Lecture 10

Function estimation

Last time we considered ways to relax the assumption that our unknown regression function (conditional expectation) is simply a plane -- That is, a linear combination of the predictor variables

Today we'll spell out a slightly more formal treatment of the basic ideas and motivate how they translate to multivariate problems -- To start, let's be a little explicit about our setup and assume that our input space is d-dimensional, meaning that $x = (x_1, \ldots, x_d)$, and assume that for some unknown function f our response variable Y satisfies

$$f(x) = E(Y|X = x) \text{ and } \text{var}(Y|X = x) = \sigma^2 \text{ for all } x \in \mathcal{X} \subset \mathbb{R}^d$$

In the special case of a normal model we have

$$Y = f(X) + \epsilon, \ \epsilon \sim N(0, \sigma^2)$$

Function estimation

Now, instead of thinking of our predictors as vectors and our model as a member of the column space of the model matrix, let's take a step back and introduce a p-dimensional linear space of functions $g \in \mathbb{G}$ defined on $\mathcal{X}$ consisting of all

$$g(x) = g(x; \beta) = \beta_1 B_1(x) + \cdots + \beta_p B_p(x)$$

where $B_1(x), \ldots, B_p(x)$ are basis functions of $\mathbb{G}$

In this expanded treatment, we think of the elements of $\mathbb{G}$ as being candidate descriptions for the regression function f

Function estimation

For most of this quarter we have been assuming that p = d+1 and that our basis functions are

$$B_1(x) = 1 \text{ and } B_{j+1}(x) = x_j \text{ for } j = 1, \ldots, d$$

Ironically, we've spent big part of this class questioning this model and examining diagnostics to reveal its shortcomings -- Let's look at other examples

Polynomials

If we let d=1, then $\mathbb{G}$ might be the space of all polynomials of **order k** -- That is, we consider all combinations of the basis elements

$$B_j(x) = x^{j-1} \text{ for } j = 1, \ldots, k \text{ and } x \in \mathbb{R}$$

and the space has dimension p=k

If d > 1, we might consider the space of all polynomials of **(coordinate) order k** -- That is, we consider all combinations of the basis elements

$$x_1^{k_1} x_2^{k_2} \cdots x_d^{k_d} \text{ for } k_j \leq k$$

giving a space of dimension p= k^d

Splines

For d=1 again, the space of cubic splines with m knots has dimension m+4 and in general, the spline space of order k has dimension p = m+k (cubics are polynomials with order 4)

What about d>1? How do we generalize this? Let's look at a general construction that works for any set of function spaces and not just splines...

Tensor products

If we let $\mathbb{G}_1, \ldots, \mathbb{G}_d$ be d spaces with dimensions $p_1, \ldots, p_d$, respectively, where $\mathbb{G}_j$ has basis set

$$B_{j,1}(x_j), \ldots, B_{j,p_j}(x_j)$$

then we can define the tensor-product space $\mathbb{G}_1 \otimes \mathbb{G}_2 \otimes \cdots \otimes \mathbb{G}_d$ (you were dying to use that symbol, right?!) to be the space of all functions of the form

$$g(x; \beta) = \sum_{j_1=1}^{p_1} \cdots \sum_{j_d=1}^{p_d} \beta_{j_1,\ldots,j_d} B_{1,j_1}(x_1) \cdots B_{d,j_d}(x_d)$$

which has dimension $p_1 \cdots p_d$

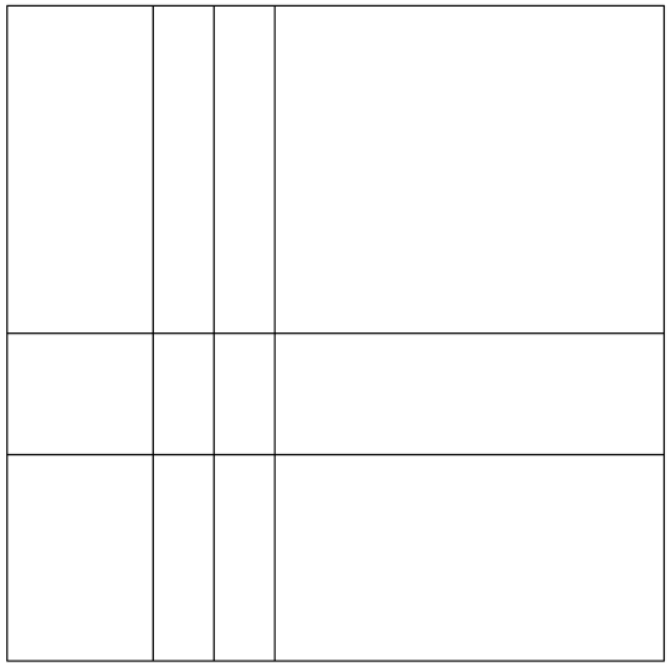
Multivariate splines

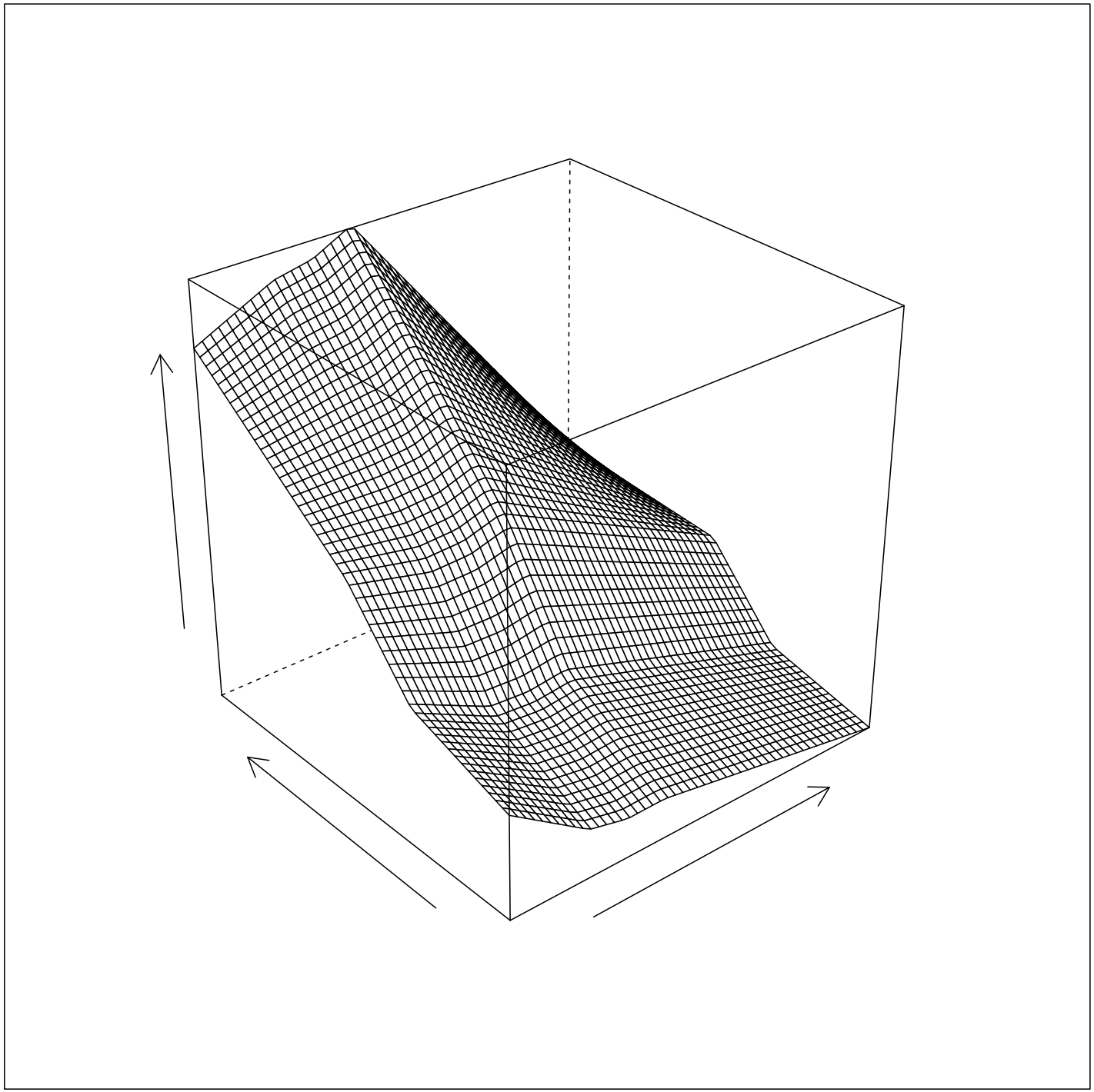
Given this simple construction, let's now take each $\mathbb{G}_j, j = 1, \ldots, d$ to be a spline space with possibly varying order and separate knot sets

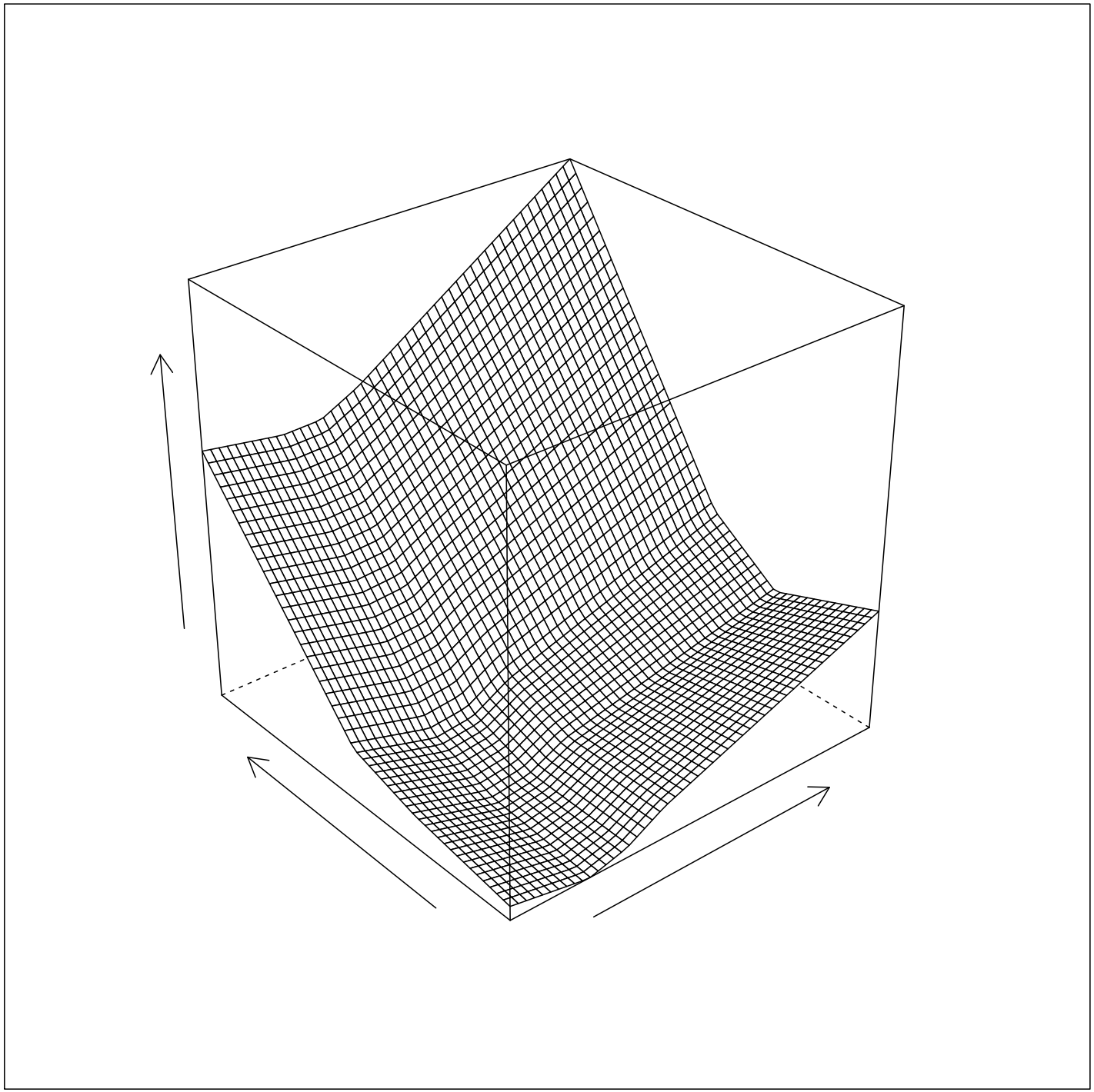
After a little thought, it should be clear that the resulting space is a polynomial inside cubes defined by the locations of the knots on each variable

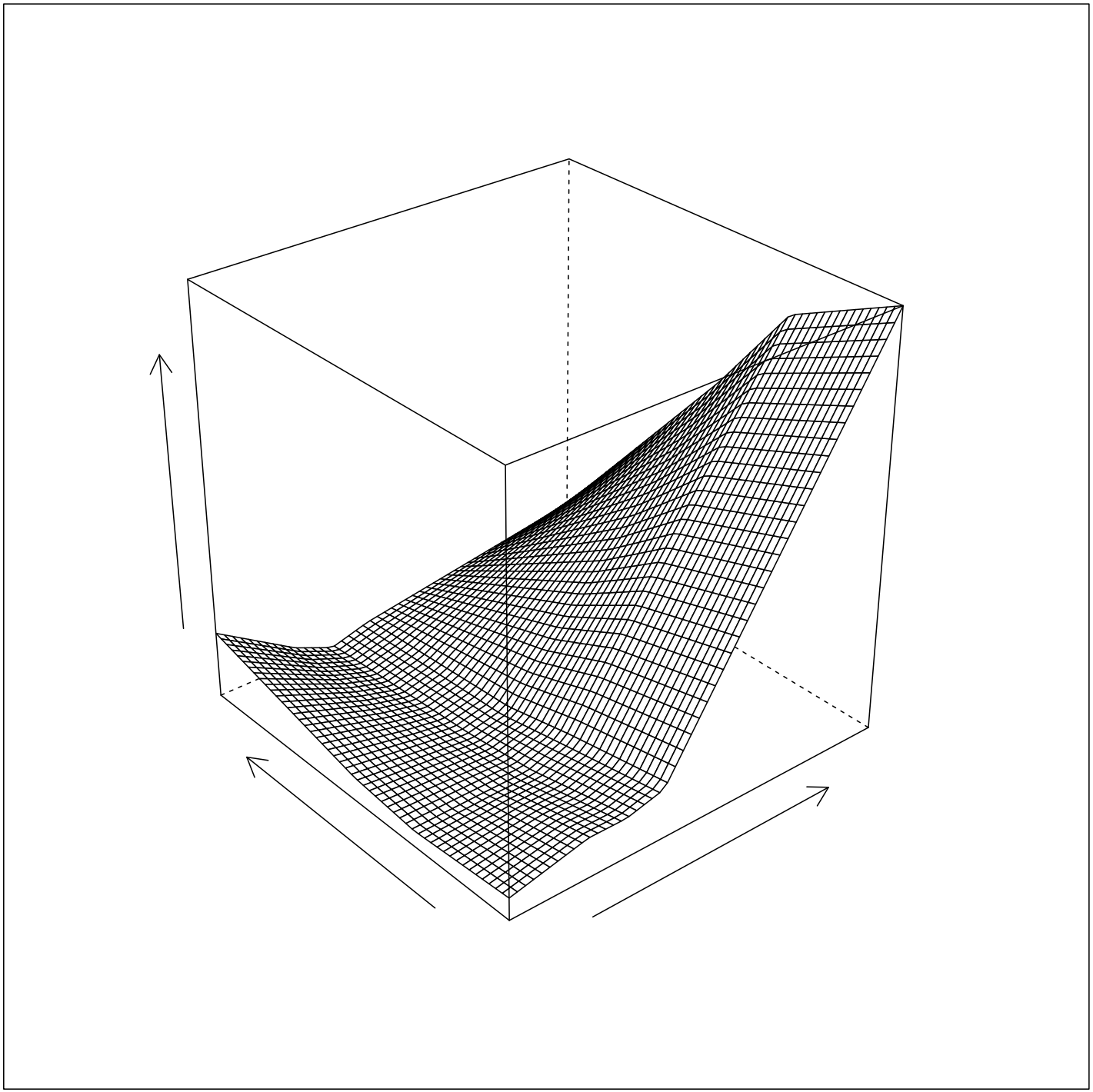
Here we have d=2 and $\mathcal{X}$ is the unit square -- We have three knots in the x-direction (0.2, 0.3 and 0.4) and two knots on the y (at 0.3 and 0.5)

The tensor product of two spline spaces defined with these knots has single polynomial pieces in each box and breaks in the derivatives as you cross the knot lines -- The degrees and smoothness depending on your original choices in the two spaces







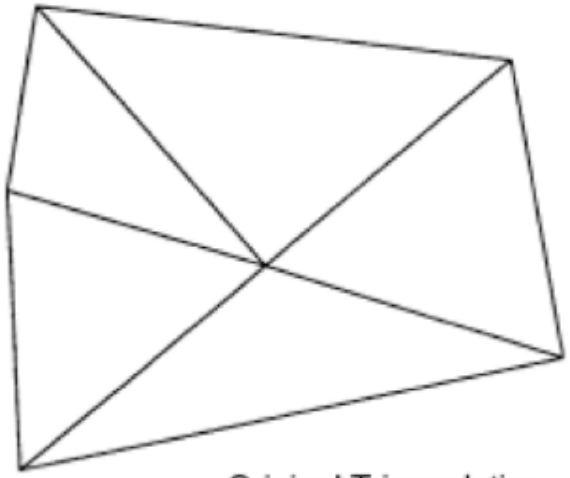


Multivariate splines

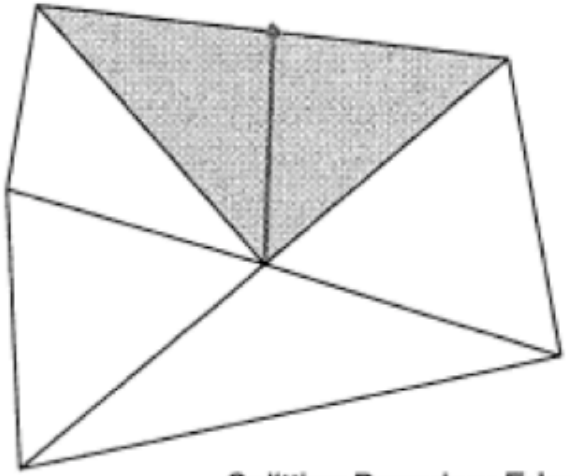
There is one line of reasoning that tries to build genuinely multivariate spline spaces; starting with an extension of “piecewise” polynomials, we could define regions in the plane, say, to be our pieces

We would then introduce constraints to smooth things out; in the case of univariate splines, we end up with compactly supported basis functions (bumps) -- for a multivariate problem this becomes harder

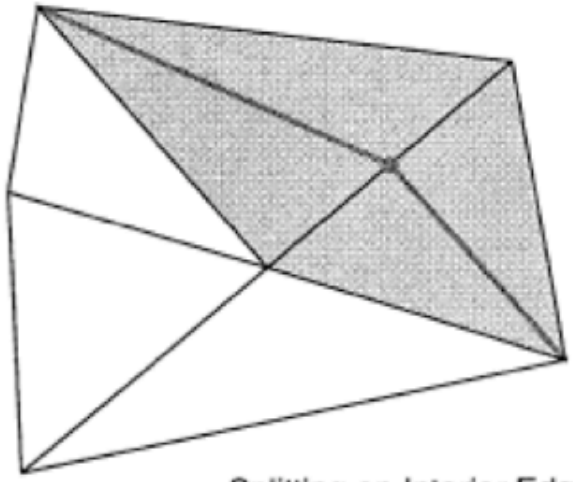
Here’s one simple example...



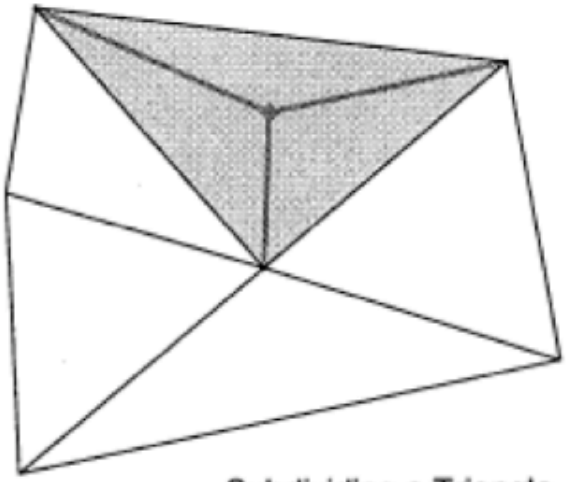
Original Triangulation



Splitting Boundary Edge



Splitting an Interior Edge

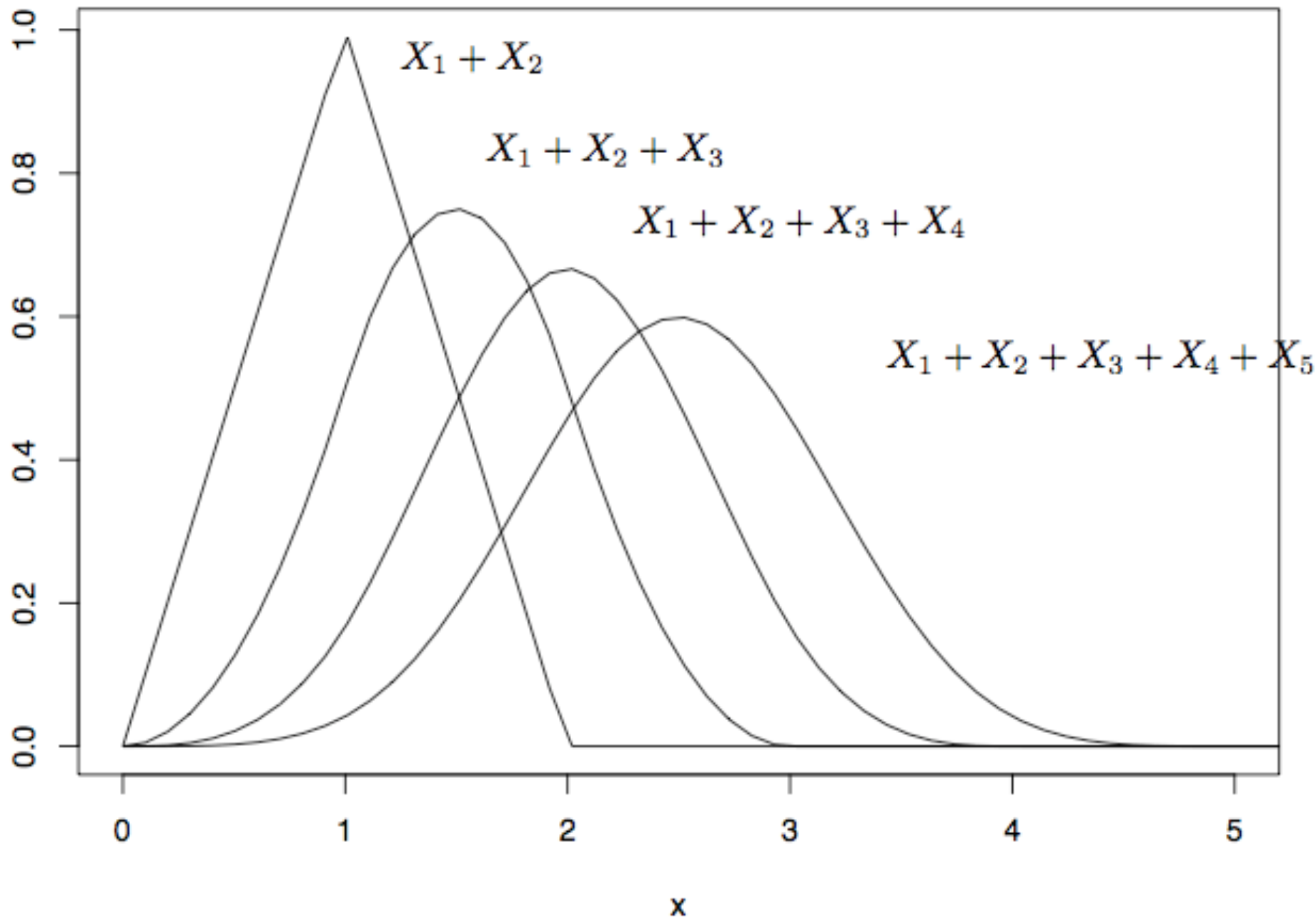


Subdividing a Triangle

Multivariate splines

There is also an extension (truly beautiful) known as simplex splines that take the convolution construction we alluded to earlier and creates multivariate bumps over triangulations

Again, for equally spaced knots we have...


