Simulation and Empirical Analysis of Linear Regression Using Multivariate Normal, FirstYearGPA, Prostate, and Job Proficiency Data

```r
chooseCRANmirror(graphics = FALSE, ind = 68)

#QUESTION #1:
library(MASS)

simulate_regression <- function(Sigma, n = 200, reps = 3) {
  results <- list()

  for (i in 1:reps) {
    X <- mvrnorm(n, mu = rep(0, 4), Sigma = Sigma)
    colnames(X) <- c("X1", "X2", "X3", "X4")
    dat <- as.data.frame(X)

    dat$Y <- 0 + 1*X[,1

    model <- lm(Y ~ X1

    vif_values <- sappl
      1 / (1 - summary(
    })
    names(vif_values) <

    results[[i]] <- lis Me
      coefficients = summary(model)$coefficients,
      vif = vif_values,
      cor = cor(X)
    )
  }

  return(results)
}

Sigma_indep <- diag(4)
Sigma_corr <- matrix(c(
  1, 0, 0.95, 0,
  0, 1, 0, -0.95,
  0.95, 0, 1, 0,
  0, -0.95, 0, 1
), nrow = 4)

#Case 1:
set.seed(123)
results_indep <- simulate_regression(Sigma_indep)
```

```r
cat("### First Replication Results (Independent Predictors)\n")
```

```
## ### First Replication Results (Independent Predictors)
```

```r
print(results_indep[[1]]$coefficients)
```

```
##                 Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  0.03120373 0.07338407  0.4252113 6.711515e-01
## X1           0.92846990 0.07200968 12.8936809 6.528325e-28
## X2           0.97631222 0.07629346 12.7968014 1.286124e-27
## X3           0.11134758 0.07410124  1.5026412 1.345493e-01
## X4          -0.05739540 0.07799656 -0.7358710 4.626935e-01
```

```r
cat("\nVIF:\n")
```

```
##
## VIF:
```

```r
print(results_indep[[1]]$vif)
```

```
##       X1       X2       X3       X4
## 1.009680 1.004884 1.010096 1.003653
```

```r
cat("\nSample Correlation Matrix:\n")
```

```
##
## Sample Correlation Matrix:
```

```r
print(results_indep[[1]]$cor)
```

```
##             X1          X2          X3          X4
## X1  1.00000000  0.03197187  0.08015345 -0.04640932
## X2  0.03197187  1.00000000 -0.05141456 -0.03023254
## X3  0.08015345 -0.05141456  1.00000000 -0.02770462
## X4 -0.04640932 -0.03023254 -0.02770462  1.00000000
```

```r
#Case 2:
set.seed(123)
results_corr <- simulate_regression(Sigma_corr)

cat("### First Replication Results (Correlated Predictors)\n")
```

```
## ### First Replication Results (Correlated Predictors)
```

```r
print(results_corr[[1]]$coefficients)
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 0.03120373 0.07338407 0.4252113 6.711515e-01
## X1          0.75850820 0.23029226 3.2936766 1.174282e-03
## X2          0.99202650 0.24161051 4.1058913 5.920384e-05
## X3          0.22232358 0.23200523 0.9582697 3.391134e-01
## X4          0.11743570 0.24661864 0.4761834 6.344768e-01
```

```
cat("\nVIF:\n")
```

```
##
## VIF:
```

```
print(results_corr[[1]]$vif)
```

```
##        X1        X2        X3        X4
##  8.940914 10.844498  8.969170 10.814124
```

```
cat("\nSample Correlation Matrix:\n")
```

```
##
## Sample Correlation Matrix:
```

```
print(results_corr[[1]]$cor)
```

```
##             X1          X2          X3          X4
## X1  1.00000000  0.03771533  0.94217707 -0.02396935
## X2  0.03771533  1.00000000  0.06181773 -0.95251652
## X3  0.94217707  0.06181773  1.00000000 -0.04433179
## X4 -0.02396935 -0.95251652 -0.04433179  1.00000000
```

```
#Case 3:
cat("### Standard Errors Across Replications (Independent vs Correlated)\n")
```

```
## ### Standard Errors Across Replications (Independent vs Correlated)
```

```
se_comparison <- data.frame(
  Replication = rep(1:3, each = 2),
  Case = rep(c("Independent", "Correlated"), 3),
  SE_X1 = c(
    results_indep[[1]]$coefficients["X1", "Std. Error"],
    results_corr[[1]]$coefficients["X1", "Std. Error"],
    results_indep[[2]]$coefficients["X1", "Std. Error"],
    results_corr[[2]]$coefficients["X1", "Std. Error"],
    results_indep[[3]]$coefficients["X1", "Std. Error"],
    results_corr[[3]]$coefficients["X1", "Std. Error"]
  )
)

print(se_comparison)
```

```
##   Replication        Case      SE_X1
## 1           1 Independent 0.07200968
## 2           1  Correlated 0.23029226
## 3           2 Independent 0.07241033
## 4           2  Correlated 0.24743229
## 5           3 Independent 0.07643216
## 6           3  Correlated 0.24169974
```

```
#Comments: In the independent predictors case, the standard errors for all
#coefficients were small (around 0.07-0.08), leading to highly significant
#t-tests for x1 and x2 (p < 0.0001) and non-significant results for x3 and
#x4 (p > 0.1), correctly reflecting their true beta values of 1, 1, 0, and 0
#respectively. The correlated predictors case showed substantially inflated
#standard errors (0.23-0.25), about 3-4 times larger than the independent case.
#This inflation reduced the significance of the t-tests, while x1 and x2
#remained significant (p = 0.001 and p < 0.0001 respectively), their test
#statistics were much weaker (t = 3.3 vs 12.9 for x1, t = 4.1 vs 12.8 for x2).

