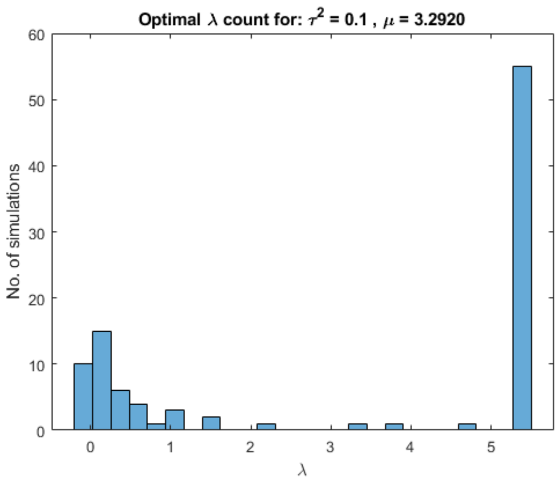
Ridge Regression programming exercise

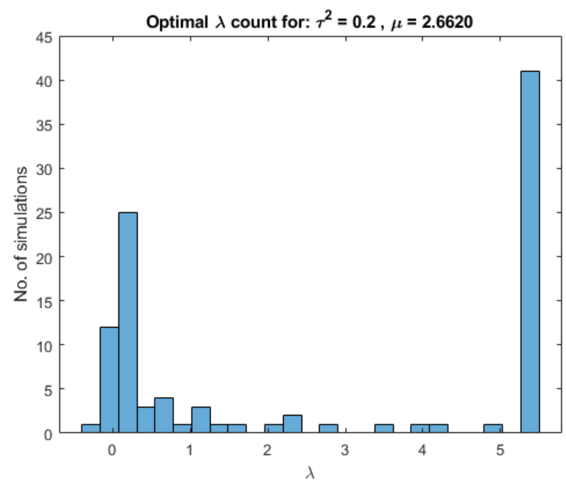
February 19, 2019

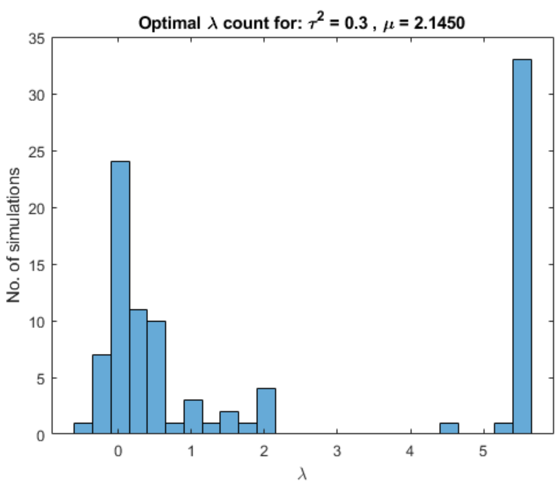
1 Optimal values of λ for every τ

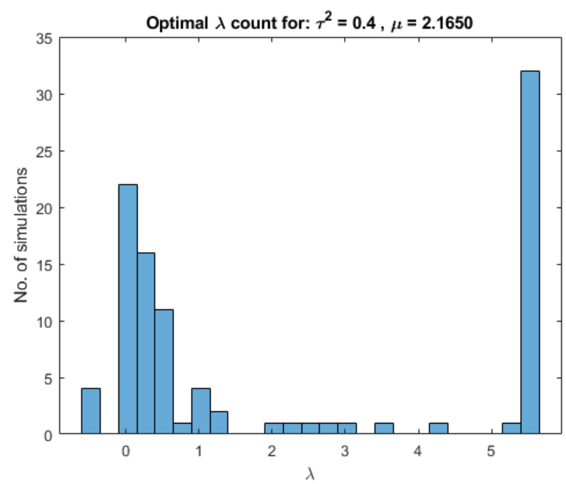
Using the code presented in Listing 1, random datasets of size 10x2 were generated, in accordance with the way X1, X2, and Y were defined in the assignment. In Listing 2, the procedure for ridge regression is shown. For every value of tau, first a 'rough' estimate of the best λ from [0,0.5,...,5] is chosen by optimizing for average squared error and leave-one-out cross-validation. Then, a 'finer' estimate of the best λ was obtained by the same process using a finer grid of [$\lambda-0.5$,$\lambda-0.4$,...,$\lambda+0.5$].

