Regression models to predict reproductive rates, movie sales and energy consumption

Bootstrapping | Stepwise Selection | Shrinkage Methods

Ben Murarotto

April 2025

Contents

1 Aphid Dataset

The aphid dataset contains records of the number of eggs produced on three different days by each of the 20 aphids. The reproductive rate of a species can be defined as:

$$R_0 = \sum_{x=0}^{\infty}(l_x m_x)$$

where lx is the proportion of females surviving to each age, and mx is the average number of offspring produced at each age.

Here I created a function to calculate R0 for the population:

```
rep.fn <- function(data, index) {
  l = data$l[index]
  m = data$m[index]
  return(sum(l*m))

}
```

1.1 Bootstrapping for resampling

Since our dataset contains a small sample size of 20 observations, let us resample with replacement to better understand range of the true mean rate and variability.

```
aphid.df <- read.csv("Aphid.csv", header = T)

eggs.df <- subset(aphid.df, select = -ID)
m <- c()
l <- c()
for (col in colnames(eggs.df)){
  m <-  c(m, (sum(eggs.df[[col]])/sum(eggs.df[[col]] != 0)))
  l <- c(l, (sum(eggs.df[[col]] != 0)/nrow(eggs.df)))

}
boot.df <- data.frame(m=m, l=l)
```

```
library(boot)

set.seed(1)  # for reproducibility
boot.result <- boot(data = boot.df, statistic = rep.fn, R = 100000)

print(boot.result$t0)
```

```
## [1] 11.7
```

```
print(sd(boot.result$t))
```

```
## [1] 0.4234028
```

```
print(boot.ci(boot.result, type = "perc"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 100000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.result, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   (10.8, 12.6 )
## Calculations and Intervals on Original Scale
```

Based on our resampling, we are 95% certain that the true reproductive rate lies between 10.8 and 12.6 offspring per day.

2 Hollywood Dataset

The dataset Hollywood contains information about 10 movie releases. There are 4 variables: Receipts = first year box office receipts/millions Production = total production costs/millions Promo = total promotional costs/millions Books = total book sales/millions

2.1 Fitting linear regression.

We are attempting to fit a linear regression model to predict number of receipts based on the 3 sales variables.

First we will fit and test the full model.

```
hwood.df <- read.csv("Hollywood.csv", header=T)

hwood.lm <- lm(Receipts ~ Production + Promo + Books ,data=hwood.df)
summary(hwood.lm)
```

```
##
## Call:
## lm(formula = Receipts ~ Production + Promo + Books, data = hwood.df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.4384  -3.1695   0.8499   3.5134   9.6207
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.6760     6.7602   1.135   0.2995
## Production    3.6616     1.1178   3.276   0.0169 *
## Promo         7.6211     1.6573   4.598   0.0037 **
## Books         0.8285     0.5394   1.536   0.1754
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.541 on 6 degrees of freedom
## Multiple R-squared:  0.9668, Adjusted R-squared:  0.9502
## F-statistic: 58.22 on 3 and 6 DF,  p-value: 7.913e-05
```

```
newdata <- data.frame(
  Production = 7,
  Promo = 7,
  Books = 10
)

prediction1 <- predict(hwood.lm, newdata = newdata, interval = "confidence")

prediction1
```

```
##       fit      lwr      upr
## 1 94.9393 81.08573 108.7929
```

Here we have a confidence interval between 81.06 and 108.79 however we should cast some skepticism on this range due to the poor health of our dataset which only contains 10 observations.

2.2 Bootstrapping for resampling.

Lets create a bootstrapping function that takes the dataframe, an index and the prediction data to create a better estimate for said prediction.

```r
boot.fn <- function(data, index, newdata){
  model <- lm(Receipts ~ Production + Promo + Books, data = data[index, ])
  prediction <- predict(model, newdata = newdata)
  return(prediction)
}
library(boot)
results <- boot(data = hwood.df, statistic = boot.fn, R = 10000, newdata = newdata)
boot.ci(results, type = "perc")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   ( 76.74, 108.37 )
## Calculations and Intervals on Original Scale
```

```r
se <- sd(results$t)
print(se)
```

```
## [1] 12.7688
```

```r
t0 <- prediction1[1]
moe <- 1.96*se
upper <- t0 + moe
lower <- t0 - moe

print(prediction1)
```

```
##       fit      lwr      upr
## 1 94.9393 81.08573 108.7929
```

```r
cat("The 95% CI for our statistic using bootstrapped standard error is: \n", lower, upper)
```

```
## The 95% CI for our statistic using bootstrapped standard error is:
##  69.91244 119.9662
```

We can see that there is a drastic difference between the dataset prediction range and the bootstrapped range for the prediction. The bootstrapped range is more conservative to account for greater variation in the coefficient estimates and provides a more realistic range for a true result.

