An Introduction to Regularized Regression

Machine Learning and Causal Inference

Susan Athey

Thanks to Sendhil Mullainathan for sharing slides; see Mullainathan and Speiss (2017) JEP which covers much of this material

What we do in Econometrics: The Case of Regression

- Specify a model:

$$Y_i = f(X_i) + \epsilon_i = X_i\beta + \epsilon_i$$

- Data set has observations i=1,..,n
- Use OLS regression on the entire dataset to construct an estimate $\hat{\beta}$
- Discuss assumptions under which some components of $\hat{\beta}$ have a causal interpretation
- Consider that S_n (set of observed units, i=1,..,n) is a random sample from a much larger population.
- Construct confidence intervals and test the hypothesis that some components are equal to zero.
- Theorem: OLS is BLUE (Best Linear Unbiased Estimator)
  - Best = lowest-variance

Goals of Prediction and Estimation

- Goal of estimation: unbiasedness

$$E[\hat{f}] = f$$

- Goal of prediction: loss minimization

$$L(f) = E_{(x,y)}\ell(f(x), y)$$

$$\hat{f} \approx \min_{f \in \mathcal{F}} L(f)$$

  - E.g. $\ell(f(x), y) = (f(x) - y)^2$
  - Use the data to pick a function that does well on a new data point

Key assumptions in both cases

- Stationary data generating process

$$\text{Data}$$
$$S_n = (y_i, x_i)\ iid$$

- Estimation:
  - Interested in a parameter of that process
- Prediction:
  - Interested in predicting *y*

High v. Low Dimensional Analysis

- We have discussed prediction as a high dimensional construct
- Practically that is where it is useful
- But to understand how high dimensional prediction works we must unpack an implicit presumption
  - Presumption: Our known estimation strategies would be great predictors *if they were feasible*

A Simple OLS example

- Suppose we truly live in a linear world

$$y = \beta_0 + \beta_1 x_1 + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0, \sigma_\epsilon^2) \quad x_1 \sim N(0,1)$$

- Write x = (1,x)

$$y = \beta x + \varepsilon.$$

OLS seems like a good predictor

$$L(\hat{f}^{\mathrm{OLS}}) = \mathrm{E}_{(y,x)}(\hat{\beta}'x - y)^2 = (\hat{\beta}_0 - \beta_0)^2 + (\hat{\beta}_1 - \beta_1)^2 + \sigma_\varepsilon^2$$

So wouldn't we want the $\hat{\beta}$ with $\mathrm{E}_{S_n}(\hat{\beta}) = \beta$?

Especially since it is known to be efficient

An Even Simpler Set-up

- Let's get even lower dimensional
- No variables at all
- Suppose you get the data of the type:

$$y_i = \mu + \epsilon_i$$

- You would like to estimate the mean

Forming an estimator of the mean

$$\begin{aligned} \hat{\mu} &= \alpha \bar{y} \\ E[\hat{\mu}] &= \alpha \mu \end{aligned}$$

- Minimize bias: $\alpha = 1$
- The sample mean is an unbiased estimator
  - Also what you would get from OLS regression on a constant

A prediction problem

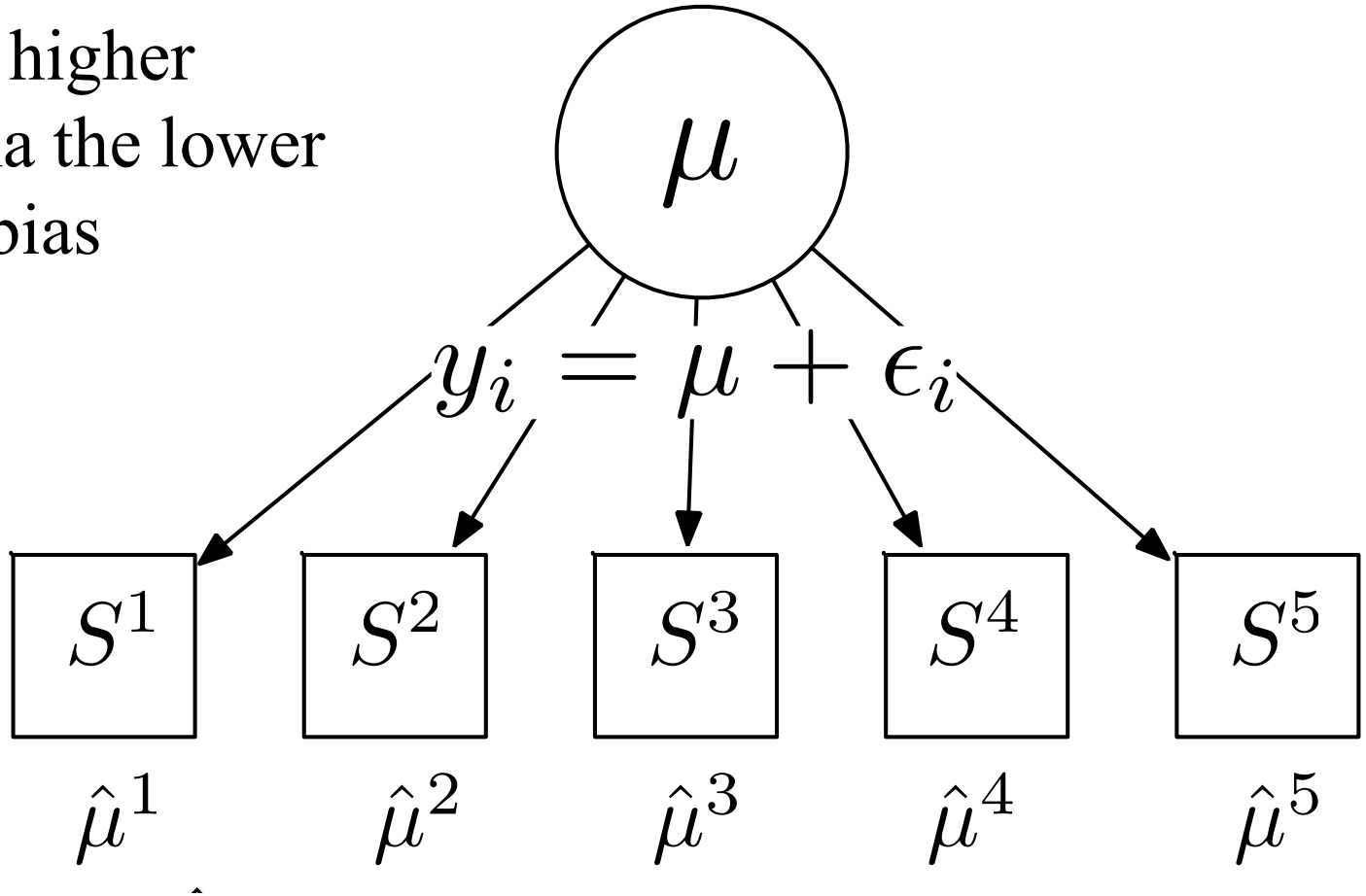
- In the same setup, you are given n data points
- You would like to guess the value of a new data point from the same distribution
- Goal: minimize quadratic loss of prediction

Best Predictor

$$\begin{aligned} \hat{\mu} &= \alpha \bar{y} \\ E[\hat{\mu}] &= \alpha \mu \\ E[\ell(\hat{\mu}, y)] &= [(1-\alpha)\mu]^2 + \frac{1}{n}\alpha^2 \sigma_\epsilon^2 + \sigma_\epsilon^2 \end{aligned}$$

$$\hat{\mu} = \alpha \bar{y}$$

The higher alpha the lower the bias



The higher alpha the more variable across samples it is

Key problem

- The unbiased estimator has a nice property:

$$E[\hat{\mu}|\mu] = \mu$$

- But getting that property means large sample to sample variation of estimator

- This sample to sample variation means that in any particular finite sample I'm paying the cost of being off on all my predictions

Intuition

- I see your first test score. What should my prediction of your next test be?
  - Your first test score is an unbiased estimator
  - But it is very variable

- Note: "Bayesian" intuition
  - Even simpler: what was my guess before I saw any information
  - Shrink to that
  - In this example I'm shrinking to zero

But in a way you know this

- As empiricists you already have this intuition

Back to Simple OLS example

- Suppose we truly live in a linear world

$$y = \beta_0 + \beta_1 x_1 + \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0, \sigma_\epsilon^2) \quad x_1 \sim N(0, 1)$$

- Write x = (1,x)

$$y = \beta x + \varepsilon.$$

A Simple Example

- You run a one variable regression and get

$$\hat{\beta}_0^{\text{OLS}} = 0 \pm .2$$
$$\hat{\beta}_1^{\text{OLS}} = 2 \pm 10$$

- Would you use the OLS coefficients to predict

- Or drop the first variable and use this:

$$\hat{\beta}_0^{OLS} = \arg\min_{\beta_0} \widehat{\mathbb{E}}_{S^n}(\beta_0 - y)^2 = \widehat{\mathbb{E}}_{S^n} y$$

Deciding whether to drop

- Suppose in the (impossible) case we got the true world right.
  - (0,2) are the right coefficients
- Of course OLS does perfectly (by assumption).
- But how would OLS do on new samples…where (0,2) being the generating coefficients?
  - We're giving OLS a huge leg up here.

OLS Performance

$$
\begin{aligned}
&\mathcal{L}_n(\mathrm{OLS}) - \sigma_\varepsilon^2 = \\
&= \quad \mathrm{E}_{(y,x)}\, \mathrm{E}_{S_n}[\beta' x - (\hat{\beta}^{\mathrm{OLS}})' x]^2 \\
&= \quad \mathrm{E}_{(y,x)}[\underbrace{(\beta' x - (\mathrm{E}_{S_n} \hat{\beta}^{\mathrm{OLS}})' x)^2}_{\text{unbiased}} \to 0 + \mathrm{Var}_{S_n}((\hat{\beta}^{\mathrm{OLS}})' x)] \\
&= \quad \mathrm{Var}_{S_n}(\hat{\beta}_0^{\mathrm{OLS}}) + \mathrm{Var}_{S_n}(\hat{\beta}_1^{\mathrm{OLS}})
\end{aligned}
$$

What if we dropped the variable

$$\mathcal{L}_n(\mathrm{OLS}_0) - \sigma_\varepsilon^2 =$$

$$
\begin{aligned}
&\mathcal{L}_n(\mathrm{OLS}) - \sigma_\varepsilon^2 = \\
&= \mathrm{E}_{(y,x)}\, \mathrm{E}_{S_n}[\beta' x - (\hat{\beta}^{\mathrm{OLS}})' x]^2 \\
&= \mathrm{E}_{(y,x)}[\underbrace{(\beta' x - (\mathrm{E}_{S_n}\hat{\beta}^{\mathrm{OLS}})' x)^2}_{\text{unbiased}} \to 0 + \mathrm{Var}_{S_n}((\hat{\beta}^{\mathrm{OLS}})' x)] \\
&= \mathrm{Var}_{S_n}(\hat{\beta}_0^{\mathrm{OLS}}) + \mathrm{Var}_{S_n}(\hat{\beta}_1^{\mathrm{OLS}})
\end{aligned}
$$

