10 Shrinkage Estimation

```
library(glmnet)
```

Shrinkage estimation is a highly valuable technique in the context of high-dimensional regression analysis. It allows for the estimation of regression models with more regressors than observations.

10.1 Mean squared error

The key measure of estimation accuracy is the **mean squared error (MSE)**. The MSE of an estimator $\hat{\theta}$ for a parameter θ is

$$MSE(\hat{\theta}) = E[(\hat{\theta} - \theta)^2].$$

The MSE can be decomposed into the variance plus squared bias:

$$MSE(\hat{\theta}) = \underbrace{E[(\hat{\theta} - E[\hat{\theta}])^2]}_{=Var[\hat{\theta}]} + \underbrace{(E[\hat{\theta}] - \theta)^2}_{=Bias(\hat{\theta})^2}$$

Proof. Subtracting and adding $E[\hat{\theta}]$ gives

$$\begin{aligned}(\hat{\theta} - \theta)^2 &= (\hat{\theta} - E[\hat{\theta}] + E[\hat{\theta}] - \theta)^2 \\ &= (\hat{\theta} - E[\hat{\theta}])^2 + 2(\hat{\theta} - E[\hat{\theta}])(\underbrace{E[\hat{\theta}] - \theta}_{Bias(\hat{\theta})}) + \underbrace{(E[\hat{\theta}] - \theta)^2}_{=Bias(\hat{\theta})^2}.\end{aligned}$$

The middle term is zero after taking the expectation:

$$E[(\hat{\theta} - \theta)^2] = \underbrace{E[(\hat{\theta} - E[\hat{\theta}])^2]}_{=Var[\hat{\theta}]} + 2\underbrace{E[\hat{\theta} - E[\hat{\theta}]]}_{=0} Bias(\hat{\theta}) + Bias(\hat{\theta})^2.$$

□ □

For instance, consider an i.i.d. sample $X_1, \dots, X_n$ with population mean $E[X_i] = \mu$ and variance $Var[X_i] = \sigma^2$. Let's study the sample mean

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

as an estimator of μ. You will find that

$$E[\hat{\mu}] = \mu, \quad Var[\hat{\mu}] = \frac{\sigma^2}{n}.$$

Proof. By the linearity of the expectation, we have

$$E[\hat{\mu}] = \frac{1}{n} \sum_{i=1}^{n} \underbrace{E[X_i]}_{\mu} = \mu.$$

The independence of $X_1, \dots, X_n$ implies

$$Var[\hat{\mu}] = \frac{1}{n^2} Var\left[\sum_{i=1}^{n} X_i\right] = \frac{1}{n^2} \sum_{i=1}^{n} Var[X_i] = \frac{\sigma^2}{n}$$

□ □

The sample mean is unbiased for μ, i.e., $Bias(\hat{\mu}) = E[\hat{\mu}] - \mu = 0$. The MSE equals its variance:

$$MSE(\hat{\mu}) = \frac{\sigma^2}{n}.$$

The sample mean is the best unbiased estimator for the population mean, but there exists estimators with a lower MSE if we allow for a small bias.

10.2 A simple shrinkage estimator

Let us shrink our sample mean a bit towards 0 and define the alternative estimator

$$\tilde{\mu} = (1 - w)\hat{\mu}, \quad w \in [0, 1].$$

Setting the shrinkage weight to $w = 0$ gives $\tilde{\mu} = \hat{\mu}$ (no shrinkage) and $w = 1$ gives $\tilde{\mu} = 0$ (full shrinkage). Our shrinkage estimator has the bias

$$Bias(\tilde{\mu}) = E[(1 - w)\hat{\mu}] - \mu = (1 - w) \underbrace{E[\hat{\mu}]}_{=\mu} - \mu = -w\mu.$$