3 Energy Dataset

In this dataset, we'll predict appliances energy consumption in the energy dataset. The data include measurements of temperature and humidity sensors from a wireless network, weather from a nearby airport station and recorded energy use of lighting fixtures.

3.1 Preparing for cross-validation.

We are going to do some processing of the dataframe and create a random 50/50 train/test split.

```
energy.df <- read.csv("Energy.csv")
set.seed(1)
n <- floor(0.5 * nrow(energy.df))
index <- sample(seq_len(nrow(energy.df)), size = n)

train <- energy.df[index, ]
test  <- energy.df[-index, ]
train_subset <- subset(train, select = -c(date))
test_subset <- subset(test, select = -c(date))
```

3.2 Fitting a linear model.

Let's fit a linear model using least squares on the training set using all predictors, and assess the test error obtained.

```
mod1 <- lm(data = train_subset, Appliances ~ .)
coefficients(mod1)
```

```
## (Intercept)      lights          T1        RH_1          T2        RH_2
## -77.2068753   2.3290983  -2.9188301  14.5580571 -14.8097525 -12.2761474
##          T3        RH_3          T4        RH_4          T5        RH_5
##  25.6376851   4.2959599  -4.7315365  -0.4838778  -2.3568142   0.1176488
##          T6        RH_6          T7        RH_7          T8        RH_8
##   6.7208436   0.3827258   5.3463522  -1.1047276   8.6420515  -4.2456324
##          T9        RH_9       T_out Press_mm_hg      RH_out   Windspeed
## -16.5741039  -1.5713475 -10.5913358   0.3295977  -1.3348512   1.5018589
##  Visibility   Tdewpoint
##   0.1095687   5.9526860
```

```
summary(mod1)
```

```
##
## Call:
## lm(formula = Appliances ~ ., data = train_subset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -216.24  -43.77  -19.25    5.51  792.73
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -77.20688  134.77338  -0.573  0.56675
## lights        2.32910    0.13630  17.088  < 2e-16 ***
## T1           -2.91883    2.63966  -1.106  0.26886
## RH_1         14.55806    0.96438  15.096  < 2e-16 ***
## T2          -14.80975    2.33697  -6.337 2.44e-10 ***
## RH_2        -12.27615    1.10265 -11.133  < 2e-16 ***
## T3           25.63769    1.50884  16.992  < 2e-16 ***
## RH_3          4.29596    0.94905   4.527 6.07e-06 ***
## T4           -4.73154    1.44739  -3.269  0.00108 **
## RH_4         -0.48388    0.91065  -0.531  0.59518
## T5           -2.35681    1.68285  -1.400  0.16140
```

```
## RH_5          0.11765    0.12352   0.952  0.34089
## T6            6.72084    0.90456   7.430 1.18e-13 ***
## RH_6          0.38273    0.09657   3.963 7.45e-05 ***
## T7            5.34635    1.87174   2.856  0.00429 **
## RH_7         -1.10473    0.61356  -1.801  0.07181 .
## T8            8.64205    1.37333   6.293 3.25e-10 ***
## RH_8         -4.24563    0.52905  -8.025 1.13e-15 ***
## T9          -16.57410    2.49515  -6.643 3.25e-11 ***
## RH_9         -1.57135    0.58597  -2.682  0.00734 **
## T_out       -10.59134    2.14418  -4.940 7.96e-07 ***
## Press_mm_hg   0.32960    0.15283   2.157  0.03106 *
## RH_out       -1.33485    0.44278  -3.015  0.00258 **
## Windspeed     1.50186    0.49206   3.052  0.00228 **
## Visibility    0.10957    0.08268   1.325  0.18512
## Tdewpoint     5.95269    2.07560   2.868  0.00414 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 94.25 on 9841 degrees of freedom
## Multiple R-squared:  0.1664, Adjusted R-squared:  0.1643
## F-statistic: 78.59 on 25 and 9841 DF,  p-value: < 2.2e-16
```

