DSA 8020 R Session 3: Multiple Linear Regression II

Whitney Huang, Clemson University

Contents

Species diversity on the Galapagos Islands 1
Load the data 1
General Linear F-Test 1
Visualizing the F-Test 3
Prediction 5
Multicollinearity 6
Key Takeaways 13

Species diversity on the Galapagos Islands

We use the `gala` dataset from the `faraway` package.
For this analysis, we remove the variable *Endemics* to focus on modeling total species counts.

Load the data

```
# Load dataset
library(faraway)
data(gala)

# Remove "Endemics" column (2nd column) to simplify analysis
galaNew <- gala[, -2] # removing "Endemics"
```

General Linear F-Test

We use the general linear F-test to compare two **nested models**:

- Reduced model: simpler model (fewer predictors)
- Full model: more complex model (additional predictors)

The goal is to test whether the additional predictor(s) significantly improve the model.

```r
## First example

# Reduced model: uses Elevation only
M1 <- lm(Species ~ Elevation, data = galaNew)
summary(M1)
```

```
##
## Call:
## lm(formula = Species ~ Elevation, data = galaNew)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -218.319  -30.721  -14.690    4.634  259.180
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.33511   19.20529   0.590     0.56
## Elevation    0.20079    0.03465   5.795 3.18e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 78.66 on 28 degrees of freedom
## Multiple R-squared:  0.5454, Adjusted R-squared:  0.5291
## F-statistic: 33.59 on 1 and 28 DF,  p-value: 3.177e-06
```

```r
# Full model: adds Area as an additional predictor
M2 <- lm(Species ~ Elevation + Area, data = galaNew)
summary(M2)
```

```
##
## Call:
## lm(formula = Species ~ Elevation + Area, data = galaNew)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -192.619  -33.534  -19.199    7.541  261.514
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17.10519   20.94211   0.817  0.42120
## Elevation    0.17174    0.05317   3.230  0.00325 **
## Area         0.01880    0.02594   0.725  0.47478
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79.34 on 27 degrees of freedom
## Multiple R-squared:  0.554,  Adjusted R-squared:  0.521
## F-statistic: 16.77 on 2 and 27 DF,  p-value: 1.843e-05
```

