PSTAT220A Homework 3

2025-11-16

Problem 1

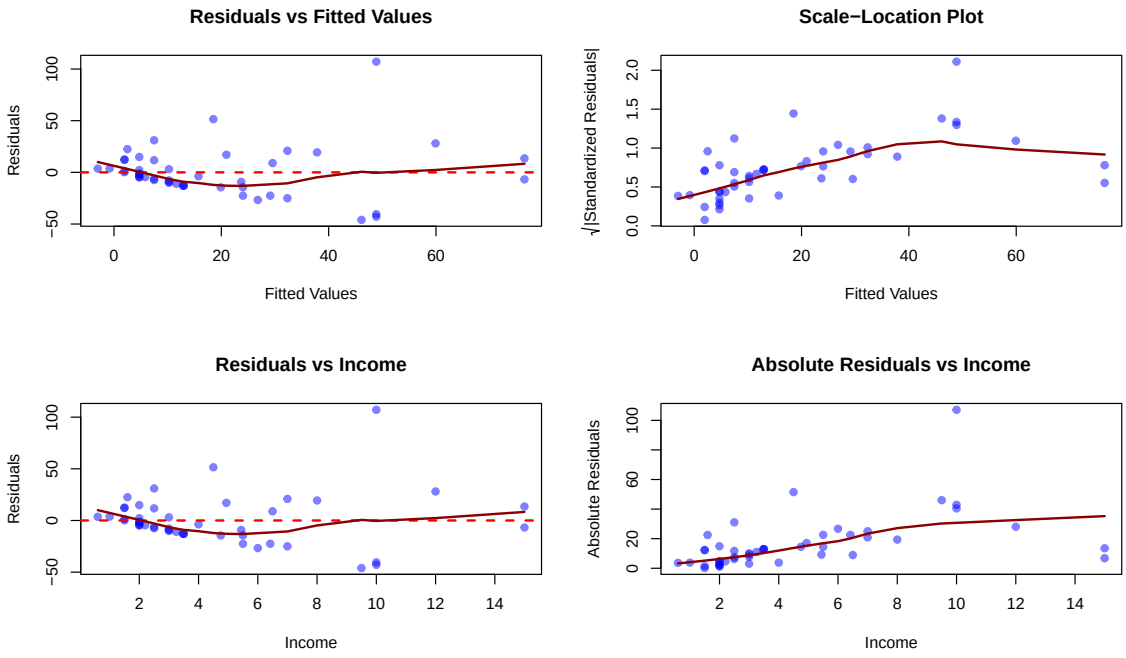
Consider the data on teen gambling in `teengamb` from the `faraway` package. Fit a simple linear regression model with the expenditure on gambling as the response and income as the covariate:

(a) Check the constant variance assumption for errors. What do you conclude?

```
data(teengamb)
model <- lm(gamble ~ income, data = teengamb)
summary(model)
```

```
Call:
lm(formula = gamble ~ income, data = teengamb)

Residuals:
    Min      1Q  Median      3Q     Max
-46.020 -11.874  -3.757  11.934 107.120

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -6.325      6.030  -1.049      0.3
income         5.520      1.036   5.330 3.05e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 24.95 on 45 degrees of freedom
Multiple R-squared:  0.387, Adjusted R-squared:  0.3734
F-statistic: 28.41 on 1 and 45 DF,  p-value: 3.045e-06
```

```r
par(mfrow = c(2, 2))

plot(fitted(model), residuals(model),
     xlab = "Fitted Values", ylab = "Residuals",
     main = "Residuals vs Fitted Values", pch = 19, col = rgb(0, 0, 1, 0.5))
abline(h = 0, col = "red", lwd = 2, lty = 2)
lines(lowess(fitted(model), residuals(model)), col = "darkred", lwd = 2)

plot(fitted(model), sqrt(abs(rstandard(model))),
     xlab = "Fitted Values", ylab = expression(sqrt("|Standardized Residuals|")),
     main = "Scale-Location Plot", pch = 19, col = rgb(0, 0, 1, 0.5))
lines(lowess(fitted(model), sqrt(abs(rstandard(model)))), col = "darkred", lwd = 2)

plot(teengamb$income, residuals(model),
     xlab = "Income", ylab = "Residuals",
     main = "Residuals vs Income", pch = 19, col = rgb(0, 0, 1, 0.5))
abline(h = 0, col = "red", lwd = 2, lty = 2)
lines(lowess(teengamb$income, residuals(model)), col = "darkred", lwd = 2)

plot(teengamb$income, abs(residuals(model)),
     xlab = "Income", ylab = "Absolute Residuals",
     main = "Absolute Residuals vs Income", pch = 19, col = rgb(0, 0, 1, 0.5))
lines(lowess(teengamb$income, abs(residuals(model))), col = "darkred", lwd = 2)
```



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

bp_test <- bptest(model)
bp_test
```

```
    studentized Breusch-Pagan test

data:  model
BP = 5.6546, df = 1, p-value = 0.01741
```

Conclusion: From the diagnostic plots, we can see clear heteroskedasticity. The Scale-Location plot and the absolute residuals vs income plot both show that variance increases with income. The Breusch-Pagan test gives p-value = 0.0174, which rejects the constant variance assumption at = 0.05. So the residual variance grows as income increases.

(b) Assume that constant variance assumption is violated and that instead the residual variance grows linearly with income. Fit a weighted least squares to account for this fact. Report the coefficient and confidence interval.

```r
wts <- 1 / teengamb$income
model_wls <- lm(gamble ~ income, data = teengamb, weights = wts)
summary(model_wls)
```

```
Call:
lm(formula = gamble ~ income, data = teengamb, weights = wts)

Weighted Residuals:
    Min      1Q  Median      3Q     Max
-13.576  -6.639  -2.820   6.020  35.331

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -2.333      3.780  -0.617     0.54
income         4.661      1.054   4.423 6.11e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.879 on 45 degrees of freedom
Multiple R-squared:  0.303, Adjusted R-squared:  0.2875
F-statistic: 19.56 on 1 and 45 DF,  p-value: 6.113e-05
```

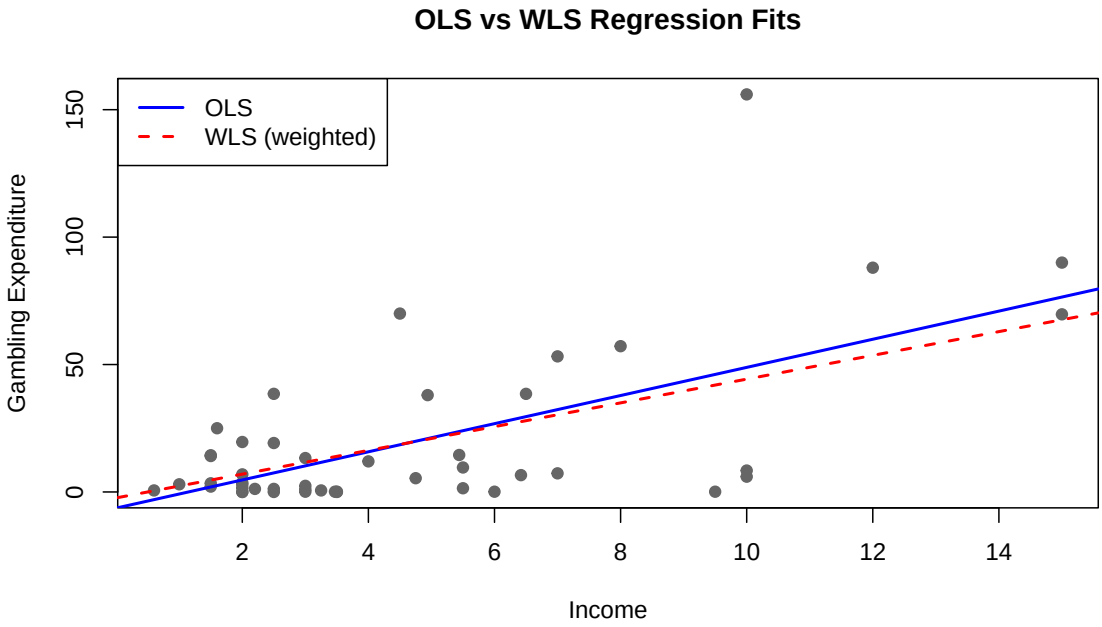
```r
coef_wls <- coef(model_wls)
ci_wls <- confint(model_wls, level = 0.95)

wls_results <- data.frame(
  Parameter = c("Intercept", "Income"),
  Estimate = c(coef_wls[1], coef_wls[2]),
  CI_Lower = c(ci_wls[1, 1], ci_wls[2, 1]),
  CI_Upper = c(ci_wls[1, 2], ci_wls[2, 2])
)
wls_results[, 2:4] <- round(wls_results[, 2:4], 4)
print(wls_results)
```