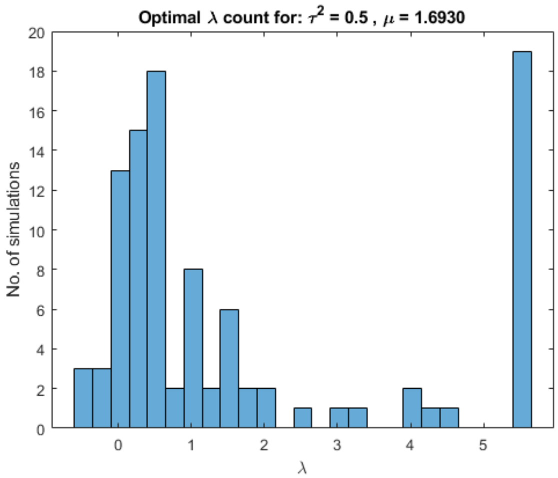
This was repeated 100 times for every τ and the values so obtained were plotted in histograms, shown below. The mean value for the best λ is mentioned in the caption of the plots, each of which correspond to a particular value of λ. We can see that every plot approximates a bimodal normal distribution.

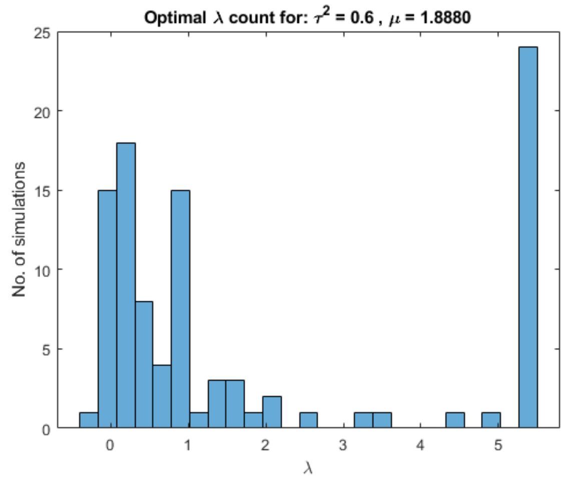


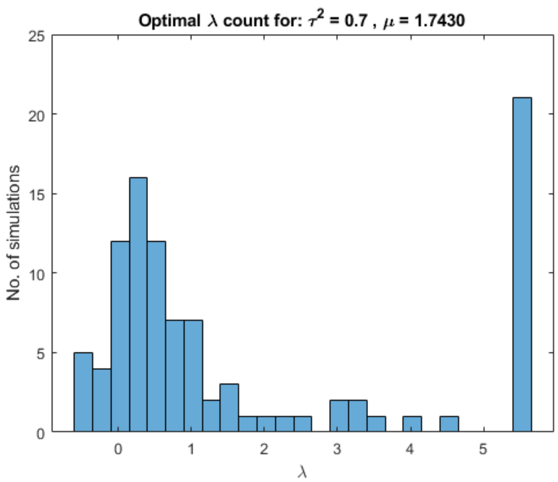


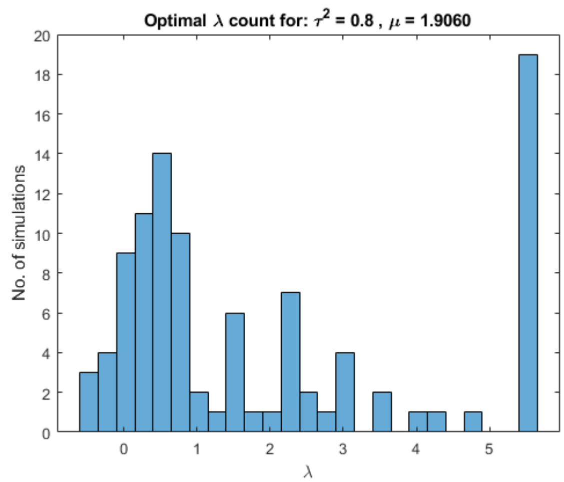


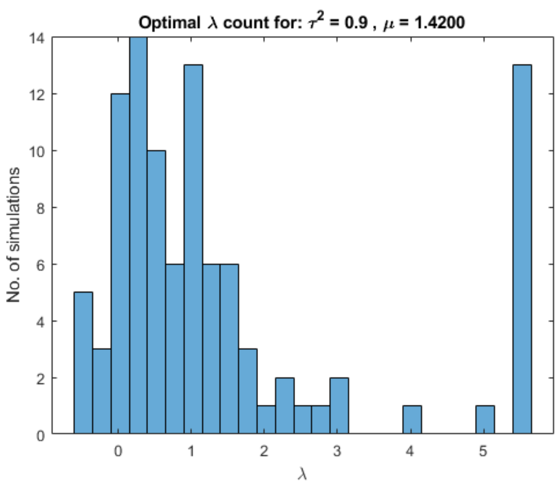












Listing 1: Generating random datasets

```
function dataset = data(N, p, tau_sq)
x1 = randn(N,1);

mean_u = 0;
std_u = sqrt(tau_sq);
u = std_u.*randn(10,1) + mean_u;
```

```
x2 = (sqrt(1-tau_sq)* x1) + u;

epsilon = randn(N,1);
y = x1-x2+epsilon;

dataset = [x1 x2 y];
end
```

Listing 2: Performing Ridge Regression

```
N = 10;
p = 2;
tau_sq_vector = linspace(0.1,0.9,9);
lambda_vector = linspace(0,5,11);
results_tau = [];
results_lambda = [];

for tau_index = 1:9

    % select a tau
    tau_sq = tau_sq_vector(tau_index);

    results_tau = [results_tau tau_sq];

    % generate a dataset
    D = data(N, p, tau_sq);
    X = D(:,1:2);
    Y = D(:,end);

    % initialise counters to keep track of the best mse
    % and lambda values encountered
    minimum_mse = Inf;
    best_lambda = Inf;

    for lambda_index=1:11

        % select a lambda
        lambda = lambda_vector(lambda_index);

        total_error = 0;

        % cross validation to fit and predict
        for num=1:N
            X_fit = X;
            X_predict = X_fit(num,:);
            X_fit(num,:) = [];

            Y_fit = Y;
            Y_predict = Y_fit(num);
            Y_fit(num) = [];