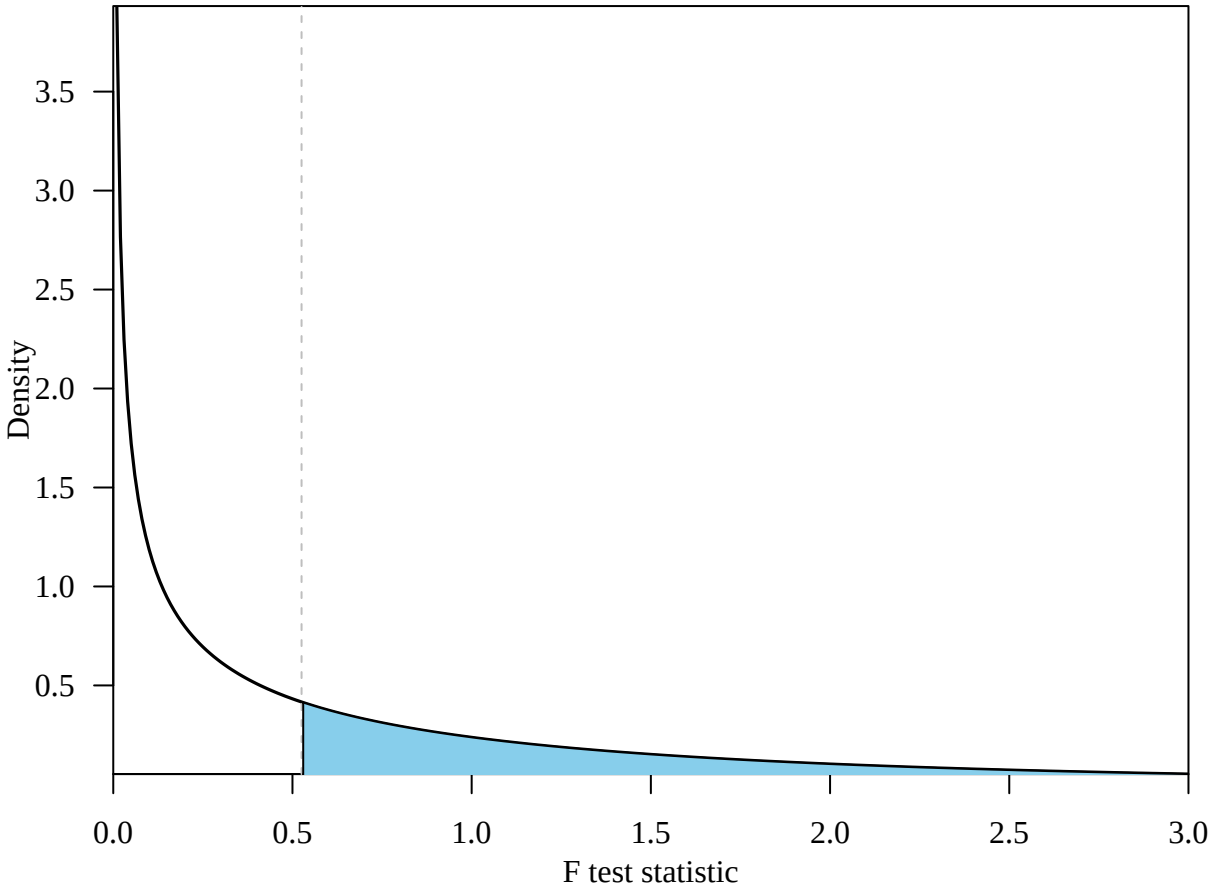
```r
# Perform general linear F-test:
# H0: Area does NOT improve the model (beta_Area = 0)
# H1: Area improves the model (beta_Area != 0)
anova(M1, M2)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Elevation
## Model 2: Species ~ Elevation + Area
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     28 173254
## 2     27 169947  1      3307 0.5254 0.4748
```

Visualizing the F-Test

The shaded region represents the p-value (right tail of the F-distribution).

```r
# Plot F distribution with df1 = 1 and df2 = 27
par(las = 1, mar = c(4, 4, 1, 0.5), mgp = c(2, 1, 0), family = "serif")

# Create grid for F-distribution
xg <- seq(0, 3, 0.01)
yg <- df(xg, 1, 27)

# Plot density curve
plot(xg, yg, type = "l",
     xaxs = "i", yaxs = "i",
     lwd = 1.6, xlab = "F test statistic", ylab = "Density")

# Observed test statistic (from anova output)
abline(v = 0.5254, lty = 2, col = "gray")

# Shade right-tail area (p-value)
polygon(c(xg[xg > 0.5254], rev(xg[xg > 0.5254])),
        c(yg[xg > 0.5254],
          rep(0, length(yg[xg > 0.5254]))),
        col = "skyblue")
```



Second Example

Compare a reduced model with selected predictors to the full model including all predictors.

```
# Full model: includes all available predictors
Full <- lm(Species ~ ., data = galaNew)

# Reduced model: includes only Elevation and Adjacent
Reduce <- lm(Species ~ Elevation + Adjacent, data = galaNew)

# General linear F-test
# Tests whether additional predictors in Full model improve fit
anova(Reduce, Full)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ Elevation + Adjacent
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     27 100003
## 2     24  89231  3     10772 0.9657  0.425
```

Interpretation

- If the p-value is small ⇒ reject H_0: added predictors improve the model

- If the p-value is large ⇒ fail to reject H_0: simpler model is sufficient

This test helps determine whether additional variables meaningfully contribute to explaining variation in the response.

