Making Decisions in High Dimensions

Matt Taddy, Chicago Booth

faculty.chicagobooth.edu/matt.taddy/research

High Dimensional Decisions

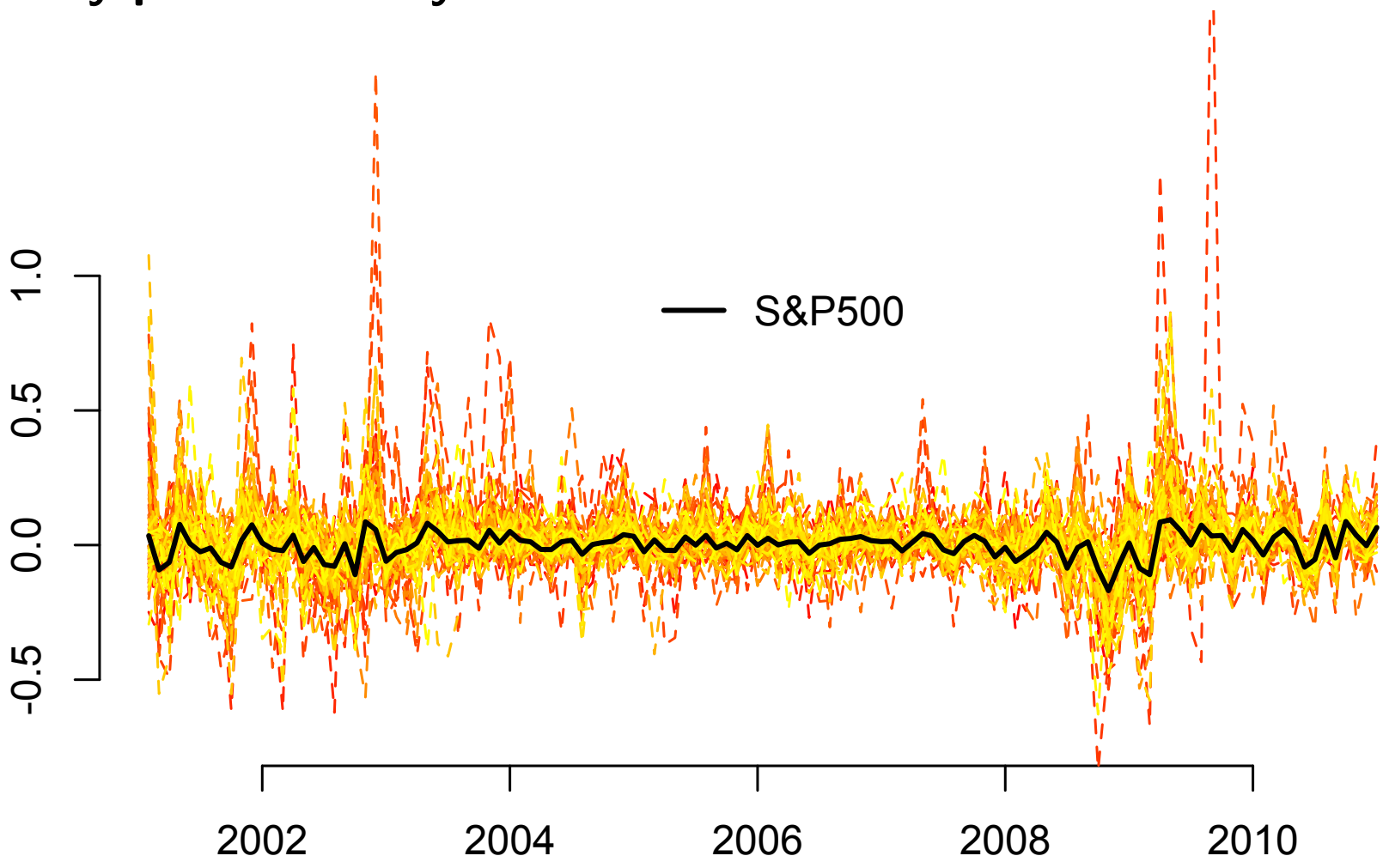
One of the ways that data is 'Big' is in the number of inputs.