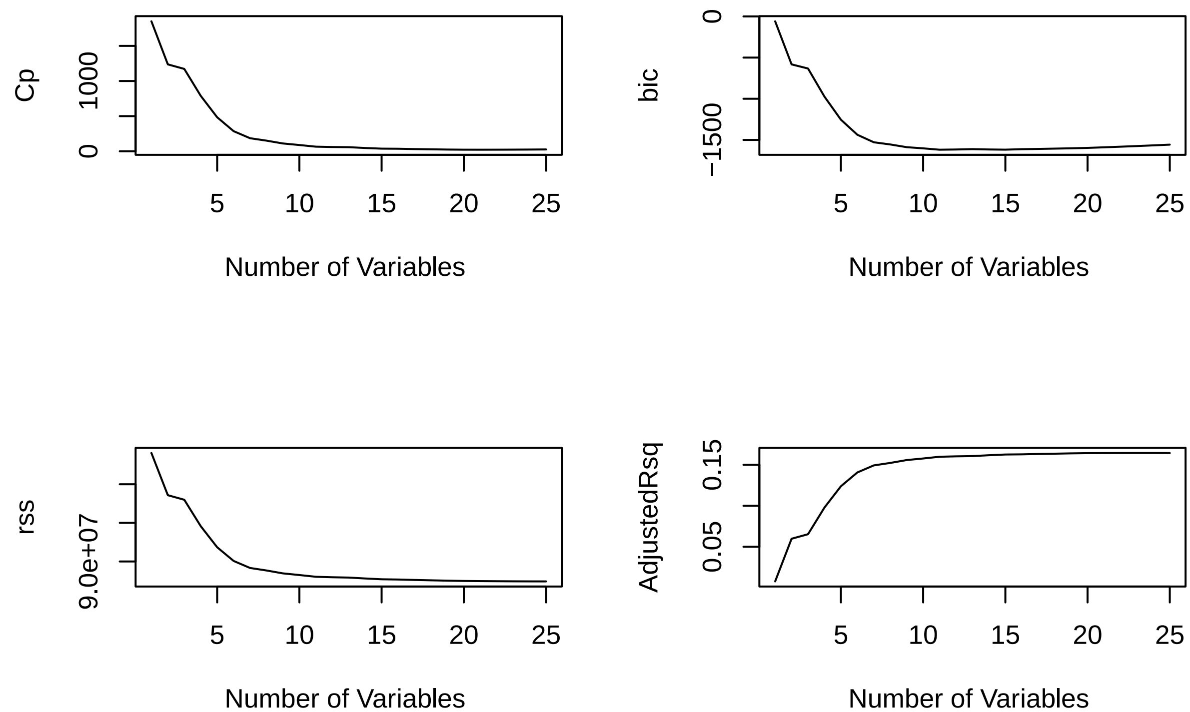
```
full.mod.prediction <- predict(mod1, newdata = test_subset)
y_test <- test_subset$Appliances
mse.full.mod <- mean((full.mod.prediction-y_test)^2)

cat("Test Error for full linear model: \n", mse.full.mod)
```

```
## Test Error for full linear model:
##  8734.338
```

3.3 Stepwise selection of predictors.

Let's use stepwise selection to optimise the model. We are going to use backwards selection and work back from the model we just created.

```
library(leaps)
regfit.bwd <- regsubsets(Appliances ~ .,data=train_subset, nvmax = 25, method = "backward")
reg.summary <- summary(regfit.bwd)

par(mfrow=c(2,2))
plot(reg.summary$cp ,xlab="Number of Variables ",ylab="Cp", type="l")
plot(reg.summary$bic,xlab="Number of Variables ",ylab="bic", type="l")
plot(reg.summary$rss ,xlab="Number of Variables ",ylab="rss", type="l")
plot(reg.summary$adjr2 ,xlab="Number of Variables ",ylab="AdjustedRsq", type="l")
```



Based on our plots of model selection metrics, we see that the steepest change occurs around the 8-10 variable mark. We should then fit a regression model using the ideal 9 predictors and then validate it on the test data.

```
coef(regfit.bwd ,9)
```

```
## (Intercept)      lights        RH_1          T2        RH_2          T3
##   181.610169    2.236306   17.389015  -16.768419  -13.129127   22.713689
##           T6          T8        RH_8          T9
##     1.903290    8.263074   -4.997911  -20.942966
```

```
best_vars <- names(coef(regfit.bwd, 9))[-1]
formula_str <- paste("Appliances ~", paste(best_vars, collapse = " + "))
formula_final <- as.formula(formula_str)
final.lm <- lm(data = train_subset, formula_final)
pred.final <- predict(final.lm, newdata = test_subset)

y_test <- test_subset$Appliances
lm_mse <- mean((pred.final - y_test)^2)
cat("Mean MSE for Final LM: ", lm_mse)
```

```
## Mean MSE for Final LM:  8811.405
```

