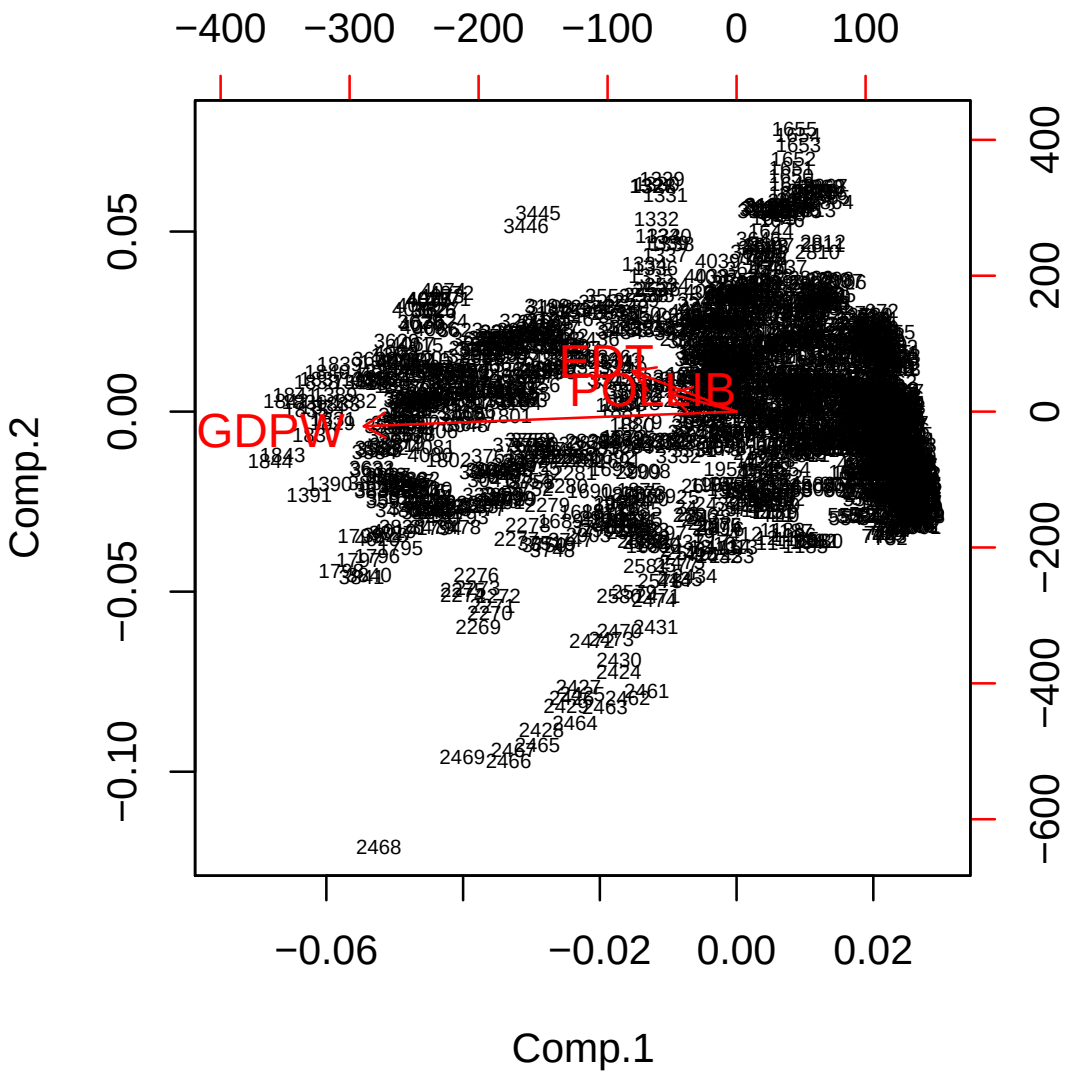
Let's apply the biplot to our Democracy & Development data

It will help to start small. Let's look at just a few continuous variables:

- Development
- Political liberties
- Average educational attainment

In R, we use the following code:

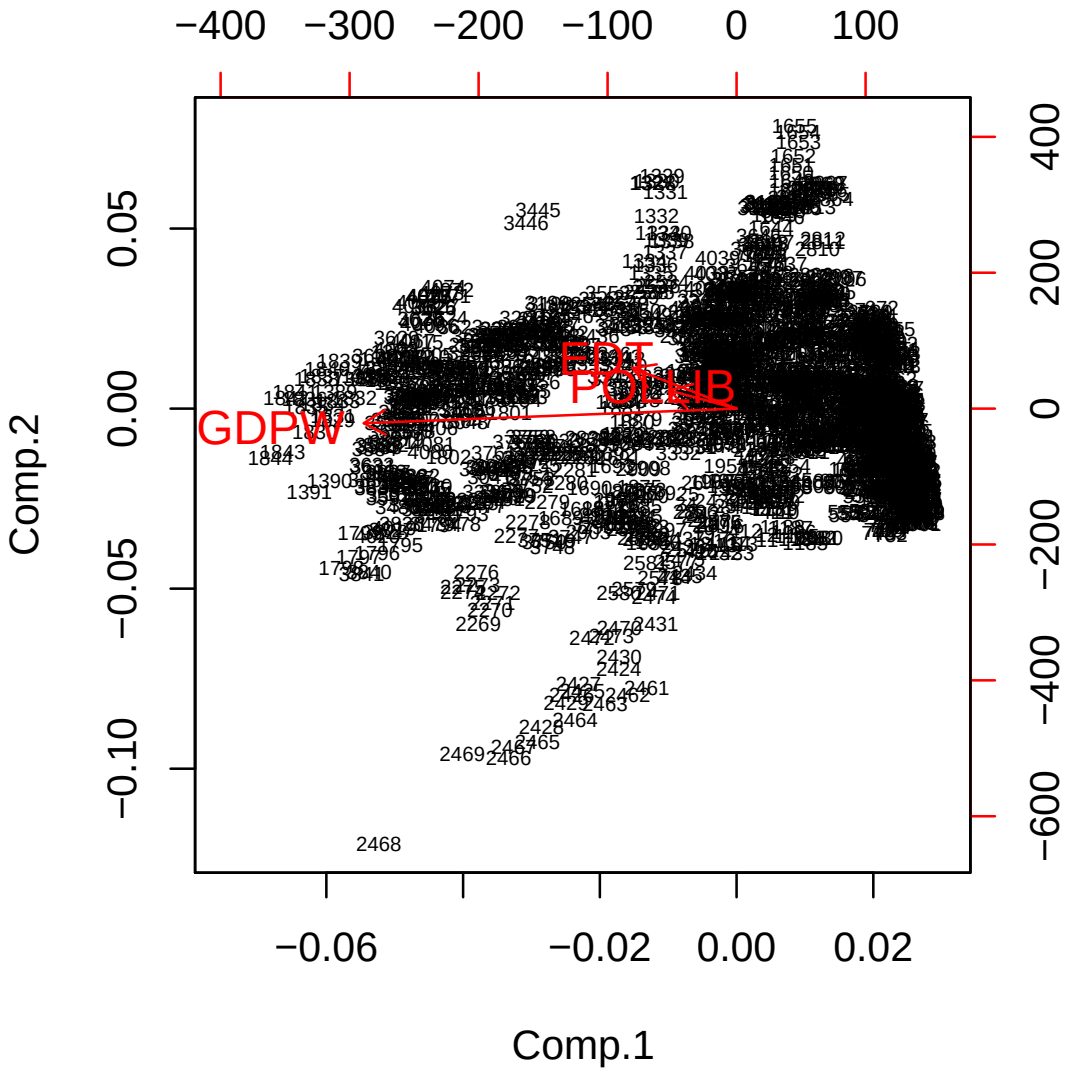
```
pca<-princomp(~POLLIB+GDPW+EDT)
biplot(pca)
```



`biplot(pca,xlim=c(-0.075,0.03),cex=c(0.5,1.2))`

`biplot()` has several options, but is surprisingly inflexible.

I had a hard time producing remotely readable graphics



The fans of points up and down from the data axes look like outliers

We can look up these points.

Jamaica features prominently in the top fan and Iran in the bottom fan

Where do these tools fit in?

Three questions:

- What do the data look like?
- Does my model look like the data?
- What does my model look like?

Note a key difference between:

Partial effects	Unconditioned relationship
regression	correlation matrices
GAMs	PCA/biplots
Coplots	scatterplot matrices

Because of concerns with confounders, the left column is preferred in social science

But the right column is still useful as for exploration/model building

Interactive plotting

R doesn't do interaction or motion well.

For that matter, neither do journals printed on dead trees

A better model for science may be interactive graphic applets embedded in hypertext articles.

(Playing with your interactive graphic, your reader might find something you missed. A bit scary, no?)

My guess: this is coming.

For now, we need to use specialized packages for interactivity.

There are several options; I recommend Ggobi.

Ggobi

Like R, Ggobi has an unimpressive start-up "screen".

Contains many tools that are most powerful when used together

- Scatterplot matrices
- Parallel coordinate plots
- Brushing
- Grand tours
  (Roughly, random rotation of the biplot)
- Guided tours
  (Roughly, user-guided search through the biplot)

Let's explore. . .