Biased Estimation: Ridge Regression

A brief introduction to ridge regression for dealing with collinearity.

AUTHOR

Andrew Zieffler

PUBLISHED

Oct. 22, 2020

The data in *equal-educational-opportunity.csv* consist of data taken from a random sample of 70 schools in 1965 (Chatterjee & Hadi, 2012; Coleman et al., 1966; Mosteller & Moynihan, 1972). We will use these data to mimic one of the original regression analyses performed; examining whether the level of school facilities was an important predictor of student achievement after accounting for the variation in faculty credentials and peer influence.

```r
# Load libraries
library(broom)
library(car)
library(corrr)
library(glmnet)
library(tidyverse)


# Read in data
eeo = read_csv("../data/equal-education-opportunity.csv")
head(eeo)
```

```
# A tibble: 6 x 4
  achievement faculty    peer school
        <dbl>   <dbl>   <dbl>  <dbl>
1      -0.431   0.608  0.0351  0.166
2       0.800   0.794  0.479   0.534
3      -0.925  -0.826 -0.620  -0.786
4      -2.19   -1.25  -1.22   -1.04
5      -2.85    0.174 -0.185   0.142
6      -0.662   0.202  0.128   0.273
```

To examine the RQ, the following model was posited:

$$\text{Achievement}_i = \beta_0 + \beta_1(\text{Faculty}_i) + \beta_2(\text{Peer}_i) + \beta_3(\text{School}_i) + \epsilon_i$$

Unfortunately, the model suffered from strong collinearity resulting in poor estimates of the coefficients and very large SEs. This means we cannot effectively control for peer influence and faculty credentials, which means we cannot answer the RQ.

Biased Estimation

The Gauss-Markov Theorem posits many attractive features of the OLS regression model. Two of these properties are that the OLS estimators will be unbiased and that the sampling variances of the coefficients are as small as possible [1] . In our example, the fitted model showed strong evidence of collinearity; which makes the SEs super large even if they are the minimum of all possible linear, unbiased estimates.

One method of dealing with collinearity is to use a biased estimation method. These methods forfeit unbiasedness to decrease the size of the sampling variances; it is bias–variance tradeoff. The goal with these methods is to trade a small amount of bias in the estimate for a large reduction in the sampling variances for the coefficients.

The bias–variance tradeoff is a commonplace, especially in prediction models. You can explore it in more detail at http://scott.fortmann-roe.com/docs/BiasVariance.html.

Ridge Regression

To date, the most commonly used biased estimation method in the social sciences is ridge regression. Instead of finding the coefficients that minimize the sum of squared errors, ridge regression finds the coefficients that minimize a penalized sum of squares, namely:

$$\text{SSE}_{\text{Penalized}} = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + d\sum_{j=1}^{p} \beta_j^2$$

where $d \geq 0$ is a scalar value, β_j is the *j*th regression coefficient, and y_i and $\hat{y}_i$ are the observed and model fitted values, respectively. One way to think about this formula is:

$$\text{SSE}_{\text{Penalized}} = \text{SSE} + \text{Penalty}$$

Minimizing over this penalized sum of squared error has the effect of "shrinking" the mean square error and coefficient estimates toward zero. As such, ridge regression is part of a larger class of methods known as *shrinkage methods*.

The *d* value in the penalty term controls the amount of shrinkage. When $d = 0$, the entire penalty term is 0, and the $\text{SSE}_{\text{Penalized}} = \text{SSE}$. Minimizing this will, of course, produce the OLS estimates. The bigger *d* is, the more the MSE and coefficients shrink toward zero. At the extreme end, $d = \infty$ will shrink every coefficients to zero.

Matrix Formulation of Ridge Regression

Recall that the OLS estimates are given by:

$$\boldsymbol{\beta} = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{Y}$$

If the $\mathbf{X}^\intercal\mathbf{X}$ matrix is ill-conditioned, computing the inverse results in a fair amount of inaccuracy which translates into bad estimates of the coefficients and standard errors. Unfortunately collinearity causes the $\mathbf{X}^\intercal\mathbf{X}$ matrix to be ill-conditioned. By inflating the diagonal elements of $\mathbf{X}^\intercal\mathbf{X}$, we can condition the matrix, which will hopefully lead to more stable coefficient estimates.