Prediction

In this section, we use a fitted multiple linear regression model to make predictions for a "typical" individual. Here, "typical" is defined by using the median value of each predictor.

```
# Load the fat dataset
data(fat)

# Fit a multiple linear regression model
# Response: brozek
# Predictors: body measurements and age
lmod <- lm(brozek ~ age + weight + height + neck + chest + abdom + hip + thigh +
             knee + ankle + biceps + forearm + wrist,
           data = fat)
```

Next, extract the design matrix X and compute the median value for each column.
The design matrix includes the intercept and all predictors used in the model.

```
# Extract the design matrix from the fitted model
X <- model.matrix(lmod)

# Compute median value for each column of the design matrix
# This creates one representative predictor profile
(x0 <- apply(X, 2, median))
```

```
## (Intercept)         age      weight      height        neck       chest
##        1.00       43.00      176.50       70.00       38.00       99.65
##       abdom         hip       thigh        knee       ankle      biceps
##       90.95       99.30       59.00       38.50       22.80       32.05
##     forearm       wrist
##       28.70       18.30
```

Now we compute the predicted value manually and compare it with the result from `predict()`.

We also obtain: * A prediction interval for a future individual observation

- A confidence interval for the mean response

```
# Manually compute predicted response:
# y0 = x0' * beta_hat
(y0 <- sum(x0 * coef(lmod)))
```

```
## [1] 17.49322
```

```r
# Predicted value using predict()
predict(lmod, new = data.frame(t(x0)))
```

```
##        1
## 17.49322
```

```r
# Prediction interval for one future observation
# This interval includes individual-level random variation
predict(lmod, new = data.frame(t(x0)), interval = "prediction")
```

```
##        fit     lwr      upr
## 1 17.49322 9.61783 25.36861
```

```r
# Confidence interval for the mean response
# This interval describes uncertainty in the estimated average response
predict(lmod, new = data.frame(t(x0)), interval = "confidence")
```

```
##        fit      lwr      upr
## 1 17.49322 16.94426 18.04219
```

Multicollinearity

In this section, we use a Monte Carlo simulation to study the effect of multicollinearity on regression coefficient estimates.

The true linear model is

$$y = 4 + 0.8x_1 + 0.6x_2 + \epsilon.$$

We compare two scenarios:

1. **High multicollinearity:** x_1 and x_2 are strongly correlated with $\rho = 0.9$.
2. **No multicollinearity:** x_1 and x_2 are independent with $\rho = 0$.

For each scenario, we repeat the simulation 500 times. In each repetition, we:

1. Generate a sample of size $n = 30$.
2. Generate predictors x_1 and x_2.
3. Generate the response y from the true model.
4. Fit the multiple linear regression model.
5. Store the estimated regression coefficients and R^2.