The variance is

$$Var[\tilde{\mu}] = Var[(1-w)\hat{\mu}] = (1-w)^2 Var[\hat{\mu}] = (1-w)^2 \frac{\sigma^2}{n},$$

and the MSE is

$$MSE(\tilde{\mu}) = Var[\tilde{\mu}] + Bias(\tilde{\mu})^2 = (1-w)^2 \frac{\sigma^2}{n} + w^2\mu^2.$$

The optimal weight in terms of the MSE is

$$w^* = \frac{1}{1 + n\mu^2/\sigma^2}$$

Proof. We take the derivative of $mse(\tilde{\mu})$ across w to obtain the first order condition:

$$-2(1-w)\sigma^2/n + 2w\mu^2 = 0.$$

Solving for w gives $w(1 + n\mu^2/\sigma^2) = 1$. Then, w^* is the global minimum because the second derivative is $2\sigma^2/n + 2\mu^2 > 0$. □ □

For instance, if $\mu = 1$, $\sigma^2 = 1$, and $n = 99$, we have $w^* = 0.01$.

The shrunk sample mean

$$\tilde{\mu}^* = (1-w^*)\hat{\mu} = \frac{n\mu^2/\sigma^2}{1 + n\mu^2/\sigma^2} \frac{1}{n} \sum_{i=1}^{n} X_i$$

has a lower MSE than the usual sample mean:

$$MSE(\tilde{\mu}^*) = (1-w^*)^2 \frac{\sigma^2}{n} + (w^*)^2\mu^2 < \frac{\sigma^2}{n} = mse(\hat{\mu})$$

This is a remarkable result because it tells us that the sample mean is not the best we can do in the MSE sense to estimate a population mean. The shrinked estimator is more efficient. It is biased, but the bias vanishes asymptotically since $\lim_{n\to\infty} w^* = 0$.

The optimal shrinkage parameter w^* is infeasible because μ^2/σ^2 is unknown. While insightful theoretically, this result is not directly applicable in empirical work, and taking sample means is still recommended.

However, the shrinkage principle can be very useful in the context of high-dimensional regression.

10.3 High-dimensional regression

Least squares regression works well when the number of regressors k is small relative to the number of observations n. In a previous section on "too many regressors", we discussed how ordinary least squares (OLS) can overfit when k is too large compared to n. Specifically, if $k = n$, the OLS regression line perfectly fits the data.

Many economic applications involve categorical variables that are transformed into a large number of dummy variables. If we include pairwise interaction terms among J variables, we get another $\sum_{i=1}^{J-1} i = J(J-1)/2$ regressors (for example, 190 for J=20 and 4950 for J=100).

Accounting for further nonlinearities by adding squared and cubic terms or higher-order interactions can result in thousands or even millions of regressors. Many of these regressors may provide low informational value, but it is difficult to determine a priori which are relevant and which are irrelevant.

If $k > n$, the OLS estimator is not uniquely defined because $\boldsymbol{X'X}$ does not have full rank. If $k \approx n$ the matrix $\boldsymbol{X'X}$ can be near singular, resulting in numerically unstable OLS coefficients or high variance.

For the vector-valued (k-variate) estimator $\hat{\boldsymbol{\beta}}_{ols}$ the (conditional) MSE is

$$\begin{aligned} MSE(\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X}) &= E[(\hat{\boldsymbol{\beta}}_{ols} - \boldsymbol{\beta})'(\hat{\boldsymbol{\beta}}_{ols} - \boldsymbol{\beta})|\boldsymbol{X}] \\ &= Var[\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X}] + Bias(\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X})(Bias(\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X}))', \end{aligned}$$

where, under random sampling, OLS is unbiased:

$$Bias(\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X}) = E[\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X}] - \boldsymbol{\beta} = \boldsymbol{0}.$$

Consequently, the MSE of OLS equals its variance:

$$MSE(\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X}) = Var[\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X}] = (\boldsymbol{X'X})^{-1}\boldsymbol{X'DX}(\boldsymbol{X'X})^{-1}.$$