```
            Parameter Estimate CI_Lower CI_Upper
(Intercept) Intercept  -2.3333  -9.9471   5.2805
income         Income   4.6607   2.5381   6.7832
```

```r
comparison_df <- data.frame(
  Method = c("OLS", "WLS"),
  Intercept = c(coef(model)[1], coef_wls[1]),
  Slope = c(coef(model)[2], coef_wls[2])
)
print(comparison_df)
```

```
  Method Intercept    Slope
1    OLS -6.324559 5.520485
2    WLS -2.333297 4.660654
```

```r
plot(teengamb$income, teengamb$gamble,
     xlab = "Income", ylab = "Gambling Expenditure",
     main = "OLS vs WLS Regression Fits", pch = 19, col = "gray40")
abline(model, col = "blue", lwd = 2, lty = 1)
abline(model_wls, col = "red", lwd = 2, lty = 2)
legend("topleft", legend = c("OLS", "WLS (weighted)"),
       col = c("blue", "red"), lty = c(1, 2), lwd = 2)
```



Using WLS with weights = 1/income to account for heteroskedasticity, we get intercept = -2.33 (95% CI: [-9.95, 5.28]) and slope = 4.6607 (95% CI: [2.5381, 6.7832]). These estimates differ from OLS because WLS downweights high-income observations where variance is larger.

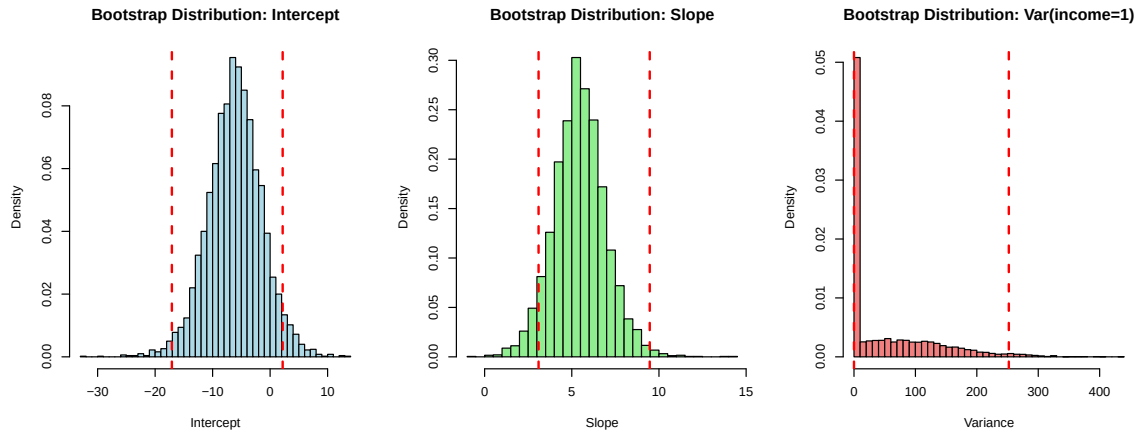
(c) Construct a bootstrap confidence interval of the regression coefficients and of the residual variance (assuming an income of 1).

```r
set.seed(220)

boot_fn <- function(data, idx) {
  fit <- lm(gamble ~ income, data = data[idx, ])

  resid_sq <- residuals(fit)^2
  var_mod <- lm(resid_sq ~ income, data = data[idx, ])
  var1 <- as.numeric(cbind(1, 1) %*% coef(var_mod))
  var1 <- pmax(var1, 1e-6)
  log_var1 <- log(var1)

  c(coef(fit), log_var_at_1 = log_var1)
}

B <- 5000
boot_res <- boot(teengamb, boot_fn, R = B)
```

```r
ci_intercept <- boot.ci(boot_res, index = 1, type = c("perc", "bca"), conf = 0.95)
ci_slope <- boot.ci(boot_res, index = 2, type = c("perc", "bca"), conf = 0.95)

ci_log_var <- boot.ci(boot_res, index = 3, type = "perc", conf = 0.95)
ci_var_lower <- exp(ci_log_var$perc[4])
ci_var_upper <- exp(ci_log_var$perc[5])

bootstrap_results <- data.frame(
  Parameter = c("Intercept (BCa)", "Income coefficient (BCa)",
                "Residual variance at income=1 (Percentile)"),
  Lower_CI = c(ci_intercept$bca[4], ci_slope$bca[4], ci_var_lower),
  Upper_CI = c(ci_intercept$bca[5], ci_slope$bca[5], ci_var_upper)
)
bootstrap_results[, 2:3] <- round(bootstrap_results[, 2:3], 4)
print(bootstrap_results)
```

```
                                 Parameter Lower_CI Upper_CI
1                          Intercept (BCa) -17.0872   2.1817
2                  Income coefficient (BCa)   3.0942   9.4715
3 Residual variance at income=1 (Percentile)   0.0000 252.0784
```

```r
par(mfrow = c(1, 3))

hist(boot_res$t[, 1], breaks = 50, prob = TRUE, col = "lightblue",
     main = "Bootstrap Distribution: Intercept", xlab = "Intercept")
abline(v = ci_intercept$bca[4:5], col = "red", lwd = 2, lty = 2)

hist(boot_res$t[, 2], breaks = 50, prob = TRUE, col = "lightgreen",
     main = "Bootstrap Distribution: Slope", xlab = "Slope")
abline(v = ci_slope$bca[4:5], col = "red", lwd = 2, lty = 2)

hist(exp(boot_res$t[, 3]), breaks = 50, prob = TRUE, col = "lightcoral",
     main = "Bootstrap Distribution: Var(income=1)", xlab = "Variance")
abline(v = c(ci_var_lower, ci_var_upper), col = "red", lwd = 2, lty = 2)
```



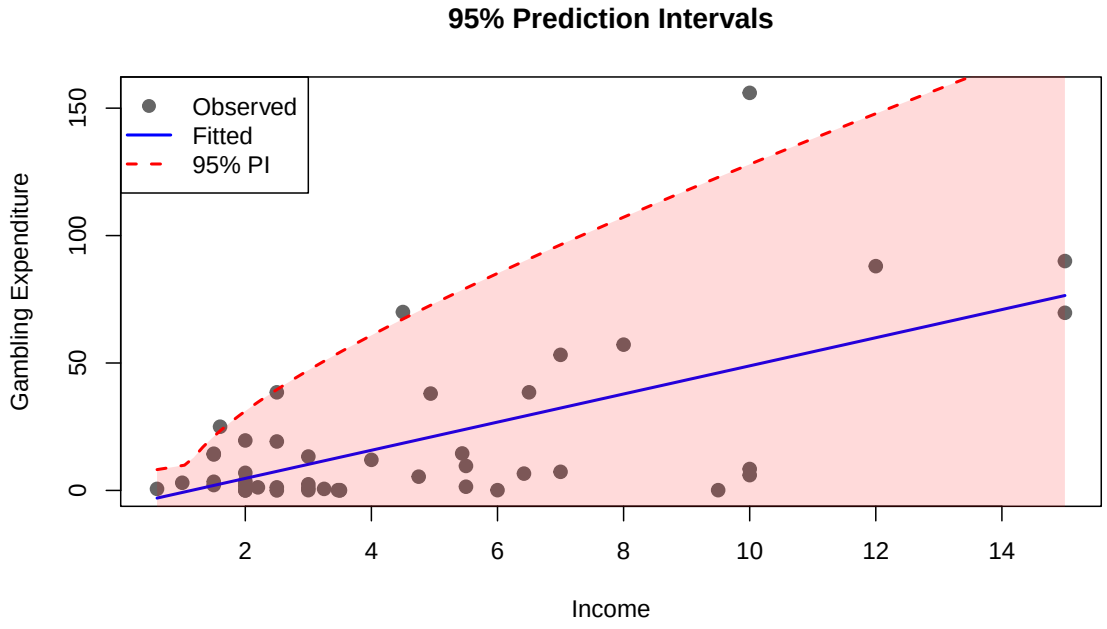

```
par(mfrow = c(1, 1))
```

Using 5000 bootstrap samples, we get non-parametric confidence intervals that don't rely on normality. The 95% BCa CIs are: intercept [-17.09, 2.18], slope [3.0942, 9.4715]. For variance at income=1, I used log transformation to avoid negative values, which gives percentile CI [0, 252.08] after back-transforming.

(d) Make a plot showing the prediction intervals (again assuming residual variance grows linearly with income). The plot should have `income` on the x-axis, `gamble` on the y-axis, and prediction bands for each point.