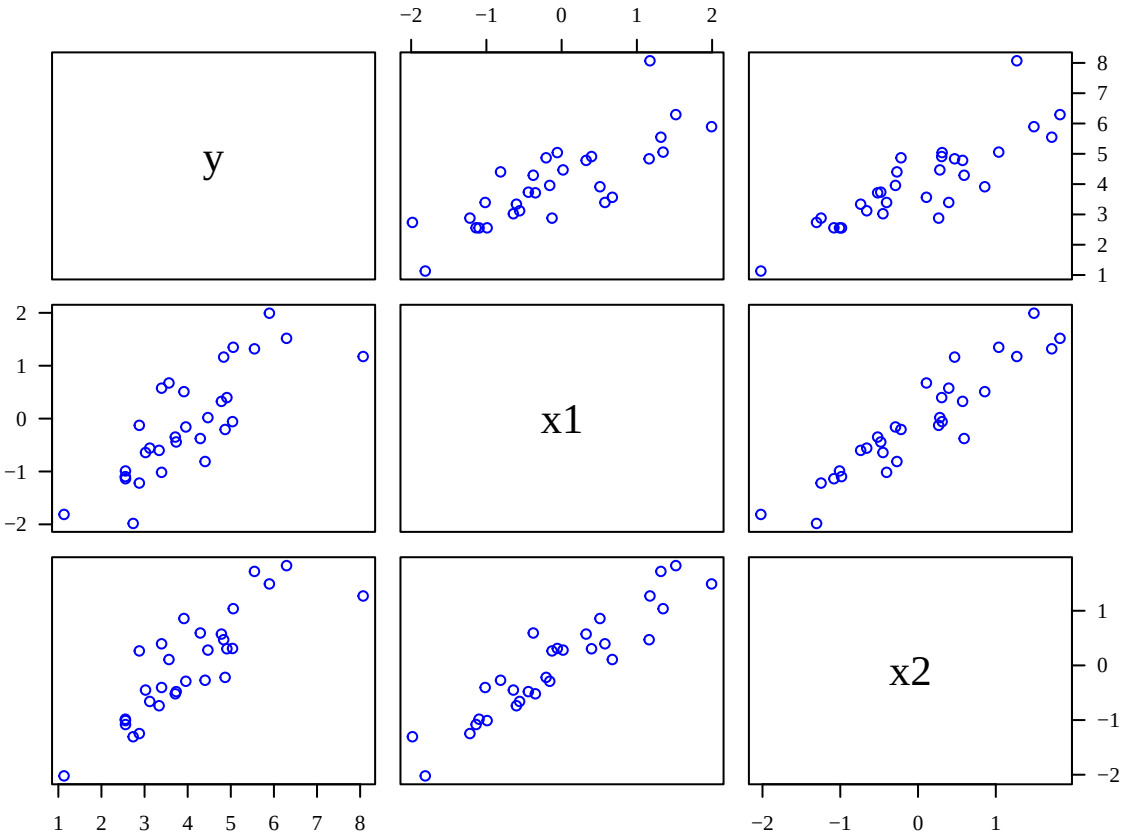
The goal is to see how multicollinearity affects the stability of coefficient estimates, even when the overall model fit remains strong.

```r
# Set seed for reproducibility
set.seed(123)

# Number of Monte Carlo repetitions
N <- 500
```

```r
# Sample size for each simulated dataset
n <- 30

# Load MASS package for multivariate normal simulation
library(MASS)

# True regression coefficients
beta0 <- 4
beta1 <- 0.8
beta2 <- 0.6

# Correlation between x1 and x2 under multicollinearity
rho <- 0.9

# Covariance matrix for highly correlated predictors
Sigma_collinear <- matrix(c(1, rho, rho, 1), nrow = 2)

# Generate N simulated datasets of correlated predictors
# The resulting array has dimensions: n x 2 x N
x <- replicate(N, mvrnorm(n = n, mu = c(0, 0),
                          Sigma = Sigma_collinear))

# Create storage for simulated responses
y <- array(dim = c(n, N))

# Generate response values for each simulated dataset
for (i in 1:N) {
  y[, i] <- beta0 +
    beta1 * x[, 1, i] +
    beta2 * x[, 2, i] +
    rnorm(n, mean = 0, sd = 1)
}
```

Examine One Simulated Dataset Before summarizing all 500 simulations, we first examine the first simulated dataset.
This helps us see what multicollinearity looks like in one example.

```r
# Extract the first simulated dataset
sim1 <- data.frame(
  y  = y[, 1], x1 = x[, 1, 1], x2 = x[, 2, 1])

# Scatterplot matrix to visualize pairwise relationships
par(las = 1, mar = c(4, 4, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
pairs(sim1, las = 1, col = "blue")
```




```
# Correlation matrix
# Large correlation between x1 and x2 indicates multicollinearity
cor(sim1)
```

```
##            y        x1        x2
## y  1.0000000 0.7987777 0.8481084
## x1 0.7987777 1.0000000 0.9281514
## x2 0.8481084 0.9281514 1.0000000
```

```
# Compute variance inflation factors (VIFs)
# VIF values larger than 5 or 10 suggest serious multicollinearity
library(faraway)
vif(lm(y ~ x1 + x2, data = sim1))
```

```
##       x1       x2
## 7.218394 7.218394
```

