module\_14\_assignment\_rmd

### Problem 1  
df <- MPV::p12.8  
n <- 5  
set.seed(1); chosen\_rows <- sort(sample(seq(1, nrow(df)), n))  
df <- df[chosen\_rows,]  
  
# part (a)  
plot(df$x, df$y)  
f\_0 <- function(x, theta1, theta2) {  
 return(theta1 \* exp(theta2 \* x))  
}  
x\_seq <- seq(0, 10, length.out = 1e3)  
lines(x\_seq, f\_0(x = x\_seq, theta1 = 1, theta2 = 0.5))  
  
# Functions start  
theta1\_deriv\_calc <- function(x, theta2) {  
 return(exp(theta2 \* x))  
}  
theta2\_deriv\_calc <- function(x, theta1, theta2) {  
 return(theta1 \* x \* exp(theta2 \* x))  
}  
cutoff\_fun <- function(theta\_new, theta\_old, delta) {  
 cutoff\_calc <- (theta\_new - theta\_old) / theta\_hat\_old  
   
 # Cutoff when both less than delta  
 if (cutoff\_calc[1] < delta & cutoff\_calc[2] < delta) {  
 return(TRUE)  
 } else {  
 return(c(FALSE, round(cutoff\_calc[1], 4), round(cutoff\_calc[2], 4)))  
 }  
}  
beta\_hat\_calc <- function(x, y, Z, theta1, theta2) {  
 return(solve(t(Z) %\*% Z) %\*% t(Z) %\*%  
 (y - f\_0(x = x, theta1 = theta1, theta2 = theta2)))  
}  
RSS\_calc <- function(y, y\_hat) {  
 return(sum((y\_hat - y)^2))  
}  
# Functions end  
  
# Initialize theta-0  
theta1 <- 1; theta2 <- 0.5  
theta\_vec <- c(theta1, theta2)  
  
# Initialize derivatives, Z  
theta1\_deriv <- theta1\_deriv\_calc(x = df$x, theta2 = theta\_vec[2])  
theta2\_deriv <- theta2\_deriv\_calc(x = df$x,  
 theta1 = theta\_vec[1], theta2 = theta\_vec[2])  
Z <- cbind(theta1\_deriv, theta2\_deriv)  
  
# Initialize beta-hat  
beta\_hat <- solve(t(Z) %\*% Z) %\*% t(Z) %\*% df$y  
  
# Generate theta-hat-1  
theta\_hat\_update <- beta\_hat + theta\_vec  
theta\_hat\_old <- theta\_vec  
  
# Initialize cutoff  
delta <- 1e-6  
cutoff <- cutoff\_fun(theta\_new = theta\_hat\_update,  
 theta\_old = theta\_hat\_old,  
 delta = delta)  
  
# Initial RSS  
y\_hat <- f\_0(x = df$x,  
 theta1 = theta\_hat\_update[1], theta2 = theta\_hat\_update[2])  
RSS <- RSS\_calc(y = df$y, y\_hat = y\_hat)  
  
counter <- 0  
while(!cutoff[1]) {  
 # Update counter  
 counter <- counter + 1  
 print(paste0('Counter: ', counter))  
 # Calculate Z  
 theta1\_deriv <- theta1\_deriv\_calc(x = df$x, theta2 = theta\_hat\_update[2])  
 theta2\_deriv <- theta2\_deriv\_calc(x = df$x,  
 theta1 = theta\_hat\_update[1], theta2 = theta\_hat\_update[2])  
 Z <- cbind(theta1\_deriv, theta2\_deriv)  
   
 # Calculate beta-hat  
 beta\_hat <- beta\_hat\_calc(x = df$x, y = df$y, Z = Z,  
 theta1 = theta\_hat\_update[1], theta2 = theta\_hat\_update[2])  
   
 # theta\_{k+1} = theta\_k + beta\_k  
 theta\_hat\_old <- theta\_hat\_update  
 theta\_hat\_update <- beta\_hat + theta\_hat\_update  
   
 # Calculate cutoff  
 cutoff <- cutoff\_fun(theta\_new = theta\_hat\_update, theta\_old = theta\_hat\_old, delta = delta)  
 print(paste0('Cutoff:', cutoff[2], '; ', cutoff[3]))  
   
