Cross Validation

Lecture 6

Connor Dowd

April 15th, 2021

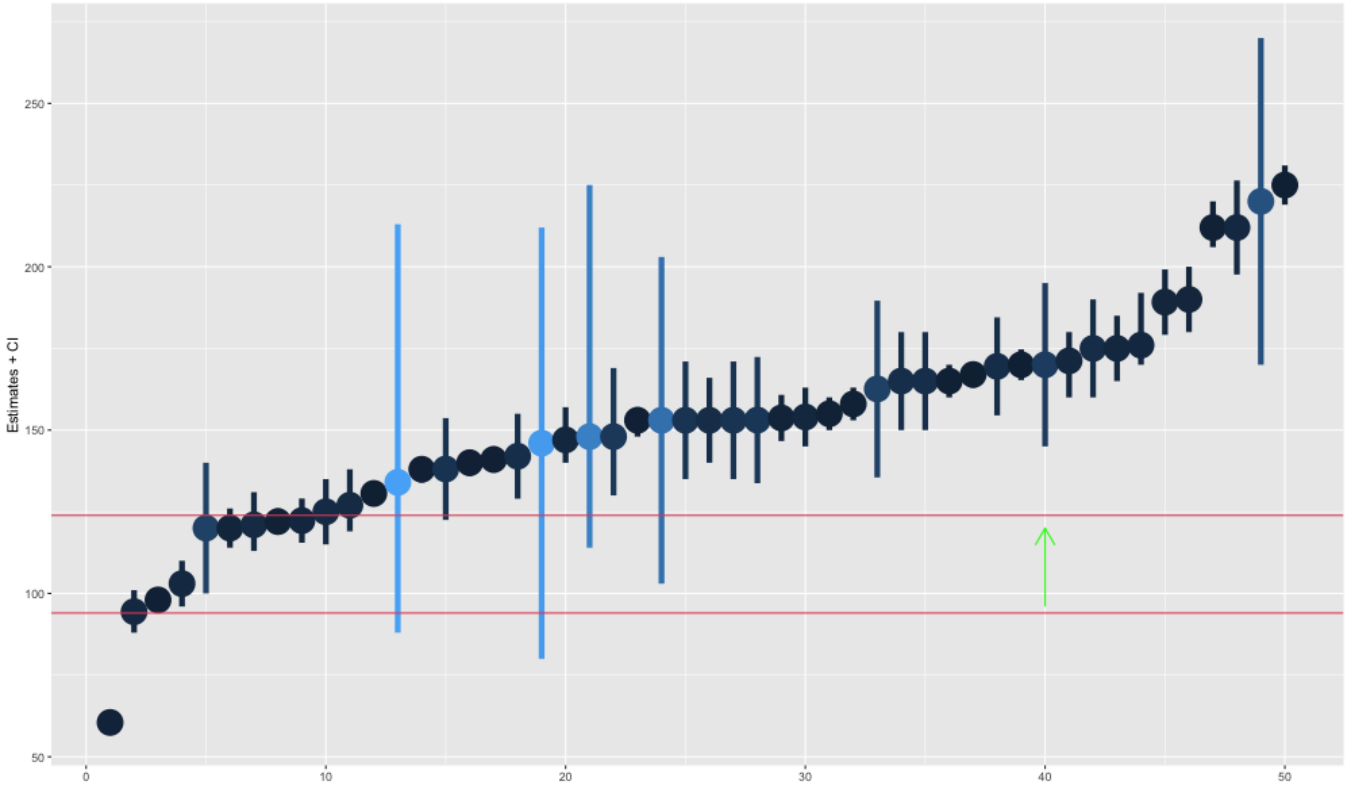
Today's Class

1. Quick Review
   - Out of Sample Performance $\neq$ In sample Performance
   - AIC
   - Stepwise
   - LASSO
2. Cross Validation
3. Bias-Variance
4. Computational Notes
   - Cross Validation
   - Sparse Matrices
   - LASSO with Factors
5. HW Discussion:
   - HW2 Solutions
   - HW3 setup

Quick Review

Predictions

About halfway through!



OOS performance

In sample, adding variables always improves predictive accuracy.

- IS R^2 for full model (200 vars): 56%
- IS R^2 for FDR cut model (25 vars): 18%

Out of Sample, we may gain predictive accuracy by dropping variables.

- OOS R^2 for the full model: -5.87.
- OOS R^2 for the cut model: 0.10.

THIS IS NEGATIVE. Out-of-Sample, we would be much **BETTER** off predicting the mean than with the full model.

OOS R^2 estimated with k-fold cross-validation.

AIC

We can (with some assumptions) estimate OOS deviance using Akaike's Information Criterion. For a model M with k variables estimated to be $\hat{\beta}_M$

$$AIC = 2k + Dev(\hat{\beta}_M) = 2k - 2log(L(\hat{\beta}_M|data))$$

This was in our basic model output.

```
(Dispersion parameter for gaussian family taken to be 0.4829706)

    Null deviance: 30079  on 28946  degrees of freedom
Residual deviance: 13975  on 28935  degrees of freedom
AIC: 61094
```

This can help us compare models to choose one.

Stepwise

But there are too many possible subsets of our variables to compare all of them. So we need some other method for coming up with a subset to compare.

Stepwise regression does this. It starts with a simple model, and "greedily" adds the best new variable repeatedly until adding a new variable no longer improves the AIC.

Because each choice depends on the current model parameters, small changes to the data can have big consequences for our model choices. This instability is bad for our OOS prediction.

But AIC estimates OOS deviance/prediction errors

AIC estimates OOS deviance, and it does a good job of it, *for a given model*.

But once we started using AIC to choose our models, it ceased to be a good estimate of our deviance.

AIC was estimating the prediction errors of one model, not of a whole procedure which picks a model.

"When a measure becomes a target, it ceases to be a good measure" – Goodhart's Law