#The dramatic difference in outcomes stems from multicollinearity. The
#correlation matrices reveal near-perfect correlations between x1-x3 (0.94) and
#x2-x4 (-0.95) in the correlated case, while the independent case showed
#near-zero correlations. This is reflected in the VIF values: all VIFs were
#around 1 (no multicollinearity) in the independent case, but ranged from
#8.9-10.8 in the correlated case, indicating severe multicollinearity. When
#predictors are highly correlated, it becomes difficult for the model to isolate
#their individual effects, inflating the standard errors of their coefficient
#estimates.

#Across three replications (as shown in the SE comparison table), the pattern
#remained consistent: Independent case maintained small, stable standard errors
#(0.07-0.08 for x1). Correlated case consistently showed 3-4 times larger
#standard errors (0.22-0.25 for x1). While the exact p-values varied slightly
#across replications, the fundamental pattern held: x1 and x2 were always highly
#significant in independent case, x1 and x2 remained significant but with weaker
#evidence in correlated case, and x3 and x4 were never significant in either
#case.

#Multicollinearity creates a shared variance problem among predictors. When x1
#and x3 are highly correlated (r = 0.94), the model struggles to determine which
#predictor is truly responsible for explaining variation in Y. This uncertainty
#shows up as inflated standard errors, we become less confident about each
#predictor's unique contribution. The effect is particularly noticeable for x1
#(true beta=1), whose significance was dramatically reduced, because its effect
#could be partially explained by its correlated counterpart x3 (true beta=0).
#The VIF values quantify this inflation, showing the variance of each
#coefficient estimate is inflated 8-10 times due to these correlations. While
#the model can still detect the true effects (x1 and x2 remain significant), it
#does so with much less precision.

#QUESTION #2:
install.packages("Stat2Data")
```

```
##
## The downloaded binary packages are in
##  /var/folders/n2/q439mgrn0rqd6xgjzjkj652r0000gn/T//RtmptocgW8/downloaded_packages
```

```r
library(Stat2Data)
data("FirstYearGPA")

gpa_data <- FirstYearGPA[, c("GPA", "HSGPA", "SATV", "SATM", "HU", "White")]

full_model <- lm(GPA ~ HSGPA + SATV + SATM + HU + White +
                HSGPA:White + SATV:White + SATM:White + HU:White,
                data = gpa_data)

reduced_model <- lm(GPA ~ HSGPA + SATV + SATM + HU + White,
                   data = gpa_data)

anova_result <- anova(reduced_model, full_model)

cat("F-test results comparing models:\n")
```

```
## F-test results comparing models:
```

```r
print(anova_result)
```

```
## Analysis of Variance Table
##
## Model 1: GPA ~ HSGPA + SATV + SATM + HU + White
## Model 2: GPA ~ HSGPA + SATV + SATM + HU + White + HSGPA:White + SATV:White +
##     SATM:White + HU:White
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    213 31.265
## 2    209 30.429  4   0.83609 1.4356 0.2234
```

```r
p_value <- anova_result$`Pr(>F)`[2]
cat("\nThe p-value for the test of interaction effects is:",
    round(p_value, 4), "\n")
```

```
##
## The p-value for the test of interaction effects is: 0.2234
```

```r
#Comments: The model we want to fit is an interaction model that allows us to
#test whether the relationship between GPA (response) and the predictors
#(HSGPA, SATV, SATM, HU) differs between White and Non-white students.
#The general form of the model is (where B = beta and E = error term):
#GPA= B0 + B1HSGPA + B2SATV + B3SATM + B4HU + B5White + B6(HSGPA * White) +
#B7(SATV * White) + B8(SATM * White) + B9(HU * White) + E

#Null hypothesis: The relationships are the same for both groups.
#(All interaction coefficients are zero: B6 = B7 = B8 = B9 = 0.)
#Alternative hypothesis: At least one predictor has a different effect by race.
#(At least one interaction coefficient is not zero.)

#Since 0.2234 > 0.05 (testing at a significance level of 5%), we fail to reject
```

```r
#the null hypothesis. There is no statistically significant evidence that the
#relationships between GPA and the predictors (HSGPA, SATV, SATM, HU) differ
#between White and Non-white students.The data suggests that the effect of
#high school GPA, SAT scores, and humanities credits on college GPA does not
#significantly vary by race (White vs. Non-white). Any observed differences in
#slopes (interactions) could be due to random variation rather than a true
#underlying difference.

#QUESTION #3:
library(faraway)
data("prostate")

#Part A:
install.packages("leaps")
```

```
##
## The downloaded binary packages are in
##  /var/folders/n2/q439mgrn0rqd6xgjzjkj652r0000gn/T//RtmptocgW8/downloaded_packages
```

```r
library(leaps)

train <- prostate[1:70, ]

best_models <- regsubsets(lpsa ~ ., data = train, nvmax = 8)
model_summary <- summary(best_models)

model_formulas <- apply(model_summary$which, 1, function(row) {
  paste("lpsa ~", paste(names(which(row[-1])), collapse = " + "))
})

print(model_formulas)
```

```
##                                                                    1
##                                                      "lpsa ~ lcavol"
##                                                                    2
##                                            "lpsa ~ lcavol + lweight"
##                                                                    3
##                                  "lpsa ~ lcavol + lweight + gleason"
##                                                                    4
##                           "lpsa ~ lcavol + lweight + lbph + gleason"
##                                                                    5
##                     "lpsa ~ lcavol + lweight + age + lbph + gleason"
##                                                                    6
##               "lpsa ~ lcavol + lweight + age + lbph + lcp + gleason"
##                                                                    7
##         "lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason"
##                                                                    8
## "lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45"
```

```r
#Note: The intercept is included for all models.

