Bias, Efficiency, and the Gauss Markov Theorem

Ben Gilbert

9/1/2021

Bias, Efficiency, and the Gauss Markov Theorem

Overview

This program illustrates bias and inefficiency when Gauss Markov assumptions fail:

- **Bias**: Deviation of expected value of sample parameter estimate from "true" population parameter
- **Efficiency**: The variance of the sample parameter estimate should be as small as possible
- **Consistency**: Distribution of sample parameter estimate should converge to population value as sample size grows

The Gauss-Markov theorem states that OLS (Ordinary Least Squares) is the Best (lowest variance) Linear Unbiased Estimator (BLUE) IF:

1. True model is linear in parameters and residuals:

$$y_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + e_t$$

2. X variables (right hand side) are not constants or perfectly correlated with each other
3. Residuals "e" have constant variance (homoskedasticity vs. heteroskedasticity)
    - Not more noisy for some X's than others
4. Residuals "e" are uncorrelated with each other
    - No peer effects, no serial correlation
5. All X variables are uncorrelated with the residual e
    - Observed X is not picking up some unobserved or uncontrolled factor

In each case we will run the linear regression

$$y_t = \beta_0 + \beta_1 x_{1t} + \beta_2 x_{2t} + e_t$$

on the data, but the "true" model or "data generating process" is different.

Environment Setup

Loading Required Packages

Load (and install if necessary) any packages that we want to use.

```
# Set working directory (adjust path as needed)
setwd("C:/Users/gilbe/Dropbox/Econometrics/TimeSeriesCourse")

# Load required packages
# install.packages("MASS") # Uncomment to install if needed
library(MASS)              # For multivariate normal distribution
# install.packages("car")  # Uncomment to install if needed
library(car)               # For scatterplot matrices
```

---

1. Violating Assumption 1: Non-linear Model

When the true model is not linear in parameters, OLS estimates will be biased.

Data Generation

```
# Set seed for reproducibility
set.seed(826)

# Create covariance matrix of x1, x2, and e
covariance_matrix <- matrix(c(4,1,0,1,2,0,0,0,1), 3, 3)
covariance_matrix
```

```
##      [,1] [,2] [,3]
## [1,]    4    1    0
## [2,]    1    2    0
## [3,]    0    0    1
```

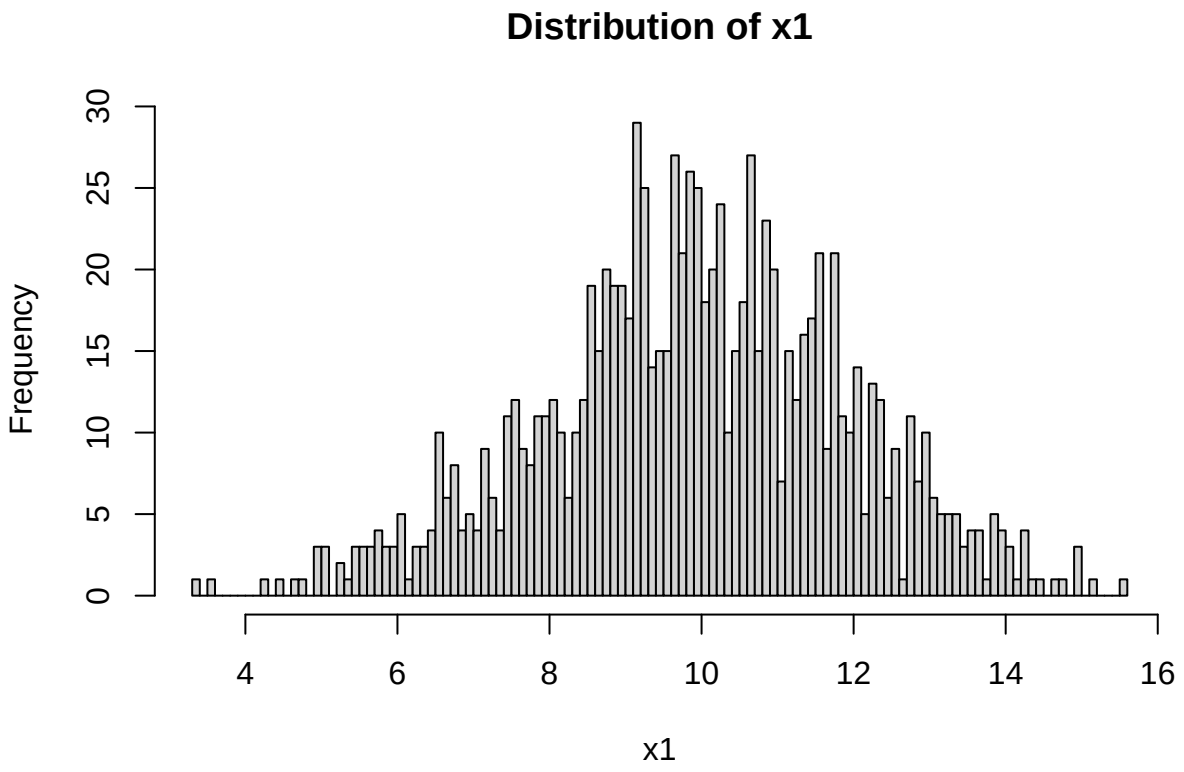
```
# Notice: e does not covary with x1 or x2 (assumption 5)
# Also x1 and x2 can covary, but not perfectly (assumption 2)

# Define mean vector for x1, x2, and e
mean_vector <- c(10, 3, 0)

# Generate multivariate normal data
simulated_data <- mvrnorm(n=1000, mu=mean_vector, Sigma=covariance_matrix)

# Assign column names
colnames(simulated_data) <- c("x1", "x2", "e")

# Convert to data frame for easier manipulation
simulated_data <- as.data.frame(simulated_data)
head(simulated_data)
```

```
##          x1        x2          e
## 1  9.815559 2.8349325  1.0499626
## 2  8.443130 2.1208185  0.7884180
## 3  9.618013 3.3454521 -0.8777746
## 4  6.371419 1.7631024  0.6244005
## 5 10.091449 0.6709607  0.2481989
## 6  5.828882 1.1333412  0.4228365
```

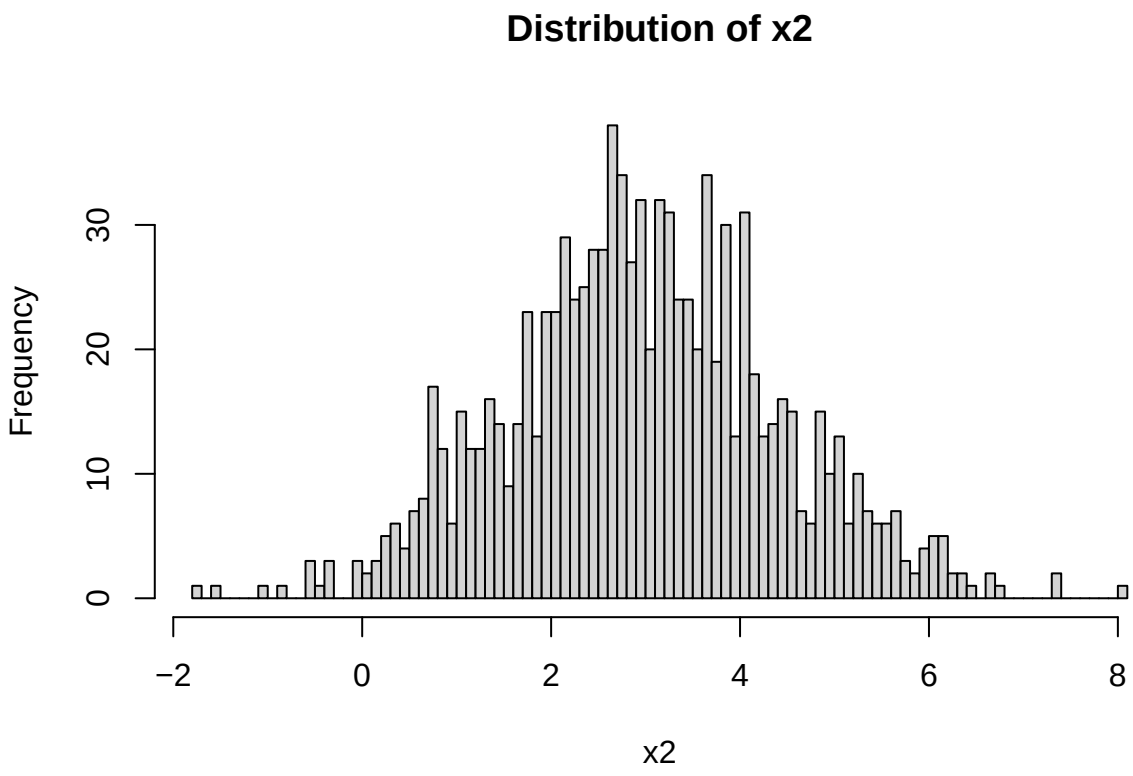
Exploratory Data Analysis

Visualize the distributions of the variables:

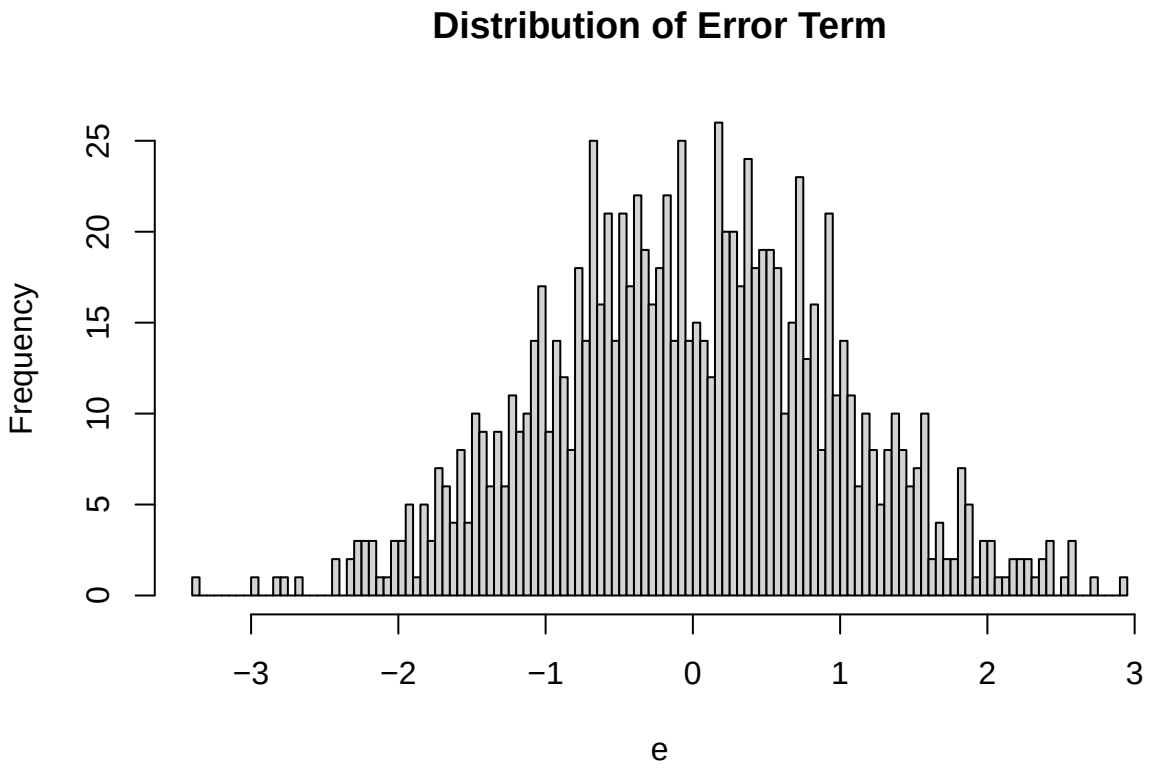
```
# Plot histograms of each variable
hist(simulated_data$x1, breaks = 100, main="Distribution of x1",
     xlab="x1", ylab="Frequency")
```




```
hist(simulated_data$x2, breaks = 100, main="Distribution of x2",
     xlab="x2", ylab="Frequency")
```




```
hist(simulated_data$e, breaks = 100, main="Distribution of Error Term",
     xlab="e", ylab="Frequency")
```



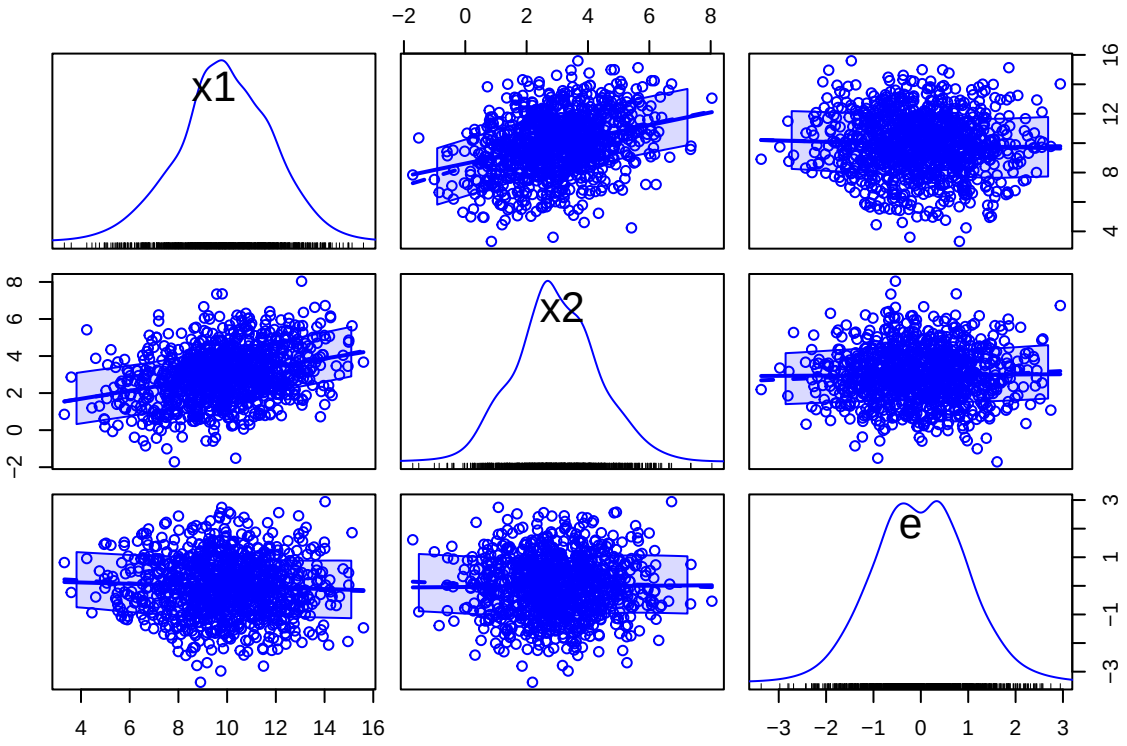
```r
# Calculate and display correlation and covariance matrices
cov(simulated_data)
```

```
##             x1         x2          e
## x1  4.02081193 0.87289399 -0.09254668
## x2  0.87289399 2.00633796  0.01490771
## e  -0.09254668 0.01490771  1.00873015
```

```r
cor(simulated_data)
```

```
##             x1         x2          e
## x1  1.00000000 0.30732832 -0.04595327
## x2  0.30732832 1.00000000  0.01047904
## e  -0.04595327 0.01047904  1.00000000
```

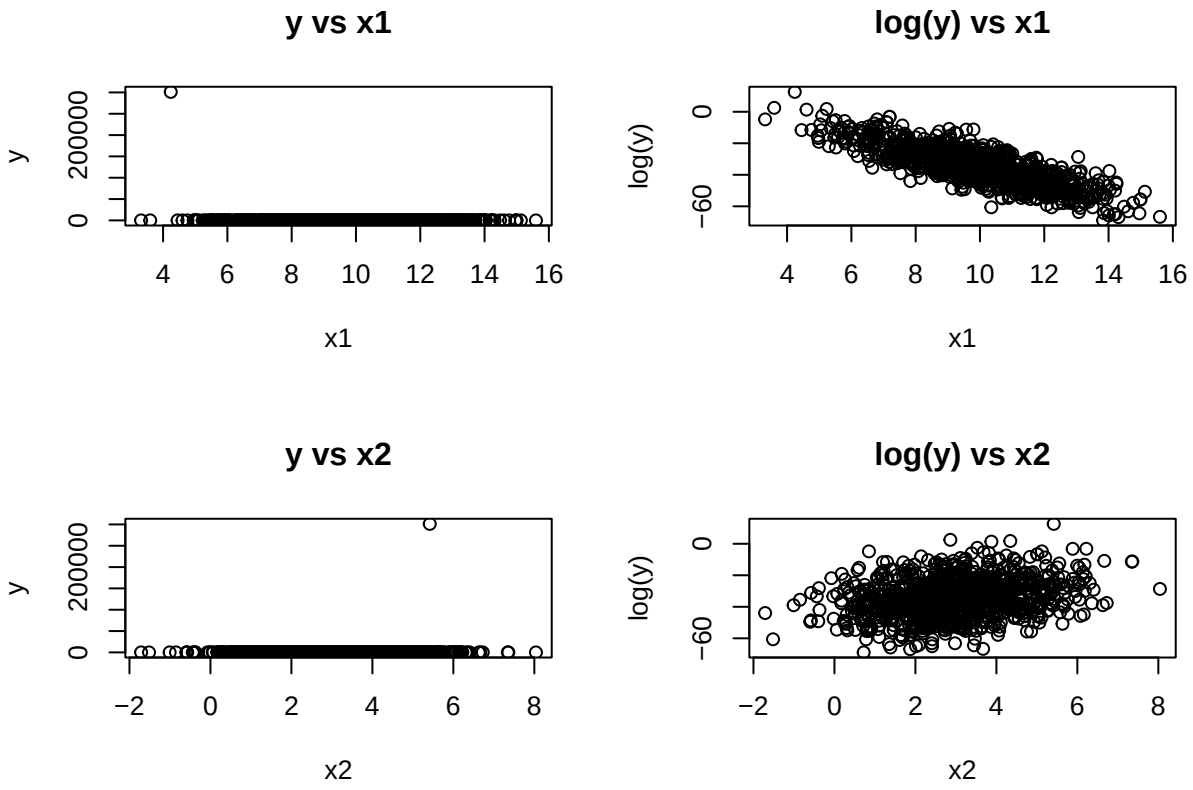
```r
# Create scatterplot matrix to visualize relationships
scatterplotMatrix(simulated_data)
```



Generate Non-linear Outcome Variable

```r
# Generate exponential outcome (non-linear in parameters)
outcome_y <- exp(10 - 6*simulated_data$x1 + 5*simulated_data$x2 + simulated_data$e)
# Note: log(y) is linear in parameters and residual, but y is not

# Plot relationships between predictors and outcome
par(mfrow=c(2,2))
```

```r
plot(simulated_data$x1, outcome_y, main="y vs x1", xlab="x1", ylab="y")
plot(simulated_data$x1, log(outcome_y), main="log(y) vs x1", xlab="x1", ylab="log(y)")
plot(simulated_data$x2, outcome_y, main="y vs x2", xlab="x2", ylab="y")
plot(simulated_data$x2, log(outcome_y), main="log(y) vs x2", xlab="x2", ylab="log(y)")
```




```r
par(mfrow=c(1,1))
```

Model Fitting

```r
# Run linear regression on misspecified model (should be biased)
model_misspecified <- lm(outcome_y ~ x1 + x2, data=simulated_data)
summary(model_misspecified)
```

```
##
## Call:
## lm(formula = outcome_y ~ x1 + x2, data = simulated_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -3802  -1104   -264    495 296312
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4009.1     1519.8   2.638 0.008470 **
## x1            -557.4      156.9  -3.553 0.000399 ***
```

```
## x2             609.8      222.1   2.746 0.006148 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9462 on 997 degrees of freedom
## Multiple R-squared:  0.01545,    Adjusted R-squared:  0.01347
## F-statistic: 7.822 on 2 and 997 DF,  p-value: 0.000426
```

```
# Run correct model with log transformation (should be unbiased)
model_correct <- lm(log(outcome_y) ~ x1 + x2, data=simulated_data)
summary(model_correct)
```

```
##
## Call:
## lm(formula = log(outcome_y) ~ x1 + x2, data = simulated_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3647 -0.6814 -0.0060  0.6925  3.0024
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.19399    0.16125   63.22   <2e-16 ***
## x1          -6.02720    0.01665 -362.07   <2e-16 ***
## x2           5.01926    0.02357  212.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.004 on 997 degrees of freedom
## Multiple R-squared:  0.9931, Adjusted R-squared:  0.993
## F-statistic: 7.126e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

---

2. Violating Assumption 2: Perfect Multicollinearity

When X variables are perfectly correlated or constant, the model becomes unidentifiable.

Data Generation with Perfect Correlation