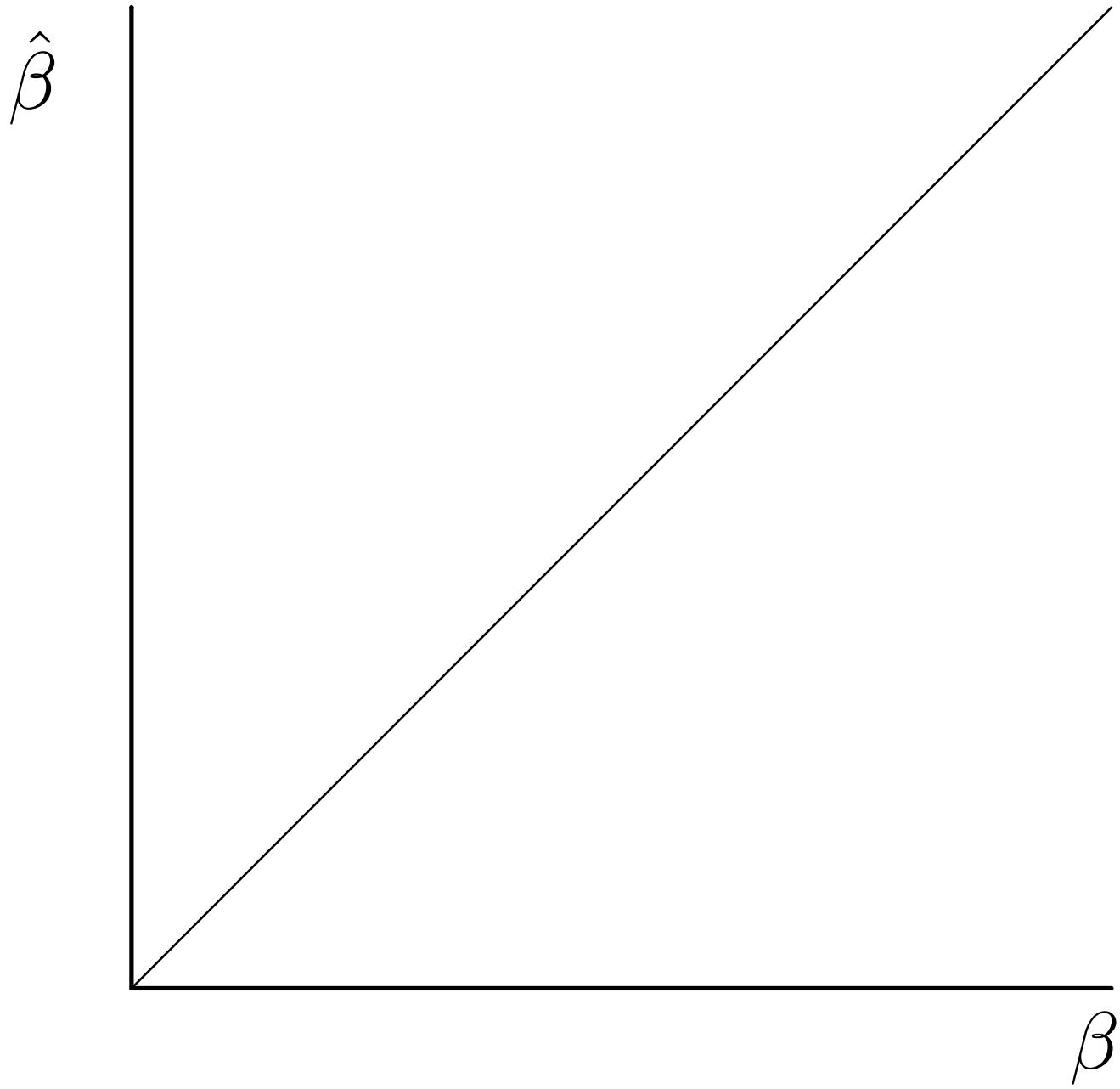
$$
\begin{aligned}
&\mathcal{L}_n(\mathrm{OLS}_0) - \sigma_\varepsilon^2 = \\
&= \mathrm{E}_{(y,x)}\, \mathrm{E}_{S_n}[\beta' x - (\hat{\beta}^{\mathrm{OLS}_0})' x]^2 \\
&= \mathrm{E}_{(y,x)}[\underbrace{(\beta' x - (\mathrm{E}_{S_n}\hat{\beta}^{\mathrm{OLS}_0})' x)^2}_{\text{bias}} + \mathrm{Var}_{S_n}((\hat{\beta}^{\mathrm{OLS}_0})' x)] \\
&= (0-2)^2 + \mathrm{Var}_{S_n}(\hat{\beta}_0^{\mathrm{OLS}_0}) + \mathrm{Var}_{S_n}(\hat{\beta}_1^{\mathrm{OLS}_0}) \to 0
\end{aligned}
$$

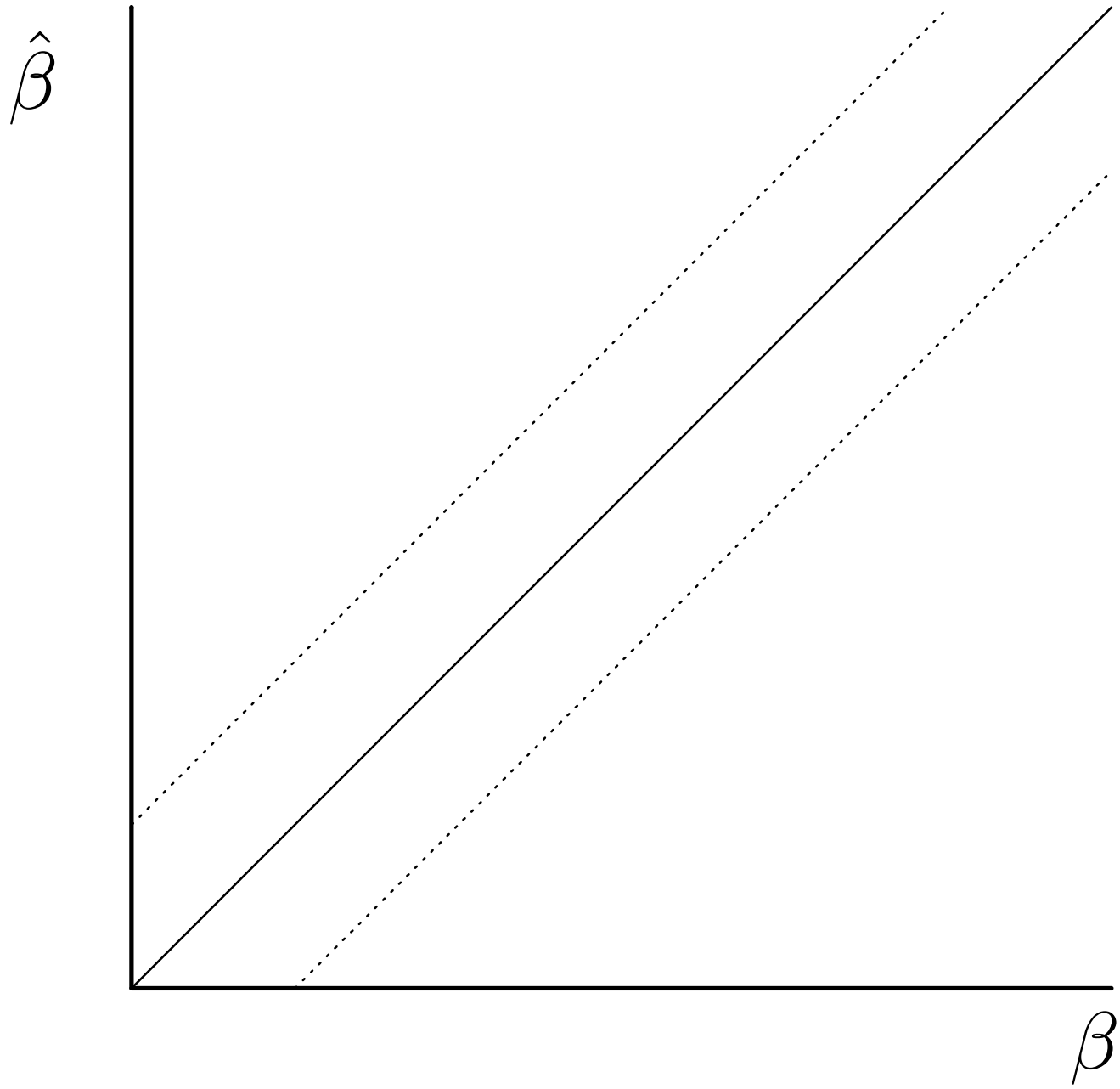
$$
\mathcal{L}_n(\mathrm{OLS}) - \mathcal{L}_n(\mathrm{OLS}_0) = \underbrace{\mathrm{Var}(\hat{\beta}_1^{OLS})}_{\text{variance}} - \underbrace{(0-2)^2}_{\text{bias}}
$$

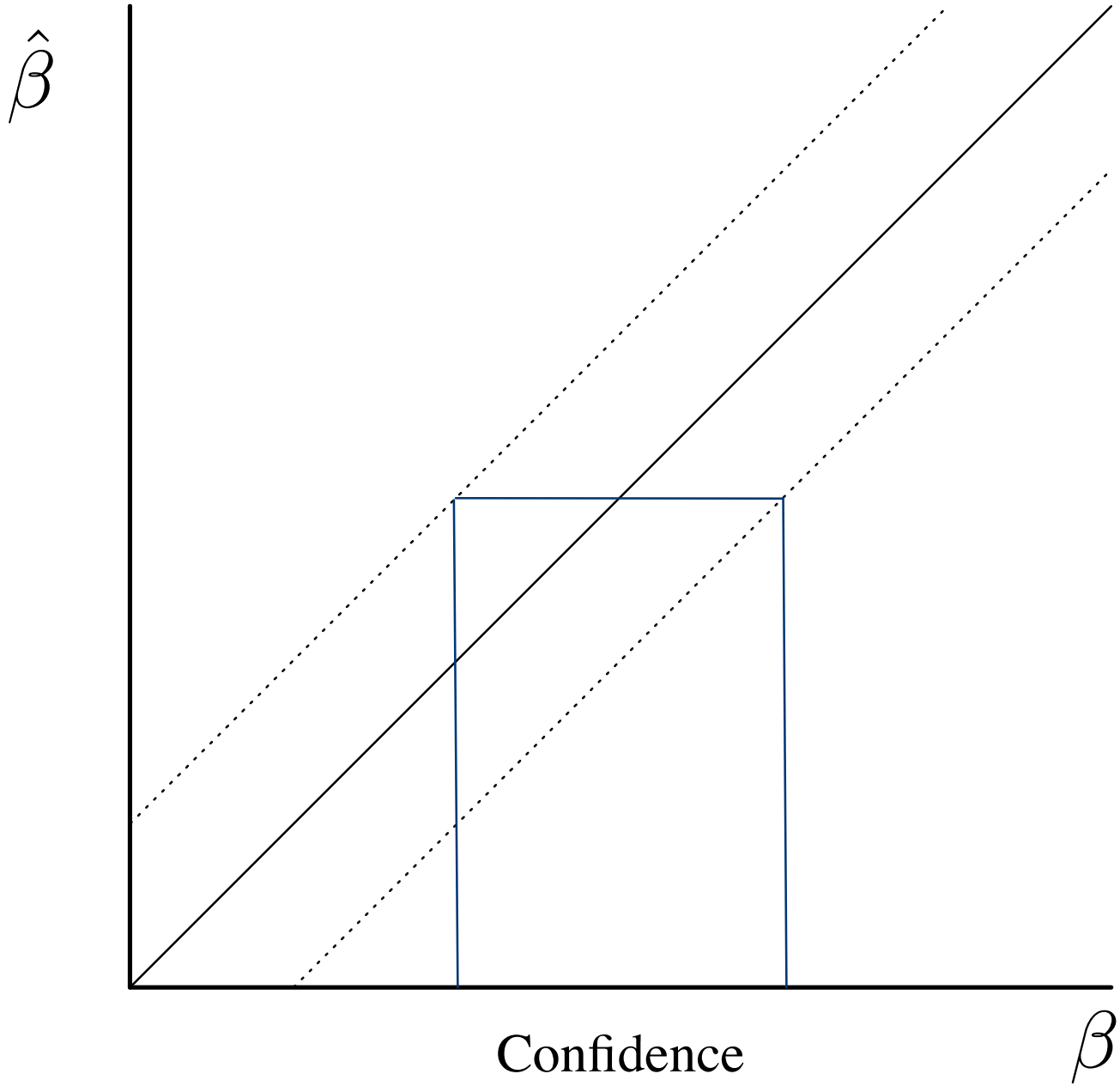
Your standard error worry!