3.4 Fitting a ridge regression model.

Now we're going to fit a ridge regression model on the training set, with lambda chosen by cross-validation.

```r
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

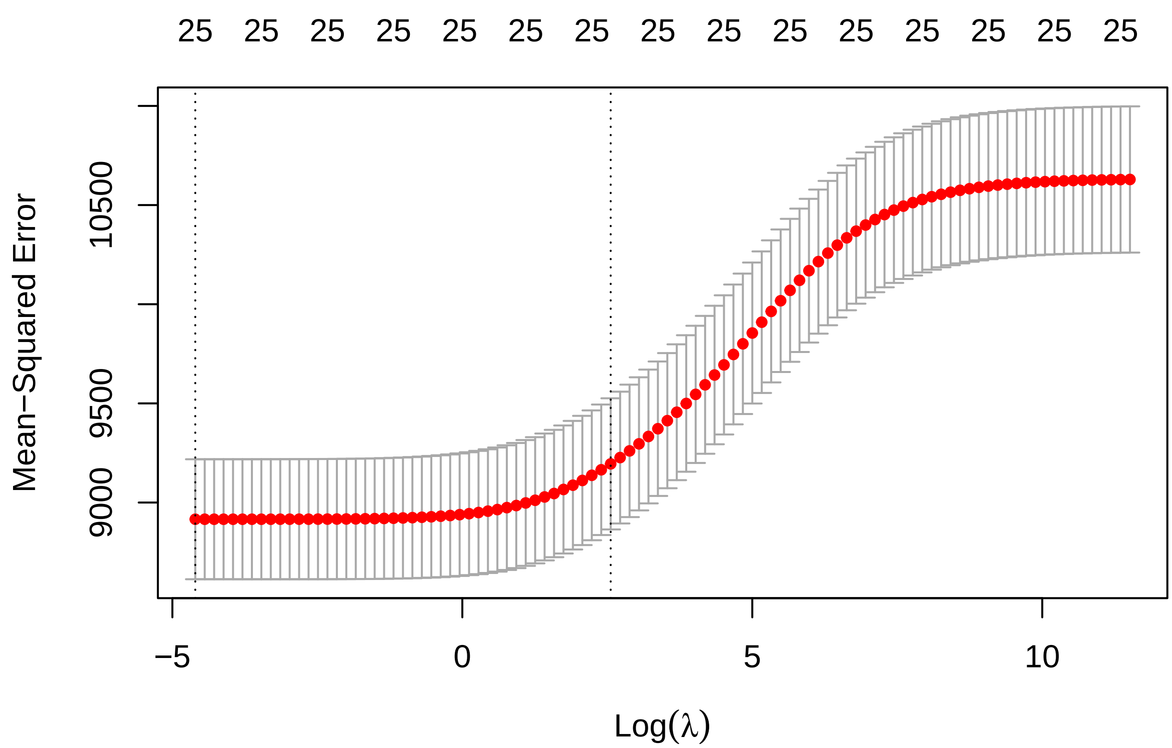
```r
x_train=model.matrix(Appliances~., data=train_subset)[,-1]
y_train=train_subset$Appliances
x_test <- model.matrix(Appliances ~ ., data = test_subset)[,-1]
y_test <- test_subset$Appliances


grid <- 10^seq(5, -2, length = 100)
ridge_cv <- cv.glmnet(x_train, y_train, alpha = 0, lambda = grid)
best_lambda_ridge <- ridge_cv$lambda.min

ridge_pred <- predict(ridge_cv, s = best_lambda_ridge, newx = x_test)
ridge_mse <- mean((ridge_pred - y_test)^2)