This number often grows with the sample size (e.g., words in text, websites in browsers) and you never reach a statistically safe place.

For this type of Big, we need dimension reduction techniques: project from the full input set down to a useful low-D summary.

Crucially, this must be focused on the decisions you'd like to make.

Fancy plot: monthly stock returns



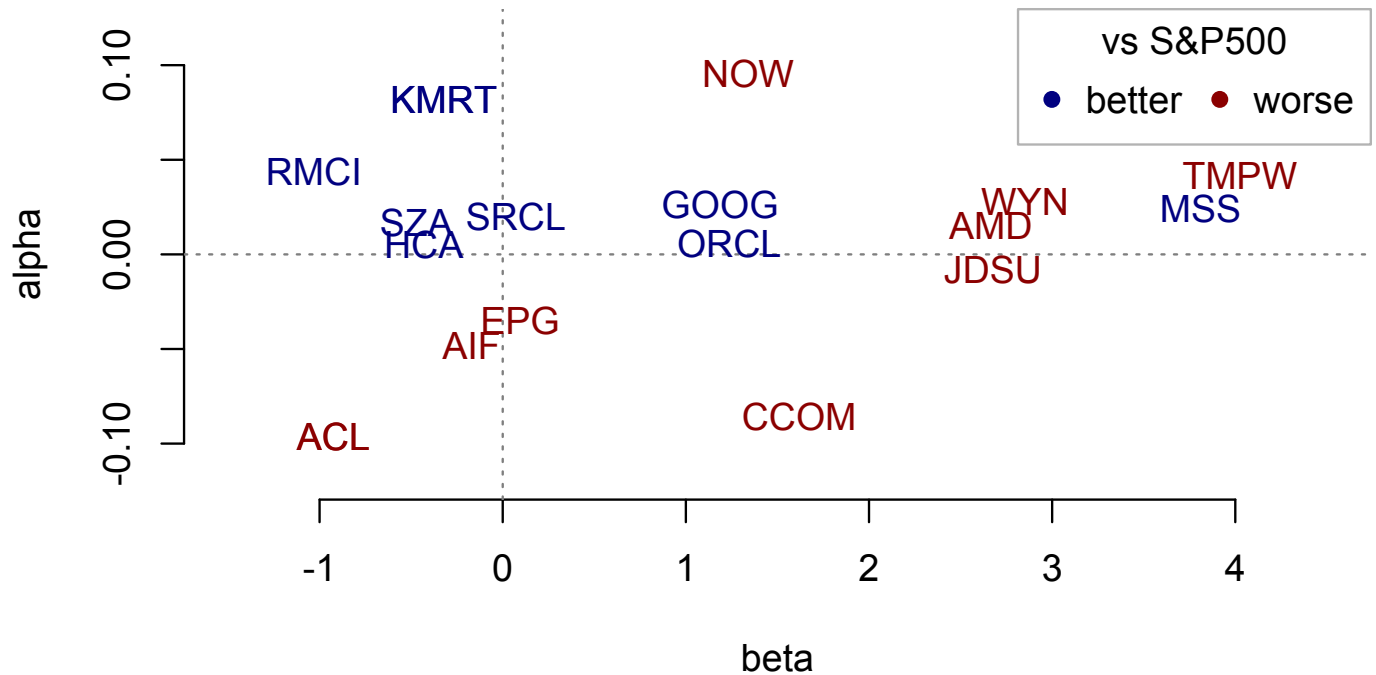
What do we learn?

Useful plot: market model coefficients

Fit a line between stock returns R_t and market returns M_t (SP).

$$R_t \approx \alpha + \beta M_t$$

α is money you make regardless of what the market does.
β is the asset's sensitivity to broad market movements.



Many problems in BD involve a response of interest ('y')
and a set of covariates ('**x**') to be used for prediction.

If your 'decision' is to predict y for new $\mathbf{x}$, we need to reduce dimension in the direction of y. That is, we want to map $\mathbf{x} \rightarrow \hat{y}$.

