Selective Inference

Alexander McLain

Data Splitting

Below is an example in R using the bodyfat dataset used previously. We'll use the `glmnet` package for the LASSO regression.

This is a basic illustration and the bodyfat dataset might not be high-dimensional enough to warrant a LASSO, but it serves well to explain the concept:

We first load the `glmnet` package, which provides tools to fit LASSO models.

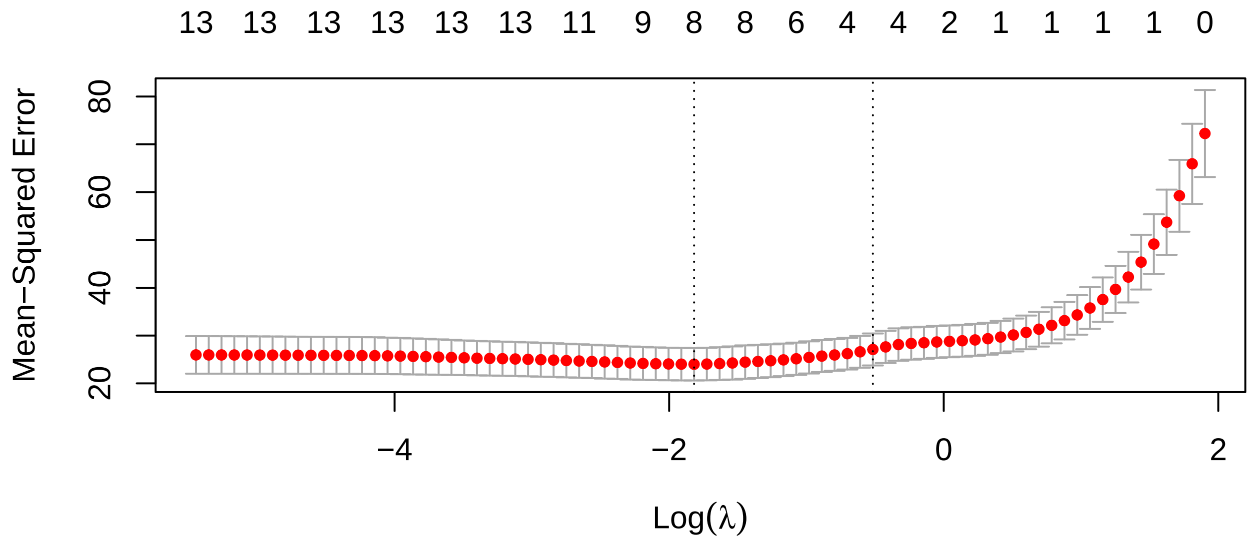
```r
# Load required libraries
library(glmnet)

# Use the bodyfat dataset
bf_dat <- read.csv("bodyfat2.csv")
# Take off body density
bf_df <- data.frame(bf_dat)[, -1]
```

Split it into a training set (selection set) for model selection and a test set (inference set) for inference.

```r
# Setting the seed for reproducibility
set.seed(123)

# Split the data into a training (selection) set and a test (inference) set
train_indices <- sample(1:nrow(bf_df),
                        floor(nrow(bf_df)*0.5))
train_data <- bf_df[train_indices, ]
test_data <- bf_df[-train_indices, ]
```

We fit a LASSO regression on the training set and identify the predictors selected by the LASSO.

```r
# Fit the LASSO model on the training data
x_train <- as.matrix(train_data[, -1])
y_train <- train_data[, 1]
lasso_model <- cv.glmnet(x_train, y_train, alpha = 1, grouped = FALSE)
plot(lasso_model)
```




```r
# Predictors selected by LASSO
coef_lasso <- coef(lasso_model, s = lasso_model$lambda.min)
coef_lasso
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  3.32272889
## age          0.09644498
## weight       .
## height      -0.23229759
## neck        -0.49906487
## chest        .
## abdomen      0.67862648
## hip          .
## thigh        .
## knee         .
## ankle        0.14386938
## biceps       0.13904008
## forearm      0.29377424
## wrist       -1.76170519
```

```r
selected_predictors <- colnames(x_train)[which(coef_lasso != 0)[-1]]
```