If a matrix has a high condition number as computed by $\kappa = \frac{|\lambda_{\text{Max}}|}{|\lambda_{\text{Min}}|}$, it is said to be ill-conditioned.

To do this, we can add some constant amount (*d*) to each of the diagonal elements of $\mathbf{X}^\intercal\mathbf{X}$, prior to finding the inverse, which, we can express as:

$$\widetilde{\boldsymbol{\beta}} = (\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}\mathbf{X}^\intercal\mathbf{Y}$$

Technically this equation is for standardized variables.

Ridge Regression in Practice: An Example

Let's fit a ridge regression model to our EEO data. For now, we will choose the value for *d*. Let's use $d = 0.1$. Prior to any matrix calculations, we first standardize all potential variables in the model. (It is recommended that you always standardize all variables prior to fitting a ridge regression because large coefficients will impact the penalty in the SSE more than small coefficients.)

```
# Standardize all variables in the eeo data frame
z_eeo = scale(eeo)

# View
head(z_eeo)
```

```
     achievement    faculty        peer     school
[1,]  -0.1983021  0.5158617 -0.01213684  0.1310782
[2,]   0.3434321  0.6871673  0.46812962  0.4901170
[3,]  -0.4153134 -0.8084583 -0.71996624 -0.7994390
[4,]  -0.9724350 -1.2024935 -1.36577136 -1.0479983
[5,]  -1.2616881  0.1150408 -0.25030749  0.1078451
[6,]  -0.2998797  0.1413252  0.08793894  0.2356566
```

The design matrix, **X**, is a 70×3 matrix (since the variables are standardized, the intercept in any regression will be set to zero and is not estimated). Here we use the `data.matrix()` function to create a design matrix based on the standardized predictors.

```
# Create and view the design matrix
X = z_eeo[ , c("faculty", "peer", "school")]
head(X)
```

```
       faculty        peer     school
[1,]  0.5158617 -0.01213684  0.1310782
[2,]  0.6871673  0.46812962  0.4901170
[3,] -0.8084583 -0.71996624 -0.7994390
[4,] -1.2024935 -1.36577136 -1.0479983
[5,]  0.1150408 -0.25030749  0.1078451
[6,]  0.1413252  0.08793894  0.2356566
```

The **Y** matrix is a 70×1 column matrix of the standardized outcome values, namely:

```
# Create and view the outcome vector
Y = z_eeo[ , "achievement"]
head(Y)
```

```
[1] -0.1983021  0.3434321 -0.4153134 -0.9724350 -1.2616881 -0.2998797
```

Since $\mathbf{X}^\intercal\mathbf{X}$ is a 3×3 matrix, $d\mathbf{I}$ must also be a 3×3 matrix (to be able to sum them). Here we use $\lambda = 0.1$ just to illustrate the concept of ridge regression:

```
# Compute and view dI
dI = 0.1 * diag(3)
dI
```

```
     [,1] [,2] [,3]
[1,]  0.1  0.0  0.0
[2,]  0.0  0.1  0.0
[3,]  0.0  0.0  0.1
```

Finally we can perform the matrix algebra to obtain the ridge regression coefficients:

```
# Compute ridge regression coefficients
b = solve(t(X) %*% X + dI) %*% t(X) %*% Y
b
```

```
              [,1]
faculty  0.4347370
peer     0.8552354
school  -0.8491600
```

The fitted ridge regression model is then,

$$\hat{\text{Achievement}}_i^\star = 0.435(\text{Faculty}_i^\star) + 0.855(\text{Peer}_i^\star) - 0.849(\text{School}_i^\star)$$

where the star-superscript denotes a standardized (*z*-scored) variable.