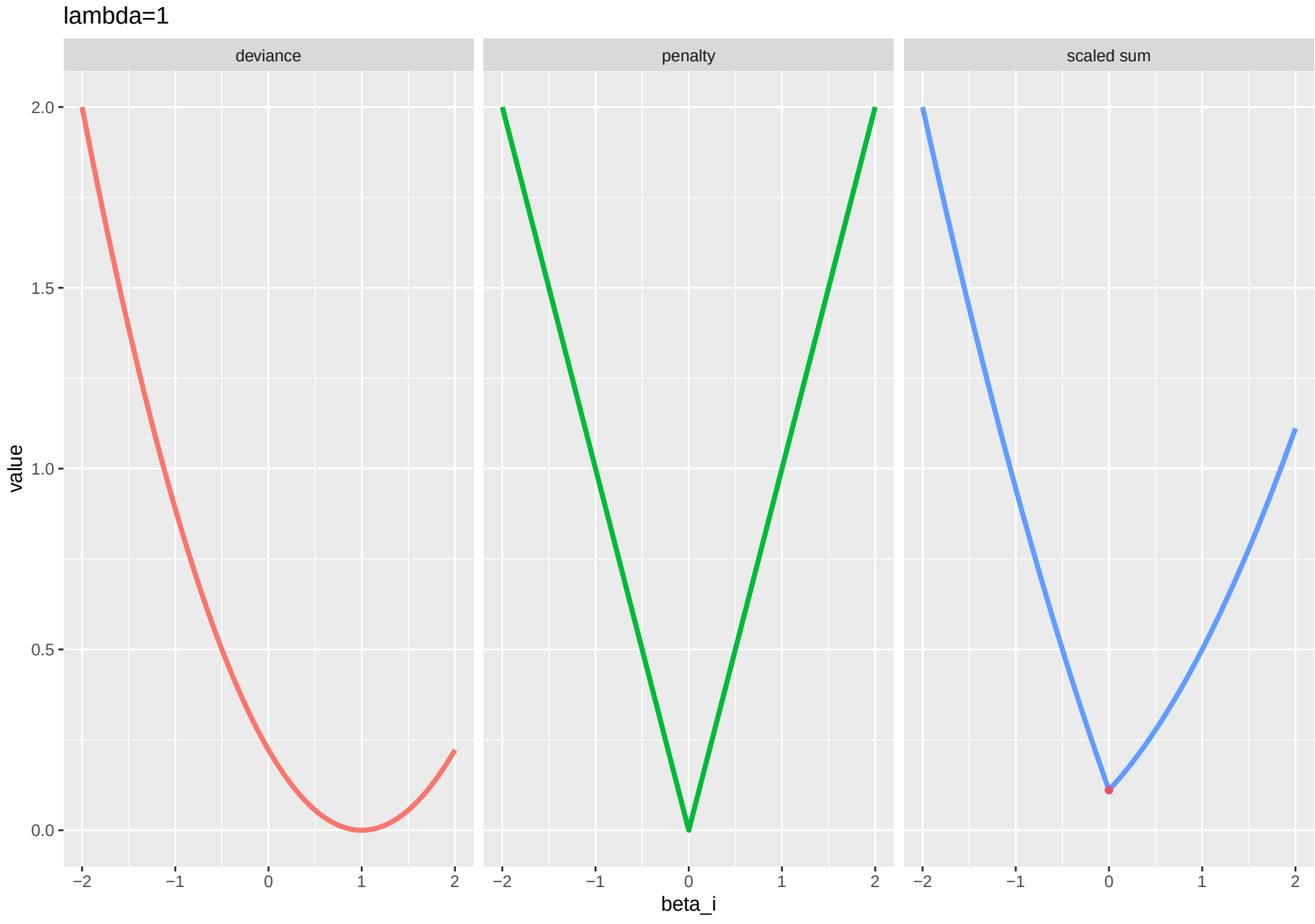
We may encounter this problem again.

LASSO

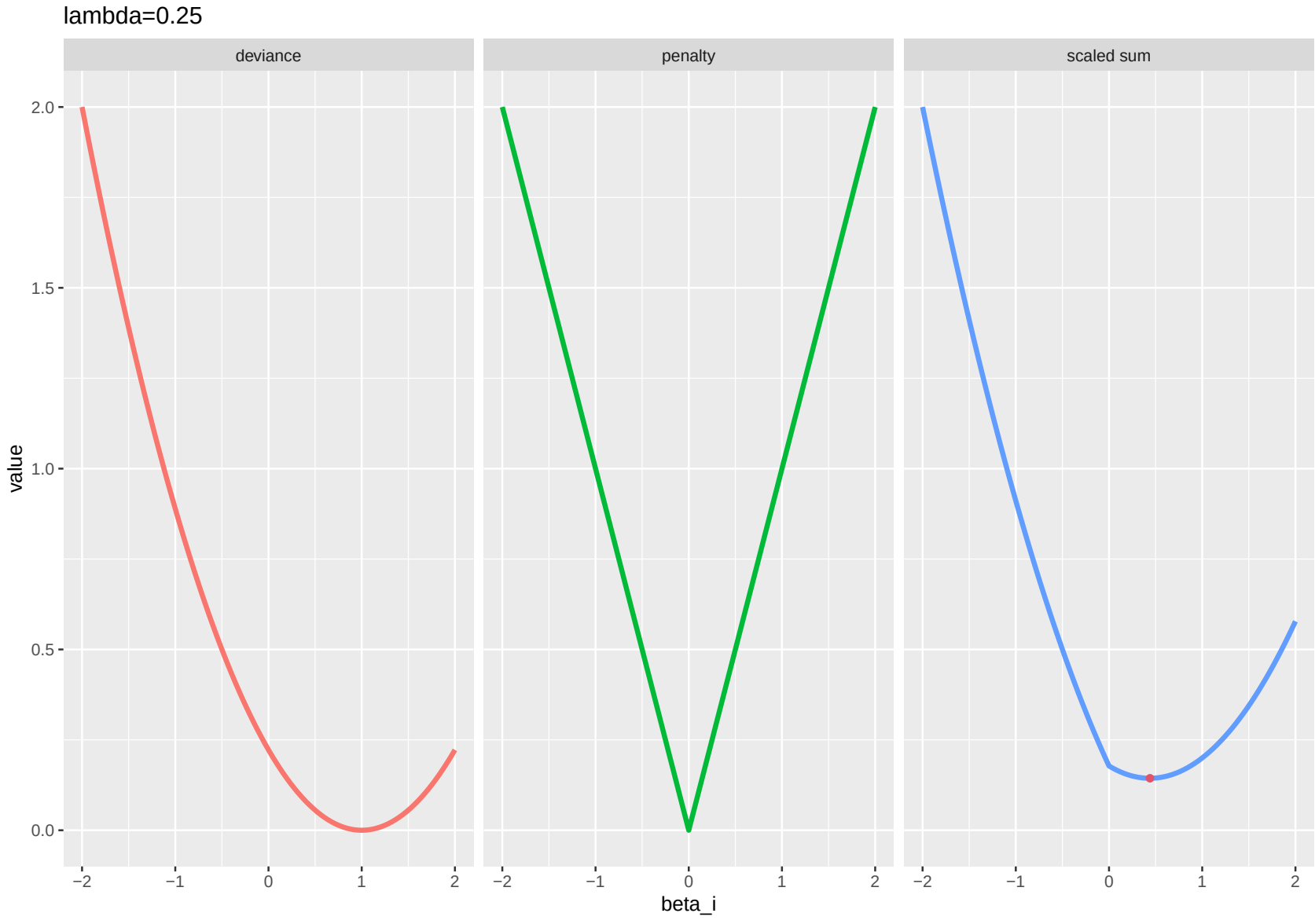
LASSO is the most commonly used regularized (or penalized) regression model. The lasso penalty is the l_1 norm of our parameters: $pen(\beta) = \sum |\beta_j|$, which penalizes larger coefficients and non-zero coefficients. So our estimates are:

$$\hat{\beta}_\lambda = \underset{\beta}{\text{argmin}} \left(Dev(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