$X_1 + X_2$
$X_1 + X_2 + X_3$
$X_1 + X_2 + X_3 + X_4$
$X_1 + X_2 + X_3 + X_4 + X_5$
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
x

Function estimation and OLS

Now, given n observations $(x_1, y_1), \ldots, (x_n, y_n)$ and a p-dimensional linear space of functions $\mathbb{G}$, we choose the coefficients $\beta = (\beta_1, \ldots, \beta_p)$ by ordinary least squares -- That is, we seek to find the $g \in \mathbb{G}$ that minimizes

$$\sum_{i=1}^{n} [y_i - g(x_i; \beta)]^2$$

We haven't gone too far because we still have $\widehat{\beta} = (M^t M)^{-1} M^t y$ where M is the model matrix $[M]_{ij} = B_j(x_i)$ and $\mathbb{G}$ has basis functions $B_1(x), \ldots, B_p(x)$

Approximation spaces

For the rest of the lecture (and maybe the next?) we'll focus on so-called approximation spaces -- That is, linear spaces that are adaptable in the sense that they can describe a wide range of smooth functions

The notion of adaptability is quantified through the approximation error -- Given a function f and a linear space $\mathbb{G}$, we define the distance

$$d(f, \mathbb{G}) = \min_{g \in \mathbb{G}} \|f - g\|_\infty$$

where $\|f - g\|_\infty = \sup_{x \in \mathcal{X}} |f(x) - g(x)|$

In-sample error revisited

A few lectures ago, we examined the expected error in our estimates and this quantity emerged -- We'll call it an (in-sample) model error

$$\mathrm{ME}(\mathbb{G}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}\left[f(x_i) - g(x_i;\widehat{\beta})\right]^2$$

If we let $\widetilde{\beta} = \mathrm{E}\widehat{\beta}$ then we saw that we could write

$$\mathrm{ME}(\mathbb{G}) = \frac{1}{n}\sum_{i=1}^{n} \left[f(x_i) - g(x_i;\widetilde{\beta})\right]^2 + \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}\left[g(x_i;\widetilde{\beta}) - g(x_i;\widehat{\beta})\right]^2$$

which is a bias (first term) variance (second term) tradeoff

In-sample error

Now, because $\widetilde{\beta} = \mathsf{E}\widehat{\beta}$ we see that

$$\widetilde{\beta} = \mathsf{E}\widehat{\beta} = (\mathsf{M}^{\mathsf{t}}\mathsf{M})^{-1}\mathsf{M}^{\mathsf{t}}\mathsf{E}\mathsf{Y} = (\mathsf{M}^{\mathsf{t}}\mathsf{M})^{-1}\mathsf{M}^{\mathsf{t}}\mathsf{f}$$

where $\mathsf{f} = (\mathsf{f}_1, \ldots, \mathsf{f}_n)$ for $\mathsf{f}_i = \mathsf{f}(\mathsf{x}_i)$

This says that $\mathbb{G}$ is just the least squares projection of the true regression function into the space

In-sample error

Put another way, $\widetilde{\beta}$ satisfies

$$\frac{1}{n}\sum_{i=1}^{n}\left[f(x_i) - g(x_i;\widetilde{\beta})\right]^2 = \min_{g\in\mathbb{G}}\frac{1}{n}\sum_{i=1}^{n}\left[f(x_i) - g(x_i)\right]^2$$

Now, if we let $g^* = \text{argmin}_{g\in\mathbb{G}}\|f-g\|_\infty$, the "best" approximation to f in $\mathbb{G}$, we have that

$$\begin{aligned}\min_{g\in\mathbb{G}}\frac{1}{n}\sum_{i=1}^{n}\left[f(x_i) - g(x_i)\right]^2 &\leq \frac{1}{n}\sum_{i=1}^{n}\left[f(x_i) - g^*(x_i)\right]^2 \\ &\leq \frac{1}{n}\sum_{i=1}^{n} d^2(f,\mathbb{G}) \\ &= d^2(f,\mathbb{G})\end{aligned}$$

In-sample error

This implies that the bias term in our in-sample model error is bounded from above by $d^2(f, \mathbb{G})$ -- We recall that the variance term (because we have p predictors in our regression equation) is $p\sigma^2/n$

Therefore, we can bound our model error associated with the space $\mathbb{G}$ as

$$ME(\mathbb{G}) \leq d^2(f, \mathbb{G}) + \frac{p\sigma^2}{n}$$

Approximation power of polynomials

OK, let's put this to work! As baby example, Jackson's Theorem (a standard result in numerical analysis) tells us the approximation power of univariate polynomials of order k (which we'll denote $\mathcal{P}_k$)

Specifically, if f has a bounded rth derivative and $\mathcal{X} = [a, b]$, then

$$d(f, \mathcal{P}_k) \leq C \left(\frac{b - a}{2k} \right)^r$$

This says that we can have the same effect on approximation error by dividing the interval into 5 pieces and use cubics (k=4) as we can by approximating f by a single polynomial of order 20

Approximation power of splines

A more refined result can be derived that relates to splines and not just (piecewise) polynomials -- If f has r continuous derivatives, then for k>r, the approximation rate for $\mathcal{S}_k(m)$, the space of splines with m equally-spaced knots in $\mathcal{X} = [a, b]$

$$d(f, \mathcal{S}_k(m)) = C \left(\frac{1}{m}\right)^r$$

In-sample error

This means that our model error associated with $\mathcal{S}_k(m)$ can be bounded from above by the expression

$$C\left(\frac{1}{m}\right)^{2r} + \frac{(m+k)\sigma^2}{n}$$

where the spline space of order k with m knots has dimension (m+k)

In-sample error

Now, for each sample size n we can balance the bias and variance terms -- We're going to be more concerned now with rates meaning how these quantities scale with increasing sample size n

The rate game is something you'll get in Stat 200c next term, but for now, let's just see roughly how big ME can be...

In-sample error

After a little algebra we find that m should increase with n at the rate

$$m \sim n^{\frac{1}{2r+1}}$$

Substituting this into the ME bound, we find that this choice produces a ME that is bounded by

$$n^{-\frac{2r}{2r+1}}$$

How do we make sense of this?

In-sample error

Notice that if we could assume that f was a member of $\mathbb{G}$, then we wouldn't have a bias term and the model error would drop like 1/n as our sample size increased (it's customary to talk about the square root of an error instead in which case you have the more familiar $1/\sqrt{n}$ "rate")

Instead, our error is decreasing at a slower rate, one that depends on the smoothness of the unknown regression function -- This, then, becomes an assumption to replace the strong "parametric" form we started our quarter with

Univariate function estimation

All of this analysis can be beefed up and there are excellent papers by Stone and Huang and Stone on the subject -- But even these rough calculations show that if we are only able to assume something about the smoothness of our regression function, our approximation spaces should become more complex as we collect more and more data

Working in this way, our estimates are always biased (we never assume the true regression function f is a polynomial or a spline) but that as we collect more and more data, our estimates are closer and closer to f

Multivariate function estimation

Now, moving from d=1 to general dimensional input spaces, we can derive a similar result, but with one important difference -- If we model with the tensor product $\mathbb{G}_1 \otimes \mathbb{G}_2 \otimes \cdots \otimes \mathbb{G}_d$ of d spline spaces, each with order k and m equally spaced knots then our model error bound is

$$n^{-\frac{2r}{2r+d}}$$

under multivariate smoothness assumptions that we won't really get into -- Still, what do you notice?

Summary

The upshot is that high-dimensional input spaces slow the rate of convergence considerably -- To achieve the same rate of decay for a 11-dimensional input space as a univariate one we have to collect n^3 points as opposed to n, assuming r=2

The escalating difficulty of the estimation problem with n is referred to as the “curse of dimensionality” -- This is a term coined by Bellman in 1961 to describe problems that scale exponentially with dimension

Curses

There are plenty of consequences of the curse to go around -- For example, suppose we were determined to use high-order polynomials as our approximation space of choice

Well, we saw that these tools had the tendency to behave badly in regions with little data -- The curse of dimensionality appears here to open holes in the input space as d gets large

To see this, assume our predictors are uniformly distributed in $[0,1]^d$ and consider a “hole” that is a subcube with side length δ -- The chance that this hole is empty (none of our n data points fall inside) is $(1-\delta^d)^n$

The chance that a given subcube of side-length $\delta = 0.5$ is empty is nearly zero for d=1,2,3 but for d=12 it’s 0.78 and for d=20 it’s effectively 1

Curses

Using the tensor product idea, we also have an exponential explosion in basis functions -- Here we can provide some structure that will help us tame the curse to some extent

Suppose for some dimension d we want to work with a space of polynomials of (coordinate) order 1 -- Then we can rewrite any function g in the space as

$$g(x) = \beta_0 + \sum_{j=1}^{d} \beta_j x_j + \sum_{1 \le j < k \le d} \beta_{jk} x_j x_k + \sum_{1 \le j < k < l \le d} \beta_{jkl} x_j x_k x_l + \cdots$$

This is the basis of the ANOVA expansions you studied in Stat 201

Cures

The same kind of construction can be applied to any tensor product space -- Recall we can write any function g in $\mathbb{G}_1 \otimes \cdots \otimes \mathbb{G}_d$ as

$$g(x) = \sum_{j_1=1}^{p_1} \cdots \sum_{j_d=1}^{p_d} \beta_{j_1,\ldots,j_d} B_{1,j_1}(x_1) \cdots B_{d,j_d}(x_d)$$

Now, suppose that each $\mathbb{G}_j$, j=1,...,d, contains the constant function and for simplicity assume that it's the first basis function for each space, $B_{j,1}$ -- Then we can collect all the terms in our tensor product expansion that involve x_1 and write

$$g_1(x_1) = \sum_{j_1=2}^{p_1} \beta_{j_1,1,\ldots,1} B_{1,j_1}(x_1)$$

Cures

We can do the same thing for the first d “main effects” -- The two-way interaction between variables 1 and 2 can be similarly written as

$$g_{1,2}(x_1, x_2) = \sum_{j_1=2}^{p_1} \sum_{j_2=2}^{p_2} \beta_{j_1,j_2,1,\ldots,1} B_{1,j_2}(x_1) B_{2,j_2}(x_2)$$

and extended to any two-way interaction

If we keep this up, we can eventually write any function g in the tensor product space as

$$g(x) = g_0 + \sum_{j_1} g_{j_1}(x_{j_1}) + \sum_{j_1<j_2} g_{j_1,j_2}(x_{j_1}, x_{j_2}) + \sum_{j_1<j_2<j_3} g_{j_1,j_2,j_3}(x_{j_1}, x_{j_2}, x_{j_3}) + \cdots$$

Cures

If we terminate the expansion with say two-way interactions, then the target of our estimation procedure is no longer the complete regression function, but a similarly truncated expansion for the regression function -- That is, the “best” approximation to f of the form

$$f^*(x) = f_0^* + \sum_{j_1} f_{j_1}^*(x_{j_1}) + \sum_{j_1<j_2} f_{j_1,j_2}^*(x_{j_1}, x_{j_2})$$

If we worked a bit harder and forced the higher order terms in our expansion of g to be orthogonal (using the data inner product) to the lower order terms, then the individual terms in the expansion of g approach their counterparts in f* as n gets large

Cures

This expansion gives us a way to tame the curse of dimensionality -- If we choose to not work with the full tensor product basis but instead maybe a model with just main effects or maybe all main effects and two-way interactions, we can improve the rate at which our model error decays

In particular, the rate becomes $n^{-\frac{2r}{2r+s}}$ where s is the count of the largest number of variables included in any one interaction term in the model on the previous page -- A main-effects model has s=1, while all two-way interactions mean that s=2

The important thing is that with this construction, estimating a d main effects is “as hard as” estimating a univariate function -- We can undercut the curse by dropping higher-order terms

But what do we lose in the process?

Functional ANOVA

A functional ANOVA model, then, involves undercutting the curse of dimensionality by dropping higher-order interactions in our estimate -- The target is no longer f but f*, the best approximation to f in the given form

A simple version of this is the so-called additive model -- Here we consider just the main effects

$$g(x) = g_0 + g_1(x_1) + \cdots + g_d(x_d)$$

where again each 1-factor term is a spline in the corresponding variable

Additive models

There are, of course, more direct ways to motivate an additive model -- In particular, you can think of it as the simplest elaboration of the ordinary linear model

$$g(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

We've gone to the trouble of providing an (at least back of the envelope) model error assessment to motivate why this form is interesting generally

Fitting (part I)

One strategy for fitting this kind of model is to learn the important interactive components using the data at hand -- The process mimics stepwise addition of variables

We've already seen the correspondence between elements of the truncated power basis and knot (breakpoint) addition or deletion in univariate fitting -- For the full multivariate model, we do essentially the same thing, but have to fold in the introduction of variables

Here's a sketch of one algorithm...

Specify an initial spline model

Stepwise addition: Continue until the maximum model size is reached or no candidates can be found

- Decide which basis functions are candidates for addition
- Add the best candidate
- Fit the model
- Evaluate the model
- If the model is better than the best previous one, save it

Stepwise deletion: Continue until the minimum model size is reached

- Decide which basis function can be removed from the model
- Remove the one that is the worst predictor
- Fit the model
- Evaluate the model
- If the model is better than the best previous one, save it

Output the best fit

Fitting (part I)

Broadly, this algorithm lets us start with a relatively simple model (maybe just an intercept-only fit) and then add components gradually

With the truncated power basis there are two kinds of basis functions -- One is a global polynomial built from $1, x, x^2, \ldots, x^k$ and the other are ramp functions of the form $(x - t)_+^k$

Typically, we constrain the addition process so that we only add spline terms after the full corresponding polynomial has been entered -- And during deletion, we remove the spline terms before the polynomial pieces

Fitting (part I)

For linear splines, the functions are broken lines in each variable -- The selection process first enters a simple linear function in a variable x_j and then entertains the spline terms $(x_j - t)_+$

Here's how these constraints play out in terms of the candidates available to add or delete at any step in the general algorithm two slides back...

Fitting (part I)

For an additive model using piecewise linear splines (a broken line in each variable), our candidate basis functions available at each step consist of

$x_j, j = 1, \ldots, d$ and

$(x_j - t_{j,m})_+$ if x_j is in the model

During backward deletion, we want to make sure we remove truncated power basis elements before the corresponding linear term -- At each step of the deletion process, the candidates for removal are

$(x_j - t_{j,m})_+$ and

x_j if there are no terms of the form $(x_j - t_{j,m})_+$ in the model

Looking at the chain of models generated in this way, the “best” can be selected using one of the selection criterion (Cp, AIC, BIC) or via a test set or even using cross validation

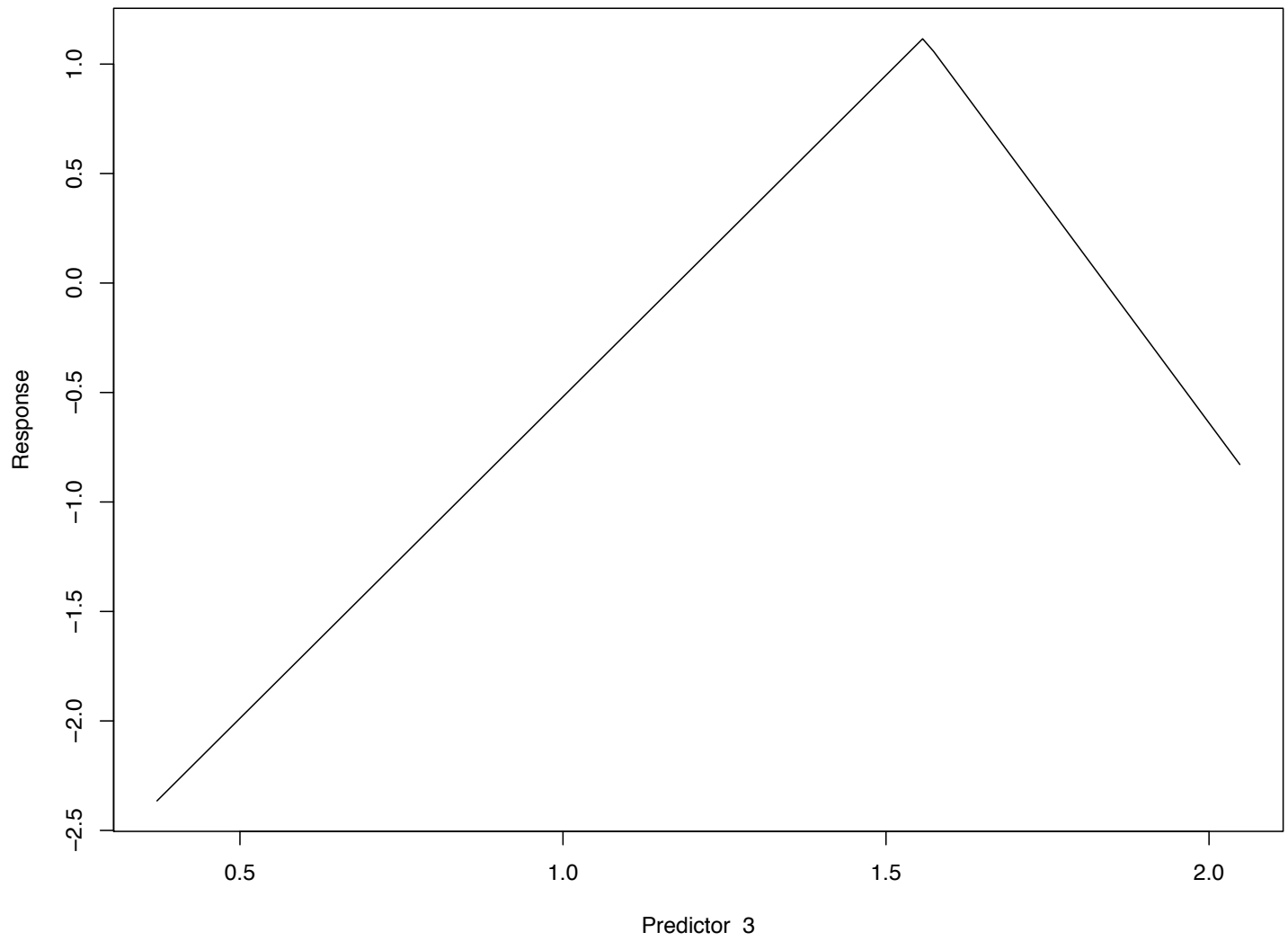
Fitting (part I)

If we want a model with at most two-factor interactions, we might impose the following constraints on the candidate addition set

$x_j, j = 1, \ldots, d$ and
$(x_j - t_{j,m})_+$ if x_j is in the model
$x_{j_1} x_{j_2}$ if x_{j_1} and x_{j_2} are in the model
$x_{j_1}(x_{j_2} - t_{j_2,m})_+$ if x_{j_1} x_{j_2} and $(x_{j_2} - t_{j_2,m})_+$ are in the model
$(x_{j_1} - t_{j_1,m_1})_+(x_{j_2} - t_{j_2,m_2})_+$ if $x_{j_1}(x_{j_2} - t_{j_2,m_2})_+$ and $x_{j_2}(x_{j_1} - t_{j_1,m_1})_+$ are in the model

Clearly this is just a bookkeeping exercise, and thankfully one that we don't have to do on our own -- The PolyMARS algorithm from Charles Kooperberg implements this strategy

```
> polymars(vul$ln_death_risk,vul[,c("ln_pop","ln_events","ln_fert","hdi")])
Call:
polymars(responses = vul$ln_death_risk, predictors = vul[, c("ln_pop",
    "ln_events", "ln_fert", "hdi")])

Model fitting

   0/1 size      RSS       GCV
1    1    1 438.2049  3.219465
2    1    2 383.5404  2.986041
3    1    3 286.1273  2.364689
4    1    4 243.2221  2.137694
5    1    5 203.6456  1.907191
6    1    6 198.8474  1.988474
7    1    7 189.3154  2.025968
8    1    8 182.6697  2.096973
9    1    9 175.1646  2.162526
10   1   10 168.4340  2.242465
11   1   11 162.4136  2.338755
12   1   12 158.1545  2.471164
13   1   13 156.0660  2.655187
14   1   14 154.1593  2.866599
...
62   0    2 401.4299  3.125320
63   0    1 438.2049  3.219465


Model produced

  pred1    knot1 pred2 knot2      coefs        SE
1     0       NA     0    NA -4.3963974 0.5157676
2     3       NA     0    NA  2.9370508 0.3132665
3     3 1.558145     0    NA -6.9173484 1.1380678
4     2       NA     0    NA  1.1998149 0.1606159
5     1       NA     0    NA -0.4721077 0.0908349

Rsquared : 0.535
> fit <- polymars(vul$ln_death_risk,vul[,c("ln_pop","ln_events","ln_fert","hdi")])
> plot(fit,3)
```


Response
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0.5
1.0
1.5
2.0
Predictor 3

Fitting (part I)

PolyMARS is half acronym -- MARS stands for Multivariate Adaptive Regression Splines, a procedure developed by Jerry Friedman in 1990, citing similarities with the tree-based procedures we saw earlier

Similar algorithms were developed by Breiman (DKCV -- Delete knot, cross validate, and PIMPLE, the PI implementation) and Hastie and Silverman (TURBO)

In short, these routines take the adaptive strategy we mentioned for univariate fitting using the truncated power basis -- A method that's essentially the approach taken by Patricia Smith

Comments

These procedures are aggressively adaptive, identifying both the important variables and their functional form -- They are also useful because they provide simple graphics to examine the different functional components, viewing the main effects and interactions

The selection criteria embedded in these procedures account for the expansive search by increasing the penalties associated with model size (in effect correcting the degrees of freedom used in finding the basis elements)

The truncated power basis is not the only technique for working with “free knot” splines -- Other work attempts to treat the knot locations as another parameter and minimizing their configuration in a less greedy way

Finally, Luo and Wahba have proposed a hybrid selection/shrinkage routine in which knots are placed adaptively and then penalized fit is performed in this space (using the second derivative penalty brought up last lecture)

In short, the 90s was a time of incredible activity in this area!

Fitting (part II)

Using splines, of course, doesn't mean we have to place knots in an adaptive way -- Instead we can just place them at equally spaced quantiles of the input data

This puts them in the same class of smoothers as local polynomials and smoothing splines, at least superficially, in that they have a single "handle" to control the amount of smoothness and the flexibility of these spaces at some point is related to the density input data nearby

In this case, they can, like all these other procedures, be written as a linear smoother with $\hat{y} = Sy$ for a (possibly hard to actually compute) n-by-n matrix S

Fitting (part II)

Buja, Hastie and Tibshirani refer to these as “linear smoothers” in the sense that the predictions at the design points are just linear combinations of the observed data -- Contrast this to the adaptive knot schemes we just presented in which the smoother depends heavily on the input data

These authors present a simple approach to using these “black box” smoothers to fit an additive model -- It involves a process known as backfitting...

Iterative solutions for linear systems

When looking at how to derive OLS estimates, we focused mainly on “direct methods” that involved various matrix decompositions of the model matrix -- The QR decomposition, the SVD and even the LU decomposition implied by Gaussian elimination

In contrast to these direct methods are so-called “iterative methods” which generate a sequence of approximate solutions -- Backfitting builds from one such method, Gauss-Seidel iterations

Iterative approaches

If we are trying to solve a linear system of the form $Ax = b$ where A is a 3-by-3 matrix -- Assuming it has no zero diagonal elements, the solution can be written as

$$
\begin{aligned}
x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\
x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\
x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}
\end{aligned}
$$

If we have an approximation x^k to the solution, a natural way to generate a new approximation x^{k+1} is to compute

$$
\begin{aligned}
x_1^{k+1} &= (b_1 - a_{12}x_2^k - a_{13}x_3^k)/a_{11} \\
x_2^{k+1} &= (b_2 - a_{21}x_1^k - a_{23}x_3^k)/a_{22} \\
x_3^{k+1} &= (b_3 - a_{31}x_1^k - a_{32}x_2^k)/a_{33}
\end{aligned}
$$

Gauss-Seidel iterations

Gauss-Seidel is only slightly different, making use of the updates as they are available -- That is

$$
\begin{aligned}
x_1^{k+1} &= (b_1 - a_{12}x_2^k - a_{13}x_3^k)/a_{11} \\
x_2^{k+1} &= (b_2 - a_{21}x_1^{k+1} - a_{23}x_3^k)/a_{22} \\
x_3^{k+1} &= (b_3 - a_{31}x_1^{k+1} - a_{32}x_2^{k+1})/a_{33}
\end{aligned}
$$

Another view

For the previous version of our normal linear model we assumed that

$$E[Y|X = x] = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

If instead of conditioning on the complete vector X, we use just a single variable, we find that

$$\begin{aligned} E[Y|X_k = x_k] &= E[E[Y|X_1, \ldots, X_k, \ldots, X_p]|X_k = x_k] \\ &= E\left[\sum_{j=0}^{p} \beta_j X_j | X_k = x_k\right] \\ &= \beta_k x_k + E\left[\sum_{j \neq k}^{p} \beta_j X_j | X_k = x_k\right] \end{aligned}$$

Another view

Turning this around we find that

$$\begin{aligned}\beta_k x_k &= E[Y|X_k = x_k] - E\left[\sum_{j\neq k}\beta_j X_j | X_k = x_k\right] \\ &= E\left[Y - \left(\sum_{j\neq k}\beta_j X_j\right) | X_k = x_k\right]\end{aligned}$$

The term in the second line is a partial residual -- It's the difference between Y and what we expect it to be ignoring the kth variable

Backfitting

This equivalence suggests an iterative process in which we cycle through the predictors, regressing the partial residuals on the “left out” variable -- This is known as backfitting

It’s an legitimate alternative to fitting least squares problems (or more generally solving linear systems) especially when we have lots of variables or if our data are sparse in some way

In statistics, this method comes up not just to fit a linear regression but also in the context of fitting an additive model with one of the “linear smoothers” we mentioned earlier...

