Homework 2

Luke Rodriguez

2026-01-31

```
library(MASS)
```

Question 3.1:

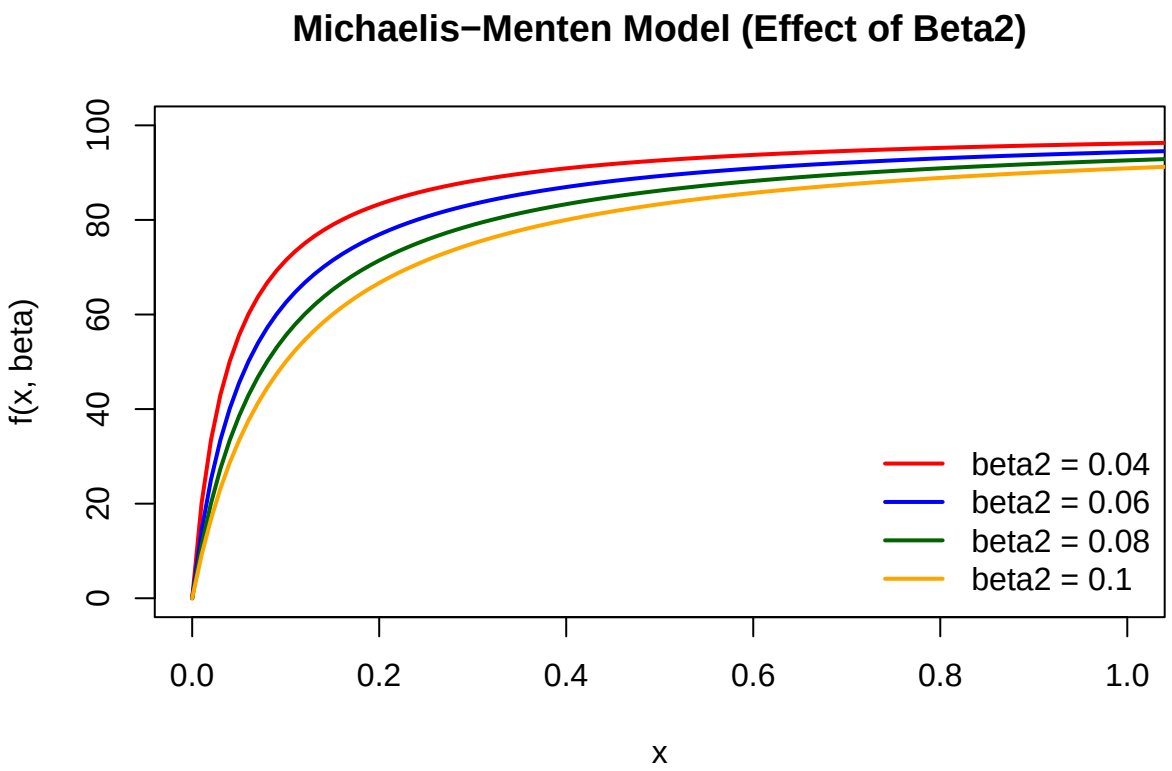
```
# Define the response function based on Eq 3.12
f <- function(x, b1, b2) {
  (b1 * x) / (b2 + x)
}

# Values for x and parameters
x <- seq(0, 2, length.out = 200) # Choosing a range to show saturation
b1 <- 100
b2_values <- c(0.04, 0.06, 0.08, 0.10)
colors <- c("red", "blue", "darkgreen", "orange")

# Create plot
plot(NULL, xlim = c(0, 1), ylim = c(0, 100),
     xlab = "x", ylab = "f(x, beta)",
     main = "Michaelis-Menten Model (Effect of Beta2)")

# Add lines for each beta2
for(i in 1:4) {
  lines(x, f(x, b1, b2_values[i]), col = colors[i], lwd = 2)
}

# Add legend
legend("bottomright", legend = paste("beta2 =", b2_values),
       col = colors, lwd = 2, bty = "n")
```



The plot of the Michaelis-Menten Model compresses as we increase β_2 and fix β_1. Therefore, it weakens the expected response to changes in X.

