Machine Learning for Economists - Assignment 1

2024-04-19

Assignment 1 - Machine Learning for Economists

Authors: Simon Merten, Matriculation Number 7375404
Leon Kvas, Matriculation Number 7371206
Marius Meise, Matriculation Number 7369546

Questions & Answers

1. We will consider out-of-sample predictions at x0 = (2, 2, . . . , 2). What value of f(x0) do you expect?

Given the model:

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \dots + \beta_p x_{pi} + \epsilon_i$$

where $\beta_1 = 2$, $\beta_2 = 3$, and $\beta_j = 0$ for $j > 2$. We consider out-of-sample predictions at $x_0 = (2, 2, \dots, 2)$. The expected value of $f(x_0)$, assuming no error ($\epsilon_i = 0$), can be calculated like this:

$$f(x_0) = \beta_1 \times 2 + \beta_2 \times 2 + \sum_{j=3}^{10} \beta_j \times 2$$

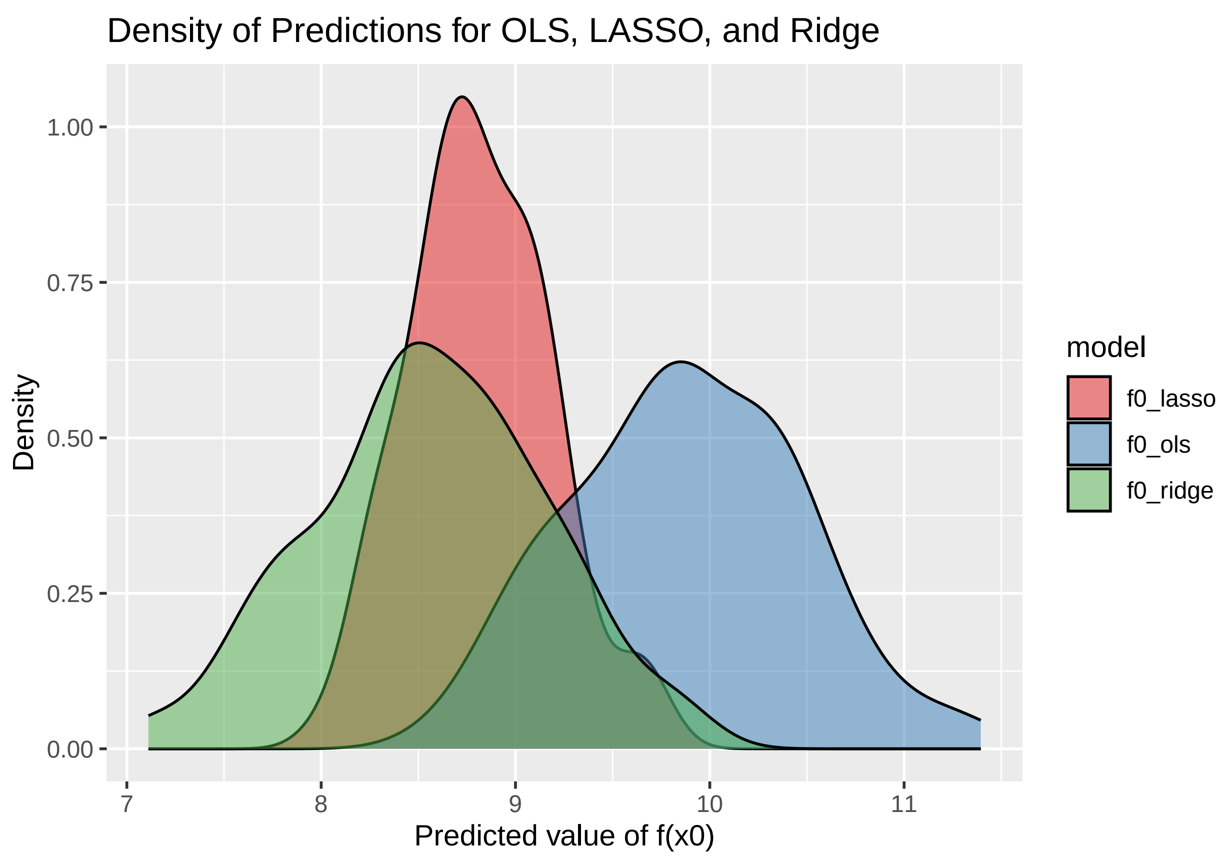
Given the coefficients, this simplifies to:

$$f(x_0) = 2 \times 2 + 3 \times 2 + 0 = 10$$

2. Run the code as is. Plot the distribution of predictions ^f(x0) for the different models. What do you observe?

```r
library(glmnet)
library(tidyverse)

nsim = 100 # Number of simulations
nobs = 100 # Number of observations in the simulated dataset
p = 10 # Number of regressors
beta = c(2,3, rep(0,p-2)) # Coefficient vector
                          # (2 for beta1, 3 for beta2,
                          # 0s for remaining betas)
```

```r
x0 = c(rep(2, p)) # out-of-sample observation

# Initialize vectors so that we can store results
f0_ols = c()
f0_lasso = c()
f0_ridge = c()

for (s in 1:nsim) {

  X = MASS::mvrnorm(nobs, #Randomly drawn X variables
                         mu = rep(0, p),
                         Sigma = diag(p)
                         )

  y = X %*%beta + rnorm(nobs) # Simulate y with normal noise

  # Fit OLS model
  fitOLS = lm(y~X)
  f0_ols[s] = fitOLS$coefficients%*%c(1, x0) # Prediction at x0

  # Fit LASSO regression
  fitLASSO = cv.glmnet(X, y, alpha = 1)
  f0_lasso[s] = t(coef(fitLASSO, s="lambda.1se"))%*%c(1, x0) %>% as.numeric()

  # Fit Ridge regression
  fitRidge = cv.glmnet(X, y, alpha = 0)
  f0_ridge[s] = t(coef(fitRidge, s="lambda.1se"))%*%c(1, x0) %>% as.numeric()
}

# Data frame for plotting
predictions <- data.frame(f0_ols, f0_lasso, f0_ridge)

# Melting data frame for easier plotting
predictions_long <- pivot_longer(predictions, cols = everything(),
                                 names_to = "model",
                                 values_to = "prediction")

# Plotting
ggplot(predictions_long, aes(x = prediction, fill = model)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density of Predictions for OLS, LASSO, and Ridge",
       x = "Predicted value of f(x0)",
       y = "Density") +
  scale_fill_brewer(palette = "Set1")
```



```
mean(f0_ols)
```

```
## [1] 9.880421
```

```
mean(f0_lasso)
```

```
## [1] 8.820232
```

```
mean(f0_ridge)
```

```
## [1] 8.556245
```

By examining the distributions of the predictions, it becomes apparent that the distributions of LASSO and Ridge are more similar to each other in terms of expectation than they are to OLS. The mean of the predictions for LASSO and Ridge is below 9, whereas for OLS, it is around 10. These results suggest the bias introduced in the regularized models. The unbiased OLS regression, in its expectation, approximates the true value of 10.