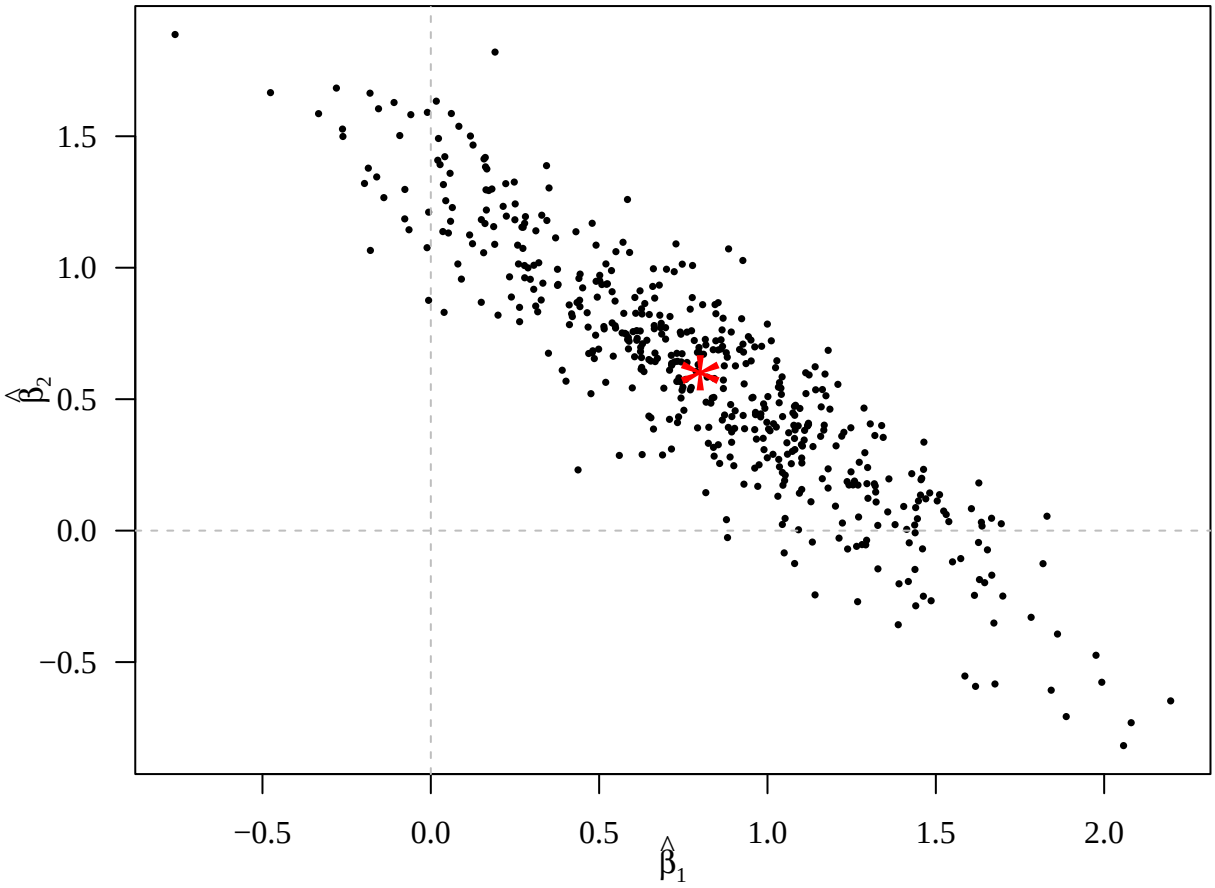
Examine Coefficient Estimates Under Multicollinearity Now we fit the regression model to each simulated dataset and store the estimated coefficients.