A general tactic is to deal in averages and lines.
We'll model the conditional mean for y given $\mathbf{x}$,

$$\mathbb{E}[\ y \mid \mathbf{x}\ ] = f(\mathbf{x}'\boldsymbol{\beta})$$

$\mathbf{x} = [1, x_1, x_2, \ldots x_p]$ is your vector of covariates.

$\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, \ldots \beta_p]$ are the corresponding coefficients.

Some basic facts about linear models

The model is always $\mathbb{E}[y|\mathbf{x}] = f(\mathbf{x}\boldsymbol{\beta})$.

- Gaussian (linear): $y \sim \mathrm{N}(\mathbf{x}\boldsymbol{\beta}, \sigma^2)$.
- Binomial (logistic): $\mathrm{p}(y = 1) = e^{\mathbf{x}\boldsymbol{\beta}}/(1 + e^{\mathbf{x}\boldsymbol{\beta}})$.

Likelihood (LHD) is $\mathrm{p}(y_1|\mathbf{x}_1) \times \mathrm{p}(y_2|\mathbf{x}_2) \cdots \times \mathrm{p}(y_n|\mathbf{x}_n)$.

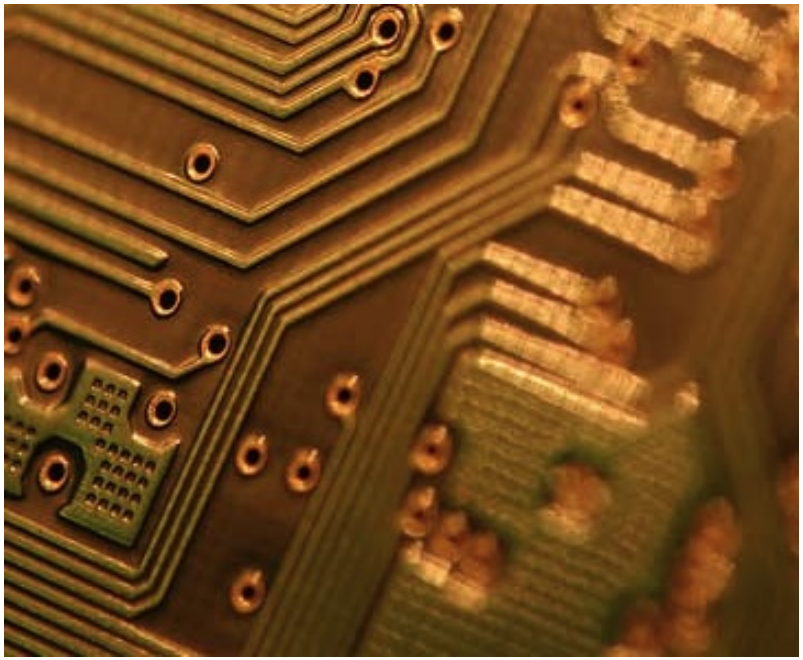
The Deviance (dev) is proportional to -log(LHD).

$\hat{\boldsymbol{\beta}}$ is commonly fit to maximize LHD $\Leftrightarrow$ minimize deviance.

Fit is summarized by $R^2 = 1 - \mathrm{dev}(\hat{\boldsymbol{\beta}})/\mathrm{dev}(\boldsymbol{\beta} = 0)$.

The only R^2 we ever really care about is out-of-sample R^2.

Example: Semiconductor Manufacturing Processes



Very complicated operation
Little margin for error.

Hundreds of diagnostics
Useful or debilitating?

We want to focus reporting and better predict failures.