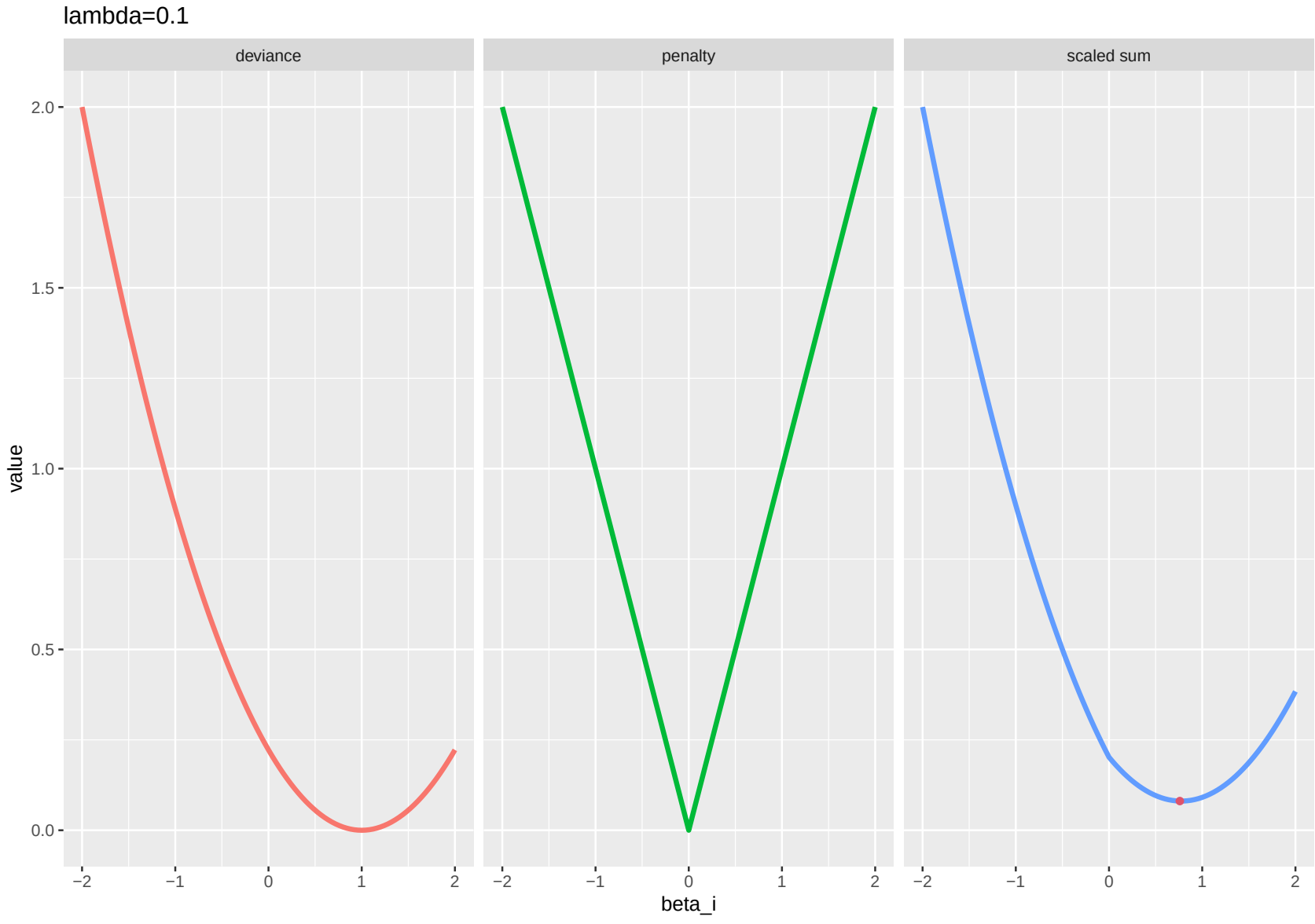
LASSO


lambda=1
deviance
penalty
scaled sum
value
beta_i
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2

LASSO


lambda=0.25
deviance
penalty
scaled sum
value
beta_i
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2

LASSO


lambda=0.1
deviance
penalty
scaled sum
value
2.0
1.5
1.0
0.5
0.0
−2
−1
0
1
2
beta_i

LASSO

We've framed LASSO as the solution to an optimization, given some weight parameter λ.

$$\hat{\beta}_{LASSO} = \underset{\beta}{\text{argmin}} Dev(\beta) + \lambda \sum |\beta_j|$$

It may be important, for reference to other places, to know that this can be rewritten as:

$$\hat{\beta}_{LASSO} = \underset{\beta}{\text{argmin}} Dev(\beta) \quad s.t. \quad \sum |\beta_j| = ||\beta||_1 \leq t$$

Where the constraint t is a bound on our l_1 norm. For any given dataset, there is a correspondence between λ and t. So these representations are different ways of looking at the same problem.

Cross Validation

Recap

We could, like the bootstrap, repeat our testing-training procedure many times. But we want to guarantee each observation gets used as an 'out-of-sample' observation at least once.

Instead, we will "fold" the data. We will partition it (split into exclusive and exhaustive groups) into K different groups of observations. Then, `for k=1:K`

- Use observations in group k as test data.
- Train the models on the remaining data (for every λ)
- Predict the observations in group k using those models
- Record the prediction errors for each lambda

This guarantees that each observation is left out once, and improves the performance of our routine.

Picking K

There are several options:

- Leave-one-out Cross-validation: AKA $K = n$ is great, but much slower (fits every model under consideration n times)
- $K = 5$ corresponds to 5 different 20% leave-out samples.
- $K = 20$ corresponds to 20 different 5% leave-out samples.

Most people set $K \in [5, 20]$. I'll mostly use 10.

$\implies$ Optimizing K is very 3rd order. Not worth worrying about too much beyond time considerations and some preference for larger K.

All Together.

We have a LASSO path indexed by $\lambda_1 < \lambda_2 < ... < \lambda_T$.

Cross-Validation for λ:
For each of $k = 1, ..., K$ folds:
1. Fit the path $\hat{\beta}^k_{\lambda_1}, ..., \hat{\beta}^k_{\lambda_T}$ using the data not in fold k. 2. Get the fitted deviance for new data: $-log\ P[y^k | X^k, \hat{\beta}^k_{\lambda_t}]$ where k denotes fold membership.

This gives us K draws of the OOS deviance for each λ_t.

Choose the best $\hat{\lambda}$, and fit your model to all the data with that $\hat{\lambda}$.

New Example: Comscore

This is from Comscore data. This data is about consumer spending on websites. We will try to predict household internet spending as a function of browser history.