If multicollinearity is severe, the coefficient estimates for x_1 and x_2 may vary substantially from one simulation to another, even though the true values are fixed.

```
# Store estimated regression coefficients
# Rows: intercept, beta1, beta2
```

```r
# Columns: Monte Carlo repetitions
beta_hat_collinear <- array(dim = c(3, N))

# Store R-squared values
R2_collinear <- numeric(N)

# Fit model repeatedly
for (i in 1:N) {

  # Fit multiple linear regression for the i-th simulated dataset
  fit_i <- lm(y[, i] ~ x[, 1, i] + x[, 2, i])

  # Save estimated coefficients
  beta_hat_collinear[, i] <- coef(fit_i)

  # Save R-squared
  R2_collinear[i] <- summary(fit_i)$r.squared
}
```

```r
# Plot estimated beta1 and beta2 values across simulations
par(las = 1, mar = c(4, 4, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
plot(beta_hat_collinear[2, ], beta_hat_collinear[3, ],
     pch = 16, cex = 0.5,
     xlab = expression(hat(beta)[1]),
     ylab = expression(hat(beta)[2]))

# Add true coefficient values
points(beta1, beta2, pch = "*", cex = 3, col = "red")

# Reference lines
abline(h = 0, lty = 2, col = "gray")
abline(v = 0, lty = 2, col = "gray")
```



Interpretation:

Each point represents one simulated dataset.
The red star shows the true coefficient values $(0.8, 0.6)$.

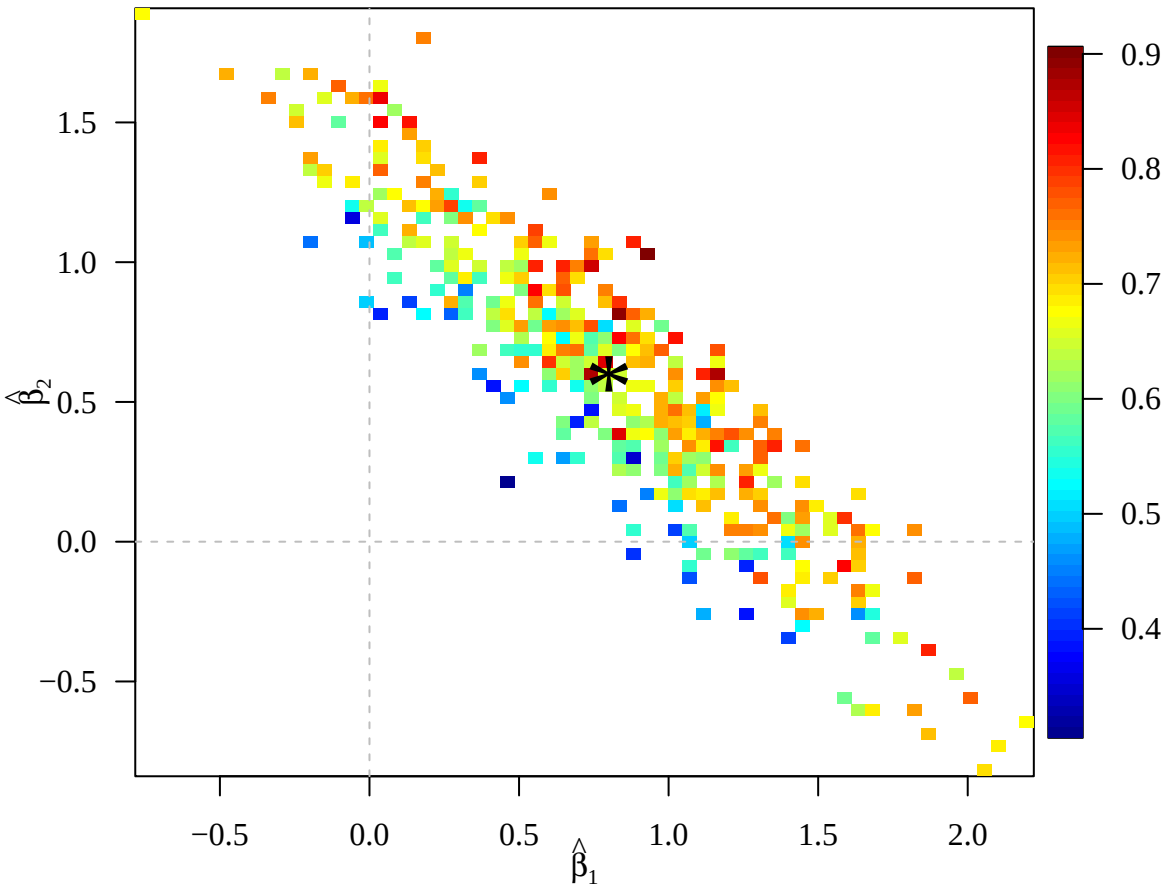
Under multicollinearity, the estimated coefficients can be highly unstable.
The model may have difficulty separating the individual effects of x_1 and x_2 because they contain overlapping information.

```
# Summarize R-squared values across simulations
summary(R2_collinear)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3099  0.6049  0.6776  0.6630  0.7343  0.9016
```

```
# Visualize how R-squared varies over coefficient estimates
library(fields)

par(las = 1, mar = c(4, 4, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
quilt.plot(beta_hat_collinear[2, ], beta_hat_collinear[3, ],
           R2_collinear,
           xlab = expression(hat(beta)[1]),
           ylab = expression(hat(beta)[2]))

# Add true coefficient values
points(beta1, beta2, pch = "*", cex = 3)
```

```r
# Reference lines
abline(h = 0, lty = 2, col = "gray")
abline(v = 0, lty = 2, col = "gray")
```