Using a Built-In Function

We can also use the `lm.ridge()` function from the **MASS** package to fit a ridge regression. This function uses a formula based on variables from a data frame in the same fashion as the `lm()` function. It also takes the argument `lambda=` which specifies the values of *d* to use in the penalty term.

```
# Create data frame for use in lm.ridge()
z_data = z_eeo %>%
  data.frame()

# Load MASS library
library(MASS)

# Fit ridge regression
ridge_1 = lm.ridge(achievement ~ -1 + faculty + peer + school, data = z_data, lambda = 0.1

# View coefficients
tidy(ridge_1)
```

```
# A tibble: 3 x 5
  lambda    GCV term     estimate scale
   <dbl>  <dbl> <chr>       <dbl> <dbl>
1    0.1 0.0121 faculty     0.433 0.993
2    0.1 0.0121 peer        0.850 0.993
3    0.1 0.0121 school     -0.845 0.993
```

Comparison to the OLS Coefficients

Fitting a standardized OLS model to the data, we find:

```
# Convert the scaled data to a data frame
z_data = z_eeo %>%
  data.frame()
```

```r
# Fit standardized OLS model
lm.1 = lm(achievement ~ faculty + peer + school - 1, data = z_data)
coef(lm.1)
```

```
  faculty       peer     school 
0.5248647  0.9449056 -1.0272986 
```

Comparing these coefficients to the coefficients from the ridge regression:

Table 1: Comparison of the coefficients from the OLS (d=0) and ridge regression (d=0.1) based on the standardized data.

Predictor	$d = 0$	$d = 0.1$
Faculty	0.525	0.435
Peer	0.945	0.855
School	-1.027	-0.849

Based on this comparison, the ridge regression has "shrunk" the estimate of each coefficient toward zero. Remember, the larger the value of d, the more the coefficient estimates will shrink toward 0. The table below shows the coefficient estimates for four different values of d.

Table 2: Comparison of the coefficients from the OLS (d=0) and three ridge regressions (d=0.01, d=0.1, and d=0.5) based on the standardized data.

Predictor	$d = 0$	$d = 0.01$	$d = 0.1$	$d = 0.5$
Faculty	0.525	0.514	0.436	0.260
Peer	0.945	0.935	0.856	0.657
School	-1.027	-1.007	-0.851	-0.480

Choosing *d*

In practice you need to specify the *d* value to use in the ridge regression. Ideally you want to choose a value for *d* that:

- Introduces the least amount of bias possible, while also
- Obtaining better sampling variances.

This is an impossible task without knowing the true values of the coefficients (i.e., the β values). There are, however, some empirical methods to help us in this endeavor.

Ridge Trace

The plot of the *ridge trace* displays the ridge regression coefficients as a function of *d*. Examining this plot, the analyst can use it to pick a *d* value. To do this, pick the smallest value for *d* that produces stable regression coefficients. Here we examine the values of *d* where $d = \{0, 0.001, 0.002, 0.003, \ldots, 100\}$.

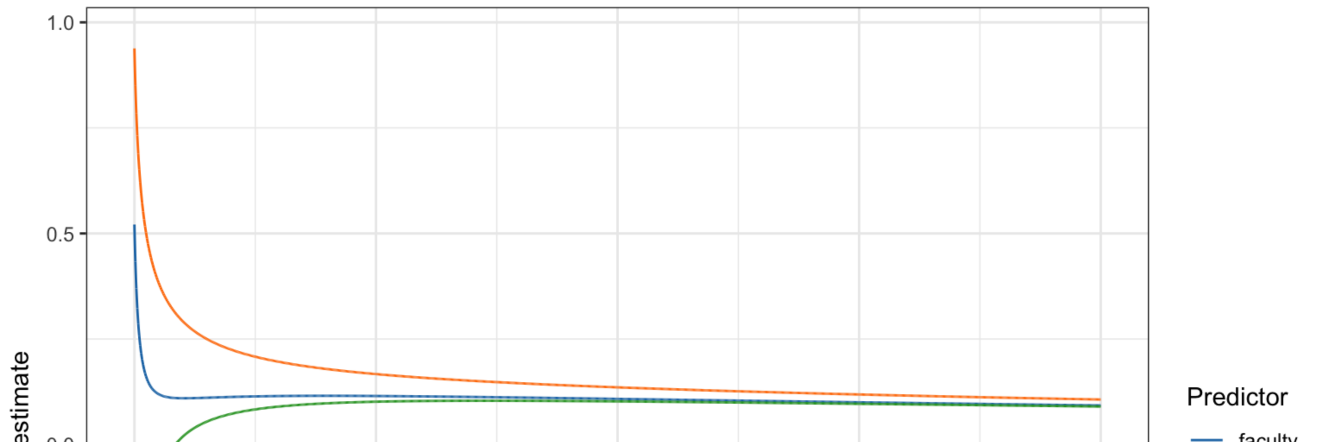
To create this plot, we fit a ridge regression that includes a sequence of values in the `lambda=` argument of the `lm.ridge()` function. Then we use the `tidy()` function from the **broom** library to summarize the output from this model. This output includes the coefficient estimates for each of the *d* values in our sequence. We can then create a line plot of the coefficient values versus the *d* values for each predictor.

```
# Fit ridge model across several lambda values
ridge_models = lm.ridge(achievement ~ -1 + faculty + peer + school, data = z_data, lambda =

# Get tidy() output
out = tidy(ridge_models)
out
```

```
# A tibble: 300,003 x 5
   lambda     GCV term     estimate scale
    <dbl>   <dbl> <chr>       <dbl> <dbl>
 1  0      0.0122 faculty     0.521 0.993
 2  0      0.0122 peer        0.938 0.993
 3  0      0.0122 school     -1.02  0.993
 4  0.001  0.0122 faculty     0.520 0.993
 5  0.001  0.0122 peer        0.937 0.993
 6  0.001  0.0122 school     -1.02  0.993
 7  0.002  0.0122 faculty     0.519 0.993
 8  0.002  0.0122 peer        0.936 0.993
 9  0.002  0.0122 school     -1.02  0.993
10  0.003  0.0122 faculty     0.518 0.993
# … with 299,993 more rows
```

```
# Ridge trace
ggplot(data = out, aes(x = lambda, y = estimate)) +
  geom_line(aes(group = term, color = term)) +
  theme_bw() +
  xlab("d value") +
  ylab("Coefficient estimate") +
  ggsci::scale_color_d3(name = "Predictor")
```