Datanames:

- Covariates X: `xweb`
- outcomes Y: `yspend`

In R

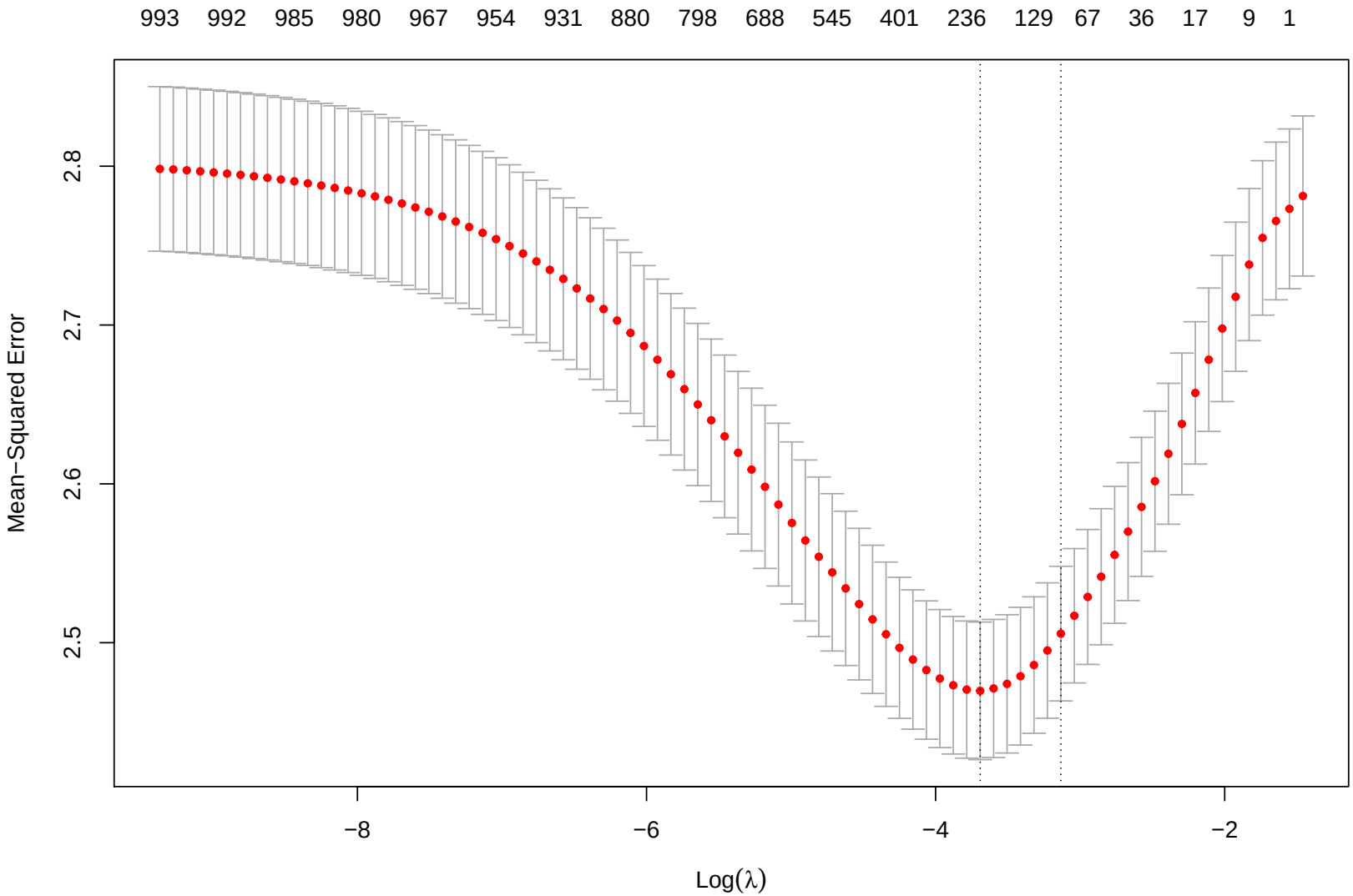
Again, Cross-validation is very easy and relatively fast for LASSO in R.

```
cv.spender = cv.glmnet(xweb,yspend)
```

And there is a nice plot for it too

CV LASSO plot

```
plot(cv.spender)
```



Mechanically What is Happening?

`for (k in 1:K)`

1. Estimate model on all data not in fold `k`
2. Calculate that model's OOS prediction errors with data in fold `k`
3. Repeat for all models under consideration

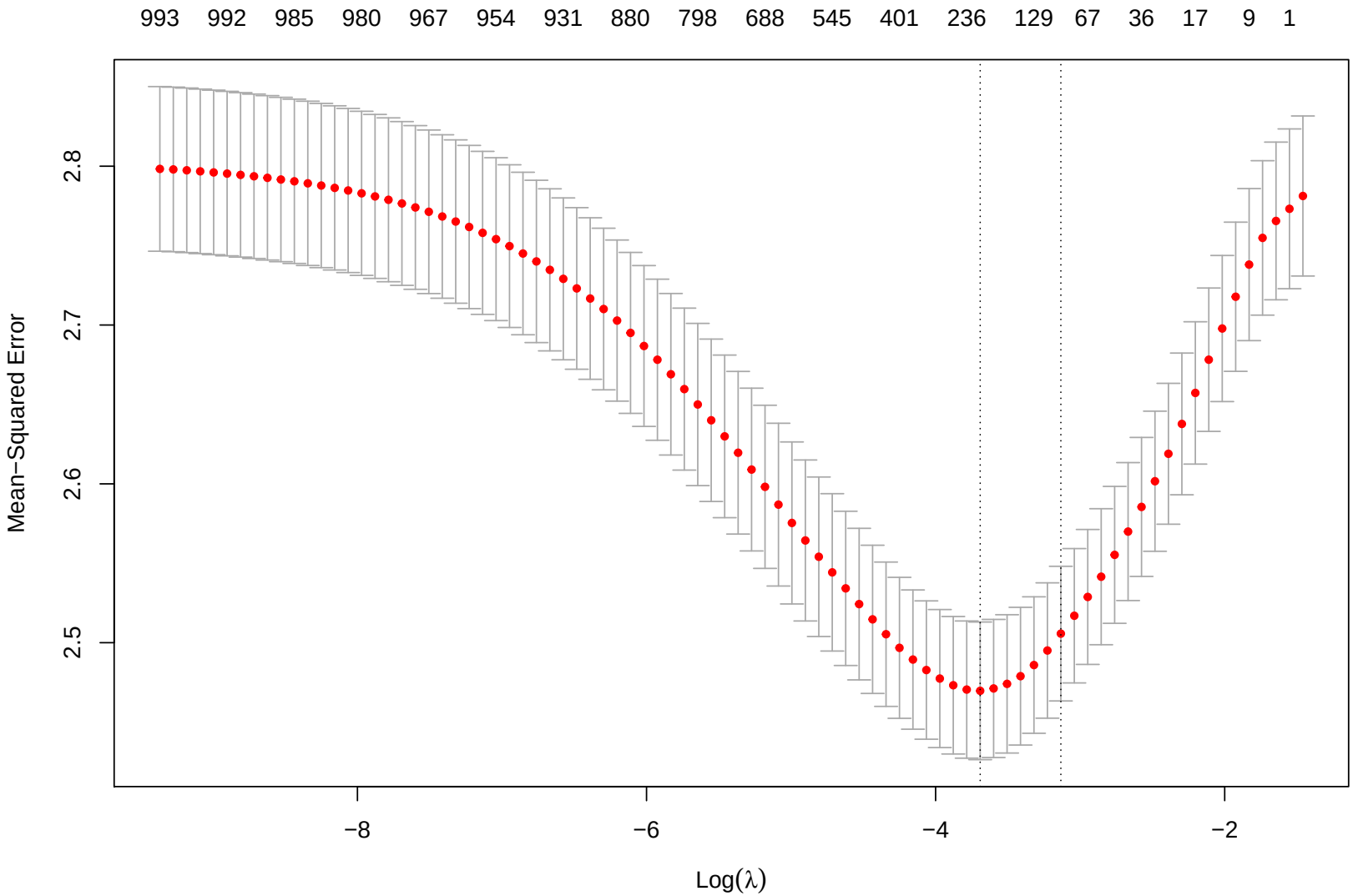
So we have `K` estimates of deviance (MSE here) for each model. Now:

1. Estimate overall/mean deviance for each model.
2. Estimate our estimation error (standard errors) in that deviance.
3. Build CIs for Deviance

That plot shows the estimated mean/overall deviance for each lambda (across `K`) as a red point, and it shows the error bars on that estimate.

CV LASSO plot

```
plot(cv.spender)
```



LASSO Outputs

Once we've done the cross-validation its easy to pick out our coefficients.

```
sum(coef(cv.spender,s="lambda.1se") != 0)
```

```
## [1] 101
```

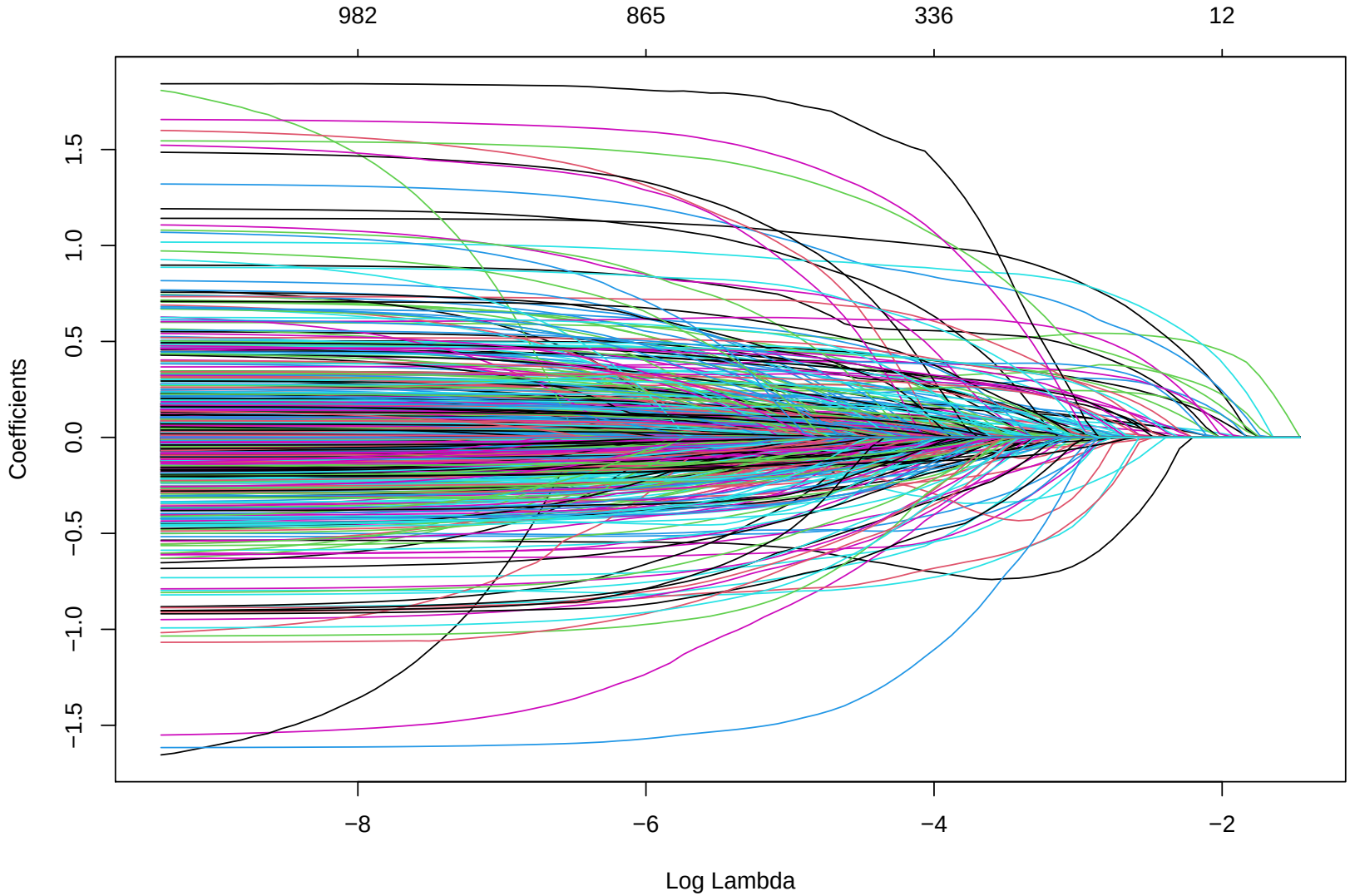
```
sum(coef(cv.spender,s="lambda.min") != 0)
```

```
## [1] 207
```

"Min" vs "1se" is to do with a concern about overfitting. The 1SE model is the smallest model that has predictions which perform very similarly to the true model.

LASSO Outputs

Recall, small changes in our choice don't have a massive effect on our coefficients, which evolve smoothly with changes in lambda.



Cross-Validation: A general technique

Cross-validation will be a general technique that we use for model selection the rest of this course.

The central element is that we are being careful about what data we use to train our model and what data we use to select our model. We **do not** use the same data for both.

K-fold cross validation lets us use our data maximally efficiently, while ensuring we maintain that separation between 'training' and 'testing' data.

Cross-validation: Problems

K-fold Cross validation as laid out above is not perfect. It has flaws.

1. The observations must be independent. Time-series or clustering will require different (related) approaches

Cross-validation: Problems

K-fold Cross validation as laid out above is not perfect. It has flaws.

1. The observations must be independent. Time-series or clustering will require different (related) approaches
2. If we want to estimate the error distributions of the entire procedure (including CV), we need a new test sample that was held out from the entire CV model selection procedure. (See Goodhart's law again)

Cross-validation: Problems

K-fold Cross validation as laid out above is not perfect. It has flaws.

1. The observations must be independent. Time-series or clustering will require different (related) approaches
2. If we want to estimate the error distributions of the entire procedure (including CV), we need a new test sample that was held out from the entire CV model selection procedure. (See Goodhart's law again)
3. Using deviances as our measure of prediction error only works within classes of models that have the same likelihood function.

Cross-validation: Problems

K-fold Cross validation as laid out above is not perfect. It has flaws.