Where does your standard error intuition come from?

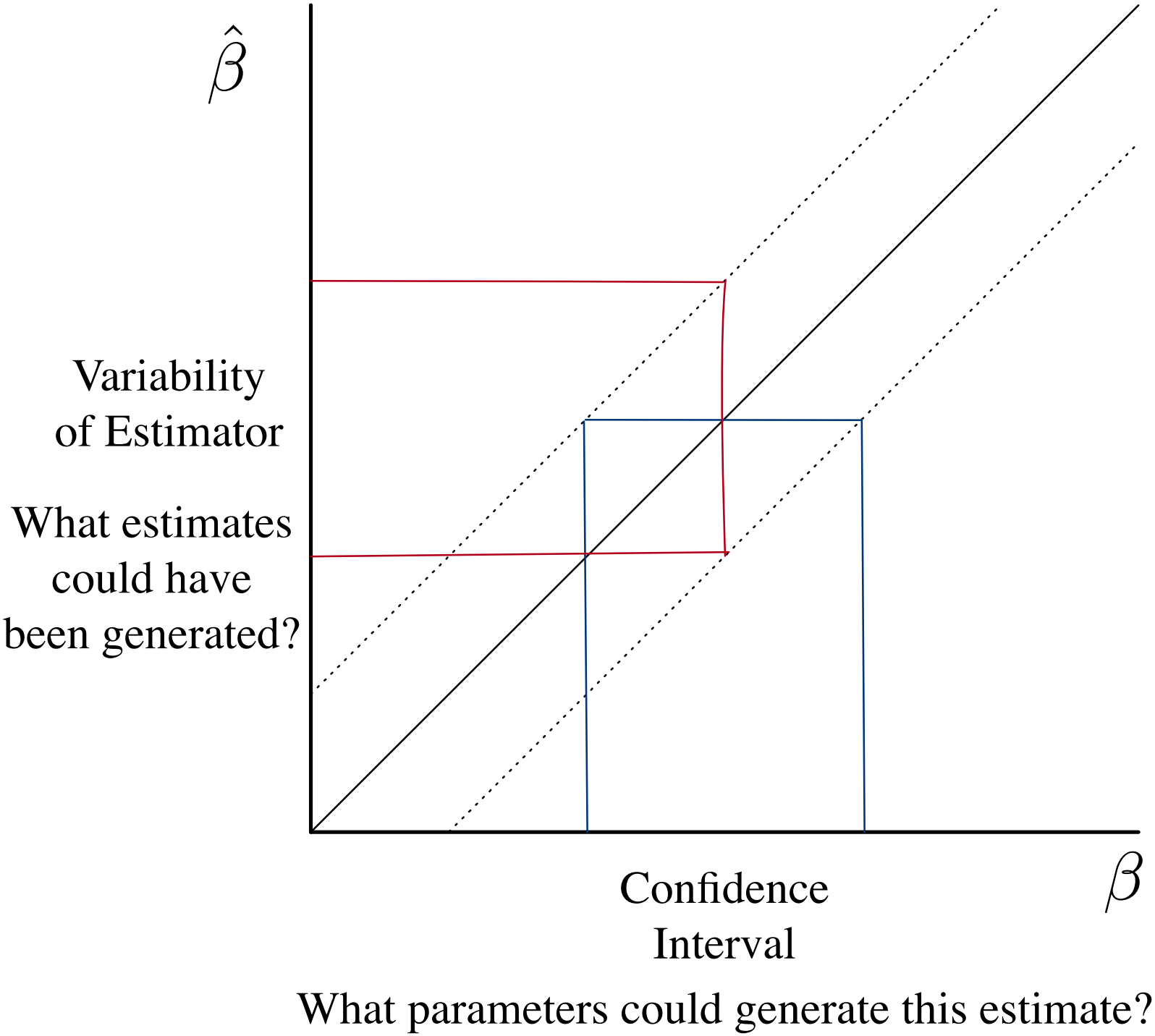
- You see a standard error
- You think “that variable is not ‘significant’” so you might not want to include it.
- But this is misleading


$\hat{\beta}$
β


$\hat{\beta}$
β



What parameters could generate this estimate?


$\hat{\beta}$
Variability
of Estimator
What estimates
could have
been generated?
β
Confidence
Interval
What parameters could generate this estimate?

Your Standard Error Worry

- For hypothesis testing se tells you whether the coefficient is significant are not
- For prediction it's telling you how variable an estimator using it really is

Dual purposes of the standard eror

- The standard error also tells you that even if you're right on average:
  - Your estimator will produce a lot of variance
  - And then in those cases you make systematic prediction mistakes.
- Bias variance tradeoff
  - Being right on average on the coefficient is not equal to the best predictor.

The Problem Here

- Prediction quality suffers from:
  - Biased coefficients
  - Variability in *estimated* coefficients
    - Even if the true coefficient is 2, in any sample, we will estimate something else
- OLS is lexicographic
  - First ensure unbiased
  - Amongst unbiased estimators: seek efficiency
- Good predictions must trade these off

Two Variable Example

- Belaboring the point here…
- Assume now that we have two variables
  - As before, both normally distributed unit variance
- Your estimator produces

$$\hat{\beta}_0^{OLS} = 0 \pm .2$$

$$\hat{\beta}_1^{OLS} = 2 \pm 10$$

What would you do now?

- Logic above suggests you would drop both variables?
- Or keep both variables?
- It really depends on how you feel about the variance (10)?

Calculation

$$\mathcal{L}_n(\mathrm{OLS}) - \mathcal{L}_n(\mathrm{OLS}_0) = \overbrace{\mathrm{Var}(\hat{\beta}_1^{\mathrm{OLS}}) + \mathrm{Var}(\hat{\beta}_2^{\mathrm{OLS}})}^{\text{variance}} - \overbrace{((0-2)^2 + (0-2)^2)}^{\text{bias}} + \underbrace{2\rho_{12}\,\mathrm{Cov}(\hat{\beta}_1^{\mathrm{OLS}}, \hat{\beta}_2^{\mathrm{OLS}})}_{\text{covariance variance}} - \underbrace{2\rho_{12}(0-2)^2}_{\text{covariance bias}}$$

$$\mathcal{L}_n(\mathrm{OLS}) - \mathcal{L}_n(\mathrm{OLS}_0) = \underbrace{\mathrm{Var}(\hat{\beta}_1^{OLS})}_{\text{variance}} - \underbrace{(0-2)^2}_{\text{bias}}$$

Hidden in Bias-Variance Tradeoff

- Covariance is central
- The standard error on several variables can be large, even though together their effect is highly consistent
- For prediction covariance between x matters

In a way this problem is not important

$$\mathcal{L}_n(\text{OLS}) - \mathcal{L}_n(\text{OLS}_0) = \underbrace{\text{Var}(\hat{\beta}_1^{OLS})}_{\text{variance}} - \underbrace{(0-2)^2}_{\text{bias}}$$

- The variance term diminishes with sample size
  - Prediction-estimation wedge falls off as $\frac{1}{n}$
- But variance term increases with "variables"
  - Prediction-estimation rises with k
- So this is a problem when…
  - Function class high dimensional relative to data $\frac{k}{n}$

What this means practically

- In some cases what you already know (estimation) is perfectly fine for prediction
  - This is why ML textbooks teach OLS, etc.
  - They are perfectly useful for the kinds of prediction problems ML tries to solve *in low dimensional settings*
- But in high dimensional settings…
  - Note: high dimensional does not ONLY mean lots of variables! It can mean rich interactions.

So far…

- All this gives you a flavor of how the prediction task is not mechanically a consequence of the estimation task
- But it doesn't really tell you **how** to predict
  - Bias variance tradeoff is entirely unactionable
  - What's the bias?
  - What's the variance?
  - This is not really a tradeoff you can make
- A different look at the same problem produces a practical insight though

Back to OLS

$$\hat{\beta}^{\text{OLS}} = \arg\min_{\beta} \widehat{\mathbb{E}}_{S_n}(\beta' x - y)^2$$

AVERAGES NOTATION: $\widehat{\mathbb{E}}_{S_n}$ for sample ave. for sample S_n

$$\beta^*_{\text{prediction}} = \arg\min_{\beta} \mathrm{E}_{(y,x)}(\beta' x - y)^2$$

- The real problem here is minimizing the "wrong" thing: In-sample fit vs out-of-sample fit

Overfit problem

- OLS looks good with the sample you have
  - It's the best you can do *on this sample*
- Bias-variance improving predictive power is about improving *out of sample* predictive power
- Problem is OLS by construction overfits
  - We overfit in estimation

This problem is exactly why wide data is troubling

- Similarly think of the wide data case
- Why are we worried about having so many variables?
- We'll fit very well (perfectly if $k > n$) in sample
- But arbitrarily badly out of sample

Understanding overfit

- Let’s consider a general class of algorithms

A General Class of Algorithms

- Let $L(f) = \int_{x,y} \ell(f(x), y) dP(x, y)$ for some loss function ℓ (e.g. squared error)
  - Note: L is an unknown function: we don't know P
- Consider algorithms of the form

$$\hat{f}_{A,S_n} = \arg\min_{f \in \mathcal{F}_{\mathcal{A}}} \hat{L}_{S_n}(f)$$

  - $\hat{L}_{S_n}$ is used here as shorthand for sample mean observations in sample S_n of size n
  - OLS is an *empirical loss minimizer*: it minimizes the sample average over observed data of the loss function

- So empirical loss minimization algorithms are defined by the function class they choose from
- For estimation what we typically do…
  - Show that empirical loss minimizers generate unbiasedness

Empirical Loss minimization

- Leads to unbiasedness/consistency
  - Fit the data you have…
  - In a frequentist world "on average" (across all S_n) this will produce the right thing
  - This is usually how we prove consistency/unbiasedness
- Other variants:
  - MLE

Some Notation

- Define

$$f^* = \arg\min_{f \in \mathcal{F}} L(f)$$

The best we can do

$$f_A^* = \arg\min_{f \in \mathcal{F}_A} L(f)$$

The best in the subset of functions that the algorithm looks at

  - Recall: L is infeasible b/c we don't know true data-generating process
- Contrast the latter with:

$$\hat{f}_{A,S_n} = \arg\min_{f \in \mathcal{F}_{\mathcal{A}}} \hat{L}_{S_n}(f)$$

What the in-sample loss minimizer actually produces given a sample

Performance of Algorithm

- Performance of a predictor

$$L(\hat{f}_{A,S_n})$$

- Performance of an Algorithm

$$\mathcal{L}_n(A) := E_{S_n} L(\hat{f}_{A,S_n})$$

  - Algorithm's expected loss
  - (Suppress S_n in some of the notation for estimator)

The performance of *A*

$$\mathcal{L}_n(A) = \underbrace{L(f^*)}_{\text{irreducible error}} + \overbrace{L(f_A^*) - L(f^*)}^{\text{approximation error}} + \underbrace{\mathrm{E}_{S_n}(L(\hat{f}_A) - L(f_A^*))}_{\text{estimation error}},$$

