2017 ■ Vol. 13 ■ no. 1

Adding bias to reduce variance in psychological results: A tutorial on penalized regression

Nathaniel E. Helwig[a,]

[a]Department of Psychology and School of Statistics, University of Minnesota

Abstract ■ Regression models are commonly used in psychological research. In most studies, regression coefficients are estimated via maximum likelihood (ML) estimation. It is well-known that ML estimates have desirable large sample properties, but are prone to overfitting in small to moderate sized samples. In this paper, we discuss the benefits of using penalized regression, which is a form of penalized likelihood (PL) estimation. Informally, PL estimation can be understood as introducing bias to estimators for the purpose of reducing their variance, with the ultimate goal of providing better solutions. We focus on the Gaussian regression model, where ML and PL estimation reduce to ordinary least squares (OLS) and penalized least squares (PLS) estimation, respectively. We cover classic OLS and stepwise regression, as well as three popular penalized regression approaches: ridge regression, the lasso, and the elastic net. We compare the different penalties (or biases) imposed by each method, and discuss the resulting features each penalty encourages in the solution. To demonstrate the methods, we use an example where the goal is to predict a student's math exam performance from 30 potential predictors. Using a step-by-step tutorial with R code, we demonstrate how to (i) load and prepare the data for analysis, (ii) fit the OLS, stepwise, ridge, lasso, and elastic net models, (iii) extract and compare the model fitting results, and (iv) evaluate the performance of each method. Our example reveals that penalized regression methods can produce more accurate and more interpretable results than the classic OLS and stepwise regression solutions.

Acting Editor ■ Denis Cousineau (Université d'Ottawa)

Keywords ■ penalized least squares, ordinary least squares, ridge, lasso, elastic net. **Tools** ■ R.

helwig@umn.edu

NEH: 0000-0003-2907-1497

10.20982/tqmp.13.1.p001

Introduction

Since the mid-19th century, researchers have employed a variety of methods in an attempt to understand relationships between psychological variables. Over the years, the General Linear Model (GLM) has proven to be a useful framework for analyzing psychological data; note that the GLM framework incorporates linear regression, one-way and factorial analysis of variance, analysis of covariance, etc., which have become some of the most commonly used statistical tools in psychological research (see Azen & Budesco, 2009; Chartier & Faulkner, 2008; Cousineau, 2005). In most applications of the GLM to psychological data, the ordinary least squares (OLS) loss function is used to estimate the model coefficients. The OLS coefficients are equivalent to the Maximum Likelihood Estimates (MLEs) of the coefficients under the assumption that the model error terms are independent, identically distributed (iid) Gaussian variables with mean zero.

Despite their widespread use, it is well-known that the OLS coefficients can perform poorly under certain situations, e.g., highly correlated predictors and/or small signal to noise ratios (SNRs). Furthermore, it is well-known that MLEs have desirable large sample properties (e.g., consistency and efficiency), but may not be ideal for analyzing small to moderate sized samples of data. In any real psychological study the sample size is finite, so the desirable large sample properties of the MLEs may not be attained.

An undesirable consequence of relying on ML estimation in small to moderate sized samples is overfitting, which refers to the phenomenon of finding spurious effects. In the case of unbiased estimators such as the OLS coefficients, the over-fitting phenomenon can be attributed to the high variability of the OLS coefficients in small to moderate sized samples. This is particularly true when the SNR in the data is small, which may be the case in many psychological studies.

When the GLM is being applied to test clearly defined hypotheses, it is useful to give some consideration to the sample size that would be needed to find effects of certain magnitudes (Cohen, 1992) and with reasonable accuracy (Kelley & Maxwell, 2003). However, in many cases the sample size is out of the control of the researcher, e.g., due to budget constraints or study design, so it may not always be possible to obtain enough subjects to confidently rely on OLS regression. Furthermore, given the wealth of data collected in some domains of psychology (e.g., education, social, health, personality, etc.), it is becoming common to have more potential predictor variables than subjects, i.e., $n < p$. In such cases, the OLS regression coefficients are not uniquely defined, so some alternative approach is needed. One possibility is to fit the regression model to a subset of the predictors, where the subset is selected via some prior knowledge. Another possibility is to use a data-driven approach such as penalized regression.

Penalized regression is a form of penalized likelihood estimation where the goal is to find an estimator that provides an optimal balance between fitting the data and providing a parsimonious solution (see Hastie, Tibshirani, & Friedman, 2009). These approaches introduce a "penalty" to the OLS loss function, where the penalty is designed to encourage more interpretable and/or stable results. Unlike the classic OLS estimator, penalized least squares (PLS) estimators are biased due to the imposed penalty. However, by introducing a small bias, it may be possible to substantially reduce the variability of the estimator, resulting in a better behaved estimator than the corresponding unpenalized estimator. Consequently, penalized estimation can be informally understood as introducing bias to estimators for the purpose of reducing their variance, with the goal of providing solutions that are more interpretable and more likely to validate in new samples of data.

In this paper, we discuss how penalized regression methods can be a useful tool for improving the replicability of psychological research. We note that other authors have recently discussed the benefits of penalized estimation for psychology (Jacobucci, Grimm, & McArdle, 2016; McNeish, 2015); however, our tutorial is unique in the sense that we unify all of the penalized regression methods under the elastic net umbrella—which is not discussed in the previous tutorials on this topic. We begin by reviewing the GLM and OLS estimation, as well as some classic approaches to predictor selection, i.e., stepwise regression and p-value model selection. Next, we define three concepts of statistical estimators—bias, variance, and mean squared error—that are crucial to understanding the logic of penalized regression estimators. We then discuss three penalized regression methods (ridge, lasso, elastic net), where each concept is introduced as a simple modification to OLS loss function. We compare the different penalties (or biases) imposed by each method, and discuss the resulting features each penalty encourages in the solution. To demonstrate the methods, we use an example where the goal is to predict a student's math exam performance from a variety of potential predictors. Using a step-by-step tutorial with embedded R code (R Core Team, 2016), we demonstrate how to both fit and evaluate the various regression models. Our results reveal that the penalized regression approaches can provide more accurate and more informative solutions than the classic OLS and stepwise regression coefficients.

1 Regression Background

1.1 *Ordinary Least Squares Regression*

Given data collected from n subjects, the general linear model (GLM) assumes that

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i \tag{1}$$

for $i = 1, \ldots, n$ where y_i is the observed response variable for the i-th subject, x_{ij} is the j-th observed predictor for the i-th subject, $(\beta_0, \beta_1, \ldots, \beta_p)$ are the unknown regression coefficients, and $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ are independent, identically distributed (iid) Gaussian error terms. This implies that $(y_i|\mathbf{x}_i) \overset{\text{ind}}{\sim} N(\beta_0 + \sum_{j=1}^p \beta_j x_{ij}, \sigma^2)$, i.e., conditioned on the predictors $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})'$, the y_i are independent, normally distributed variables with mean $\beta_0 + \sum_{j=1}^p \beta_j x_{ij}$ and homogenous variance σ^2. The GLM can be written more compactly in matrix form such as

$$\mathbf{y} = \mathbf{1}_n \beta_0 + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2}$$

where $\mathbf{y} = (y_1, \ldots, y_n)'$ is the $n \times 1$ response vector, $\mathbf{1}_n = \{1\}_{n\times 1}$ is an $n \times 1$ vector of ones, $\mathbf{X} = \{x_{ij}\}_{n\times p}$ is the $n \times p$ design matrix, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)'$ is the $p \times 1$ vector of slope coefficients, and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)' \sim N(\mathbf{0}_n, \sigma^2 \mathbf{I}_n)$ is the $n \times 1$ error vector. The previously mentioned model assumptions imply that $(\mathbf{y}|\mathbf{X}) \sim N(\mathbf{1}_n\beta_0 + \mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n)$, i.e., conditioned on the design matrix $\mathbf{X}$, the response vector $\mathbf{y}$ follows a multivariate normal distribution with mean vector $\mathbf{1}_n\beta_0 + \mathbf{X}\boldsymbol{\beta}$ and covariance matrix $\sigma^2\mathbf{I}_n$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix.

The unknown coefficients of the GLM (i.e., β_0 and $\boldsymbol{\beta}$) are often obtained by minimizing the ordinary least squares (OLS) function

$$\begin{aligned}\text{OLS}(\beta_0, \boldsymbol{\beta}) &= \sum_{i=1}^{n} \left(y_i - \beta_0 - \textstyle\sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \\ &= \|\mathbf{y} - \mathbf{1}_n \beta_0 - \mathbf{X}\boldsymbol{\beta}\|^2\end{aligned} \quad (3)$$

where $\|\mathbf{u}\|^2 = \sum_{i=1}^{n} u_i^2$ denotes the squared Euclidean norm for any vector $\mathbf{u} = (u_1, \ldots, u_n)'$. The coefficients that minimize the OLS function in Equation (3) have the form

$$\begin{aligned}\hat{\beta}_0 &= \bar{y} - \textstyle\sum_{j=1}^{p} \hat{\beta}_j \bar{x}_j \\ \hat{\boldsymbol{\beta}} &= (\mathbf{X}_c'\mathbf{X}_c)^{-1}\mathbf{X}_c'\mathbf{y}_c\end{aligned} \quad (4)$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ are the sample means of Y and X_j (respectively), and $\mathbf{X}_c = \mathbf{CX}$ and $\mathbf{y}_c = \mathbf{Cy}$ are mean-centered versions of the design matrix $\mathbf{X}$ and response vector $\mathbf{y}$ (respectively) with $\mathbf{C} = \mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n'$ denoting a centering matrix.

1.2 Standardized Regression

The coefficient β_j represents expected change in Y for one-unit change in X_j holding other predictors constant. If X_j and X_k are measured in comparable units, then the magnitudes of β_j and β_k will be comparable. However, X_j and X_k are measured in different units (e.g., age and intelligence), the scales of β_j and β_k are not comparable with one another, which confounds the interpretation of the model results. To resolve this, the standardized regression model transforms the response and predictor variables into Z-scores (i.e., mean zero and variance one) before fitting the model. More specifically, the standardized regression model has the form

$$y_i^* = \sum_{j=1}^{p} \beta_j^* x_{ij}^* + \epsilon_i^* \quad (5)$$

where $y_i^* = (y_i - \bar{y})/s_y$ is the standardized response variable with $s_y^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$ denoting the sample variance of Y, and $x_{ij}^* = (x_{ij} - \bar{x}_j)/s_j$ is the j-th standardized predictor with $s_j = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2$ denoting the sample variance of X_j. In this case, β_j^* represents the expected change in Y standard deviations (SDs) for one SD change in X_j holding other predictors constant.