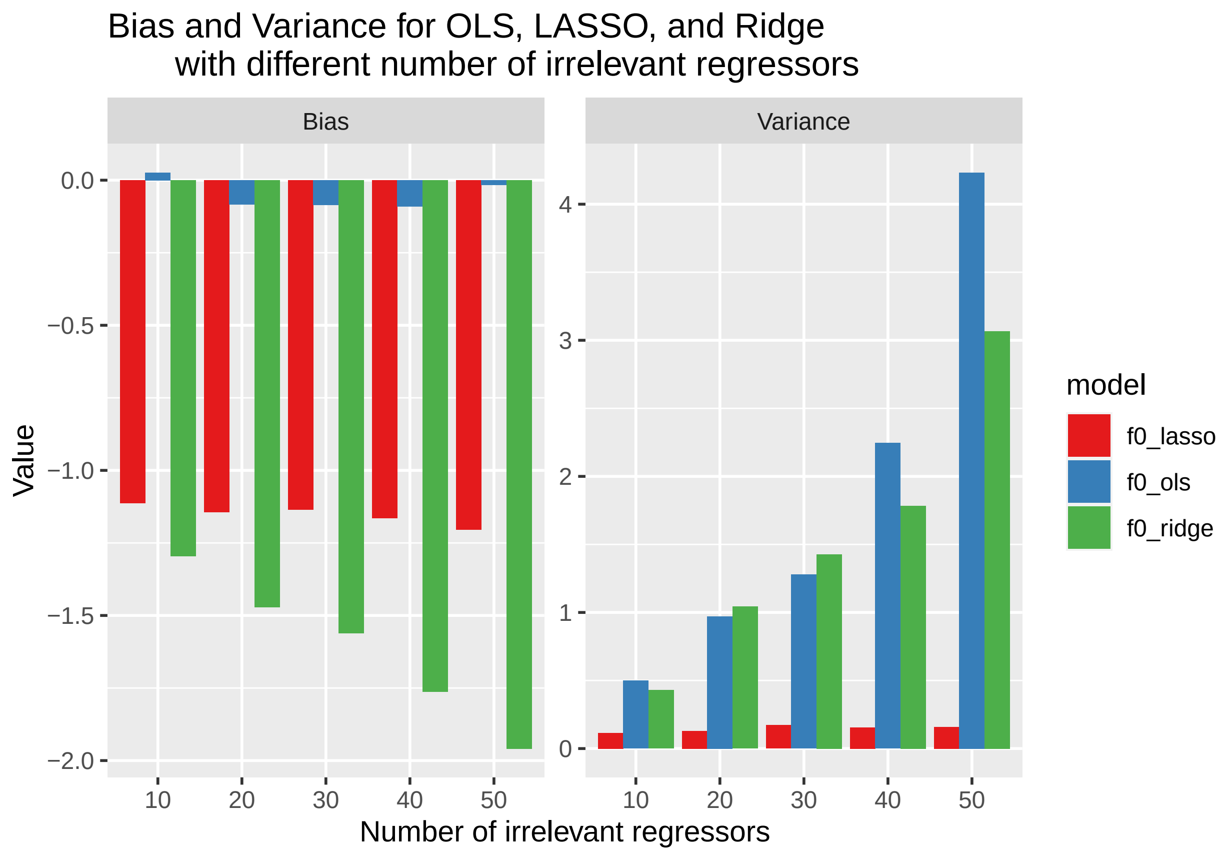
3. Compute bias, variance and mean squared error for the three different models at x0, our out-of-sample observation.

```r
# True value of f(x0)
true_value <- 10

# Calculate Bias, Variance, and MSE for each model
statistics <- predictions_long %>%
  group_by(model) %>%
  summarise(
    Bias = mean(prediction) - true_value,
    Variance = var(prediction),
    MSE = mean((prediction - true_value)^2),
    .groups = 'drop'
  )

print(statistics)
```

```
## # A tibble: 3 x 4
##   model      Bias Variance   MSE
##   <chr>     <dbl>    <dbl> <dbl>
## 1 f0_lasso -1.18     0.131 1.52
## 2 f0_ols   -0.120    0.345 0.356
## 3 f0_ridge -1.44     0.358 2.44
```