Understanding estimation error:

$$\mathrm{E}_{S_n}(L(\hat{f}_A) - L(f_A^*)) = \mathrm{E}_{S_n}(\hat{L}(\hat{f}_A) - \hat{L}(f_A^*)) + \mathrm{E}_{S_n}(L(\hat{f}_A) - \hat{L}(\hat{f}_A))$$

"Wrong" function looks good in-sample

Algorithm does not see this

Basic Tradeoff

- These two terms go hand in hand:

$$\mathcal{L}_n(A) = \underbrace{L(f^*)}_{\text{irreducible error}} + \overbrace{L(f_A^*) - L(f^*)}^{\text{approximation error}} + \underbrace{\mathrm{E}_{S_n}(L(\hat{f}_A) - L(f_A^*))}_{\text{estimation error}},$$

$$\overbrace{\mathrm{E}_{S_n}(L(\hat{f}_A) - L(f_A^*))}^{\text{out-of-sample, } \geq 0} = \overbrace{\mathrm{E}_{S_n}(\hat{L}(\hat{f}_A) - \hat{L}(f_A^*))}^{\text{in-sample, } \leq 0} + \underbrace{\mathrm{E}_{S_n}(L(\hat{f}_A) - \hat{L}(\hat{f}_A))}_{\text{unseen overfit}}$$

Approximation – Overfit Tradeoff

- If we reduce set of *f* to reduce possible over-fit:

$$\underbrace{\mathrm{E}_{S_n}(L(\hat{f}_A) - \hat{L}(\hat{f}_A))}_{\text{unseen overfit}}$$

- Then we fit fewer "true functions" and drive up

$$\overbrace{L(f_A^*) - L(f^*)}^{\text{approximation error}}.$$

- Only way to avoid this is if we knew information about *f** so we could shrink the set

Unobserved overfit

- So the problem of prediction really is managing unobserved overfit

$$\underbrace{L(\hat{f}_A)}_{\text{unobserved out-of-sample}} = \overbrace{\hat{L}(\hat{f}_A)}^{\text{observed in-sample}} + \underbrace{(L(\hat{f}_A) - \hat{L}(\hat{f}_A))}_{\text{unobserved overfit}}$$

- We do well in-sample. But some of that "fit" is overfit.

Return to the original example

OLS OLS_0

Greater Chance To Overfit

Less Chance To Overfit

- We drove down overfit by doing a constrained optimization

Basic Tradeoff at the Heart of Machine Learning

- Bigger function classes…
  - The more likely we are to get to the truth (less approximation)
  - The more likely we are to overfit
- So we want to not just minimize in-sample error given a class of functions
- We also want to decide on the class of functions
  - More expressive means less approximation error
  - More expressive means more overfit

Let's do the same thing here

Unconstrained

$$\hat{f}_{A,S_n} = \arg\min_{f \in \mathcal{F}_{\mathcal{A}}} \hat{L}_{S_n}(f)$$

But we are worried about $\underbrace{\mathrm{E}_{S_n}(L(\hat{f}_A) - \hat{L}(\hat{f}_A))}_{\text{unseen overfit}}$

So why not do this instead?

$\arg\min_{f \in \mathcal{F}_{\mathcal{A}}} \hat{L}_{S_n}(f)$

s.t. $R(f) \leq c$

Complexity measure: tendency to overfit

Return to the original example

OLS	OLS_0
Greater Overfit	Less Overfit
Better approximation	Worse approximation
More **Expressive** *R(f) higher*	Less **Expressive** *R(f) lower*

- Reduce overfit by approximating worse
- Choose less expressive function class

Constrained minimization

- We could do a constrained minimization
- But notice that this is equivalent to:

$$\hat{f}_{A_\lambda, S_n} = \arg\min_{f \in \mathcal{F}_\mathcal{A}} \hat{L}_{S_n}(f) + \underbrace{\lambda R(f)}_{\text{want: } \approx L(f) - \hat{L}(f)}$$

- Complexity measure should capture tendency to overfit

Basic insight

- Data has signal and noise
- More expressive function classes-
  - Allow us to pick up more of the signal
  - But also pick up more of the noise
- So the problem of prediction becomes the problem of *choosing expressiveness*

Overall Structure

- Create a regularizer that:
  - Measures expressiveness
- Penalize algorithm for choosing more expressive functions
  - Tuning parameter lambda
- Let it weigh this penalty against in-sample fit

Linear Example

- Linear function class $x \mapsto \beta' x \; (\beta \in \mathbb{R}^{k+1})$

- Regularized linear regression

$$\hat{\beta}_\lambda^R = \arg\min_{\beta \in \mathbb{R}^{k+1}} \widehat{\mathbb{E}}_{S_n} (\beta' x - y)^2 - \lambda R(\beta)$$

Regularizers for Linear Functions

- Linear functions more expressive if use more variables

$$R(\beta) = \sum_{j=1}^{k} 1_{\beta_j \neq 0}$$

- Can transform coefficients

$$R(\beta) = \sum_{j=1}^{k} |\beta_j|^p$$

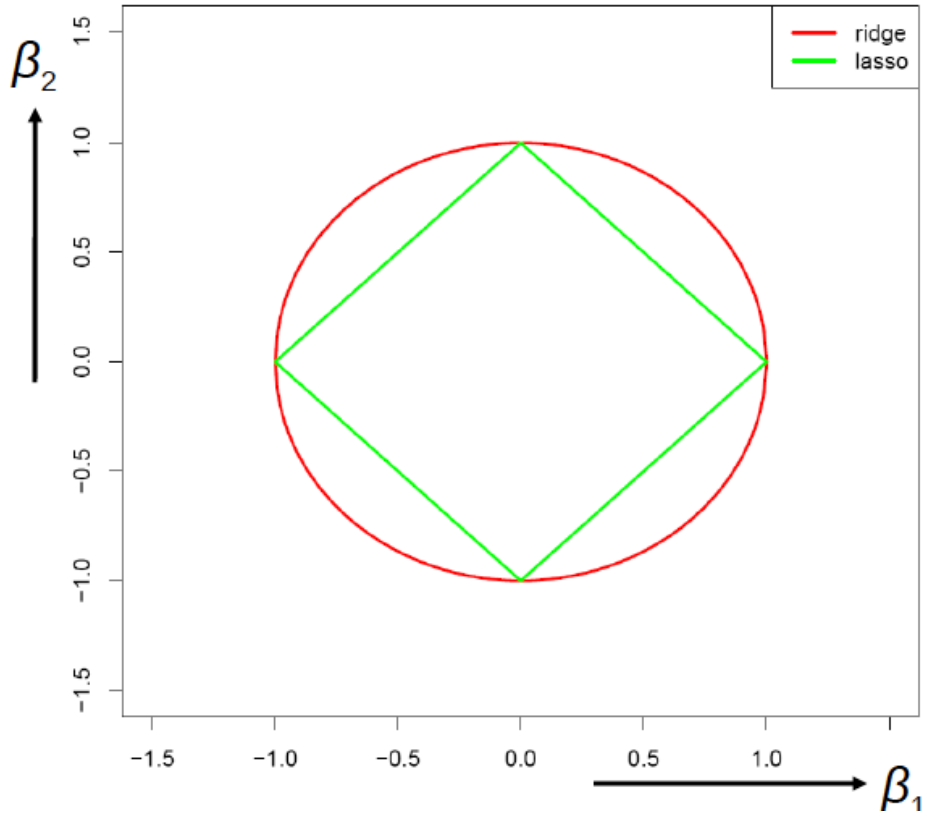
Computationally More Tractable

- Lasso

$$\mathcal{F}_{1,c} = \{f_\gamma; \sum_{j=1}^{k} |\gamma_j| \leq c\}$$

- Ridge

$$\mathcal{F}_{2,c} = \{f_\gamma; \sum_{j=1}^{k} \gamma_j^2 \leq c\}$$



What makes a good regularizer?

- You might think…
  - Bayesian assumptions
  - Example: Ridge
- A good regularizer can build in beliefs
- Those are great and useful when available
- But central force is tendency to overfit
- Example:
  - Even if true world were not sparse or priors were not normal you'd still do this

Summary

- Regularization is one half of the secret sauce
- Gives a single-dimensional way of deciding of capturing expressiveness

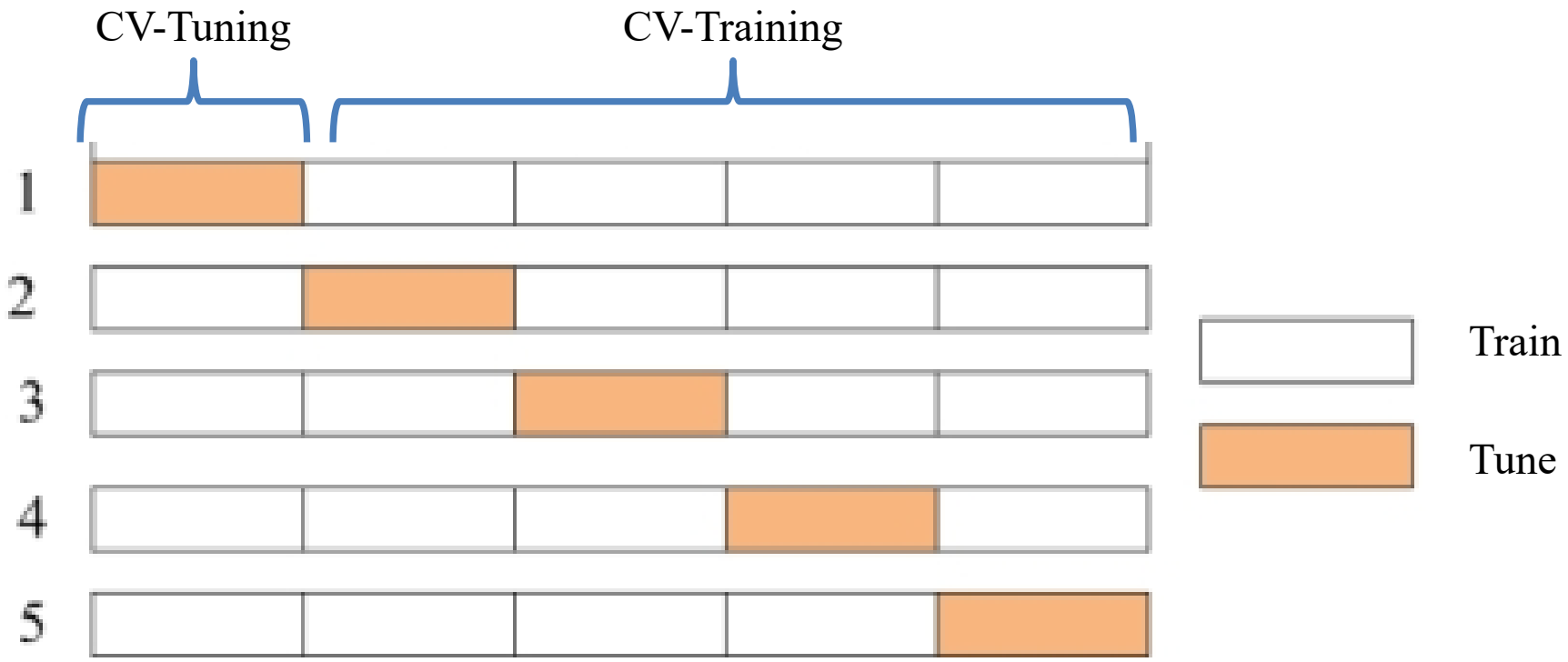
$$\hat{f}_{A_\lambda, S_n} = \arg\min_{f \in \mathcal{F}_\mathcal{A}} \hat{L}_{S_n}(f) + \lambda R(f)$$

- Still missing ingredient is lambda

Choosing lambda

- How much should we penalize expressiveness?
- How do you make the over-fit approximation tradeoff?
- The **tuning** problem.
- Use cross-validation

How Does Cross Validation Work?