            estimator = inv(X_fit' * X_fit + lambda* eye(2)) * X_fit' * Y_fit;
            Y_hat = X_predict*estimator;
            error = (Y_predict - Y_hat).^2;
            total_error = total_error + error;
        end

```

```
        % average squared error for a given lambda
        mse = total_error / N;

        % check if current mse is better than minimum mse encountered uptil now
        % if yes, store the optimal lambda
        if mse < minimum_mse
            minimum_mse = mse;
            best_lambda = lambda;
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % give a finer grid around the chosen lambda
    best_lambda_vector = linspace(best_lambda-0.5,best_lambda+0.5,11);

    minimum_mse_finer = Inf;
    best_lambda_finer = 10;

    for lambda_index=1:11

        lambda = best_lambda_vector(lambda_index);

        total_error = 0;

        % cross validation to fit and predict
        for num=1:N
            X_fit = X;
            X_predict = X_fit(num,:);
            X_fit(num,:) = [];

            Y_fit = Y;
            Y_predict = Y_fit(num);
            Y_fit(num) = [];

            estimator = inv(X_fit' * X_fit + lambda* eye(2)) * X_fit' * Y_fit;
            Y_hat = X_predict*estimator;
            error = (Y_predict - Y_hat).^2;
            total_error = total_error + error;
        end

        % average squared error for a given lambda
        mse = total_error / N;

        % check if current mse is better than minimum mse encountered uptil now
        % if yes, store the optimal lambda
        if mse < minimum_mse_finer
            minimum_mse_finer = mse;
            best_lambda_finer = lambda;
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    results_lambda = [results_lambda best_lambda_finer];

end
```