4. How do bias and variance depend on the number of irrelevant regressors?

```r
# function to run the simulation with different number of irrelevant regressors
run_simulation1 <- function(p) {
  beta = c(2,3, rep(0,p-2)) # Coefficient vector
  x0 = c(rep(2, p)) # out-of-sample observation
  f0_ols = c()
  f0_lasso = c()
  f0_ridge = c()
  for (s in 1:nsim) {
    X = MASS::mvrnorm(nobs, #Randomly drawn X variables
                      mu = rep(0, p),
                      Sigma = diag(p)
    )
    y = X %*%beta + rnorm(nobs) # Simulate y with normal noise
    # Fit OLS model
    fitOLS = lm(y~X)
    f0_ols[s] = fitOLS$coefficients%*%c(1, x0) # Prediction at x0
    # Fit LASSO regression
    fitLASSO = cv.glmnet(X, y, alpha = 1)
    f0_lasso[s] = t(coef(fitLASSO, s="lambda.1se"))%*%c(1, x0) %>% as.numeric()
    # Fit Ridge regression
    fitRidge = cv.glmnet(X, y, alpha = 0)
    f0_ridge[s] = t(coef(fitRidge, s="lambda.1se"))%*%c(1, x0) %>% as.numeric()
  }

  # Data frame for plotting
  predictions <- data.frame(f0_ols, f0_lasso, f0_ridge)
  # Melting data frame for easier plotting
  predictions_long <- pivot_longer(predictions,
                                   cols = everything(),
                                   names_to = "model",
                                   values_to = "prediction")

  # True value of f(x0)
  true_value <- 10

  # Calculate Bias, Variance, and MSE for each model
  statistics <- predictions_long %>%
    group_by(model) %>%
    summarise(
      Bias = mean(prediction) - true_value,
      Variance = var(prediction),
      MSE = mean((prediction - true_value)^2),
      .groups = 'drop'
    )

  return(statistics)
}

p_values <- c(10, 20, 30, 40, 50)
results <- map_dfr(p_values, run_simulation1, .id = "p")
results$p <- as.factor(results$p)
results_long <- pivot_longer(results,
                             cols = c(Bias, Variance),
                             names_to = "statistic",
                             values_to = "value")

ggplot(results_long, aes(x = p, y = value, fill = model)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~statistic, scales = "free_y") +
  labs(title = "Bias and Variance for OLS, LASSO, and Ridge
       with different number of irrelevant regressors",
       x = "Number of irrelevant regressors", y = "Value") +
  scale_fill_brewer(palette = "Set1") +
  scale_x_discrete(labels = p_values)
```



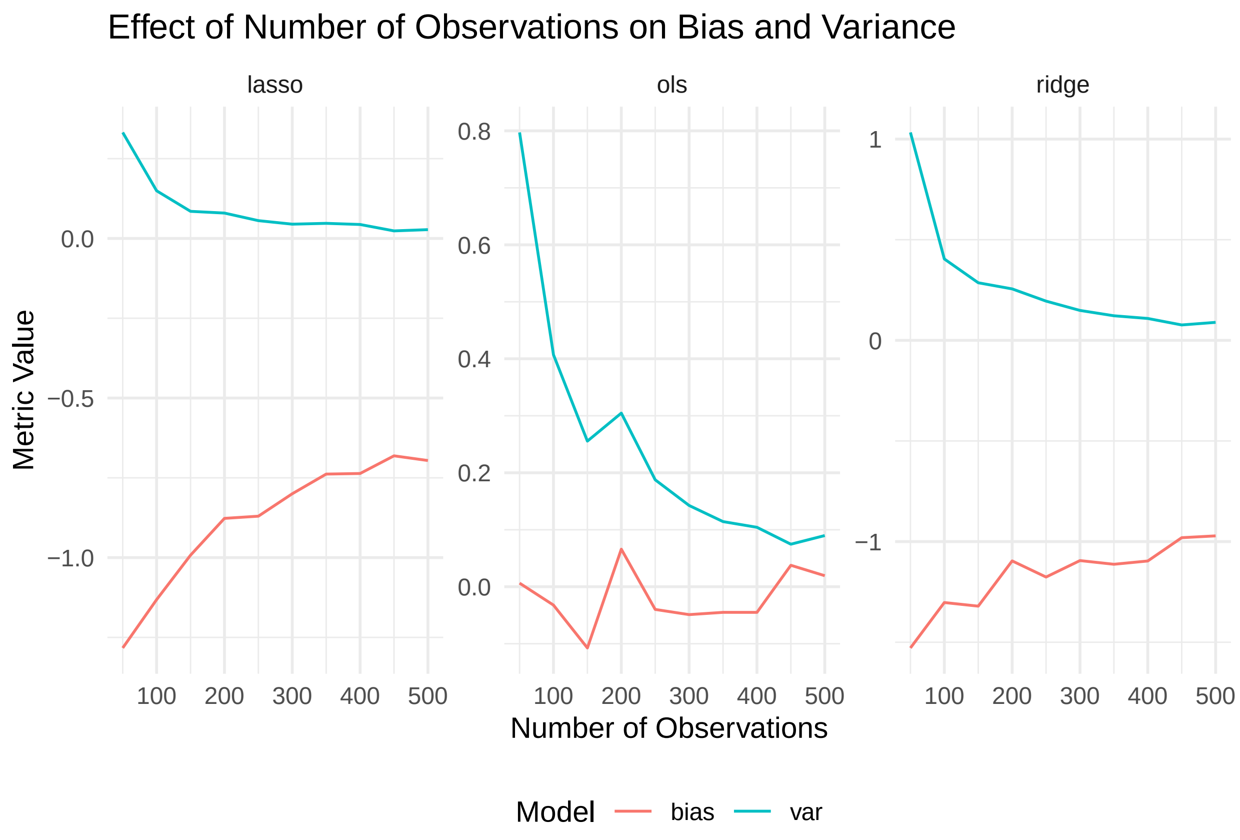
Interpretation: Bias: As the number of irrelevant regressors increases, the bias of OLS regression remains relatively constant. In contrast, the bias of regularized models grows with the addition of more irrelevant regressors.