1. The observations must be independent. Time-series or clustering will require different (related) approaches
2. If we want to estimate the error distributions of the entire procedure (including CV), we need a new test sample that was held out from the entire CV model selection procedure. (See Goodhart's law again)
3. Using deviances as our measure of prediction error only works within classes of models that have the same likelihood function.
   - Remember that the deviance has a constant C we've been ignoring? If you change your error distribution (e.g. from Normal to Logistic), that constant changes

Cross-validation: Problems

K-fold Cross validation as laid out above is not perfect. It has flaws.

1. The observations must be independent. Time-series or clustering will require different (related) approaches
2. If we want to estimate the error distributions of the entire procedure (including CV), we need a new test sample that was held out from the entire CV model selection procedure. (See Goodhart's law again)
3. Using deviances as our measure of prediction error only works within classes of models that have the same likelihood function.
   - Remember that the deviance has a constant C we've been ignoring? If you change your error distribution (e.g. from Normal to Logistic), that constant changes
   - E.g. comparisons between Linear probability models (using `lm` for binary outcomes) and logits (using `glm` family=binomial) require care.

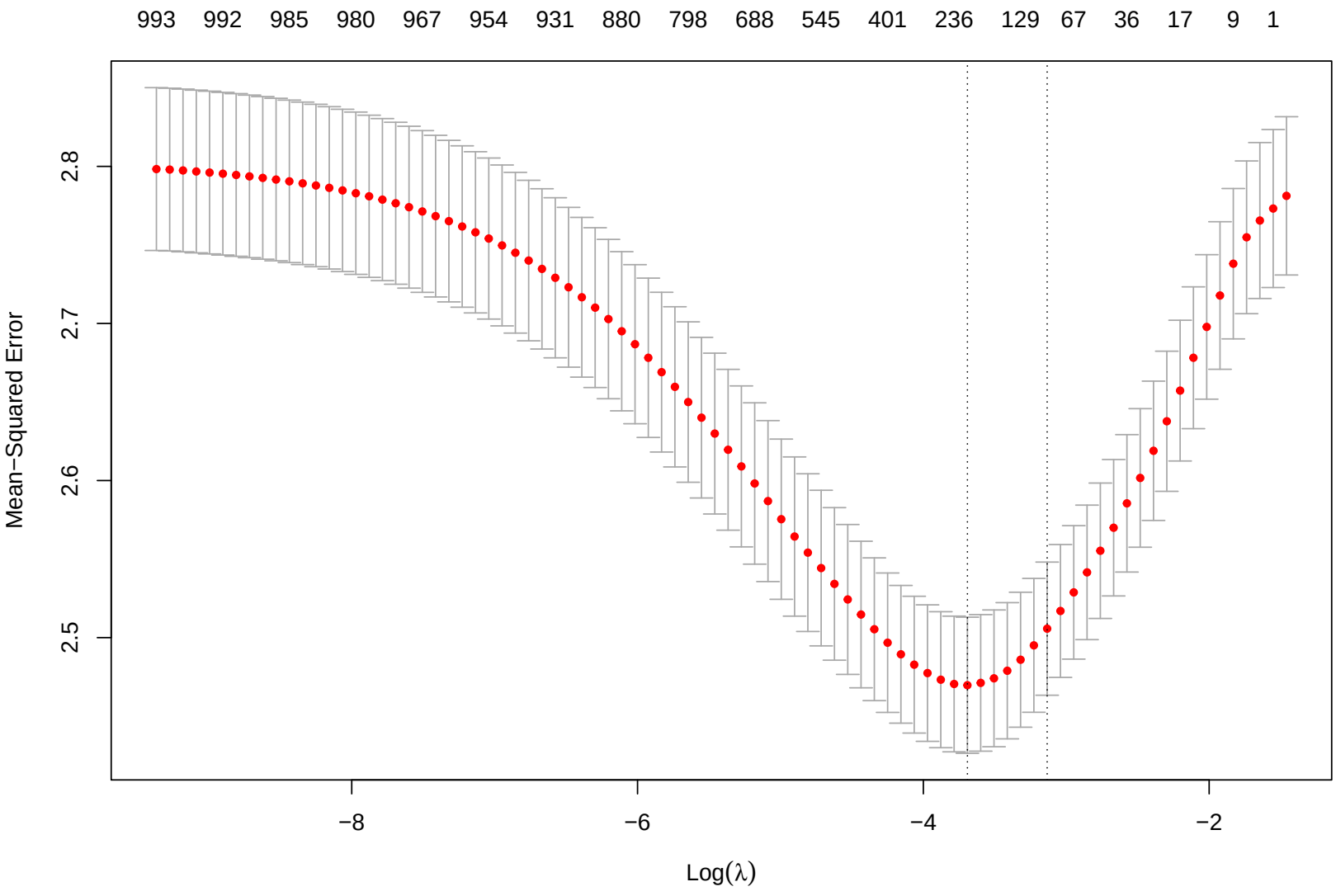
Cross-validation: Problems

K-fold Cross validation as laid out above is not perfect. It has flaws.

1. The observations must be independent. Time-series or clustering will require different (related) approaches
2. If we want to estimate the error distributions of the entire procedure (including CV), we need a new test sample that was held out from the entire CV model selection procedure. (See Goodhart's law again)
3. Using deviances as our measure of prediction error only works within classes of models that have the same likelihood function.
   - Remember that the deviance has a constant C we've been ignoring? If you change your error distribution (e.g. from Normal to Logistic), that constant changes
   - E.g. comparisons between Linear probability models (using `lm` for binary outcomes) and logits (using `glm` family=binomial) require care.
   - Not insurmountable – you just need to think about what your loss function actually **is**. The issue here is that there are two

Bias-Variance Tradeoff

Mean Squared Error



Mean Squared Error

The Y axis was "Mean Squared Error". This is defined as:

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

The interior of that square can be rewritten:

$$y - \hat{y} = (y - E[y]) + (E[y] - \hat{y}) = \epsilon + Err(\hat{y})$$

So the interior of the sum can be:

$$\implies (y - \hat{y})^2 = (\epsilon + Err(\hat{y}))^2 = \epsilon^2 + 2\epsilon Err(\hat{y}) + Err(\hat{y})^2$$

Thus the expected MSE:

$$E[MSE] = E[(y - \hat{y})^2] = E[\epsilon^2] + E[2\epsilon Err(\hat{y})] + E[Err(\hat{y})^2] =$$

$$= Var(\epsilon) + 0 + E[Err(\hat{y})^2]$$

Bias-Variance Decomposition

$Var(\epsilon) = \sigma^2$ is irreducible. So we want to minimize $E[Err(\hat{y})^2]$.

$$Err(\hat{y}) = E[y] - \hat{y} = (E[y] - E[\hat{y}]) + (E[\hat{y}] - \hat{y})$$

For any given model, M, (i.e. a linear model with some covariates):

$$B(M) = E[y] - E[\hat{y}] \quad \& \quad Var(M) = E[\hat{y}] - \hat{y}$$

`B(M)` is a measure of how correct `M` can be. This is the difference between your model with optimal parameters and the true model. A Bias.