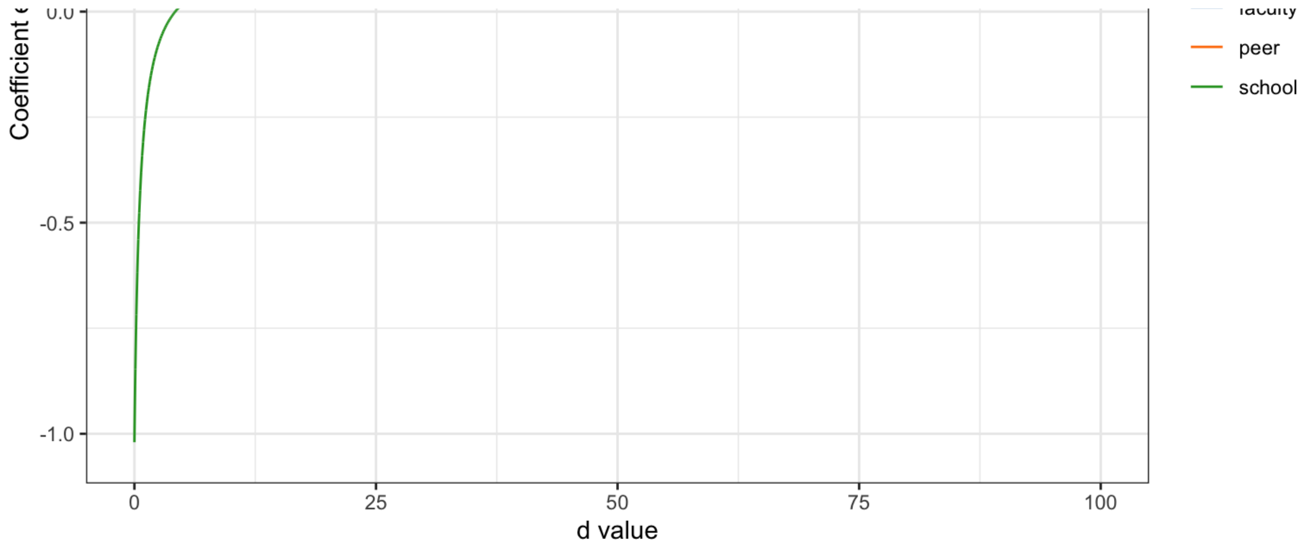


Figure 1: Ridge plot showing the size of the standardized regression coefficients for d values between 0 (OLS) and 100.