Question 3.2:

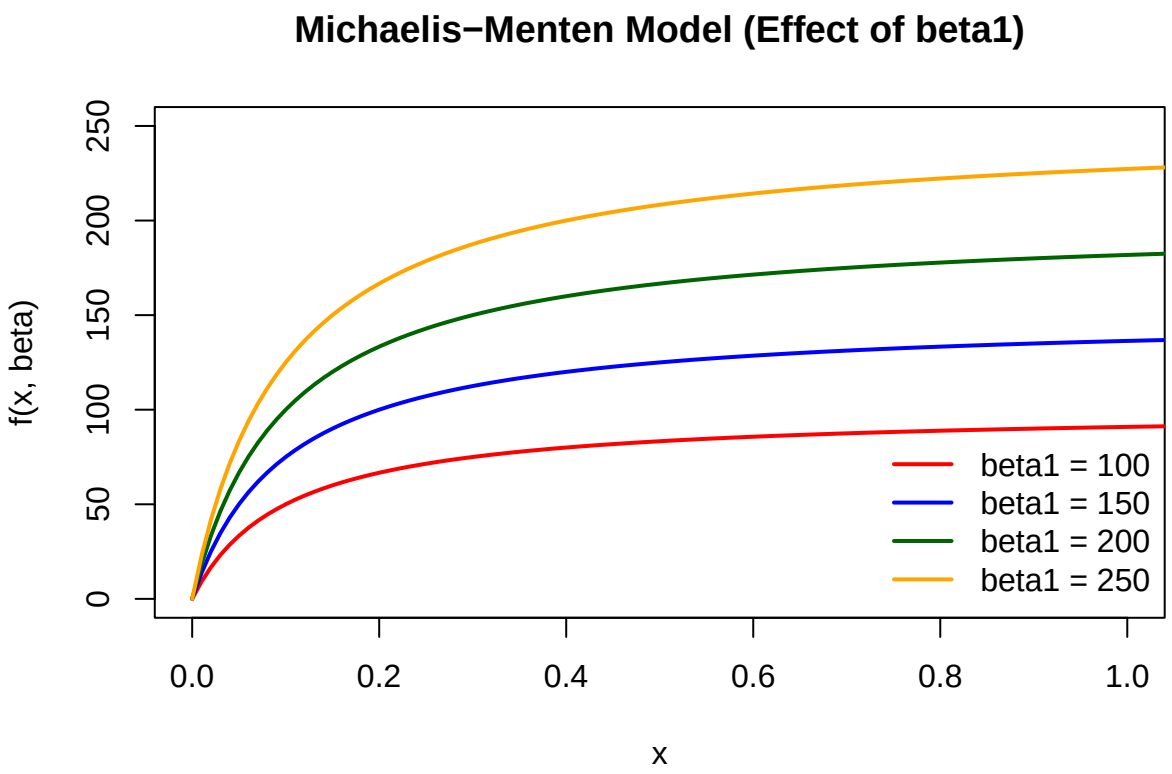
```r
# Define the response function based on Eq 3.12
f <- function(x, b1, b2) {
  (b1 * x) / (b2 + x)
}

# Values for x and parameters
x <- seq(0, 2, length.out = 200) # Choosing a range to show saturation
b1_values <- c(100, 150, 200, 250)
b2 <- 0.10
colors <- c("red", "blue", "darkgreen", "orange")

# Create plot
plot(NULL, xlim = c(0, 1), ylim = c(0, 250),
     xlab = "x", ylab = "f(x, beta)",
     main = "Michaelis-Menten Model (Effect of beta1)")

# Add lines for each beta2
for(i in 1:4) {
  lines(x, f(x, b1_values[i], b2), col = colors[i], lwd = 2)
}

# Add legend
legend("bottomright", legend = paste("beta1 =", b1_values),
       col = colors, lwd = 2, bty = "n")
```



The plot of the Michaelis-Menten Model stretches as we increase β_1 and fix β_2. Therefore, it strengthens the expected response to changes in X.