`Var(M)` is a measure of how well we can estimate the model. How close to the optimal parameters do we get?

$$\implies E[MSE] = \sigma^2 + B(M) + Var(M)$$

Bias-Variance Decomposition

- As we make our models more flexible (add parameters), the Bias of the optimal parameters goes down.

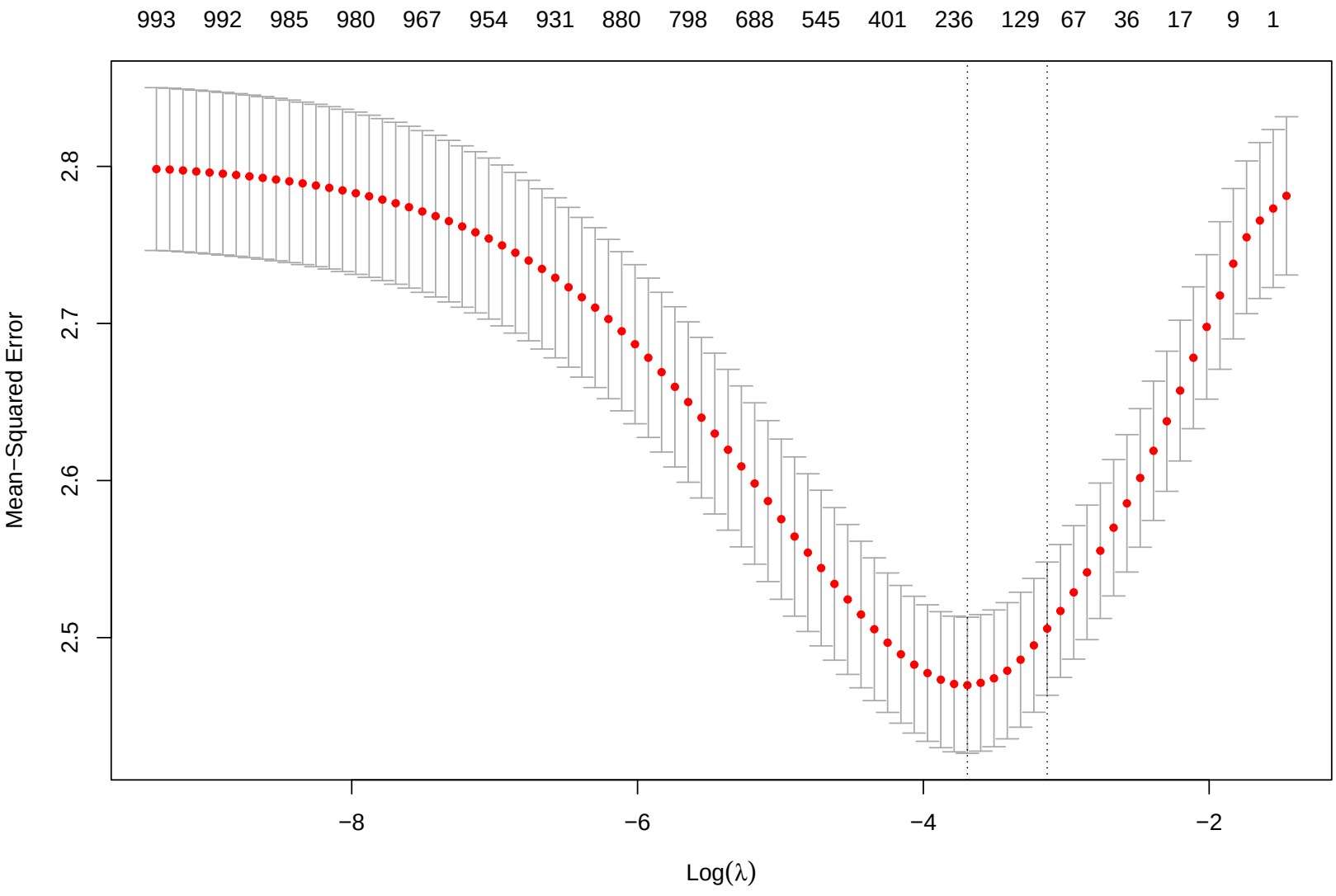
Bias-Variance Decomposition

- As we make our models more flexible (add parameters), the Bias of the optimal parameters goes down.
- But at the same time, we have to estimate more, which is challenging, so we increase the distance between our estimated model, and the optimal parameters for that model.

Bias-Variance Decomposition

- As we make our models more flexible (add parameters), the Bias of the optimal parameters goes down.
- But at the same time, we have to estimate more, which is challenging, so we increase the distance between our estimated model, and the optimal parameters for that model.
- This is why we see a U-shape in most Cross-Validation plots measuring out of sample performance. With too few parameters, we have a lot of model bias. With too many parameters, we have a lot of model variance.

CV LASSO Plot



Computational Notes

Brief Outline

We're going to cover some in the weeds details, with example R code.

- LASSO & Factors
- Cross Validation Example Code
- Sparse Matrices

LASSO & Factors

We've discussed LASSO at some length. It shrinks every parameter towards 0.

- This can be an issue for factors.
  - Reference level gets different treatment than other levels
  - Not a major issue. But worth addressing
  - Solution: Relevel factors so that the first level is irrelevant.
- Also for Fixed Effects:
  - We want to fully control for some group membership.
  - Shrinking would be remove that control.
  - Solution: Don't penalize FE params.

LASSO FEs in R

Instead of

```
mod = glmnet(xweb,yspend)
```

Multiply the lambda penalty by 0 for coefficients that should stay in the model:

```
pen.multiplier = rep(1,ncol(xweb)) #Vector of 1s
pen.multiplier[10:20] = 0          #Replace some with 0
mod = glmnet(xweb,yspend,penalty.factor=pen.multiplier)
```

Now those parameters won't be shrunk at all.

- Mostly useful for causal models.

LASSO Factors

Make `NA` the reference level for Factors.

```
# oj$brand  is our factor variable.
oj$brand = factor(oj$brand,levels=c(NA,levels(oj$brand)),
                           exclude=NULL)
```

This will shrink every factor level evenly when you use a LASSO with `demo$factor`.

Cross-Validation in R – By hand

Setup

```
n = nrow(xweb) #Number of observations
K = 10         #Number of folds
fold_id = sample(rep(1:K,length.out=n),n,replace=F) #Rando
lambda_seq = spender$lambda #Grab the lambdas glmnet likes
```

Cross-Validation in R – By hand

Function for estimating prediction error given a sequence λ and fold ID.

```
pred_err_fun = function(k,lambdas) {
  leaveout_indices = which(fold_id==k) #Find the fold
  model = glmnet(xweb[-leaveout_indices,],
                 log(yspend[-leaveout_indices]),
                 lambda=lambdas)  #Train
  preds = predict(model,newx=xweb[leaveout_indices,]) #Pre
  nlambda = length(lambdas)#Each col of preds is preds for
  truth = log(yspend[leaveout_indices])  #True outcomes
  out = numeric(nlambda) #Initialize
  for (t in 1:nlambda) { #For each lambda
    out[t] = mean((preds[,t]-truth)^2) #Save the squared e
  }
  out #Return vector MSE for each lambda.
}
```

Cross-Validation in R – By hand

Run function on each fold ID.

```
cv_mse = sapply(1:K,pred_err_fun,lambdas=lambda_seq)
```

Each column is one fold, each row is one lambda. Entries are the OOS MSE with that fold/lambda combo. Lets get the mean and sd of the OOS MSE for each lambda.