```
# Set seed for reproducibility
set.seed(826)

# Create covariance matrix for x1 and e
covariance_matrix <- matrix(c(4,0,0,1), 2, 2)
covariance_matrix
```

```
##      [,1] [,2]
## [1,]    4    0
## [2,]    0    1
```

```r
# Notice e does not covary with x1 (assumption 5)

# Define mean vector
mean_vector <- c(10, 0)

# Generate data
simulated_data <- mvrnorm(n=1000, mu=mean_vector, Sigma=covariance_matrix)

# Create x2 as a perfect multiple of x1 (violates assumption 2)
x2_values <- 7 * simulated_data[,1]
simulated_data <- as.data.frame(cbind(simulated_data[,1], x2_values, simulated_data[,2]))

# Assign column names
colnames(simulated_data) <- c("x1", "x2", "e")
head(simulated_data)
```

```
##         x1        x2          e
## 1 10.22234  71.55639 -0.06505297
## 2 11.68949  81.82641 -0.17189937
## 3 10.21010  71.47071  0.36952559
## 4 13.64180  95.49257  0.19523319
## 5 10.76801  75.37608 -1.73650622
## 6 14.34835 100.43846 -0.10192330
```

Exploratory Data Analysis

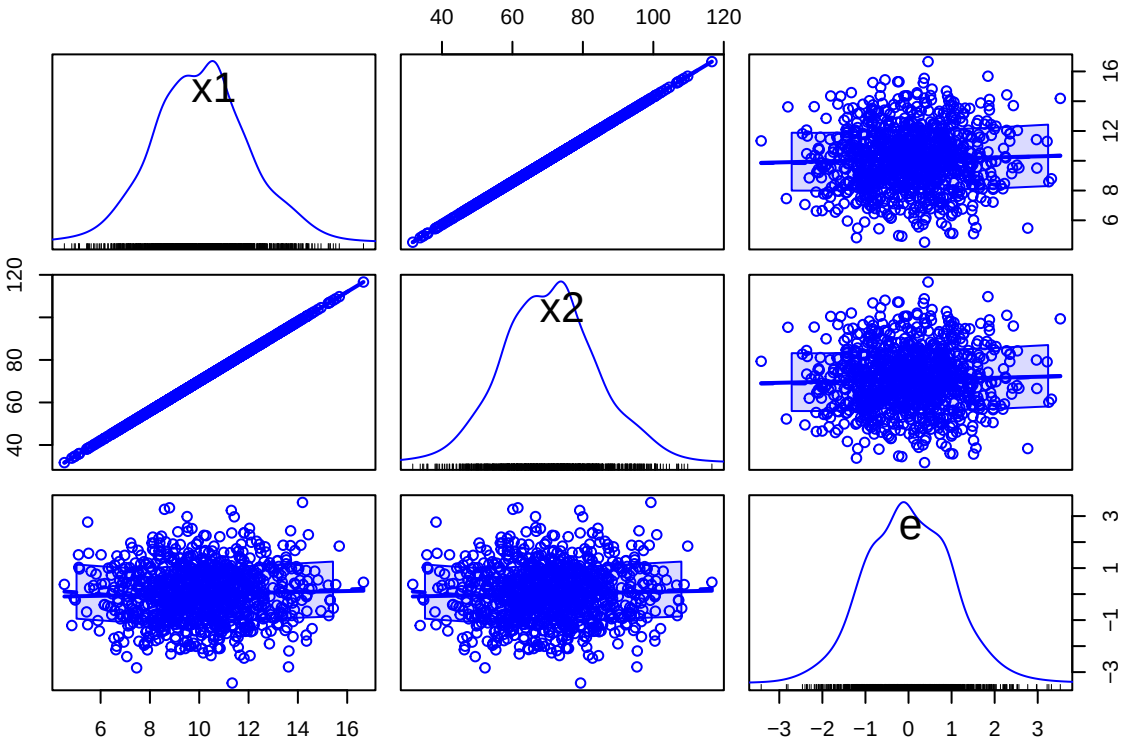
```r
# Calculate and display correlation and covariance matrices
cov(simulated_data)
```

```
##             x1          x2         e
## x1  3.93549450  27.5484615 0.07180935
## x2 27.54846153 192.8392307 0.50266547
## e   0.07180935   0.5026655 1.06201033
```

```r
cor(simulated_data)
```

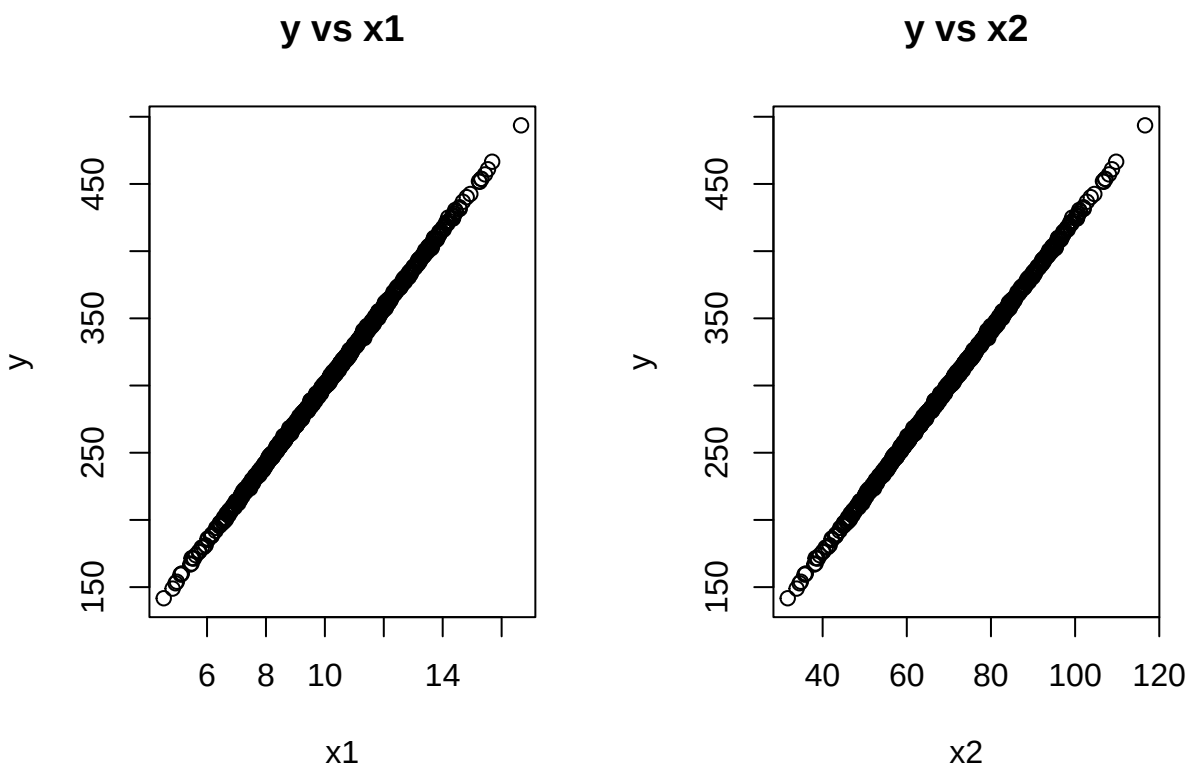
```
##            x1         x2          e
## x1 1.00000000 1.00000000 0.03512505
## x2 1.00000000 1.00000000 0.03512505
## e  0.03512505 0.03512505 1.00000000
```

```r
# Create scatterplot matrix
scatterplotMatrix(simulated_data)
```



Generate Outcome Variable

```
# Generate linear outcome
outcome_y <- 10 - 6*simulated_data$x1 + 5*simulated_data$x2 + simulated_data$e
# Note: y is linear in parameters (satisfies assumption 1)

# Plot relationships
par(mfrow=c(1,2))
plot(simulated_data$x1, outcome_y, main="y vs x1", xlab="x1", ylab="y")
plot(simulated_data$x2, outcome_y, main="y vs x2", xlab="x2", ylab="y")
```




```
par(mfrow=c(1,1))
```

Model Fitting with Perfect Multicollinearity

```
# Run linear regression (perfect multicollinearity issue)
model_multicollinear <- lm(outcome_y ~ x1 + x2, data=simulated_data)
summary(model_multicollinear)
```

```
##
## Call:
## lm(formula = outcome_y ~ x1 + x2, data = simulated_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4532 -0.7331 -0.0450  0.7074  3.4402
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.82550    0.16904   58.12   <2e-16 ***
## x1          29.01825    0.01643 1765.79   <2e-16 ***
## x2                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.03 on 998 degrees of freedom
## Multiple R-squared:  0.9997, Adjusted R-squared:  0.9997
## F-statistic: 3.118e+06 on 1 and 998 DF,  p-value: < 2.2e-16
```

```r
# Note: Intercept is 10 and coefficient is 29 = -6 + 5*7
# Some stats packages may not produce output due to perfect collinearity

# Intercept-only model (works because intercept can be constant)
mean(outcome_y)
```

```
## [1] 302.725
```

```r
summary(lm(outcome_y ~ 1))
```

```
##
## Call:
## lm(formula = outcome_y ~ 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -160.99  -39.38   -0.12   35.42  190.88
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  302.725      1.821   166.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 57.58 on 999 degrees of freedom
```

Data Generation with Constant Variable

```r
# Set seed for reproducibility
set.seed(826)

# Covariance matrix with x2 having zero variance (constant)
covariance_matrix <- matrix(c(4,0,0,0,0,0,0,0,1), 3, 3)
covariance_matrix
```