Using only the selected predictors, we fit an ordinary least squares (OLS) regression on the test set.

```r
# Fit an OLS regression on the test data using only the selected predictors
x_test <- as.matrix(test_data[, which(coef_lasso != 0)[-1]])
y_test <- test_data[, 1]
ols_model <- lm(y_test ~ x_test)

# Display the summary of the OLS model
summary(ols_model)
```

```
##
## Call:
## lm(formula = y_test ~ x_test)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -9.0851 -2.8405 -0.1438  2.8999  9.1163
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.14733   10.78030   0.106   0.9154
## x_testage     0.01086    0.03911   0.278   0.7818
## x_testheight -0.36204    0.17004  -2.129   0.0353 *
## x_testneck   -0.25638    0.32925  -0.779   0.4377
## x_testabdomen  0.82312    0.06013  13.688   <2e-16 ***
## x_testankle  -0.31704    0.29112  -1.089   0.2784
## x_testbiceps -0.01066    0.25220  -0.042   0.9663
## x_testforearm  0.10297    0.48477   0.212   0.8322
## x_testwrist  -1.03574    0.81787  -1.266   0.2079
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.167 on 117 degrees of freedom
## Multiple R-squared:  0.7637, Adjusted R-squared:  0.7476
## F-statistic: 47.28 on 8 and 117 DF,  p-value: < 2.2e-16
```

Selective Inference Tools

Here we'll use the R package named `selectiveInference` developed by the authors who pioneered this concept. It provides selective inference for LASSO, among other things.

Here's a simple example using the `bf_df` dataset:

```r
# install.packages('selectiveInference')
library(selectiveInference)
```

```
## Loading required package: intervals
```

```
##
## Attaching package: 'intervals'
```

```
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
## Loading required package: survival
```

```
## Loading required package: adaptMCMC
```

```
## Loading required package: parallel
```

```
## Loading required package: coda
```

```
## Loading required package: MASS
```

Now, let's use the `bf_df` dataset:

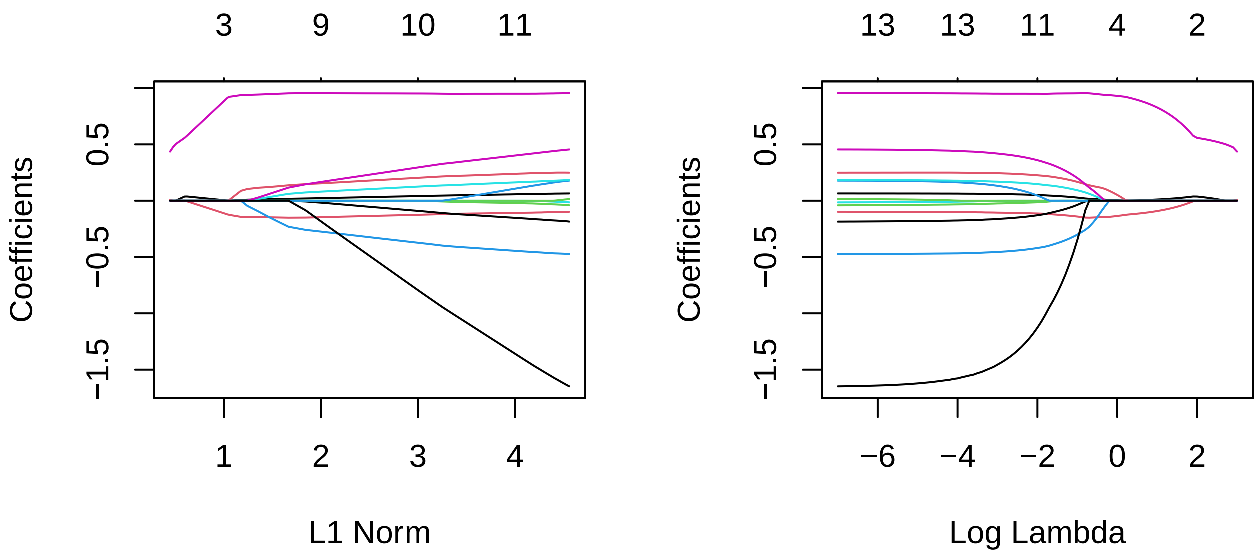
```r
X <- as.matrix(bf_df[, -1])
y <- bf_df[, 1]
```