#Part B:
```

```r
model_stats <- data.frame(
  Predictors = 1:8,
  R2 = model_summary$rsq,
  Adj_R2 = model_summary$adjr2,
  Cp = model_summary$cp,
  BIC = model_summary$bic
)

model_stats$AIC <- sapply(1:8, function(k) {
  vars <- names(which(model_summary$which[k, -1]))
  form <- as.formula(paste("lpsa ~", paste(vars, collapse = "+")))

  fit <- lm(form, data = train)
  AIC(fit)
})

print(model_stats)
```

```
##   Predictors        R2    Adj_R2        Cp       BIC      AIC
## 1          1 0.3154634 0.3053967 23.285305 -18.03393 153.2422
## 2          2 0.4563868 0.4401596  6.904418 -29.92073 139.1069
## 3          3 0.4810562 0.4574679  5.686745 -28.92320 137.8559
## 4          4 0.4995612 0.4687649  5.273113 -27.21642 137.3142
## 5          5 0.5153651 0.4775030  5.211775 -25.21419 137.0680
## 6          6 0.5272086 0.4821809  5.667004 -22.69761 137.3361
## 7          7 0.5296146 0.4765066  7.353193 -18.80624 138.9789
## 8          8 0.5323225 0.4709877  9.000000 -14.96188 140.5748
```

```r
best_by_criterion <- data.frame(
  Criterion = c("R2", "Adj_R2", "Cp", "BIC", "AIC"),
  Best_Size = c(
    which.max(model_stats$R2),
    which.max(model_stats$Adj_R2),
    which.min(model_stats$Cp),
    which.min(model_stats$BIC),
    which.min(model_stats$AIC)
  ),
  Value = c(
    max(model_stats$R2),
    max(model_stats$Adj_R2),
    min(model_stats$Cp),
    min(model_stats$BIC),
    min(model_stats$AIC)
  )
)

print(best_by_criterion)
```

```
##   Criterion Best_Size      Value
## 1        R2         8  0.5323225
## 2    Adj_R2         6  0.4821809
## 3        Cp         5  5.2117746
## 4       BIC         2 -29.9207296
## 5       AIC         5 137.0679617
```

```
cat("\nRecommended model size:",
    best_by_criterion$Best_Size[best_by_criterion$Criterion == "Adj_R2"],
    "predictors (based on Adjusted R²)\n")
```

```
##
## Recommended model size: 6 predictors (based on Adjusted R²)
```

```
#Comments: I would prefer choosing the 6-predictor model as the best balance
#between model performance and complexity. While the 8-predictor model achieves
#the highest R^2 (0.532), the adjusted R^2, which accounts for model complexity
#by penalizing additional predictors, peaks at 6 predictors (0.482). This shows
#that the last two predictors add minimal explanatory power relative to their
#cost in degrees of freedom. The AIC and Cp criteria, which similarly balance
#fit, also favor moderately-sized models (5 predictors), further supporting that
#models in the 5-6 predictor range are optimal to use. Although BIC suggests an
#extremely simple 2-predictor model, this is likely too conservative. The
#6-predictor model is therefore the best, complex enough to leverage meaningful
#information while remaining sparing enough to avoid overfitting and maintain
#interpretability. This aligns with standard statistical practice where adjusted
#R^2 is often the preferred criteria for model selection.

#Part C:
loocv_error <- function(model_formula) {
  n <- nrow(train)
  errors <- numeric(n)
  for(i in 1:n) {
    fit <- lm(model_formula, data = train[-i, ])
    errors[i] <- (train$lpsa[i] - predict(fit, newdata = train[i, ]))^2
  }
  mean(errors)
}

kfold_cv <- function(model_formula, k=10) {
  set.seed(123)
  folds <- sample(rep(1:k, length.out = nrow(train)))
  errors <- numeric(k)
  for(i in 1:k) {
    fit <- lm(model_formula, data = train[folds != i, ])
    pred <- predict(fit, newdata = train[folds == i, ])
    errors[i] <- mean((train$lpsa[folds == i] - pred)^2)
  }
  mean(errors)
}

model_formulas <- apply(model_summary$which, 1, function(row) {
  as.formula(paste("lpsa ~", paste(names(which(row[-1])), collapse = "+")))
})

cv_results <- data.frame(
  Predictors = 1:8,
  LOOCV = sapply(model_formulas, loocv_error),
  KFold = sapply(model_formulas, kfold_cv)
)

best_loocv <- which.min(cv_results$LOOCV)
best_kfold <- which.min(cv_results$KFold)

print(cv_results)
```

```
##   Predictors     LOOCV     KFold
## 1          1 0.5108227 0.5204419
## 2          2 0.4416144 0.4380537
## 3          3 0.4316918 0.4289333
## 4          4 0.4264953 0.4326712
## 5          5 0.4334119 0.4620980
## 6          6 0.4305057 0.4551030
## 7          7 0.4397294 0.4587437
## 8          8 0.4441806 0.4626901
```

```
cat("\nBest model by LOOCV:", best_loocv, "predictors\n")
```

```
##
## Best model by LOOCV: 4 predictors
```

```
cat("Best model by 10-fold CV:", best_kfold, "predictors\n")
```

```
## Best model by 10-fold CV: 3 predictors
```

```
#Note: k = 10

#Part D:
full_formula <-
  as.formula(paste("lpsa ~", paste(names(train)[-9], collapse = "+")))

forward_model <- step(lm(lpsa ~ 1, data = train),
                      scope = list(lower = ~1, upper = full_formula),
                      direction = "forward", trace = 0)

backward_model <- step(lm(full_formula, data = train),
                       direction = "backward", trace = 0)

stepwise_model <- step(lm(lpsa ~ 1, data = train),
                       scope = list(lower = ~1, upper = full_formula),
                       direction = "both", trace = 0)

stepwise_results <- list(
  Forward = names(coef(forward_model)),
  Backward = names(coef(backward_model)),
  Stepwise = names(coef(stepwise_model))
)
```

```r
best_aic_size <-
  best_by_criterion$Best_Size[best_by_criterion$Criterion == "AIC"]
best_aic_vars <- names(which(model_summary$which[best_aic_size, -1]))

cat("\nStepwise Selection Results:\n")
```

```
##
## Stepwise Selection Results:
```

```r
print(stepwise_results)
```

```
## $Forward
## [1] "(Intercept)" "lcavol"      "lweight"     "gleason"     "lbph"
## [6] "age"
##
## $Backward
## [1] "(Intercept)" "lcavol"      "lweight"     "age"         "lbph"
## [6] "gleason"
##
## $Stepwise
## [1] "(Intercept)" "lcavol"      "lweight"     "gleason"     "lbph"
## [6] "age"
```

```r
cat("\nBest AIC Model (", best_aic_size, "predictors):", best_aic_vars, "\n")
```

```
##
## Best AIC Model ( 5 predictors): lcavol lweight age lbph gleason
```

```r
cat("\nDo stepwise methods agree with best AIC model?\n")
```