We want to find the *d* value where the lines begin to flatten out; where the coefficients are no longer changing value. This is difficult to ascertain, but somewhere around $d = 50$, there doesn't seem to be a lot of change in the coefficients. This suggests that a *d* value around 50 would produce stable coefficient estimates.

It turns out that not only is it difficult to make a subjective call about where the trace lines begin to flatten, but even when people do make this determination, they often select a *d* value that is too high. A better way to obtain a *d* value is to compute a model-level metric that we can then evaluate across the models produced by the different values of *d*. One such metric is the Akiake Information Criteria (AIC) which we can compute for a ridge regression as

$$\mathrm{AIC} = n \times \ln\left(\mathbf{e}^\intercal\mathbf{e}\right) + 2(\mathit{df})$$

where n is the sample size, $\mathbf{e}$ is the vector or residuals from the ridge model, and *df* is the degrees of freedom associated with the ridge regression model. We compute the degrees of freedom as the trace of the **H** matrix, where,

$$\mathbf{H}_{\mathrm{Ridge}} = \mathbf{X}(\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}\mathbf{X}^\intercal$$

For example, to compute the AIC value associated with the ridge regression estimated using a *d* value of 0.1 we can use the followng syntax.

```r
# Compute coefficients for ridge model
b = solve(t(X) %*% X + 0.1*diag(3)) %*% t(X) %*% Y

# Compute residual vector
e = Y - (X %*% b)

# Compute H matrix
H = X %*% solve(t(X) %*% X + 0.1*diag(3)) %*% t(X)
```

```r
# Compute df
df = sum(diag(H))

# Compute AIC
aic = 70 * log(t(e) %*% e) + 2 * df
aic
```

```
         [,1]
[1,] 285.8729
```

We want to compute the AIC value for every single one of the models associated with the *d* values from our sequence we used to produce the ridge trace plot. To do this, we will create a FOR loop that will cycle through each of the *d* values, and compute the AIC. We need to store these AIC values, so we initially will create an empty vector to store them in.

```r
# Re-create the sequence of d values
d = seq(from = 0, to = 100, by = 0.001)

# Create an empty vector to store the AIC values
aic = c()

# FOR loop to cycle through the different values of d
for(i in 1:length(d)){

  b = solve(t(X) %*% X + d[i]*diag(3)) %*% t(X) %*% Y
  e = Y - (X %*% b)
  H = X %*% solve(t(X) %*% X + d[i]*diag(3)) %*% t(X)
  df = sum(diag(H))

  # Create and store the AIC value
  aic[i] = 70 * log(t(e) %*% e) + 2 * df
}
```

We can now create a data frame that includes the *d* values and the associated AIC values. Then, we find the *d* value associated with the smallest AIC value.

```r
# Create data frame of d and AIC values
my_models = data.frame(d, aic)

# Find d associated with smallest AIC
my_models %>%
  filter(aic == min(aic))
```

```
       d      aic
1 21.765 284.1859
```

A *d* value of 21.765 produces the smallest AIC value, so this is the *d* value we will adopt.

```r
# Re-fit ridge regression
ridge_3 = lm.ridge(achievement ~ -1 + faculty + peer + school, data = z_data, lambda = 21.7

# View coefficients
tidy(ridge_3)
```

```
# A tibble: 3 x 5
  lambda    GCV term     estimate scale
   <dbl>  <dbl> <chr>       <dbl> <dbl>
1   21.8 0.0118 faculty    0.115  0.993
2   21.8 0.0118 peer       0.174  0.993
3   21.8 0.0118 school     0.0994 0.993
```

Based on using $d = 21.765$, the fitted ridge regression model is:

$$\hat{\text{Achievement}}_i^\star = 0.115(\text{Faculty}_i^\star) + 0.174(\text{Peer}_i^\star) - 0.099(\text{School}_i^\star)$$

Estimating Bias

Recall that the ridge regression produces biased estimates such that $\mathbb{E}(\hat{\beta}) \neq \beta$. The amount of bias in the coefficient estimates is defined as:

$$\text{Bias}(\hat{\boldsymbol{\beta}}_{\text{Ridge}}) = -d(\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}\boldsymbol{\beta}$$

where $\boldsymbol{\beta}$ are the coefficients from the standardized regression model.