Using the selective inference package requires that `x` should be centered,

```r
x <- scale(X, TRUE, FALSE)
```

Fit a LASSO model using `cv.glmnet`:

```r
fit <- glmnet(x, y, alpha=1,
              standardize = FALSE,
              intercept = TRUE,
              lambda = exp(seq(-7,3,0.1)) )

par(mfrow=c(1,2))
plot(fit); plot(fit, xvar = "lambda")
```




```r
lambda <- 1
# extract coef for a given lambda minus the intercept term
beta <- coef(fit, x=x, y=y, s = lambda, exact = TRUE)[-1]
```

Apply the `fixedLassoInf` function from the `selectiveInference` package:

```r
`?`(fixedLassoInf)
```

```
Inference for the lasso, with a fixed lambda

Description:

     Compute p-values and confidence intervals for the lasso estimate,
     at a fixed value of the tuning parameter lambda

Usage:

     fixedLassoInf(x,
                   y,
                   beta,
                   lambda,
                   family = c("gaussian", "binomial", "cox"),
                   intercept=TRUE,
                   add.targets=NULL,
                   status=NULL,
                   sigma=NULL,
                   alpha=0.1,
                   type=c("partial","full"),
                   tol.beta=1e-5,
```

```
                  tol.kkt=0.1,
                  gridrange=c(-100,100),
                  bits=NULL,
                  verbose=FALSE,
                  linesearch.try=10)
```

```r
# Fit to the data (note the scaling of lambda by n)
result <- fixedLassoInf(x, y, beta, lambda = lambda * nrow(x))
print(result)
```

```

Call:
fixedLassoInf(x = x, y = y, beta = beta, lambda = lambda * nrow(x))

Standard deviation of noise (specified or estimated) sigma = 4.309

Testing results at lambda = 252.000, with alpha = 0.100

 Var   Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
   1  0.018   0.654   0.838    -0.555    0.041       0.050      0.049
   2 -0.180  -6.799   0.003    -0.273   -0.090       0.050      0.050
   6  0.968  14.608   0.000     0.890    1.724       0.049      0.050
   8  0.252   2.169   0.448    -0.784    0.402       0.050      0.049

Note: coefficients shown are partial regression coefficients
```

The result will provide selective p-values for the coefficients, among other things. Remember, these p-values account for the model selection process and provide a valid inference even when considering the adaptiveness of LASSO.

Conformal Inference

For this example, we'll use the `bf_df` dataset. The `glmnet` package will be used for LASSO regression.