 # Print update values  
 print(paste0('theta1:',  
 round(theta\_hat\_update[1], 4),  
 '; theta2',  
 round(theta\_hat\_update[2], 4)))  
   
 # Print RSS  
 y\_hat <- f\_0(x = df$x,  
 theta1 = theta\_hat\_update[1], theta2 = theta\_hat\_update[2])  
 RSS <- RSS\_calc(y = df$y, y\_hat = y\_hat)  
 print(paste0('RSS:', round(RSS, 4)))  
}  
  
plot(df$x, df$y,  
 main = 'Initial estimate and updated estimate of parameters',  
 xlab = 'x', ylab = 'y')  
legend("topleft", legend = c('Initial', 'Update'),  
 lty = c(2,2), col = c('black', 'red'))  
lines(x\_seq, f\_0(x = x\_seq, theta1 = 1, theta2 = 0.5), lty = 2)  
lines(x\_seq, f\_0(x = x\_seq,  
 theta1 = theta\_hat\_update[1],  
 theta2 = theta\_hat\_update[2]),  
 col = 'red', lty = 2)  
  
# part (b)  
SS\_T <- t(df$y) %\*% df$y - ((sum(df$y))^2) / n  
SS\_model <- SS\_T - RSS  
MS\_Res <- RSS / (n - 2)  
  
F\_0 <- (SS\_model / 2) / MS\_Res  
qf(0.95, df1 = 2, df2 = 3)  
  
# part (c)  
cov\_mat <- MS\_Res \* solve(t(Z) %\*% Z)  
sqrt(diag(cov\_mat))  
  
# part (d)  
C <- solve(t(Z) %\*% Z)  
standard\_errors <- sqrt(diag(MS\_Res \* C))  
t\_test\_statics <- theta\_hat\_update / standard\_errors  
alpha <- 0.05  
qt(1 - alpha/2, df = 3)  
  
# part (e)  
e <- df$y - y\_hat  
beta\_hat\_calc <- function(X, y) {  
 X <- as.matrix(X)  
 beta\_hat <- solve(t(X) %\*% X) %\*% t(X) %\*% y  
 return(beta\_hat)  
}  
norm\_prob\_plot <- function(residual\_var, x\_label,  
 main\_title = 'Normal Probability Plot',  
 y\_label = 'Probability', n\_size=n) {  
 ones <- rep(1, n)  
 sorted\_residuals <- sort(residual\_var)  
 cumulative\_probability <- (1:n\_size - 0.5) / n\_size  
 plot(sorted\_residuals, cumulative\_probability, main = main\_title,  
 xlab = x\_label,  
 ylab = y\_label)  
 X\_temp <- cbind(ones, sorted\_residuals)  
 beta\_hat\_temp <- beta\_hat\_calc(X=X\_temp,y=cumulative\_probability)  
 abline(beta\_hat\_temp)  
}  
norm\_prob\_plot(residual\_var = e, x\_label = 'Sorted Residuals')  
order(e, decreasing = FALSE)  
e[order(e, decreasing = FALSE)]  
  
res\_vs\_fitted\_plot <- function(residual\_var,  
 main\_title,  
 y\_label,  
 x\_label = 'Predicted Response',  
 pred\_response = y\_hat) {  
 plot(pred\_response, residual\_var, main = main\_title,  
 xlab = x\_label,  
 ylab = y\_label,  
 ylim = c(min(residual\_var)-sd(residual\_var),  
 max(residual\_var)+sd(residual\_var)))  
}  
res\_vs\_fitted\_plot(residual\_var = e,  
 main\_title = 'Residuals vs. Predicted Response',  
 y\_label = 'Residuals')  
  
  
### Problem 2  
df <- MPV::p12.11  
n <- 14  
set.seed(1); chosen\_rows <- sort(sample(seq(1, nrow(df)), n))  
df <- df[chosen\_rows,]  
  
# part (a)  
plot(df$x, df$y, main = 'Problem 12.11 Y vs. X', xlab = 'x', ylab = 'y')  
  
# part (b)  
f\_0 <- function(x, theta1, theta2, theta3) {  
 return(theta1 - theta2 \* exp(- theta3 \* x))  
}  
  
x\_seq <- seq(0, 40, length.out = 1e3)  
lines(x\_seq, f\_0(x = x\_seq, theta1 = 0.4, theta2 = -0.3, theta3 = 0.15))  
  