10.4 Ridge Regression

To avoid that $(\boldsymbol{X'X})^{-1}$ becomes very large or undefined for large k, we can introduce a shrinkage parameter λ and define the **ridge regression estimator**

$$\hat{\boldsymbol{\beta}}_{ridge} = (\boldsymbol{X'X} + \lambda \boldsymbol{I}_k)^{-1}\boldsymbol{X'Y}. \tag{10.1}$$

This estimator is well defined and does not suffer from multicollinearity problems, even if $k > n$. The inverse $(\boldsymbol{X'X} + \lambda \boldsymbol{I}_k)^{-1}$ exists as long as $\lambda > 0$. For $\lambda = 0$, the ridge estimator coincides with the OLS estimator.

While the OLS estimator is motivated from the minimization problem

$$\min_{\boldsymbol{\beta}}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}),$$

the ridge estimator is the minimizer of

$$\min_{\boldsymbol{\beta}}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})+\lambda\boldsymbol{\beta}'\boldsymbol{\beta}. \tag{10.2}$$

The minimization problem introduces a penalty for large values of $\boldsymbol{\beta}$. The solution is then shrunk towards zero by $\lambda > 0$.

10.5 Standardization

It is common practice to standardize the regressors in ridge regression:

$$\widetilde{X}_{ij}=\frac{X_{ij}-\overline{\boldsymbol{X}}_j}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_{ij}-\overline{\boldsymbol{X}}_j)^2}},\quad \overline{\boldsymbol{X}}_j=\frac{1}{n}\sum_{i=1}^{n}X_{ij}$$

Without standardization, variables with larger scales (i.e., larger variances) will disproportionately influence the penalty term through $\lambda\boldsymbol{\beta}'\boldsymbol{\beta}=\lambda\sum_{j=1}^{k}\beta_j^2$. Variables with smaller variance may be under-penalized, while those with larger variance may be over-penalized.

Standardization ensures that each variable contributes equally to the penalty term, making the penalty independent of the scale of the variables.

Standardizing makes the coefficient estimates more interpretable, as they will all be on the same scale, which helps in understanding the relative importance of each variable.

10.6 Ridge Properties

The bias of the ridge estimator is

$$Bias(\hat{\boldsymbol{\beta}}_{ridge}|\boldsymbol{X})=-\lambda(\boldsymbol{X}'\boldsymbol{X}+\lambda\boldsymbol{I}_k)^{-1}\boldsymbol{\beta},$$

and the covariance matrix is

$$Var[\hat{\boldsymbol{\beta}}_{ridge}|\boldsymbol{X}]=(\boldsymbol{X}'\boldsymbol{X}+\lambda\boldsymbol{I}_k)^{-1}\boldsymbol{X}'\boldsymbol{D}\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X}+\lambda\boldsymbol{I}_k)^{-1}.$$

In the homoskedastic linear regression model, we have

$$MSE(\hat{\boldsymbol{\beta}}_{ridge}|\boldsymbol{X})<MSE(\hat{\boldsymbol{\beta}}_{ols}|\boldsymbol{X})$$

if $0<\lambda<2\sigma^2/\boldsymbol{\beta}'\boldsymbol{\beta}$.

Similarly to the sample mean case, the upper bound $2\sigma^2/\boldsymbol{\beta}'\boldsymbol{\beta}$ does not give practical guidance for selecting λ because $\boldsymbol{\beta}$ and σ^2 are unknown.

10.7 Mean squared prediction error

The optimal value for λ minimizes the MSE, but estimating the MSE of the ridge estimator is not straightforward because it depends on the parameter $\boldsymbol{\beta}$ being estimated. Instead, it is better to focus on the out-of-sample mean squared prediction error (MSPE).

Let $(Y_1, \boldsymbol{X}_1), \ldots, (Y_n, \boldsymbol{X}_n)$ be our data set (in-sample observations) with ridge estimator Equation 10.1, and let $(Y^{oos}, \boldsymbol{X}^{oos})$ be another observation pair (out-of-sample observation) that is independently drawn from the same population as $(Y_1, \boldsymbol{X}_1), \ldots, (Y_n, \boldsymbol{X}_n)$.