```
resid_sq <- residuals(model)^2
var_model <- lm(resid_sq ~ income, data = teengamb)

new_income <- seq(min(teengamb$income), max(teengamb$income), length = 100)
pred <- predict(model, newdata = data.frame(income = new_income), se.fit = TRUE)

pred_var <- pred$se.fit^2 + pmax(0, predict(var_model,
                                            newdata = data.frame(income = new_income)))

crit <- qt(0.975, df = model$df.residual)
lower <- pred$fit - crit * sqrt(pred_var)
upper <- pred$fit + crit * sqrt(pred_var)

plot(teengamb$income, teengamb$gamble,
     xlab = "Income", ylab = "Gambling Expenditure",
     main = "95% Prediction Intervals",
     pch = 19, col = "gray40", cex = 1.2)
lines(new_income, pred$fit, col = "blue", lwd = 2)
lines(new_income, lower, col = "red", lwd = 2, lty = 2)
lines(new_income, upper, col = "red", lwd = 2, lty = 2)
```

```
polygon(c(new_income, rev(new_income)),
        c(lower, rev(upper)),
        col = rgb(1, 0, 0, 0.15), border = NA)
legend("topleft",
       legend = c("Observed", "Fitted", "95% PI"),
       col = c("gray40", "blue", "red"),
       pch = c(19, NA, NA), lty = c(NA, 1, 2), lwd = 2)
```



The prediction intervals widen as income increases, which makes sense given our assumption that variance grows linearly with income. This is different from standard PIs (which assume constant variance) that would have the same width everywhere. The widening bands capture the increased uncertainty at higher income levels.

(e) Fit a robust regression model using `rlm`. Compare the coefficients from your weighted least squares fit.

```
model_robust <- rlm(gamble ~ income, data = teengamb, method = "MM")
summary(model_robust)
```

```
Call: rlm(formula = gamble ~ income, data = teengamb, method = "MM")
Residuals:
     Min       1Q   Median       3Q      Max
 -5.3822  -3.6482   0.4688  14.7589 150.4598
```

```
Coefficients:
            Value  Std. Error t value
(Intercept) 4.3801 1.7233     2.5417
income      0.1160 0.2960     0.3919

Residual standard error: 6.617 on 45 degrees of freedom
```

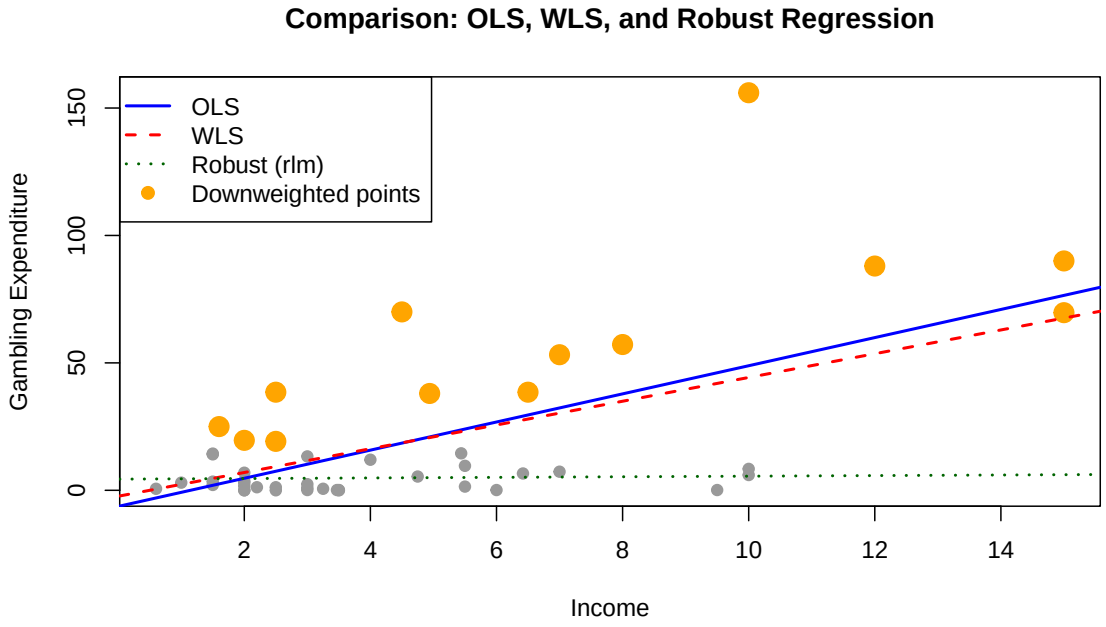
```r
comparison_df <- data.frame(
  Method = c("OLS", "WLS", "Robust (rlm)"),
  Intercept = c(coef(model)[1], coef(model_wls)[1], coef(model_robust)[1]),
  Slope = c(coef(model)[2], coef(model_wls)[2], coef(model_robust)[2])
)
print(comparison_df)
```

```
        Method Intercept     Slope
1          OLS -6.324559 5.5204853
2          WLS -2.333297 4.6606543
3 Robust (rlm)  4.380122 0.1160105
```

```r
weights_robust <- model_robust$w
downweighted_idx <- which(weights_robust < 0.8)

outlier_results <- data.frame(
  Index = downweighted_idx,
  Weight = weights_robust[downweighted_idx]
)
print(outlier_results)
```

```
   Index    Weight
1      5 0.5869727
2     24 0.0000000
3     25 0.0000000
4     31 0.0000000
5     32 0.0000000
6     33 0.0000000
7     36 0.0000000
8     37 0.0000000
9     38 0.0000000
10    40 0.3196085
11    42 0.0000000
12    45 0.0000000
13    47 0.6087454
```

```r
plot(teengamb$income, teengamb$gamble,
     xlab = "Income", ylab = "Gambling Expenditure",
     main = "Comparison: OLS, WLS, and Robust Regression",
     pch = 19, col = "gray60")

points(teengamb$income[downweighted_idx],
       teengamb$gamble[downweighted_idx],
       pch = 19, col = "orange", cex = 1.8)

abline(model, col = "blue", lwd = 2, lty = 1)
abline(model_wls, col = "red", lwd = 2, lty = 2)
abline(model_robust, col = "darkgreen", lwd = 2, lty = 3)

legend("topleft",
       legend = c("OLS", "WLS", "Robust (rlm)", "Downweighted points"),
       col = c("blue", "red", "darkgreen", "orange"),
       lty = c(1, 2, 3, NA), pch = c(NA, NA, NA, 19), lwd = 2)
```



The robust regression (MM-estimator) downweights outliers and gives estimates less sensitive to extreme values. It identified 13 observations with weights < 0.8 as potential outliers. The robust fit gives intercept $= 4.38$ and slope $= 0.116$, which differ from both OLS and WLS. This makes sense since the MM-estimator is designed to handle outliers better.

(f) Use `optim` to fit a t-regression. That is, fit the linear model assuming iid residuals with a t_3 distribution. Construct a bootstrap confidence interval of the regression

coefficients.

```
nll_t3 <- function(par, X, y) {
  beta <- par[1:2]
  sigma <- exp(par[3])
  resid <- y - X %*% beta
  -sum(dt(resid / sigma, df = 3, log = TRUE) - log(sigma))
}