Remember that $d = 0$ was the OLS model. In that case,

$$\begin{aligned}\text{Bias}(\hat{\boldsymbol{\beta}}_{\text{OLS}}) &= 0(\mathbf{X}^\intercal\mathbf{X} + 0\mathbf{I})^{-1}\boldsymbol{\beta} \\ &= 0\end{aligned}$$

There is no bias in the OLS coefficients. Of course, as *d* increases, the bias in the coefficient estimates will also increase. We can use our sample data to estimate the bias, it is likely a poor estimate, as to obtain the bias we really need to know the true β-parameters (which of course we do not know).

```r
# OLS estimates
b_ols = solve(t(X) %*% X) %*% t(X) %*% Y

# Compute dI
dI = 21.765*diag(3)

# Estimate bias in ridge regression coefficients
-21.765 * solve(t(X) %*% X + dI) %*% b_ols
```

```
-21.765 * solve(t(X) %*% X + dI) %*% b_ols
```

```
              [,1]
faculty -0.4086488
peer    -0.7703215
school   1.1277056
```

It looks like the faculty and peer coefficients are biased downward and the school coefficient is biased upward in our ridge regression model. We could also see this in the ridge trace plot we created earlier.

```
# Ridge trace
ggplot(data = out, aes(x = lambda, y = estimate)) +
  geom_line(aes(group = term, color = term)) +
  geom_vline(xintercept = 21.765, linetype = "dotted") +
  theme_bw() +
  xlab("d value") +
  ylab("Coefficient estimate") +
  ggsci::scale_color_d3(name = "Predictor")
```