Variance: The variance of LASSO regression remains consistent as the number of irrelevant regressors increases, what can be explained by the regularization effect that drives coefficients towards zero. Conversely, the variance of OLS and Ridge regression increases with the inclusion of more irrelevant regressors.

5. How do bias and variance depend on the number of observations?

```r
library(glmnet)
library(tidyverse)
library(MASS)

# Function to run simulation for a given number of observations
run_simulation <- function(nobs) {
  nsim <- 100
  p <- 10
  beta <- c(2, 3, rep(0, p - 2))
  x0 <- rep(2, p)

  # Vectors to store results
  results <- tibble(
    nobs = rep(nobs, nsim),
    f0_ols = numeric(nsim),
    f0_lasso = numeric(nsim),
    f0_ridge = numeric(nsim)
  )

  for (s in 1:nsim) {
    X <- MASS::mvrnorm(n = nobs, mu = rep(0, p), Sigma = diag(p))
    y <- X %*% beta + rnorm(nobs)

    # OLS fit
    fitOLS <- lm(y ~ X - 1)
    results$f0_ols[s] <- as.numeric(coef(fitOLS) %*% x0)

    # LASSO fit
    fitLASSO <- cv.glmnet(X, y, alpha = 1)
    results$f0_lasso[s] <- as.numeric(predict(fitLASSO,
                                              newx = matrix(x0, ncol = p),
                                              s = "lambda.1se"))

    # Ridge fit
    fitRidge <- cv.glmnet(X, y, alpha = 0)
    results$f0_ridge[s] <- as.numeric(predict(fitRidge,
                                              newx = matrix(x0, ncol = p),
                                              s = "lambda.1se"))
  }

  # Calculate bias and variance
  stats <- results %>%
    summarise(
      nobs = first(nobs),
      bias_ols = mean(f0_ols) - 10,
      var_ols = var(f0_ols),
      bias_lasso = mean(f0_lasso) - 10,
      var_lasso = var(f0_lasso),
      bias_ridge = mean(f0_ridge) - 10,
      var_ridge = var(f0_ridge)
    )

  stats
}

# Set of observations to run the simulations on
nobs_values <- seq(50, 500, by = 50)