```
##      [,1] [,2] [,3]
## [1,]    4    0    0
## [2,]    0    0    0
## [3,]    0    0    1
```

```r
# Notice e does not covary with x1 or x2 (assumption 5)
# But x2 has no variance (violates assumption 2)

# Define mean vector
mean_vector <- c(10, 3, 0)

# Generate data
simulated_data <- mvrnorm(n=1000, mu=mean_vector, Sigma=covariance_matrix)

# Assign column names
```

```r
colnames(simulated_data) <- c("x1", "x2", "e")
simulated_data <- as.data.frame(simulated_data)
head(simulated_data)
```

```
##         x1 x2           e
## 1 9.777659  3  0.06505297
## 2 8.310513  3  0.17189937
## 3 9.789899  3 -0.36952559
## 4 6.358205  3 -0.19523319
## 5 9.231989  3  1.73650622
## 6 5.651648  3  0.10192330
```

```r
# Exploratory analysis
cov(simulated_data)
```

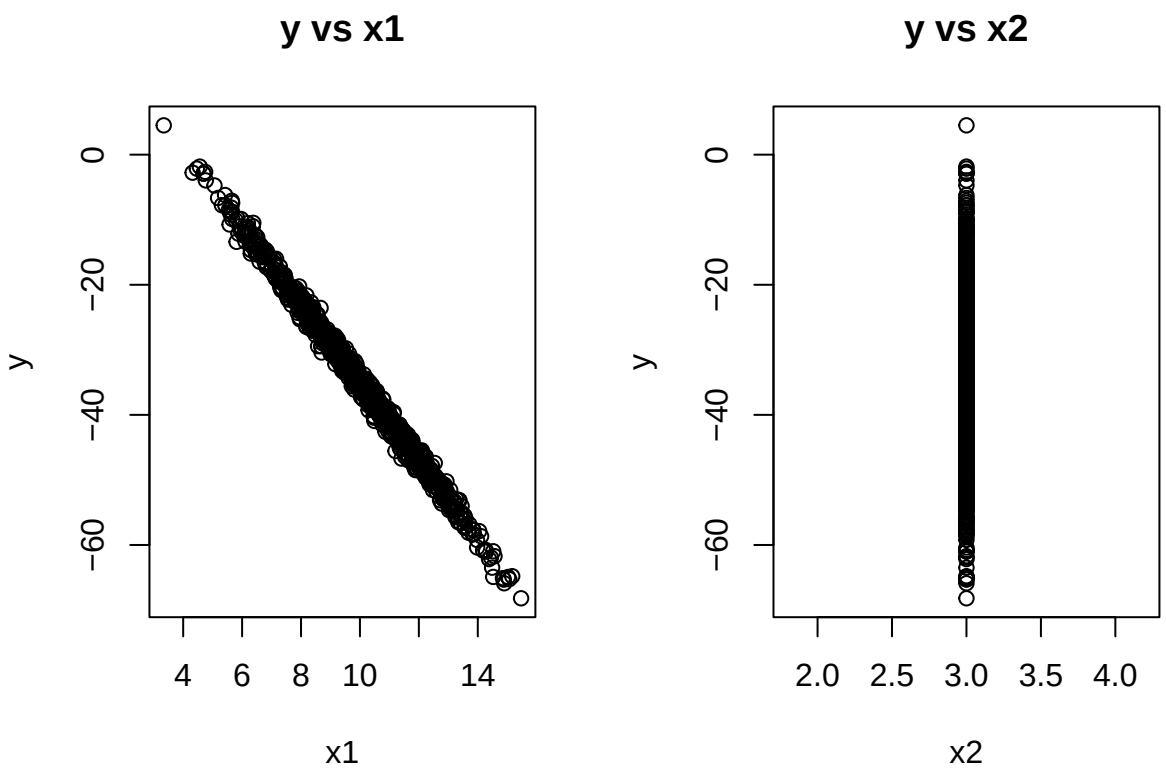
```
##            x1 x2          e
## x1 3.93549450  0 0.07180935
## x2 0.00000000  0 0.00000000
## e  0.07180935  0 1.06201033
```

```r
cor(simulated_data)
```

```
## Warning in cor(simulated_data): the standard deviation is zero
```

```
##            x1 x2          e
## x1 1.00000000 NA 0.03512505
## x2         NA  1         NA
## e  0.03512505 NA 1.00000000
```

```r
#scatterplotMatrix(simulated_data)

# Generate outcome variable
outcome_y <- 10 - 6*simulated_data$x1 + 5*simulated_data$x2 + simulated_data$e
par(mfrow=c(1,2))
plot(simulated_data$x1, outcome_y, main="y vs x1", xlab="x1", ylab="y")
plot(simulated_data$x2, outcome_y, main="y vs x2", xlab="x2", ylab="y")
```




```
par(mfrow=c(1,1))

# Run regression
model_constant <- lm(outcome_y ~ x1 + x2, data=simulated_data)
summary(model_constant)
```

```
##
## Call:
## lm(formula = outcome_y ~ x1 + x2, data = simulated_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4402 -0.7074  0.0450  0.7331  3.4532
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.80957    0.16603   149.4   <2e-16 ***
## x1          -5.98175    0.01643  -364.0   <2e-16 ***
## x2                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.03 on 998 degrees of freedom
## Multiple R-squared:  0.9925, Adjusted R-squared:  0.9925
## F-statistic: 1.325e+05 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
# Note: Intercept is 25 = 10 + 5*3 and coefficient is -6
```

3. Violating Assumption 3: Heteroskedasticity

When residual variance is not constant, OLS estimates remain unbiased but are inefficient.

Data Generation

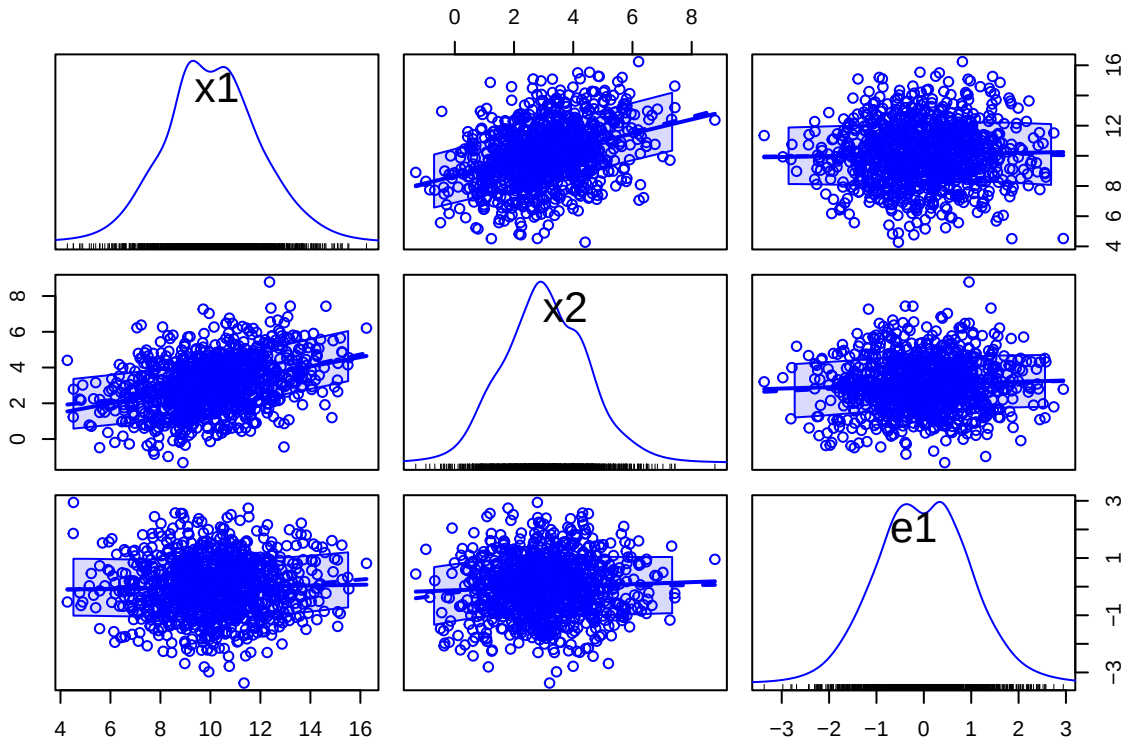
```
# Set seed for reproducibility
set.seed(826)

# Covariance matrix for x1 and x2
covariance_matrix <- matrix(c(4,1,1,2), 2, 2)
covariance_matrix
```

```
##      [,1] [,2]
## [1,]    4    1
## [2,]    1    2
```

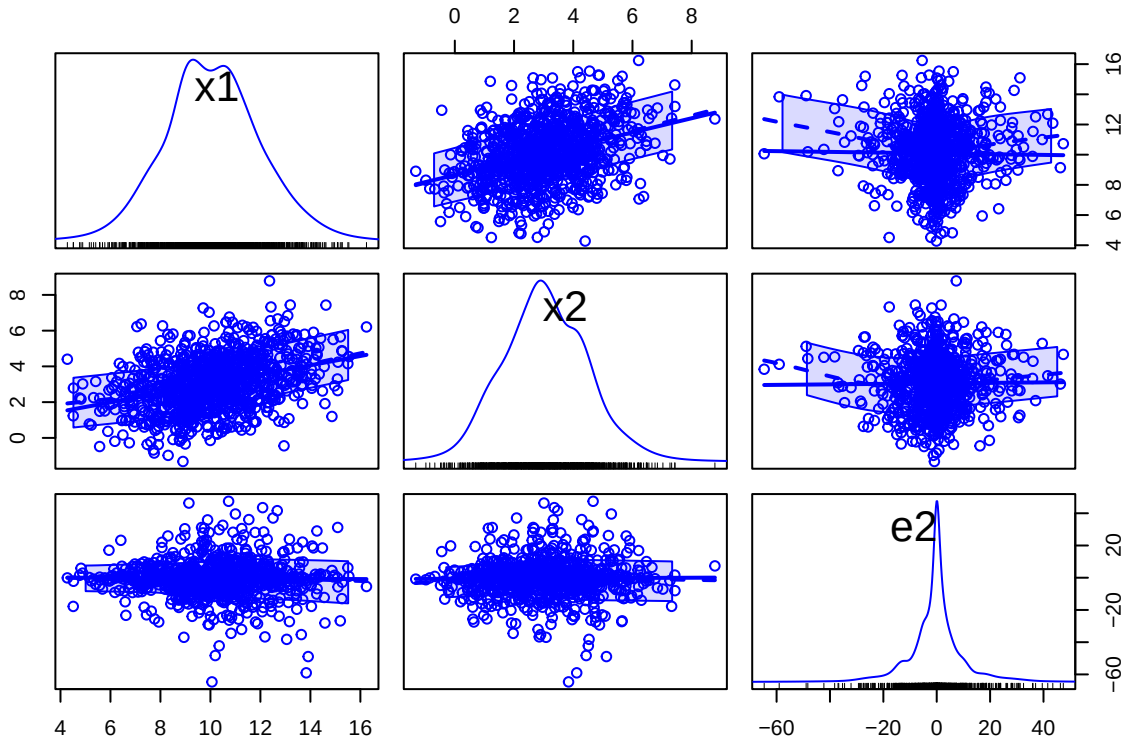
```
# Define mean vector
mean_vector <- c(10, 3)