The mean squared prediction error (MSPE) is

$$MSPE(\hat{\boldsymbol{\beta}}_{ridge}) = E[(Y^{oos} - (\boldsymbol{X}^{oos})'\hat{\boldsymbol{\beta}}_{ridge})^2].$$

Note that $(Y^{oos}, \boldsymbol{X}^{oos})$ is independent of $\hat{\boldsymbol{\beta}}_{ridge}$ because it has not been used for estimation. $\widehat{Y}(\boldsymbol{X}^{oos}) = (\boldsymbol{X}^{oos})'\hat{\boldsymbol{\beta}}_{ridge}$ is the predicted value of Y^{oos}.

To estimate the MSPE, we can use a **split sample**.

1) We divide our observations randomly into a training sample (in-sample) of size n_{train} and a testing sample (out-of-sample) of size n_{test} with $n = n_{train} + n_{test}$:

$$(Y_1^{ins}, \boldsymbol{X}_1^{ins}), \ldots (Y_{n_{train}}^{ins}, \boldsymbol{X}_{n_{train}}^{ins}), \quad (Y_1^{oos}, \boldsymbol{X}_1^{oos}), \ldots (Y_{n_{test}}^{oos}, \boldsymbol{X}_{n_{test}}^{oos})$$

2) We estimate $\boldsymbol{\beta}$ using the training sample:

$$\hat{\boldsymbol{\beta}}_{ridge}^{ins} = \left(\sum_{i=1}^{n_{train}} \boldsymbol{X}_i^{ins} (\boldsymbol{X}_i^{ins})' + \lambda \boldsymbol{I}_k \right)^{-1} \sum_{i=1}^{n_{train}} \boldsymbol{X}_i^{ins} Y_i^{ins}.$$

3) We evaluate the empirical MSPE using the testing sample,

$$\widehat{MSPE}_{split} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left(Y_i^{oos} - (\boldsymbol{X}_i^{oos})' \hat{\boldsymbol{\beta}}_{ridge}^{ins} \right)^2 \tag{10.3}$$

Steps 2 and 3 are repeated for different values for λ. We select the value for λ that gives the smallest estimated MSPE.

10.8 Cross validation

A problem with the split sample estimator is that it highly depends on the choice of the two subsamples. An alternative is to select m subsamples (folds) and evaluate the MSPE using each fold separately:

m-fold cross validation

1) Divide the sample into $j = 1, \dots, m$ randomly chosen folds/subsamples of approximately equal size:

$$\begin{aligned}
&(Y_1^{(1)}, \boldsymbol{X}_1^{(1)}), \dots, (Y_{n_1}^{(1)}, \boldsymbol{X}_{n_1}^{(1)}) \\
&(Y_1^{(2)}, \boldsymbol{X}_1^{(2)}), \dots, (Y_{n_2}^{(2)}, \boldsymbol{X}_{n_2}^{(2)}) \\
&\quad \vdots \\
&(Y_1^{(m)}, \boldsymbol{X}_1^{(m)}), \dots, (Y_{n_m}^{(m)}, \boldsymbol{X}_{n_m}^{(m)})
\end{aligned}$$

2) Select $j \in \{1, \dots, m\}$ as left-out test sample and use the other subsamples to compute the ridge estimator $\hat{\boldsymbol{\beta}}_{ridge}^{(-j)}$, where the j-th fold is not used.

3) Compute Equation 10.3 using the j-th folds as a test sample, i.e.,

$$\widehat{MSPE}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} \left(Y_i^{(j)} - (\boldsymbol{X}_i^{(j)})' \hat{\boldsymbol{\beta}}_{ridge}^{(-j)} \right)^2$$

4) The m-fold cross validation estimator is the weighted average over the m subsample estimates of the MSPE:

$$\widehat{MSPE}_{mfold} = \sum_{j=1}^{m} \frac{n_j}{n} \widehat{MSPE}_j,$$

where $n = \sum_{j=1}^{m} n_j$ is the total number of observations.