# Functions start  
theta1\_deriv\_calc <- function(x) {  
 return(rep(1, length(x)))  
}  
theta2\_deriv\_calc <- function(x, theta3) {  
 return(-exp(-theta3 \* x))  
}  
theta3\_deriv\_calc <- function(x, theta2, theta3) {  
 return(theta2 \* x \* exp(-theta3 \* x))  
}  
cutoff\_fun <- function(theta\_new, theta\_old, delta) {  
 cutoff\_calc <- (theta\_new - theta\_old) / theta\_hat\_old  
   
 # Cutoff when both less than delta  
 if (cutoff\_calc[1] < delta &  
 cutoff\_calc[2] < delta &  
 cutoff\_calc[3] < delta) {  
 return(TRUE)  
 } else {  
 return(c(FALSE,  
 round(cutoff\_calc[1], 4),  
 round(cutoff\_calc[2], 4),  
 round(cutoff\_calc[3], 4)))  
 }  
}  
beta\_hat\_calc <- function(x, y, Z, theta1, theta2, theta3) {  
 return(solve(t(Z) %\*% Z) %\*% t(Z) %\*%  
 (y - f\_0(x = x, theta1 = theta1, theta2 = theta2, theta3 = theta3)))  
}  
RSS\_calc <- function(y, y\_hat) {  
 return(sum((y\_hat - y)^2))  
}  
# Functions end  
  
# Initialize theta-0  
theta1 <- 0.5; theta2 <- -0.3; theta3 <- 0.15  
theta\_vec <- c(theta1, theta2, theta3)  
  
# Initialize derivatives, Z  
theta1\_deriv <- theta1\_deriv\_calc(x = df$x)  
theta2\_deriv <- theta2\_deriv\_calc(x = df$x, theta3 = theta\_vec[3])  
theta3\_deriv <- theta3\_deriv\_calc(x = df$x,  
 theta2 = theta\_vec[2], theta3 = theta\_vec[3])  
  
Z <- cbind(theta1\_deriv, theta2\_deriv, theta3\_deriv)  
  
# Initialize beta-hat  
beta\_hat <- solve(t(Z) %\*% Z) %\*% t(Z) %\*% df$y  
  
# Generate theta-hat-1  
theta\_hat\_update <- beta\_hat + theta\_vec  
theta\_hat\_old <- theta\_vec  
  
# Initialize cutoff  
delta <- 1e-6  
cutoff <- cutoff\_fun(theta\_new = theta\_hat\_update,  
 theta\_old = theta\_hat\_old,  
 delta = delta)  
  
# Initial RSS  
y\_hat <- f\_0(x = df$x,  
 theta1 = theta\_hat\_update[1],  
 theta2 = theta\_hat\_update[2],  
 theta3 = theta\_hat\_update[3])  
RSS <- RSS\_calc(y = df$y, y\_hat = y\_hat)  
  
# Run the while-loop  
counter <- 0  
while(!cutoff[1]) {  
 # Update counter  
 counter <- counter + 1  
 print(paste0('Counter: ', counter))  
 # Calculate Z  
 theta1\_deriv <- theta1\_deriv\_calc(x = df$x)  
 theta2\_deriv <- theta2\_deriv\_calc(x = df$x, theta3 = theta\_hat\_update[3])  
 theta3\_deriv <- theta3\_deriv\_calc(x = df$x,  
 theta2 = theta\_hat\_update[2], theta3 = theta\_hat\_update[3])  
 Z <- cbind(theta1\_deriv, theta2\_deriv, theta3\_deriv)  
   
 # Calculate beta-hat  
 beta\_hat <- beta\_hat\_calc(x = df$x, y = df$y, Z = Z,  
 theta1 = theta\_hat\_update[1],  
 theta2 = theta\_hat\_update[2],  
 theta3 = theta\_hat\_update[3])  
   
 # theta\_{k+1} = theta\_k + beta\_k  
 theta\_hat\_old <- theta\_hat\_update  
 theta\_hat\_update <- beta\_hat + theta\_hat\_update  
   