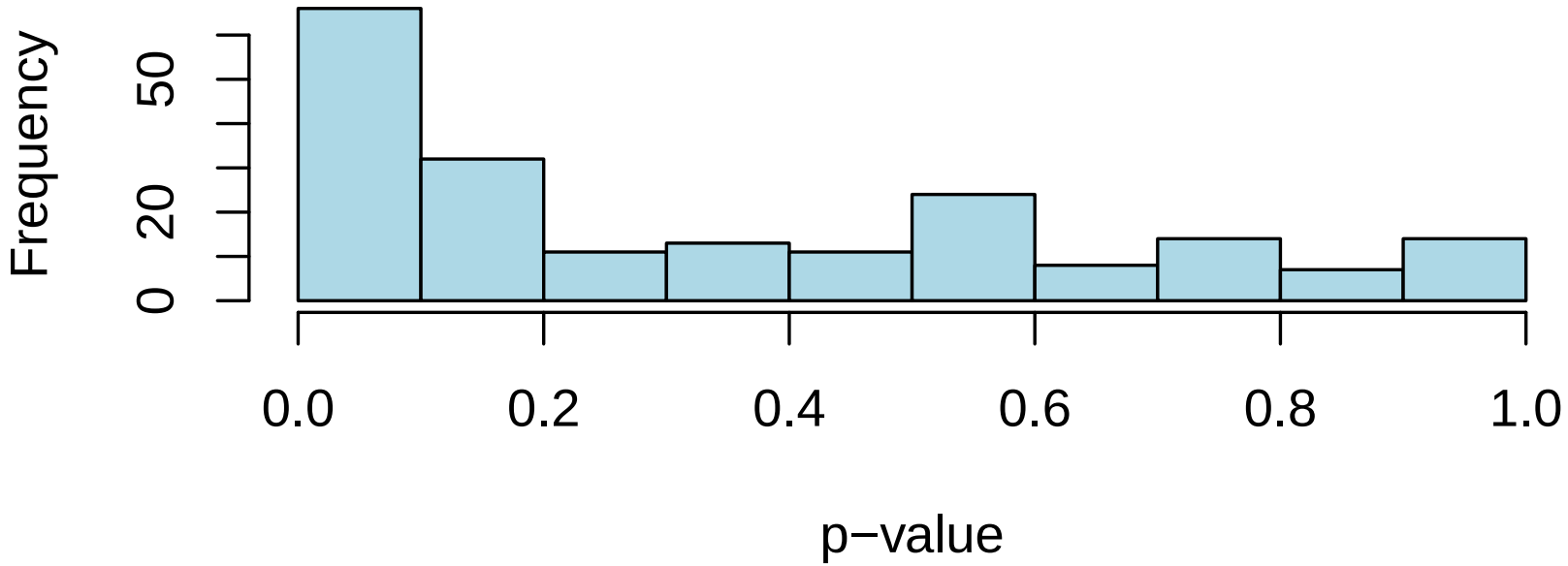
$\mathbf{x}$ is 200 input signals, y has 100/1500 failures.

Logistic regression for failure of chip i is

$$p_i = \mathrm{p}(\mathtt{fail}_i|\mathbf{x}_i) = e^{\alpha+\mathbf{x}_i\boldsymbol{\beta}}/(1+e^{\alpha+\mathbf{x}_i\boldsymbol{\beta}})$$

The full model has $R^2 = 0.56$ (based on *binomial* deviance).

The p-values for these 200 coefficients:



Some are clustered at zero, the rest sprawl out to one.

FDR of $q = 0.1$ yields $\alpha = 0.0122$ p-value rejection cut-off.

Implies 25 'significant', of which approx 22-23 are true signals.

A *cut* model, using only these 25 signals, has $R^2_{cut} = 0.18$.
This is much smaller than the full model's $R^2_{full} = 0.56$.