Letting $\mathbf{C} = \mathbf{I}_n - n^{-1}\mathbf{1}_n\mathbf{1}_n'$ denote a centering matrix and $\mathbf{S} = \text{diag}(s_1, \ldots, s_p)$ denote a diagonal scaling matrix, the standardized design matrix can be written as $\mathbf{X}_{cs} = \mathbf{CXS}^{-1}$. Similarly, $\mathbf{y}_{cs} = \mathbf{Cy}s_y^{-1}$ is standardized version of Y. The OLS solution for $\boldsymbol{\beta}^*$ has the form

$$\begin{aligned}\hat{\boldsymbol{\beta}}^* &= (\mathbf{X}_{cs}'\mathbf{X}_{cs})^{-1}\mathbf{X}_{cs}'\mathbf{y}_{cs} \\ &= (\mathbf{S}^{-1}\mathbf{X}_c'\mathbf{X}_c\mathbf{S}^{-1})^{-1}\mathbf{S}^{-1}\mathbf{X}_c'\mathbf{y}_c s_y^{-1} \\ &= s_y^{-1}\mathbf{S}(\mathbf{X}_c'\mathbf{X}_c)^{-1}\mathbf{X}_c'\mathbf{y}_c = s_y^{-1}\mathbf{S}\hat{\boldsymbol{\beta}}\end{aligned} \quad (6)$$

where $\hat{\boldsymbol{\beta}}$ is the least squares estimate without standardizing. Consequently, the standardized regression coefficients are related to the original (i.e., unstandardized) coefficients via a simple rescaling: $\hat{\beta}_j^* = \hat{\beta}_j s_j / s_y$ for all $j = 1, \ldots, p$. Finally, note that the least squares estimate of the intercept is zero in the standardized regression model, given that $\bar{y} = \bar{x}_j = 0$ for all $j = 1, \ldots, p$, so the intercept term can be omitted.

1.3 Variable Selection

1.3.1 Stepwise regression

Given a collection of potential predictor variables $X_1, \ldots, X_p$, *stepwise regression* uses an automated selection algorithm to determine which predictors should be included in the model. The three common variations of stepwise regression include: (i) forward selection only, (ii) backward elimination only, and (iii) both forward selection and backward elimination. All three methods iteratively add/remove predictors to/from the model according to some user-determined criterion. However, due to the stepwise nature of these algorithms, stepwise regression does not necessarily evaluate the chosen criterion for all 2^p possible models. As a result, there is no guarantee that stepwise regression will produce the model (among the 2^p candidate models) that optimizes the chosen criterion.

In this paper, we implement stepwise regression by minimizing either Akaike's (1974) An Information Criterion (AIC) or Schwarz's (1978) Bayesian Information Criterion (BIC):

$$\begin{aligned}\text{AIC}(\beta_0, \boldsymbol{\beta}) &= n\ln(\tilde{\sigma}^2) + 2\nu \\ \text{BIC}(\beta_0, \boldsymbol{\beta}) &= n\ln(\tilde{\sigma}^2) + \log(n)\nu\end{aligned} \quad (7)$$

where $\tilde{\sigma}^2 = \boldsymbol{\epsilon}'\boldsymbol{\epsilon}/n = \sum_{i=1}^{n}\epsilon_i^2/n$ is the MLE of the error variance σ^2, and ν is the number of parameters in the model. Adding more predictors will reduce the estimated model error variance (as measured by $\tilde{\sigma}^2$), but will increase the model complexity (as measured by ν). Thus, at each iteration of the stepwise regression algorithm we seek to minimize the AIC or BIC, i.e., find a balance between model fit and parsimony. We note that other criteria could be used in the stepwise regression algorithm, e.g., Mallows's (1973) C_p or Allen's (1974) PRESS statistic. The AIC is optimal when the fit model is an approximation to some unknown true model (Yang, 2005)—which is likely the case with all real data—so the AIC is often the default choice in applications of stepwise regression.

1.3.2 *One-step p-value selection*

Instead of an iterative stepwise regression algorithm, a more crude approach would be to use a single step where predictors with p-values smaller than some predetermined level (e.g, $p < 0.05$) are retained in the model, and all other predictors are removed from the model. We do not recommend this p-value model selection approach, given that the (Type III) p-values with all of the predictors included in the model may not be an accurate representation of the utility of each predictor, particularly if predictors are correlated with one another. Note that the (Type III) p-value for the j-th predictor tests the significance of X_j after including (i.e., conditioning on) the other $p - 1$ predictors, which does not reveal whether X_j should be included in (or excluded from) the regression model. However, we suspect that some authors may consider a variant of this p-value model selection approach, so we include this method in our example.

1.4 *Bias, Variance, and Mean-Squared Error*

Let $\hat{\boldsymbol\theta} = (\hat\beta_0, \hat\beta_1, \dots, \hat\beta_p)'$ and $\boldsymbol\theta = (\beta_0, \beta_1, \dots, \beta_p)'$ denote the $(p+1)\times 1$ vectors containing the OLS estimated and unknown true regression coefficients, respectively. Assuming that the linear model in Equation (1) is the true model, the statistical *bias* of the OLS estimators are given by

$$\text{bias}(\hat{\boldsymbol\theta}) = \text{E}(\hat{\boldsymbol\theta}) - \boldsymbol\theta \tag{8}$$

where $\text{E}(\hat{\boldsymbol\theta})$ denotes the expected value of OLS estimated regression coefficients. In other words, bias is the expected difference between the estimated regression coefficients and the unknown true regression coefficients. An estimator is said to be *unbiased* if the expected value of the estimates is equal to the parameter being estimated (i.e., if bias = 0). It is well known that the OLS coefficients are unbiased estimates of the regression coefficients under the assumptions specified in Equation (1).

The *mean-squared error* of the OLS estimator is given by

$$\text{MSE}(\hat{\boldsymbol\theta}) = \text{E}(\|\hat{\boldsymbol\theta}-\boldsymbol\theta\|^2) = \text{E}([\hat\beta_0-\beta_0]^2)+\sum_{j=1}^{p}\text{E}([\hat\beta_j-\beta_j]^2) \tag{9}$$

which is the total expected squared difference between the estimated and true coefficients. Although not intuitive from its definition, the MSE can be rewritten as

$$\text{MSE}(\hat{\boldsymbol\theta}) = \text{V}(\hat{\boldsymbol\theta}) + \text{bias}^2(\hat{\boldsymbol\theta}) \tag{10}$$

where $\text{V}(\hat{\boldsymbol\theta}) = \sum_{j=0}^{p}\text{V}(\hat\beta_j)$ is the total variance of the regression coefficient vector with $\text{V}(\hat\beta_j)$ denoting the variance of the j-th coefficient, and $\text{bias}^2(\hat{\boldsymbol\theta}) = \sum_{j=0}^{p}[\text{E}(\hat\beta_j) - \beta_j]^2$ is the total squared bias of the regression coefficient vector. Consequently, the MSE quantifies both the accuracy of the estimator (via the bias) and the precision of the estimator (via the variance). An ideal estimator is both accurate (low bias) and precise (low variance). However, as we explain in the next section, we cannot have the best of both worlds: adding bias reduces variance (and vice versa), so we want to find some estimator that provides an optimal balance of variance and bias, i.e., an estimator that minimizes the MSE.

The covariance matrix of $\hat{\boldsymbol\theta}$ has the form $\sigma^2(\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}$ where $\tilde{\mathbf{X}} = [\mathbf{1}_n, \mathbf{X}]$. This implies that $\text{V}(\hat{\boldsymbol\theta}) = \sigma^2\text{tr}((\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1})$ where $\text{tr}(\mathbf{A}) = \sum_{k=1}^{m} a_{kk}$ is the matrix trace function, which is the sum of the diagonal elements of a square matrix. The OLS estimator is unbiased, so the MSE is equal to the variance, i.e., $\text{MSE}(\hat{\boldsymbol\theta}) = \sigma^2\text{tr}((\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1})$. In the following section, we discuss three penalized regression estimators (ridge, lasso, and elastic net) that purposely introduce a bias to the solution with the ultimate goal of reducing the MSE—by reducing the variance—of the estimator. As we will see, introducing a small bias can result in an estimator with better expected performance with respect to MSE, especially when working with small to moderate sized samples. Thus, by reducing the variability of estimators, penalized regression methods have the potential to produce more reproducible results.

2 Penalized Regression

2.1 *Ridge Regression*

When the predictors $X_1, \dots, X_p$ are uncorrelated, the OLS solution will be stable because $\mathbf{X}_c'\mathbf{X}_c$ is diagonal if predictors are uncorrelated. When the predictors in $\mathbf{X}_c$ are highly correlated, the OLS coefficients are unstable because $\mathbf{X}_c'\mathbf{X}_c$ is nearly singular. Ridge regression (Hoerl & Kennard, 1970) is a form of penalized regression that shrinks and stabilizes the coefficient estimates. *Ridge regression* minimizes the PLS function

$$\begin{aligned}\text{PLS}_2(\beta_0, \boldsymbol\beta) &= \text{OLS}(\beta_0, \boldsymbol\beta) + \lambda\text{P}_2(\boldsymbol\beta)\\ &= \sum_{i=1}^{n}\left(y_i - \beta_0 - \textstyle\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\textstyle\sum_{j=1}^{p}\beta_j^2\\ &= \|\mathbf{y} - \mathbf{1}_n\beta_0 - \mathbf{X}\boldsymbol\beta\|^2 + \lambda\boldsymbol\beta'\boldsymbol\beta\end{aligned} \tag{11}$$

where $\text{P}_2(\boldsymbol\beta) = \sum_{j=1}^{p}\beta_j^2$ and $\lambda > 0$ is a *regularization parameter*. Given λ, the coefficients that minimize the ridge PLS function in Equation (11) have the form

$$\begin{aligned}\hat\beta_{0(\lambda)} &= \bar y - \textstyle\sum_{j=1}^{p}\hat\beta_{j(\lambda)}\bar x_j\\ \hat{\boldsymbol\beta}_\lambda &= (\mathbf{X}_c'\mathbf{X}_c + \lambda\mathbf{I}_p)^{-1}\mathbf{X}_c'\mathbf{y}_c\end{aligned} \tag{12}$$

where $\mathbf{X}_c$ and $\mathbf{y}_c$ are defined as they were in Equation (4). Unless units of $X_1, \ldots, X_p$ are comparable, the penalty is typically applied to the standardized regression coefficients such that $\|\boldsymbol{\beta}^*\|^2$ gets closer to zero.