5) Repeat these steps over a grid of tuning parameters for λ, and select the value for λ that minimizes $\widehat{MSPE}_{mfold}$.

Common values for m are $m = 5$ and $m = 10$. The larger m, the less biased the estimation of the MSPE is, but also the more computationally expensive the cross validation becomes.

The largest possible value for m is $m = n$, where each observation represents a fold. This is also known as leave-one-out cross validation (LOOCV). LOOCV might be useful for small datasets but is often infeasible for large dataset because of the large computation time.

10.9 L2 Regularization: Ridge

The ℓ_p-norm of a vector $\boldsymbol{a} = (a_1, \dots, a_k)'$ is defined as

$$\|\boldsymbol{a}\|_p = \left(\sum_{j=1}^{k} |a_j|^p \right)^{1/p}.$$

Important special cases are the ℓ_1-norm and ℓ_2-norm:

$$\|\boldsymbol{a}\|_1 = \sum_{j=1}^{k} |a_j|, \quad \|\boldsymbol{a}\|_2 = \left(\sum_{j=1}^{k} a_j^2 \right)^{1/2} = \sqrt{\boldsymbol{a}'\boldsymbol{a}}.$$

The ℓ_1-norm is the sum of absolute values, and the ℓ_2-norm, also known as the Euclidean norm, represents the length of the vector in the Euclidean space.

Ridge regression is also called **L2 regularization** because it penalizes the sum of squared errors by the square of the ℓ_2-norm of the coefficient vector, $\|\boldsymbol{\beta}\|_2^2 = \boldsymbol{\beta}'\boldsymbol{\beta}$. Ridge is the solution to the minimization problem Equation 10.2, which can be written as

$$\hat{\boldsymbol{\beta}}_{ridge} = \text{argmin}_{\boldsymbol{\beta}} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_2^2.$$

10.10 L1 Regularization: Lasso

An alternative approach is **L1 regularization**, also known as **lasso**. The lasso estimator is defined as

$$\hat{\boldsymbol{\beta}}_{lasso} = \text{argmin}_{\boldsymbol{\beta}} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1,$$

where $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{k} |\beta_j|$.

The **elastic net** estimator is a hybrid method. It combines L1 and L2 regularization using a weight $0 \leq \alpha \leq 1$:

$$\hat{\boldsymbol{\beta}}_{net,\alpha} = \text{argmin}_{\boldsymbol{\beta}} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda(\alpha \|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2).$$

This includes ridge ($\alpha = 0$) and lasso ($\alpha = 1$) as special cases.

Ridge has a closed form solution given by Equation 10.1. Lasso and elastic net with $\alpha > 0$ require numerical solutions by means of quadratic programming. The solution typically involves some zero coefficients.

10.11 Implementation in R

Let's consider the `mtcars` dataset, which is available in base R. Have a look at `?mtcars` to see the data description.

We estimate a ridge regression model to predict the variable `mpg` (miles per gallon) using the other variables. We consider the values $\lambda = 0.5$ and $\lambda = 2.5$.

Ridge, lasso, and elastic net are implemented in the `glmnet` package. The `glmnet()` function requires matrix-valued data as input. The `model.matrix()` command is useful because it produces the regressor matrix $\boldsymbol{X}$ and converts categorical variables into dummy variables.

```
Y = mtcars$mpg
X = model.matrix(mpg ~., data = mtcars)[,-1]
## Number of observations n and regressors k:
dim(X)
```

```
[1] 32 10
```

```
fit.ridge1 = glmnet(x=X, y=Y, alpha=0, lambda = 0.5)
fit.ridge2 = glmnet(x=X, y=Y, alpha=0, lambda = 2.5)
fits = cbind(coef(fit.ridge1), coef(fit.ridge2))
colnames(fits) = c("lambda=0.5", "lambda=2.5")
fits
```

```
11 x 2 sparse Matrix of class "dgCMatrix"
               lambda=0.5   lambda=2.5
(Intercept) 19.420400249 21.179818696
cyl         -0.250698757 -0.368541841
disp        -0.001893223 -0.005184086
hp          -0.013079878 -0.011710951
drat         0.978514241  1.052837310
wt          -1.902328296 -1.264016952
qsec         0.316107066  0.164790158
vs           0.472551434  0.755205256
am           2.113922488  1.655241565
gear         0.631836101  0.546732963
carb        -0.661215998 -0.560023425
```