Comparison: Independent Predictors Next, we repeat the same Monte Carlo experiment, but now x_1 and x_2 are independent.

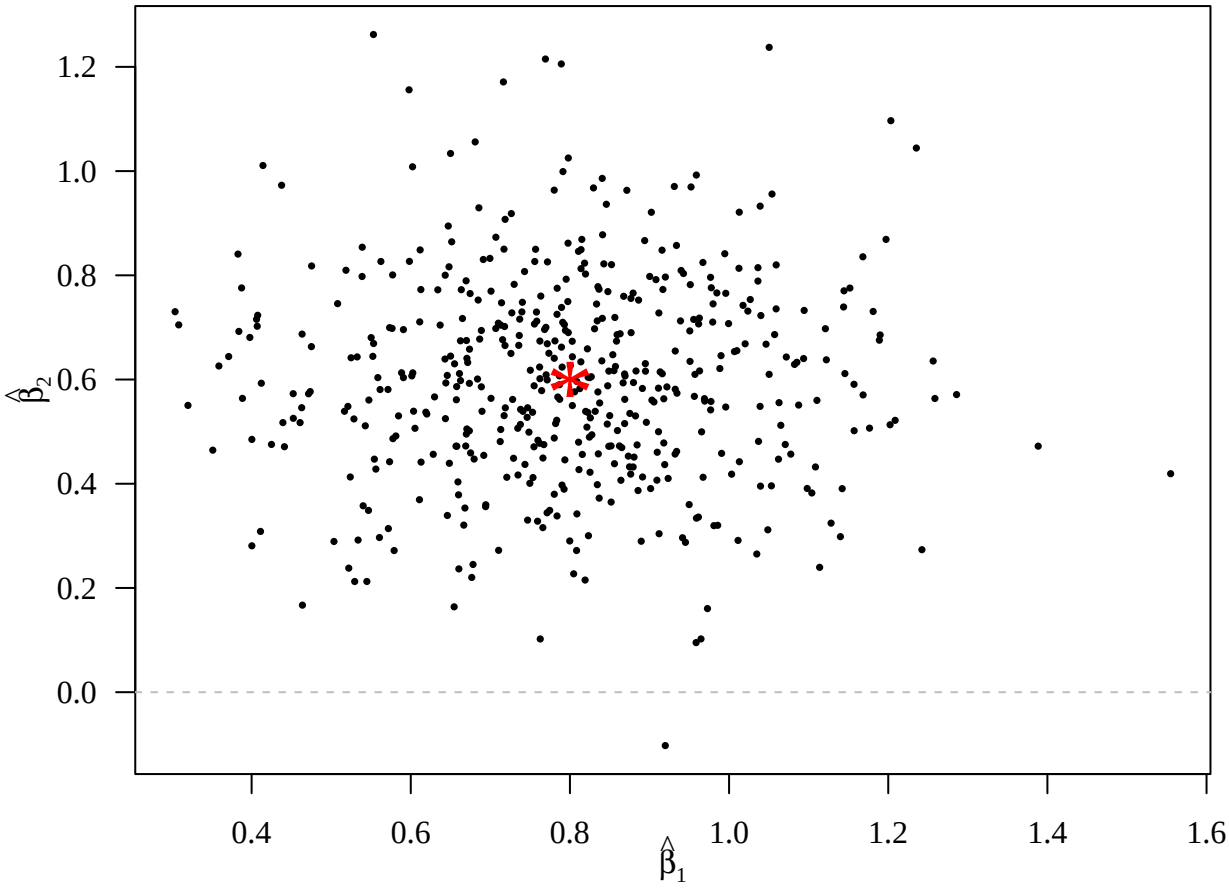
This allows us to compare the behavior of coefficient estimates with and without multicollinearity.

```r
# Covariance matrix for independent predictors
Sigma_independent <- matrix(c(1, 0, 0, 1), nrow = 2)

# Generate N simulated datasets with independent predictors
x_ind <- replicate(N, mvrnorm(n = n, mu = c(0, 0),
                              Sigma = Sigma_independent))

# Create storage for responses
y_ind <- array(dim = c(n, N))

# Generate response values
for (i in 1:N) {
  y_ind[, i] <- beta0 +
    beta1 * x_ind[, 1, i] +
    beta2 * x_ind[, 2, i] +
```

```
    rnorm(n, mean = 0, sd = 1)
}
```

```
# Store coefficient estimates and R-squared values
beta_hat_independent <- array(dim = c(3, N))
R2_independent <- numeric(N)

# Fit models repeatedly
for (i in 1:N) {

  # Fit multiple linear regression
  fit_i <- lm(y_ind[, i] ~ x_ind[, 1, i] + x_ind[, 2, i])

  # Save coefficient estimates
  beta_hat_independent[, i] <- coef(fit_i)

  # Save R-squared
  R2_independent[i] <- summary(fit_i)$r.squared
}
```

```
par(las = 1, mar = c(4, 4, 1, 0.5), mgp = c(2, 1, 0), family = "serif")