The regularization parameter $\lambda > 0$ controls the trade-off between fitting the data and shrinking the coefficients: $\text{OLS}(\beta_0, \boldsymbol{\beta})$ measures fit and $\text{P}_2(\boldsymbol{\beta})$ measures size. As $\lambda \to 0$ we have $\hat{\boldsymbol{\beta}}_\lambda \to \hat{\boldsymbol{\beta}}$, and as $\lambda \to \infty$ we have $\hat{\boldsymbol{\beta}}_\lambda \to \mathbf{0}_p$. Consequently, as $\lambda \to 0$ the bias decreases and the variance increases, and as $\lambda \to \infty$ the bias increases and the variance decreases. Our goal is to find a λ that provides an optimal balance between fitting the data and shrinking the coefficients, i.e., the λ that minimizes the MSE. Interestingly, Hoerl and Kennard (1970) showed that there *always* exists some $\lambda > 0$ such that the expected MSE of the ridge estimator is less than that of the OLS estimator. Thus, we can always improve the expected performance of the OLS estimator by shrinking the coefficients towards zero.

To choose the regularization parameter, we could consider minimizing the *ordinary cross-validation* (OCV) criterion

$$\text{OCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \beta_{0(\lambda)}^{[i]} - \mathbf{x}_i'\boldsymbol{\beta}_\lambda^{[i]}\right)^2 \tag{13}$$

where $(\beta_{0(\lambda)}^{[i]}, \boldsymbol{\beta}_\lambda^{[i]})$ minimizes $\text{PLS}_2(\beta_0, \boldsymbol{\beta})$ holding out the i-th observation's data. Letting $\hat{\mathbf{y}}_\lambda = \mathbf{H}_\lambda \mathbf{y} = \{\hat{y}_{i(\lambda)}\}_{n\times 1}$ denote the ridge fitted values, where $\hat{y}_{i(\lambda)} = \hat{\beta}_{0(\lambda)} + \mathbf{x}_i'\hat{\boldsymbol{\beta}}_\lambda$, we can rewrite the OCV criterion using the results of the full model fit to all n data points, such as

$$\text{OCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_{i(\lambda)}}{1 - h_{ii(\lambda)}}\right)^2 \tag{14}$$

where $h_{ii(\lambda)}$ is the i-th diagonal of $\mathbf{H}_\lambda = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n' + \mathbf{X}_c(\mathbf{X}_c'\mathbf{X}_c + \lambda\mathbf{I}_p)^{-1}\mathbf{X}_c'$, which is the ridge equivalent of the *hat matrix*, i.e., the matrix defining the linear combination that turns the response Y into the fitted values $\hat{Y}$, see Hastie et al. (2009).

Using the OCV to select λ, the influence of each observation depends on the leverage score $h_{ii(\lambda)}$, which could be quite different. To adjust for this, we could consider a weighted OCV criterion

$$\text{WOCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n} w_i\left(\frac{y_i - \hat{y}_{i(\lambda)}}{1 - h_{ii(\lambda)}}\right)^2 \tag{15}$$

where $w_i > 0$ is the weight assigned to the i-th observation. Setting $w_i = \frac{(1-h_{ii(\lambda)})^2}{[n^{-1}\text{tr}(\mathbf{I}-\mathbf{H}_\lambda)]^2}$ replaces each $h_{ii(\lambda)}$ with its average value, resulting in the *Generalized Cross-Validation* (GCV) (Golub, Heath, & Wahba, 1979) criterion

$$\text{GCV}(\lambda) = \frac{\frac{1}{n}\|(\mathbf{I} - \mathbf{H}_\lambda)\mathbf{y}\|^2}{[1 - \text{tr}(\mathbf{H}_\lambda)/n]^2} = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_{i(\lambda)})^2}{(1 - \nu_\lambda/n)^2} \tag{16}$$

where $\nu_\lambda = \text{tr}(\mathbf{H}_\lambda)$ is the *effective degrees of freedom* of the ridge estimator with regularization parameter λ. The GCV criterion is often preferred over the OCV criterion for a few reasons: (i) the GCV is simpler to compute than the OCV, (ii) the logic of OCV, i.e., "leave-one-out", does not make much sense if there are replicate (or near replicate) predictor scores, i.e., if $\mathbf{x}_h = \mathbf{x}_i$ (or $\mathbf{x}_h \approx \mathbf{x}_i$) for some $h \neq i$, which will likely be the case as the number of points n grows, and (iii) the GCV tends to be "rather more reliable" than the OCV (Ramsay & Silverman, 2005, pg. 97). Thus, the GCV is our preferred method for selecting the ridge regularization parameter.

2.2 *Lasso Regression*

Ridge regression shrinks and stabilizes coefficient estimates, but (for $\lambda < \infty$) all regression coefficients remain non-zero. Consequently, ridge regression is not useful for selecting relevant predictors. In other words, ridge regression will not necessarily help you determine which predictors should or should not be included in a regression model. In contrast, the *least absolute shrinkage and selection operator* (*lasso*) (R. Tibshirani, 1996, 2011) can accomplish the coefficient shrinkage and variable selection simultaneously. Lasso regression minimizes the PLS function

$$\begin{aligned}\text{PLS}_1(\beta_0, \boldsymbol{\beta}) &= \text{OLS}(\beta_0, \boldsymbol{\beta}) + \lambda\text{P}_1(\boldsymbol{\beta}) \\ &= \sum_{i=1}^{n}\left(y_i - \beta_0 - \textstyle\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\textstyle\sum_{j=1}^{p}|\beta_j|\end{aligned} \tag{17}$$

where $\text{P}_1(\boldsymbol{\beta}) = \sum_{j=1}^{p}|\beta_j|$ and $\lambda > 0$ is a regularization parameter. As in the ridge case, unless units of $X_1, \ldots, X_p$ are comparable, the penalty is typically applied to the standardized regression coefficients. Unlike the OLS and ridge PLS problems, there is no closed-form solution (i.e., simple formula) for the optimal lasso regression coefficients. Obtaining the lasso estimate $\hat{\boldsymbol{\beta}}_\lambda$ is a convex optimization problem, which requires an iterative algorithm (see Efron, Hastie, Johnstone, & Tibshirani, 2004; Friedman, Hastie, & Tibshirani, 2010).

Similar to the ridge situation, in lasso regression we still have (i) $\hat{\boldsymbol{\beta}}_\lambda \to \hat{\boldsymbol{\beta}}$ as $\lambda \to 0$, and (ii) $\hat{\boldsymbol{\beta}}_\lambda \to \mathbf{0}_p$ as $\lambda \to \infty$. However, unlike ridge regression, as the lasso regularization parameter increases, some regression coefficients remain non-zero and some will go to zero. See Friedman et al. (2010) for an eloquent explanation of how the lasso penalty forces coefficients to zero via a *soft-thresholding* operator. This key feature of the lasso estimator enables the approach to determine which predictors are most important for explaining the variation in the response variable, which often results in sparser and more easily interpretable regression solutions.

To choose λ, we could consider using the GCV criterion. However, defining the effective degrees of freedom is not as straightforward because the lasso fitted values cannot be written as a simple linear transformation of the response vector. To overcome the issue of needing to define the effective degrees of freedom of the lasso estimator, it is typical to use a K-fold CV procedure to select λ, which begins by randomly splitting the data into $K > 1$ different groups (or folds). Let $(\mathbf{y}_k, \mathbf{X}_k)$ denote k-th fold's data, and let $(\mathbf{y}_{[k]}, \mathbf{X}_{[k]})$ denote the full data set holding out the k-th fold's data. The K-fold CV procedure for lasso regression is carried out as follows:

1. Separately for each fold $k = 1, \ldots, K$
   - fit model using $(\mathbf{y}_{[k]}, \mathbf{X}_{[k]})$ to estimate $(\hat{\beta}_{0(\lambda)}^{[k]}, \hat{\boldsymbol{\beta}}_{\lambda}^{[k]})$
   - define CV-MSE loss for k-th fold as $E_k(\lambda) = \|\mathbf{y}_k - \hat{\beta}_{0(\lambda)}^{[k]}\mathbf{1}_{n_k} - \mathbf{X}_k\hat{\boldsymbol{\beta}}_{\lambda}^{[k]}\|^2$
2. Choose λ that minimizes CV-MSE$(\lambda) = \frac{1}{K}\sum_{k=1}^{K} E_k(\lambda)$

Typical choices for the number of folds include $K = 2$, $K = 5$, or $K = 10$.

2.3 Elastic Net Regression

The lasso performs well in many situations, but it is not without its limitations. In particular, it is well-known that (i) when $n < p$ the lasso can only retain n non-zero coefficients, (ii) when $n > p$ and the predictors are highly correlated, ridge regression tends to outperform the lasso, and (iii) when two predictors are highly correlated, lasso tends to select one predictor and ignore the other (correlated) predictor. The *elastic net* (Zou & Hastie, 2005) combines the lasso (L_1) and ridge (L_2) penalties to address these issues. The elastic net minimizes the PLS function

$$\mathrm{PLS}_{\alpha}(\beta_0, \boldsymbol{\beta}) = \mathrm{OLS}(\beta_0, \boldsymbol{\beta}) + \lambda \mathrm{P}_{\alpha}(\boldsymbol{\beta}) \tag{18}$$

where $\mathrm{P}_{\alpha}(\boldsymbol{\beta}) = \sum_{j=1}^{p}\{\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|\}$, $\lambda > 0$, and $0 \leq \alpha \leq 1$ controls the influence of the L_1 and L_2 penalties. Like in ridge and lasso, the penalty is typically applied to the standardized regression coefficients unless units of $X_1, \ldots, X_p$ are comparable.