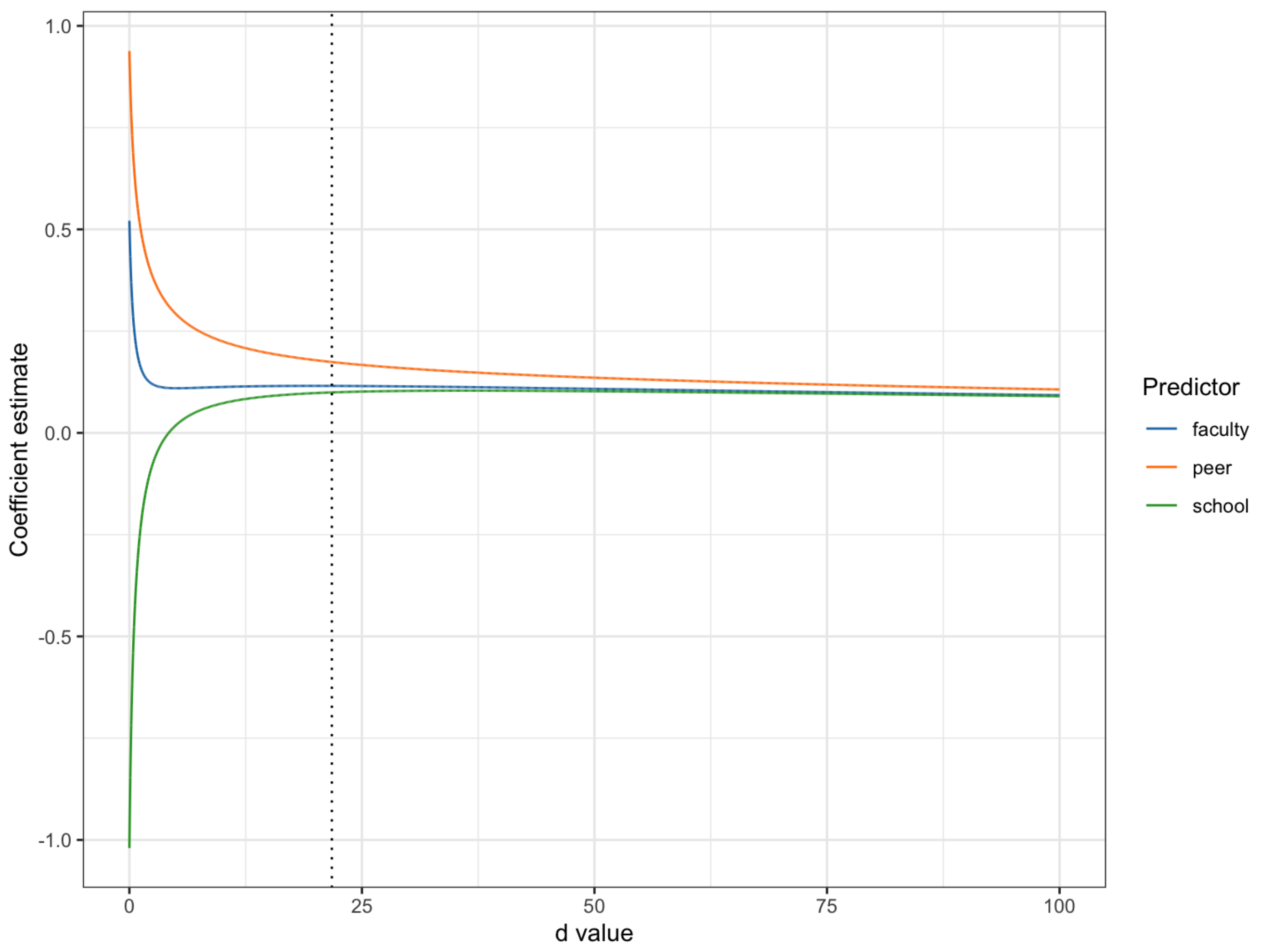


Figure 2: Ridge plot showing the size of the standardized regression coefficients for d values between 0 (OLS) and 100. The vertical dotted line is shown at d=21.765, the value of d that has the minimum GCV value.

In this plot, the estimates for the peer and faculty coefficients are being shrunken downward from the OLS estimates (at $d = 0$), and the school coefficient is being "shrunken" upward (closer to 0). Moreover, the bias is simply the difference between the OLS estimates and the ridge coefficient estimates.

```
# Difference b/w OLS and ridge coefficients
tidy(ridge_3)$estimate - b_ols
```

```
              [,1]
faculty -0.4094594
peer    -0.7708088
school   1.1267390
```

Sampling Variances, Standard Errors, and Confidence Intervals

Recall that the sampling variation for the coefficients is measured via the variance. Mathematically, the variance estimate of a coefficient is defined as the expectation of the squared difference between the parameter value and the estimate:

$$\sigma^2_{\hat{\beta}} = \mathbb{E}\left[(\hat{\beta} - \beta)^2\right]$$

Using rules of expectations, we can re-write this as:

$$\sigma^2_{\hat{\beta}} = \mathbb{E}\left[\left(\hat{\beta} - \mathbb{E}[\beta]\right)^2\right] + \left(\mathbb{E}[\hat{\beta} - \beta]\right)^2$$

The first term in this sum represents the variance in $\hat{\beta}$ and the second term is the squared amount of bias in $\hat{\beta}$. As a sum,

$$\sigma^2_{\hat{\beta}} = \text{Var}(\hat{\beta}) + \text{Bias}(\hat{\beta})^2$$

In the OLS model, the bias of the estimate is 0, and this reduces to the variance of the estimate. In ridge regression, the bias term is not zero. The bias–variance tradeoff says that by increasing bias, we will decrease the variance. So while the second term will get bigger, the first will get smaller. The hope is that overall we can reduce the amount of sampling variance in the coefficients. However, since the sampling variance includes the square of the bias, we have to be careful that we don't increase bias too much, or it will be counterproductive.

The fact that the sampling variance for the coefficients is dependent on both the variance and the amount of bias is a major issue in estimating the sampling variation for a coefficient in ridge regression. This means we need to know how much bias there is to get a true accounting of the sampling variation. As Goeman et al. (2018) notes,

> *Unfortunately, in most applications of penalized regression it is impossible to obtain a sufficiently precise estimate of the bias…calculations can only give an assessment of the variance of the estimates. Reliable estimates of the bias are only available if reliable unbiased estimates are available, which is typically not the case in situations in which penalized estimates are used.*

He goes on to make it clear why most programs do not report SEs for the coefficients:

> *Reporting a standard error of a penalized estimate therefore tells only part of the story. It can give a mistaken impression of great precision, completely ignoring the inaccuracy caused by the bias. It is certainly a mistake to make confidence statements that are only based on an assessment of the variance of the estimates.*