cat("Ridge Regression Test MSE:", ridge_mse, "\n")
```

```
## Ridge Regression Test MSE: 8734.613
```

```r
cat("Ridge Regression Best Lambda", best_lambda_ridge, "\n")
```

```
## Ridge Regression Best Lambda 0.01
```

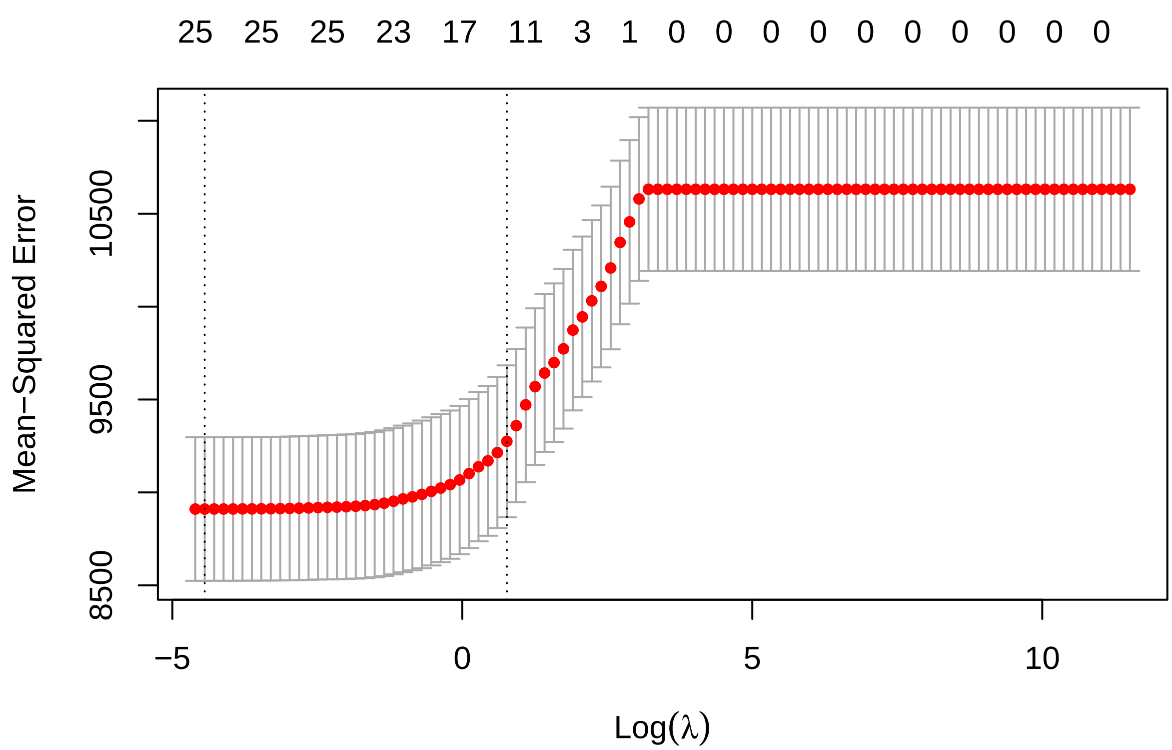
```r
plot(ridge_cv)
```



3.5 Fitting a LASSO model.

Using a similar code structure we can create a LASSO model on the training set, with lambda chosen by cross-validation.

```
lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1, lambda = grid)
best_lambda_lasso <- lasso_cv$lambda.min

lasso_pred <- predict(lasso_cv, s = best_lambda_lasso, newx = x_test)
lasso_mse <- mean((lasso_pred - y_test)^2)

lasso_coef <- predict(lasso_cv, s = best_lambda_lasso, type = "coefficients")
nonzero_coef <- sum(lasso_coef != 0) - 1

plot(lasso_cv)
```




```
cat("Lasso Test MSE:", lasso_mse, "\n")
```

```
## Lasso Test MSE: 8733.679
```

```
cat("Ridge Regression Best Lambda", best_lambda_lasso, "\n")
```

```
## Ridge Regression Best Lambda 0.01176812
```

```
cat("Number of non-zero coefficients:", nonzero_coef, "\n")
```

```
## Number of non-zero coefficients: 25
```

```r
cat("Full model MSE: ", mse.full.mod,"\n")
```

```
## Full model MSE:  8734.338
```

```r
cat("Stepwise model MSE: ", lm_mse,"\n")
```

```
## Stepwise model MSE:  8811.405
```

```r
cat("Ridge model MSE: ", ridge_mse,"\n")
```

```
## Ridge model MSE:  8734.613
```

```r
cat("LASSO model MSE: ", lasso_mse,"\n")
```

```
## LASSO model MSE:  8733.679
```

All four models produced very similar test mean squared errors, all around 8730–8810. The full model and the LASSO model had the lowest test errors (8734.34 and 8733.73, respectively), while stepwise selection performed slightly worse.

This may indicate that the predictors in the dataset are informative and not overly collinear. While LASSO offers the added benefit of variable selection, the overall accuracy of all models is quite similar. In conclusion, any of the models could be considered appropriate, with a slight preference for LASSO.