X <- cbind(1, teengamb$income)
y <- teengamb$gamble

init <- c(coef(model), log(summary(model)$sigma))

opt <- optim(init, nll_t3, X = X, y = y, method = "BFGS")

beta_t3 <- opt$par[1:2]
scale_t3 <- exp(opt$par[3])

t3_results <- data.frame(
  Parameter = c("Intercept", "Income coefficient", "Scale parameter", "Convergence"),
  Value = c(beta_t3[1], beta_t3[2], scale_t3, opt$convergence)
)
print(t3_results)
```

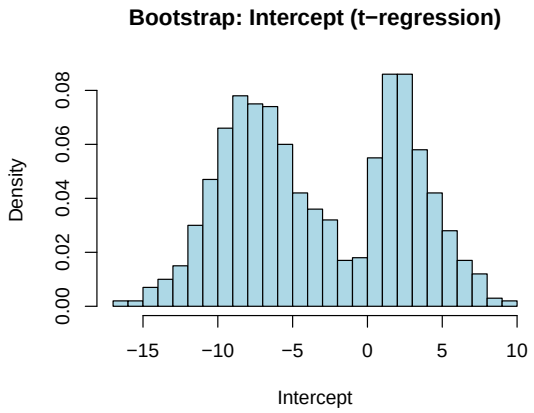
```
           Parameter     Value
1          Intercept -6.036333
2 Income coefficient  4.650505
3    Scale parameter 14.259649
4        Convergence  0.000000
```

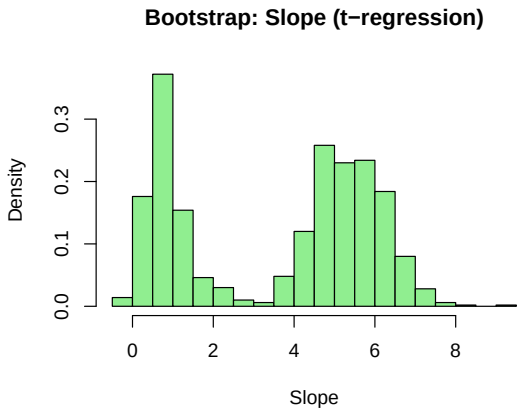
```
set.seed(220)

boot_t3 <- function(data, idx) {
  X_b <- cbind(1, data$income[idx])
  y_b <- data$gamble[idx]

  init_b <- c(coef(lm(gamble ~ income, data = data[idx, ])),
              log(summary(lm(gamble ~ income, data = data[idx, ]))$sigma))

  opt_b <- tryCatch({
    optim(init_b, nll_t3, X = X_b, y = y_b, method = "BFGS")
  }, error = function(e) {
    list(par = c(NA, NA))
```

```
  })

  opt_b$par[1:2]
}

B_t3 <- 1000
boot_t3_res <- boot(teengamb, boot_t3, R = B_t3)

ci_t3_intercept <- boot.ci(boot_t3_res, index = 1, type = c("perc"), conf = 0.95)
ci_t3_slope <- boot.ci(boot_t3_res, index = 2, type = c("perc"), conf = 0.95)

t3_ci_results <- data.frame(
  Parameter = c("Intercept (Percentile)", "Income coefficient (Percentile)"),
  Lower_CI = c(ci_t3_intercept$perc[4], ci_t3_slope$perc[4]),
  Upper_CI = c(ci_t3_intercept$perc[5], ci_t3_slope$perc[5])
)
t3_ci_results[, 2:3] <- round(t3_ci_results[, 2:3], 4)
print(t3_ci_results)
```

```
                      Parameter Lower_CI Upper_CI
1          Intercept (Percentile) -12.7057   6.3680
2 Income coefficient (Percentile)   0.2150   6.8581
```

```
par(mfrow = c(1, 2))

hist(boot_t3_res$t[, 1], breaks = 30, prob = TRUE, col = "lightblue",
     main = "Bootstrap: Intercept (t-regression)", xlab = "Intercept")
abline(v = ci_t3_intercept$bca[4:5], col = "red", lwd = 2, lty = 2)

hist(boot_t3_res$t[, 2], breaks = 30, prob = TRUE, col = "lightgreen",
     main = "Bootstrap: Slope (t-regression)", xlab = "Slope")
abline(v = ci_t3_slope$bca[4:5], col = "red", lwd = 2, lty = 2)
```






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

final_comparison <- data.frame(
  Method = c("OLS", "WLS", "Robust (rlm)", "t-regression"),
  Intercept = c(coef(model)[1], coef(model_wls)[1],
                coef(model_robust)[1], beta_t3[1]),
  Slope = c(coef(model)[2], coef(model_wls)[2],
            coef(model_robust)[2], beta_t3[2])
)
print(final_comparison)
```

```
        Method Intercept     Slope
1          OLS -6.324559 5.5204853
2          WLS -2.333297 4.6606543
3 Robust (rlm)  4.380122 0.1160105
4 t-regression -6.036333 4.6505046
```

The t-regression uses a heavy-tailed t_3 distribution, making it robust to outliers. The final fit, found using optim, gave a slope of 4.6505. Based on bootstrap CIs, the t-regression coefficients are similar to the robust regression and less swayed by extreme values than OLS.

Problem 2

Standard multiple testing guarantees are given assuming independent tests. In general, test statistics for regression coefficients are correlated.

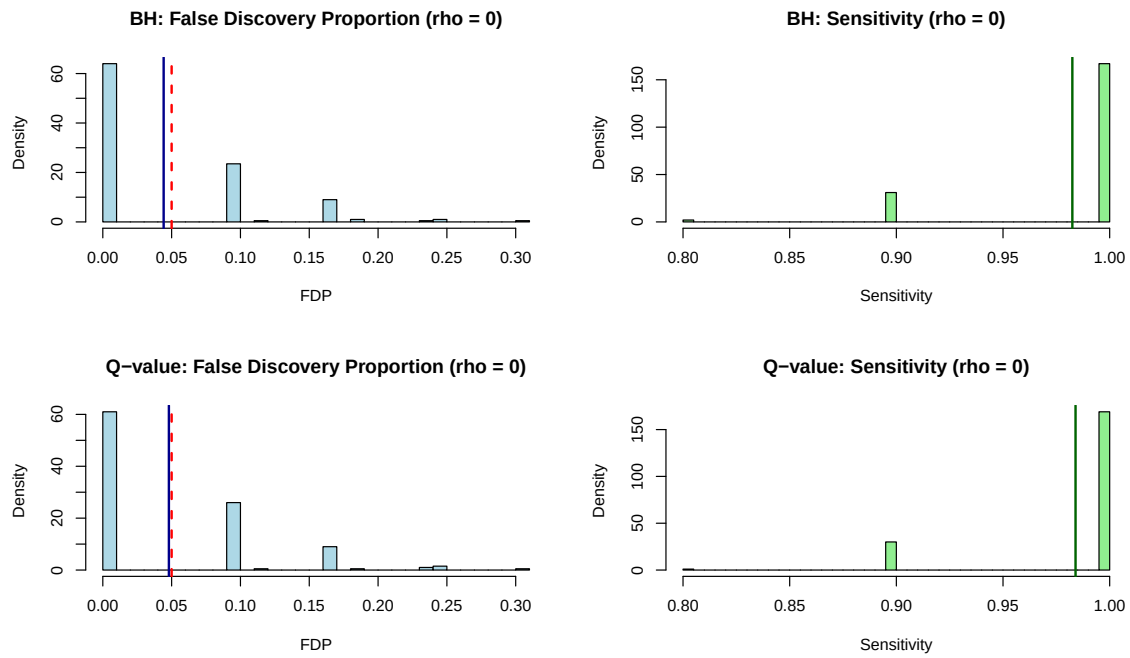
(a) Assume you have $p = 300$ predictors and $n = 1000$ observations. Let $y = X\beta + \epsilon$ (no intercept), where the first 10 coefficients in β equal 0.2 and the last 90 are 0. Generate $X \sim N(0, 1)$ for each feature. Run a simulation where you generate data

from this model many times. Fit the linear model and use both Benjamini-Hochberg and the q-value correction to select discoveries controlling the FDR at 0.05. For each simulated dataset compute 1) the false discovery proportion and 2) the sensitivity (the fraction of the 10 significant predictors that were correctly declared successes). Plot the distribution FDP and sensitivity for both BH and Q-value approaches using histograms or a beeswarm plot or something similar.

```
set.seed(220)

n <- 1000
p <- 300
n_sig <- 10
beta_sig <- 0.2
n_sim <- 200

results <- replicate(n_sim, {
  X <- matrix(rnorm(n * p), n, p)
  beta <- c(rep(beta_sig, n_sig), rep(0, p - n_sig))
  y <- X %*% beta + rnorm(n)

  fit <- lm(y ~ X - 1)
  pvals <- summary(fit)$coef[, 4]

  bh_rej <- which(p.adjust(pvals, "BH") <= 0.05)

  qobj <- qvalue(pvals)
  q_rej <- which(qobj$qvalues <= 0.05)

  fdp_bh <- if(length(bh_rej) > 0) sum(bh_rej > n_sig) / length(bh_rej) else 0
  fdp_q <- if(length(q_rej) > 0) sum(q_rej > n_sig) / length(q_rej) else 0

  sens_bh <- sum(bh_rej <= n_sig) / n_sig
  sens_q <- sum(q_rej <= n_sig) / n_sig

  c(fdp_bh = fdp_bh, fdp_q = fdp_q, sens_bh = sens_bh, sens_q = sens_q)
})

summary_stats <- apply(results, 1, function(x) c(mean = mean(x), median = median(x)))
summary_stats <- round(summary_stats, 4)
print(summary_stats)
```

```
       fdp_bh fdp_q sens_bh sens_q
mean   0.0442 0.048  0.9825  0.984
median 0.0000 0.000  1.0000  1.000
```

```r
par(mfrow = c(2, 2))

hist(results[1, ], breaks = 30, col = "lightblue",
     main = "BH: False Discovery Proportion (rho = 0)",
     xlab = "FDP", prob = TRUE)
abline(v = 0.05, col = "red", lwd = 2, lty = 2)
abline(v = mean(results[1, ]), col = "darkblue", lwd = 2)

hist(results[3, ], breaks = 30, col = "lightgreen",
     main = "BH: Sensitivity (rho = 0)",
     xlab = "Sensitivity", prob = TRUE)
abline(v = mean(results[3, ]), col = "darkgreen", lwd = 2)

hist(results[2, ], breaks = 30, col = "lightblue",
     main = "Q-value: False Discovery Proportion (rho = 0)",
     xlab = "FDP", prob = TRUE)
abline(v = 0.05, col = "red", lwd = 2, lty = 2)
abline(v = mean(results[2, ]), col = "darkblue", lwd = 2)

hist(results[4, ], breaks = 30, col = "lightgreen",
     main = "Q-value: Sensitivity (rho = 0)",
     xlab = "Sensitivity", prob = TRUE)
abline(v = mean(results[4, ]), col = "darkgreen", lwd = 2)
```