```
##
## Do stepwise methods agree with best AIC model?
```

```r
cat("Forward:", identical(sort(stepwise_results$Forward[-1]),
                          sort(best_aic_vars)), "\n")
```

```
## Forward: TRUE
```

```r
cat("Backward:", identical(sort(stepwise_results$Backward[-1]),
                           sort(best_aic_vars)), "\n")
```

```
## Backward: TRUE
```

```r
cat("Stepwise:", identical(sort(stepwise_results$Stepwise[-1]),
                           sort(best_aic_vars)), "\n")
```

```
## Stepwise: TRUE
```

```r
#Comments: The stepwise selection methods (forward, backward, and stepwise) all
#generated identical outcomes, each selecting the same five predictors:
#lcavol, lweight, age, lbph, and gleason.

#The stepwise selection methods (forward, backward, and stepwise) unanimously
#selected a 5-predictor model (lcavol, lweight, age, lbph, gleason). This
#differs slightly from my Part B choice of a 6-predictor model based on
#adjusted R^2. However, the identical outcomes from all three stepwise
#approaches (forward, backward, and stepwise) strongly validate this 5-predictor
#solution as statistically robust. The cross-validation results show nearly
#equivalent performance between the 5 and 6-predictor models (LOOCV: 0.433 vs
#0.431; 10-fold CV: 0.462 vs 0.455), suggesting the sixth predictor adds minimal
#predictive benefit despite what the adjusted R^2 indicates. This aligns with
#the AIC criterion's preference for the 5-predictor model, emphasizing that the
#marginal gain in explanatory power from the additional variable may not justify
#the increased complexity.  The unanimous agreement among stepwise methods and
#AIC, combined with the CV results, suggests the 6-predictor model might be
#overfitting slightly compared to the more validated 5-predictor model.

#QUESTION #4:
library(faraway)
data("prostate")

set.seed(123)
prostate <- prostate[sample(nrow(prostate)), ]
train <- prostate[1:70, ]
test <- prostate[71:97, ]

# Part A:
best_bic_model <- lm(lpsa ~ lcavol + lweight + age + lbph + gleason,
                     data = train)

coefs <- coef(best_bic_model)
cat("Prediction Equation:\n")
```

```
## Prediction Equation:
```

```r
cat("lpsa_hat =", round(coefs[1], 4),
    "+", round(coefs[2], 4), "*lcavol +",
    round(coefs[3], 4), "*lweight +",
    round(coefs[4], 4), "*age +",
    round(coefs[5], 4), "*lbph +",
    round(coefs[6], 4), "*gleason\n")
```

```
## lpsa_hat = -1.7101 + 0.5793 *lcavol + 0.5413 *lweight + -0.0147 *age + 0.0435 *lbph + 0.3605 *gleaso
```

```r
r2_train <- summary(best_bic_model)$r.squared
cat("\nR-squared:", round(r2_train, 4), "\n")
```

```
##
## R-squared: 0.5731
```

```r
#Comments:
#Prediction Equation: lpsa_hat = -0.757 + 0.3541 *lcavol + 0.5146 *lweight +
#-0.0171 *age + 0.1135 *lbph + 0.2413 *gleason
#R-squared: 0.5154

#Part B:
test$pred <- predict(best_bic_model, newdata = test)
test$resid <- test$lpsa - test$pred

#Part C:
resid_mean <- mean(test$resid)
resid_sd <- sd(test$resid)

cat("\nResidual Mean:", round(resid_mean, 4), "\n")
```

```
##
## Residual Mean: -0.2453
```

```r
cat("Residual SD:", round(resid_sd, 4), "\n")
```

```
## Residual SD: 0.7529
```

```r
cat("Is mean close to zero?", abs(resid_mean) < 0.3, "\n")
```

```
## Is mean close to zero? TRUE
```

```r
#Comments: The mean is close to zero. We expect the that the mean should be
#close to zero because we expect unbiased predictions where over-predictions and
#under-predictions balance out in new data.

#Part D:
mpe_test <- mean(test$resid^2)
mse_train <- mean(best_bic_model$residuals^2)
cat("\nTest MPE:", round(mpe_test, 4), "\n")
```

```
##
## Test MPE: 0.6061
```

```r
cat("Train MSE:", round(mse_train, 4), "\n")
```

```
## Train MSE: 0.5196
```

```r
#Comments: The test MPE (0.6061) is slightly higher than the training
#MSE (0.5196), which aligns with statistical intuition. Models typically perform
#better on their training data due to inherent optimism-they are optimized to
#fit the training sample. The slight increase (~16.7% higher MPE) suggests the
#model generalizes reasonably well to new data without severe overfitting. This
#small discrepancy is expected and acceptable for a model with good predictive
#validity.