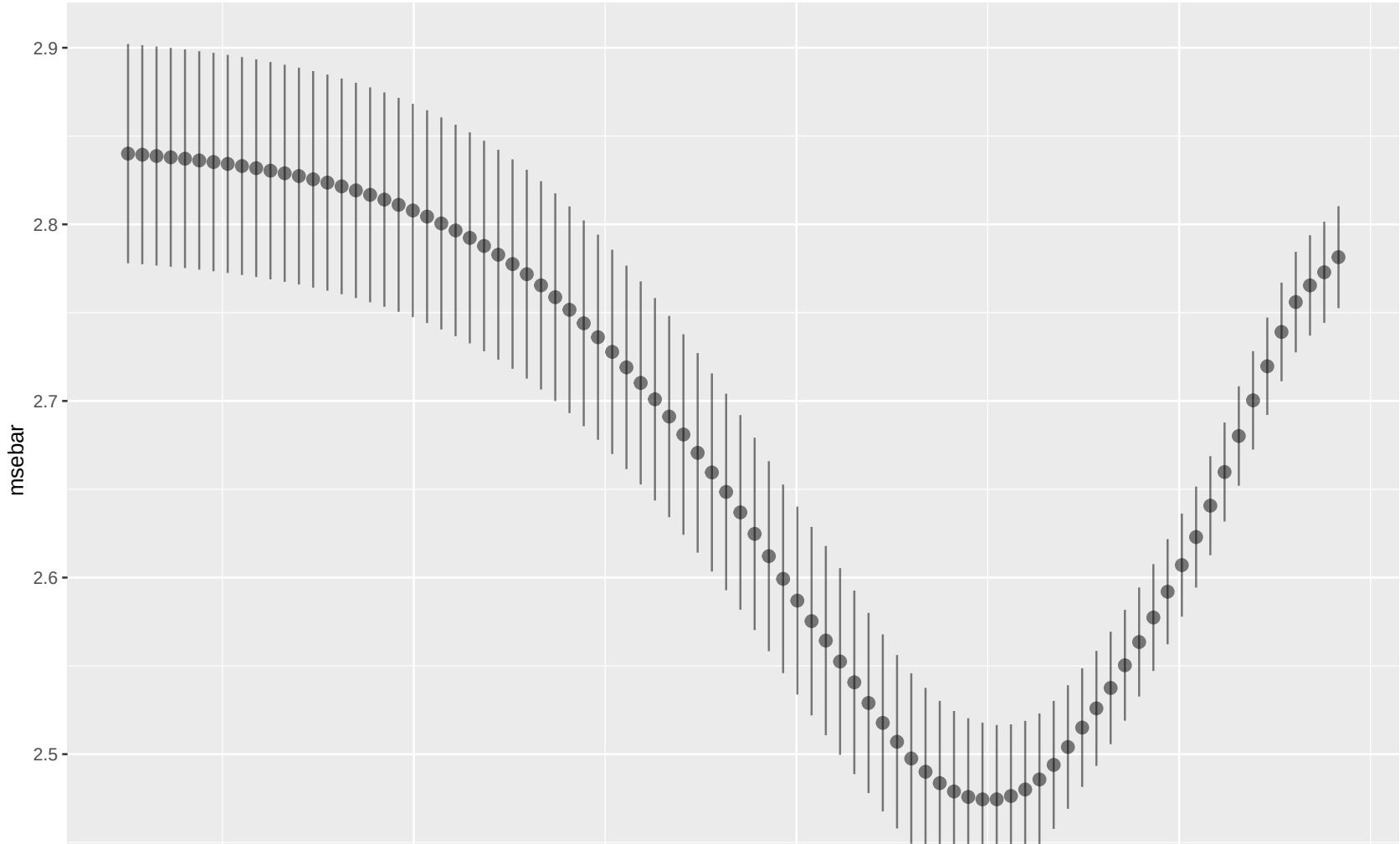
```
#Find mean and SE of that mean for each Row
cv_msd = apply(cv_mse,1,function(x)c(mean(x),sd(x)/sqrt(K)))
cv_msd = t(cv_msd) #apply does a nasty rotation
colnames(cv_msd) = c("msebar","sehat") #Give some names
cv_msd = as_tibble(cv_msd) #Make a DF
cv_msd$lambda = lambda_seq #Add the values of lambda
cv_msd = cv_msd %>% mutate(lb = msebar-sehat,
                           ub = msebar+sehat)
```

To plot, we want error bars that are $\bar{mse} \pm sd(MSE)/\sqrt{(K)}$. NB this is not a 95% CI, it is more like a 70% CI.

Cross-Validation in R – By hand

Now we can plot.

```
ggplot(cv_msd,aes(x=log(lambda),y=msebar,ymin=lb,ymax=ub))+
  geom_pointrange(alpha=0.5)
```



Simple Triplet Matrices

Sometimes your data will be very sparse (more than two-thirds 0s). It can then be efficient to ignore 0s when we store the data.

Usually the computer stores matrices as a big vector of numbers, with some dimension values (nrow, ncol) saved so that it knows where each row begins.

A *simple triplet matrix* only stores the indices and values of non-zero entries to your matrix. i.e. it is made up of

`row i, column j, entry value x`

Example STM

```
##      [,1] [,2]
## [1,]    0    2
## [2,]    1    0
## [3,]    0    0
## [4,]    0    3
```

is stored as

```
##      row i col j val
## [1,]     1     2   2
## [2,]     2     1   1
## [3,]     4     2   3
```

STM for Design Matrices

We can implement this in R using the `sparseMatrix` type from the `Matrix` package (which is auto-loaded by `glmnet`). Or, more simply, we can use `sparse.model.matrix` to build a sparse model matrix.

```
oj.sparse.matrix = sparse.model.matrix(logmove~.,data=oj)
oj.model = glmnet(oj.sparse.matrix,oj$logmove)
oj.sparse.matrix[1:5,1:5]
```

```
## 5 x 5 sparse Matrix of class "dgCMatrix"
##   (Intercept) store2 store5 store8 store9
## 1           1      1      .      .      .
## 2           1      1      .      .      .
## 3           1      1      .      .      .
## 4           1      1      .      .      .
## 5           1      1      .      .      .
```

CURRENT STOP

Wrap up

Things to do

HW 3 is out and due Wednesday of next week.

Rehash

- LASSO gives us fast automatic variable selection for a given level of penalty
- Cross Validation can help us choose that level of penalty, and a lot more.
  - We could also use AIC or other tools.
  - But lasso trades computational power for those theoretical assumptions
- We face a tradeoff between bias and variance in our model choices
- And a lot to think about computationally

Bye!