In elastic net regression, setting $\alpha = 0$ results in the ridge regression solution, and setting $\alpha = 1$ results in the lasso solution. Setting $0 < \alpha < 1$ uses a hybrid penalty that is a combination of the ridge and lasso. Furthermore, note that setting $\lambda = 0$ produces the OLS solution, so the elastic net provides a flexible umbrella framework through which the previously discussed methods can be viewed. For the elastic net, there are two parameters to tune via K-fold CV: λ and α. In this case, it is typical to select a grid of reasonable α values to try, e.g., $\alpha \in \{0, 0.1, 0.2, \ldots, 0.9, 1\}$, and then use K-fold CV to select an optimal λ for each α. Note that it is necessary to ensure that the random fold assignments are preserved across the different model fittings to ensure that the differences in the CV scores are not due to sampling differences. The pair of regularization parameters $(\hat{\lambda}, \hat{\alpha})$ that minimize the CV score are then used for prediction purposes.

2.4 Summary

We summarize some of the important characteristics of the discussed penalized regression estimators in Table 1. All three methods (i.e., ridge, lasso, elastic net) modify the OLS loss function by incorporating a penalty term related to the size of the regression coefficients. All three discussed penalized regression methods make it possible to build prediction models from data when there are more predictors than subjects (i.e., $n < p$), which is not possible using OLS. By including different penalty terms, it is possible to encourage different types of solutions. Ridge encourages a stabilized solution where the (squared) magnitudes of the coefficients are reduced but still greater than zero. In contrast, lasso and the elastic net can encourage sparse solutions where certain coefficients are entirely removed from the model, which allows for the selection of important predictors. The elastic net offers some advantages over the lasso in certain situations, but these advantages come with the cost of needing to tune the model with respect to two regularization parameters (λ and α). Furthermore, both the lasso and the elastic net do not directly provide any inference information, e.g., for forming confidence intervals around the estimated coefficients. We note that there has been some interesting recent developments with regards to lasso inference (Lockhart, Taylor, Tibshirani, & Tibshirani, 2014; Taylor & Tibshirani, 2015); however, when p-values and/or confidence intervals are needed, the bootstrap is often employed (see Efron & Tibshirani, 1993; R. Tibshirani, 2011).

3 Example Predicting Educational Performance

The supplementary online material contains an R script file (R Core Team, 2016) with all of the analysis code used in this example. The data are freely available online.

3.1 Overview of Data

To demonstrate the different penalized regression methods, we use student performance data that were collected by Paulo Cortez at the University of Minho in Portugal (Cortez & Silva, 2008). The data are available from the University of California-Irvine's Machine Learning repository (Lichman, 2013). In this example, we use the math performance data (student-mat.csv), which contains math exam scores and various predictor variables from $n = 395$ Portuguese students. See Table 2 for the list of 30 predictors, which contain factors relating to the student's behavior, the student's school, the student's family, etc. Unlike Cortez

Table 1 ■ Typical characteristics of different regression estimators.

	OLS	Ridge	Lasso	Elastic Net
Penalty	none	$\lambda \sum_{j=1}^{p} \beta_j^2$	$\lambda \sum_{j=1}^{p} \|\beta_j\|$	$\lambda \sum_{j=1}^{p} \{\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha\|\beta_j\|\}$
Reg. Par.	none	$\lambda > 0$	$\lambda > 0$	$\lambda > 0$ and $0 \leq \alpha \leq 1$
CV Method	none	any	K-fold	K-fold
$n < p$ Case	N/A	applicable	selects $\leq n$	applicable
Collinearity	unstable	stable	selects one	selects both
Strength	MLE	MSE	selection	ridge-lasso hybrid
Weakness	overfitting	no selection	no inference	no inference

Note. Reg. Par. = Regularization Parameter and CV = Cross-Validation. CV procedure "any" refers to any of OCV, GCV or K-fold.

et al., we choose to predict the student's scores on the first exam (instead of the final grade) because we hope to identify factors that cause students to fall behind early in the semester. By discovering factors that relate to poor math performance on the first exam, it may be possible to create student-specific interventions (e.g., tutoring or more study time) with hopes of improving the final grade.

3.2 *Data Preprocessing*

As a preliminary step, we need to load the data file into R (R Core Team, 2016), which can be accomplished using the commands:

```
datapath = "~/Desktop/psych-penreg/
    student-mat.csv"
student = read.table(datapath, sep=";",
    header=TRUE)
```

The first line of code defines the location of the data file (in this case a folder on the Desktop named "psych-penreg"), and the second line of code reads the data into the R environment via the read.table function. The option sep=";" declares the field separator used in the data file (in this case a semi-colon), and the header=TRUE option specifies that the first row of the data file contains variable (header) names.

Next, we load the two R packages that we will use: MASS (Venables & Ripley, 2002) for ridge regression and glmnet (Friedman et al., 2010) for lasso and elastic net regression:

```
library(MASS)
library(glmnet)
```

Note that it is necessary to install the packages before you can load the packages via the library function. This can be accomplished with the install.packages function or manually within the R graphical user interface.

As a next step, we define the response variable and the predictor variables:

```
y = student$G1
n = length(y)
X = model.matrix(~., data=student
    [,1:30])
X = X[,-1]
```

The first line of code creates the response vector, which are the students' scores on the first math exam (i.e., G1). The second line of code defines the sample size, i.e., $n = 395$ students. The third line of code calls R's model.matrix function to create a design matrix for our regression model. The first input of the model.matrix function is a formula specifying the model predictors, which come to the right of the tilde. In this case, we are using R's shorthand notation for "include all predictors" in the input data frame, and we are inputting the first 30 columns of the data frame. The fourth line of code removes the intercept column from the design matrix to avoid redundancy (R's regression functions will include the intercept automatically, so X should only contain predictors). We can now check the dimensions, i.e., number of rows and columns, of the model matrix

```
> dim(X)
[1] 395  39
```

Note that the model matrix has $n = 395$ rows and $p = 39$ columns. The reason that we have more columns in X (than the 30 columns that we input via student[,1:30]) is because there are four categorical predictors with multiple levels (see Table 1), which the model.matrix function automatically converts into dummy coded variables.

3.3 *OLS Model Fitting*

We begin by fitting the OLS regression model via R's lm function:

```
olsmod = lm(y ~ ., data=data.frame(X))
olsmod.sum = summary(olsmod)
olscoef = coef(olsmod)
```

Table 2 ■ Predictor variables for math performance example.

Variable	Type	Range/Levels
Student's School (school)	binary	1=MS, 0=GP
Student's Sex (sex)	binary	1=male, 0=female
Student's Age (age)	integer	15, 16, . . . , 22
Student's Address (address)	binary	1=Urban, 0=Rural
Family Size (famsize)	binary	1=≤ 3 members, 0=> 3 members
Parent Status (Pstatus)	binary	1=living together, 0=living apart
Mother's Education (Medu)	integer	1=none, . . ., 5=higher
Father's Education (Fedu)	integer	1=none, . . ., 5=higher
Mother's Job (Mjob)	categorical	home, teacher, health, services, other
Father's Job (Fjob)	categorical	home, teacher, health, services, other
Reason at School (reason)	categorical	close, reputation, courses, or other
Student's Guardian (guardian)	categorical	mother, father, other
Travel Time to School (traveltime)	integer	1=<15 min, . . ., 4=>60 min
Study Time per Week (studytime)	integer	1=<2 hrs, . . ., 4=>10 hrs
Number of Failures (failures)	integer	0, 1, 2, 3
Extra School Support (schoolsup)	binary	1=yes, 0=no
Extra Family Support (famsup)	binary	1=yes, 0=no
Extra Paid Classes (paid)	binary	1=yes, 0=no
Extra-Curricular Activities (activities)	binary	1=yes, 0=no
Attended Nursery School (nursery)	binary	1=yes, 0=no
Higher Education Interest (higher)	binary	1=yes, 0=no
Internet Access at Home (internet)	binary	1=yes, 0=no
In Romantic Relationship (romantic)	binary	1=yes, 0=no
Quality of Family (famrel)	integer	1=very bad, . . ., 5=excellent
Free Time after School (freetime)	integer	1=very low, . . . , 5=very high
Goes Out with Friends (goout)	integer	1=very low, . . . , 5=very high
Weekday Alcohol Consumption (Dalc)	integer	1=very low, . . . , 5=very high
Weekend Alcohol Consumption (Walc)	integer	1=very low, . . . , 5=very high
Health Status (health)	integer	1=very bad, . . . , 5=very good
Number of Absences (absences)	integer	0, 1, . . . , 75

Note. MS = Mousinho da Silveira and GP = Gabriel Pereira.

The first line of code fits the model, the second line of code creates an object summarizing the fit model, and the third line of code extracts the model coefficients. In the first line, we are (again) using R's shorthand notation to specify that we want to include all variables in the data frame as predictors in the model. To find the predictors that are significant at the $p < 0.05$ level, we can use the R code:

```
ix = which(olsmod.sum$coefficients[-1,4]
     < 0.05)
pvalmod.05 = lm(y ~ ., data=data.frame(X
    [,ix]))
```

where the first line of code determines which predictors are significant at the $p < 0.05$ level, and the second line of code refits the model with the selected predictors. To extract the coefficients we could use a similar procedure as was used to obtain the OLS coefficients, i.e., we could input the pvalmod.05 object into the coef function. However, this would return a vector of length seven because only six predictors (and an intercept) are included in the reduced model. For ease of comparison with the other methods, we instead create a sparse vector (of class dgCMatrix) the same length as olscoef and then place the selected coefficients in the appropriate locations of the initialized vector:

```
pvalcoef.05 = as(matrix(0, length(
    olscoef), 1), "dgCMatrix")
ix = match(names(coef(pvalmod.05)),
    names(olscoef))
pvalcoef.05[ix] = coef(pvalmod.05)
rownames(pvalcoef.05) = names(olscoef)
```

Note that the first line of code initializes the coefficient vector, the second line of code determines which coefficients from the OLS regression model were selected at the

$p < 0.05$ level, the third line of code places the selected coefficients (from the refit model) in the corresponding locations of the pvalcoef.05 vector, and the last line of code assigns names (labels) to the coefficients. We repeated this procedure using $p < 0.15$ to select the predictors, which just involves a simple modification to the previous code:

```
ix = which(olsmod.sum$coefficients[-1,4]
     < 0.15)
pvalmod.15 = lm(y ~ ., data=data.frame(X
    [,ix]))
```

The coefficient vector for the $p < 0.15$ model can be created via a simple modification of the code used for the $p < 0.05$ model, so we omit the code here. See the R script file in the supplementary materials for this omitted R code.

