Lab 6.2 - Ridge Regression and the Lasso

An Introduction to Statistical Learning

```
TO-DO
-----
[ ] Search for info about `lambda.1se` in `cv.glmnet()`.

--------------------------------------------------------------------------------
```

We will use the `Hitters` dataset.

```
library(ISLR)
sum(is.na(Hitters))
```

```
## [1] 59
```

There are 59 missing observations for `Salary` so we need to make some cleaning:

```
Hitters = na.omit(Hitters)
attach(Hitters)
```

Let's create a matrix with the observations and a targets vector:

```
x = model.matrix(Salary~., data = Hitters)[, -1]
y = Salary
```

Ridge Regression and the Lasso are performed using the `glmnet()` function in the `glmnet` library.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-2
```

1. Ridge Regression

The regression method is selected with the parameter `alpha`; if `alpha=0` then `glmnet` performs Ridge Regression, and the Lasso if `alpha=1`.

Fitting the model

We split the data in training and test subsets:

```
set.seed(1)
train = sample(1:nrow(x), nrow(x)/2)
test = (-train)
x.train = x[train,]
x.test = x[test,]
y.train = y[train]
y.test = y[test]
```

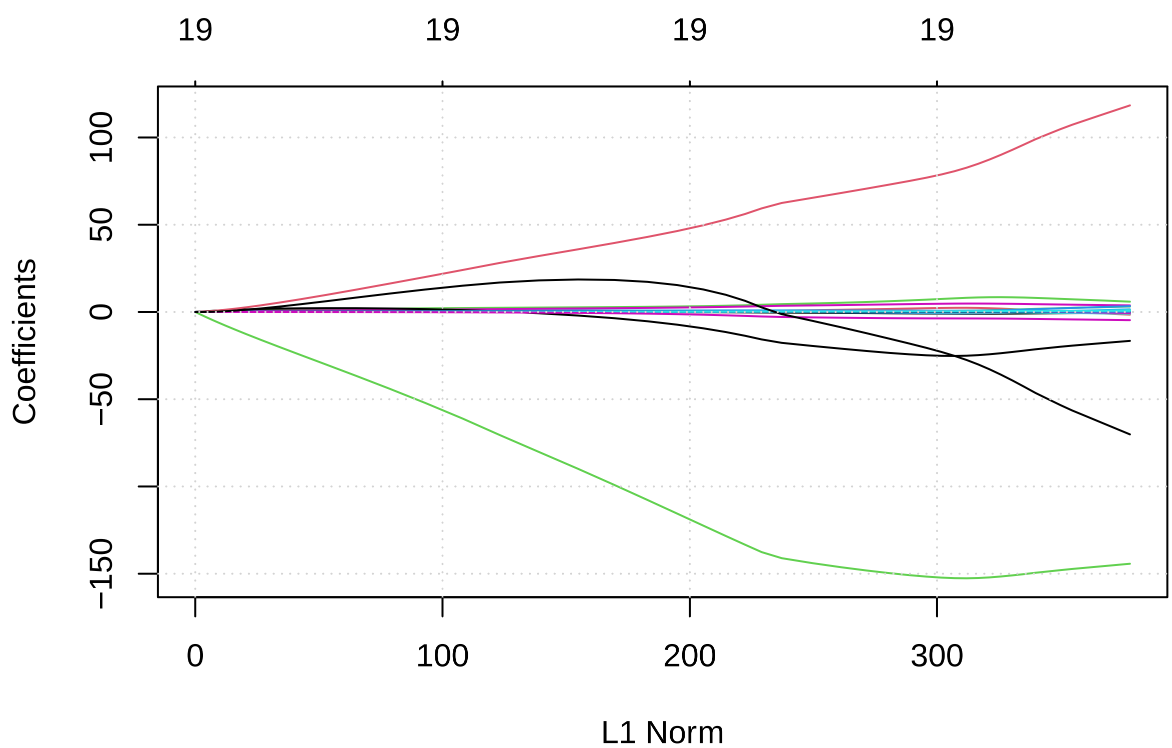
A grid of **decreasing** values for λ can be passed to the function to be used for fitting the model and selecting the best value for λ. Another way to supply values for λ is to use the parameters `nlambda` and lambda.min.ratio.

Here we create a grid of 100 values for λ, from 10^{10} to 0.01:

```
lambdas = 10^seq(10, -2, length = 100)
```

Now we fit the model, specifying a value for the convergence threshold, `thres`:

```
ridge.mod = glmnet::glmnet(x.train, y.train,
                           alpha = 0, lambda = lambdas, thresh = 1e-12)
plot(ridge.mod); grid()
```



By default, `glmnet` standardizes the variables; to avoid this, use `standardize=FALSE`.

For each λ the model has an associated vector of variable coefficients, conforming a $(num\_vars \times num\_lambdas)$ matrix.