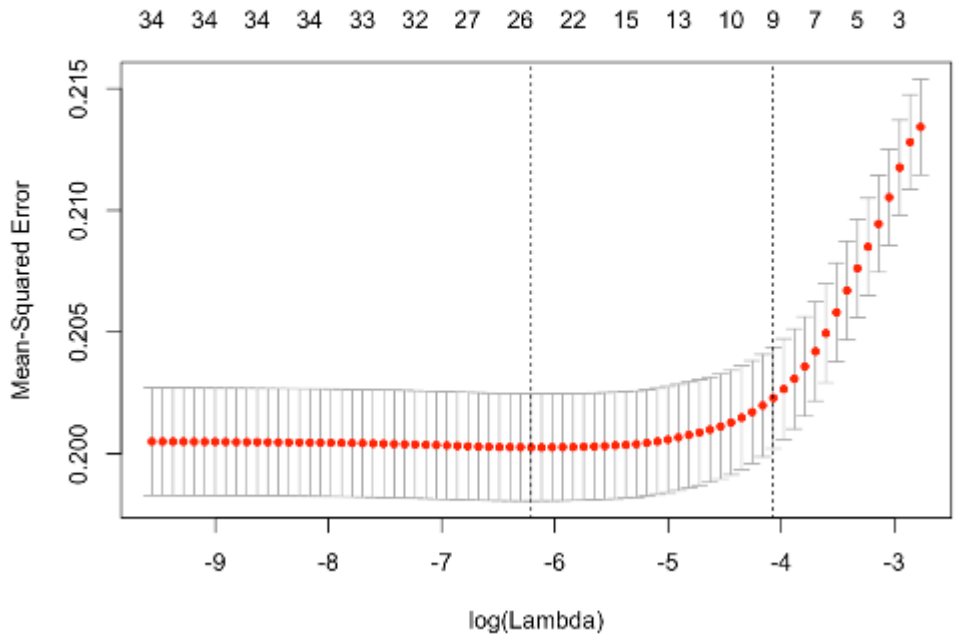
Tuning Set = 1/5 of Training Set

Cross-Validation Mechanics

- Loop over cross-validation samples
  - Train a deep tree on CV-training subset
- Loop over penalty parameters λ
  - Loop over cross-validation samples
    - Prune the tree according to penalty
    - Calculate new MSE of tree
  - Average (over c-v samples) the MSE for this penalty
- Choose the penalty λ^* that gives the best average MSE

LASSO c-v Example

```
plot(lasso.linear)
```



Creating Out-of-Sample **In Sample**

- Major point:
  - Not many assumptions
  - Don't need to know true model.
  - Don't need to know much about algorithm
- Minor but important point
  - To get asymptotics right we need to make some regularity assumptions
- Side point (to which we return)
  - We'd like to choose best algorithm for sample size *n*
  - But this will not do that. **Why?**

Why does this work?

1. Not just because we can split a sample and call it out of sample

   – It's because the thing we are optimizing is observable (easily estimable)

This is more than a trick

- It illustrates what separates prediction from estimation:
  - I can't 'observe' my prior.
    - Whether the world is truly drawn from a linear model
  - But prediction quality is observable
- Put simply:
  - Validity of predictions are measurable
  - Validity of coefficient estimators require structural knowledge

This is the essential ingredient to prediction: Prediction quality is an empirical quantity not a theoretical guarantee

Why does this work?

1. It's because the thing we are optimizing is observable

2. By focusing on prediction quality we have reduced dimensionality

To understand this…

- Suppose you tried to use this to choose coefficients
  - Ask which set of coefficients worked well out-of sample.
- Does this work?
- Problem 1: Estimation quality is unobservable
  - Need the same assumptions as algorithm to know whether you "work" out of sample
    - If you just go by fit you are ceding to say you want best predicting model
- Problem 2: No dimensionality reduction.
  - You've got as many coefficients as before to search over

$$\hat{\beta}_\lambda^R = \underset{\beta \in \mathbb{R}^{k+1}}{\arg\min}\, \mathbb{E}_{S_n}(\beta' x - y)^2 + \lambda R(\beta)$$

Method	$R(\beta)$
OLS	0
Subset selection	$\|\beta\|_0 = \sum_{j=1}^k \mathbb{1}_{\beta_j \neq 0}$
Lasso	$\|\beta\|_1 = \sum_{j=1}^k \|\beta_j\|$
Ridge	$\|\beta\|_2^2 = \sum_{j=1}^k \beta_j^2$
Elastic Net	$\alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2$

Bayesian Interpretation of Ridge

Consider the regression

$$Y_i = \sum_{k=1}^{K} \beta_k \cdot X_{ik} + \varepsilon_i$$

with

$$\varepsilon_i | X_{i1}, \ldots, X_{iK} \sim \mathcal{N}(0, \sigma^2)$$

Suppose we put a prior on the β_k:

$$\beta_k \sim \mathcal{N}(0, \tau^2)$$

and all the β_k independent. Assume σ^2 is known.

Bayesian Interpretation of Ridge

Then the posterior distribution is proportional to

$$p(\beta|\text{data}) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(Y_i - \sum_{k=1}^{K}\beta_k \cdot X_{ik}\right)^2\right)\prod_{k=1}^{K}\exp\left(-\frac{\beta_k^2}{2\tau^2}\right)$$

$$= \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(Y_i - \sum_{k=1}^{K}\beta_k \cdot X_{ik}\right)^2 - \sum_{k=1}^{K}\frac{\beta_k^2}{2\tau^2}\right)$$

$$= \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(Y_i - \beta' X_i\right)^2 - \frac{\beta'\beta}{2\tau^2}\right)$$

Bayesian Interpretation of Ridge

So, the posterior is normal, and the posterior mean minimizes

$$\sum_{i=1}^{N} \left(Y_i - \beta' X_i\right)^2 + \beta'\beta \cdot \frac{\sigma^2}{\tau^2}$$

$$= \sum_{i=1}^{N} \left(Y_i - \beta' X_i\right)^2 + \frac{\sigma^2}{\tau^2} \cdot \|\beta\|^2$$

This leads to the posterior mean

$$\left(\mathbf{X}'\mathbf{X} + I_K \cdot \sigma^2/\tau^2\right)^{-1} \mathbf{X}'\mathbf{Y}.$$

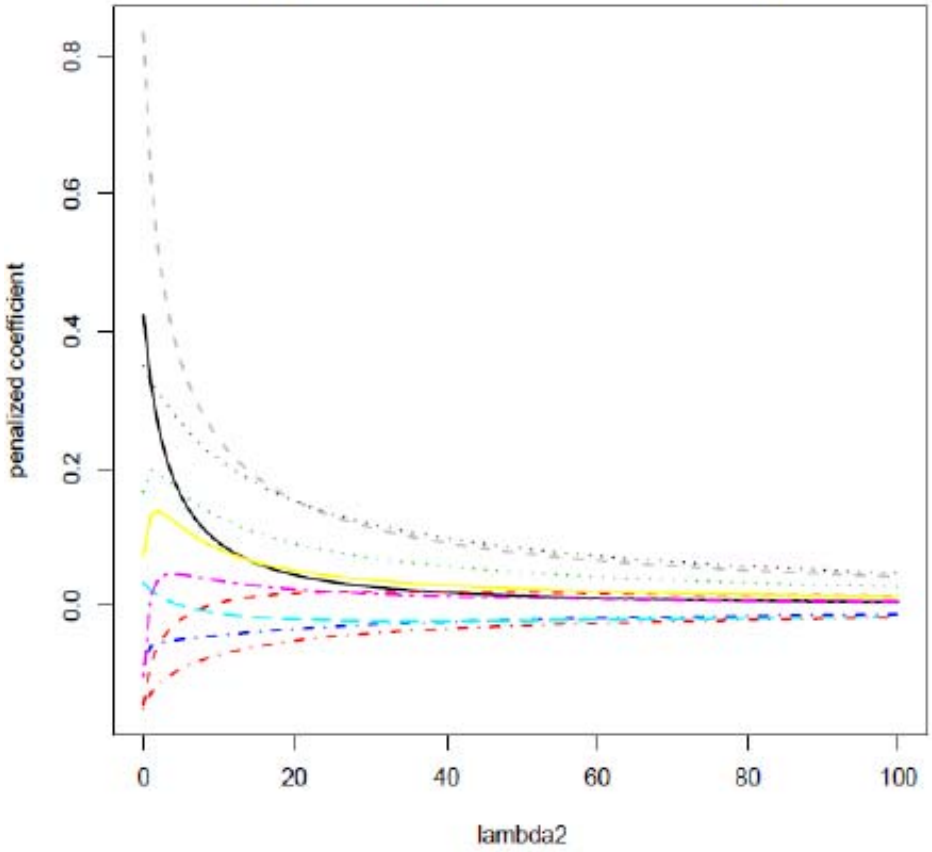
If the $\mathbf{X}'\mathbf{X}$ matrix is diagonal, all elements of β would be shrunk towards zero by the same fraction. With a non-diagonal matrix the degree of shrinkage varies.

POST-Lasso

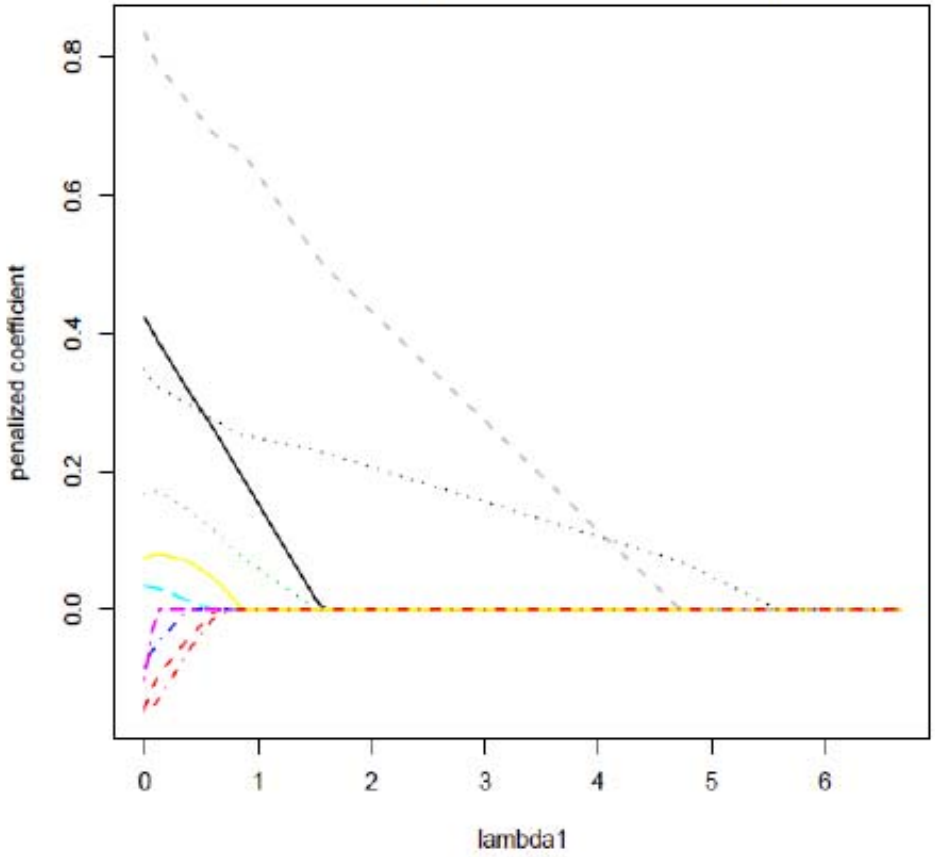
- Important distinction:
  - Use LASSO to choose variables
  - Use OLS on these variables

- How should we think about these?