# Generate data for predictors
predictors <- mvrnorm(n=1000, mu=mean_vector, Sigma=covariance_matrix)

# Generate homoskedastic residuals (constant variance)
homoskedastic_errors <- rnorm(n=1000, mean=0, sd=1)

# Generate heteroskedastic residuals (variance depends on x1 and x2)
variance_function <- (homoskedastic_errors^2) * (predictors[,1]^2 + predictors[,2]^2)
heteroskedastic_errors <- rnorm(n=1000, mean=0, sd=sqrt(variance_function))

# Create data frames
homoskedastic_data <- as.data.frame(cbind(predictors, homoskedastic_errors))
colnames(homoskedastic_data) <- c("x1", "x2", "e1")

heteroskedastic_data <- as.data.frame(cbind(predictors, heteroskedastic_errors))
colnames(heteroskedastic_data) <- c("x1", "x2", "e2")
```

Exploratory Data Analysis

```
# Create scatterplot matrices to visualize relationships
scatterplotMatrix(homoskedastic_data)
```

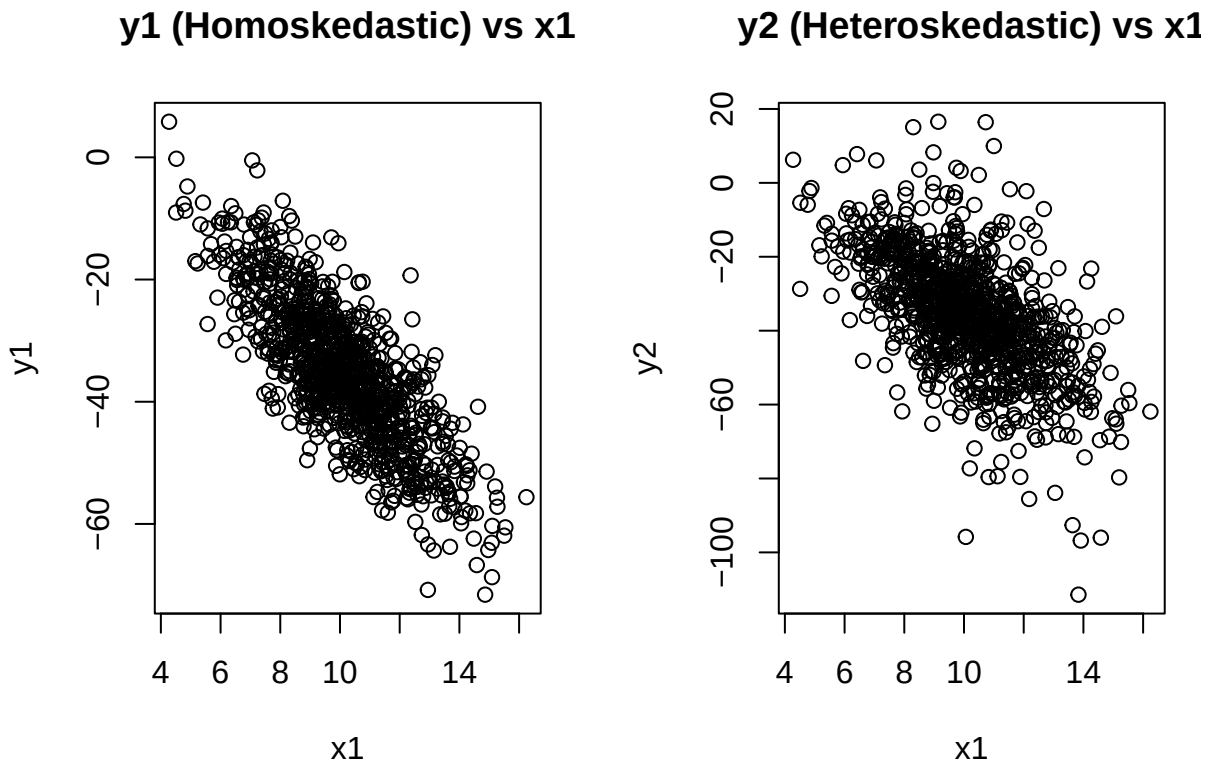



```
scatterplotMatrix(heteroskedastic_data)
```



Generate Outcome Variables

```
# Generate outcome variables
outcome_y1 <- 10 - 6*homoskedastic_data$x1 + 5*homoskedastic_data$x2 + homoskedastic_data$e1
outcome_y2 <- 10 - 6*heteroskedastic_data$x1 + 5*heteroskedastic_data$x2 + heteroskedastic_data$e2

# Plot relationships
par(mfrow=c(1,2))
plot(homoskedastic_data$x1, outcome_y1, main="y1 (Homoskedastic) vs x1",
     xlab="x1", ylab="y1")
plot(heteroskedastic_data$x1, outcome_y2, main="y2 (Heteroskedastic) vs x1",
     xlab="x1", ylab="y2")
```




```
par(mfrow=c(1,1))
```

Model Fitting with Heteroskedasticity

```
# Run linear regression on homoskedastic data
model_homoskedastic <- lm(outcome_y1 ~ x1 + x2, data=homoskedastic_data)
summary(model_homoskedastic)
```

```
##
## Call:
## lm(formula = outcome_y1 ~ x1 + x2, data = homoskedastic_data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3647 -0.6814 -0.0060  0.6925  3.0024
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.82512    0.16589   59.23   <2e-16 ***
## x1          -5.99439    0.01719 -348.62   <2e-16 ***
## x2           5.03285    0.02317  217.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.004 on 997 degrees of freedom
## Multiple R-squared:  0.9925, Adjusted R-squared:  0.9925
## F-statistic: 6.596e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# Run linear regression on heteroskedastic data
model_heteroskedastic <- lm(outcome_y2 ~ x1 + x2, data=heteroskedastic_data)
summary(model_heteroskedastic)
```

```
##
## Call:
## lm(formula = outcome_y2 ~ x1 + x2, data = heteroskedastic_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -64.361  -4.232   0.309   3.882  47.679
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.3175     1.8010   5.729 1.34e-08 ***
## x1           -6.1093     0.1867 -32.726  < 2e-16 ***
## x2            5.1291     0.2515  20.391  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.9 on 997 degrees of freedom
## Multiple R-squared:  0.5383, Adjusted R-squared:  0.5374
## F-statistic: 581.2 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# Note the differences in standard errors and R-squared
```

---

4. Violating Assumption 4: Serial Correlation

When residuals are correlated over time, OLS estimates remain unbiased but are inefficient.

Data Generation

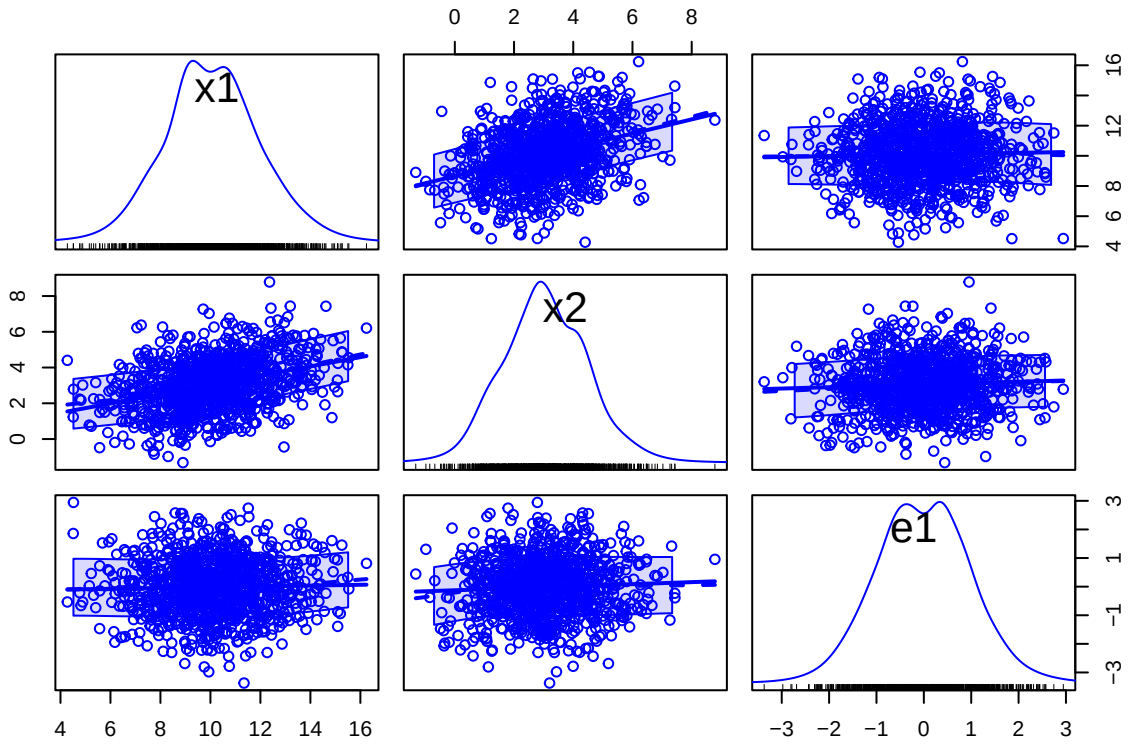
```r
# Set seed for reproducibility
set.seed(826)