Estimating Sampling Variance

In theory it is possible to obtain the sampling variances for the ridge regression coefficients using matrix algebra:

$$\boldsymbol{\sigma^2_{\beta}} = \sigma^2_{\epsilon}(\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}\mathbf{X}^\intercal\mathbf{X}(\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}$$

where σ^2_{ϵ} is the the error variance estimated from the standardized OLS model.

```
# Fit standardized model to obtain sigma^2_epsilon
glance(lm(achievement ~ -1 + faculty + peer + school, data = z_data))
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl>
1     0.206         0.171 0.904      5.80 0.00138     3  -90.7  189.
# … with 4 more variables: BIC <dbl>, deviance <dbl>,
#   df.residual <int>, nobs <int>
```

```
# Compute sigma^2_epsilon
mse = 0.9041214 ^ 2

# Compute variance-covariance matrix of ridge estimates
W = solve(t(X) %*% X + 21.765*diag(3))
var_b = mse * W %*% t(X) %*% X %*% W
```

```r
# Compute SEs
sqrt(diag(var_b))
```

```
   faculty       peer     school 
0.05483128 0.05671198 0.04134060 
```

Comparing these SEs to the SEs from the OLS regression:

Table 3: Comparison of the standard errors from the OLS (d=0) and ridge regression (d=21.765) based on the standardized data.

Predictor	$d = 0$	$d = 21.765$
Faculty	0.667	0.055
Peer	0.598	0.057
School	0.993	0.041

Based on this comparison, we can see that the standard errors from the ridge regression are quite a bit smaller than those from the OLS. We can then use the estimates and SEs to compute *t*- and *p*-values, and confidence intervals. Here we only do it for the predictor of interest from our RQ, but one could do it for all the predictors.

```r
# Compute t-value for school predictor
t = 0.09944044    / 0.05483128
t
```

```
[1] 1.813571
```

```r
# Compute df residual
H = X %*% solve(t(X) %*% X + 21.765*diag(3)) %*% t(X)
df_model = sum(diag(H))
df_residual = 69 - df_model

# Compute p-value
p = pt(-abs(t), df = df_residual) * 2
p
```

```
[1] 0.07415901
```

```r
# Compute CI
0.09944044 - qt(p = 0.975, df = df_residual) * 0.05483128
```

```
[1] -0.009974863
```

```r
0.09944044 + qt(p = 0.975, df = df_residual) * 0.05483128
```

```
[1] 0.2088557
```

This suggests that after controlling for peer influence and faculty credential, there is some slightevidence of an effect of school facilities on student achievement ($p = .074$). While the uncertainty in the 95% CI suggests that the true partial effect of school facilities may include 0, the empirical evidence is pointing toward a slight positive effect of school facilities.

Bias–Variance Tradeoff: Revisited

Now that we have seen how the bias and the sampling variance are calculated, we can study these formulae to understand why there is a bias–variance tradeoff.

$$\begin{aligned}\text{Bias}(\hat{\boldsymbol{\beta}}_{\text{Ridge}}) &= -d(\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}\boldsymbol{\beta} \\ \text{Var}(\hat{\boldsymbol{\beta}}_{\text{Ridge}}) &= \sigma^2_\epsilon(\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}\mathbf{X}^\intercal\mathbf{X}(\mathbf{X}^\intercal\mathbf{X} + d\mathbf{I})^{-1}\end{aligned}$$

Examining these two formulas, we can see that as *d* increases, the bias also increases. At the same time, increasing *d* in the formula for sampling variance will decrease. This. in a nutshell, is the bias–variance tradeoff. Decreasing one of these properties tends to increase the other. The key is to find a value of *d* that minimally increases bias to maximally decrease the sampling variances.

References

Footnotes

1. At least within the class of linear, unbiased estimators. [↩]

References

Chatterjee, S., & Hadi, A. S. (2012). *Regression analysis by example*. Wiley.

Coleman, J. S., Cambell, E. Q., Hobson, C. J., McPartland, J., Mood, A. M., Weinfield, F. D., & York, R. L. (1966). *Equality of educational opportunity*. U.S. Government Printing Office.

Goeman, J., Meijer, R., & Chaturvedi, N. (2018). *L1 and l2 penalized regression models*. R Vignette. https://cran.r-project.org/web/packages/penalized/vignettes/penalized.pdf

[illegible]

Mosteller, F., & Moynihan, D. F. (1972). *On equality of educational opportunity*. Random House.