3.4 *Stepwise Model Fitting*

To fit the stepwise regression model, we can use the step function in R:

```
stepmod.aic = step(olsmod)
```

Note that the step function is a "minimal implementation" of the stepAIC function in the MASS package (Venables & Ripley, 2002). By default, the step function performs AIC selection using both forward selection and backward elimination. To extract the coefficients, we use a similar procedure as was used for the pvalmod.05 model (see the supplementary R code). To perform stepwise regression using the BIC (instead of the default AIC), we can use the R code

```
stepmod.bic = step(olsmod, k = log(n))
```

where the k input controls the weight assigned to the model degrees of freedom in the calculation of the information criterion (default is k=2). The coefficient vector for the BIC model can be created via a simple modification of the previously discussed code (see the supplementary R code).

3.5 *Ridge Model Fitting*

To fit the ridge regression model, we can use the lm.ridge function in R's MASS package (Venables & Ripley, 2002):

```
lamseq = seq(0,300,length=1000)
ridgemod = lm.ridge(y ~ ., data=data.
    frame(X), lambda=lamseq)
```

The first line of code defines the sequence of λ values at which the GCV score will be evaluated. The second line of code calls the lm.ridge function to fit the model for each λ in the sequence. When tuning λ it is important to check to see if you have searched a large enough range of values. This can be accomplished by plotting the sequence of λ values against their corresponding GCV values:

```
plot(ridgemod$lambda, ridgemod$GCV, xlab
    ="Lambda", ylab="GCV")
lines(rep(lamseq[which.min(ridgemod$GCV)
    ],2), range(ridgemod$GCV), lty=3)
```

which should produce the plot in Figure 1. Note that we see a clear minimum in the GCV values, which occurs when $\hat{\lambda} = 126.43$. If we were to have set the maximum λ in our sequence too small (e.g., $\lambda = 50$), the GCV minimum would occur on the boundary point, implying that the search sequence needs to be expanded.

To extract the ridge coefficients, we can use the R code:

```
gcvmin = which.min(ridgemod$GCV)
ridgecoef.min = coef(ridgemod)[gcvmin,]
```

The first line of code determines which of the λ values minimized the GCV criterion. The second line of code extracts the ridge coefficients. Note that calling the coef function will return a matrix containing the coefficients for all of the models (one for each λ), so it is necessary to index the optimal coefficient vector using gcvmin.

3.6 *Lasso Model Fitting*

The first step in lasso is to split the data into folds for the K-fold CV:

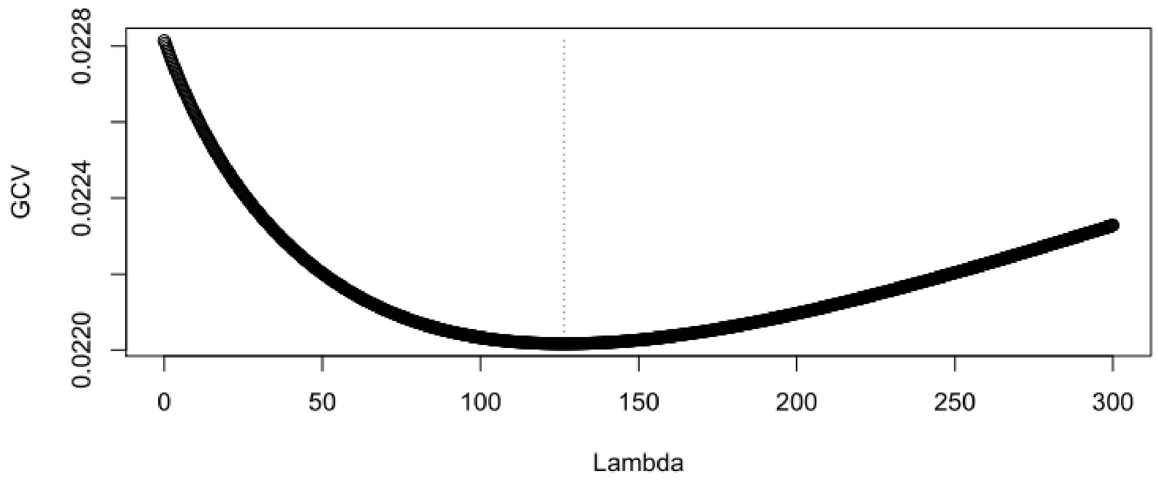
```
set.seed(1)
foldid = sample(rep(1:10, length.out=n))
```

The first line of code sets the random seed in R so that the random sampling result can be reproduced; the number input to the set.seed function is not important, but each different value will produce a different random sample. The second line randomly splits the data into $K = 10$ folds by creating a vector of integers $1, \ldots, 10$ the length of the response variable, and then randomly sampling the fold assignment for each student. Given the fold assignments, the lasso regression model can be fit using the cv.glmnet function in the glmnet R package (Friedman et al., 2010) using the following code:

```
cvlasso = cv.glmnet(X, y, foldid=foldid,
     alpha=1)
```

The first input to the cv.glmnet function is the model design matrix, and the second input is the response vector. The foldid input controls the fold assignments used for the K-fold CV procedure, and the alpha input controls the α used in the elastic net penalty ($\alpha = 1$ corresponds to lasso). To visualize the performance of the lasso estimator for different values of λ, we can create a plot similar to that created for ridge regression (see Figure 1). In this case, we plot the natural logarithm of the regularization parameter, i.e., $\log(\lambda)$, versus the CV-MSE estimate obtained from the K-fold CV procedure, see Figure 2:

Figure 1 ■ Plot of λ versus GCV(λ) for ridge regression. The dotted line denotes the value of λ that minimizes the GCV criterion.



```
plot(cvlasso)
```

As is evident from Figure 2, there is a clear minimum at $\hat{\lambda} = 0.095$ ($\log(\hat{\lambda}) = -2.350$). However, the plot also displays another line at $\hat{\lambda}_* = 0.242$ ($\log(\hat{\lambda}_*) = -1.420$), which is the largest value of λ that is within one standard error of the minimum CV-MSE. Theoretically, the minimum $\hat{\lambda}$ should be preferred, but $\hat{\lambda}_*$ will encourage sparser solutions. To obtain the two possible sets of coefficients (corresponding to $\hat{\lambda}$ and $\hat{\lambda}_*$), we can use the R code:

```
lassocoef.min = coef(cvlasso, s="lambda.
    min")
lassocoef.1se = coef(cvlasso, s="lambda
    .1se")
```

3.7 *Elastic Net Model Fitting*

Similar to the lasso model fitting procedure, the first step in elastic net is to split the data into folds for the K-fold CV. This can be accomplished using the same code as before, and using the same random seed will ensure that the fold assignments are comparable for the lasso and elastic net solutions. The next step is to define a sequence of α values, which can be accomplished using the R code:

```
alphaseq = seq(0, 1, length=21)
```

In this case, we have created a vector of 21 α values: 0, 0.05, 0.1, . . ., 0.95, 1. Next we need to fit the model by calling the $\mathrm{cv.glmnet}$ function separately for each value in $\mathrm{alphaseq}$. This can be accomplished using a simple for loop in R:

```
cvlist = vector("list", length(alphaseq)
    )
for(k in 1:length(alphaseq)){
  cvlist[[k]] = cv.glmnet(X, y, foldid=
    foldid, alpha=alphaseq[k])
}
```

The fist line of code initializes a list named cvlist of length 21, which will hold the model fitting results for each of the 21 values in $\mathrm{alphaseq}$. The remaining lines of code loop through the 21 values in $\mathrm{alphaseq}$ fitting the model via the $\mathrm{cv.glmnet}$ function.

To determine which combination of (λ, α) minimized the CV-MSE criterion, we need to collect and compare the CV-MSE results such as:

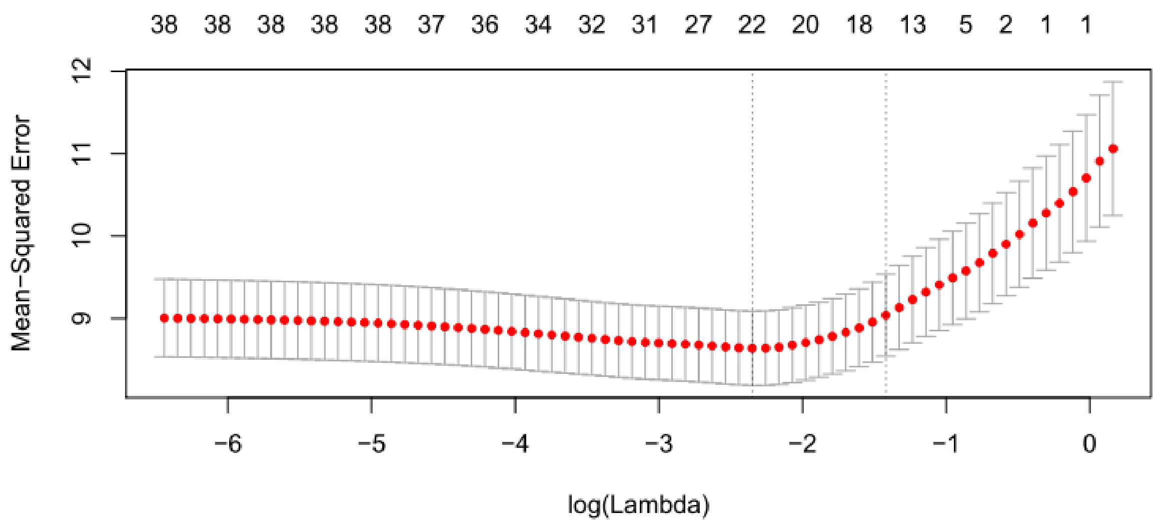
```
mincv = sapply(cvlist, function(x) min(x
    $cvm))
minid = which.min(mincv)
```

This first line of code collects the minimum (across λ) CV-MSE estimate from each of the 21 models, and the second line of code finds the minimum (across α). To visualize the elastic net fitting results, we can plot two different CV objects, see Figure 3:

```
plot(alphaseq, mincv, xlab="Alpha", ylab
    ="Mean-Squared␣Error", type="b")
plot(cvlist[[minid]])
```

The first line plots the minimum CV-MSE for each α, and the second line plots the CV-MSE results for different λ values with $\hat{\alpha} = 0.05$ set at the optimal value. We can extract the elastic net coefficients for the optimal α value using the R code:

Figure 2 ■ Plot of $\log(\lambda)$ versus CV-MSE(λ) for lasso regression. The left dotted line denotes the value of λ that minimizes the CV-MSE, and the right dotted line denotes the largest value of λ that is within one standard error of the minimum CV-MSE.



```
enetcoef.min = coef(cvlist[[minid]], s="
    lambda.min")
enetcoef.1se = coef(cvlist[[minid]], s="
    lambda.1se")
```

3.8 *Summarizing the Results*

The estimated coefficients from each model are given in Tables 3 and 4. Comparing the OLS and ridge models, the shrinkage is evident for several coefficients. For example, in the OLS model the male students are expected to have 0.9 more points on the math exam than female students, whereas the expected sex effect is only 0.6 points in the ridge model. However, note that the OLS and ridge models both retain all coefficients in the model, i.e., all coefficients have non-zero estimates. In contrast, the other approaches produce sparser solutions than the OLS and ridge models by setting some of the coefficients to zero. The p-value selection methods produced the sparest models, and the stepwise regression methods produced models slightly larger than the p-value methods. As expected, (i) smaller p-values encouraged sparser models (for the p-value selection method), and (ii) the BIC selected a sparser model than the AIC.