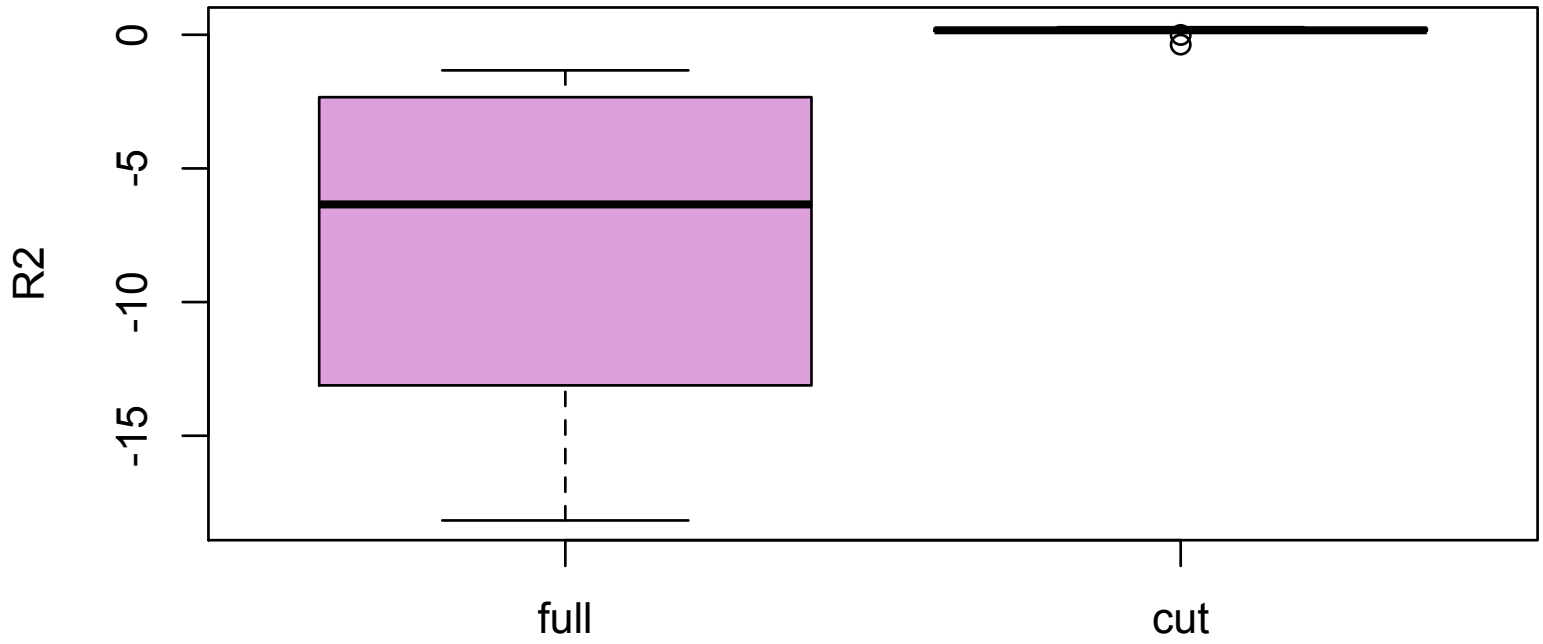
In-Sample (IS) R^2 *always* increases with more covariates.
This is exactly what MLE $\hat{\beta}$ is fit to maximize.
But how well does each model predict *new* data?

An out-of-sample (OOS) experiment

- split the data into 10 random subsets ('folds').
- Do 10x: fit model $\hat{\beta}$ using only 9/10 of data,
  and record R^2 on the left-out subset.

These OOS R^2 give us a sense of how well each
model can predict data that it has not already seen.

We gain predictive accuracy by *dropping* variables.



Cut model has mean OOS R2 of 0.09, about 1/2 in-sample R2.

The full model is terrible. It is overfit and worse than $\bar{y}$.
Negative R2 are more common than you might expect.

All that matters is Out-of-Sample R^2.
We don't care about In Sample R^2.

Using OOS experiments to choose the best model is called *cross validation*. It will be a big part of our big data lives.

Selection of 'the best' model is at the core of all big data.

But before getting to selection, we first need strategies to build good sets of candidate models to choose amongst.

Regularization

The key to contemporary statistics is regularization:

depart from optimality to stabilize a system.

Common in engineering: I wouldn't drive on an optimal bridge.

We minimize deviance

$$\min -\frac{2}{n}\log \mathrm{LHD}(\boldsymbol{\beta})$$

Regularization

The key to contemporary statistics is regularization:

depart from optimality to stabilize a system.