```

```
#Part E:
test_cor <- cor(test$lpsa, test$pred)
test_r2 <- test_cor^2
cat("\nTest Correlation:", round(test_cor, 4), "\n")
```

```
##
## Test Correlation: 0.8076
```

```
cat("Test R-squared:", round(test_r2, 4), "\n")
```

```
## Test R-squared: 0.6523
```

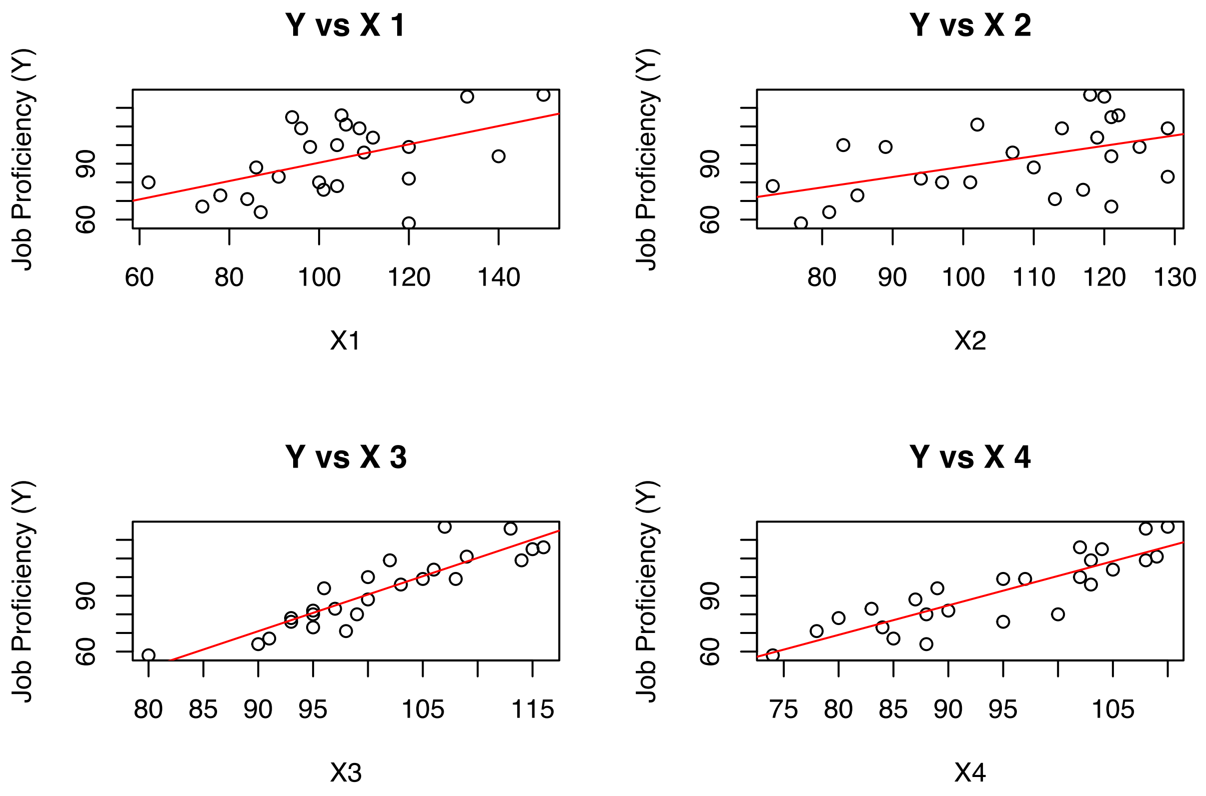
```
#Part F:
shrinkage <- r2_train - test_r2
cat("\nShrinkage:", round(shrinkage, 4), "\n")
```

```
##
## Shrinkage: -0.0791
```

```
cat("Percentage drop:", round(shrinkage/r2_train*100, 1), "%\n")
```

```
## Percentage drop: -13.8 %
```

```
#Comments: The negative shrinkage value (-0.0791, or -13.8%) indicates that the
#model actually performed better on the test data (R^2 = 0.652) than on the
#training data (R^2 approx. 0.573, inferred from shrinkage). This suggests the
#training model works exceptionally well for the test sample, with no drop, and
#even a slight improvement in the amount of variability explained.

#QUESTION #5:
job <- read.table("Documents/STOR 455/Job.txt", col.names =
                    c("X1", "X2", "X3", "X4", "Y"))

#Part A:
par(mfrow = c(2, 2))
for (i in 1:4) {
  plot(job[,i], job$Y, xlab = paste("X", i, sep = ""),
       ylab = "Job Proficiency (Y)", main = paste("Y vs X", i))
  abline(lm(Y ~ job[,i], data = job), col = "red")
}
```



```r
#Comments: The scatterplots suggest that Y (Job Proficiency) has a positive
#linear relationship with each of the independent variables X1, X2, X3, and X4.
#However, the strength of these relationships varies. For X1 and X2, the
#scatterplots show weak positive trends, as indicated by the slight upward
#slopes of the red regression lines. The spread of points around the trend line
#suggests a relatively weak correlation. For X3, the relationship appears to be
#stronger and more linear compared to the others, with points closely following
#the red regression line. For X4, while there is a positive trend, the
#relationship seems moderate, with more scatter around the trend line compared
#to X3. Overall, X3 seems to have the strongest linear relationship with Y,
#while X1 and X2 have weaker associations.