Center the response and all of predictors

Until the estimates don't change much

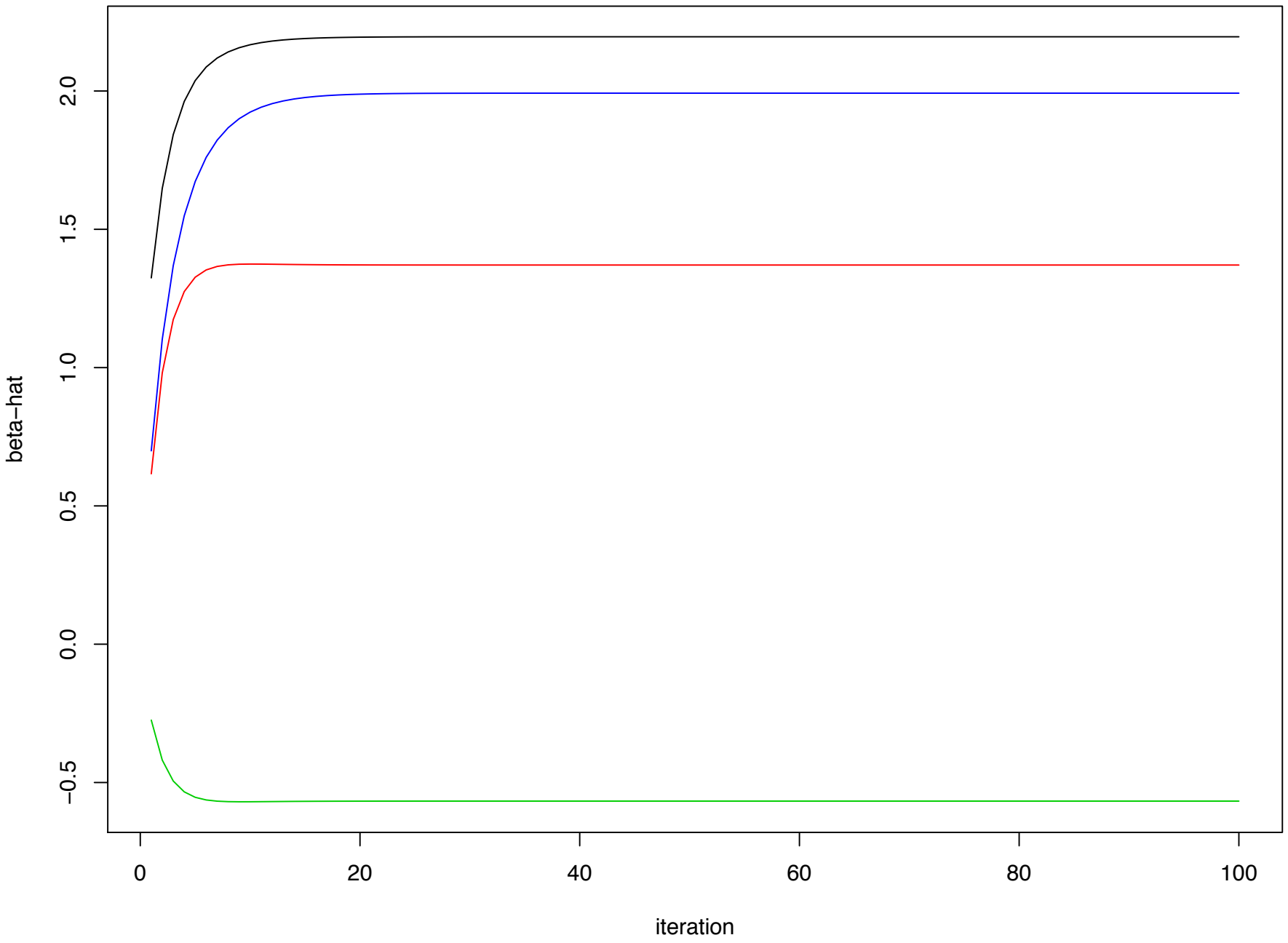
For k = 1,...,p

Form the kth partial residuals $y_k = y - \sum_{j \neq k} \widehat{\beta}_j x_j$

Regress y_k on the kth predictor $x_k = (x_{1k}, \ldots, x_{nk})^t$ and call the coefficient c_k

Set $\widehat{\beta}_k = c_k$

```
# gauss-seidel iterations to fit the vulnerability regression

y <- vul$ln_death_risk
M <- as.matrix(vul[,c("ln_fert","ln_events","ln_pop","hdi")])
M <- M-matrix(apply(M,2,mean),ncol=ncol(M),nrow=nrow(M),byrow=T)

y <- y-mean(y)
beta <- rep(0,ncol(M))

# store all the coefficients as we iterate (not strictly needed
# but we want a pretty picture at the end)
Beta <- NULL

for(i in 1:100){

   for(k in 1:ncol(M)){

      yk <- y - M[,-k]%*%beta[-k]
      ck <- coefficients(lm(yk~M[,k]-1))
      beta[k] <- ck
   }
   Beta <- rbind(Beta,beta)
}

matplot(1:100,Beta,type="l",xlab="iteration",ylab="beta-hat",lty=1)
```


beta−hat
2.0
1.5
1.0
0.5
0.0
−0.5
0
20
40
60
80
100
iteration

Backfitting and the additive model

If we really believed that our regression function has an additive form

$$E[Y|X = x] = \alpha + f(x_1) + \cdots + f(x_p)$$

If instead of conditioning on the complete vector X, we use just a single variable, we find that

$$\begin{aligned} E[Y|X_k = x_k] &= E[E[Y|X_1, \ldots, X_k, \ldots, X_p]|X_k = x_k] \\ &= E\left[\alpha + \sum_{j=1}^{p} f(X_j)|X_k = x_k\right] \\ &= f(x_k) + E\left[\alpha + \sum_{j\neq k}^{p} f(X_j)|X_k = x_k\right] \end{aligned}$$

Another view

Turning this around we find that

$$\begin{aligned} f(x_k) &= E[Y|X_k = x_k] - E\left[\alpha + \sum_{j \neq k} f(X_j)|X_k = x_k\right] \\ &= E\left[Y - \left(\alpha + \sum_{j \neq k} f(X_j)\right)|X_k = x_k\right] \end{aligned}$$

The term in the second line is a partial residual -- It's the difference between Y and what we expect it to be ignoring the kth variable

Backfitting and the additive model

Therefore, if we could estimate arbitrary one-dimensional functions we could fold that into a backfitting routine to estimate an additive model -- Ah, but here's where our linear smoothers come into play!

This is a second sense in which fitting an additive model is "no harder" than a simple univariate smooth -- We are just applying the same tool over and over again...

Center the response and all of predictors and form the smoother matrices $S_1, \ldots, S_p$

Until the estimates don't change much

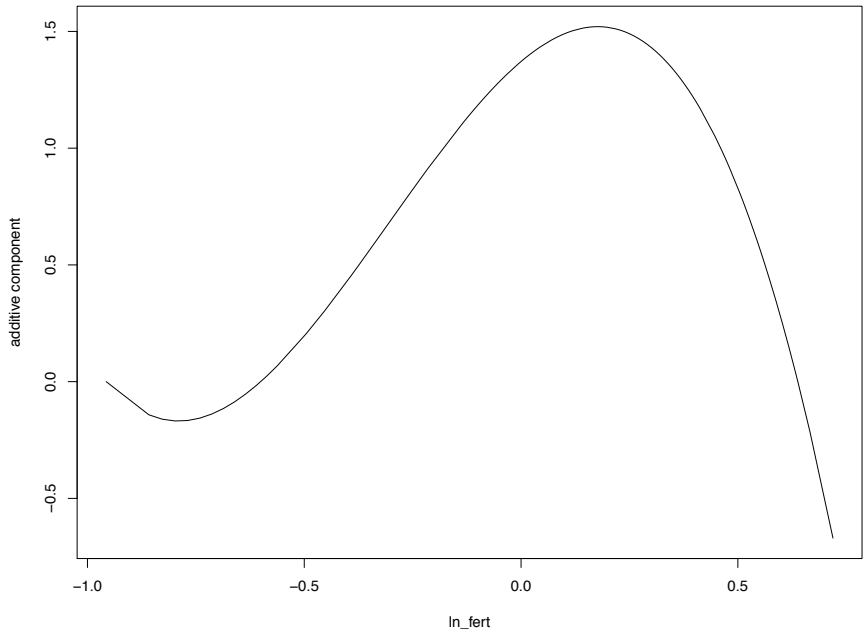
For k = 1,...,p

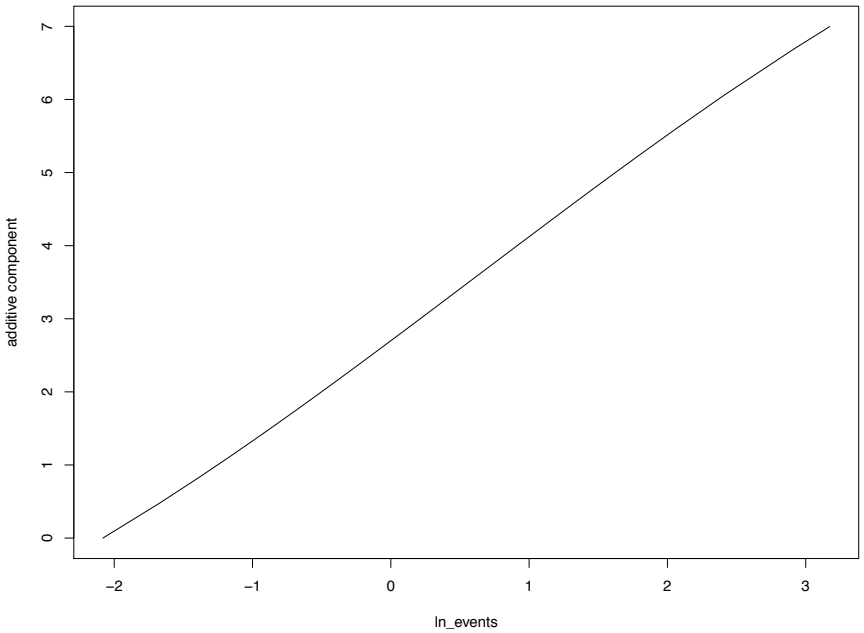
Form the kth partial residuals $y_k = y - \sum_{j \neq k} \widehat{f}_j$

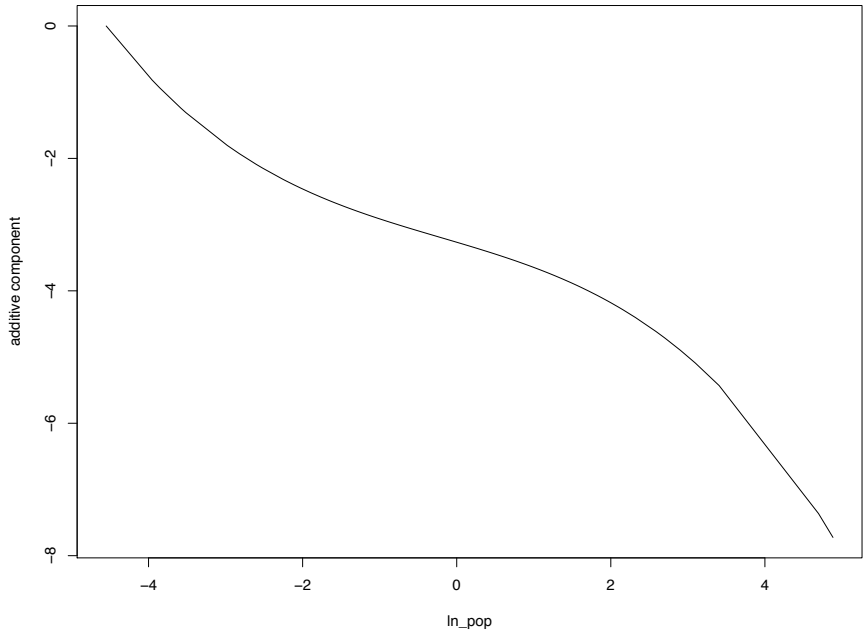
Smooth y_k using the kth predictor $x_k = (x_{1k}, \ldots, x_{nk})^t$ so that $s_k = S_k y_k$

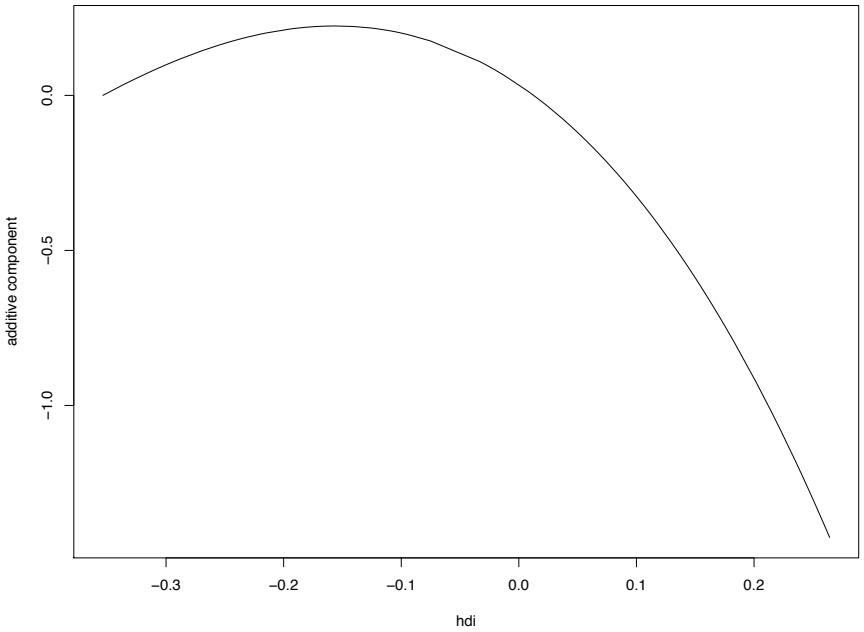
Set $\widehat{f}_k = s_k$

```
y <- vul$ln_death_risk
M <- as.matrix(vul[,c("ln_fert","ln_events","ln_pop","hdi")])
M <- M-matrix(apply(M,2,mean),ncol=ncol(M),nrow=nrow(M),byrow=T)

y <- y-mean(y)
fits <- matrix(0,ncol=ncol(M),nrow=nrow(M))

for(i in 1:1000){

   for(k in 1:ncol(M)){

      yk <- y - apply(fits[,-k],1,sum)
      fk <- predict(lm(yk~bs(M[,k],df=3)-1))
      fits[,k] <- fk
   }
}

plot(sort(M[,1]),fits[order(M[,1]),1],
     xlab="ln_fert",ylab="additive component",type="l")

plot(sort(M[,2]),fits[order(M[,2]),2],
     xlab="ln_events",ylab="additive component",type="l")

plot(sort(M[,3]),fits[order(M[,3]),3],
     xlab="ln_pop",ylab="additive component",type="l")

plot(sort(M[,4]),fits[order(M[,4]),4],
     xlab="hdi",ylab="additive component",type="l")
```


additive component
1.5
1.0
0.5
0.0
−0.5
−1.0
−0.5
0.0
0.5
ln_fert


additive component
7
6
5
4
3
2
1
0
−2
−1
0
1
2
3
ln_events


additive component
0
−2
−4
−6
−8
−4
−2
0
2
4
ln_pop


additive component
0.0
−0.5
−1.0
−0.3
−0.2
−0.1
0.0
0.1
0.2
hdi

Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.

Is a county more than 20 percent black?

NO There are not many African-Americans in this county.

YES This county has a large African-American population.

Obama wins these counties 383 to 70.

And is the high school graduation rate higher than 78 percent?

NO This is a county with less-educated voters.

YES This is a county with more educated voters.

Clinton wins these counties 704 to 89.

And is the high school graduation rate higher than 87 percent?

NO 78 to 87 percent have a diploma.

YES This is a highly educated county.

Obama wins these counties 185 to 36.

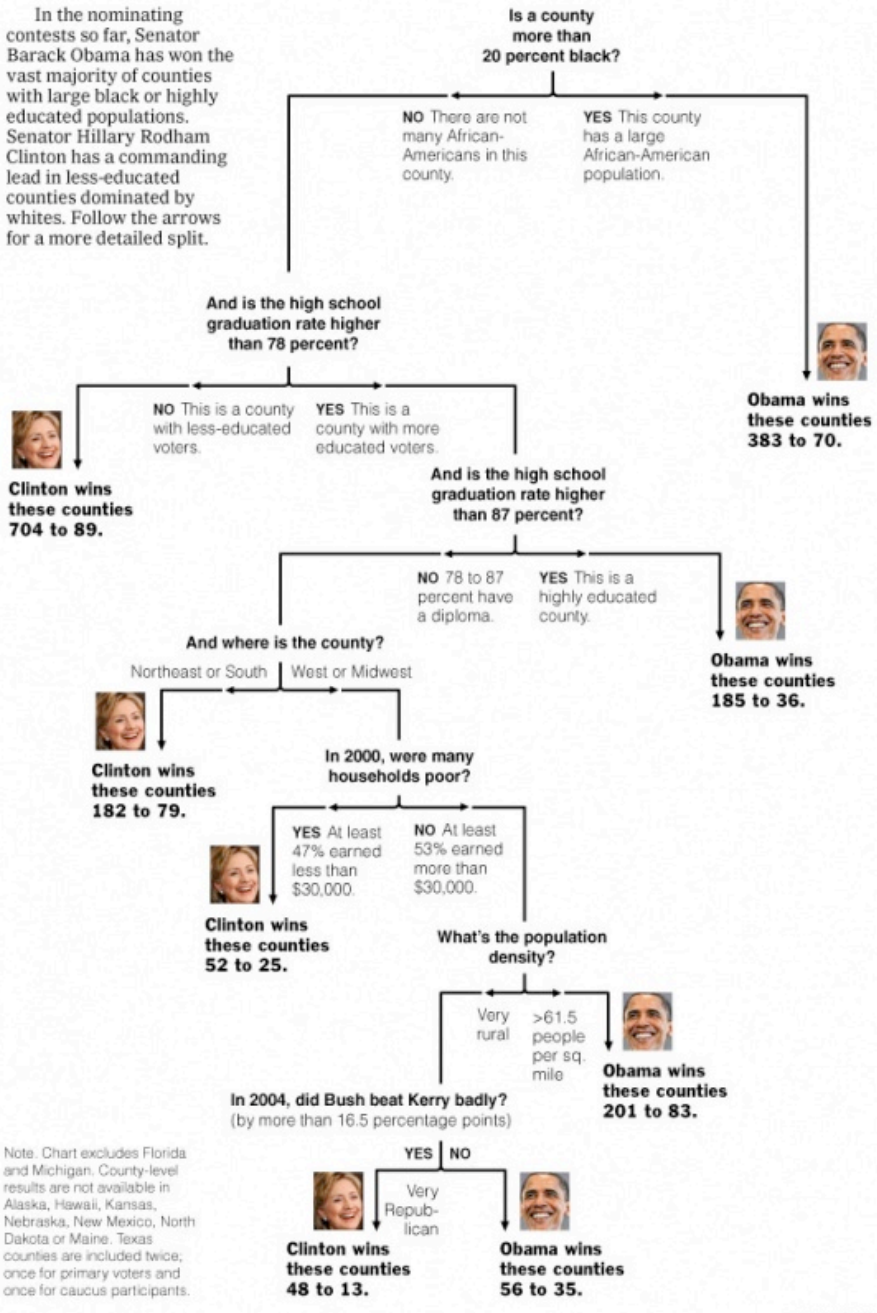
And where is the county?

Northeast or South | West or Midwest

In 2000, were many

Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.



Note. Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice, once for primary voters and once for caucus participants.

Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
THE NEW YORK TIMES

Decision trees

The figure on the previous slide was made by **Amanda Cox**, a statistician working in the graphics department at the New York Times (and yes, the analysis was done in R)

The title “Decision Tree” is a pun, as the data refer to county-wide decisions for Obama or Clinton in the Democratic Primary; **the object is also a kind of statistical model known as a decision tree**

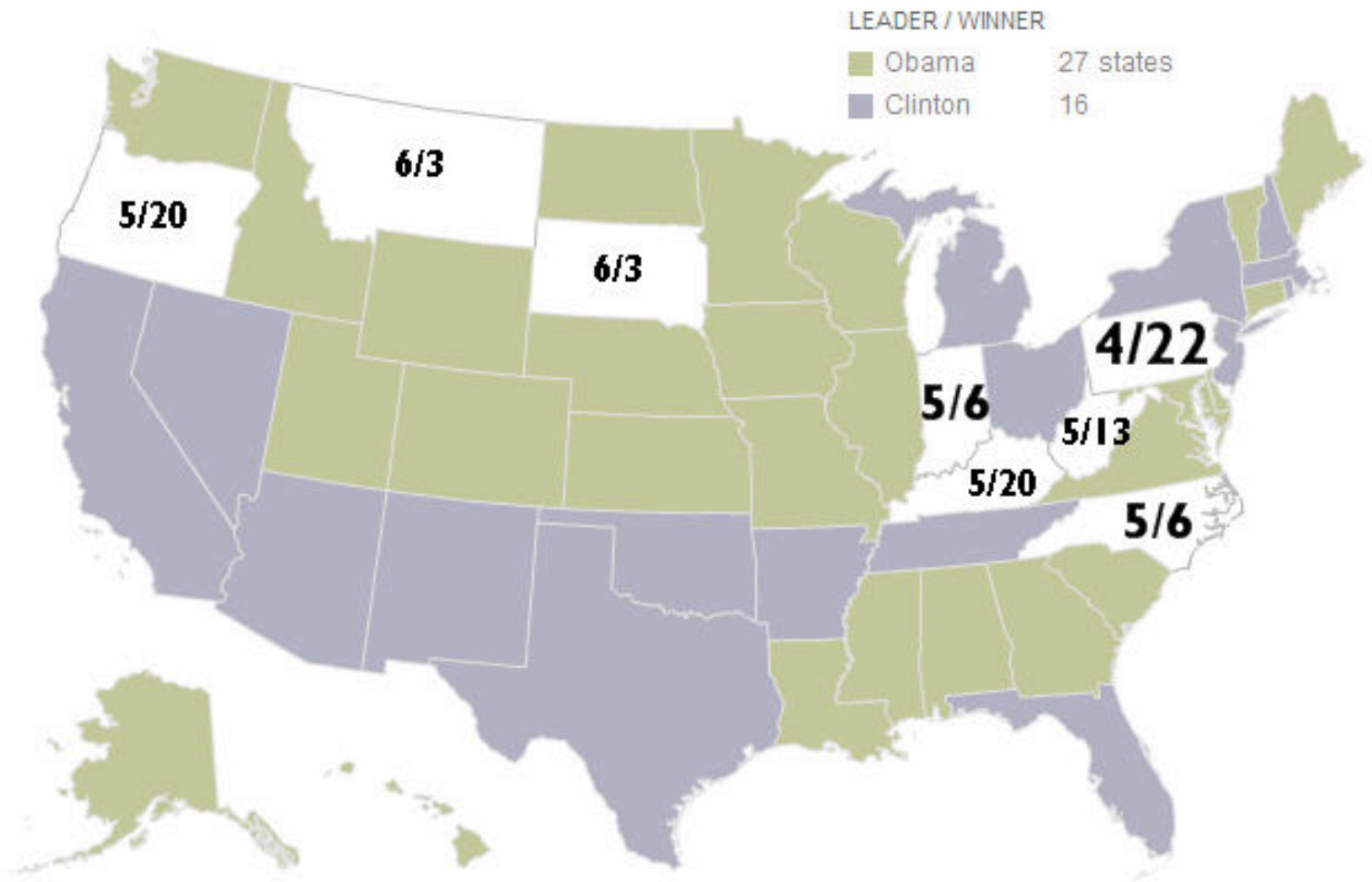
Coded in this elegant structure is **a pretty serious computational algorithm**; a relatively recent addition to the statistician’s toolbox

The Democratic Primary

Think back to April of last year, when it seemed Hillary Clinton's momentum was picking up again; the decision tree was built from a data set concerning **all the counties in states where primaries had already been held**

The unit of observation, then, is a county and variables included various **demographic measures** (age and ethnic makeup, education level, religious breakdown), **political measures** (did the county go to Bush or Kerry in 04) and **economic factors** (unemployment rate, the amount of construction in the county), and so on

```
primary = read.csv(url("http://www.stat.ucla.edu/~cocteau/primaries.csv"),head=T)

names(primary)
dim(primary)

# transformations

primary$black06pct = primary$black06/primary$pop06
primary$hisp06pct = primary$hisp06/primary$pop06
primary$white06pct = primary$white06/primary$pop06
primary$growth = 100*(primary$pop06/primary$pop00 - 1)

# drop some counties
primary = subset(primary, state_postal!="MI" )
primary = subset(primary, state_postal!="FL" )
primary = subset(primary, !(state_postal=="WA" & racetype=="Primary") )

table(primary$winner)

primary[primary$state=="CA",c("winner","county_name")]
```


LEADER / WINNER
Obama 27 states
Clinton 16
5/20
6/3
6/3
4/22
5/6
5/13
5/20
5/6

	winner	county_name
156	obama	Alameda
157	obama	Alpine
158	clinton	Amador
159	clinton	Butte
160	clinton	Calaveras
161	clinton	Colusa
162	clinton	Contra Costa
163	clinton	Del Norte
164	clinton	El Dorado
165	clinton	Fresno
166	clinton	Glenn
167	obama	Humboldt
168	clinton	Imperial
169	clinton	Inyo
170	clinton	Kern
171	clinton	Kings
172	clinton	Lake
173	obama	Lassen
174	clinton	Los Angeles
175	clinton	Madera
176	obama	Marin
177	clinton	Mariposa
178	obama	Mendocino
179	clinton	Merced
180	clinton	Modoc
181	obama	Mono
182	clinton	Monterey
183	clinton	Napa
184	obama	Nevada
185	clinton	Orange
186	clinton	Placer
187	obama	Plumas
188	clinton	Riverside
189	obama	Sacramento
190	clinton	San Benito
191	clinton	San Bernardino
192	clinton	San Diego
193	obama	San Francisco
194	clinton	San Joaquin
195	obama	San Luis Obispo
196	clinton	San Mateo
197	obama	Santa Barbara
198	clinton	Santa Clara
199	obama	Santa Cruz
200	clinton	Shasta
201	obama	Sierra
202	obama	Siskiyou
203	clinton	Solano
204	obama	Sonoma
205	clinton	Stanislaus
206	clinton	Sutter
207	clinton	Tehama
208	obama	Trinity
209	clinton	Tulare
210	clinton	Tuolumne
211	clinton	Ventura
212	obama	Yolo
213	clinton	Yuba

The Democratic Primary

Now, given all these potential predictor variables, **which ones "explain" county-level voting patterns?** Suppose, for example, we start simply, and consider whether or not a majority of the county voted for Bush in 04

	B04	K04
clinton	171	1039
obama	302	728

Therefore, Obama won about 64% of those counties not voting for Bush in 04, while Clinton won about 59% of those counties that did vote for Bush in 04

Now, consider the **simple prediction rule**: If a county voted for Bush in 04, we'll say that they will vote for Clinton in the primary, while if a county was mostly in favor of Kerry in 04, we'll assign the win to Obama

The Democratic Primary

Of course **this rule isn't perfect**; by applying it, we would make 171 + 728 = 899 mistakes (out of 2,240 counties, or about 40% error); we refer to these mistakes as having been "misclassified" by our simple rule

So the question becomes, **can we do any better?** There might be better indicators of Obama's success besides the vote in 2004 -- but how do we find them?

The decision tree encodes a large search across the complete data set; consider the top of the tree, "the root"...