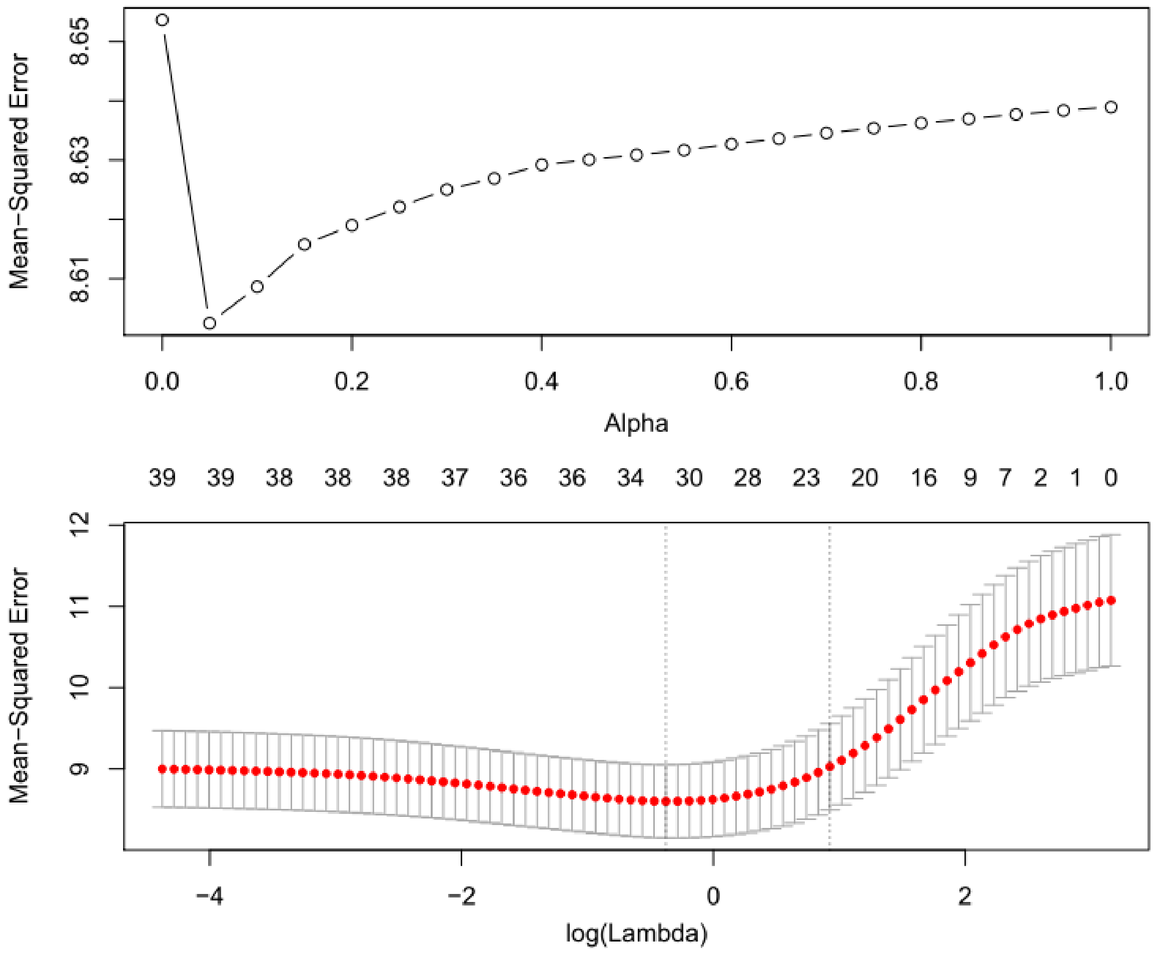
The lasso and elastic net selected larger models than the p-value and stepwise regression approaches. As expected, the lasso and elastic net models using $\hat{\lambda}_*$ (1 SE above minimum CV-MSE) produce sparser solutions than the models using $\hat{\lambda}$ (minimum CV-MSE). The selection results for the lasso and elastic net solutions are mostly similar, with the elastic net selecting the same predictors as the lasso plus some additional predictors. Some of the factors that relate to better expected math performance include: being male, having a father who is a teacher, having a mother who works in healthcare, studying more each week, and having an interest in higher education. Some of the factors that relate to poorer expected math performance include: having more past course failures, needing extra educational support (at school or home), and going out more with friends.

3.9 *Evaluating the Methods*

In this section, we conduct an auxiliary simulation study to further demonstrate the power of penalized regression. We want to emphasize that a typical application of penalized regression would include everything that we have demonstrated up to this point, but would *not* include the sort of simulation analysis that we conduct here. We conduct this auxiliary simulation study just to emphasize that penalized regression methods can be more useful than unpenalized (OLS and stepwise) estimation when working with psychological data.

With real data, the ground truth is unknown so it is typical to evaluate the performance of regression models by splitting the data into a training dataset (which is used to build the model) and the testing dataset (which is used to evaluate the model). This can be accomplished in R using the following code:

Figure 3 ■ Results for elastic net regression. Top: minimum CV-MSE for each α. Bottom: $\log(\lambda)$ versus CV-MSE(λ) with $\hat{\alpha} = 0.05$ set at the optimal value.



```
set.seed(55455)
testID = sample.int(n, 95L)
ytest = y[testID]
Xtest = X[testID,]
ytrain = y[-testID]
Xtrain = X[-testID,]
```

The first line (again) calls the set.seed function to ensure that the random sampling results are reproducible. The second line of code randomly samples 95 students (out of the 395) to be in the testing dataset. The third and fourth lines of code define the testing dataset (ytest, Xtest) by indexing the sampled rows of the full dataset. Finally, the fifth and sixth lines of code define the training dataset (ytrain, Xtrain) by indexing the non-sampled rows of the full dataset.

We used the model fitting procedure described in the previous subsections, but we replaced X and y with Xtrain and ytrain. To evaluate the performance of each method, we can calculate the mean-squared prediction error (MSPE). Letting $(\mathbf{y}_{\text{test}}, \mathbf{X}_{\text{test}})$ denote the response and design matrix for the $\tilde{n}$ test data points ($\tilde{n} = 95$ for our testing dataset), the MSPE is defined as

$$\text{MSPE}(\hat{\boldsymbol{\theta}}, \mathbf{y}_{\text{test}}, \mathbf{X}_{\text{test}}) = \tilde{n}^{-1} \|\mathbf{y}_{\text{test}} - \hat{\mathbf{y}}_{\text{test}}\|^2$$

where $\hat{\mathbf{y}}_{\text{test}} = \mathbf{1}\hat{\beta}_0 + \mathbf{X}_{\text{test}}\hat{\boldsymbol{\beta}}$ are the predictions for the test data with $(\hat{\beta}_0, \hat{\boldsymbol{\beta}})$ denoting the coefficients estimated from the training data. To obtain the MSPE for each of the fit models, we can use R code along the lines of

```
yhat.ols = cbind(1,Xtest) %*% olscoef
mse.ols = mean( (ytest - yhat.ols)^2 )
```

For the other methods, the code is nearly identical; we just need to replace olscoef with the coefficients for a different model (e.g., ridgecoef.min or lassocoef.min) in the above code. We omit the code here to save space, but the R code for forming the MSPE for each model is provided with the supplementary material.

To examine the variability of the MSPE due to random sampling (when forming the training and testing datasets), we can simply repeat the entire procedure using multiple different random samples. We did this for 100 different random splits of the data into training and testing datasets. Figure 4 displays a box plot of the MSPE across the 100 splits for each of the 10 different methods. The average MSPE (across the 100 random splits) for each method is given by

```
> meanmse
      ols     p0.05     p0.15  step.aic
 9.412707  9.313102  9.291680  9.115387
 step.bic     ridge lasso.min lasso.1se
 9.318641  8.863382  8.765813  9.369715
 enet.min  enet.1se
 8.777928  9.314690
```

These average MSPE values and the boxplots in Figure 4 reveal that the lasso and elastic net using $\hat{\lambda}$ (minimizer of CV-MSE) perform best across the 100 random splits of the data. The ridge performs next best and produces MSPEs that are only slightly larger than the lasso and elastic net solutions. The stepwise regression algorithm (with AIC selection) was the best unpenalized estimator, but the MSPEs were noticeably larger than the corresponding MSPEs of the (minimum) penalized regression methods. The p-value selection methods and the lasso and elastic net using $\hat{\lambda}_*$ (1 SE above minimizer of CV-MSE) only performed slightly better than the OLS solution, which tended to perform worse than all of the other methods.

For each of the 100 splits, we can determine which of the 10 estimators had the smallest MSPE. The total number of times (across the 100 splits) that each method was best (i.e., smallest MSPE) is given below:

```
> prctbest
      ols     p0.05     p0.15  step.aic
        1         5         9        11
 step.bic     ridge lasso.min lasso.1se
       10        13        22         8
 enet.min  enet.1se
       14         7
```

which reveals that the lasso with $\hat{\lambda}$ performed the best most frequently (22% of the time), followed by the elastic net with $\hat{\lambda}$ (14% of the time), the ridge (13% of the time), and stepwise regression with AIC (11% of the time) or BIC (10% of the time). The lasso and elastic net with $\hat{\lambda}_*$ were rarely the best method (8% and 7%, respectively), and the p-value selection methods with $p < 0.05$ and $p < 0.15$ showed a similarly poor performance (5% and 9%, respectively), Finally, note that the OLS solution was best only 1% of the time. This confirms that the penalized regression methods reliably outperform the classic OLS estimator with respect to predicting math performance. For these data, the lasso solution should be preferred, but the elastic net and ridge also performed well.