Ridge regularization path



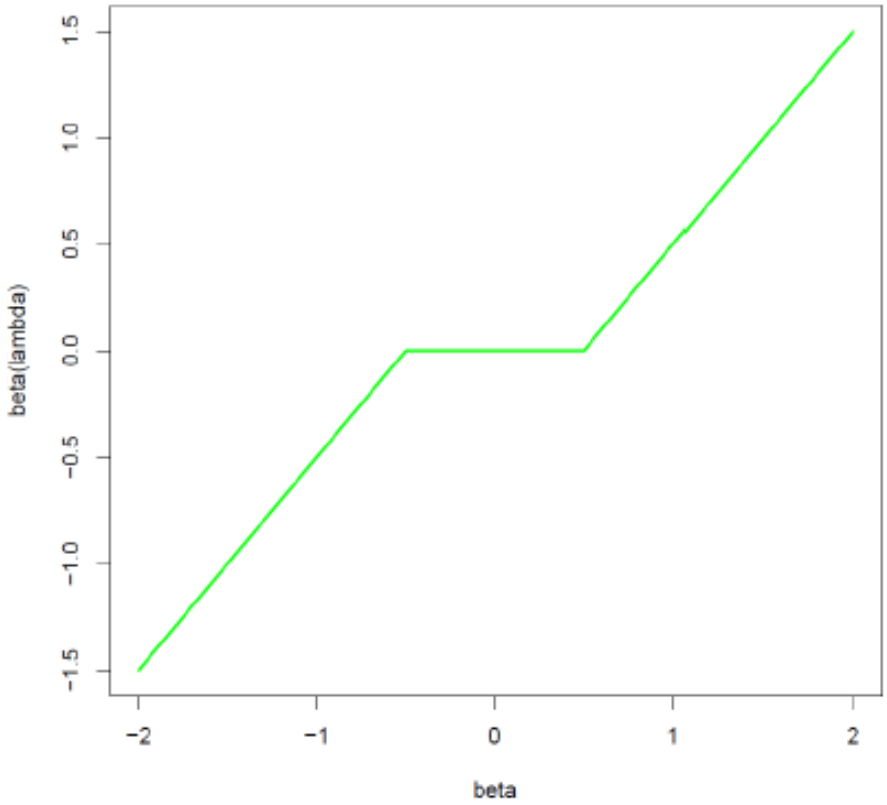
Lasso regularization path



In the orthonormal case, i.e. $\mathbf{X}^T\mathbf{X} = \mathbf{I} = (\mathbf{X}^T\mathbf{X})^{-1}$:

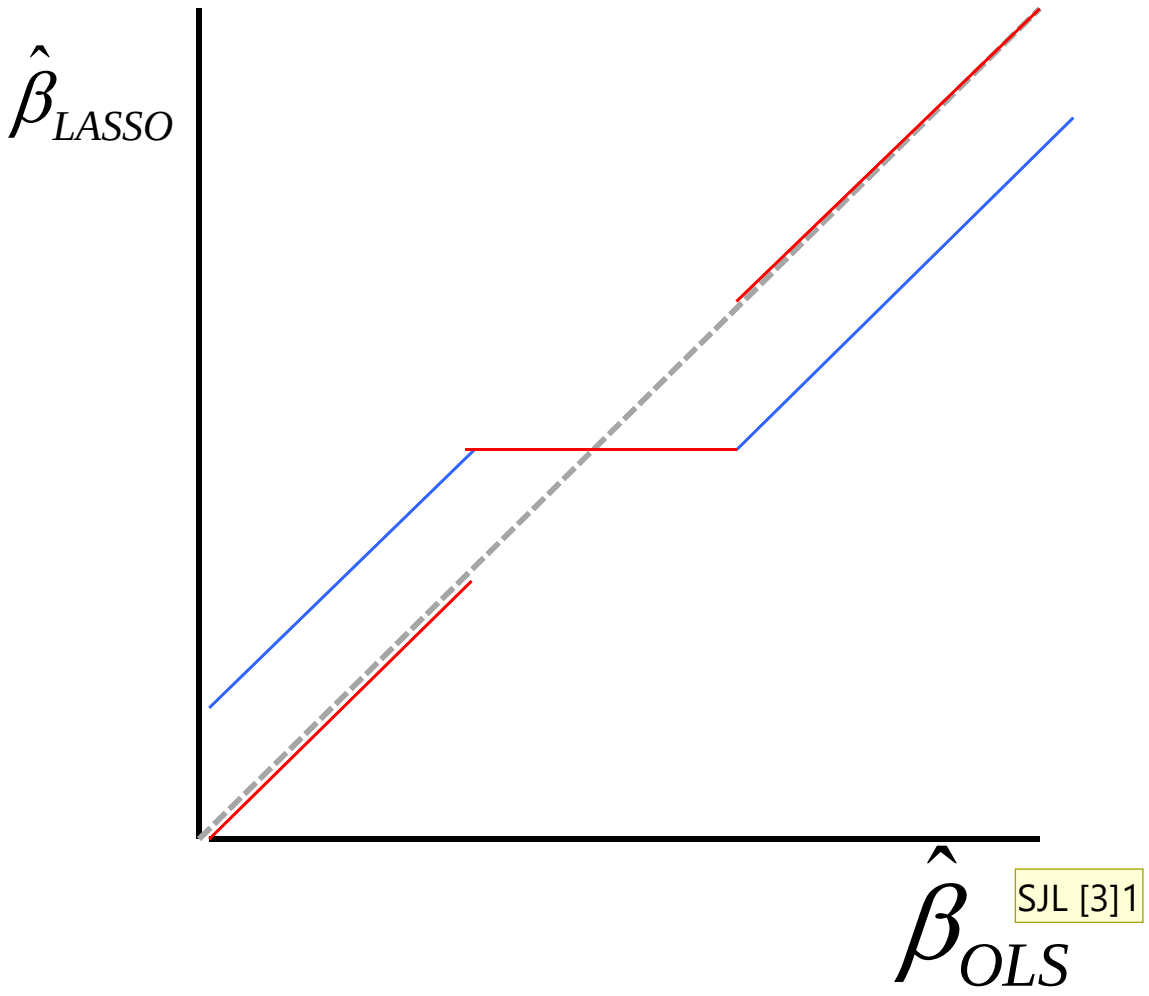
$$\hat{\beta}_j(\lambda_1) \quad = \quad \mathrm{sgn}(\hat{\beta}_j)\,(|\hat{\beta}_j| - \lambda_1/2)_+$$

That is, the lasso estimate is related to the OLS estimate via the so-called *soft threshold function* (depicted here for λ=1).



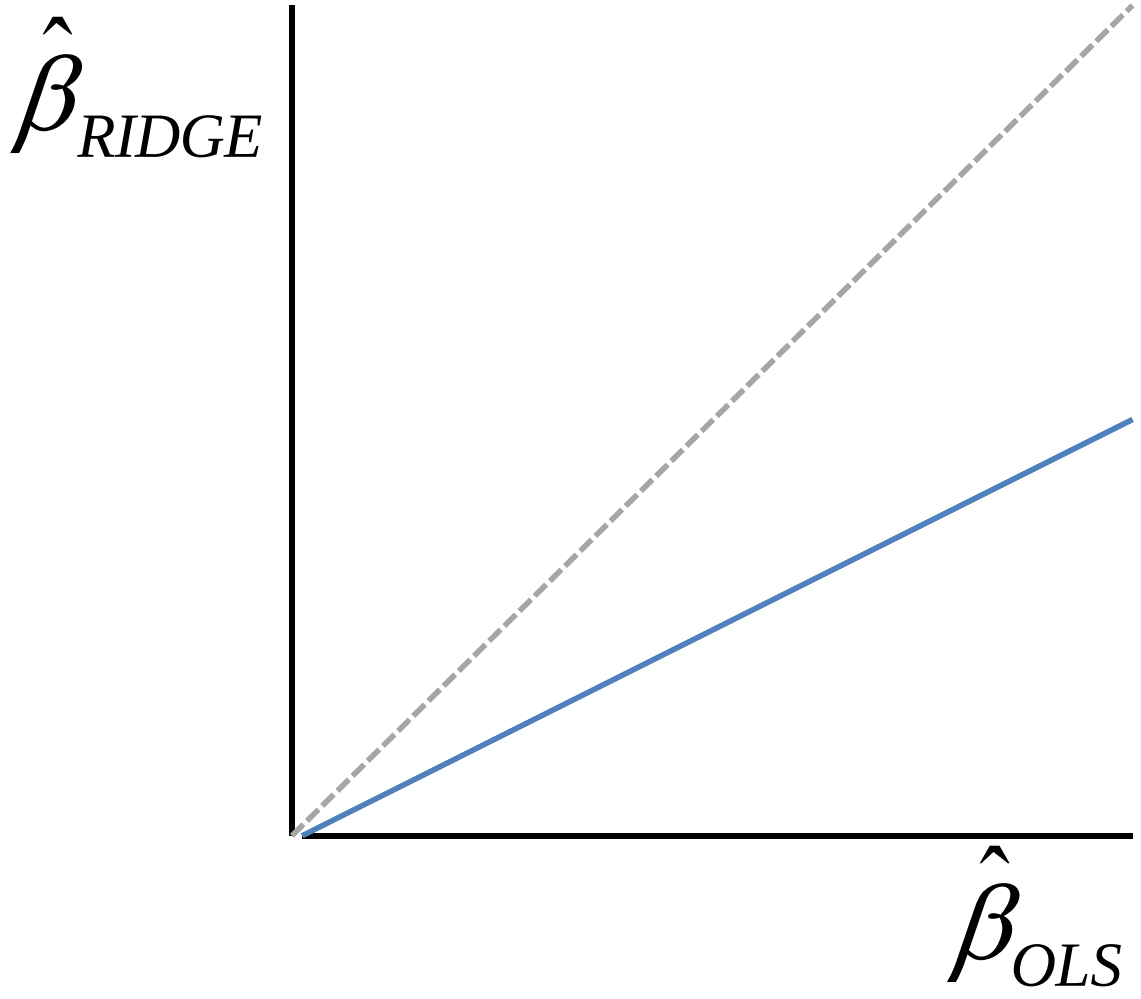
Why not Hard Thresholding?