The Democratic Primary

Decision trees work by repeatedly splitting the data into two parts; the root or first "split" is **the single division of the data into two pieces that produces the lowest misclassification error**

To investigate this a little further, let's consider the predictor that represents the percentage of a county that is African American; we now choose a breakpoint α that divides the data into two pieces (those counties with a greater percentage of African Americans and those with a smaller percentage)

We then form a table (as we did for counties that went for Bush or Kerry in 2004) and count the misclassification rate...

The Democratic Primary

Suppose we take $\alpha = 0.1$; and we end up with the following table

	Pct Af Am > 0.1	
	FALSE	TRUE
clinton	989	221
obama	554	476

Of the 697 counties that are more than 10% African American, 221 went for Clinton and 476 went for Obama (68% in favor of Obama); since Obama won more counties in this group, we label the node “Obama” and we would be making 221 errors

In the remaining 1543 counties, 898 went for Clinton and 554 for Obama (64% in favor of Clinton); then we will label this group “Clinton” and we would make 554 errors

```
tmp = primary$winner[primary$black06pct>0.1]
table(tmp)

# clinton   obama
#     221     476

tmp = primary$winner[primary$black06pct<=0.1]
table(tmp)

# clinton   obama
#     989     554
```

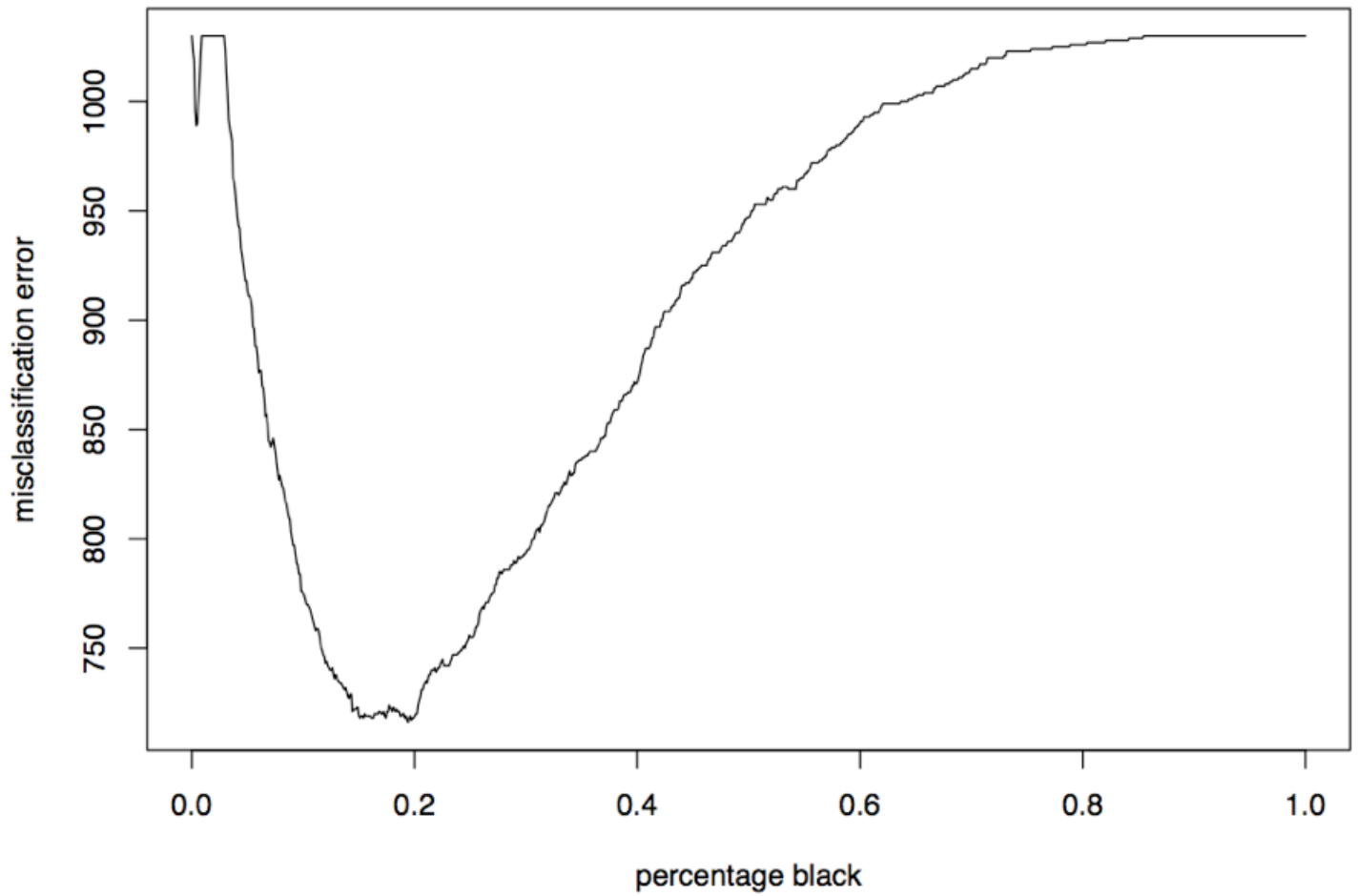
The Democratic Primary

The 20% figure at the top of the tree was obtained by finding the magic point $\alpha*$ that minimizes the misclassification errors

In fact, the search was conducted over **all the variables in the data set and all the possible splits**; and this choice produced the smallest error

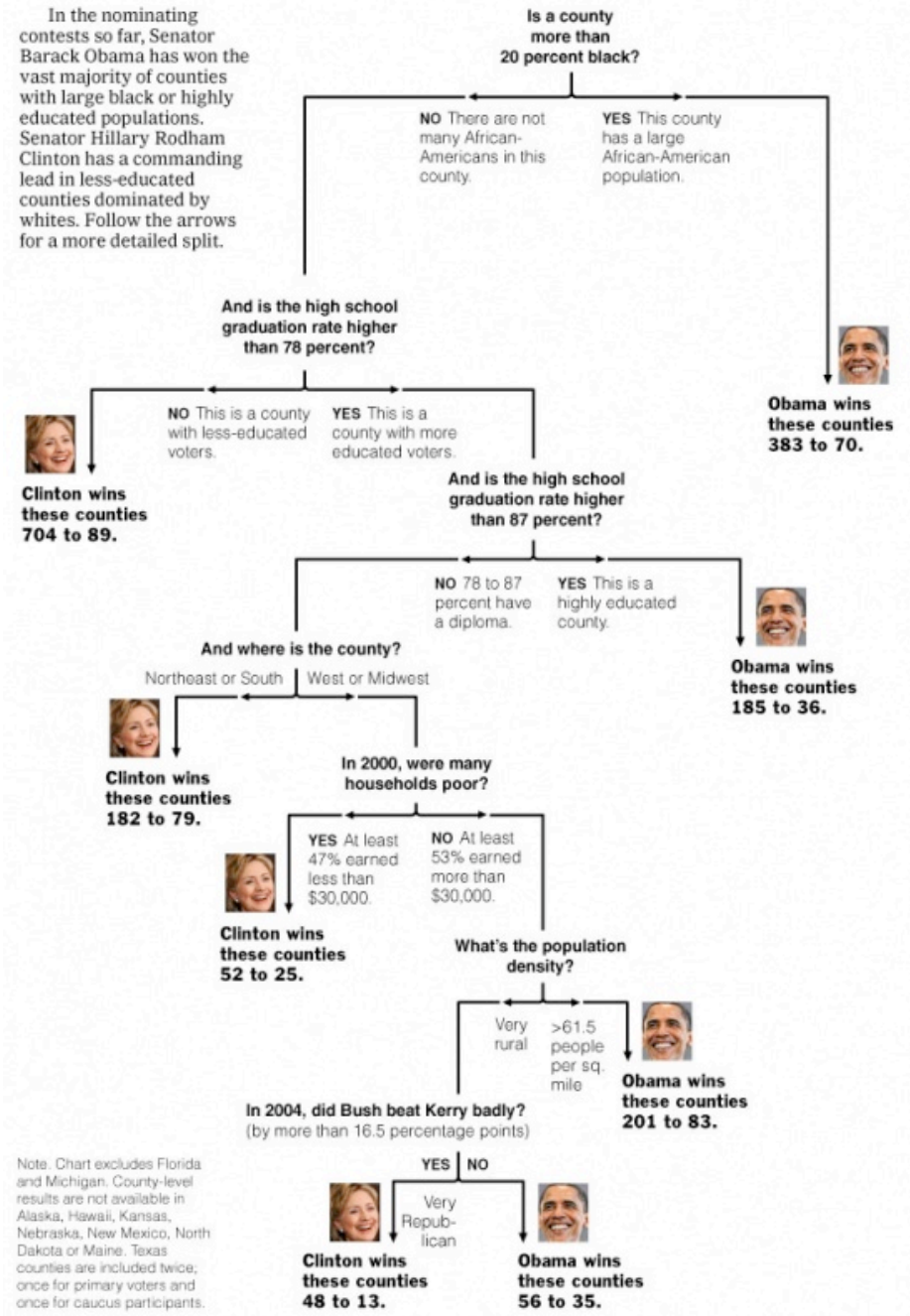
Once this node has been chosen, we work our way down the tree, conducting the same search but on the specified subsets of the data, **at each step attempting to minimize our errors**

```
num = 1000
error = rep(0,num)

fractions = seq(0,1,len=num)

for(i in 1:num){

  # right branch

  tmp = primary$winner[primary$black06pct>fractions[i]]
  error.right = min(table(tmp))

  # left branch

  tmp = primary$winner[primary$black06pct<=fractions[i]]
  error.left = min(table(tmp))
  error[i] = error.left+error.right
}

plot(fractions, error,
     xlab="percentage black",
     ylab="misclassification error",
     type="l")
```


misclassification error
750
800
850
900
950
1000
0.0
0.2
0.4
0.6
0.8
1.0
percentage black

Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.



Note. Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for caucus participants.

Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
THE NEW YORK TIMES

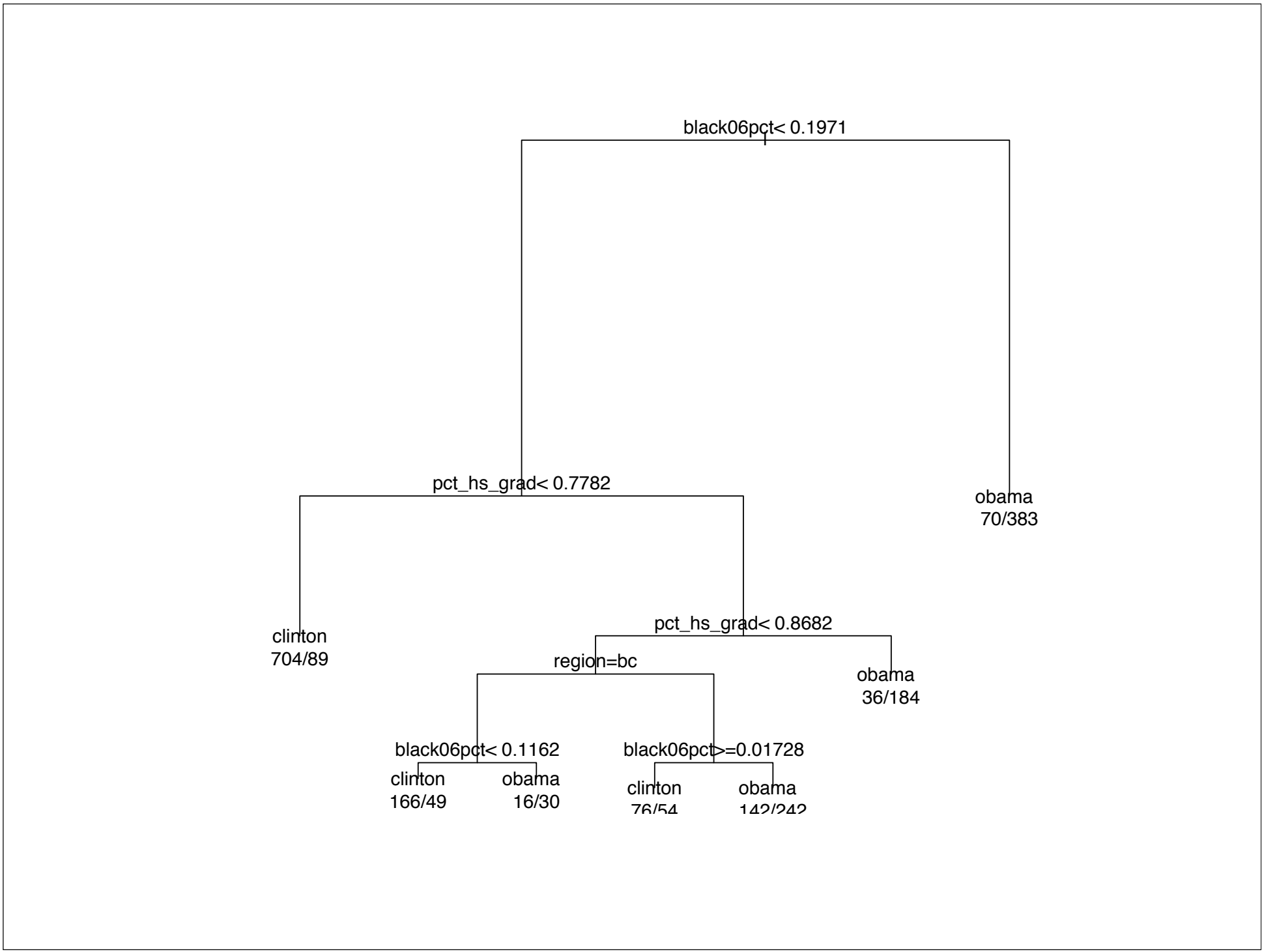
Tree-based models

Stop and think what this simple process has produced for us; we have a very **intuitive structure** (something akin to the game 20 questions) that makes evident **"important" variables that help "explain"** voting patterns

This kind of tool lives somewhere **between data analysis and modeling**; it is technically a model all by itself (making predictions) but is often used as a way to identify important predictors for another stage of model

This decision tree is part of a large class of methods called CART for Classification and Regression Trees and was developed in the 1980s as part of a move to deal with bigger and meaner data sets

```
# library rpart for recursive partitioning

library(rpart)

fit = rpart(winner~region+pct_hs_grad+black06pct,data=primary)

# look at the tree

plot(fit)
text(fit,use.n=T)

# how big should the tree be?
# cp here = complexity parameter not mallows' cp!

plotcp(fit)
```

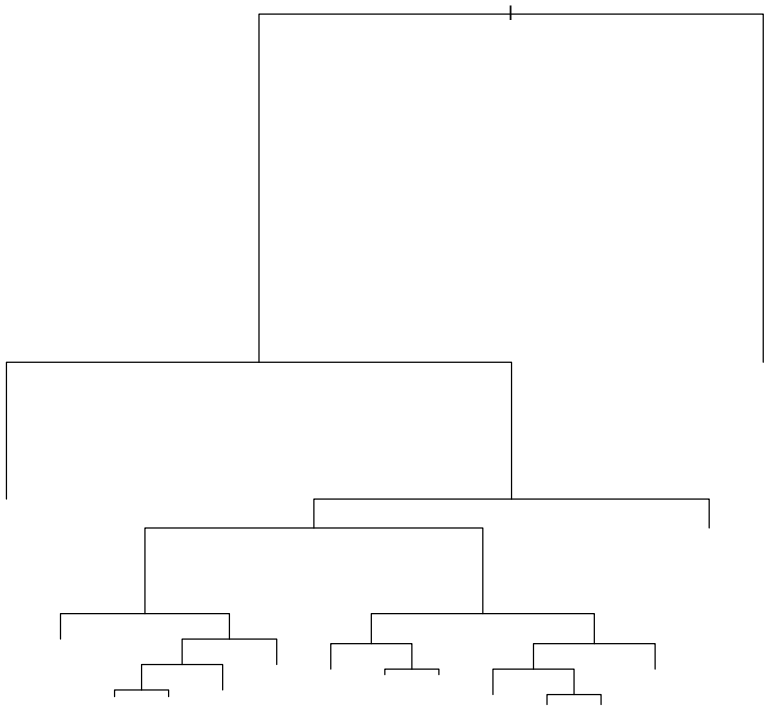

black06pct< 0.1971
pct_hs_grad< 0.7782
obama
70/383
clinton
704/89
pct_hs_grad< 0.8682
region=bc
obama
36/184
black06pct< 0.1162
black06pct>=0.01728
clinton
166/49
obama
16/30
clinton
obama

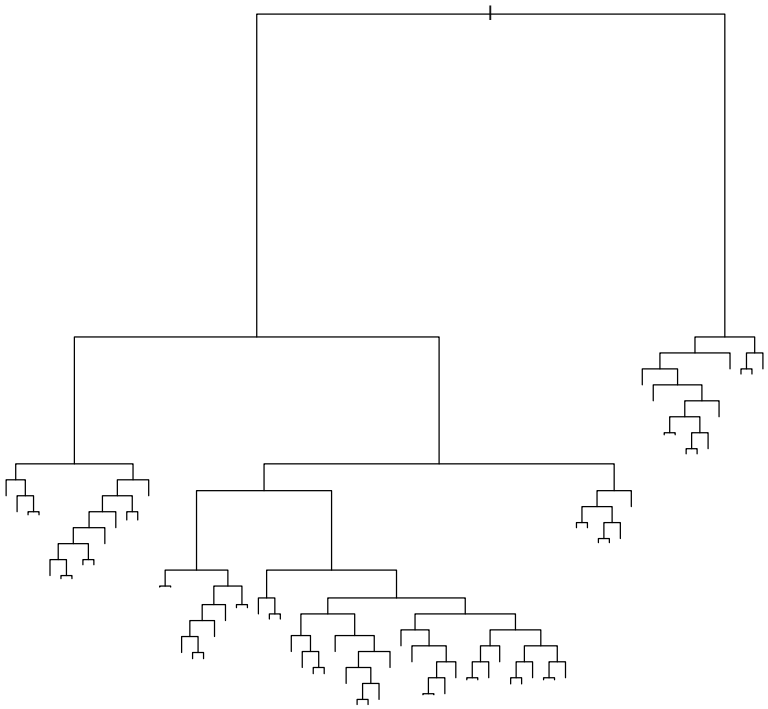
The Democratic Primary

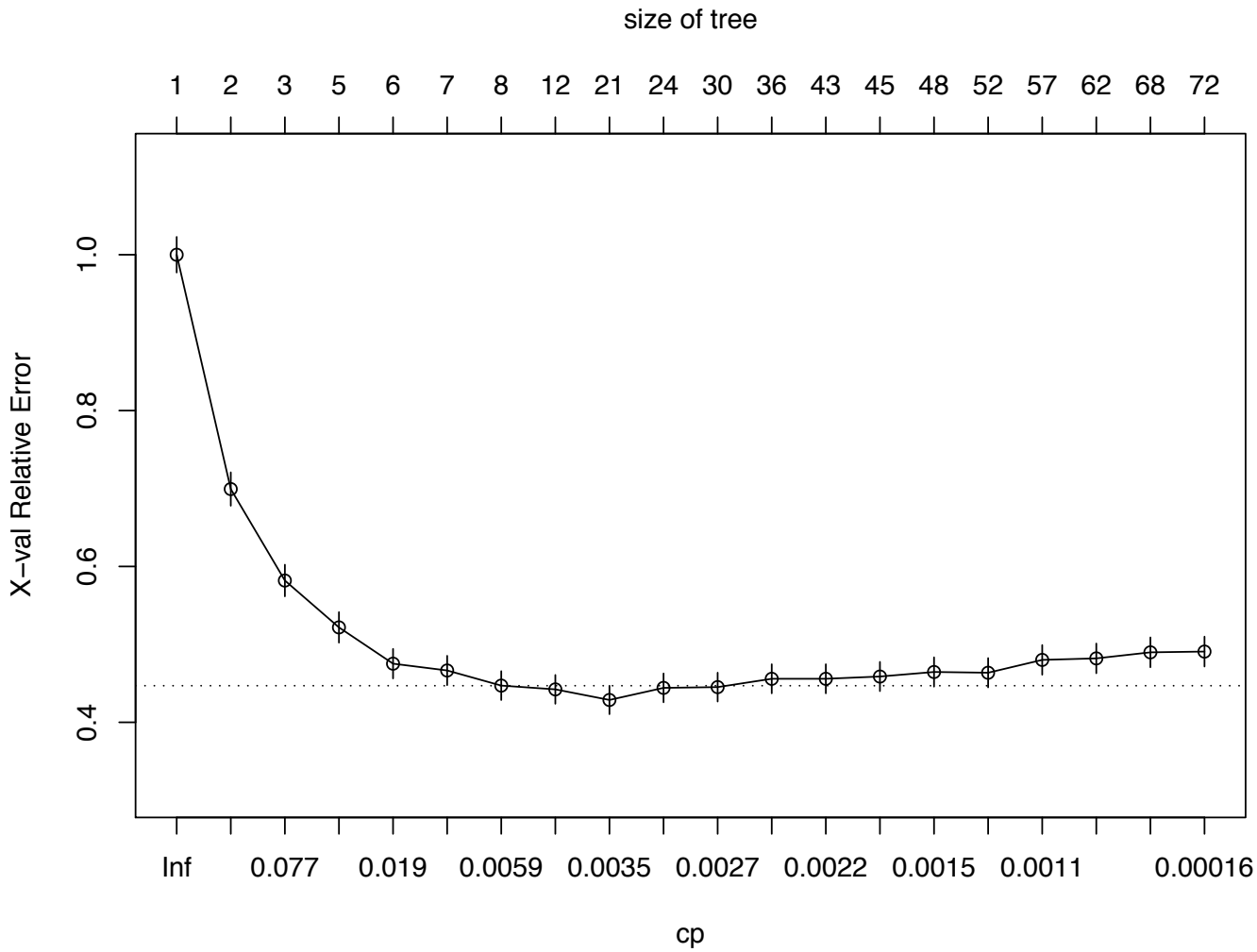
Let's think a bit more about the tree-growing process; with each split, **we cut our data at the node into two pieces** so that the "sample size" at each of the child nodes is lower than its parents

We then represent the data in the leaves with a simple model; for our 0/1 data (Obama or Clinton), we classify leaves according to majority vote

In principle we can grow trees until there's a single entry in each node -- What might be the problem with that? How do we decide to stop splitting? When we run out of data?






size of tree
1 2 3 5 6 7 8 12 21 24 30 36 43 45 48 52 57 62 68 72
X−val Relative Error
0.4 0.6 0.8 1.0
Inf 0.077 0.019 0.0059 0.0035 0.0027 0.0022 0.0015 0.0011 0.00016
cp

The Democratic Primary

In the R displays on the previous pages, **the heights of the branches correspond to the error in represented by the model**

So, the at the root, we are dealing with all 1240 counties; 1210 went for Clinton and 1030 for Obama -- therefore since Clinton won more counties, we would predict all future counties for Clinton, making 1030 mistakes

Last time we saw that the first split on the percentage of the county that is African American brought us down to 700 errors, a big drop; the next division based on education is another big drop, about half the size

As we continue to refine the tree, however, **the improvements diminish**...