# Run simulations and collect results
results <- map_df(nobs_values, run_simulation)

# Prepare data for plotting
results_long <- pivot_longer(results,
                             cols = -nobs,
                             names_to = "model_metric",
                             values_to = "value")

# Separate metric and model
```

```r
results_long <- results_long %>%
  separate(model_metric, into = c("model", "metric"), sep = "_") %>%
  mutate(model = recode(model, ols = "OLS", lasso = "LASSO", ridge = "Ridge"),
         metric = recode(metric, bias = "Bias", var = "Variance"))

# Plotting the results
ggplot(results_long,
       aes(x = nobs, y = value, group = interaction(model, metric),
           color = model)) +
  geom_line() +
  facet_wrap(~metric, scales = "free_y") +
  labs(title = "Effect of Number of Observations on Bias and Variance",
       x = "Number of Observations",
       y = "Metric Value",
       color = "Model") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

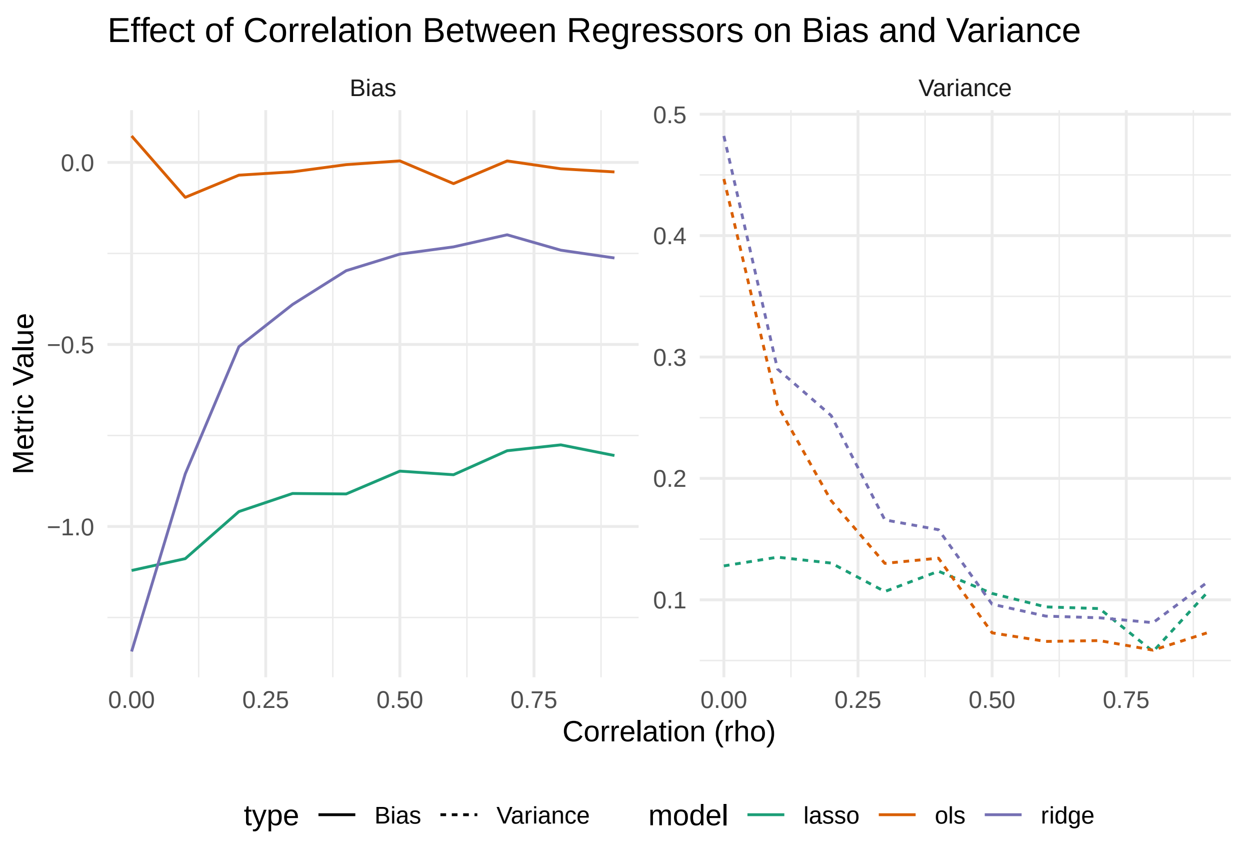


Interpretation:

The OLS model shows the most significant decrease in variance with increased sample size, which is expected as OLS tends to have higher variance with smaller samples. LASSO and Ridge show less variance across all sample sizes due to their regularization terms, which penalize large coefficients. Bias in OLS and Ridge reduces as more data is added, while LASSO's bias remains relatively constant. The decrease in bias for OLS suggests the model is benefiting from more data, as it can estimate the true model parameters more accurately with larger samples. The slight decrease in Ridge's bias with more observations suggests it too benefits from larger sample sizes, though its primary advantage is in variance reduction.

6. The simulation above assumes that regressors are uncorrelated. How do results change when correlation between regressors is instead given by > 0?