Soft Thresholding

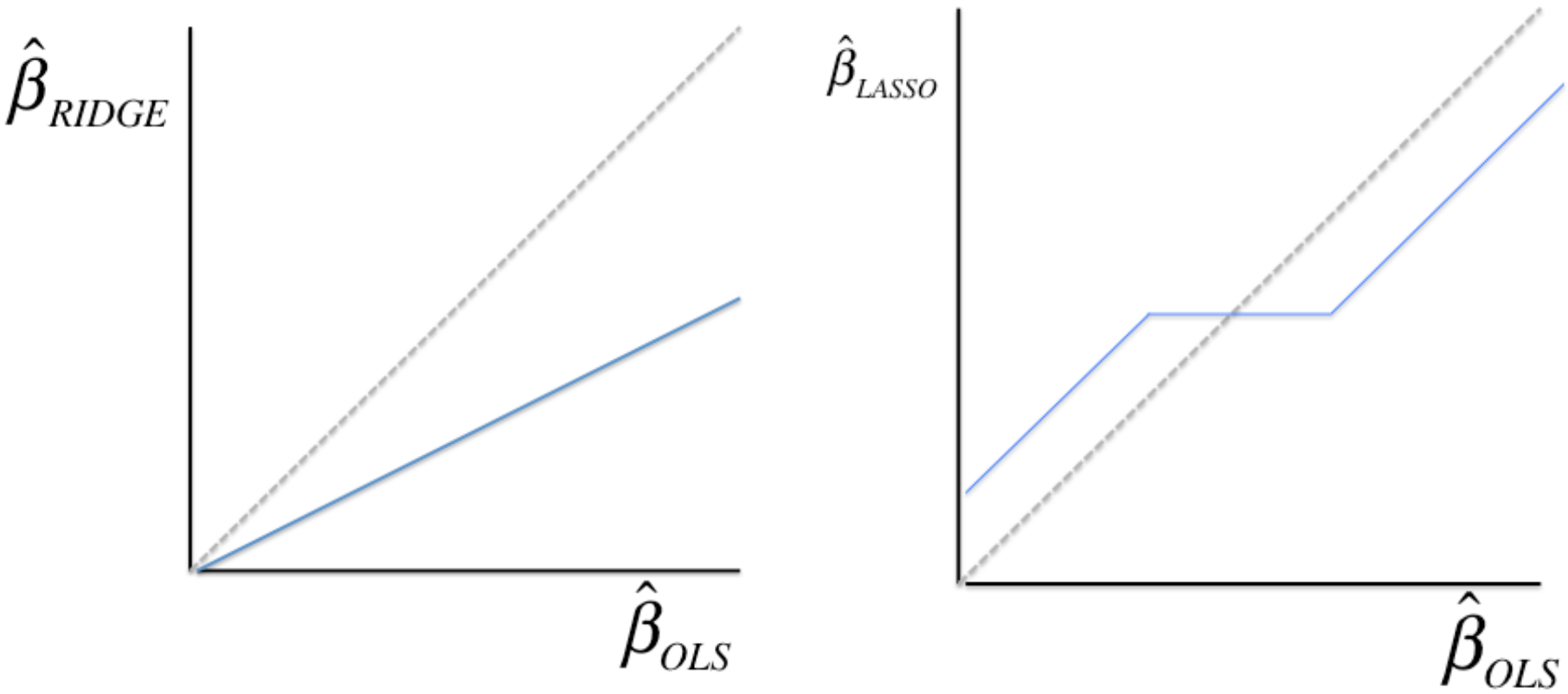


SJL [3]1

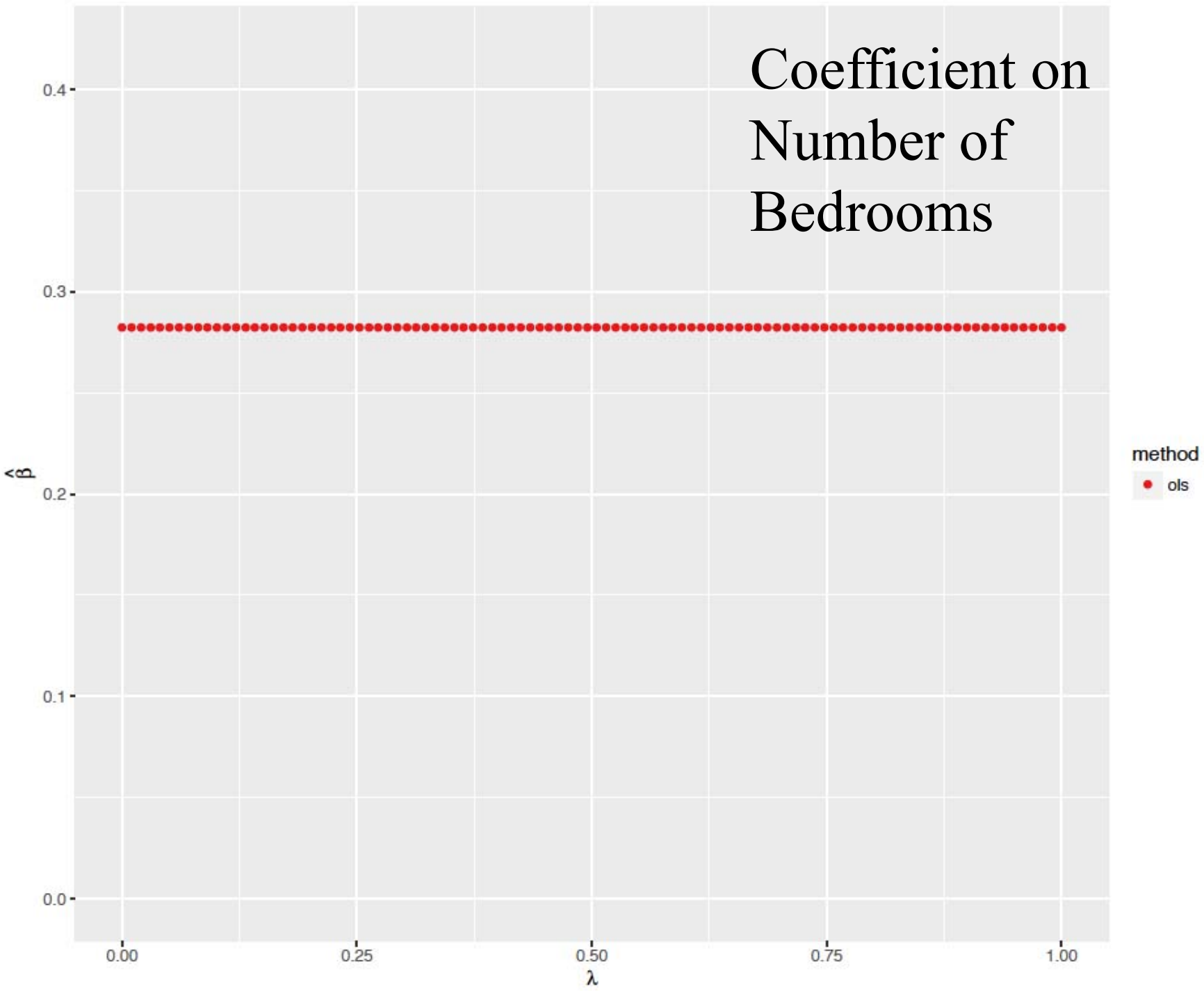
Orthonormal: $\hat{\beta}_{RIDGE} = \frac{\hat{\beta}_{OLS}}{1+\lambda}$