The Democratic Primary

In the mid-1980s a fair bit of theoretical and methodological work was devoted to understanding the behavior of this kind of algorithm; **we are using it as a bit of data analysis** (how does a “response” relate to the potential “explanatory” variables?) but it can also be used as a tool for making predictions

The R command `rpart` (for recursive partitioning) employs a “pruning” algorithm that divides the original data into several parts; iteratively leaving one part out, building a tree and evaluating it on the part that was left out -- this is called cross validation

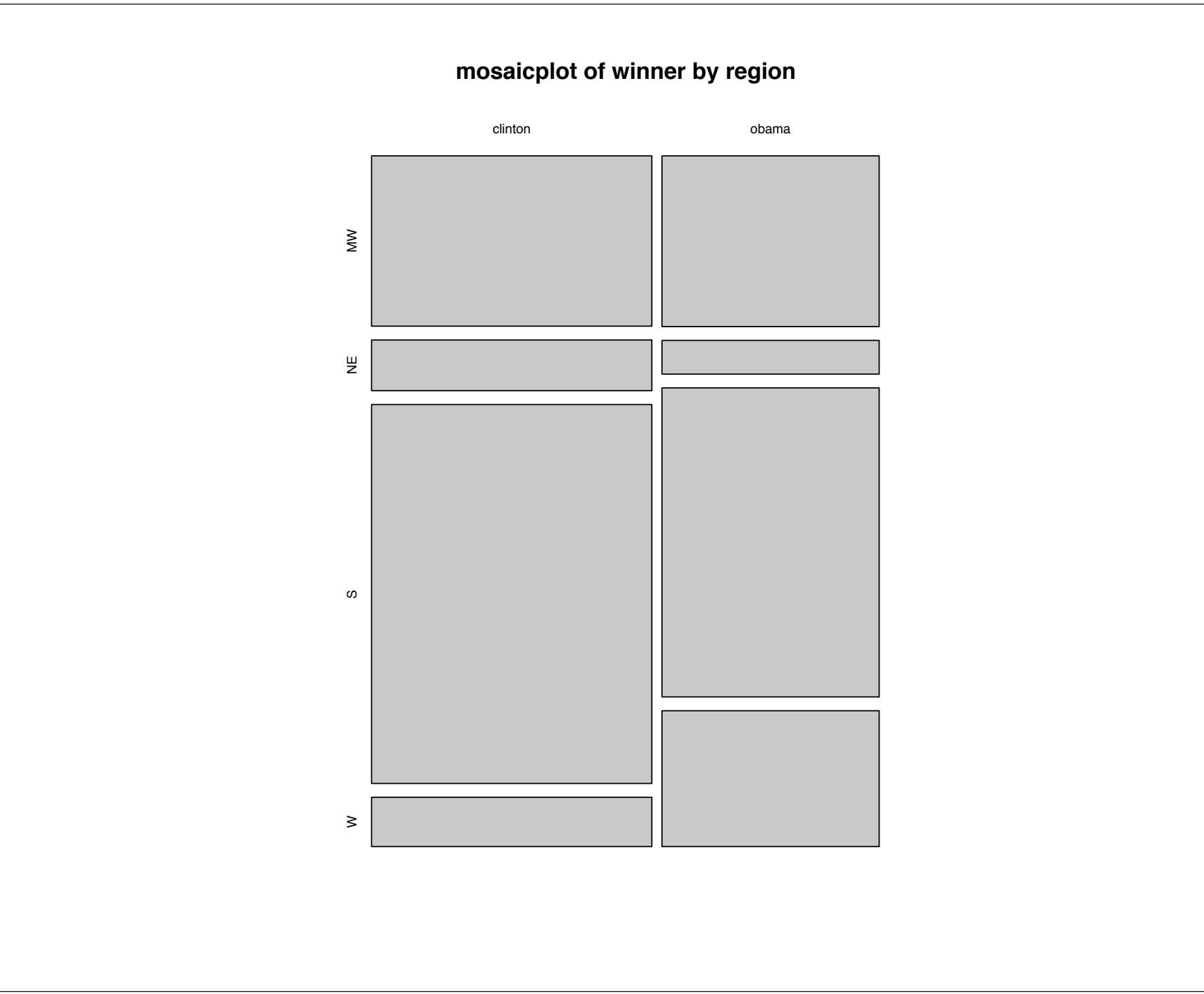
The Democratic primary

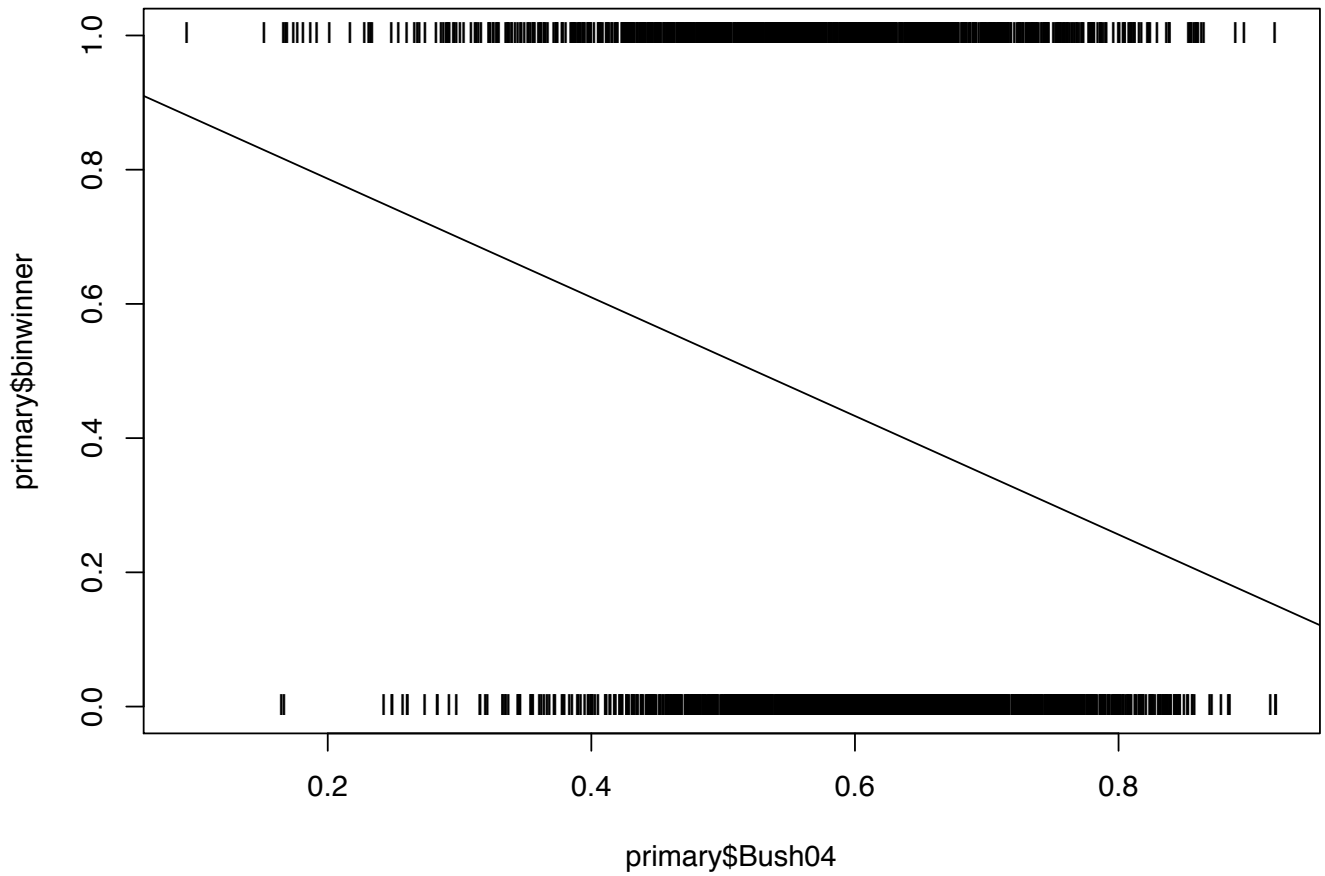
Tree models are beautiful solutions to difficult modeling problems; they are remarkably flexible and easy to read as a structure

They are, however, a huge departure from what we're used to in terms of regression models; over the next few weeks, we'll try to bring these closer together

As a start, we are going to look at a slightly older way to model with binary data, one that builds directly from your regression technology...

```
# let's a "look" at the data

# first a mosaic plot using the categorical variable "region"

mosaicplot(winner~region,data=primary)

# create a 1/0 version of the response, winner

primary$binwinner = as.numeric(primary$winner)-1

# a preliminary analysis?
plot(primary$Bush04,primary$binwinner,pch="|")

fit =lm(binwinner~Bush04,data=primary)
newpct = data.frame(Bush04=seq(0,1,len=100))
lines(newpct$Bush04,predict(fit,newdata=newpct))

plot(primary$black06pct,primary$binwinner,pch="|",ylim=c(0,1.5))

fit =lm(binwinner~black06pct,data=primary)
newpct = data.frame(black06pct=seq(0,1,len=100))
lines(newpct$black06pct,predict(fit,newdata=newpct))
```


mosaicplot of winner by region
clinton
obama
MW
NE
S
W

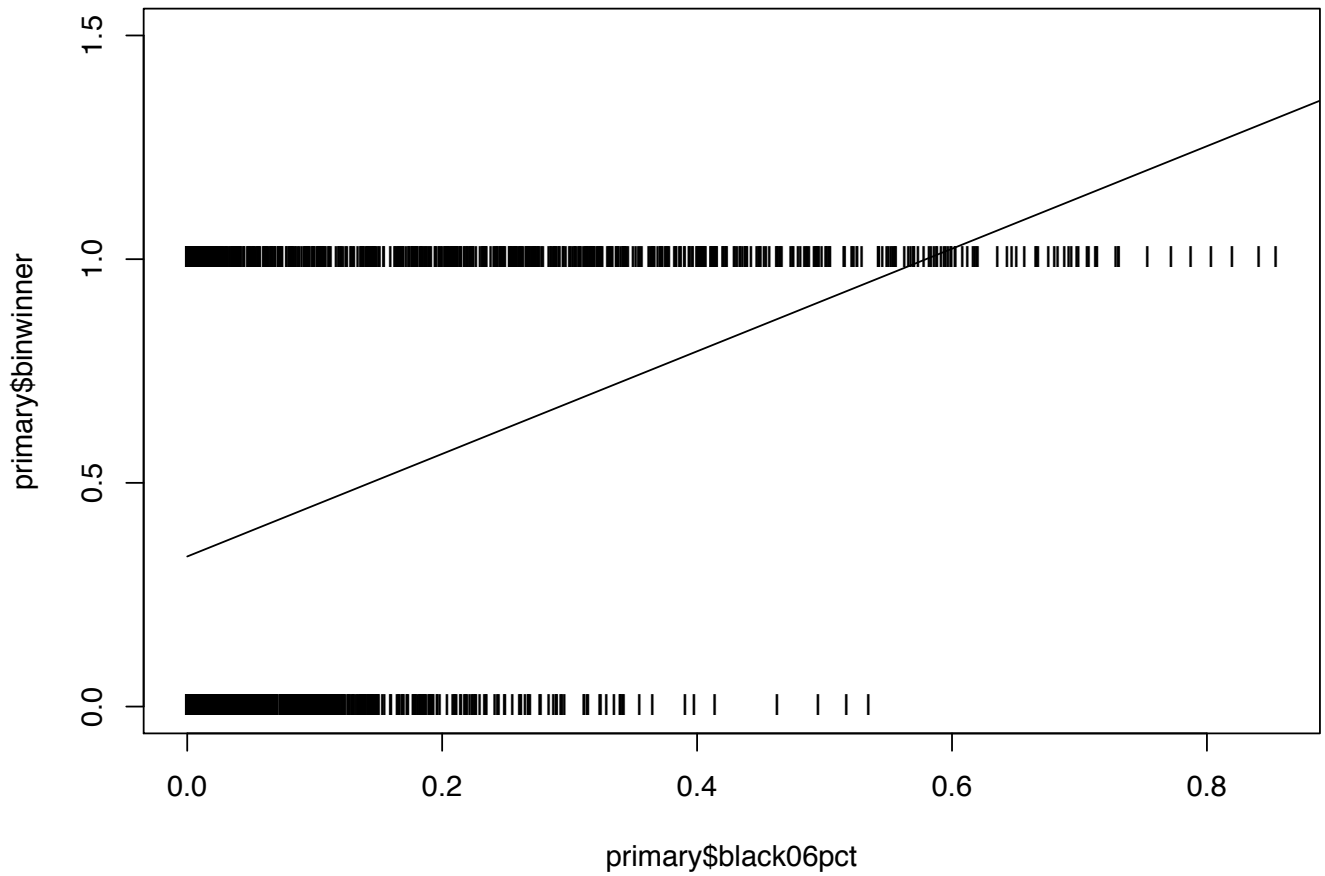

primary$binwinner
0.0
0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8
primary$Bush04

The Democratic primary

As a first pass at a model, we can try simply fitting a regression to 1/0 data; that is, use OLS to estimate the coefficients in a linear model where Obamma/ Clinton is recoded 1/0

In the case of the margin by which Bush won in 04, the predictions don't, on the face of it, look miserable; we should probably try to see if there are ways to probe the fit and see if it, in fact, makes sense

For the percentage of the county that was African American, however, we have a different picture...


primary$binwinner
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
primary$black06pct

Some formality

A statistical model begins with a (hypothetical) description of how our data were generated; we use probability to express the uncertainties involved

In both of our examples (the primary and the NYT data set), our response is binary, taking only the values 0 and 1; let's imagine that our observations are the result of a coin toss, where the properties of the coin depend on our covariates

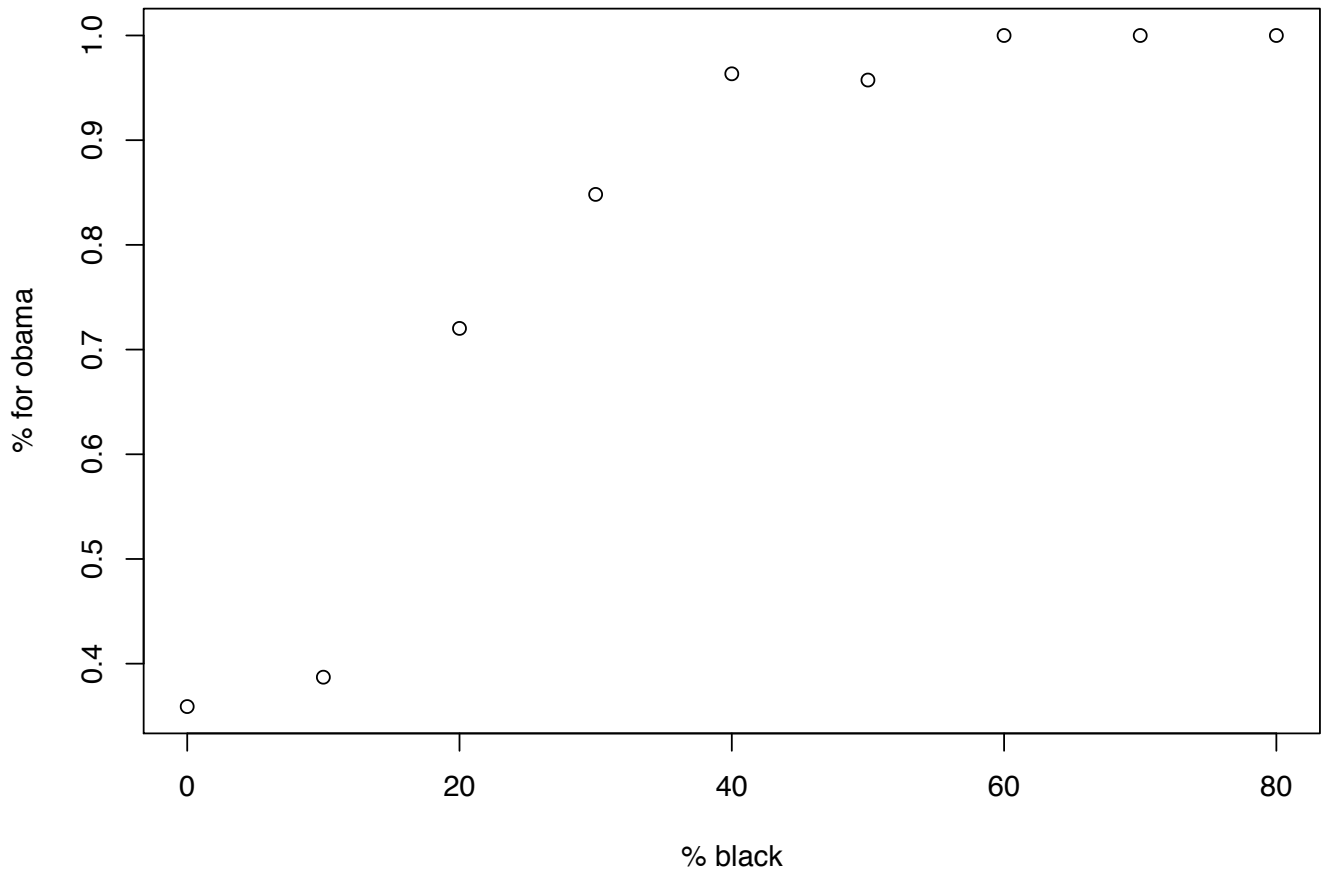
Keeping (largely) the notation from the linear model, we could say that our observation y has a Bernoulli distribution with "success" probability p(x), where x represents our covariates

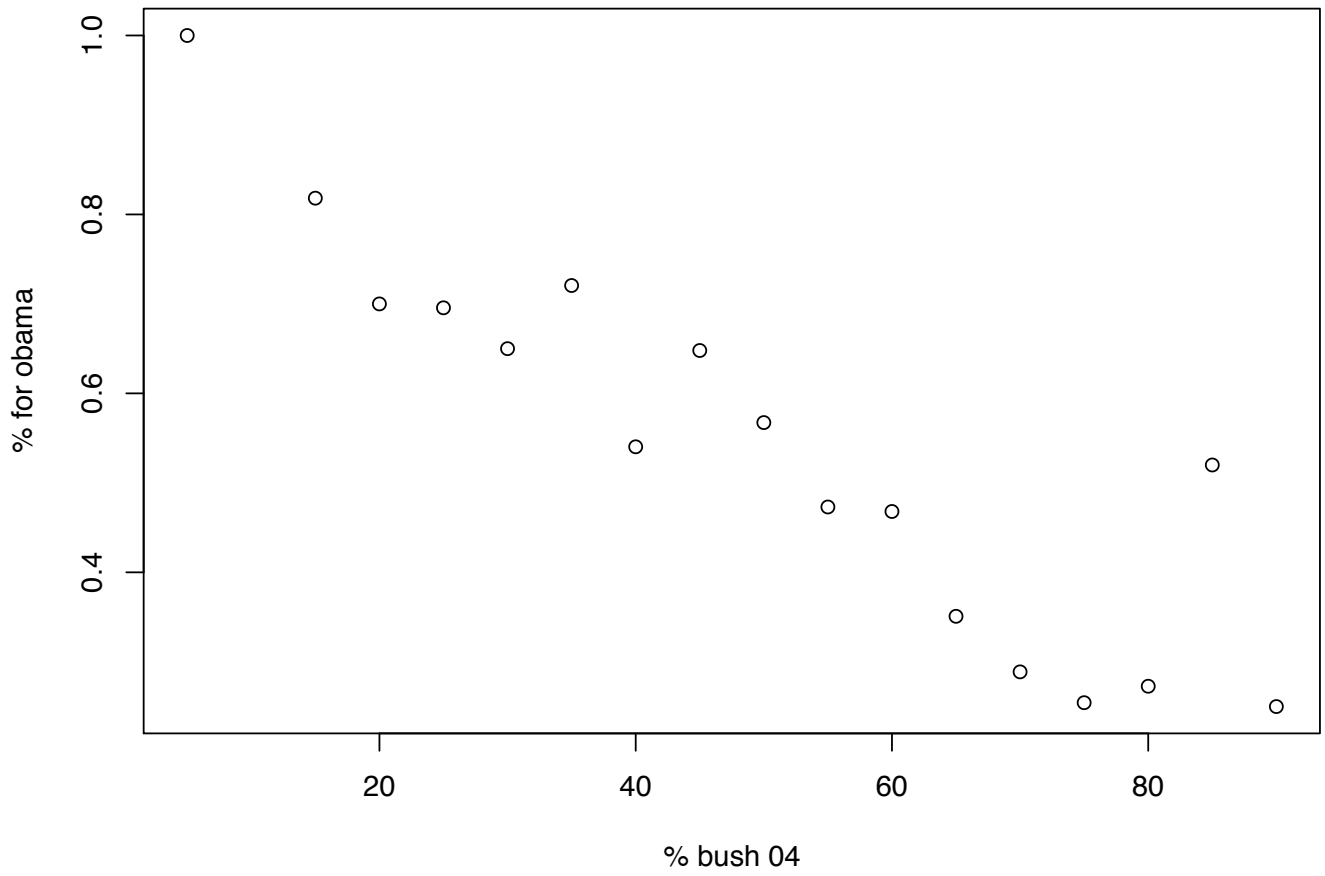
Implications

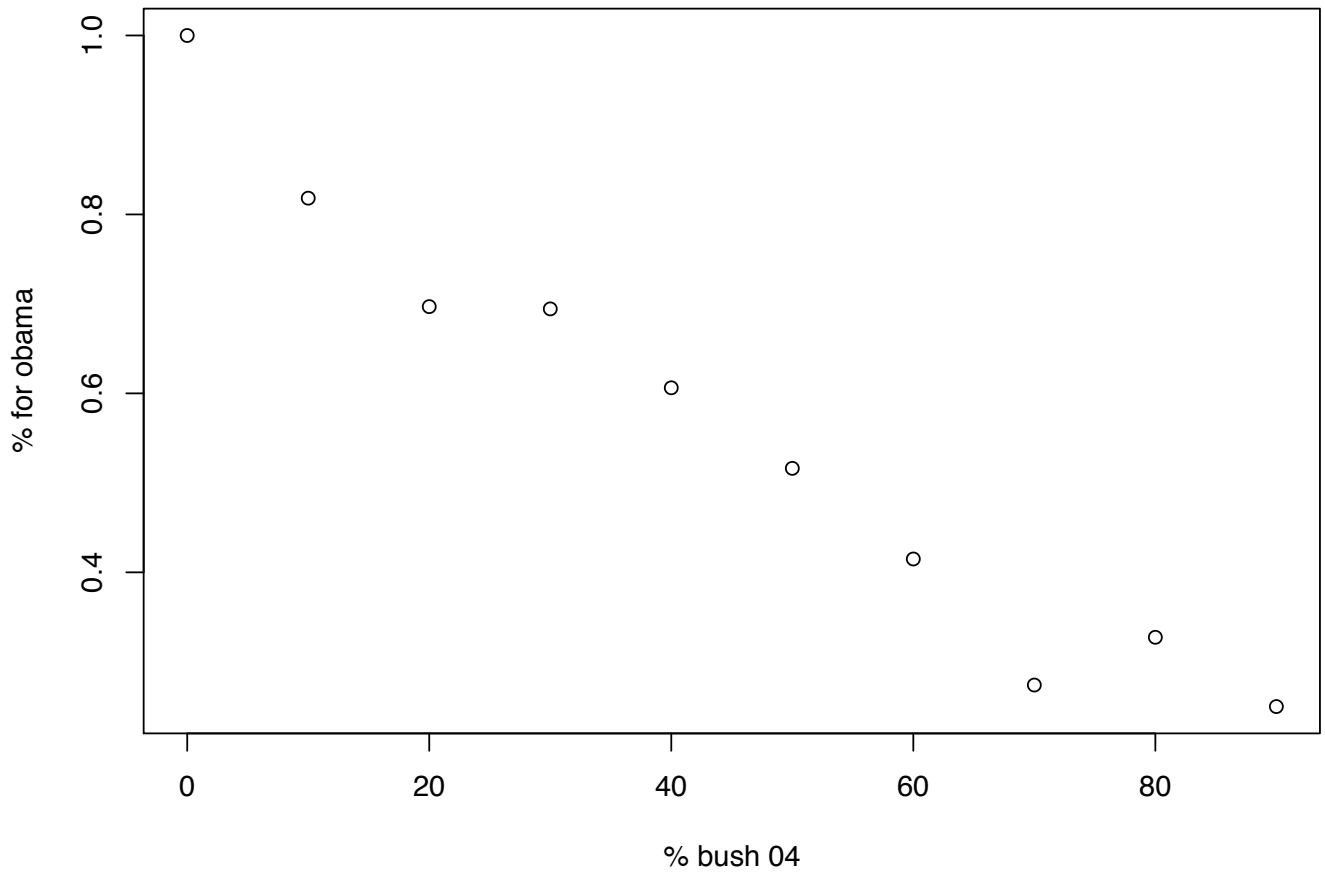
With that single modeling decision, we can now have a better look at our data

1. Divide the input data into a series of equally-sized bins
2. Within each bin, compute the proportion of successes
3. Plot the proportions against the midpoints of the bins

```
# experiments in binning

out = table(trunc(primary$black06pct*10),primary$binwinner)
out

plot(10*(0:8),out[,2]/(out[,1]+out[,2]),xlab="% black",ylab="% for obama")

out = table(trunc(primary$Bush04*10),primary$binwinner)
out

plot(10*(0:9),out[,2]/(out[,1]+out[,2]),xlab="% bush 04",ylab="% for obama")

# try increasing the number of bins...

out = table(trunc(primary$Bush04*20),primary$binwinner)
out

plot(5*as.numeric(rownames(out)),
   out[,2]/(out[,1]+out[,2]),xlab="% bush 04",ylab="% for obama")
```


% for obama
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0
20
40
60
80
% black


% for obama
1.0
0.8
0.6
0.4
20
40
60
80
% bush 04


% for obama
1.0
0.8
0.6
0.4
0
20
40
60
80
% bush 04

Binning

If we assume that p(x) is not varying very much within each bin, then the data points we are seeing are each of the form X/n where X has a binomial distribution with n (the number of points in the bin) and p

An obvious idea is to try to model these using a regression equation - -What does that amount to? Do we anticipate any problems here?

Binning

If we believe our story, then in each bin we have accumulated some number of Bernoulli random variables with approximately the same success probability

Let n_i be the number of observations in bin i and let p_i denote the "true" probability of success in that bin; we then estimate p_i with $\widehat{p}_i$, the proportion of successes in bin i

With this notation in place, we know that each $\widehat{p}_i$ has mean p_i and variance $p_i(1 - p_i)/n_i$

Unequal variances

What this means is if we were to repeat our experiment (and admittedly for the Democratic primary that is a little hard to swallow), then some points will vary more than others

Granted, our binning process was a little naive, but the general question of what you do in this case remains; in general, when you have observations with different variances, OLS is no longer BLUE

The “best” approach involves a weighted regression in which the weights are inversely proportional to the variances of each observation; this allows us to discount noisy observations in favor of the more accurate ones

In our case, however, the variances $p_i(1 - p_i)/n_i$ depend on the unknown success probabilities; now what?

Variance-stabilizing transformations

As one last ditch effort to save least squares in this context, we might invoke some very old (and very pretty) magic

The idea is to construct a transformation that does not depend on any unknown parameters; in this case, we'd like to remove the dependence on

So, can we find a transformation g such that the distribution of $g(\widehat{p}_i)$ does not depend on p_i (at least approximately)?