Common in engineering: I wouldn't drive on an optimal bridge.

We minimize deviance plus a cost on the size of coefficients.

$$\min -\frac{2}{n}\log \mathrm{LHD}(\boldsymbol{\beta}) + \lambda \sum_k |\beta_k|$$

This particular cost gives the 'lasso': the new least squares.

Decision theory: Cost in Estimation

Decision theory is based on the idea that choices have costs.
Estimation and hypothesis testing: what are the costs?

Estimation:

Deviance is the cost of distance between data and the model.
Recall: $\sum_i (y_i - \hat{y}_i)^2$ or $-\sum_i y_i \log(\hat{p}_i) - (1 - y_i) \log(1 - \hat{p}_i)$.

Testing:

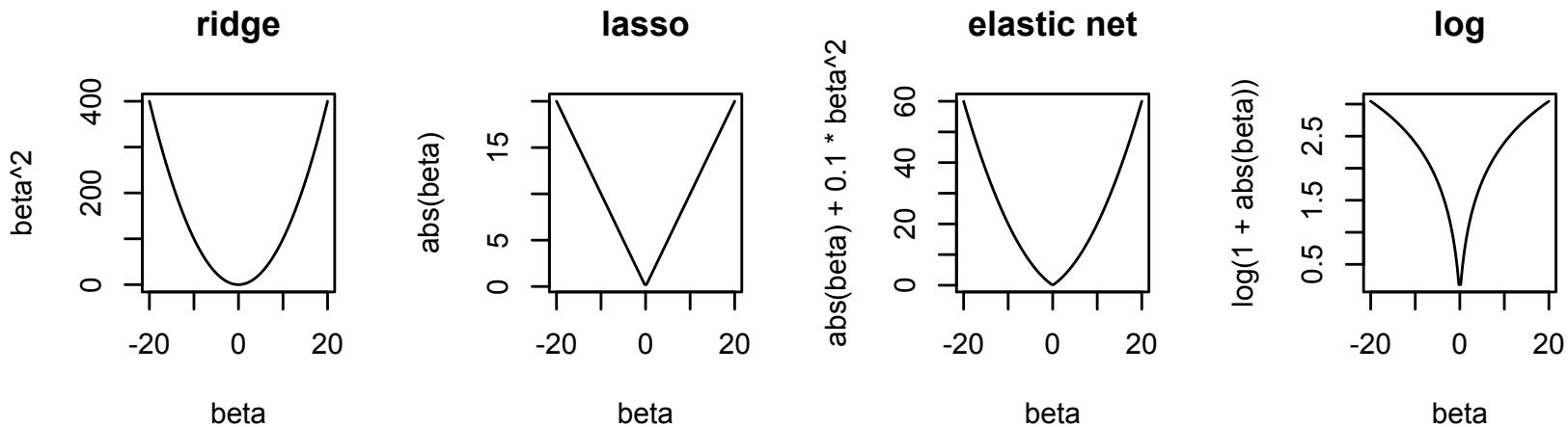
Since $\hat{\beta}_j = 0$ is *safe*, it should cost us to decide otherwise.

$\Rightarrow$ The cost of $\hat{\beta}$ is deviance plus a penalty away from zero.

[Sparse] Regularized Regression

$$\min \left\{ -\frac{2}{n} \log \mathrm{LHD}(\boldsymbol{\beta}) + \lambda \sum_j c(\beta_j) \right\}$$