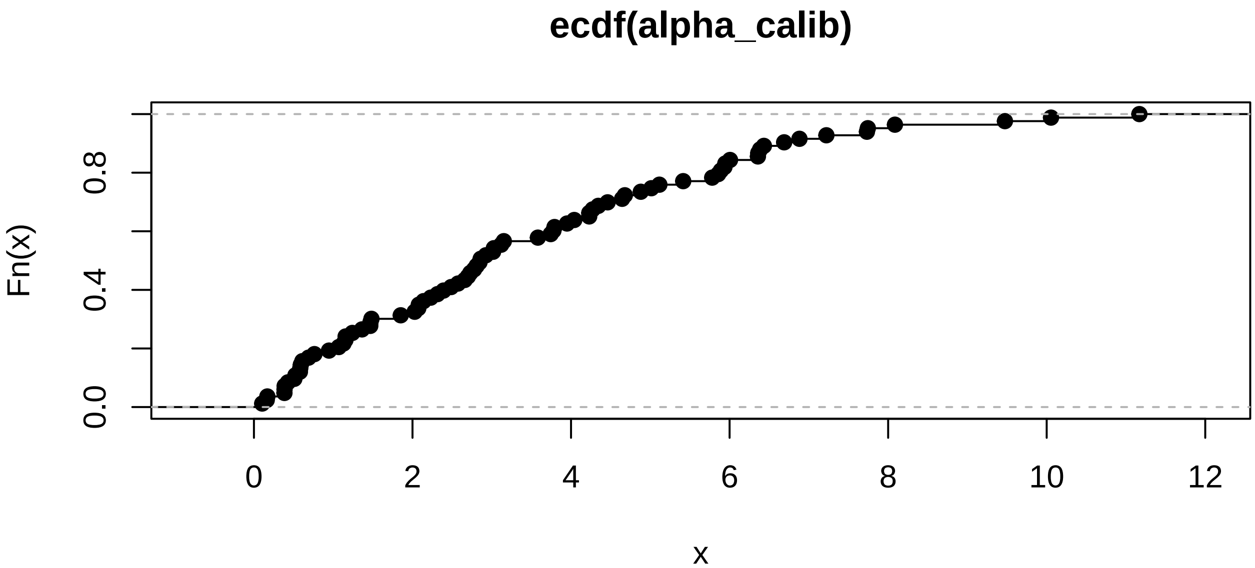
We'll implement the conformal inference manually to get the idea. In general, using the `conformalInference` package is how I would implement this.

```r
set.seed(123)

# Take one observation to predict for
pi_data <- bf_df[1, ]

# Split the rest of the data into training (2/3) and calibration (1/3) sets
train_indices <- sample(2:nrow(bf_df),
                        ceiling((nrow(bf_df)-1)*2/3)
                        )
train_data <- bf_df[train_indices, ]
calib_data <- bf_df[-c(1,train_indices), ]
c(nrow(train_data), nrow(calib_data))
```

```
## [1] 168  83
```

```r
# Fit a LASSO model to the training data
X_train <- as.matrix(train_data[, -1])
y_train <- train_data[,1]
lasso_fit <- glmnet(X_train, y_train, alpha=1, lambda=0.1)
```

```r
# Predict on the calibration set
X_calib <- as.matrix(calib_data[, -1])
y_calib <- calib_data[,1]
predictions_calib <- predict(lasso_fit, s=0.1, newx=X_calib)

# Compute nonconformity scores for the calibration set (absolute residuals)
alpha_calib <- abs(y_calib - predictions_calib)
plot(ecdf(alpha_calib))
```




```r
# Get the (n+1)*(1-alpha) largest nonconformity score
(ind_0.95 <- ceiling((length(alpha_calib)+1)*(0.95)))
```

```
## [1] 80
```

```r
(alpha_0.95 <- sort(alpha_calib)[ind_0.95])
```

```
## [1] 8.085536
```

```r
# New observation
X_pi <- as.matrix(pi_data[-1])
y_pi <- pi_data[1]
(predicted_new <- predict(lasso_fit, s=0.1, newx=as.matrix(X_pi)))
```

s1
15.47377

```r
y_pi
```

bodyfat
12.3

```r
# Use alpha_0.95 to form an 80% prediction interval.
conf_pi_lower <- predicted_new - alpha_0.95
conf_pi_upper <- predicted_new + alpha_0.95
```

```r
paste("Conformal 95% PI is:",
            round(conf_pi_lower,3),
            round(conf_pi_upper,3) )
```

```
## [1] "Conformal 95% PI is: 7.388 23.559"
```

```r
# Conformal prediction: proportion of calibration set points with a larger nonconformity score
alpha_new <- abs(y_pi - predicted_new)
p_val <- mean(c(alpha_calib) > c(alpha_new) )

paste("Empirical p-value for the new observation is:",
            round(p_val,3))
```

```
## [1] "Empirical p-value for the new observation is: 0.434"
```

This empirical p-value tells us how unusual the new observation is compared to the calibration set. If it's small (e.g., less than 0.05), the new observation is deemed atypical compared to what the model expects.