Variance-stabilizing transformations

Despite it's somewhat horrible appearance, the quantity

$$g(\widehat{p}_i) = \sin^{-1}\sqrt{\widehat{p}}$$

has an approximately normal distribution (for large enough n) with mean $\sin^{-1}\sqrt{p}$ and variance 1/4n

The ugly form comes from the fact that finding this transformation involves an integral of the form

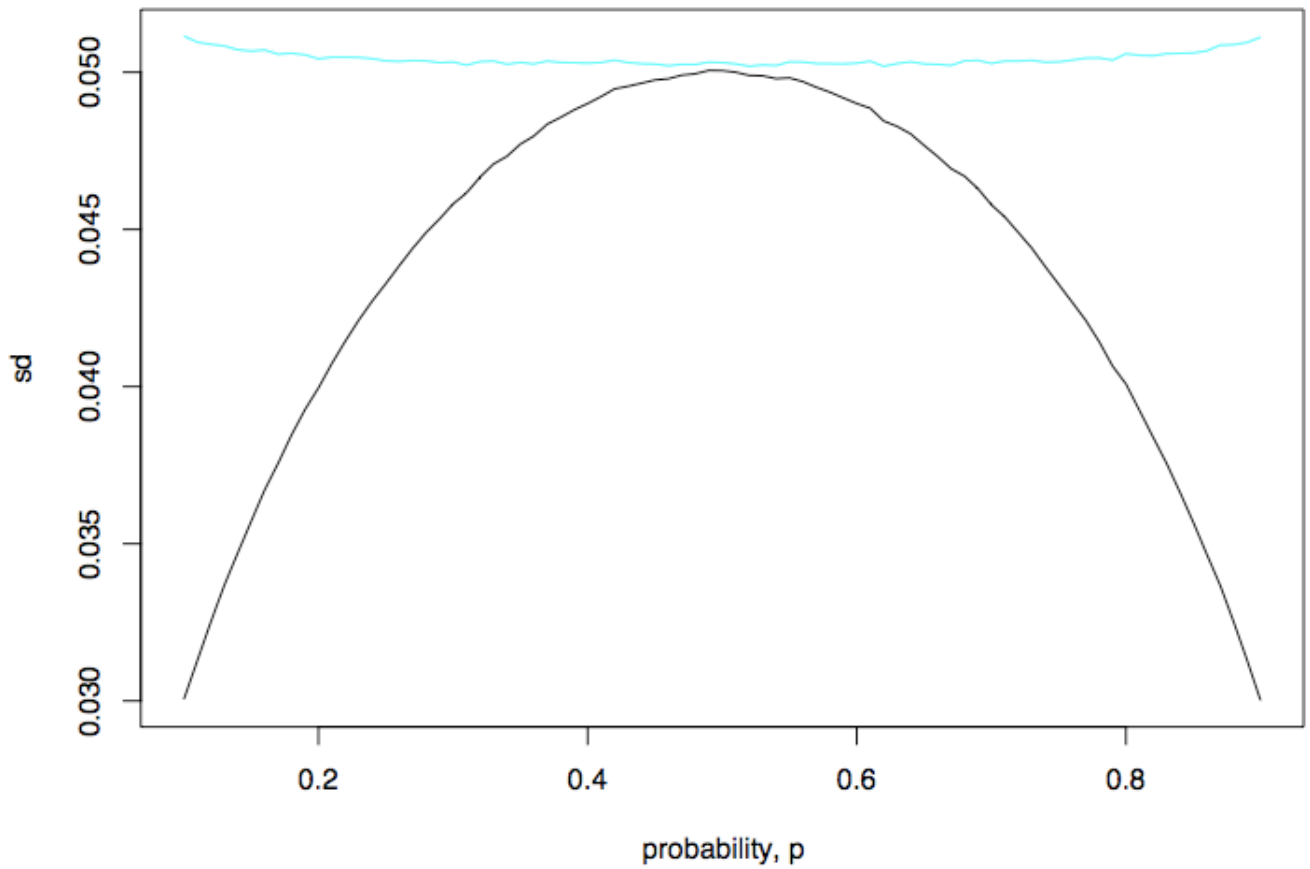
$$g(x) \propto \int \frac{dt}{\sigma(t)} = \int \frac{dt}{\sqrt{t(1-t)}}$$

which may or may not bring back your worst calculus memories

In action

We can test this out through simulation (no surprise there); on the next page we have 500 simulations of the binomial distribution with n=100 and p ranging from 0.1 to 0.9

As you can see, the transformation is performing as advertised; the standard deviation of the transformed proportions (cyan) $g(\hat{p}_i) = \sin^{-1}\sqrt{\hat{p}}$ shows very little variation as compared to those for $\hat{p}$

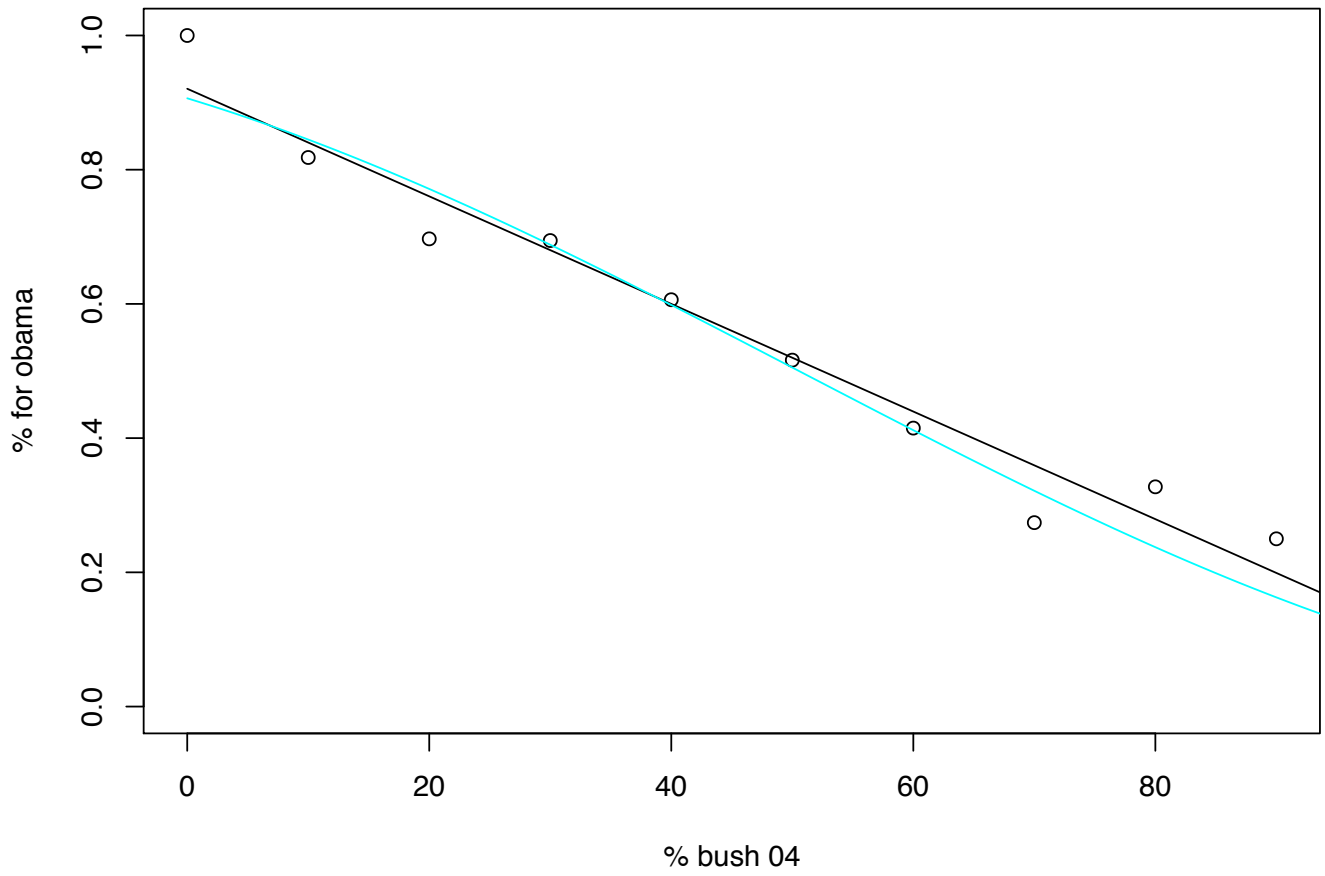

sd
0.030
0.035
0.040
0.045
0.050
0.2
0.4
0.6
0.8
probability, p

Back to the primaries...

This means that one fairly direct approach to working with binary outcomes, then, might start with some binning followed by a variance stabilizing transformation and a weighted regression; we then have to back-transform to return the predictions to the right scale

We do this on the next couple pages...

```
out = table(trunc(primary$Bush04*10),primary$binwinner)

smallp = data.frame(bins=10*as.numeric(rownames(out)),
                    phat=out[,2]/(out[,1]+out[,2]),
                    n=out[,1]+out[,2])

plot(smallp$bins,smallp$phat,
     xlab="% bush 04",ylab="% for obama",ylim=c(0,1))

fit =lm(phat~bins,data=smallp)
newsmallp = data.frame(bins=seq(0,100,len=100))
lines(newsmallp$bins,predict(fit,newdata=newsmallp))

fit2 = lm(asin(sqrt(phat))~bins,weights=n,data=smallp)
preds = sin(predict(fit2,newdata=newsmallp))^2
lines(newsmallp$bins,preds,col=5)
```


% for obama
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
% bush 04

The Democratic primary

The binning idea has some obvious problems as a methodology for working with binary outcomes

First off, binning becomes harder the more variables you consider; eventually you end up with a lot of empty bins

Next, there's a tradeoff at work in binning: We are treating the unknown probability p(x) as being constant within each bin; wide bins mean the arcsine-squareroot transform is more reliable, but wide bins also mean a coarser approximation to p(x)

```
table(primary$winner)

#     0    1
#  1210 1031

tbl = table(primary2$winner,primary2$Bush04,dnn=c("winner","bush"))

tbl

#              bush
#  winner     0    1
#       0   171 1039
#       1   302  728
```

Starting over

Let's start over, this time working a bit more closely with the underlying probabilities

In our data set, 1031/2241 = 0.46 counties were won by Obama (keep in mind that he is ahead in both the popular vote and in the number of states won)

We define the odds of Obama winning a county to be

$$\text{odds} = \frac{p}{1-p} = \frac{0.46}{0.54} = 0.85$$

Odds

Now let's consider separately the odds for those counties that George Bush won in 2004

Among those counties that Bush won in 2004, the probability of an Obama win is 728/(728+1039) = 0.41; the associated odds are

$$\text{odds} = \frac{p}{1-p} = \frac{0.41}{0.59} = 0.69$$

Among those counties that Bush lost in 2004, the probability of an Obama win is 302/(171+302) = 0.64; the associated odds are

$$\text{odds} = \frac{p}{1-p} = \frac{0.64}{0.36} = 1.78$$

Odds ratio

To compare the odds associated with two populations, it is common to consider the odds ratio

In our primary example, we compute

$$\text{oddsratio} = \frac{\text{odds}_{\text{l}}}{\text{odds}_{\text{w}}} = \frac{1.78}{0.69} = 2.58$$

In other words, the odds of Obama winning a county is over 2 and a half times greater if Bush lost that county in 2004

Odds ratio

Essentially, an odds ratio quantifies whether one group or experimental condition has higher or lower odds for achieving some binary outcome

In terms of regression terminology...

A number greater than one means positive association between the “independent” variable and the “dependent” variable (Obama winning in 2008 versus Bush in 2004)

A number between zero and one means negative association between independent and dependent variables

If the odds ratio equals one, the independent variable carries no information about the dependent variable

Log-odds

While odds are fairly direct quantities, it is often desirable to work with something a little more symmetric

$$\text{log-odds} = \log \frac{p}{1-p}$$

This transformation is also known as the logit; unlike the original probabilities, the logit can take any value between minus infinity and infinity

In the same spirit, it is often common to work with the log-odds ratio; assume we have probabilities of an event under two conditions p_0 and p_1, then

$$\text{log-odds ratio} = \log \frac{\frac{p_0}{1-p_0}}{\frac{p_1}{1-p_1}} = \text{logit}\, p_0 - \text{logit}\, p_1$$

Following our noses

Suppose we consider the following simple model for the probability of whether or not Obama wins a county

$$\text{logit}\, \text{p} = \beta_0 + \beta_1\, \text{bush}$$

where we take `bush` to be zero if Bush lost in 2004 and one otherwise; now we can relate this model to our log-odds ratio

$$\begin{aligned} \text{log-odds ratio} &= \text{logit}\, \text{p}_1 - \text{logit}\, \text{p}_0 \\ &= (\beta_0 + \beta_1 * 1) - (\beta_0 + \beta_1 * 0) \\ &= \beta_1 \end{aligned}$$

Logistic regression

And before you know it, we've arrived; if we choose to express the dependence of p on our covariate bush, p = p(bush), through the model

$$\text{logit}\, \text{p} = \beta_0 + \beta_1\, \text{bush}$$

then we can interpret the coefficient β_1 as a log-odds ratio, measuring the effect of a Bush win in 2004 on the odds that Obama will win the same county in 2008

To sum up, we started by considering the odds and ultimately odds ratios and were led to their "logged" counterparts which gave rise to the logistic model

The interpretation of the coefficients in a logistic model comes directly from the log-odds ratios and is a natural scale for working with probabilities

Logistic regression

There is nothing stopping us from extending our simple model, from the “factor” involving Bush’s win to other independent variables; say, from

$$\text{logit p} = \beta_0 + \beta_1 \text{ bush}$$

to

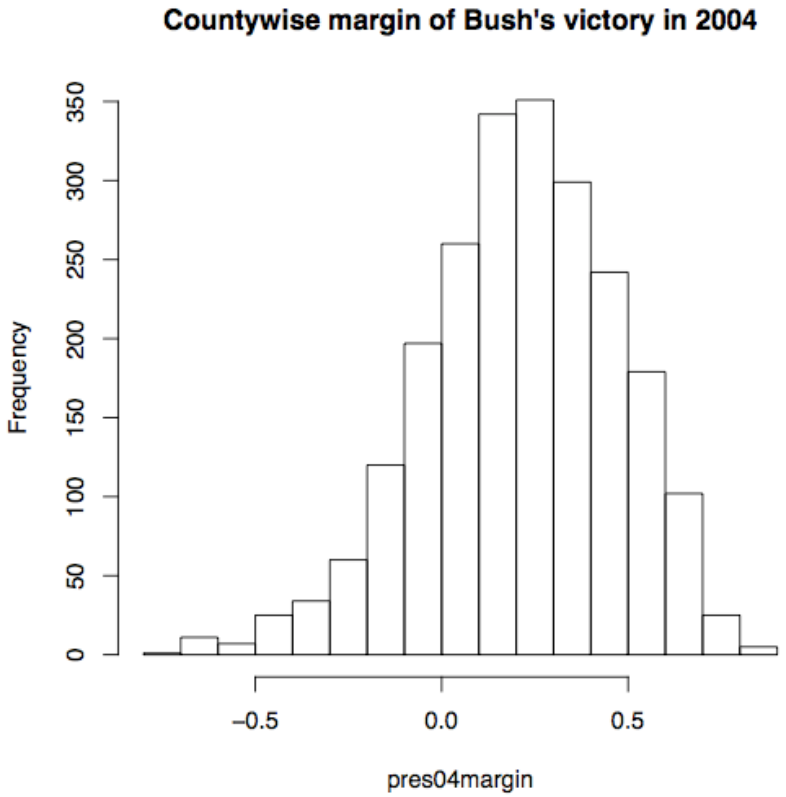
$$\text{logit p} = \beta_0 + \beta_1 \text{ bush} + \beta_2 \text{ pct\_hs\_grad}$$

and so on...

Logistic regression

The variable `bush` was derived from the (continuous) variable pres04margin which holds the margin by which Bush won each county in 2004

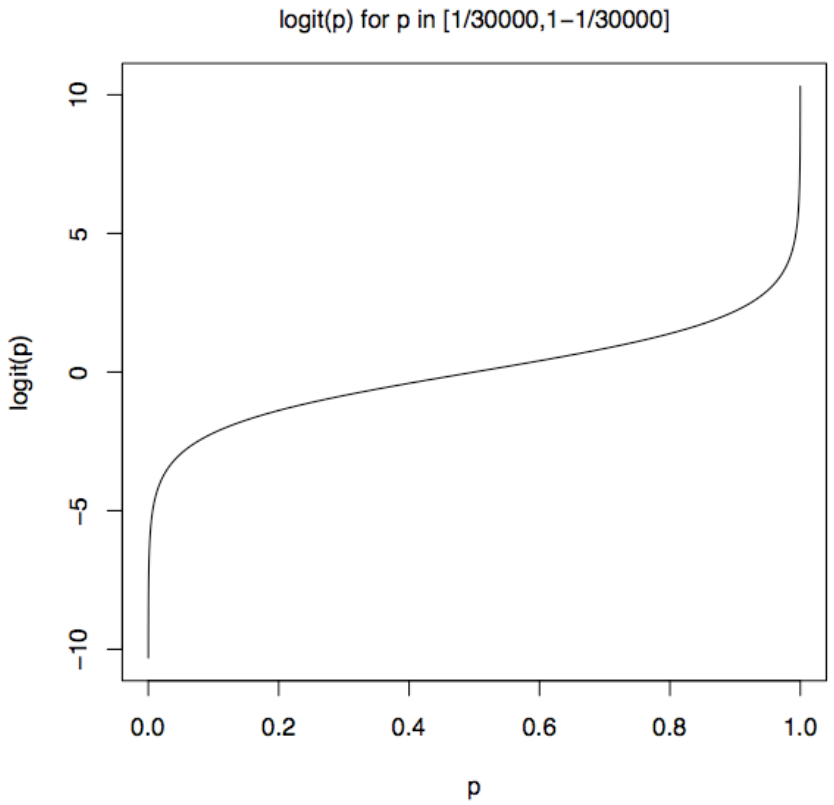
Let's now consider what our logit model implies for this continuous covariate; to get there, we need to know a little more about the logit



Logistic regression

At the right we plot the logit(p) as a function of p for a wide range of values; notice that the function has asymptotes at 0 and 1 (tending to $-\infty$ and ∞, respectively)

It should be clear from this plot that logit(p) is an invertible function...



The logit transform

... that is, for any value $v \in (-\infty, \infty)$, the p such that logit(p) = v can be found by first writing

$$\text{logit}\, p = \log \frac{p}{1-p} = v$$

and then taking the logarithm of both sides we have p/(1-p) = exp(v) or p = (1-p) exp(v); then, collecting terms p(1+exp v) = exp v, or

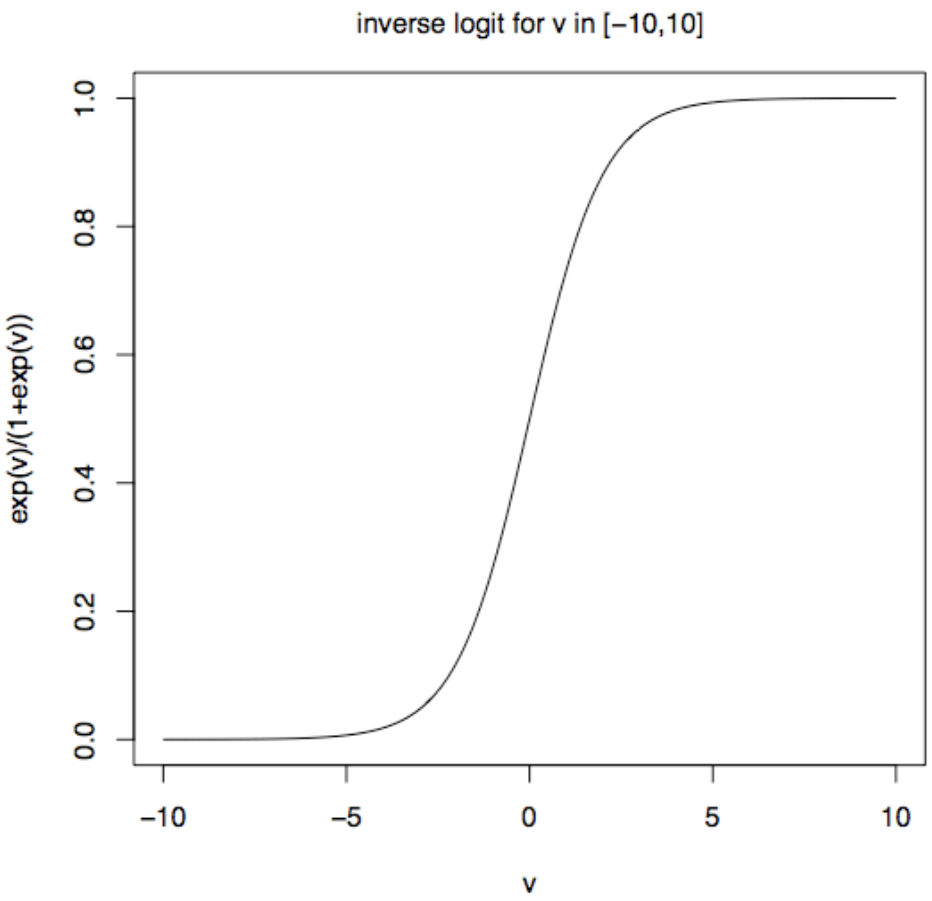
$$p = \frac{\exp v}{1 + \exp v}$$

The logit transform

At the right we plot the inverse of the logit function; for very small (negative) values, it tends to zero, while for large values it tends to 1

A function that looks like this is often called a sigmoid function in some branches of mathematics and computer science; we might also recognize it as another kind of function... any guesses?

Now, let's apply this to our "regression" context with a continuous covariate...



The logit transform

With these results in mind, suppose we choose to replace our model

$$\text{logit}\, \text{p}(\text{bush}) = \log \frac{\text{p}(\text{bush})}{1 - \text{p}(\text{bush})} = \beta_0 + \beta_1\, \text{bush}$$

where bush was a binary covariate, with the expression

$$\text{logit}\, \text{p}(\text{pres04margin}) = \beta_0 + \beta_1\, \text{pres04margin}$$

where pres04margin is now continuous; for any values of the coefficients and the covariate, we invert the logit to give us

$$\text{p}(\text{pres04margin}) = \frac{\exp(\beta_0 + \beta_1\, \text{pres04margin})}{1 + \exp(\beta_0 + \beta_1\, \text{pres04margin})}$$

The link

As an aside, in the context of logistic regression, we refer to the logit as a “link” function; it is the link that takes us from the scale of the covariates into the scale of our quantity of interest p

This idea of a link wasn’t really necessary when we studied the normal linear model; the nature of the variation in our data ($y = f(x) + e$) made it sensible to directly model

By comparison, when we used OLS to try to estimate the dependence of p on covariates, the lack of a link meant that our estimates were not constrained to be between 0 and 1; we’ll see this again in a few slides

Logistic regression

Therefore, we will imagine that in the primary race between Obama and Clinton, the probability that Obama wins a given county is determined by the toss of a coin

The probability that the coin is heads (indicating an Obama win and producing a 1 in our data set) is modeled by

$$\mathsf{p}(\mathsf{pres04margin}) = \frac{\exp(\beta_0 + \beta_1\,\mathsf{pres04margin})}{1 + \exp(\beta_0 + \beta_1\,\mathsf{pres04margin})}$$

Our data set consists of over 2,000 counties, where in each case we have a binary response $\mathsf{y} \in \{0, 1\}$ paired with the value of the covariate, pres04margin, which indicates the margin of Bush's victory in the county in 2004

We then form estimates $\widehat{\beta}_0$ and $\widehat{\beta}_1$, chosen that the resulting fit "captures the patterns" of wins and losses we see in our data; this is vague, but for now it's enough to say that there is an underlying measure that $\widehat{\beta}_0$ and $\widehat{\beta}_1$ are chosen to minimize

Logistic regression

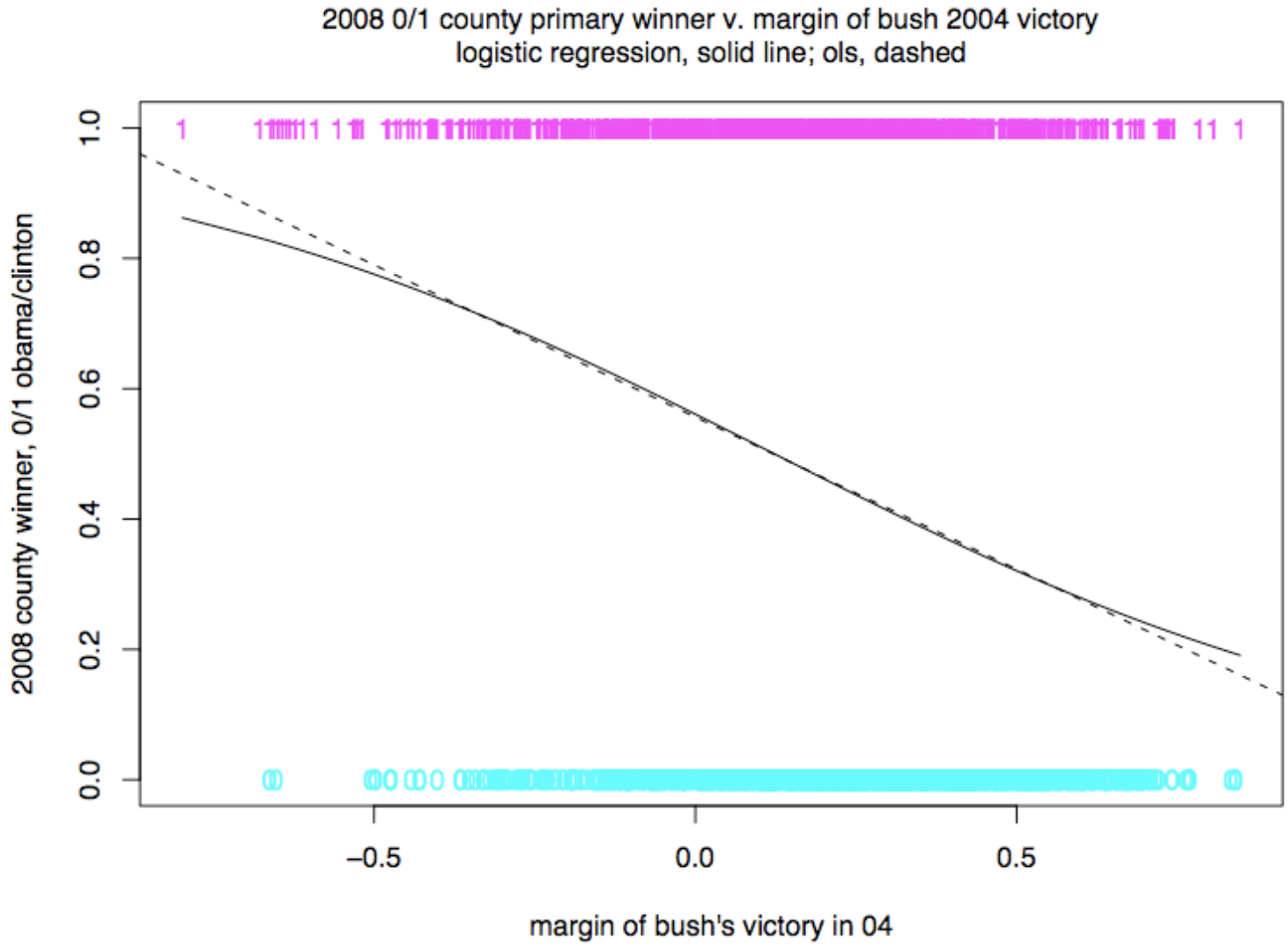
On the next page we present a plot of the 2008 county election results (1 if Obama won, 0 if Clinton won) against the margin for Bush in 04