# Plot estimated beta1 and beta2 values for independent predictors
plot(beta_hat_independent[2, ], beta_hat_independent[3, ],
     pch = 16, cex = 0.5,
     xlab = expression(hat(beta)[1]),
     ylab = expression(hat(beta)[2]))
# Add true coefficient values
points(beta1, beta2, pch = "*", cex = 3, col = "red")

# Reference lines
abline(h = 0, lty = 2, col = "gray")
abline(v = 0, lty = 2, col = "gray")
```




```
# Summarize R-squared values for independent predictor case
summary(R2_independent)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1179  0.4375  0.5325  0.5181  0.6062  0.8419
```

```
# Check VIF for the first independent simulated dataset
sim_ind1 <- data.frame(
  y  = y_ind[, 1],
  x1 = x_ind[, 1, 1],
  x2 = x_ind[, 2, 1]
)

vif(lm(y ~ x1 + x2, data = sim_ind1))
```

```
##       x1       x2
## 1.042404 1.042404
```

Key Takeaways

- Multicollinearity does not necessarily make prediction poor.
- However, it can make individual coefficient estimates unstable.
- When predictors are highly correlated, the model has difficulty separating their individual effects.

- This often leads to large standard errors and high VIF values.
- R^2 may still be high, even when coefficient estimates are unreliable.

In short:

Multicollinearity mainly affects interpretation of individual predictors, not necessarily overall prediction.