We also want to emphasize one point regarding the relation between the MSPE and the CV-MSE. Comparing Figures 2 and 3 with Figure 4, we note that the CV-MSE provides an excellent estimate of the MSPE—without the hassle of having to split the data and fit each model to 100 different training datasets. Thus, the CV-MSE is able to estimate the MSPE with (substantially) less computationally work. This provides some insight into why we want to find regularization parameters that minimize the CV-MSE criterion: these parameters should also minimize the MSPE. Finally, we want to highlight the dangers of using a single sample split to evaluate the models. Using the training and testing data corresponding to the original sample split, i.e., the split defined above with set.seed(55455), the obtained MSPE values are

```
> msetab[1,]
      ols     p0.05     p0.15  step.aic
 7.942079  8.266810  8.344716  7.960406
 step.bic     ridge lasso.min lasso.1se
 8.533376  7.075778  7.106077  7.093757
 enet.min  enet.1se
 7.072657  7.065903
```

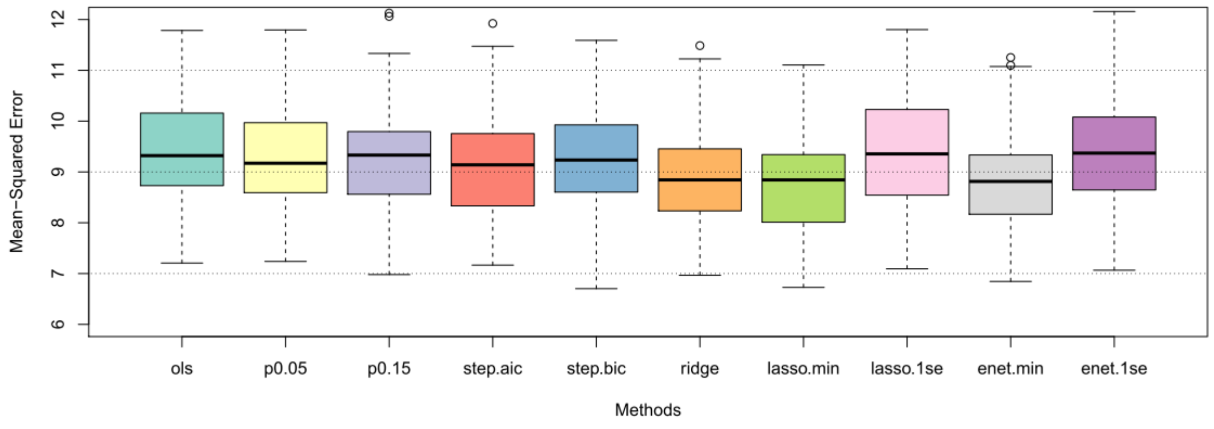
which are overly optimistic, i.e., MSPE ≈ 7 is an underestimate of the typical MSPE (≈ 9), see Figure 4. Also, note that the p-value and stepwise regression methods actually produce a larger MSPE than the OLS solution with this sample split. This emphasizes that models selected via p-value or stepwise methods are not guaranteed to cross-validate any better than the original OLS model with all of the predictors included.

4 Discussion

4.1 *Take Home Points*

In this tutorial, we discuss the potential of three popular penalized regression methods (ridge, lasso, and elastic net) for psychological research. The GLM is one of the primary statistical tools in psychology, and many researchers rely on OLS estimation when estimating the parameters of

Figure 4 ■ Box plots of mean-squared prediction error across 100 random splits of the data.



GLMs. Our tutorial clearly demonstrates that penalized regression methods can and should be considered as alternatives to classic OLS estimation when fitting GLMs. Penalized regression methods modify the classic OLS loss function by attaching a penalty term, which induces a certain bias to the solution. By encouraging different types of biases to the estimation, we can obtain solutions that are often more stable, reliable, and/or interpretable. In all three of the discussed penalized regression methods, the bias is designed to encourage solutions of "no effect", i.e., solutions where the estimated regression coefficients are closer in magnitude to zero. The three methods differ with respect to the type of bias they impose, which determines the characteristics of the estimator (see Table 1).

Despite the negative connotations of the word "bias", our tutorial has demonstrated that biased estimators can be a good thing.

Ridge regression is useful for shrinking and stabilizing OLS solutions, but will not help select which predictors are important. In contrast, the lasso can accomplish both the coefficient shrinkage and the variable selection simultaneously, making it a useful choice when one wants to determine a parsimonious prediction model from a large set of potential predictors. However, the lasso does have some restrictions when fitting models where $n < p$ or models with highly correlated predictors. The elastic net uses a penalty that is a hybrid of the ridge and lasso penalties, which allows it to overcome some of the lasso's limitations. But, in any real data situation, it is not always clear whether the lasso or elastic net should be preferred. Furthermore, the elastic net has the additional regularization parameter α, which increases the computational burden of the problem. So the real question is whether or not the tuning of the extra parameter is worth the effort in terms of prediction gains.

Our real data example clearly reveals the potential of penalized regression methods for predicting psychological outcomes (math performance in this case). Using a step-by-step tutorial with R code, we demonstrated how to (i) load and prepare the data for analysis, (ii) fit the OLS, stepwise, ridge, lasso, and elastic net models, (iii) extract and compare the model fitting results, and (iv) evaluate the predictive performance of the methods. Our example demonstrates that all three penalized regression methods reliably outperform the classic OLS and stepwise regression coefficients with respect to minimizing the MSPE. For these particular data, the lasso performed the best; however, the elastic net performed very similarly to the lasso, and the performance of the ridge was not much worse than that of the lasso and elastic net. We want to emphasize that the superior MSPE performance of the lasso was achieved using fewer predictors than the OLS, ridge, and elastic net models, which highlights the lasso's ability to build parsimonious and powerful prediction models.

4.2 Some Extensions

4.2.1 Generalized Linear Models

In this paper, we focused on the classic GLM where the response follows a normal distribution. However, the penalized regression estimators discussed in this paper can be easily extended to any Generalized Linear Model (GzLM; McCullagh & Nelder, 1989) such as logistic or Poisson regression (see Friedman et al., 2010). In this case, the same penalty term is used for each method (see Table 1). However, we no longer add the penalty term to the OLS loss function. Note that GzLMs are typically fit by maximizing the log-likelihood function corresponding to the assumed response variable distribution. And note that maximizing the log-likelihood function is equivalent to minimizing the negative of the log-likelihood function. For penalized GzLMs, the penalty term is added to the negative of the log-likelihood function, which is then minimized to estimate the regression coefficients. This is referred to as penalized likelihood estimation.

4.2.2 *Smoothing Spline Models*

The GLM discussed in this paper is a form of parametric regression, where the relationship between Y and $X_1, \ldots, X_p$ is assumed to take some predetermined form that depends on the unknown parameters in $\boldsymbol{\theta}$. In contrast, a smoothing spline model is a form of nonparametric regression (see Craven & Wahba, 1979; Gu, 2013; Hastie et al., 2009; Ramsay & Silverman, 2005; Ruppert, Wand, & Carroll, 2003; Silverman, 1985; Wahba, 1990; Wang, 2011; Wood, 2006). Unlike a parametric regression model, a nonparametric regression model does not assume that the relationship between Y and $X_1, \ldots, X_p$ has some known form involving a finite number of model parameters. Instead, the goal in nonparametric regression is to estimate the form of the relationship between Y and $X_1, \ldots, X_p$. Similar to the PLS methods discussed in this paper, a smoothing spline introduces a penalty term that encourages smoother estimates of the function to avoid overfitting. The goal is to find a balance between fitting the data (measured by OLS fit) and smoothing the function (measured by the smoothness penalty).

4.2.3 *Bootstrap Resampling*

As previously mentioned, one of the primary limitations of the lasso and elastic net is the lack of inference information obtained with the solution. That is, given the optimal lasso or elastic net model, we do not obtain any information (e.g., standard errors of coefficients) that can be used to form confidence intervals around the regression coefficients. Although there has been some recent developments for lasso inference (Lockhart et al., 2014; Taylor & Tibshirani, 2015), the nonparametric bootstrap (Efron, 1979; Efron & Tibshirani, 1993) is still often used for inference in applications of penalized regression. The bootstrap involves resampling (with replacement) the data to create B different samples of data each of size $n^* \leq n$. Typically $B \geq 10,000$ and $n^* = n$ unless the full sample size n is very large. The regression model is then fit separately to each of the B samples of data to obtain B different replicates of the regression coefficient estimates. The distribution of the coefficient estimates across the B replicates can be used to estimate the standard errors of and form confidence intervals for regression coefficients (see Efron & Tibshirani, 1993).

4.2.4 *Bootstrap Aggregating (Bagging)*

When prediction is of primary concern, the bootstrap can be used to combine prediction results from multiple models, which often results in better predictions (see Breiman, 1996). Bootstrap aggregating (or bagging) involves (i) resampling the data with replacement to create B different samples of data, and (ii) fitting the regression model separately to each of the B samples. Instead of using the bootstrap distribution of the coefficients for inference purposes, we could consider using the bootstrap distribution for prediction purposes, i.e., “bagging” our predictions. By aggregating (or averaging) the predictions from the B models, we can often obtain a prediction result that is more reliable and accurate than the result that would have been obtained via only fitting the model to the original dataset.

4.3 *Concluding Remarks*

Despite the negative connotations of the word “bias”, our tutorial has demonstrated that biased estimators can be a good thing. In particular, we have demonstrated that biased regression estimators can outperform classic OLS estimators with respect to predicting psychologically relevant outcomes. Unlike OLS estimators (which are based on large sample maximum likelihood theory) penalized regression estimators use data-driven, cross-validation routines in attempt to create reliable prediction models. Furthermore, using the lasso or elastic net, it is possible to select which predictors are most important for regression models, which can provide a more parsimonious interpretation of the model. If the goal is to obtain valid—not just significant—estimates of regression coefficients, then haphazard applications of OLS regression should be avoided. Instead, cross-validation oriented penalized regression estimators should be preferred, because such estimators encourage more parsimonious and reproducible research results.