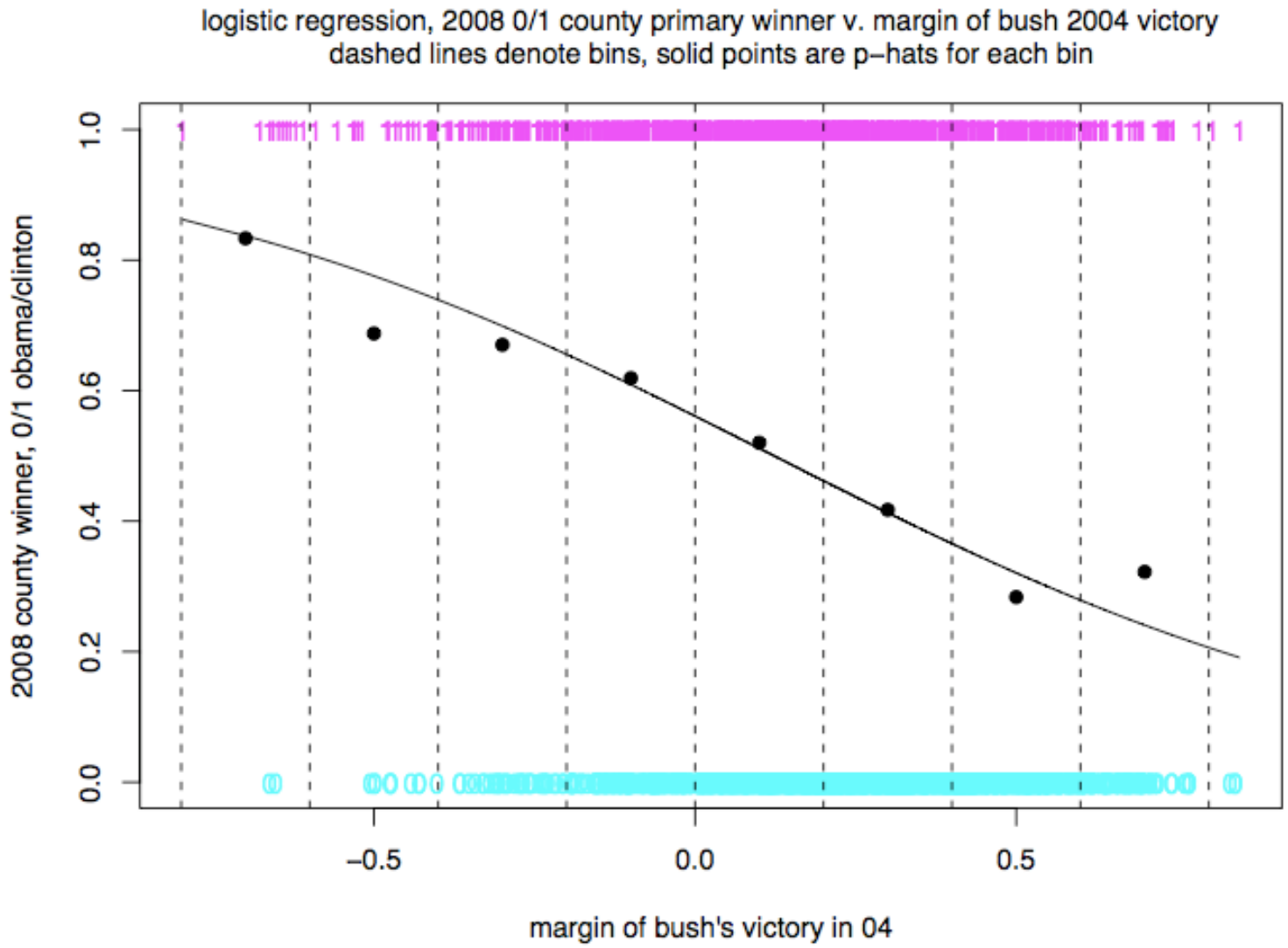
The solid curve in the middle is the result of "fitting" the relationship

$$\text{logit}\, \text{p}(\text{pres04margin}) = \beta_0 + \beta_1\, \text{pres04margin}$$

based on the county data in primary; the dashed line is the OLS fit to the 0/1 data (this is, in some sense, the limit of the binning we did last time)

What can you say about the fits?


2008 0/1 county primary winner v. margin of bush 2004 victory
logistic regression, solid line; ols, dashed
2008 county winner, 0/1 obama/clinton
0.0
0.2
0.4
0.6
0.8
1.0
−0.5
0.0
0.5
margin of bush's victory in 04


logistic regression, 2008 0/1 county primary winner v. margin of bush 2004 victory
dashed lines denote bins, solid points are p–hats for each bin
2008 county winner, 0/1 obama/clinton
0.0
0.2
0.4
0.6
0.8
1.0
–0.5
0.0
0.5
margin of bush's victory in 04

Logistic regression

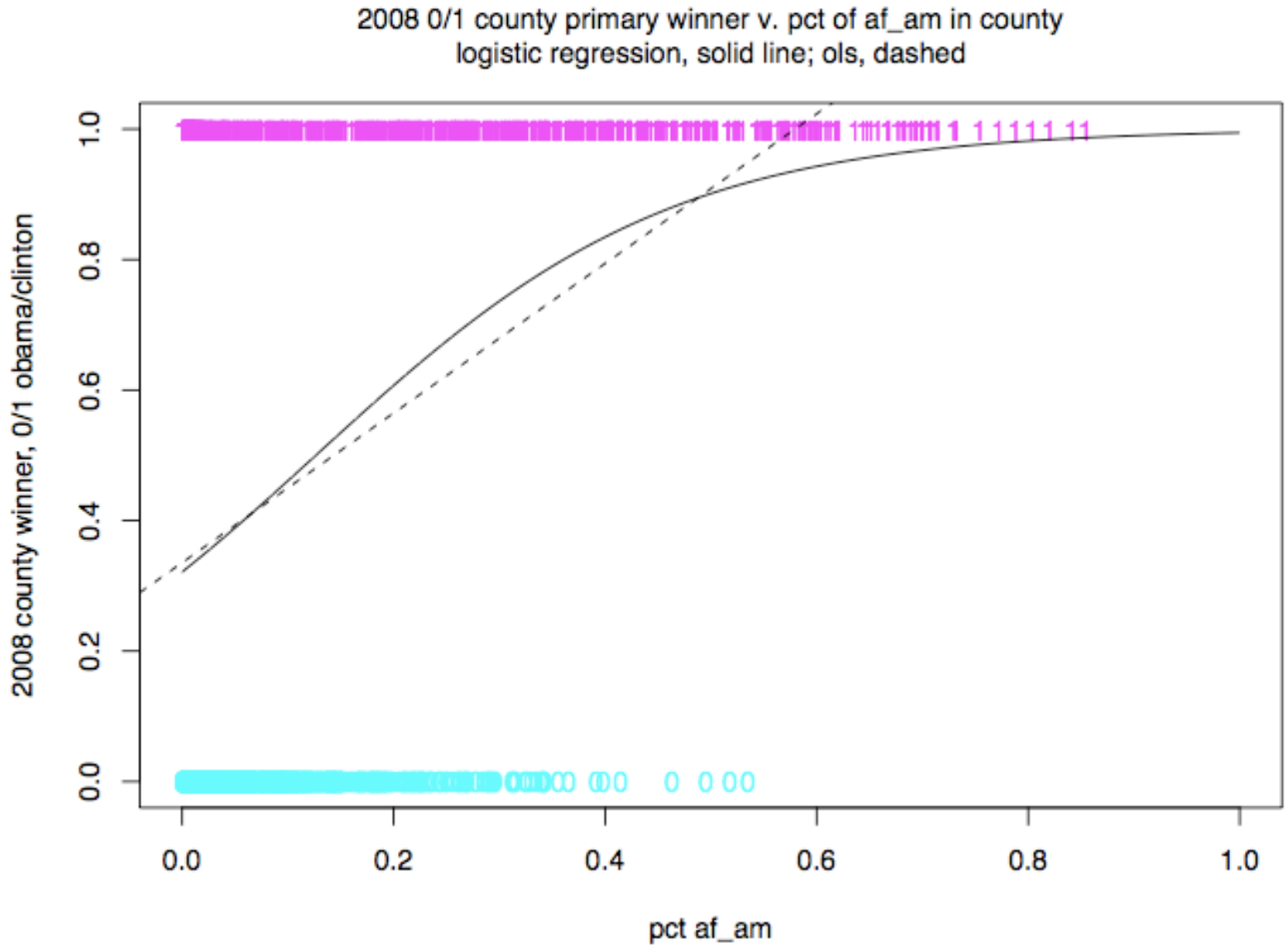
Fitting produces estimates $\widehat{\beta}_0 = 0.24$ and $\widehat{\beta}_1 = -1.99$; the negative coefficient on pres04margin means that if we compare a margin of x to a margin of x + 0.1, the odds of an Obama win drop by a factor of

$$\exp(\widehat{\beta}_1 * 0.1) = \exp(-0.2) = 0.8$$

Perhaps not surprisingly, we can compute standard errors for these estimates (if there's one thing we're good at, it's assessing uncertainty); for $\widehat{\beta}_0 = 0.24$, the SE is 0.06 and for $\widehat{\beta}_1 = -1.99$ the SE is 0.18, implying that both effects are statistically significant

On the next page, we present some sample R code, although you don't have enough detail about what's going on to really grasp all of it yet; for the moment, you are meant to see how similar things are to the normal linear model

```
> source(url("http://www.stat.ucla.edu/~cocteau/primary2.R"))
> fit <- glm(winner~pres04margin,family="binomial",data=primary2)
> summary(fit)


  Call:
  glm(formula = winner ~ pres04margin, family = "binomial", data = primary2)

  Deviance Residuals:
      Min        1Q    Median        3Q       Max
  -1.8733  -1.0670  -0.8118    1.1712    1.8196

  Coefficients:
               Estimate Std. Error z value Pr(>|z|)
  (Intercept)   0.24494    0.05678   4.314 1.60e-05 ***
  pres04margin -1.99133    0.17918 -11.113  < 2e-16 ***
  ---
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

  (Dispersion parameter for binomial family taken to be 1)

      Null deviance: 3090.8  on 2239  degrees of freedom
  Residual deviance: 2955.8  on 2238  degrees of freedom
    (21 observations deleted due to missingness)
  AIC: 2959.8

  Number of Fisher Scoring iterations: 4
```


2008 0/1 county primary winner v. pct of af_am in county
logistic regression, solid line; ols, dashed
2008 county winner, 0/1 obama/clinton
0.0
0.2
0.4
0.6
0.8
1.0
pct af_am

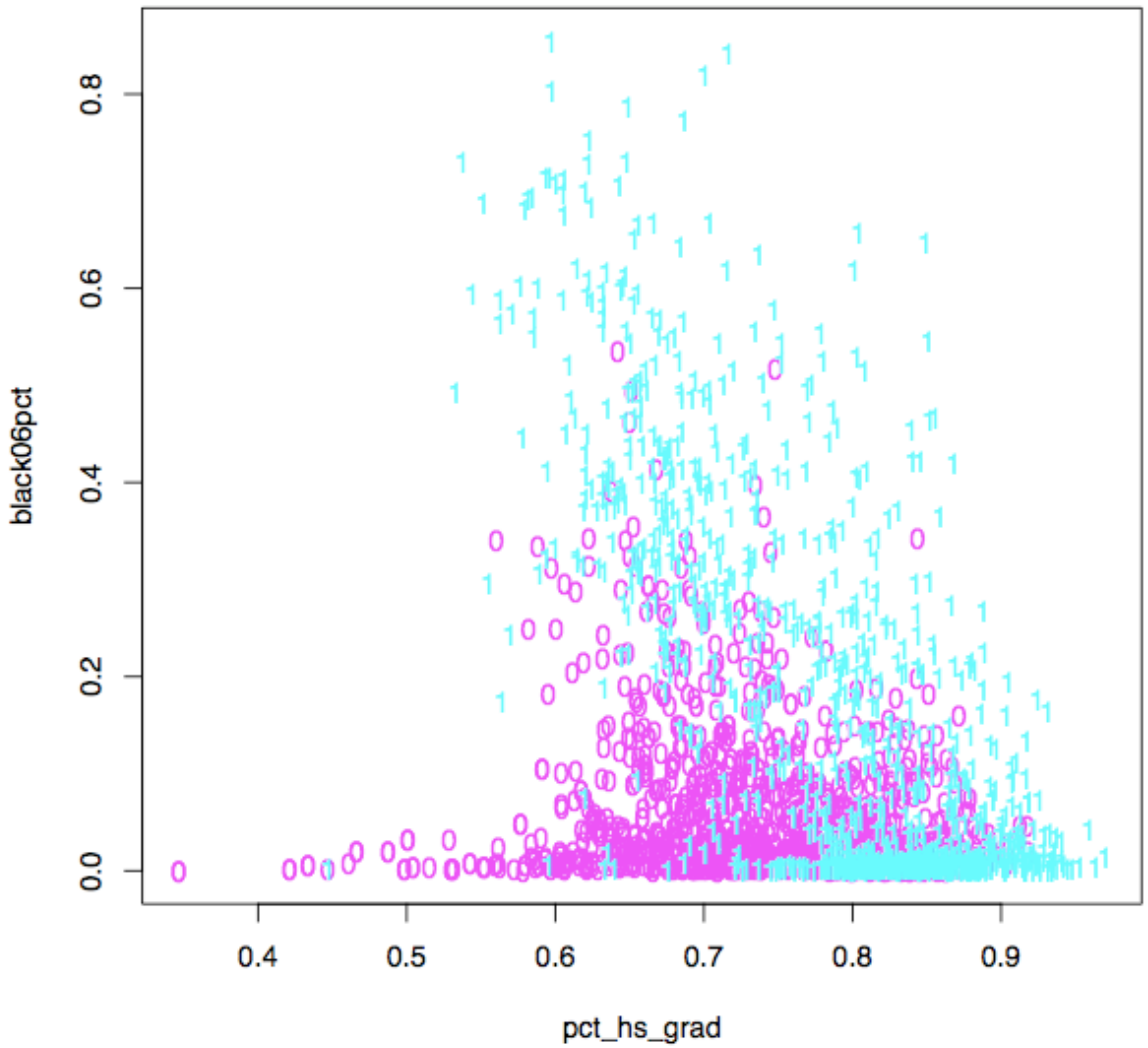
Logistic regression

Suppose we want to consider two variables simultaneously (and once we have seen two, there's nothing stopping us from getting into a lot of trouble)

We will consider the last two single-variable fits we performed; namely a model in which

$$\text{logit p} = \beta_0 + \beta_1 \text{ pct\_hs\_grad} + \beta_2 \text{ black06pct}$$

In the next slide, we plot the data with symbols that indicate who won each county, cyan for Obama and magenta for Clinton (for a very stupid reason, note that we have swapped colors in this plot)

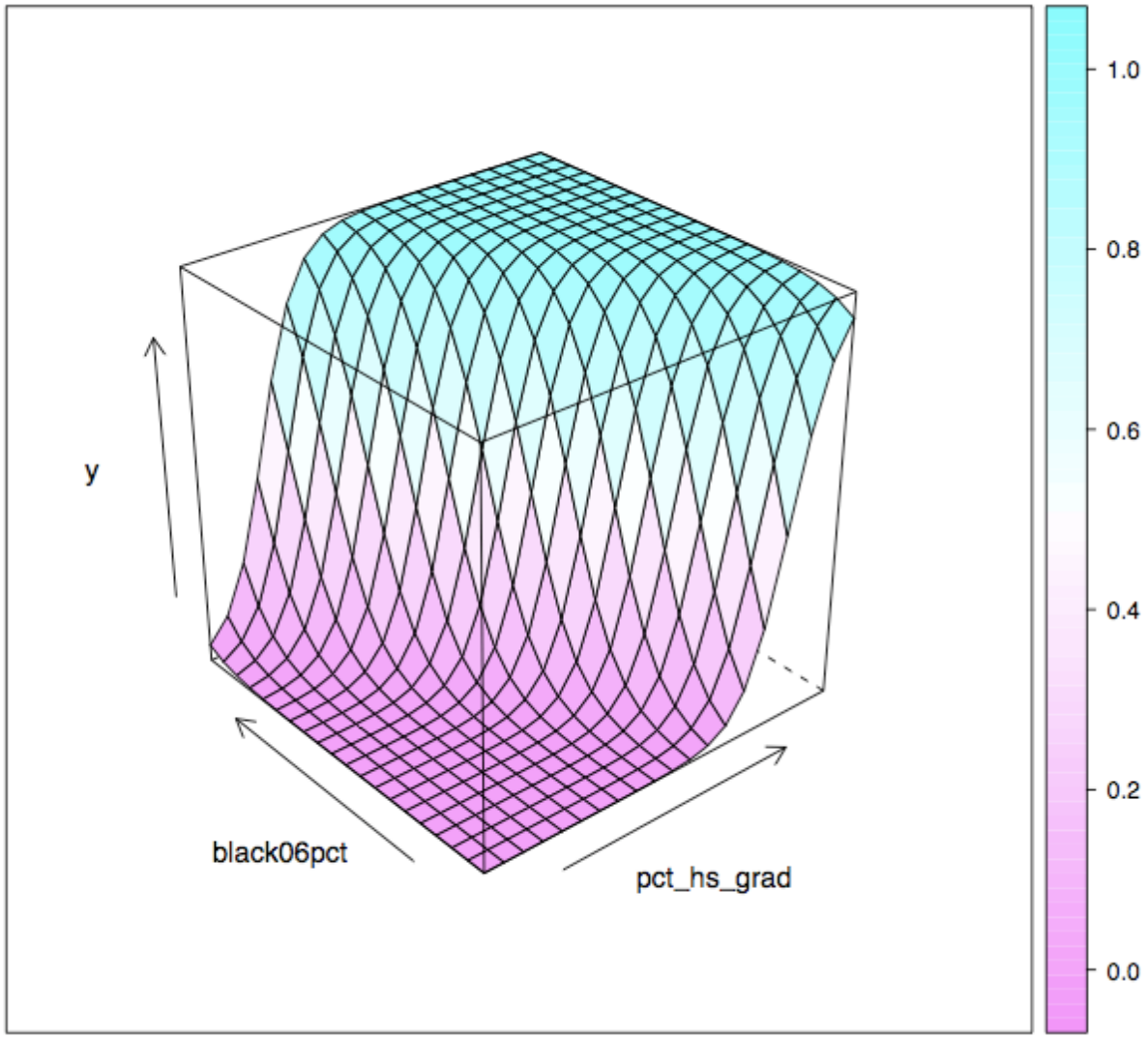

black06pct
0.0
0.2
0.4
0.6
0.8
0.4
0.5
0.6
0.7
0.8
0.9
pct_hs_grad

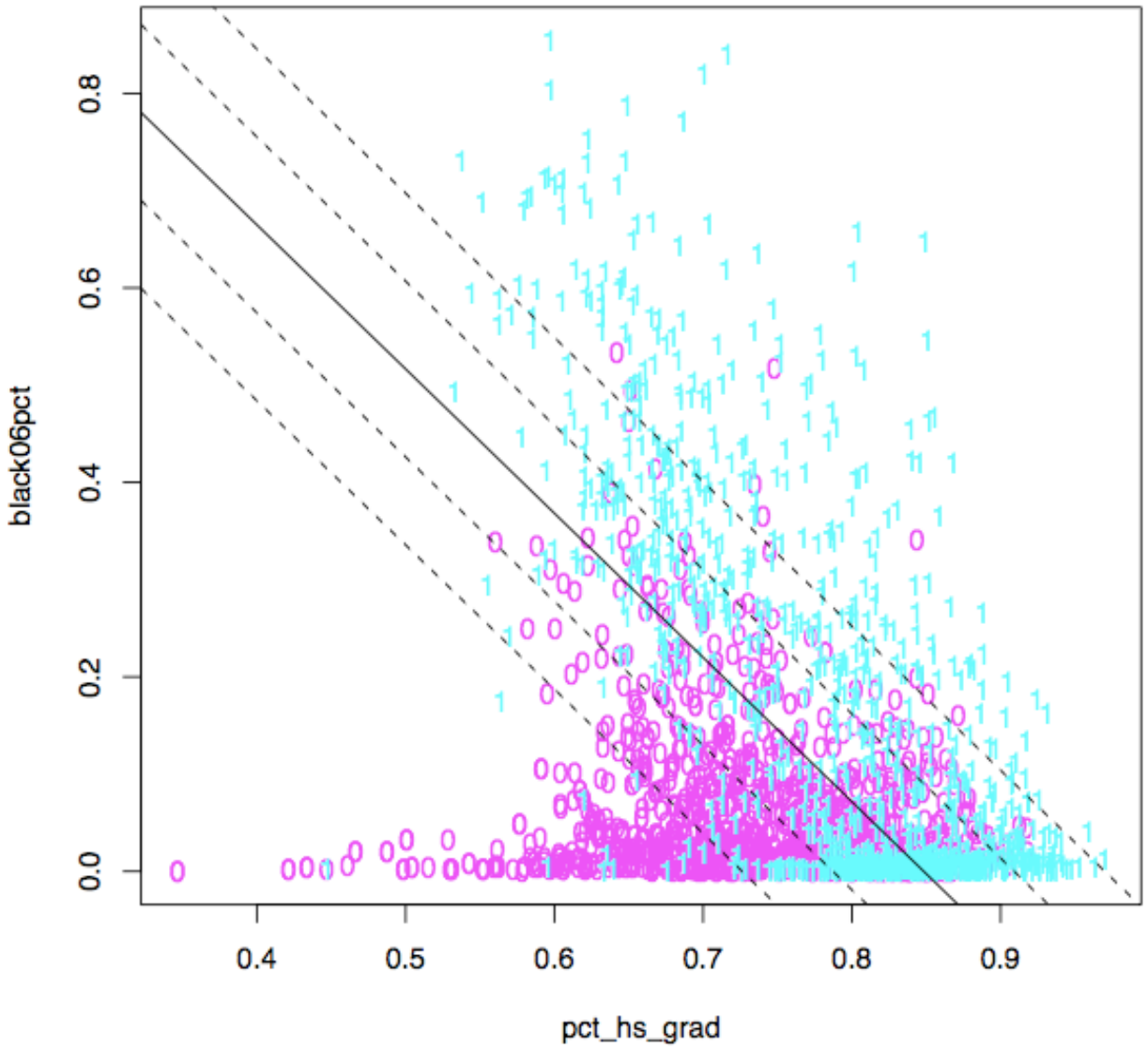
Logistic regression

Given the form of the model, we would expect the fitted probability surface to have contours that correspond to the lines

$$\mathsf{c} = \beta_1 \, \mathsf{pct\_hs\_grad} + \beta_2 \, \mathsf{black06pct}$$

In the next slide we present the fitted surface followed by a repeat of the covariate scatterplot with contour lines corresponding to the 0.1, 0.25, 0.5, 0.75 and 0.9 levels superimposed


y
black06pct
pct_hs_grad
1.0
0.8
0.6
0.4
0.2
0.0


black06pct
0.0
0.2
0.4
0.6
0.8
0.4
0.5
0.6
0.7
0.8
0.9
pct_hs_grad

Logistic regression

To compute the line associated with the 90% chance of an Obama victory, we computed

$$\text{logit}\, 0.9 = \log \frac{0.9}{0.1} \approx 2.2$$

and then rewrote the following to express `pct_hs_grad` as a function of black06pct

$$2.2 = \widehat{\beta}_0 + \widehat{\beta}_1 \,\text{pct\_hs\_grad} + \widehat{\beta}_2 \,\text{black06pct}$$

Interpreting the fit

We began this part of our logistic regression adventure by considering odds and we saw that our fitted coefficients have interpretations as a log-odds ratio

Assuming our model is worth something, we can interpret the signs of the coefficients -- Positive coefficients mean increasing values of the covariates lead to larger odds and in turn larger probabilities

We can say a little more, at least approximately...

```
> source(url("http://www.stat.ucla.edu/~cocteau/primary3.R"))
> fit <- glm(winner~.,family="binomial",data=primary3)
> summary(fit)

Call:
glm(formula = winner ~ ., family = "binomial", data = primary3)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.3568  -0.6639  -0.1736   0.7064   3.7790

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)   -12.03054    2.38004  -5.055 4.31e-07 ***
regionNE       -1.75933    0.24636  -7.141 9.25e-13 ***
regionS        -0.69357    0.20587  -3.369 0.000755 ***
regionW         1.51874    0.22424   6.773 1.26e-11 ***
pres04margin   -1.22034    0.41304  -2.955 0.003131 **
pct_less_30k   -7.22899    1.20294  -6.009 1.86e-09 ***
pct_more_100k  -8.21992    2.18772  -3.757 0.000172 ***
pct_hs_grad    14.11779    1.49400   9.450  < 2e-16 ***
pct_homeowner   1.65847    0.94561   1.754 0.079455 .
black06pct     17.52698    1.71814  10.201  < 2e-16 ***
hisp06pct       3.03879    1.71333   1.774 0.076126 .
white06pct      2.50931    1.63798   1.532 0.125533
Bush04         -0.05407    0.21547  -0.251 0.801853
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3089.3  on 2238  degrees of freedom
Residual deviance: 1931.2  on 2226  degrees of freedom
AIC: 1957.2

Number of Fisher Scoring iterations: 5
```

Divide-by-four

To help interpret the coefficients of a logistic regression, suppose we have a simple linear model (on the logit-scale)

$$p(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

The point of greatest change occurs where $\beta_0 + \beta_1 x = 0$, or at p(x) = 0.5; it turns out that the slope of the curve is greatest here as well and that its maximum is