# Covariance matrix for x1 and x2
covariance_matrix <- matrix(c(4,1,1,2), 2, 2)
covariance_matrix
```

```
##      [,1] [,2]
## [1,]    4    1
## [2,]    1    2
```

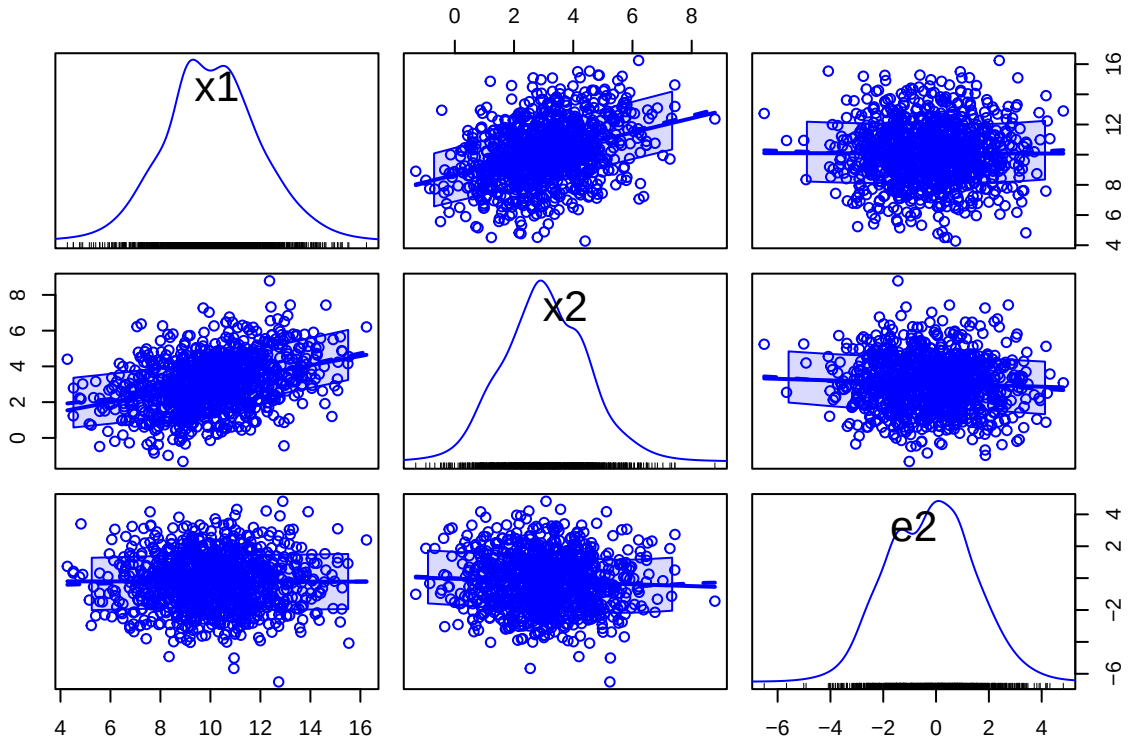
```r
# Define mean vector
mean_vector <- c(10, 3)

# Generate data for predictors
predictors <- mvrnorm(n=1000, mu=mean_vector, Sigma=covariance_matrix)

# Generate independent residuals
independent_errors <- rnorm(n=1000, mean=0, sd=1)

# Generate serially correlated residuals (AR(1) process)
correlated_errors <- arima.sim(model=list(ar=c(0.8)), n=1000, sd=1)

# Create data frames
independent_data <- as.data.frame(cbind(predictors, independent_errors))
colnames(independent_data) <- c("x1", "x2", "e1")

correlated_data <- as.data.frame(cbind(predictors, correlated_errors))
colnames(correlated_data) <- c("x1", "x2", "e2")
```

Exploratory Data Analysis

```r
# Create scatterplot matrices
scatterplotMatrix(independent_data)
```

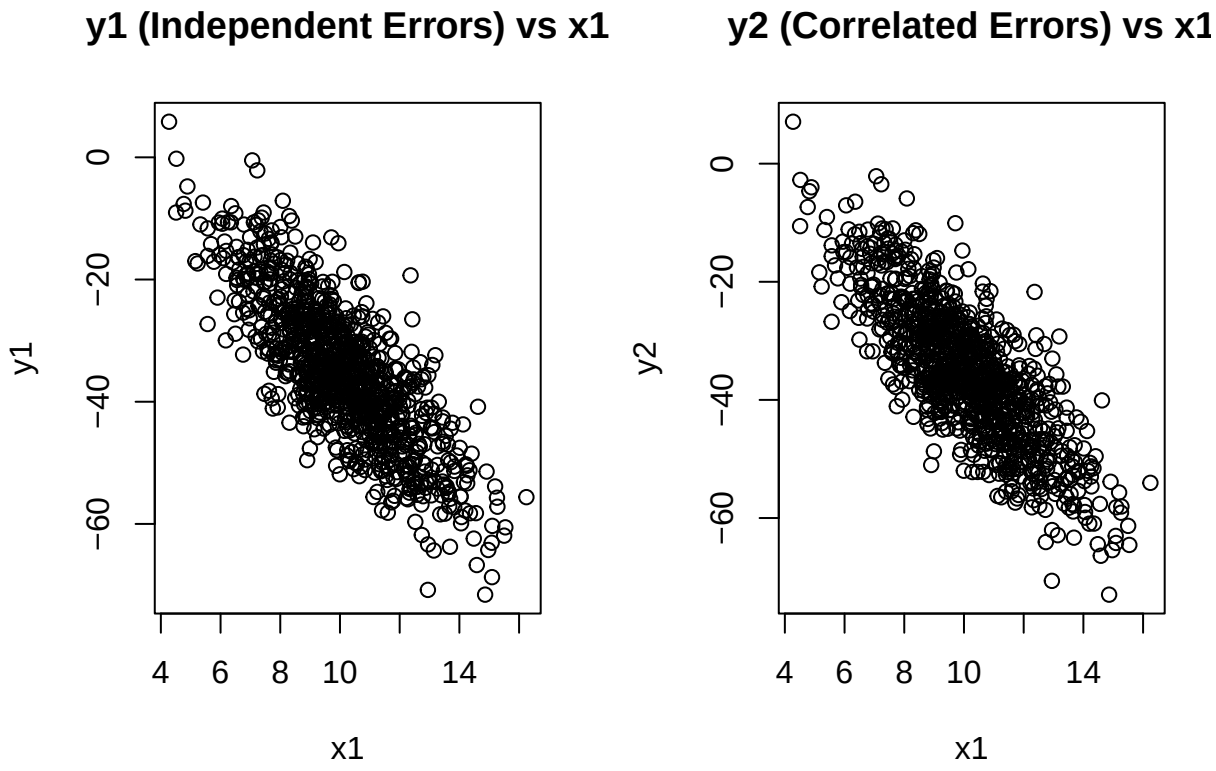



```
scatterplotMatrix(correlated_data)
```



Generate Outcome Variables

```
# Generate outcome variables
outcome_y1 <- 10 - 6*independent_data$x1 + 5*independent_data$x2 + independent_data$e1
outcome_y2 <- 10 - 6*correlated_data$x1 + 5*correlated_data$x2 + correlated_data$e2

# Plot relationships
par(mfrow=c(1,2))
plot(independent_data$x1, outcome_y1, main="y1 (Independent Errors) vs x1",
     xlab="x1", ylab="y1")
plot(correlated_data$x1, outcome_y2, main="y2 (Correlated Errors) vs x1",
     xlab="x1", ylab="y2")
```




```
par(mfrow=c(1,1))
```

Model Fitting with Serial Correlation

```
# Run linear regression on data with independent errors
model_independent <- lm(outcome_y1 ~ x1 + x2, data=independent_data)
summary(model_independent)
```

```
##
## Call:
## lm(formula = outcome_y1 ~ x1 + x2, data = independent_data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3647 -0.6814 -0.0060  0.6925  3.0024
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.82512    0.16589   59.23   <2e-16 ***
## x1          -5.99439    0.01719 -348.62   <2e-16 ***
## x2           5.03285    0.02317  217.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.004 on 997 degrees of freedom
## Multiple R-squared:  0.9925, Adjusted R-squared:  0.9925
## F-statistic: 6.596e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# Run linear regression on data with serially correlated errors
model_correlated <- lm(outcome_y2 ~ x1 + x2, data=correlated_data)
summary(model_correlated)
```

```
##
## Call:
## lm(formula = outcome_y2 ~ x1 + x2, data = correlated_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1938 -1.2369  0.0212  1.1345  4.9921
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.83236    0.27727   35.46   <2e-16 ***
## x1          -5.98476    0.02874 -208.24   <2e-16 ***
## x2           4.93465    0.03872  127.43   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.678 on 997 degrees of freedom
## Multiple R-squared:  0.9791, Adjusted R-squared:  0.9791
## F-statistic: 2.339e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# Note the differences in standard errors and R-squared
```

---

5. Violating Assumption 5: Endogeneity

When X variables are correlated with the error term, OLS estimates are biased.

5.1 Direct Correlation Between X and Error Term

```r
# Set seed for reproducibility
set.seed(826)