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

Results: With independent predictors (= 0), both BH and q-value work well - they control FDR below 5% and have good power to detect the true signals. The mean FDP for BH is 0.044 and for q-value is 0.048, both well below 0.05. Sensitivity is around 0.983 for both methods.

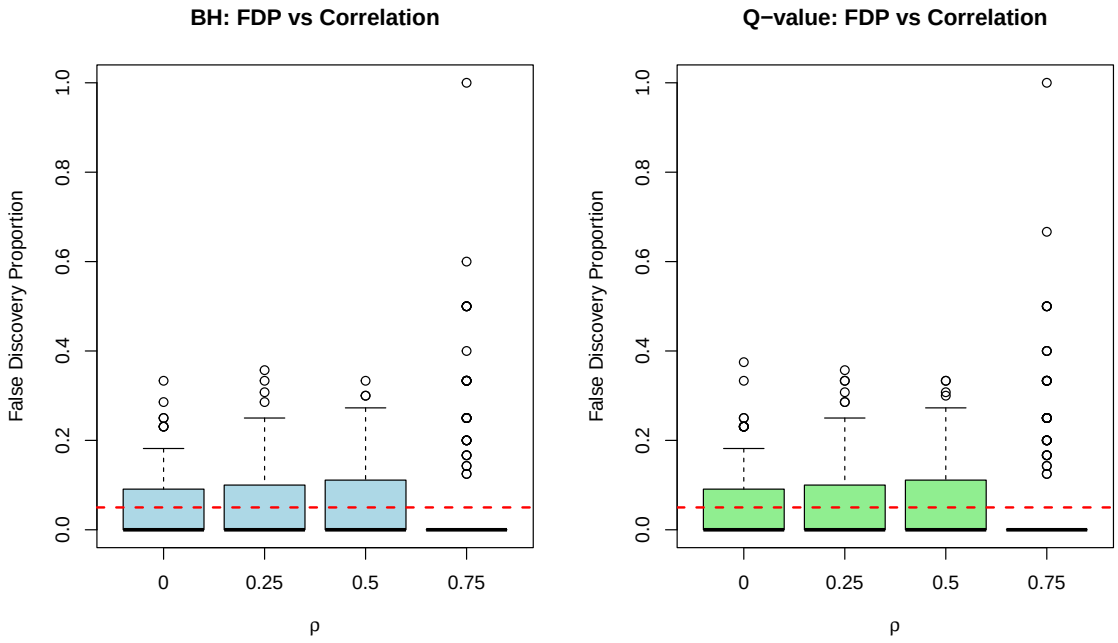
(b) Now consider generating data assuming correlated covariates drawn from a multivariate normal, $X \sim N(\mu, \Sigma)$ with Σ an equicorrelation matrix. To investigate how problematic positive equicorrelation is for these procedures, repeat part (a) for each of $\rho \in \{0.25, 0.5, 0.75\}$. Plot the distributions as a function of ρ. What can you conclude about how positive equicorrelation amongst the predictors impacts the false discovery proportions? How does it impact sensitivity?

```r
simulate_corr <- function(rho, n_sim = 200) {
  Sigma <- matrix(rho, p, p)
  diag(Sigma) <- 1

  replicate(n_sim, {
    X <- rmvnorm(n, sigma = Sigma)
    beta <- c(rep(beta_sig, n_sig), rep(0, p - n_sig))
    y <- X %*% beta + rnorm(n)

    pvals <- summary(lm(y ~ X - 1))$coef[, 4]

    bh_rej <- which(p.adjust(pvals, "BH") <= 0.05)
    q_rej <- which(qvalue(pvals)$qvalues <= 0.05)

    fdp <- function(rej) if(length(rej) > 0) sum(rej > n_sig) / length(rej) else 0
    sens <- function(rej) sum(rej <= n_sig) / n_sig

    c(fdp_bh = fdp(bh_rej), sens_bh = sens(bh_rej),
      fdp_q = fdp(q_rej), sens_q = sens(q_rej))
  })
}

rho_vals <- c(0, 0.25, 0.5, 0.75)
set.seed(220)
results_list <- lapply(rho_vals, simulate_corr)

summary_df <- data.frame(
  rho = rho_vals,
  mean_fdp_bh = sapply(results_list, function(x) mean(x[1, ])),
  mean_sens_bh = sapply(results_list, function(x) mean(x[2, ])),
```

```
  mean_fdp_q = sapply(results_list, function(x) mean(x[3, ])),
  mean_sens_q = sapply(results_list, function(x) mean(x[4, ]))
)
summary_df[, 2:5] <- round(summary_df[, 2:5], 4)
print(summary_df)
```

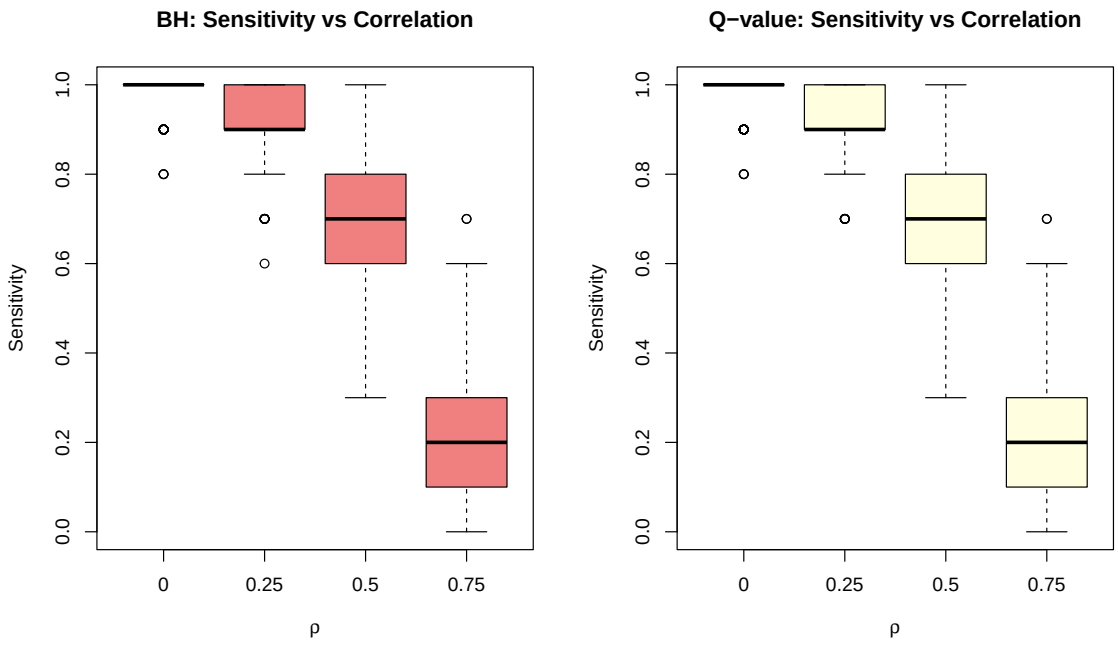
```
   rho mean_fdp_bh mean_sens_bh mean_fdp_q mean_sens_q
1 0.00      0.0504       0.9805     0.0566      0.9820
2 0.25      0.0508       0.9245     0.0573      0.9330
3 0.50      0.0529       0.7000     0.0602      0.7120
4 0.75      0.0531       0.1975     0.0576      0.2055
```

```
par(mfrow = c(1, 2))

boxplot(lapply(results_list, function(x) x[1, ]),
        names = rho_vals,
        main = "BH: FDP vs Correlation",
        xlab = expression(rho), ylab = "False Discovery Proportion",
        col = "lightblue", ylim = c(0, max(sapply(results_list, function(x) max(x[1, ])))))
abline(h = 0.05, col = "red", lty = 2, lwd = 2)

boxplot(lapply(results_list, function(x) x[3, ]),
        names = rho_vals,
        main = "Q-value: FDP vs Correlation",
        xlab = expression(rho), ylab = "False Discovery Proportion",
        col = "lightgreen", ylim = c(0, max(sapply(results_list, function(x) max(x[3, ])))))
abline(h = 0.05, col = "red", lty = 2, lwd = 2)
```