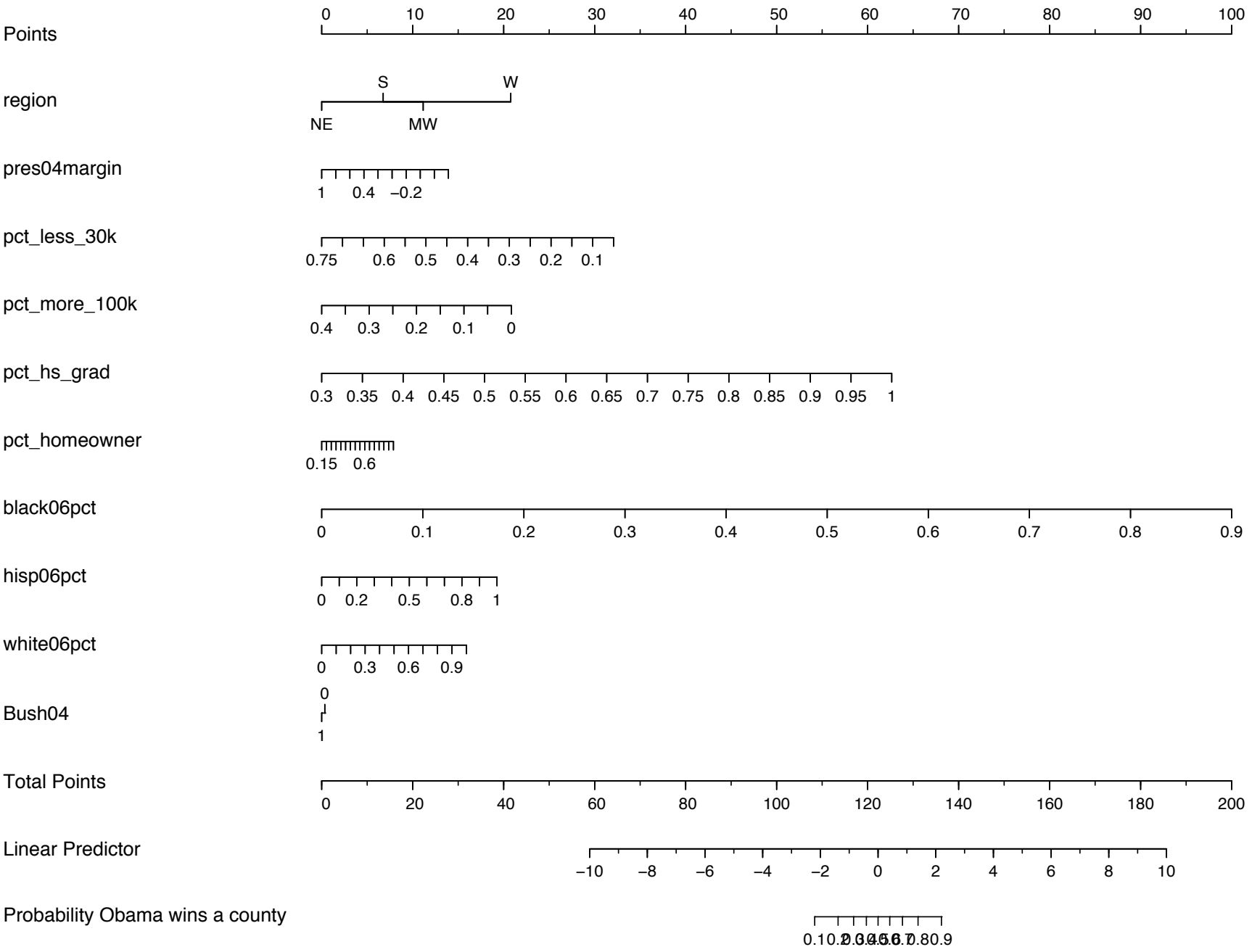
$$\frac{\beta_1 e^0}{(1 + e^0)^2} = \frac{\beta_1}{4}$$

Therefore, $\beta_1/4$ gives you an upper bound to the change in probability you'll see after a 1-unit increase in the predictor -- The advantage here is that log-odds, for all the "naturalness" I've tried to convince you of, can be hard for people to interpret

Nomograms

Inferring the action of a model from a table of regression coefficients can be a pain; what's worse, it's not really the size of the coefficients that matters, but instead the coefficient multiplied by the associated covariate

In short, the "effect" of a covariate is more than just the coefficient; a **nomogram** is a graphical way to assess these effects; the term nomogram refers to a "graphical calculating device"


Points
0 10 20 30 40 50 60 70 80 90 100
region
S W
NE MW
pres04margin
1 0.4 −0.2
pct_less_30k
0.75 0.6 0.5 0.4 0.3 0.2 0.1
pct_more_100k
0.4 0.3 0.2 0.1 0
pct_hs_grad
0.3 0.35 0.4 0.45 0.5 0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95 1
pct_homeowner
0.15 0.6
black06pct
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
hisp06pct
0 0.2 0.5 0.8 1
white06pct
0 0.3 0.6 0.9
Bush04
0
1
Total Points
0 20 40 60 80 100 120 140 160 180 200
Linear Predictor
−10 −8 −6 −4 −2 0 2 4 6 8 10
Probability Obama wins a county

```
# import a cleaned up version of the primary data...

source("http://www.stat.ucla.edu/~cocteau/primary3.R")

# load a new library... this one has A LOT of functionality

library(Design)

# random stuff you have to do to get the nomogram to appear

dist = datadist(primary3)
options(datadist="dist")

# fit a logistic regression model with lrm rather than glm
fit = lrm(winner~.,data=primary3)

# make the nomogram! we have added a line at the bottom that
# is the probability...

invlogit = function(x) exp(x)/(1+exp(x))
nomogram(fit,fun=invlogit,funlabel="Probability Obama wins a county")
```

An alternate path...

While our current data set concerns an observational study, some of the first applications of “regression-like” models for binomial data come from so-called bioassays

In this context, animals are exposed to varying levels of some toxic substance, and researchers want to model the probability that a given dose is lethal

In a dose-response model, the probability $\pi(x)$ that an animal dies after receiving a dose x is described by a tolerance distribution f,

$$\pi(x) = \int_{-\infty}^{x} f(s)ds$$

where $f(x) \geq 0$ and $\int_{-\infty}^{\infty} f(s)ds = 1$

Early applications

For example, if we take f(x) to be the uniform distribution over some interval, say [a,b], then we have

$$f(x) = \begin{cases} 1/(b-a) & a \leq s \leq b \\ 0 & \text{otherwise} \end{cases}$$

and so

$$\pi(x) = \int_a^s f(s)ds = \frac{x-a}{b-a} \quad \text{for} \quad a \leq x \leq b$$

This equation has the form $\pi(x) = \beta_0 + \beta_1 x$ where we need to impose constraints on the coefficients

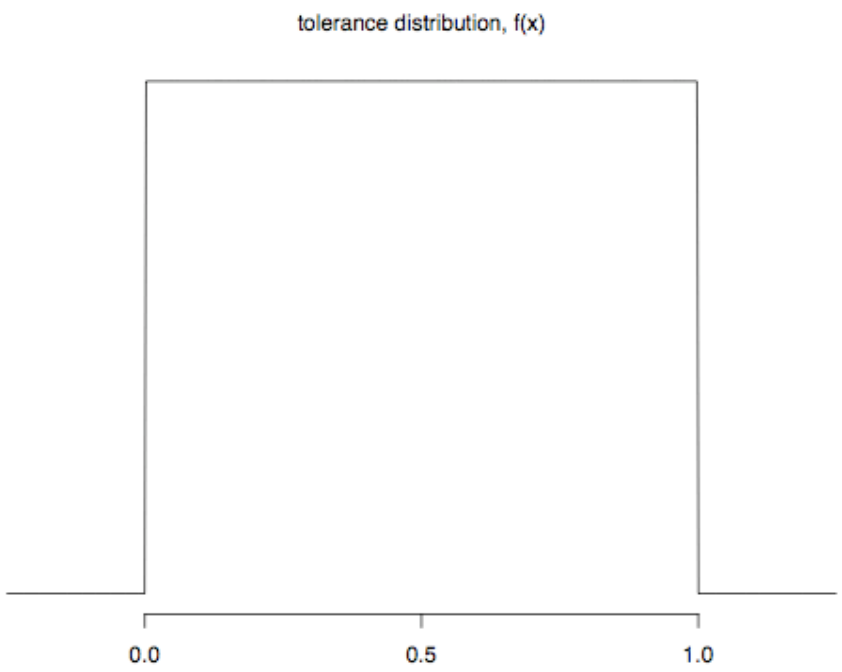
$$\beta_0 = \frac{-a}{b-a} \quad \text{and} \quad \beta_1 = \frac{1}{b-a}$$

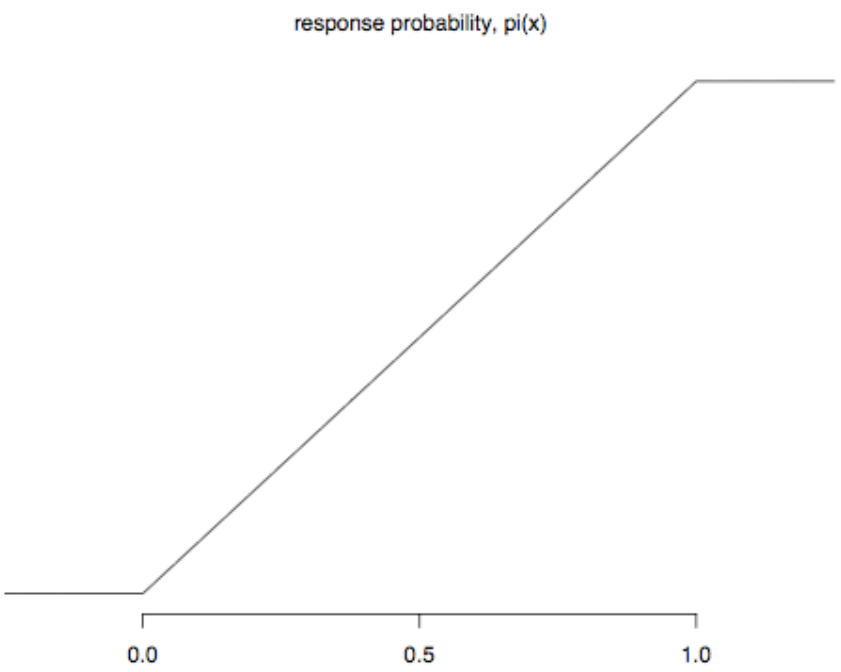
Early applications

This is a kind of linear model, describing the probability of death from a given dose -- Notice that we need to impose constraints on the coefficients to make sure that the resulting probability estimates are between 0 and 1

In terms we've described before, this model uses an identity "link" between the scale of the data (the probability $\pi(x)$) and the covariates, this time just the dose x

Given all the constraints, however, this model is rarely used...


tolerance distribution, f(x)
0.0
0.5
1.0


response probability, pi(x)
0.0
0.5
1.0

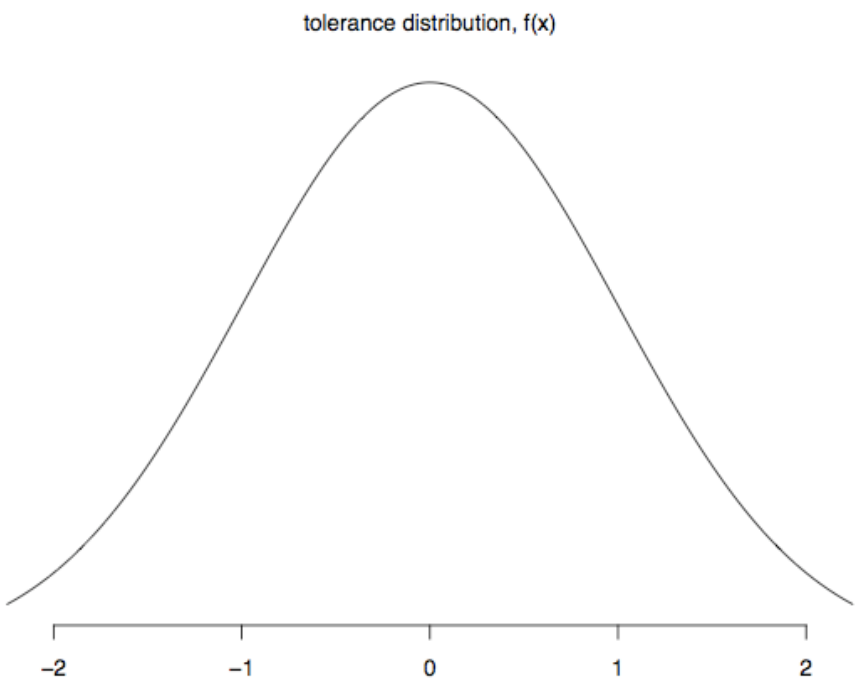
Early applications

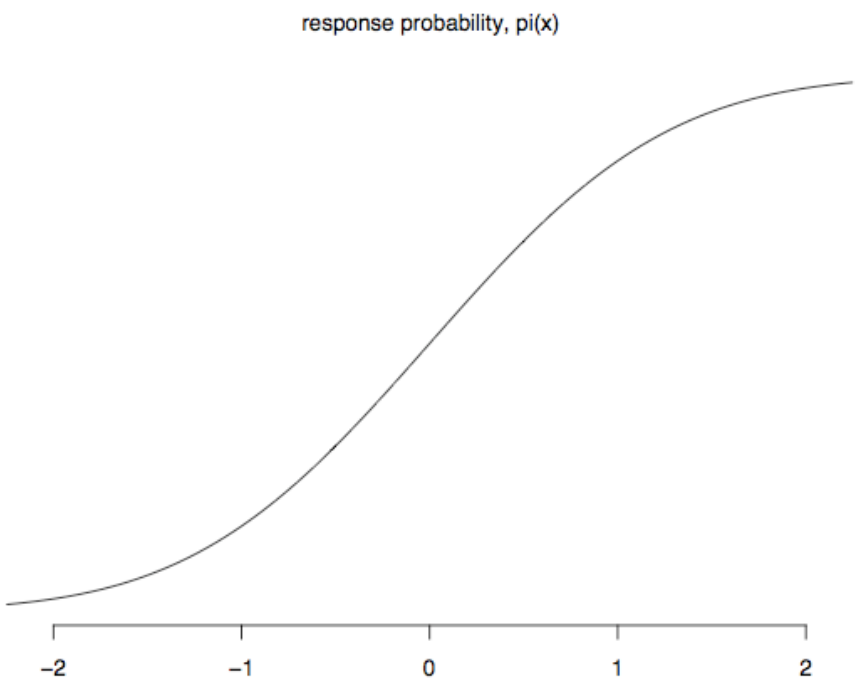
Instead, researchers started to lean on the central limit theorem; that is, if there are a large number of factors that act additively to produce a response, then a normal distribution emerges

Taking f(s) to be a normal distribution with some mean μ and standard deviation σ , we have

$$\begin{aligned}
\pi(x) &= \int_{-\infty}^{x} f(s)ds \\
&= \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(s-\mu)^2/2\sigma^2} ds \\
&= \Phi\left(\frac{x-\mu}{\sigma}\right)
\end{aligned}$$

where Φ denotes the cumulative probability function for the standard normal distribution


tolerance distribution, f(x)
−2
−1
0
1
2


response probability, pi(x)
−2
−1
0
1
2

Early applications

Therefore, since

$$\pi(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$$

we have $\Phi^{-1}(\pi) = \beta_0 + \beta_1 x$ where $\beta_0 = -\mu/\sigma$ and $\beta_1 = 1/\sigma$

Here, the “link” between the data and the covariates is the inverse cumulative normal probability function

In this context, the link function Φ^{-1} is often called the **probit**, for probability unit

Early applications

Probit models are used in both the biological and social sciences; it's attraction is that in some applications the model is natural

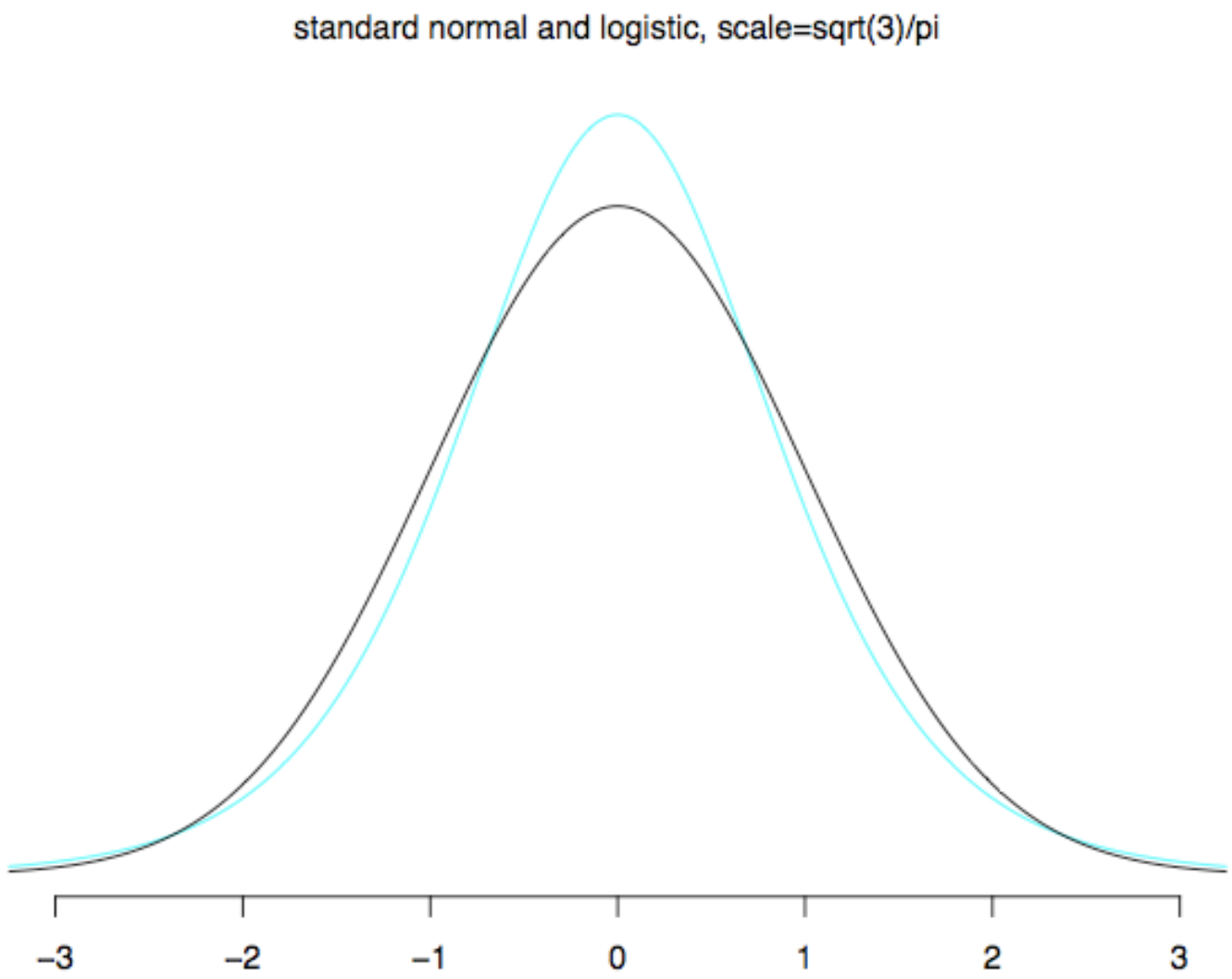
For example, $x = \mu$ is called the median lethal dose or LD50, because it is the dose that can be expected to kill half the animals; LD50 is often used as a general indicator of toxicity

And logistic regression?

We can cast logistic regression in these terms also; rather than the uniform or the normal CDF, logistic regression takes as its tolerance function, well, the logistic distribution

$$f(t) = \frac{e^{-(t-\mu)/s}}{s\left(1 + e^{-(t-\mu)/s}\right)^2}$$

The logistic is also a two-parameter location-scale family (the variance of the standard logistic with mean 0 and scale 1 is $\pi^2/3$)


standard normal and logistic, scale=sqrt(3)/pi
−3
−2
−1
0
1
2
3

Logistic regression

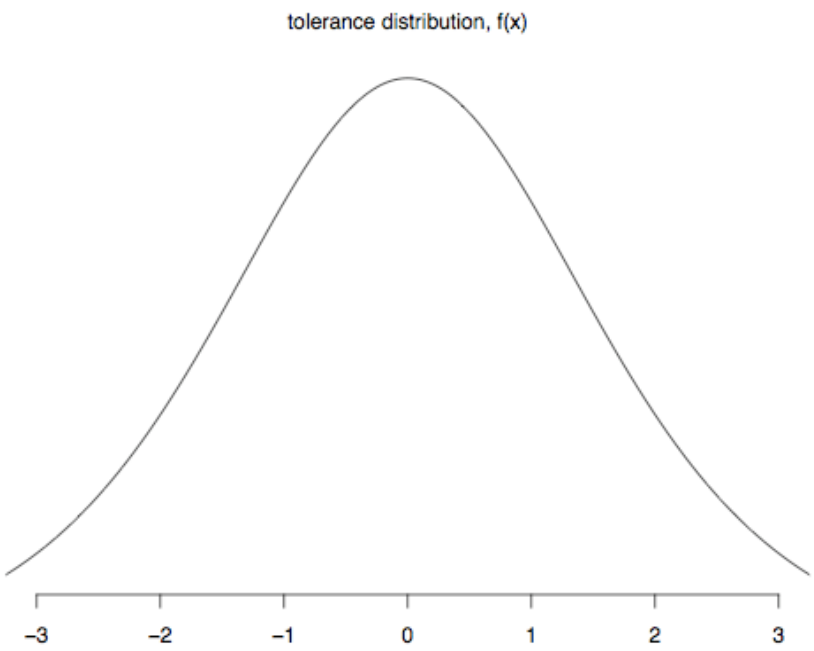
After a little calculus (and not very much, really), we find that the CDF of the logistic distribution is

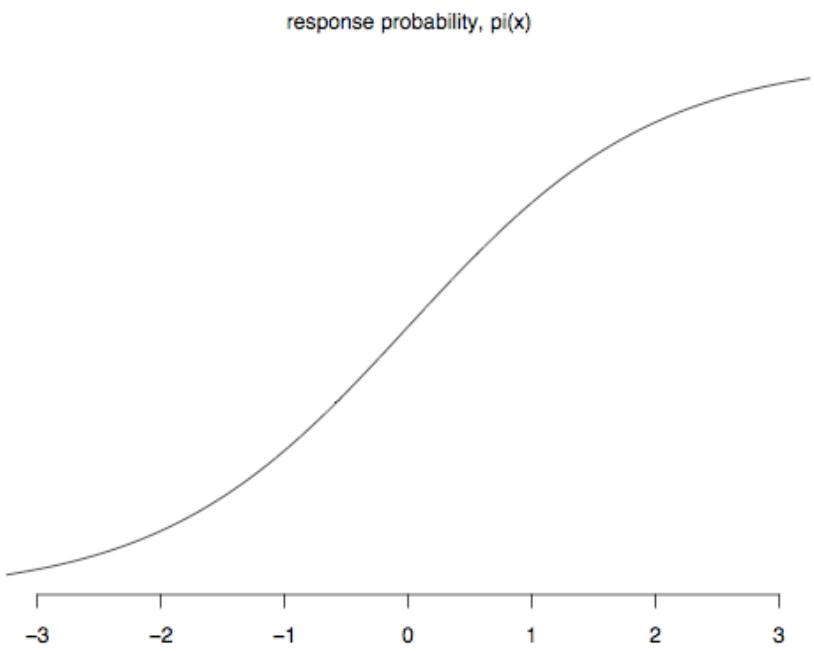
$$\begin{aligned}\int_{-\infty}^{x} f(t)dt &= \int_{-\infty}^{x} \frac{e^{-(t-\mu)/s}}{s\left(1+e^{-(t-\mu)/s}\right)^2}dt \\ &= \frac{1}{1+e^{-(x-\mu)/s}} \\ &= \frac{e^{(x-\mu)/s}}{1+e^{(x-\mu)/s}}\end{aligned}$$

and so if we define $\beta_0 = -\mu/s$ and $\beta_1 = 1/s$, then

$$\pi(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} \quad \text{or} \quad \operatorname{logit}\pi(x) = \beta_0 + \beta_1 x$$

The term **logit** was coined in the 40s, with a specific reference to the probit defined earlier


tolerance distribution, f(x)
−3
−2
−1
0
1
2
3


response probability, pi(x)
−3
−2
−1
0
1
2
3

Three roads to logistic regression

1. We originally approached the topic through odds ratios, trying to make a case that this represents a natural scale for regression-style models that deal in probabilities

2. When we worked out the KL divergence, we found that the logit also emerged quite naturally when we rewrote the Bernoulli probability model (recall we were trying to derive a distance between a saturated model and a proposed linear structure)

3. Finally, today we took a historical view of the subject and followed the development through dose-response models

Roads to logistic regression

We approached the topic through odds ratios, making a case that this represents a natural scale for regression-style models

We took an historical view of the subject following the development of dose-response models

Finally, we will look at KL divergence and see that the logit emerges as a canonical parameter in a Bernoulli distribution

A mechanism

In addition to its pedagogical value, the dose-response framework give us another way to view our binary models

Let's introduce an unobserved (often called "latent") variable that will be used to generate the binary data that we actually observe

$$y_i = \begin{cases} 1 & \text{if } z_i > 0 \\ 0 & \text{if } z_i \leq 0 \end{cases}$$

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

where the ϵ_i are independent and have the logistic distribution

A mechanism

If ϵ has the logistic distribution with mean 0 and scale parameter 1,
then $z = \beta_0 + \beta_1 x + \epsilon$ also has the logistic distribution, but with mean $\beta_0 + \beta_1 x$
and scale parameter 1

Therefore,

$$\text{Prob}\,(y = 1) = \text{Prob}\,(z > 0) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}}$$

using the form of the logistic distribution given earlier

A latent variable approach

We can think of the unobserved or latent variables as describing a county's propensity to vote for Obama or Clinton, say -- By collecting data, all we are able to see really is the sign of this latent variable (positive meaning we see a 1 and negative meaning a 0)

By forming a model, we might be able to see into these propensities a bit more, getting a handle on their magnitude

Logit or probit or ... ?

There's nothing worse than recognizing you have a choice where you didn't think you had one -- Or rather you had been making a choice for no good reason

The choice of link (probit, logit, etc.) is something that comes up in GLMs and there exist formal tests to help you decide the "goodness" of your link (yes, there's a test for that!) if you embed it, say, in a flexible family

For the moment, we'll treat the probit model as a historical motivator and not really discuss the choice but Pregibon (1980) is a great place to look for more information