By default the regressors are standardized. Therefore the coefficients represent the marginal effects as the change in the response variable for a one standard deviation change in the

regressor. For instance, with $\lambda = 0.5$, the coefficient of `wt` (weight) is -1.9, which means that a one standard deviation increase in weight leads to a decrease of 1.9 miles per gallon.

When we exclude the intercept, the average coefficient size (with respect to the ℓ_2 norm) becomes small for larger values of λ:

```
c(
  sqrt(sum(coef(fit.ridge1)[-1]^2)),
  sqrt(sum(coef(fit.ridge2)[-1]^2))
)
```

```
[1] 3.204323 2.606156
```

The lasso estimator ($\alpha = 1$) sets many coefficients equal to zero:

```
fit.lasso1 = glmnet(x=X, y=Y, alpha=1, lambda = 0.5)
fit.lasso2 = glmnet(x=X, y=Y, alpha=1, lambda = 2.5)
fits = cbind(coef(fit.lasso1), coef(fit.lasso2))
colnames(fits) = c("lambda=0.5", "lambda=2.5")
fits
```

```
11 x 2 sparse Matrix of class "dgCMatrix"
             lambda=0.5 lambda=2.5
(Intercept) 35.88689755 30.0625817
cyl         -0.85565434 -0.7090799
disp         .           .
hp          -0.01411517  .
drat         0.07603453  .
wt          -2.67338139 -1.7358069
qsec         .           .
vs           .           .
am           0.48651385  .
gear         .           .
carb        -0.10722338  .
```

The `cv.glmnet()` command estimates the optimal shrinkage parameter using 10-fold cross validation:

```
set.seed(123) ## for reproducibility
cv.glmnet(x=X, y=Y, alpha = 0)$lambda.min
```

```
[1] 2.746789
```

```
cv.glmnet(x=X, y=Y, alpha = 1)$lambda.min
```

```
[1] 0.8007036
```