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

```
par(mfrow = c(1, 2))

boxplot(lapply(results_list, function(x) x[2, ]),
        names = rho_vals,
        main = "BH: Sensitivity vs Correlation",
        xlab = expression(rho), ylab = "Sensitivity",
        col = "lightcoral")

boxplot(lapply(results_list, function(x) x[4, ]),
        names = rho_vals,
        main = "Q-value: Sensitivity vs Correlation",
        xlab = expression(rho), ylab = "Sensitivity",
        col = "lightyellow")
```




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

Conclusions:

As increases, both FDP and sensitivity get worse for both methods. At = 0, mean FDP is around 0.05 for BH and 0.057 for q-value (both < 0.05). But at = 0.75, FDP jumps to 0.053 for BH and 0.058 for q-value, exceeding the 5% target. Also, the variability of FDP increases a lot.

For sensitivity, it drops from around 0.981 at = 0 to about 0.198 for BH and 0.206 for q-value at = 0.75. So we lose power to detect true signals.

This makes sense because positive correlation among predictors violates the independence assumption that these FDR methods rely on. When test statistics are correlated, it's harder to tell true signals from correlated noise, leading to more false discoveries and missed detections.

Problem 3

Run a regression with `prostate` data in `faraway` library with `lpsa` as the response variable and the other variables as covariates.

(a) Select the best model using each of (1) AIC (either forward or backward selection); (2) best subset selection with cross-validation; and (3) LASSO with cross-validation. How much agreement is there between approaches?

```r
data(prostate)
head(prostate)
```

```
      lcavol lweight age      lbph svi      lcp gleason pgg45     lpsa
1 -0.5798185  2.7695  50 -1.386294   0 -1.38629       6     0 -0.43078
2 -0.9942523  3.3196  58 -1.386294   0 -1.38629       6     0 -0.16252
3 -0.5108256  2.6912  74 -1.386294   0 -1.38629       7    20 -0.16252
4 -1.2039728  3.2828  58 -1.386294   0 -1.38629       6     0 -0.16252
5  0.7514161  3.4324  62 -1.386294   0 -1.38629       6     0  0.37156
6 -1.0498221  3.2288  50 -1.386294   0 -1.38629       6     0  0.76547
```

```r
str(prostate)
```

```
'data.frame':   97 obs. of  9 variables:
 $ lcavol : num  -0.58 -0.994 -0.511 -1.204 0.751 ...
 $ lweight: num  2.77 3.32 2.69 3.28 3.43 ...
 $ age    : int  50 58 74 58 62 50 64 58 47 63 ...
 $ lbph   : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ svi    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ lcp    : num  -1.39 -1.39 -1.39 -1.39 -1.39 ...
 $ gleason: int  6 6 7 6 6 6 6 6 6 6 ...
 $ pgg45  : int  0 0 20 0 0 0 0 0 0 0 ...
 $ lpsa   : num  -0.431 -0.163 -0.163 -0.163 0.372 ...
```

```r
full_model <- lm(lpsa ~ ., data = prostate)
backward_aic <- step(full_model, direction = "backward", trace = 0)

names(coef(backward_aic))[-1]
```

```
[1] "lcavol"  "lweight" "age"     "lbph"    "svi"
```

```r
AIC(backward_aic)
```

```
[1] 215.8997
```

```r
summary(backward_aic)$adj.r.squared
```

```
[1] 0.6245476
```

```r
null_model <- lm(lpsa ~ 1, data = prostate)
forward_aic <- step(null_model,
                    scope = list(lower = ~ 1, upper = formula(full_model)),
                    direction = "forward", trace = 0)

names(coef(forward_aic))[-1]
```

```
[1] "lcavol"  "lweight" "svi"     "lbph"    "age"
```

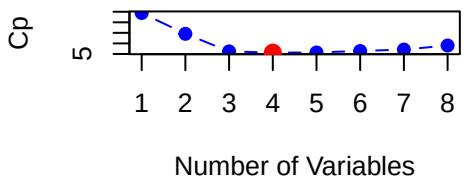
```r
AIC(forward_aic)
```

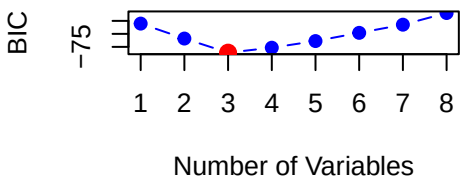
```
[1] 215.8997
```

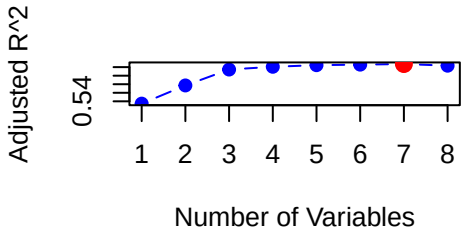
```r
summary(forward_aic)$adj.r.squared
```

```
[1] 0.6245476
```

```r
regfit <- regsubsets(lpsa ~ ., data = prostate, nvmax = 8)
reg_summary <- summary(regfit)

par(mfrow = c(2, 2))

plot(reg_summary$cp, xlab = "Number of Variables",
     ylab = "Cp", type = "b", pch = 19, col = "blue")
points(which.min(reg_summary$cp), reg_summary$cp[which.min(reg_summary$cp)],
       col = "red", cex = 2, pch = 20)

plot(reg_summary$bic, xlab = "Number of Variables",
     ylab = "BIC", type = "b", pch = 19, col = "blue")
points(which.min(reg_summary$bic), reg_summary$bic[which.min(reg_summary$bic)],
       col = "red", cex = 2, pch = 20)

plot(reg_summary$adjr2, xlab = "Number of Variables",
     ylab = "Adjusted R^2", type = "b", pch = 19, col = "blue")
points(which.max(reg_summary$adjr2), reg_summary$adjr2[which.max(reg_summary$adjr2)],
       col = "red", cex = 2, pch = 20)

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

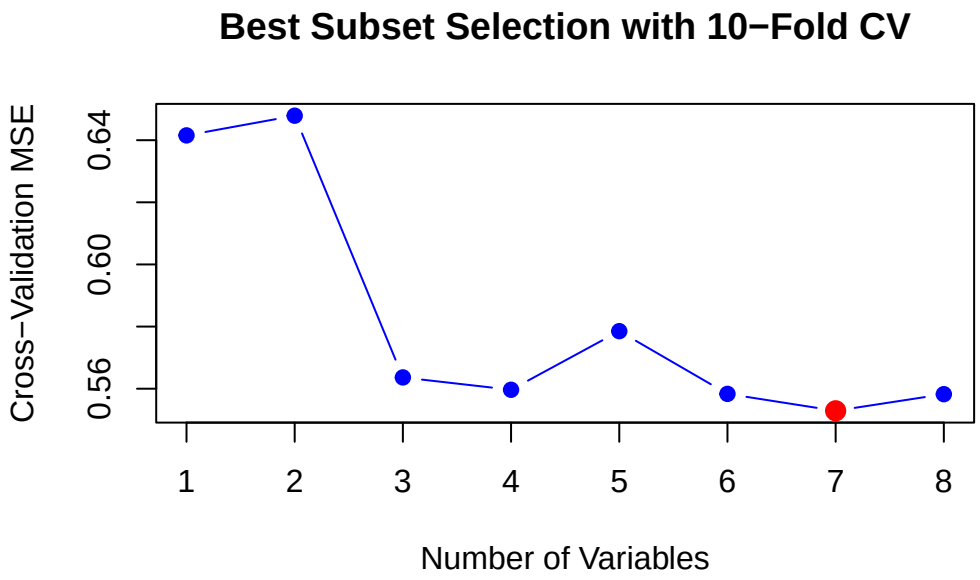