# Create a random positive definite covariance matrix
n <- 3
random_matrix <- matrix(runif(n^2)*2-1, ncol=n)
covariance_matrix <- t(random_matrix) %*% random_matrix
covariance_matrix
```

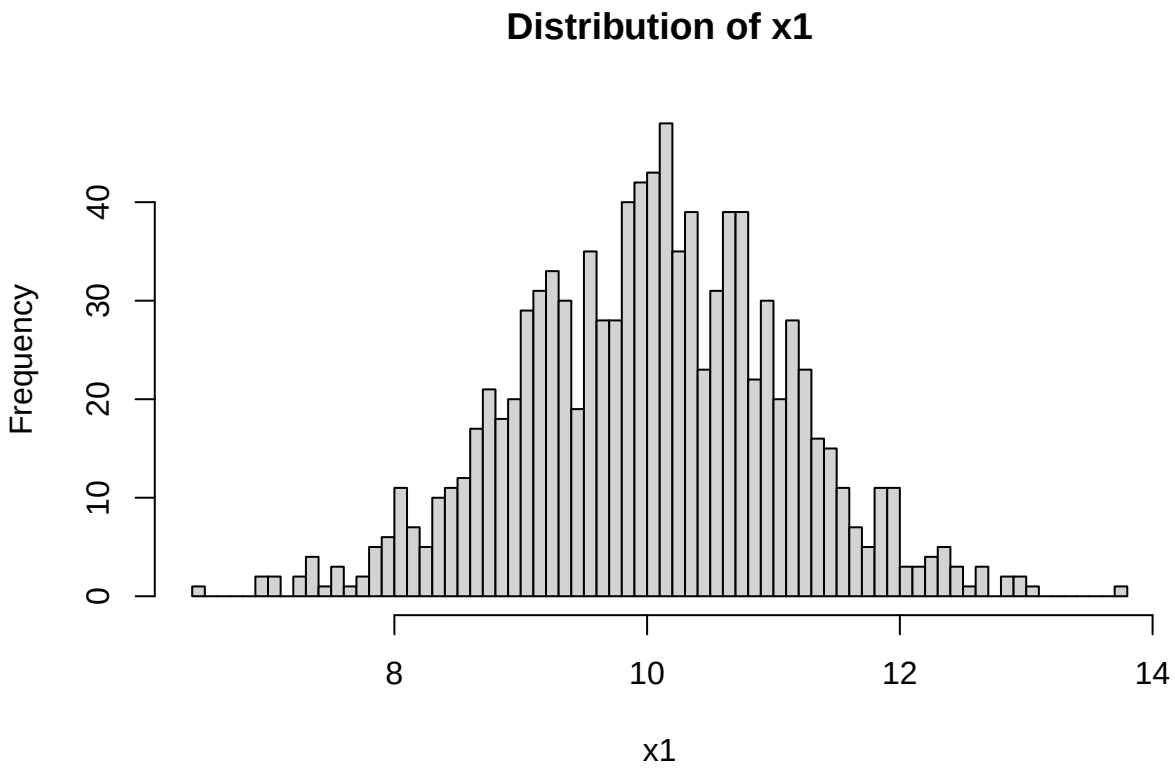
```
##            [,1]       [,2]      [,3]
## [1,]  1.1012487 -0.4083661 0.5785058
## [2,] -0.4083661  1.4481911 0.6653081
## [3,]  0.5785058  0.6653081 1.0935305
```

```r
# Note: Error term (e1) covaries with x1 and x2

# Define mean vector
mean_vector <- c(10, 3, 0)

# Generate data
endogenous_data <- mvrnorm(n=1000, mu=mean_vector, Sigma=covariance_matrix)

# Assign column names
colnames(endogenous_data) <- c("x1", "x2", "e1")
endogenous_data <- as.data.frame(endogenous_data)
head(endogenous_data)
```

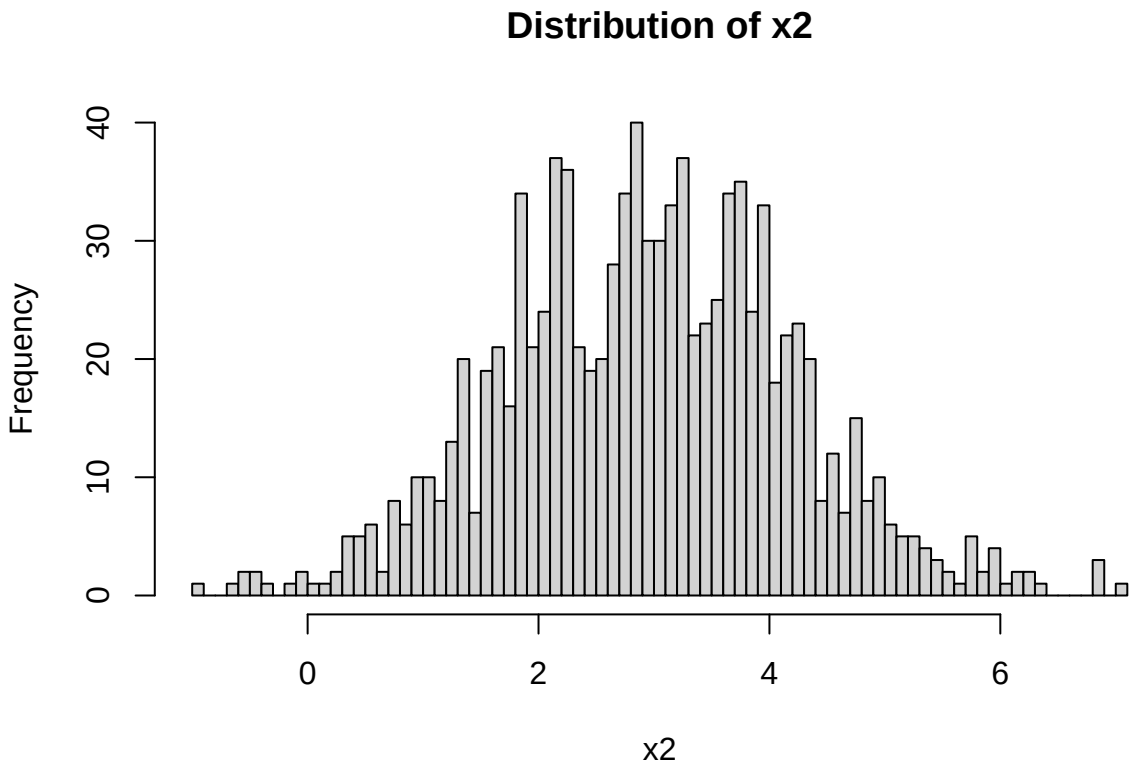
```
##          x1       x2         e1
## 1  7.733325 3.255037 -1.6600797
## 2 10.031196 4.345290  0.6720729
## 3 10.853379 3.439955  0.6454713
## 4 10.285622 3.138029  0.7302525
## 5 11.973566 3.084988  1.9555648
## 6  9.263035 5.055792  0.2941055
```

Exploratory Data Analysis

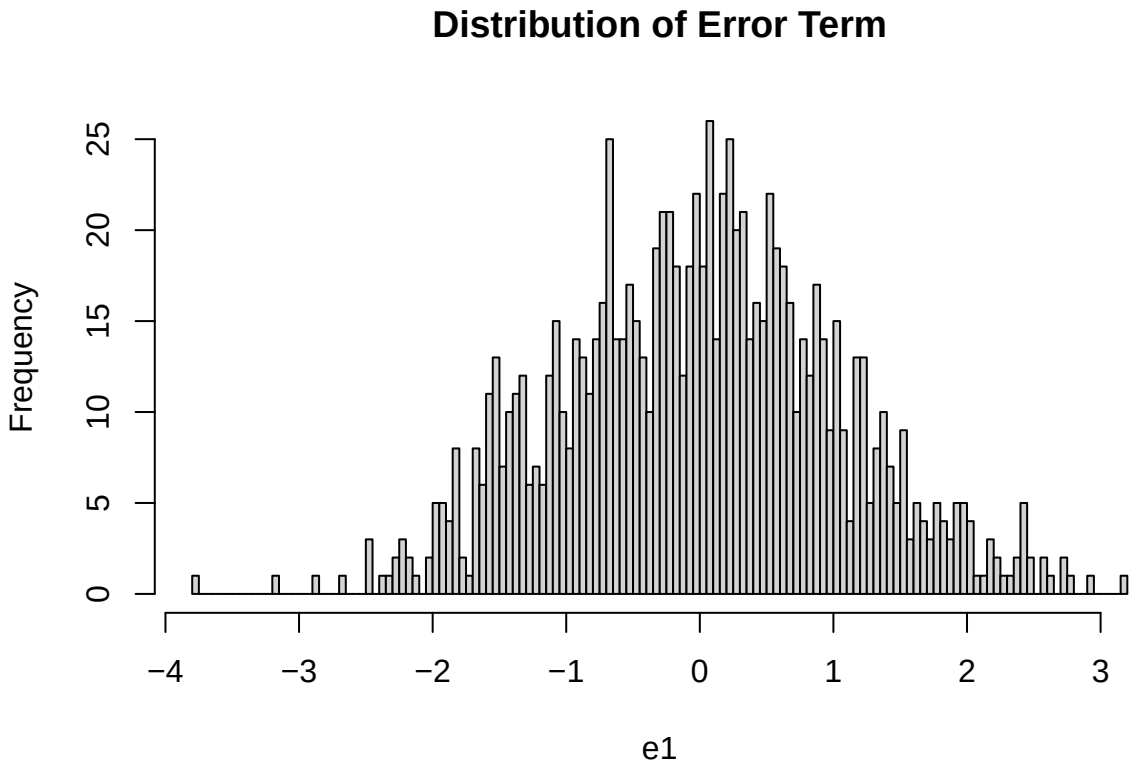
```r
# Plot histograms
hist(endogenous_data$x1, breaks = 100, main="Distribution of x1",
     xlab="x1", ylab="Frequency")
```




```r
hist(endogenous_data$x2, breaks = 100, main="Distribution of x2",
     xlab="x2", ylab="Frequency")
```