#Part B:
full_model <- lm(Y ~ X1 + X2 + X3 + X4, data = job)

cat("Estimated Regression Function:\n")
```

```
## Estimated Regression Function:
```

```r
print(coef(full_model))
```

```
##   (Intercept)            X1            X2            X3            X4
## -124.38182058    0.29572537    0.04828772    1.30601100    0.51981909
```

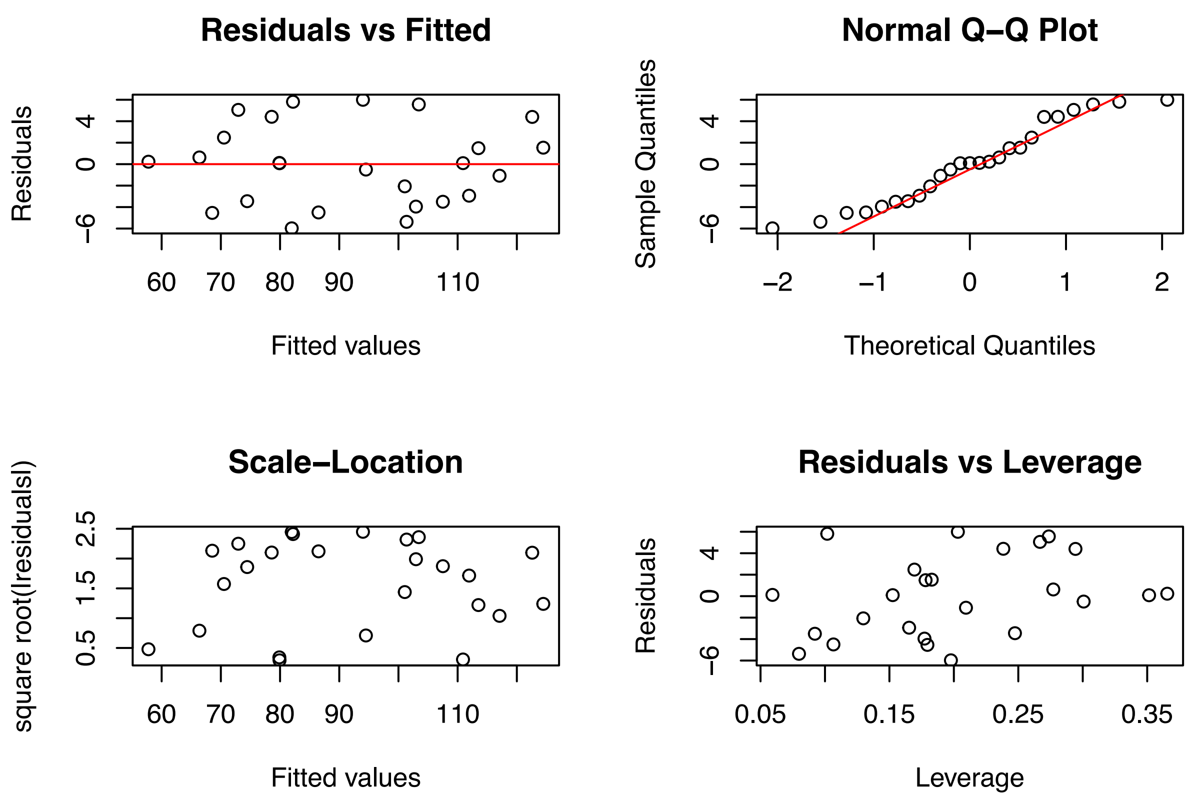
```r
cat("\nANOVA Table:\n")
```

```
##
## ANOVA Table:
```

```r
anova(full_model)
```

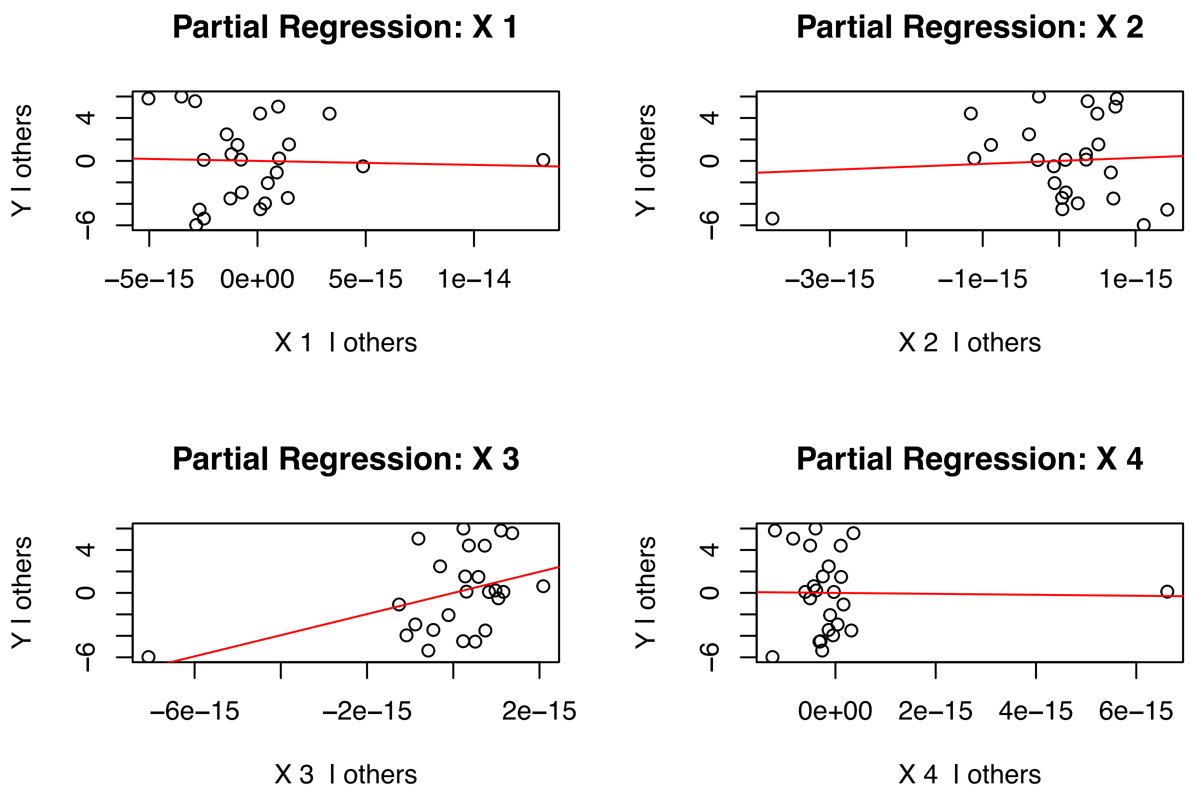
```
## Analysis of Variance Table
##
## Response: Y
##           Df Sum Sq Mean Sq F value    Pr(>F)    
## X1         1 2395.9  2395.9 142.620 1.480e-10 ***
## X2         1 1807.0  1807.0 107.565 1.708e-09 ***
## X3         1 4254.5  4254.5 253.259 8.045e-13 ***
## X4         1  260.7   260.7  15.521   0.00081 ***
## Residuals 20  336.0    16.8                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#Comments: The estimated regression function based on the model output is:
#Predicted Y = -124.38 + 0.296X1 + 0.048X2 + 1.306X3 + 0.520X4. This equation
#represents the predicted Job Proficiency (Y) based on the independent
#variables X1, X2, X3, and X4.