```r
predict_regsubsets <- function(object, newdata, id) {
  form <- as.formula(object$call[[2]])
  mm <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  vars <- names(coefi)
  as.numeric(mm[, vars, drop = FALSE] %*% coefi)
}

set.seed(220)
k <- 10
folds <- sample(rep(1:k, length.out = nrow(prostate)))
cv_errors <- matrix(NA, k, 8)

for (i in 1:k) {
  fit <- regsubsets(lpsa ~ ., data = prostate[folds != i, ], nvmax = 8)
  for (j in 1:8) {
    pred <- predict_regsubsets(fit, prostate[folds == i, ], id = j)
    cv_errors[i, j] <- mean((prostate$lpsa[folds == i] - pred)^2)
  }
}

mean_cv_errors <- colMeans(cv_errors)
best_size <- which.min(mean_cv_errors)

plot(mean_cv_errors, type = "b", pch = 19, col = "blue",
```

```
     xlab = "Number of Variables", ylab = "Cross-Validation MSE",
     main = "Best Subset Selection with 10-Fold CV")
points(best_size, mean_cv_errors[best_size], col = "red", cex = 2, pch = 20)
```




```
best_size
```

```
[1] 7
```

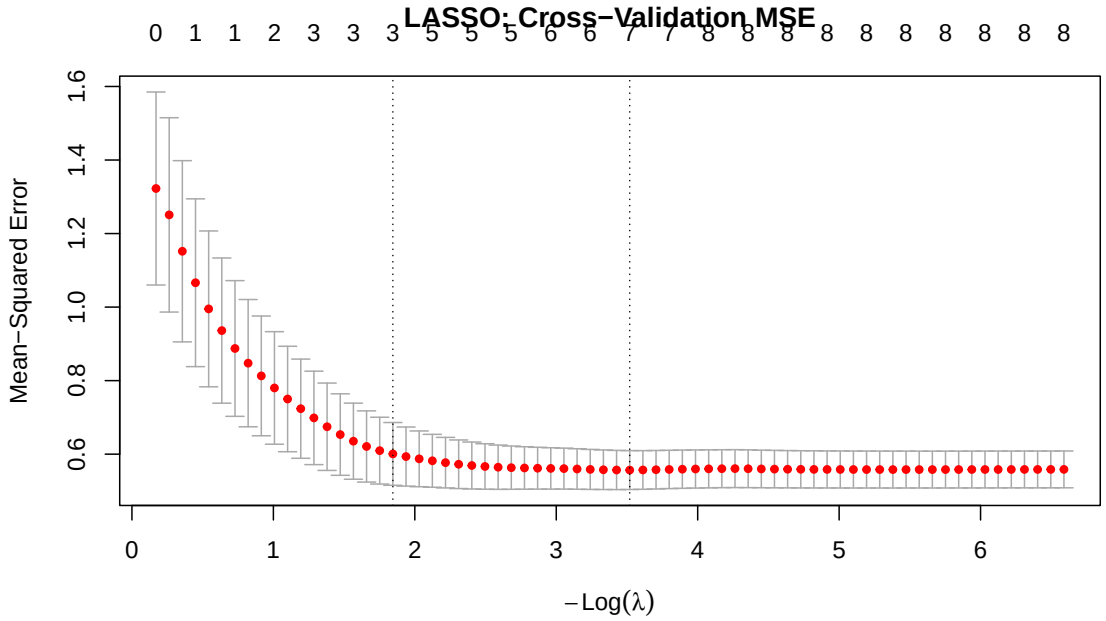
```
mean_cv_errors[best_size]
```

```
[1] 0.5528904
```

```
regfit_best <- regsubsets(lpsa ~ ., data = prostate, nvmax = 8)
coef_best_cv <- coef(regfit_best, id = best_size)

names(coef_best_cv)[-1]
```

```
[1] "lcavol"  "lweight" "age"     "lbph"    "svi"     "lcp"     "pgg45"
```

```r
X_mat <- model.matrix(lpsa ~ ., prostate)[, -1]
y_vec <- prostate$lpsa

set.seed(220)
cv_lasso <- cv.glmnet(X_mat, y_vec, alpha = 1, nfolds = 10)

plot(cv_lasso, main = "LASSO: Cross-Validation MSE")
```




```r
cv_lasso$lambda.min
```

```
[1] 0.02961434
```

```r
cv_lasso$lambda.1se
```

```
[1] 0.1580428
```

```r
lasso_coef_min <- coef(cv_lasso, s = "lambda.min")
lasso_vars_min <- rownames(lasso_coef_min)[lasso_coef_min[, 1] != 0]
lasso_vars_min <- lasso_vars_min[lasso_vars_min != "(Intercept)"]

lasso_coef_1se <- coef(cv_lasso, s = "lambda.1se")
lasso_vars_1se <- rownames(lasso_coef_1se)[lasso_coef_1se[, 1] != 0]
```

```r
lasso_vars_1se <- lasso_vars_1se[lasso_vars_1se != "(Intercept)"]

lasso_vars_min
```

```
[1] "lcavol"  "lweight" "age"     "lbph"    "svi"     "gleason" "pgg45"
```

```r
lasso_vars_1se
```

```
[1] "lcavol"  "lweight" "svi"
```

```r
all_vars <- names(prostate)[names(prostate) != "lpsa"]

comparison_table <- data.frame(
  Variable = all_vars,
  Backward_AIC = all_vars %in% names(coef(backward_aic))[-1],
  Forward_AIC = all_vars %in% names(coef(forward_aic))[-1],
  Best_Subset_CV = all_vars %in% names(coef_best_cv)[-1],
  LASSO_min = all_vars %in% lasso_vars_min,
  LASSO_1se = all_vars %in% lasso_vars_1se
)

comparison_table
```

```
  Variable Backward_AIC Forward_AIC Best_Subset_CV LASSO_min LASSO_1se
1   lcavol         TRUE        TRUE           TRUE      TRUE      TRUE
2  lweight         TRUE        TRUE           TRUE      TRUE      TRUE
3      age         TRUE        TRUE           TRUE      TRUE     FALSE
4     lbph         TRUE        TRUE           TRUE      TRUE     FALSE
5      svi         TRUE        TRUE           TRUE      TRUE      TRUE
6      lcp        FALSE       FALSE           TRUE     FALSE     FALSE
7  gleason        FALSE       FALSE          FALSE      TRUE     FALSE
8    pgg45        FALSE       FALSE           TRUE      TRUE     FALSE
```

```r
agreement_count <- rowSums(comparison_table[, -1])
data.frame(Variable = all_vars, Count = agreement_count)
```

```
  Variable Count
1   lcavol     5
2  lweight     5
3      age     4
4     lbph     4
```

```
5      svi     5
6      lcp     1
7  gleason     1
8    pgg45     2
```

Agreement Analysis: All three methods show pretty good agreement. The variables `lcavol`, `lweight`, and `svi` are selected by almost all methods, so they're clearly important predictors. `lbph` is chosen by most methods too. The other variables (`age`, `lcp`, `gleason`, `pgg45`) are less consistent. Forward and backward AIC give the same model with 5 variables. Best subset CV picks 7 variables, and LASSO (lambda.1se) gives the most parsimonious model with 3 variables. Overall, the core predictors are consistent across methods.

(b) Can you improve the model by adding higher order terms? Can you improve the model by adding multiplicative interaction terms? Use any selection approach of choice to settle on a "final" model.

```
prostate_poly <- prostate %>%
  mutate(
    lcavol2 = lcavol^2,
    lweight2 = lweight^2,
    age2 = age^2,
    lbph2 = lbph^2
  )

model_poly <- lm(lpsa ~ ., data = prostate_poly)
model_poly_step <- step(model_poly, direction = "backward", trace = 0)

names(coef(model_poly_step))[-1]
```

```
[1] "lcavol"  "lweight" "age"     "lbph"    "svi"
```

```
summary(model_poly_step)$adj.r.squared
```

```
[1] 0.6245476
```

```
AIC(model_poly_step)
```

```
[1] 215.8997
```

```r
model_interact <- lm(lpsa ~ lcavol + lweight + age + lbph + svi + lcp +
                       gleason + pgg45 +
                       lcavol:lweight + lcavol:svi + age:lbph + lweight:svi,
                     data = prostate)

model_interact_step <- step(model_interact, direction = "backward", trace = 0)