```r
library(glmnet)
library(tidyverse)
library(MASS)

run_simulation_corr <- function(rho, nobs = 100, nsim = 100) {
  p <- 10
  beta <- c(2, 3, rep(0, p - 2))
  x0 <- rep(2, p)

  # Vectors to store results
  sim_results <- tibble(
    rho = rho,
    f0_ols = numeric(nsim),
    f0_lasso = numeric(nsim),
    f0_ridge = numeric(nsim)
  )

  # Construct the correlation matrix
  Sigma <- matrix(rho, nrow = p, ncol = p)
  diag(Sigma) <- 1

  for (s in 1:nsim) {
    X <- MASS::mvrnorm(n = nobs, mu = rep(0, p), Sigma = Sigma)
    y <- X %*% beta + rnorm(nobs)

    # OLS fit
    fitOLS <- lm(y ~ X - 1)
    sim_results$f0_ols[s] <- as.numeric(coef(fitOLS) %*% x0)

    # LASSO fit
    fitLASSO <- cv.glmnet(X, y, alpha = 1)
    sim_results$f0_lasso[s] <- as.numeric(predict(fitLASSO,
                                                  newx = matrix(x0, ncol = p),
                                                  s = "lambda.1se"))

    # Ridge fit
    fitRidge <- cv.glmnet(X, y, alpha = 0)
    sim_results$f0_ridge[s] <- as.numeric(predict(fitRidge,
                                                  newx = matrix(x0, ncol = p),
                                                  s = "lambda.1se"))
  }

  # Calculate bias and variance
  stats <- sim_results %>%
    summarise(
      bias_ols = mean(f0_ols) - 10,
      var_ols = var(f0_ols),
      bias_lasso = mean(f0_lasso) - 10,
      var_lasso = var(f0_lasso),
      bias_ridge = mean(f0_ridge) - 10,
      var_ridge = var(f0_ridge)
    )

  cbind(stats, rho = rho) # Add rho to the results
}

# Set of correlations to run the simulations on
rho_values <- seq(0, 0.9, by = 0.1)

# Run simulations and collect results
results_corr <- map_df(rho_values, run_simulation_corr)

# Plotting the results
results_corr_long <- pivot_longer(results_corr,
                                  cols = starts_with("bias_"),
                                  names_to = "model",
                                  values_to = "value",
                                  names_prefix = "bias_")
results_corr_long$type <- "Bias"

variance_corr_long <- pivot_longer(results_corr,
                                   cols = starts_with("var_"),
                                   names_to = "model",
                                   values_to = "value",
                                   names_prefix = "var_")
variance_corr_long$type <- "Variance"

metrics_corr_long <- bind_rows(results_corr_long, variance_corr_long)

ggplot(metrics_corr_long, aes(x = rho, y = value, color = model, linetype = type)) +
  geom_line() +
  facet_wrap(~type, scales = "free_y") +
  labs(title = "Effect of Correlation Between Regressors on Bias and Variance",
       x = "Correlation (rho)",
       y = "Metric Value") +
  theme_minimal() +
  scale_color_brewer(palette = "Dark2") +
  theme(legend.position = "bottom")
```



Interpretation:

The behavior of LASSO and Ridge in terms of bias is as expected; they introduce some bias as a trade-off for lower variance. The reduction in variance across all models with increased correlation suggests a regularization effect, which is stronger for LASSO and Ridge. The slight increase in bias for OLS with higher correlation could be a result of the instability of coefficient estimates when predictors are correlated, but this warrants further investigation to confirm. The flattening of variance at high correlation levels for LASSO and Ridge suggests that beyond a certain point, increasing correlation does not lead to further reductions in variance, indicating a limit to how much the models can compensate for multicollinearity.