#The ANOVA table shows that the F-statistic for the overall model is very large,
#and the p-values for each predictor are highly significant (p < 0.001). This
#indicates that the model explains a significant portion of the variance in Y,
#meaning the regression model as a whole is statistically significant.

#Part C:
par(mfrow = c(2, 2))
plot(fitted(full_model), residuals(full_model),
     xlab = "Fitted values", ylab = "Residuals",
     main = "Residuals vs Fitted")
abline(h = 0, col = "red")

qqnorm(residuals(full_model))
qqline(residuals(full_model), col = "red")

plot(fitted(full_model), sqrt(abs(residuals(full_model))),
     xlab = "Fitted values", ylab = "square root(|residuals|)",
     main = "Scale-Location")

plot(hatvalues(full_model), residuals(full_model),
     xlab = "Leverage", ylab = "Residuals",
     main = "Residuals vs Leverage")
```



```
#Comments:Residuals vs. Fitted Plot: This plot shows whether residuals exhibit
#any systematic pattern. Ideally, residuals should be randomly scattered around
#zero. In this case, while there is no clear curvature, some heteroscedasticity
#(variance increasing with fitted values) may be present. This suggests a
#possible need for transformation.

#Normal Q-Q Plot: The residuals should follow a normal distribution if the model
#assumptions hold. Here, the points mostly follow the straight line, but there
#are deviations in the tails, suggesting potential non-normality, possibly due
#to outliers or skewness.

#Scale-Location Plot: This plot helps assess homoscedasticity
#(constant variance of residuals). The spread of square root(|residuals|)
#appears to increase slightly with fitted values, reinforcing the potential
#heteroscedasticity concern. A transformation of the dependent variable might
#improve the model.

#Residuals vs. Leverage Plot: This plot identifies influential points. There are
#no extreme leverage points, indicating no single observation is
#disproportionately influencing the model.

#Conclusion: The slight heteroscedasticity suggests a transformation
#(such as log or square root) of the response variable could improve model fit.
#The normality concern in the Q-Q plot may not be severe, but further checks can
#be performed. No significant leverage points were found, so influential
#observations do not seem to be an issue. Overall, the model appears reasonable
```

```
#but could benefit from further refinement, particularly in addressing
#heteroscedasticity.

#Part D:
par(mfrow = c(2, 2))
for (i in 1:4) {
  y_resid <- residuals(lm(Y ~ . -job[,i], data = job))

  x_resid <- residuals(lm(job[,i] ~ . -Y -job[,i], data = job))

  plot(x_resid, y_resid,
       xlab = paste("X", i, " | others"),
       ylab = "Y | others",
       main = paste("Partial Regression: X", i))
  abline(lm(y_resid ~ x_resid), col = "red")
}
```



```
#Comments: Findings for Each Independent Variable (X1, X2, X3, X4):
#X1 (Top Left Plot): The points appear randomly scattered with no strong trend.
#The regression line is almost flat, indicating a weak relationship between X1
#and the response variable (Y), given the other variables in the model. This
#suggests X1 might not be a significant predictor.

#X2 (Top Right Plot): Similar to X1, there is no strong pattern
#in the residuals. The red regression line is slightly increasing, but the
```

```r
#spread of points suggests a weak correlation. X2 likely has little effect on Y
#after controlling for other variables.

#X3 (Bottom Left Plot): The trend line has a slightly more noticeable upward
#slope compared to X1 and X2. There is some evidence that X3 has a weak positive
#effect on Y, but the dispersion of points suggests high variance. X3 might
#contribute slightly to the model but may not be a strong predictor.

#X4 (Bottom Right Plot): The scatter is relatively random with no strong trend.
#The regression line is nearly flat, suggesting little to no effect of X4 on Y
#after accounting for other predictors. X4 is likely not a
#significant predictor.

#Part E:
vif_values <- sapply(1:4, function(i) {
  1 / (1 - summary(lm(job[,i] ~ . -Y -job[,i], data = job))$r.squared)
})
```

```
## Warning in summary.lm(lm(job[, i] ~ . - Y - job[, i], data = job)): essentially
## perfect fit: summary may be unreliable
## Warning in summary.lm(lm(job[, i] ~ . - Y - job[, i], data = job)): essentially
## perfect fit: summary may be unreliable
## Warning in summary.lm(lm(job[, i] ~ . - Y - job[, i], data = job)): essentially
## perfect fit: summary may be unreliable
## Warning in summary.lm(lm(job[, i] ~ . - Y - job[, i], data = job)): essentially
## perfect fit: summary may be unreliable
```

```r
names(vif_values) <- colnames(job)[1:4]

cat("Variance Inflation Factors:\n")
```

```
## Variance Inflation Factors:
```

```r
print(vif_values)
```

```
##  X1  X2  X3  X4
## Inf Inf Inf Inf
```

```r
largest_vif <- which.max(vif_values)
cat("\nLargest VIF is for", names(largest_vif), ":",
    round(vif_values[largest_vif], 2), "\n")
```

```
##
## Largest VIF is for X1 : Inf
```

```r
#Comments: The largest VIF is for X1, and its value is Inf (infinite). A VIF of
#infinity indicates perfect multicollinearity, meaning X1 is perfectly linearly
#related to one or more of the other predictor variables (likely due to a linear
#combination or near-duplicate information). This is a issue because
#multicollinearity inflates the standard errors of the coefficient estimates,
#making it difficult to assess the individual contribution of X1. In simpler
#terms, it means the model cannot reliably estimate the effect of X1 on the
#response variable because its information is already captured by the
#other predictors.