We can use ridge and lasso to estimate linear models with more variables than observations. The command `^2` includes all pairwise interaction terms, which produces 55 variables in total. The dataset has $n = 32$ observations.

```
X.large = model.matrix(mpg ~. ^2, data = mtcars)[,-1]
dim(X.large) # more regressors than observations
```

```
[1] 32 55
```

```
fit.ridgelarge = glmnet(x=X.large, y=Y, alpha=0, lambda = 0.5)
fit.lassolarge = glmnet(x=X.large, y=Y, alpha=1, lambda = 0.5)
fits = cbind(
  coef(fit.ridgelarge), coef(fit.lassolarge)
)
colnames(fits) = c("ridge", "lasso")
fits
```

```
56 x 2 sparse Matrix of class "dgCMatrix"
                    ridge        lasso
(Intercept)  1.315259e+01 23.655330629
cyl         -4.061218e-02 -0.036308043
disp        -8.137358e-04  .
hp          -5.588290e-03  .
drat         4.386174e-01  .
wt          -5.547986e-01 -1.301739306
qsec         2.308772e-01  .
vs           6.705889e-01  .
am           4.379822e-01  .
gear         8.788479e-01  .
carb        -1.537294e-01  .
cyl:disp     6.830897e-05  .
cyl:hp       1.351742e-04  .
cyl:drat     2.455464e-02  .
cyl:wt      -2.621868e-03  .
```

```
cyl:qsec     3.358094e-03  .
cyl:vs       1.591177e-01  .
cyl:am       6.102385e-02  .
cyl:gear     3.481957e-02  .
cyl:carb     7.499023e-04  .
disp:hp      8.592521e-06  .
disp:drat   -9.421536e-05  .
disp:wt      2.191122e-04  .
disp:qsec   -1.789464e-05  .
disp:vs     -1.280463e-03  .
disp:am     -9.043597e-03  .
disp:gear   -3.601317e-04  .
disp:carb   -1.255358e-04  .
hp:drat     -2.086003e-03  .
hp:wt        4.404097e-04  .
hp:qsec     -4.347470e-04 -0.001328046
hp:vs       -1.858343e-02  .
hp:am       -2.604620e-03  .
hp:gear     -3.464491e-04  .
hp:carb      9.107116e-04  .
drat:wt     -1.766081e-01 -0.337667877
drat:qsec    3.828881e-02  0.073725291
drat:vs      1.123963e-01  .
drat:am      5.047132e-02  .
drat:gear    8.294201e-02  .
drat:carb   -4.770358e-02  .
wt:qsec     -3.289204e-02  .
wt:vs       -3.239643e-01  .
wt:am       -4.197733e-01  .
wt:gear     -1.890703e-01  .
wt:carb     -1.497574e-02  .
qsec:vs      3.114409e-02  .
qsec:am      5.199239e-02  .
qsec:gear    7.035311e-02  0.041623415
qsec:carb   -1.859676e-02  .
vs:am        8.688134e-01  2.429571498
vs:gear      3.311330e-01  .
vs:carb     -2.768199e-01  .
am:gear      1.462749e-01  .
am:carb      1.588431e-01  .
gear:carb    8.165764e-03  .
```

To get the fitted values you may use the `predict()` command:

```
Yhatridge = predict(fit.ridgelarge, newx = X.large)
Yhatlasso = predict(fit.lassolarge, newx = X.large)
Yhats = cbind(Y, Yhatridge, Yhatlasso)
colnames(Yhats) = c("Y", "Yhat-ridge", "Yhat-lasso")
Yhats
```

```
                     Y Yhat-ridge Yhat-lasso
Mazda RX4         21.0   20.94312   21.64528
Mazda RX4 Wag     21.0   20.47797   21.14997
Datsun 710        22.8   26.12112   25.98585
Hornet 4 Drive    21.4   19.57785   19.91064
Hornet Sportabout 18.7   17.25059   17.35026
Valiant           18.1   19.25815   19.52858
Duster 360        14.3   14.80168   15.42082
Merc 240D         24.4   23.06386   22.50685
Merc 230          22.8   23.69586   22.78181
Merc 280          19.2   18.47341   19.75241
Merc 280C         17.8   18.75521   19.92770
Merc 450SE        16.4   15.39830   15.79922
Merc 450SL        17.3   16.19856   16.61670
Merc 450SLC       15.2   16.21931   16.54465
Cadillac Fleetwood 10.4  12.25717   12.57063
Lincoln Continental 10.4 11.74625   11.88810
Chrysler Imperial 14.7   11.64161   11.58002
Fiat 128          32.4   28.79845   27.43656
Honda Civic       30.4   31.07410   29.68475
Toyota Corolla    33.9   30.63399   28.72288
Toyota Corona     21.5   22.35048   22.60097
Dodge Challenger  15.5   17.17402   17.68091
AMC Javelin       15.2   17.70056   17.97138
Camaro Z28        13.3   14.14050   14.67766
Pontiac Firebird  19.2   16.37763   16.39890
Fiat X1-9         27.3   29.32240   27.93021
Porsche 914-2     26.0   26.15812   24.43481
Lotus Europa      30.4   28.93150   27.72235
Ford Pantera L    15.8   16.69717   17.16642
Ferrari Dino      19.7   20.27929   20.20595
Maserati Bora     15.0   14.07394   14.80373
Volvo 142E        21.4   23.30782   24.50302
```

10.12 R-codes

metrics-sec10.R