 # Calculate cutoff  
 cutoff <- cutoff\_fun(theta\_new = theta\_hat\_update, theta\_old = theta\_hat\_old, delta = delta)  
 print(paste0('Cutoff:', cutoff[2], '; ', cutoff[3], '; ', cutoff[3]))  
   
 # Print update values  
 print(paste0('theta1:', round(theta\_hat\_update[1], 4),  
 '; theta2', round(theta\_hat\_update[2], 4),  
 '; theta2', round(theta\_hat\_update[2], 4)))  
   
 # Print RSS  
 y\_hat <- f\_0(x = df$x,  
 theta1 = theta\_hat\_update[1],  
 theta2 = theta\_hat\_update[2],  
 theta3 = theta\_hat\_update[3])  
 RSS <- RSS\_calc(y = df$y, y\_hat = y\_hat)  
 print(paste0('RSS:', round(RSS, 4)))  
}  
  
plot(df$x, df$y,  
 main = 'Initial estimate and updated estimate of parameters',  
 xlab = 'x', ylab = 'y')  
legend("topright", legend = c('Initial', 'Update'),  
 lty = c(2,2), col = c('black', 'red'))  
x\_seq <- seq(0, 40, length.out = 1e3)  
lines(x\_seq, f\_0(x = x\_seq, theta1 = 0.4, theta2 = -0.3, theta3 = 0.15), lty = 2)  
lines(x\_seq, f\_0(x = x\_seq,  
 theta1 = theta\_hat\_update[1],  
 theta2 = theta\_hat\_update[2],  
 theta3 = theta\_hat\_update[3]), lty = 2, col = 'red')  
  
# part (c)  
SS\_T <- t(df$y) %\*% df$y - ((sum(df$y))^2) / n  
SS\_model <- SS\_T - RSS  
MS\_Res <- RSS / (n - 3)  
  
F\_0 <- (SS\_model / 3) / MS\_Res  
qf(0.95, df1 = 3, df2 = n - 3)  
  
# part (d)  
C <- solve(t(Z) %\*% Z)  
standard\_errors <- sqrt(diag(MS\_Res \* C))  
alpha <- 0.05  
t\_value <- qt(1 - alpha / 2, n - 3)  
round(theta\_hat\_update + t\_value \* standard\_errors, 4)  
round(theta\_hat\_update - t\_value \* standard\_errors, 4)  
  
# part (e)  
e <- df$y - y\_hat  
beta\_hat\_calc <- function(X, y) {  
 X <- as.matrix(X)  
 beta\_hat <- solve(t(X) %\*% X) %\*% t(X) %\*% y  
 return(beta\_hat)  
}  
norm\_prob\_plot <- function(residual\_var, x\_label,  
 main\_title = 'Normal Probability Plot',  
 y\_label = 'Probability', n\_size=n) {  
 ones <- rep(1, n)  
 sorted\_residuals <- sort(residual\_var)  
 cumulative\_probability <- (1:n\_size - 0.5) / n\_size  
 plot(sorted\_residuals, cumulative\_probability, main = main\_title,  
 xlab = x\_label,  
 ylab = y\_label)  
 X\_temp <- cbind(ones, sorted\_residuals)  
 beta\_hat\_temp <- beta\_hat\_calc(X=X\_temp,y=cumulative\_probability)  
 abline(beta\_hat\_temp)  
}  
norm\_prob\_plot(residual\_var = e, x\_label = 'Sorted Residuals')  
order(e, decreasing = FALSE)  
e[order(e, decreasing = FALSE)]  
  
res\_vs\_fitted\_plot <- function(residual\_var,  
 main\_title,  
 y\_label,  
 x\_label = 'Predicted Response',  
 pred\_response = y\_hat) {  
 plot(pred\_response, residual\_var, main = main\_title,  
 xlab = x\_label,  
 ylab = y\_label,  
 ylim = c(min(residual\_var)-sd(residual\_var),  
 max(residual\_var)+sd(residual\_var)))  
}  
res\_vs\_fitted\_plot(residual\_var = e,  
 main\_title = 'Residuals vs. Predicted Response',  
 y\_label = 'Residuals')