```r
hist(endogenous_data$e1, breaks = 100, main="Distribution of Error Term",
     xlab="e1", ylab="Frequency")
```



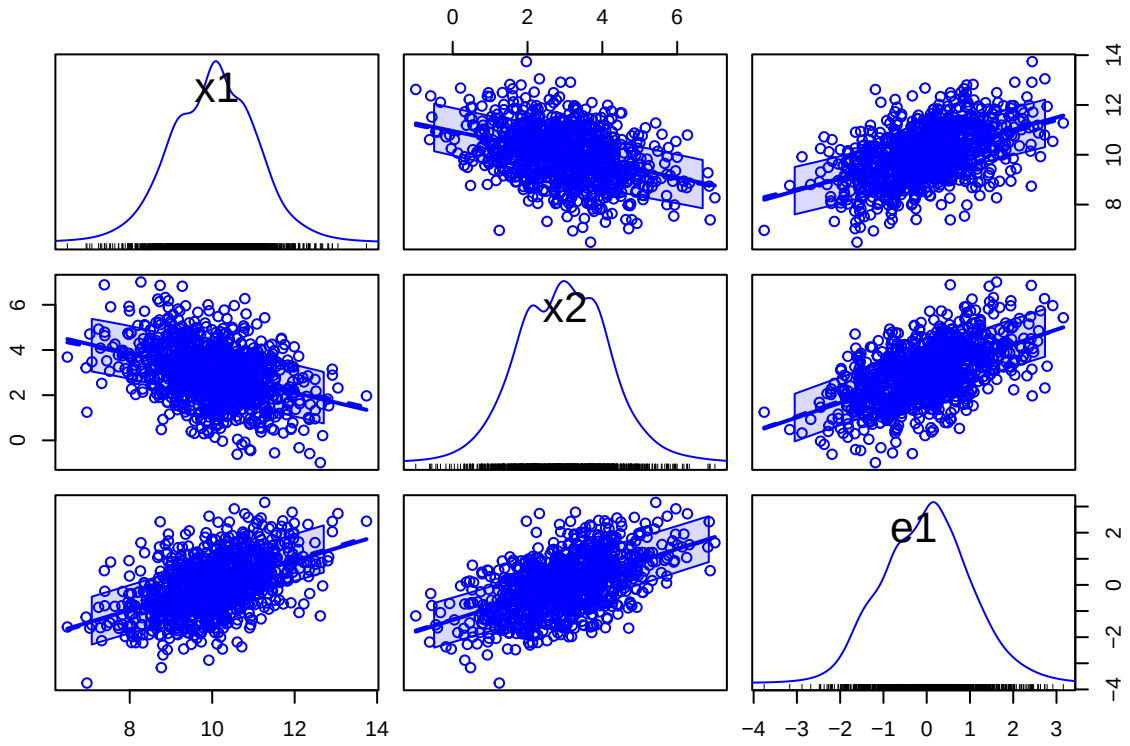

```r
# Calculate and display correlation and covariance matrices
cov(endogenous_data)
```

```
##            x1         x2        e1
## x1  1.1261792 -0.4863156 0.5318302
## x2 -0.4863156  1.5560936 0.7052121
## e1  0.5318302  0.7052121 1.0883354
```

```r
cor(endogenous_data)
```

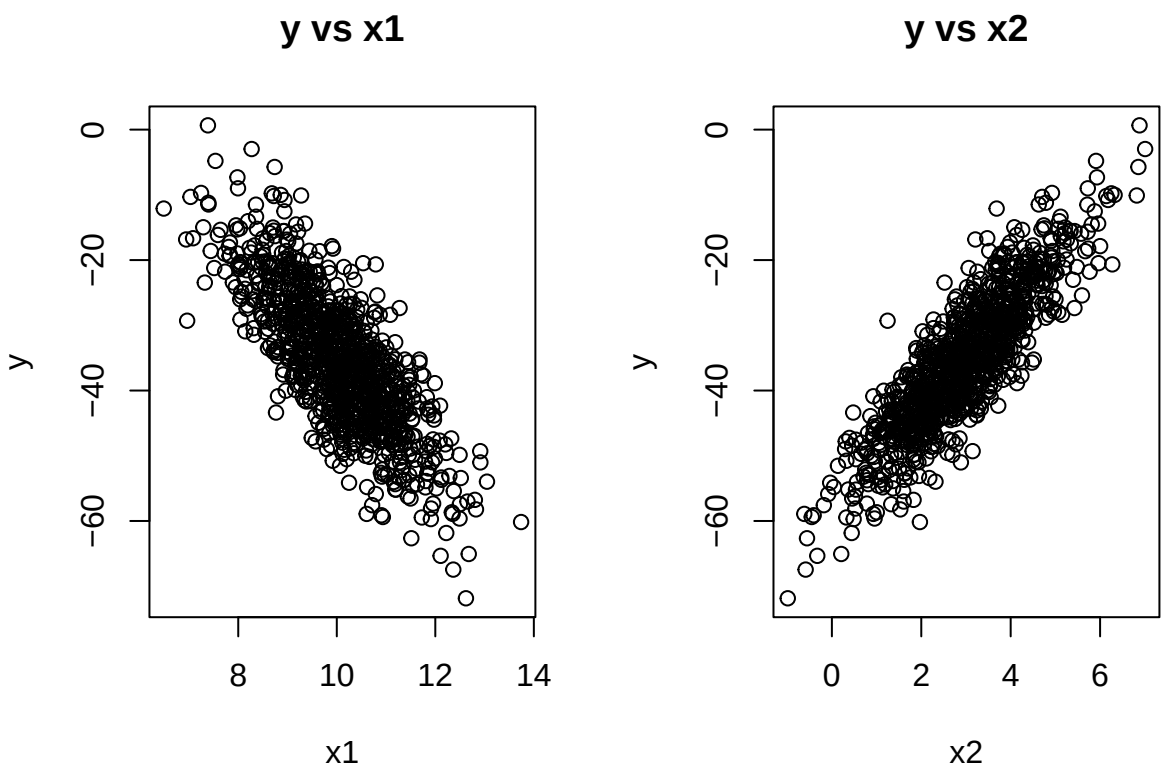
```
##            x1         x2        e1
## x1  1.0000000 -0.3673640 0.4803833
## x2 -0.3673640  1.0000000 0.5419017
## e1  0.4803833  0.5419017 1.0000000
```

```r
# Create scatterplot matrix
scatterplotMatrix(endogenous_data)
```



Generate Outcome Variable

```
# Generate outcome variable
outcome_y1 <- 10 - 6*endogenous_data$x1 + 5*endogenous_data$x2 + endogenous_data$e1

# Plot relationships
par(mfrow=c(1,2))
plot(endogenous_data$x1, outcome_y1, main="y vs x1", xlab="x1", ylab="y")
plot(endogenous_data$x2, outcome_y1, main="y vs x2", xlab="x2", ylab="y")
```




```
par(mfrow=c(1,1))
```

Model Fitting with Endogeneity

```
# Run linear regression with endogenous variables
model_endogenous <- lm(outcome_y1 ~ x1 + x2, data=endogenous_data)
summary(model_endogenous)
```

```
##
## Call:
## lm(formula = outcome_y1 ~ x1 + x2, data = endogenous_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.54018 -0.29630 -0.01474  0.28478  1.25077
##
## Coefficients:
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  0.20152    0.15636    1.289    0.198
## x1          -5.22785    0.01391 -375.858   <2e-16 ***
## x2           5.69451    0.01183  481.250   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4339 on 997 degrees of freedom
## Multiple R-squared:  0.9983, Adjusted R-squared:  0.9983
## F-statistic: 2.923e+05 on 2 and 997 DF,  p-value: < 2.2e-16
```

```r
# Note: Coefficients are biased (different from true values -6 and 5)
```

5.2 Omitted Variable Bias

```r
# Set seed for reproducibility
set.seed(826)

# Covariance matrix
covariance_matrix <- matrix(c(4,1,0,1,2,0,0,0,1), 3, 3)
covariance_matrix
```

```
##      [,1] [,2] [,3]
## [1,]    4    1    0
## [2,]    1    2    0
## [3,]    0    0    1
```

```r
# Note: e does not covary with x1 or x2 (satisfies assumption 5)

# Define mean vector
mean_vector <- c(10, 3, 0)

# Generate data
complete_data <- mvrnorm(n=1000, mu=mean_vector, Sigma=covariance_matrix)

# Assign column names
colnames(complete_data) <- c("x1", "x2", "e")
complete_data <- as.data.frame(complete_data)

# Generate outcome variable
outcome_y2 <- 10 - 6*complete_data$x1 + 5*complete_data$x2 + complete_data$e

# Run correct model with both variables
model_complete <- lm(outcome_y2 ~ x1 + x2, data=complete_data)
summary(model_complete)
```

```
##
## Call:
## lm(formula = outcome_y2 ~ x1 + x2, data = complete_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3647 -0.6814 -0.0060  0.6925  3.0024
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.19399    0.16125   63.22   <2e-16 ***
## x1          -6.02720    0.01665 -362.07   <2e-16 ***
## x2           5.01926    0.02357  212.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.004 on 997 degrees of freedom
## Multiple R-squared:  0.9931, Adjusted R-squared:  0.993
## F-statistic: 7.126e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# Run model with omitted variable x1
model_omit_x1 <- lm(outcome_y2 ~ x2, data=complete_data)
summary(model_omit_x1)
```

```
##
## Call:
## lm(formula = outcome_y2 ~ x2, data = complete_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.793  -7.934  -0.057   7.530  41.335
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -41.7047     0.8497 -49.083   <2e-16 ***
## x2            2.3970     0.2580   9.291   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.55 on 998 degrees of freedom
## Multiple R-squared:  0.07961,    Adjusted R-squared:  0.07869
## F-statistic: 86.32 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
# Run model with omitted variable x2
model_omit_x2 <- lm(outcome_y2 ~ x1, data=complete_data)
summary(model_omit_x2)
```

```
##
## Call:
## lm(formula = outcome_y2 ~ x1, data = complete_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.8197  -4.4152  -0.2228   4.5777  22.8015
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   14.327      1.091   13.13   <2e-16 ***
## x1            -4.938      0.108  -45.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.843 on 998 degrees of freedom
## Multiple R-squared:  0.677,  Adjusted R-squared:  0.6766
## F-statistic:  2091 on 1 and 998 DF,  p-value: < 2.2e-16
```

```r
# Note: Coefficients are biased due to omitted variable bias
```

5.3 Autocorrelated Dependent Variable

```r
# Set seed for reproducibility
set.seed(826)

# Generate an autocorrelated error term (AR(1) process)
ar_errors <- arima.sim(model=list(ar=c(0.8)), n=999, sd=1)

# Generate an autocorrelated outcome variable
ar_outcome <- numeric(1000)
ar_outcome[1] <- rnorm(n=1, mean=10, sd=1)  # Initial value

for(i in 2:1000) {
  ar_outcome[i] <- 10 + 0.4*ar_outcome[i-1] + ar_errors[i-1]
}

# Estimate AR(1) model
arima_model <- arima(ar_outcome, order=c(1,0,0))
arima_model
```

```
##
## Call:
## arima(x = ar_outcome, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.9155    16.3267
## s.e.  0.0131     0.3842
##
## sigma^2 estimated as 1.076:  log likelihood = -1456.61,  aic = 2919.23
```