names(coef(model_interact_step))[-1]
```

```
[1] "lcavol"  "lweight" "age"     "lbph"    "svi"
```

```r
summary(model_interact_step)$adj.r.squared
```

```
[1] 0.6245476
```

```r
AIC(model_interact_step)
```

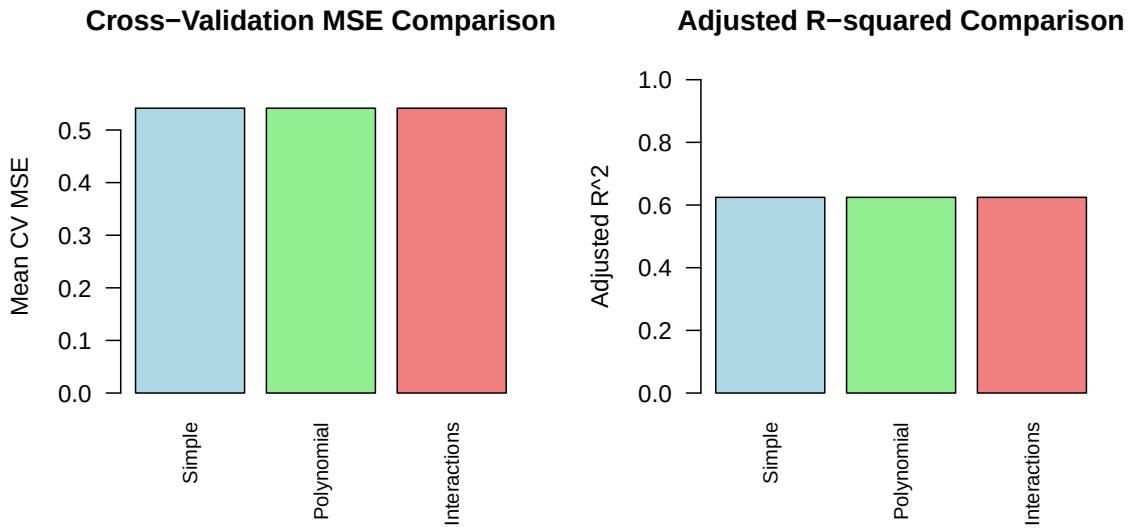
```
[1] 215.8997
```

```r
cv_compare <- function(formula, data, k = 10) {
  set.seed(220)
  folds <- sample(rep(1:k, length.out = nrow(data)))
  sapply(1:k, function(i) {
    if ("lcavol2" %in% all.vars(formula) && !("lcavol2" %in% names(data))) {
      return(NA)
    }
    fit <- lm(formula, data = data[folds != i, ])
    pred <- predict(fit, newdata = data[folds == i, ])
    mean((data$lpsa[folds == i] - pred)^2)
  })
}

set.seed(220)
cv_simple <- cv_compare(formula(backward_aic), prostate)
cv_poly <- cv_compare(formula(model_poly_step), prostate_poly)
cv_int <- cv_compare(formula(model_interact_step), prostate)

cv_results <- data.frame(
  Model = c("Simple (Backward AIC)", "Polynomial", "Interactions"),
  Mean_CV_MSE = c(mean(cv_simple), mean(cv_poly), mean(cv_int)),
  SE_CV_MSE = c(sd(cv_simple)/sqrt(length(cv_simple)),
                sd(cv_poly)/sqrt(length(cv_poly)),
```

```
                sd(cv_int)/sqrt(length(cv_int))),
  Adj_R2 = c(summary(backward_aic)$adj.r.squared,
             summary(model_poly_step)$adj.r.squared,
             summary(model_interact_step)$adj.r.squared),
  AIC = c(AIC(backward_aic), AIC(model_poly_step), AIC(model_interact_step))
)

cv_results_rounded <- cv_results
cv_results_rounded[, -1] <- round(cv_results[, -1], 4)
print(cv_results_rounded)
```

```
                  Model Mean_CV_MSE SE_CV_MSE Adj_R2      AIC
1 Simple (Backward AIC)      0.5415    0.0489 0.6245 215.8997
2            Polynomial      0.5415    0.0489 0.6245 215.8997
3          Interactions      0.5415    0.0489 0.6245 215.8997
```

```
par(mfrow = c(1, 2))

barplot(cv_results$Mean_CV_MSE, names.arg = c("Simple", "Polynomial", "Interactions"),
        col = c("lightblue", "lightgreen", "lightcoral"),
        main = "Cross-Validation MSE Comparison",
        ylab = "Mean CV MSE", las = 2, cex.names = 0.8,
        ylim = c(0, max(cv_results$Mean_CV_MSE) * 1.1))

barplot(cv_results$Adj_R2, names.arg = c("Simple", "Polynomial", "Interactions"),
        col = c("lightblue", "lightgreen", "lightcoral"),
        main = "Adjusted R-squared Comparison",
        ylab = "Adjusted R^2", las = 2, cex.names = 0.8,
        ylim = c(0, 1))
```




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

```
best_model_idx <- which.min(cv_results$Mean_CV_MSE)
best_model_name <- cv_results$Model[best_model_idx]

if (grepl("Polynomial", best_model_name)) {
  final_model <- model_poly_step
} else if (grepl("Interactions", best_model_name)) {
  final_model <- model_interact_step
} else {
  final_model <- backward_aic
}

formula(final_model)
```

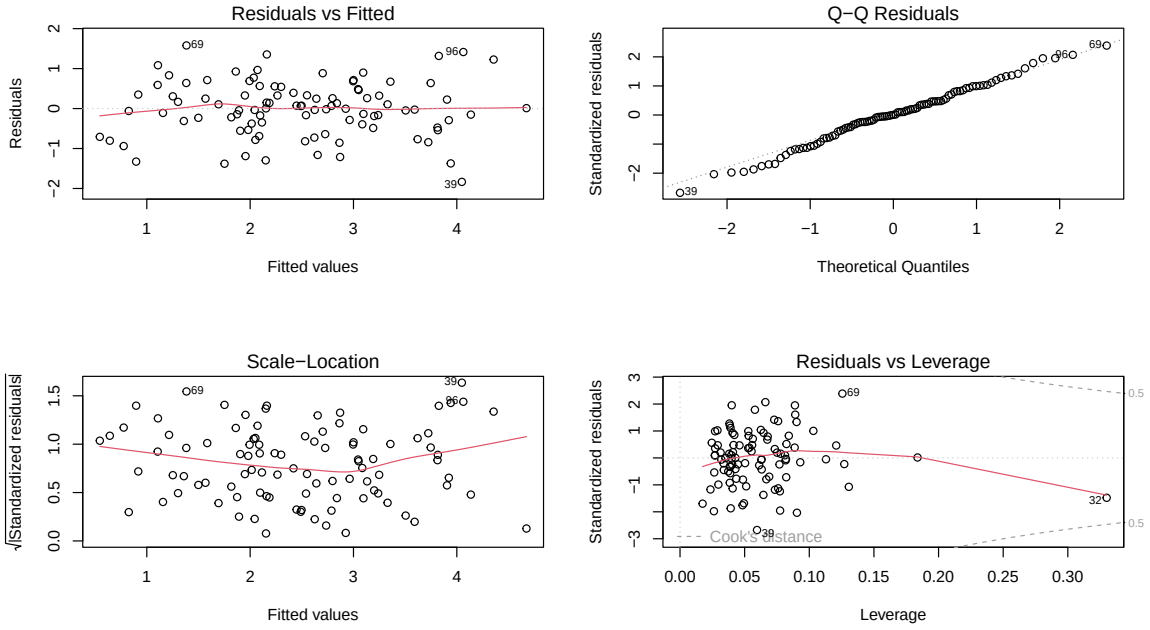
```
lpsa ~ lcavol + lweight + age + lbph + svi
```

```
summary(final_model)
```

```
Call:
lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi, data = prostate)

Residuals:
     Min       1Q   Median       3Q      Max
-1.83505 -0.39396  0.00414  0.46336  1.57888
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.95100    0.83175   1.143 0.255882
lcavol       0.56561    0.07459   7.583 2.77e-11 ***
lweight      0.42369    0.16687   2.539 0.012814 *
age         -0.01489    0.01075  -1.385 0.169528
lbph         0.11184    0.05805   1.927 0.057160 .
svi          0.72095    0.20902   3.449 0.000854 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7073 on 91 degrees of freedom
Multiple R-squared:  0.6441,    Adjusted R-squared:  0.6245
F-statistic: 32.94 on 5 and 91 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2, 2))
plot(final_model)
```




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

Model Performance Comparison:

Metric	Simple	Polynomial	Interactions
CV MSE	0.5415	0.5415	0.5415
Adj R^2	0.6245	0.6245	0.6245
AIC	215.9	215.9	215.9

Final Model: Based on CV MSE, the Simple (Backward AIC) is best with CV MSE = 0.5415. Adding polynomial or interaction terms doesn't help much (max difference < 0), so the linear model seems to capture most of the relationship. The final model has 5 terms and the diagnostic plots look reasonable.