```
dim(coef(ridge.mod))
```

```
## [1]  20 100
```

To get the list of λ we use `ridge.mod$lambda`:

```
ridge.mod$lambda[1:10]
```

```
##  [1] 10000000000  7564633276  5722367659  4328761281  3274549163  2477076356
##  [7]  1873817423  1417474163  1072267222   811130831
```

To access the coefficients for a given λ position:

```
coef(ridge.mod)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 413.274463418   0.028461196   0.096884840   0.567798137   0.195484514
##           RBI         Walks         Years        CAtBat         CHits
##   0.205056907   0.289854034   1.038896071   0.003935810   0.014914437
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.120260428   0.029692335   0.031357942   0.038028323   1.295515032
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
##  -7.832335492   0.011781119   0.002692306  -0.006207090   0.446577378
```

We can compute the L2 of the coefficients associated to $\lambda[50] = 11497.57$

```
sqrt(sum(coef(ridge.mod)[-1, 50]^2))
```

```
## [1] 8.05094
```

Making predictions

To make predictions with a given value of $\lambda = 50$:

```
ridge.pred = predict(ridge.mod, newx = x.test, s = 50)
ridge.pred[1:20,]
```

```
##      -Alvin Davis     -Andre Dawson -Andres Galarraga  -Alfredo Griffin
##         672.15801        1132.70177         477.26464         455.38492
##         -Al Newman  -Argenis Salazar    -Andres Thomas   -Andre Thornton
##         268.56598          76.15011         163.14777        1143.54464
##     -Alan Trammell      -Alex Trevino   -Andy VanSlyke       -Buddy Bell
##         982.25196         254.05524         578.86462        1290.44374
## -Buddy Biancalana       -Bruce Bochy      -Barry Bonds    -Bobby Bonilla
##          40.68114         143.94300         544.26703         298.59814
##      -Bill Buckner    -Billy Hatcher     -Bill Madlock -BillyJo Robidoux
##        1286.06022         190.92501         857.58669         281.11927
```

Let's compute the test error:

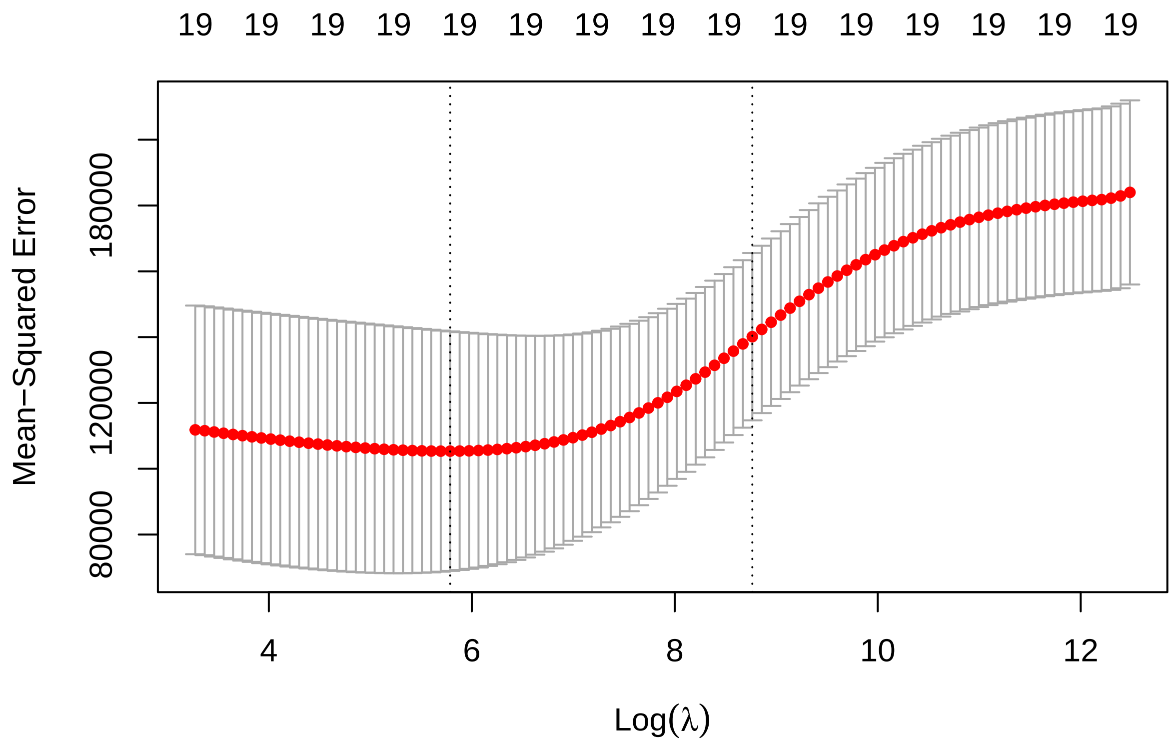
```
mean((ridge.pred - y.test)^2)
```