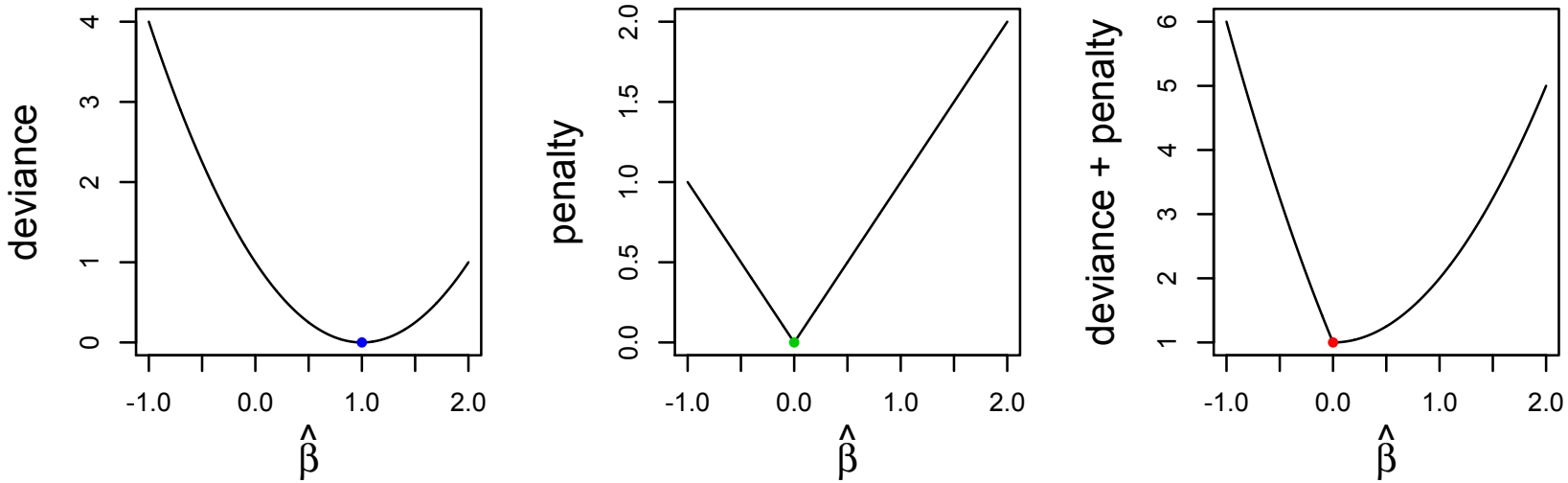
$\lambda > 0$ is the penalty weight, c is a cost (penalty) function.
$c(\beta)$ will be lowest at $\beta = 0$ and we pay more for $|\beta| > 0$.



Options: ridge β^2, lasso $|\beta|$, elastic net $\alpha\beta^2 + |\beta|$, $\log(1 + |\beta|)$.

Penalization can yield automatic variable selection

The minimum of a smooth + pointy function can be at the point.



Anything with an absolute value (e.g., lasso) will do this.

There are MANY penalty options and far too much theory.

Think of lasso as a baseline, and others as variations on it.

Lasso Regularization Paths

The lasso fits $\hat{\boldsymbol{\beta}}$ to minimize $-\frac{2}{n}\log \text{LHD}(\boldsymbol{\beta}) + \lambda \sum_j |\beta_j|$.

We'll do this for a *sequence* of penalties $\lambda_1 > \lambda_2 ... > \lambda_T$.

Then we can apply model selection tools to choose best $\hat{\lambda}$.

Path estimation:

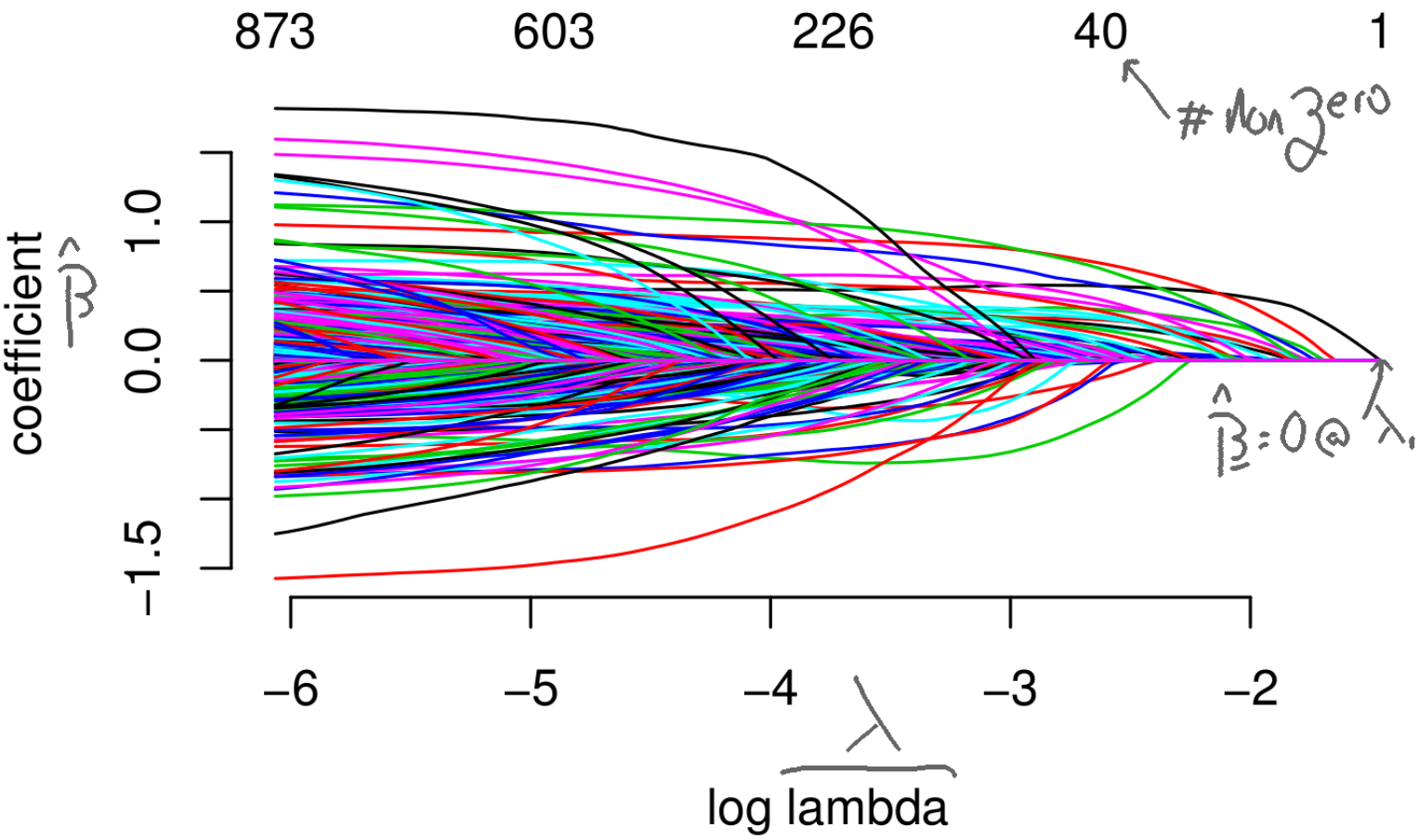
Start with big λ_1 so big that $\hat{\boldsymbol{\beta}} = \mathbf{0}$.

For $t = 2 \dots T$: update $\hat{\boldsymbol{\beta}}$ to be optimal under $\lambda_t < \lambda_{t-1}$.

Since estimated $\hat{\boldsymbol{\beta}}$ changes smoothly along this path:

- It's fast! Each update is easy.
- It's stable: optimal λ_t may change a bit from sample to sample, but that won't affect the model much.

The whole enterprise is easiest to understand visually.



The algorithm moves *right to left*.

The y-axis is $\hat{\boldsymbol{\beta}}$ (each line a different $\hat{\beta}_j$) as a function of λ_t.

Example: Comscore web browser data

The previous plot is household log-online-spend regressed onto % of time spent on various websites (each β_j a different site).

Comscore records info on browsing
and purchasing behavior for annual panels of households.

I've extracted 2006 data for the 1000 most heavily trafficked websites and for 10,000 households that spent at least 1$.

Why do we care? Predict consumption from browser history.

e.g., to control for base-level spending, say, in estimating advertising effectiveness. You'll see browser history of users when they land, but likely not what they have bought.

http://faculty.chicagobooth.edu/matt.taddy/teaching/comscore.R