Authors’ note

I thank Denis Cousineau for helpful comments on an early draft of this paper.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723. doi:10.1109/TAC.1974.1100705

Allen, D. M. (1974). The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, *16*, 125–127. doi:10.1080/00401706.1974.10489157

Azen, R. & Budesco, D. (2009). Applications of multiple regression in psychological research. In R. E. Millsap & A. Maydeu-Olivares (Eds.), *The sage handbook of quantitative methods in psychology* (pp. 285–310). London: SAGE. doi:10.4135/9780857020994.n13

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140. doi:10.1023/A:1018054314350

Chartier, S. & Faulkner, A. (2008). General Linear Models: an integrated approach to statistics. *Tutorial in Quantitative Methods for Psychology*, *4*, 65–78. doi:10.20982/tqmp.04.2.p065

Cohen, J. (1992). A power primer. *Psychological Bulletin*, *112*, 155–159. doi:10.1037/0033-2909.112.1.155

Cortez, P. & Silva, A. (2008). Using data mining to predict secondary school student performance. In A. Brito & J. Teixeira (Eds.), *Proceedings of 5th FUture BUsiness TEChnology conference (FUBUTEC 2008)* (pp. 5–12).

Cousineau, D. (2005). The rise of quantitative methods in psychology. *Tutorial in Quantitative Methods for Psychology*, *1*, 1–3. doi:10.20982/tqmp.01.1.p001

Craven, P. & Wahba, G. (1979). Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, *31*, 377–403. doi:10.1007/BF01404567

Efron, B. (1979). Bootstrap methods: another look at the Jackknife. *Annals of Statistics*, *7*, 1–26. doi:10.1214/aos/1176344552

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, *34*, 407–499. doi:10.1214/009053604000000067

Efron, B. & Tibshirani, R. (1993). *An introduction to the bootstrap*. Boca Raton: Chapman & Hall.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*, 1–22. doi:10.18637/jss.v033.i01

Golub, G. H., Heath, M., & Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, *21*, 215–223. doi:10.1080/00401706.1979.10489751

Gu, C. (2013). *Smoothing spline ANOVA models* (Second). New York: Springer-Verlag.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. New York: Springer-Verlag.

Hoerl, A. & Kennard, R. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, *12*, 55–67. doi:10.1080/00401706.1970.10488634

Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized structural equation modeling. *Structural Equation Modeling*, *23*, 555–566. doi:10.1080/10705511.2016.1154793

Kelley, K. & Maxwell, S. E. (2003). Sample size for multiple regression: obtaining regression coefficients that are accurate, not simply significant. *Psychological Methods*, *8*, 305–321. doi:10.1037/1082-989X.8.3.305

Lichman, M. (2013). UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml

Lockhart, R., Taylor, J., Tibshirani, R., & Tibshirani, R. (2014). A significance test for the lasso. *Annals of Statistics*, *42*, 413–468. doi:10.1214/13-AOS1175

Mallows, C. L. (1973). Some comments on C_p. *Technometrics*, *15*, 661–675. doi:10.2307/1271437

McCullagh, P. & Nelder, J. A. (1989). *Generalized linear models* (Second). London: Chapman and Hall.

McNeish, D. M. (2015). Using lasso for predictor selection and to assuage overfitting: a method long overlooked in behavioral sciences. *Multivariate Behavioral Research*, *50*, 471–484. doi:10.1080/00273171.2015.1036965

R Core Team. (2016). *R: a language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. Retrieved from http://www.R-project.org/

Ramsay, J. O. & Silverman, B. W. [Bernard W.]. (2005). *Functional data analysis* (Second). New York: Springer.

Ruppert, D., Wand, M. P., & Carroll, R. J. (2003). *Semiparametric regression*. Cambridge: Cambridge University Press.

Schwarz, G. E. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–464. doi:10.1214/aos/1176344136

Silverman, B. W. [B. W.]. (1985). Aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society, Series B*, *47*, 1–52. doi:10.2307/2345542

Taylor, J. & Tibshirani, R. J. (2015). Statistical learning and selective inference. *Proceedings of the National Academy of Sciences*, *112*, 7629–7634. doi:10.1073/pnas.1507583112

Tibshirani, R. (1996). Regression and shrinkage via the lasso. *Journal of the Royal Statistical Society, Series B*, *58*, 267–288. doi:10.2307/2346178

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society, Series B*, *73*, 273–282. doi:10.1111/j.1467-9868.2011.00771.x

Venables, W. N. & Ripley, B. D. (2002). *Modern applied statistics with S* (Fourth). ISBN 0-387-95457-0. New York:

Springer. Retrieved from http://www.stats.ox.ac.uk/pub/MASS4

Wahba, G. (1990). *Spline models for observational data*. Philadelphia: Society for Industrial and Applied Mathematics.

Wang, Y. (2011). *Smoothing splines: methods and applications*. Boca Raton: CRC Press.

Wood, S. N. (2006). *Generalized additive models: an introduction with R*. Boca Raton: Chapman & Hall.

Yang, Y. (2005). Can the strengths of AIC and BIC be shared? a conflict between model identification and regression estimation. *Biometrika*, *92*, 937–950. doi:10.1093/biomet/92.4.937

Zou, H. & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, *67*, 301–320. doi:10.1111/j.1467-9868.2005.00503.x

Open practices

The *Open Material* badge was earned because supplementary material(s) are available on the journal's web site.

Citation

Helwig, N. E. (2017). Adding bias to reduce variance in psychological results: A tutorial on penalized regression. *The Quantitative Methods for Psychology*, *13*(1), 1–19. doi:10.20982/tqmp.13.1.p001

Received: 01/01/2016 ∼ Accepted: 23/11/2016

Tables 3 and 4 follow on next pages.

Table 3 ■ Estimated coefficients for unpenalized regression models.

	OLS	$p < 0.05$	$p < 0.15$	Step-AIC	Step-BIC
(Intercept)	11.375	11.466	10.737	10.868	10.932
schoolMS	0.010	.	.	.	.
sexM	0.894	0.809	0.849	0.846	0.770
age	-0.070	.	.	.	.
addressU	0.151	.	.	.	.
famsizeLE3	0.429	.	.	.	.
PstatusT	0.154	.	.	.	.
Medu	0.118	.	.	.	.
Fedu	0.144	.	.	.	.
Mjobhealth	0.926	.	.	1.782	1.852
Mjobother	-0.782	.	-0.986	.	.
Mjobservices	0.467	.	.	1.237	1.237
Mjobteacher	-0.923	.	.	.	.
Fjobhealth	-0.553	.	.	.	.
Fjobother	-1.135	.	-0.642	-1.036	.
Fjobservices	-0.994	.	.	-0.910	.
Fjobteacher	1.187	.	.	1.352	2.229
reasonhome	0.166	.	.	.	.
reasonother	-0.181	.	.	.	.
reasonreputation	0.444	.	.	.	.
guardianmother	0.050	.	.	.	.
guardianother	0.866	.	.	.	.
traveltime	-0.025	.	.	.	.
studytime	0.605	0.611	0.573	0.625	0.678
failures	-1.314	-1.445	-1.326	-1.321	-1.425
schoolsupyes	-2.155	-1.947	-1.947	-2.117	-2.016
famsupyes	-0.979	-0.596	-0.770	-0.895	-0.843
paidyes	-0.102	.	.	.	.
activitiesyes	-0.053	.	.	.	.
nurseryyes	0.030	.	.	.	.
higheryes	1.141	.	1.635	1.298	.
internetyes	0.255	.	.	.	.
romanticyes	-0.211	.	.	.	.
famrel	0.026	.	.	.	.
freetime	0.255	.	0.251	0.249	.
goout	-0.414	-0.349	-0.404	-0.429	-0.373
Dalc	-0.063	.	.	.	.
Walc	-0.025	.	.	.	.
health	-0.168	.	-0.179	-0.231	.
absences	0.012	.	.	.	.

Note. An entry of . indicates that a variable was not selected by a given method.

Table 4 ■ Estimated coefficients for penalized regression models.

	Ridge	Lasso ($\hat{\lambda}$)	Lasso ($\hat{\lambda}_*$)	E-Net ($\hat{\lambda}$)	E-Net ($\hat{\lambda}_*$)
(Intercept)	10.398	10.268	10.945	10.133	10.370
schoolMS	0.007	.	.	.	.
sexM	0.619	0.589	0.171	0.627	0.280
age	-0.025	.	.	0.000	.
addressU	0.129	0.066	.	0.115	0.028
famsizeLE3	0.356	0.214	.	0.318	0.129
PstatusT	0.037	.	.	.	.
Medu	0.121	0.012	0.069	0.101	0.106
Fedu	0.139	0.120	0.025	0.132	0.101
Mjobhealth	0.758	0.994	0.259	0.780	0.393
Mjobother	-0.551	-0.399	-0.401	-0.531	-0.325
Mjobservices	0.432	0.624	0.176	0.453	0.215
Mjobteacher	-0.396	.	.	-0.332	.
Fjobhealth	-0.032	.	.	.	.
Fjobother	-0.434	-0.159	-0.041	-0.378	-0.140
Fjobservices	-0.310	.	.	-0.228	.
Fjobteacher	1.222	1.502	0.953	1.284	0.731
reasonhome	0.079	.	.	0.031	.
reasonother	-0.087	.	.	-0.047	.
reasonreputation	0.351	0.266	0.030	0.326	0.161
guardianmother	-0.075	.	.	-0.030	.
guardianother	0.379	0.284	.	0.337	.
traveltime	-0.072	.	.	-0.047	-0.027
studytime	0.420	0.453	0.228	0.430	0.226
failures	-0.973	-1.217	-1.113	-1.049	-0.687
schoolsupyes	-1.620	-1.770	-1.378	-1.664	-1.004
famsupyes	-0.701	-0.680	-0.273	-0.705	-0.320
paidyes	-0.058	.	.	-0.009	.
activitiesyes	0.013	.	.	.	.
nurseryyes	0.062	.	.	.	.
higheryes	1.060	0.879	0.385	1.038	0.696
internetyes	0.206	0.015	.	0.160	0.018
romanticyes	-0.145	.	.	-0.087	.
famrel	0.022	.	.	.	.
freetime	0.156	0.111	.	0.147	0.010
goout	-0.294	-0.291	-0.149	-0.298	-0.154
Dalc	-0.060	.	.	-0.038	-0.010
Walc	-0.076	-0.053	.	-0.073	-0.059
health	-0.123	-0.101	.	-0.115	-0.039
absences	0.004	.	.	0.001	.

Note. An entry of . indicates that a variable was not selected by a given method.