```
## [1] 144260.1
```

Using Cross-Validation to select λ

The package `glmnet` includes `cv.glmnet()`, a version of the `glmnet()` function that can perform cross-validation. The resulting output of this function can be plotted:

```
set.seed(1)
cv.out.ridge = cv.glmnet(x.train, y.train, alpha = 0)
plot(cv.out.ridge)
```



The vertical lines in the plot mark the values for `lambda.min` and `lambda.1se`, in this case in logarithmic scale.

By default, a 10-fold CV is performed

To select the best λ:

```
bestlam.ridge = cv.out.ridge$lambda.min
cat(sprintf("Best lambda: %.2f [log = %.2f]", bestlam.ridge, log(bestlam.ridge)))
```

```
## Best lambda: 326.08 [log = 5.79]
```

Now we can make predictions using the previously fitted model, `ridge.mod`, and the best value for λ that we just obtained:

```
ridge.pred = predict(ridge.mod, newx = x.test, s = bestlam.ridge)
mean((ridge.pred - y.test)^2)
```

```
## [1] 139856.6
```

We obtain a smaller error than before.

Fitting the complete model

We can now refit the model with all the data:

```
out.ridge = glmnet(x, y, alpha = 0, lambda = lambdas) # lambda is optional
pred.ridge = predict(out.ridge, s = bestlam.ridge, type='coefficients')
pred.ridge[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs           RBI
```

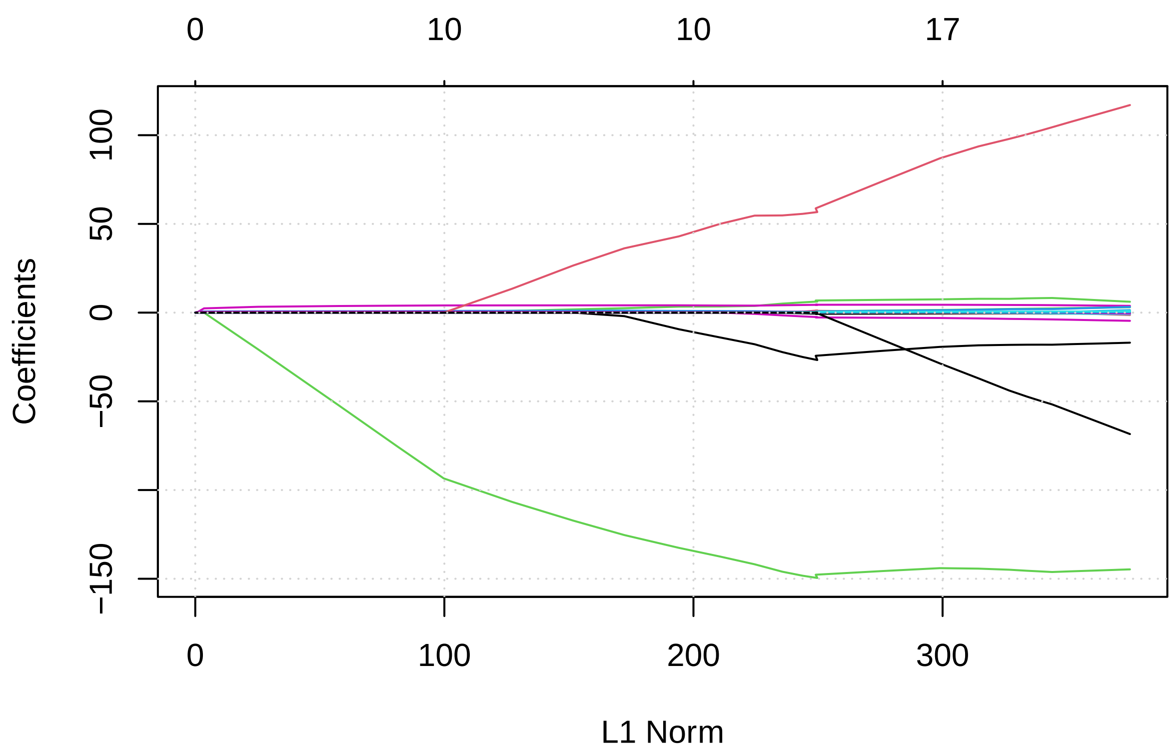
```
## 15.46209975   0.07640574   0.86308801   0.59870362   1.06416544   0.87873337
##        Walks        Years        CAtBat        CHits       CHmRun        CRuns
##   1.62579483   1.35341840   0.01131653   0.05732472   0.40542580   0.11455464
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##   0.12166650   0.05295541  22.17770610 -79.18681200   0.16648537   0.02959948
##       Errors   NewLeagueN
##  -1.37068562   9.06869822
```

2. The Lasso

Lasso regression is also performed with `glmnet()` and `alpha=1`:

```
lasso.mod = glmnet(x.train, y.train, alpha=1, lambda=lambdas)
plot(lasso.mod); grid()
```

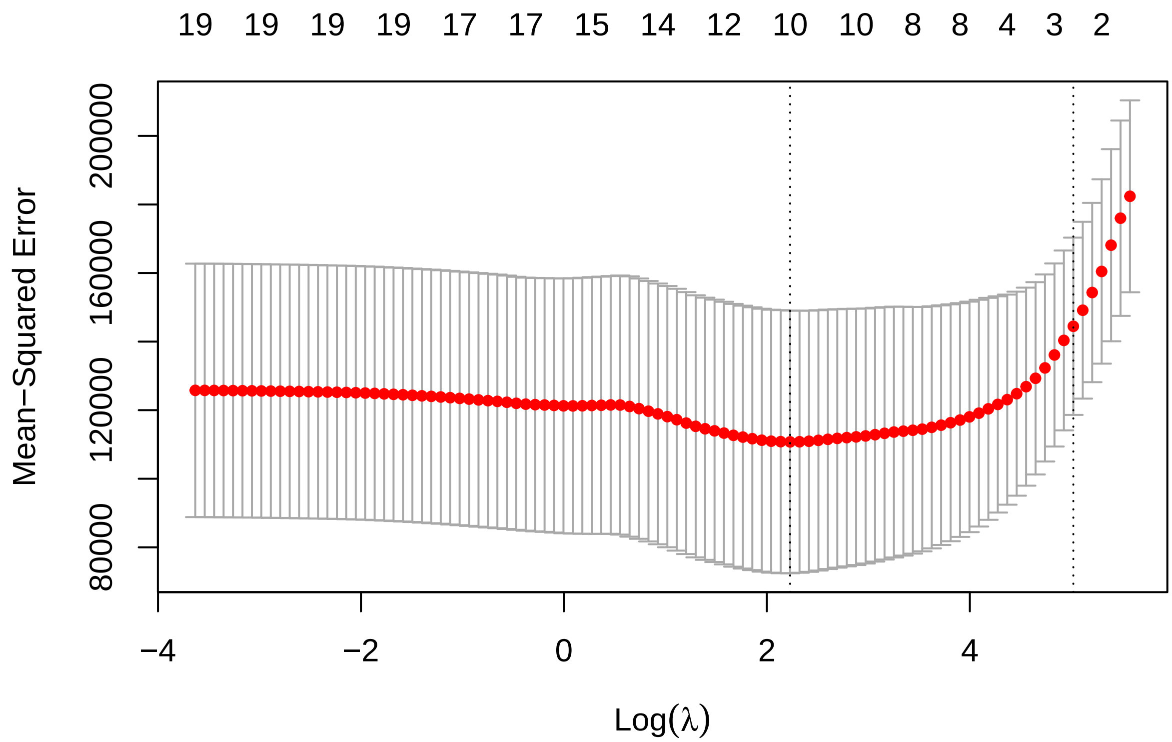
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



Fitting the model

Let's fit a Lasso model using Cross-Validation:

```
set.seed(1)
cv.out.lasso = cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.out.lasso)
```



In this case we get a different value for the best λ:

```
bestlam.lasso = cv.out.lasso$lambda.min
cat(sprintf("Best lambda: %.2f [log = %.2f]", bestlam.lasso, log(bestlam.lasso)))
```

```
## Best lambda: 9.29 [log = 2.23]
```

Making predictions

We use the obtained best λ to make predictions:

```
lasso.pred = predict(cv.out.lasso, newx = x.test, s = bestlam.lasso)
mean((lasso.pred - y.test)^2)
```

```
## [1] 143668.8
```

Fiting the complete model

Now we can use the best λ to fit a model with all the data:

```
out.lasso = glmnet(x, y, alpha = 1, lambda = lambdas)
pred.lasso = predict(out.lasso, type = 'coefficients', s = bestlam.lasso)
```

One advantage of the Lasso over Ridge Regression is that its coefficients are sparse:

```
pred.lasso[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##    1.27479059   -0.05497143    2.18034583    0.00000000    0.00000000
##           RBI         Walks         Years        CAtBat         CHits
```

```
##     0.00000000     2.29192406    -0.33806109     0.00000000     0.00000000 
##         CHmRun          CRuns           CRBI         CWalks        LeagueN 
##     0.02825013     0.21628385     0.41712537     0.00000000    20.28615023 
##      DivisionW        PutOuts        Assists         Errors     NewLeagueN 
## -116.16755870     0.23752385     0.00000000    -0.85629148     0.00000000
```