#Part F:
hat_threshold <- 2 * mean(hatvalues(full_model))
warning_threshold <- 0.30

leverage_points <- data.frame(
  Observation = seq_along(hatvalues(full_model)),
  HatValue = round(hatvalues(full_model), 4),
  Status = ifelse(hatvalues(full_model) > hat_threshold,
                  ifelse(hatvalues(full_model) > 0.4,
                        "High Leverage (>0.4)",
                        "Warning (greater than or equal to0.30)"),
                  "Normal")
)

cat("Standard leverage threshold (2*(p+1)/n):", round(hat_threshold, 4), "\n")
```

```
## Standard leverage threshold (2*(p+1)/n): 0.4
```

```r
cat("Warning threshold for near-high leverage: greater than or equal to",
    warning_threshold, "\n\n")
```

```
## Warning threshold for near-high leverage: greater than or equal to 0.3
```

```r
notable_points <- leverage_points[leverage_points$HatValue >=
                                    warning_threshold, ]
notable_points <- notable_points[order(-notable_points$HatValue), ]

if (nrow(notable_points) > 0) {
  cat("Notable leverage points:\n")
  print(notable_points, row.names = FALSE)

  cat("\nFull data for notable points:\n")
  print(job[notable_points$Observation, ])
} else {
  cat("No observations with hat values greater than or equal to",
      warning_threshold, "\n")
  closest <- leverage_points[order(-leverage_points$HatValue), ][1:3,]
  cat("\nTop 3 highest leverage points:\n")
  print(closest, row.names = FALSE)
}
```

```
## Notable leverage points:
##  Observation HatValue Status
##            7   0.3656 Normal
##            2   0.3515 Normal
##           13   0.3007 Normal
##
```

```
## Full data for notable points:
##     X1  X2 X3  X4  Y
## 7  120  77 80  74 58
## 2   62  97 99 100 80
## 13 140 121 96  89 94
```

```
#Comments: The threshold for identifying high leverage points is calculated as:
#Threshold = 2 * mean(hat-values). The high leverage threshold was 0.4. There
#were no high leverage points were detected (no observations had hat values
#exceeding 0.4).

#Part G:
n <- nrow(job)
p <- length(coef(full_model))
h <- hatvalues(full_model)
std_res <- residuals(full_model) / (summary(full_model)$sigma * sqrt(1 - h))
student_res <- std_res * sqrt((n - p - 1) / (n - p - std_res^2))

outliers <- which(abs(student_res) > 2)
cat("Potential outliers (|rstudent| > 2):\n")
```

```
## Potential outliers (|rstudent| > 2):
```

```
if (length(outliers) > 0) {
  print(outliers)
} else {
  cat("No outliers detected")
}
```

```
## No outliers detected
```

```
#Part H:
reduced_model <- lm(Y ~ X1 + X3, data = job)

SSE_reduced <- sum(residuals(reduced_model)^2)
SSE_full <- sum(residuals(full_model)^2)
df_diff <- df.residual(reduced_model) - df.residual(full_model)
F_stat <- ((SSE_reduced - SSE_full)/df_diff) /
  (SSE_full/df.residual(full_model))
p_value <- pf(F_stat, df_diff, df.residual(full_model), lower.tail = FALSE)

cat("F-test comparing full and reduced models:\n")
```

```
## F-test comparing full and reduced models:
```

```
cat("F =", round(F_stat, 3), "on", df_diff, "and",
    df.residual(full_model), "DF, p-value =", round(p_value, 4), "\n")
```

```
## F = 8.056 on 2 and 20 DF, p-value = 0.0027
```

```r
#Comments: Null Hypothesis: The reduced model is sufficient, meaning X2 and X4
#do not provide significant additional predictive power.
#Alternative Hypothesis: The reduced model is insufficient, meaning at least one
#of X2 or X4 significantly improves the model.

#Since p-value (0.0027) < 0.05, we reject the null hypothesis at the 5%
#significance level. This result suggests that at least one of X2 or X4
#contributes significantly to predicting job proficiency (Y). Removing these
#variables weakens the model, indicating that the full model is preferable over
#the reduced model.

#Part I:
job$X1_plus_X3 <- job$X1 + job$X3
constrained_model <- lm(Y ~ X1_plus_X3, data = job)

SSE_constrained <- sum(residuals(constrained_model)^2)
SSE_reduced <- sum(residuals(reduced_model)^2)
F_stat <- ((SSE_constrained - SSE_reduced)/1) /
  (SSE_reduced/df.residual(reduced_model))
p_value <- pf(F_stat, 1, df.residual(reduced_model), lower.tail = FALSE)

cat("Test of B1' = B3':\n")
```

```
## Test of B1’ = B3’:
```

```r
cat("F =", round(F_stat, 3), "on 1 and", df.residual(reduced_model),
    "DF, p-value =", round(p_value, 4), "\n")
```

```
## F = 106.511 on 1 and 22 DF, p-value = 0
```

```r
#Comments: (B = beta, E = error term) Full Model: Y = B0 + B1X1 + B3X3 + E
#Reduced Model: Y = B0 + B'(X1+X3) + E
#Null Hypothesis: B'1 = B'3 (the combined effect of X1 and X3 is valid).
#Alternative Hypothesis: B'1 does not equal B'3 (the assumption of equal
#coefficients is incorrect).
#The test statistic is F = ((SSEconstrained - SSEreduced) / 1) /
#(SSEreduced/dfreduced). At the 1% significance level (alpha = 0.01), we reject
#the null hypothesis if the computed F-statistic exceeds the critical F-value
#for 1 and 22 degrees of freedom. The computed F-statistic = 106.511 and the
#p-value = 0 (less than 0.01). Since p-value < 0.01, we reject the null
#hypothesis and conclude that B'1 does not equal B'3. This means the assumption
#that X1 and X3 contribute equally to predicting job proficiency (Y) is
#incorrect, and they should be treated as separate predictors in the model.
```