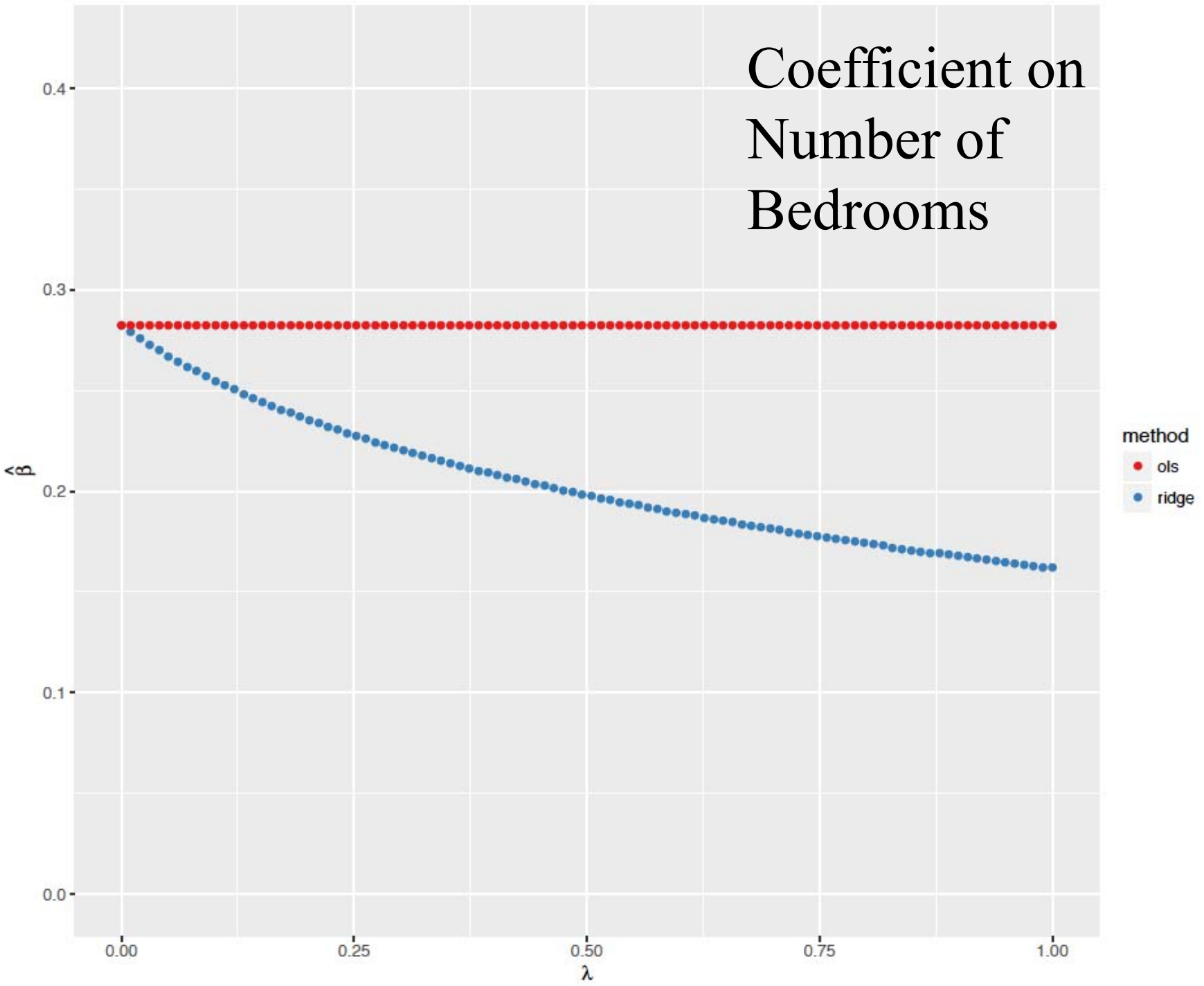


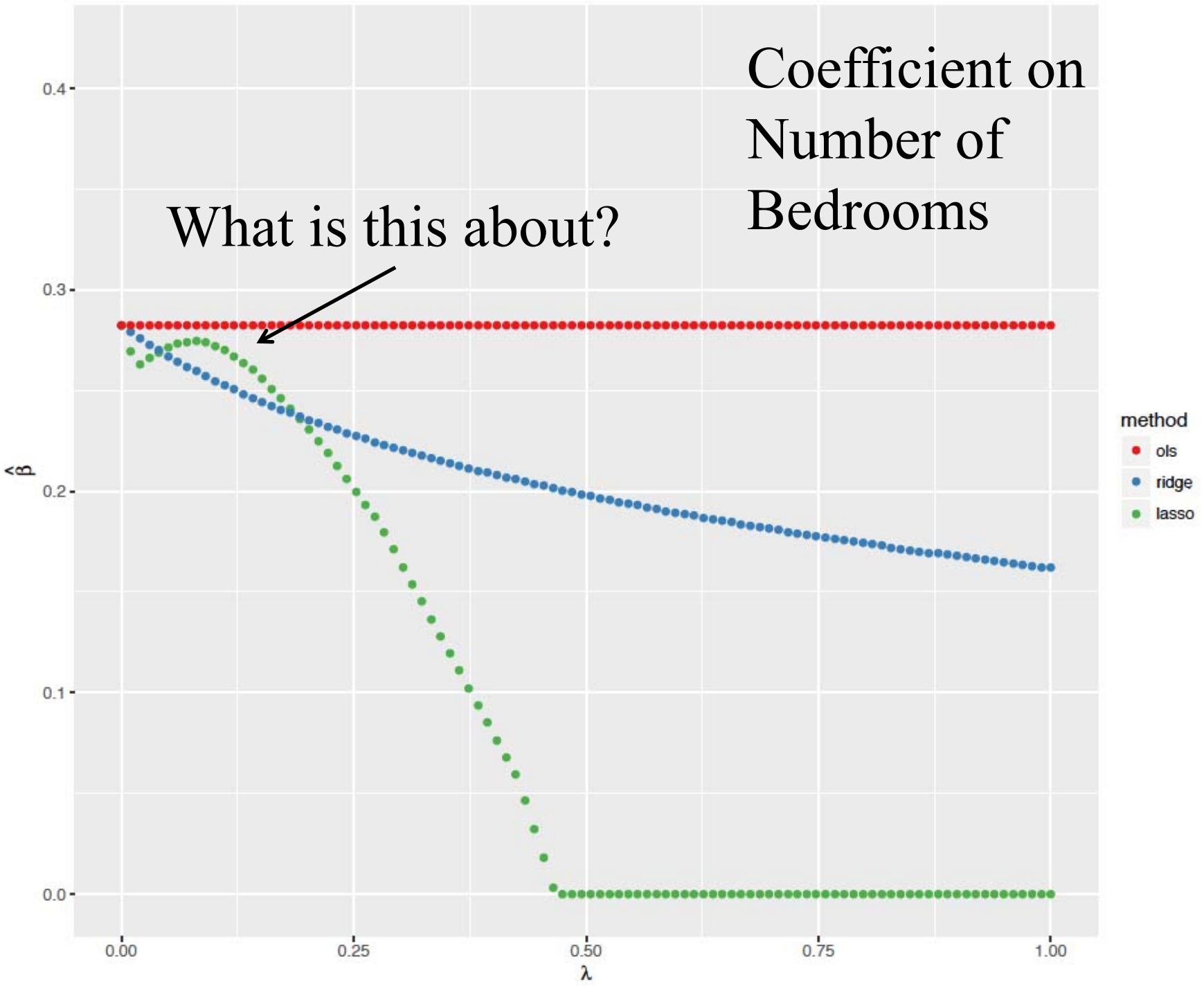
SJL [2]

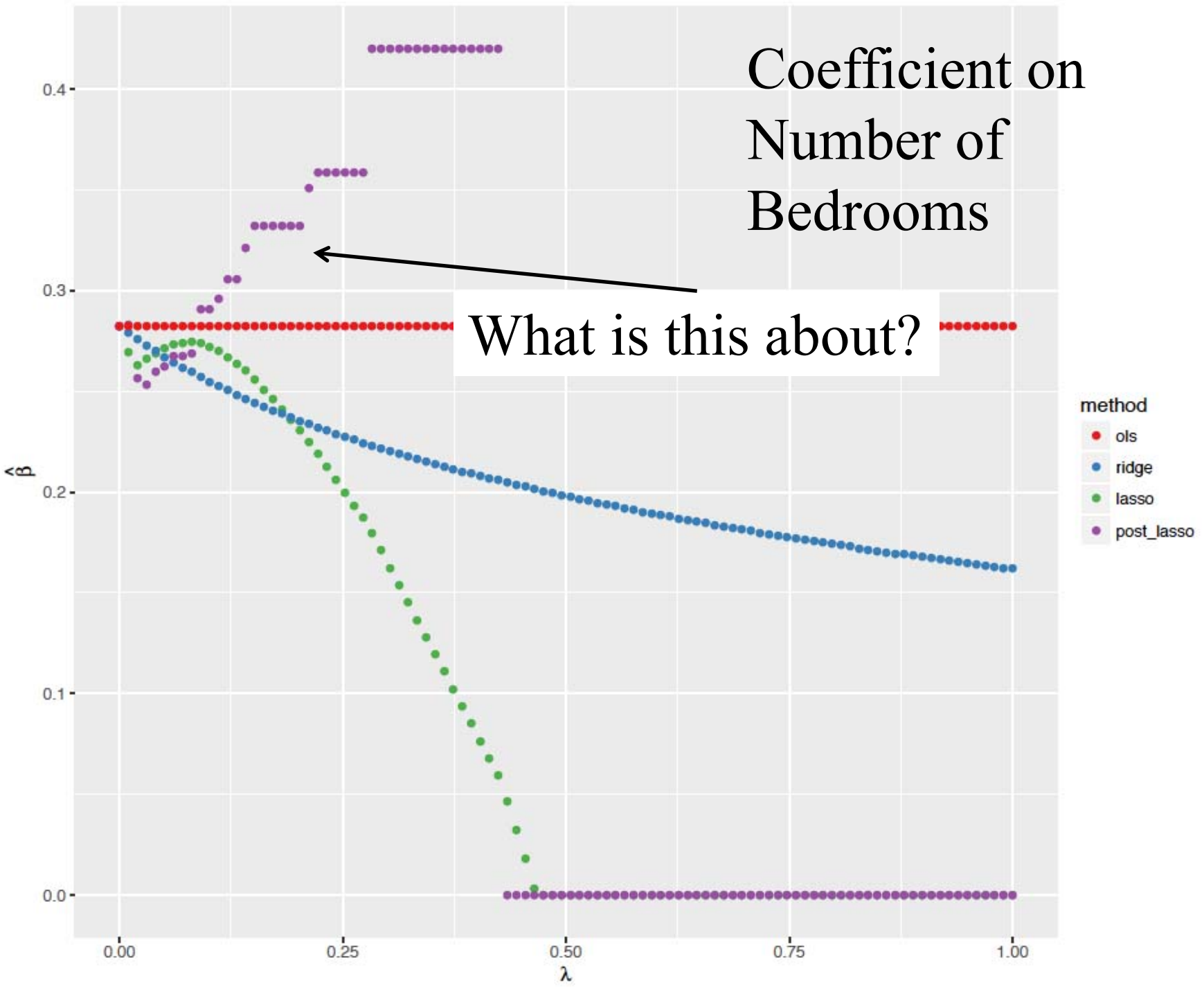


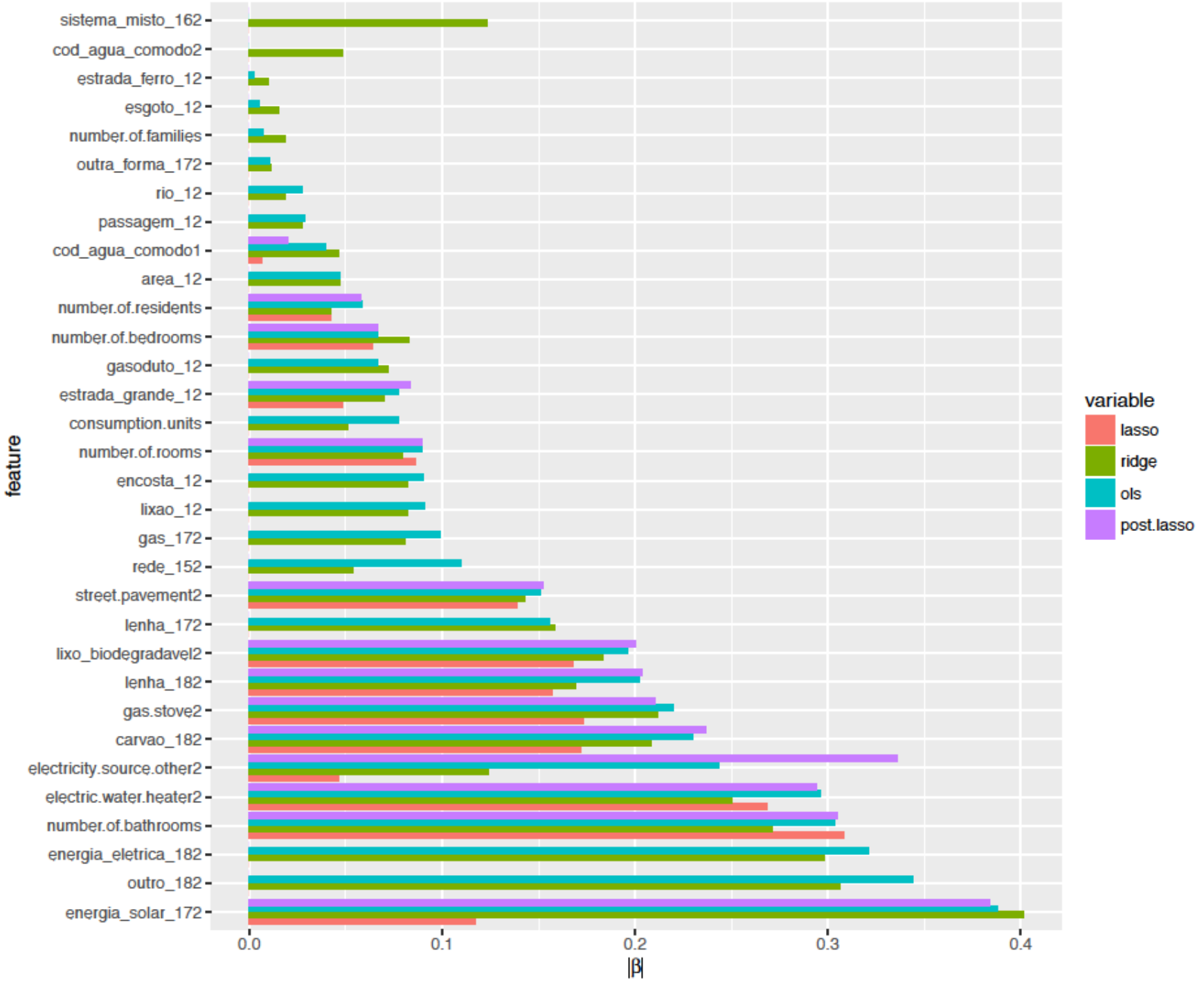
Can be very misleading


Coefficient on Number of Bedrooms
0.4
0.3
0.2
0.1
0.0
β̂
0.00
0.25
0.50
0.75
1.00
λ
method
ols


Coefficient on Number of Bedrooms
0.4
0.3
0.2
0.1
0.0
β̂
0.00
0.25
0.50
0.75
1.00
λ
method
ols
ridge


Coefficient on Number of Bedrooms
What is this about?
0.4
0.3
0.2
0.1
0.0
$\hat{\beta}$
0.00
0.25
0.50
0.75
1.00
λ
method
ols
ridge
lasso


Coefficient on Number of Bedrooms
What is this about?
0.4
0.3
0.2
0.1
0.0
$\hat{\beta}$
0.00
0.25
0.50
0.75
1.00
λ
method
ols
ridge
lasso
post_lasso


sistema_misto_162
cod_agua_comodo2
estrada_ferro_12
esgoto_12
number.of.families
outra_forma_172
rio_12
passagem_12
cod_agua_comodo1
area_12
number.of.residents
number.of.bedrooms
gasoduto_12
estrada_grande_12
consumption.units
number.of.rooms
encosta_12
lixao_12
gas_172
rede_152
street.pavement2
lenha_172
lixo_biodegradavel2
lenha_182
gas.stove2
carvao_182
electricity.source.other2
electric.water.heater2
number.of.bathrooms
energia_eletrica_182
outro_182
energia_solar_172
feature
0.0
0.1
0.2
0.3
0.4
|β|
variable
lasso
ridge
ols
post.lasso