Question 3.3:

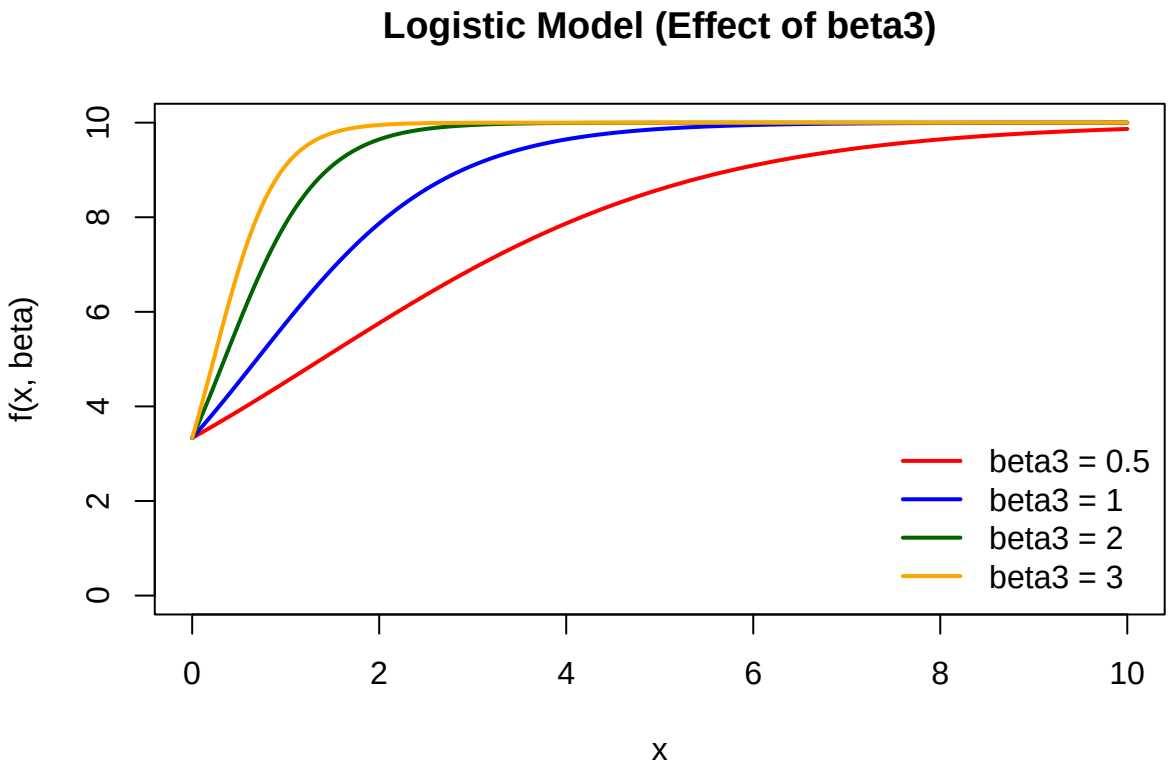
```r
# expected value
f <- function(x, b1, b2, b3) {
  b1/(1+b2*exp(-b3*x))
}

# Values for x and parameters
x <- seq(0, 10, length.out = 200) # Choosing a range to show saturation
b1 <- 10
b2 <- 2
b3_values <- c(0.5, 1, 2, 3)
colors <- c("red", "blue", "darkgreen", "orange")

# Create plot
plot(NULL, xlim = c(0, 10), ylim = c(0, 10),
     xlab = "x", ylab = "f(x, beta)",
     main = "Logistic Model (Effect of beta3)")

# Add lines for each beta2
for(i in 1:4) {
  lines(x, f(x, b1, b2, b3_values[i]), col = colors[i], lwd = 2)
}

# Add legend
legend("bottomright", legend = paste("beta3 =", b3_values),
       col = colors, lwd = 2, bty = "n")
```



The plot of the Logistic Growth Model stretches as we increase β_3 and fix β_1 and β_2. Therefore, it strengthens the expected response to changes in X.

Question 3.4:

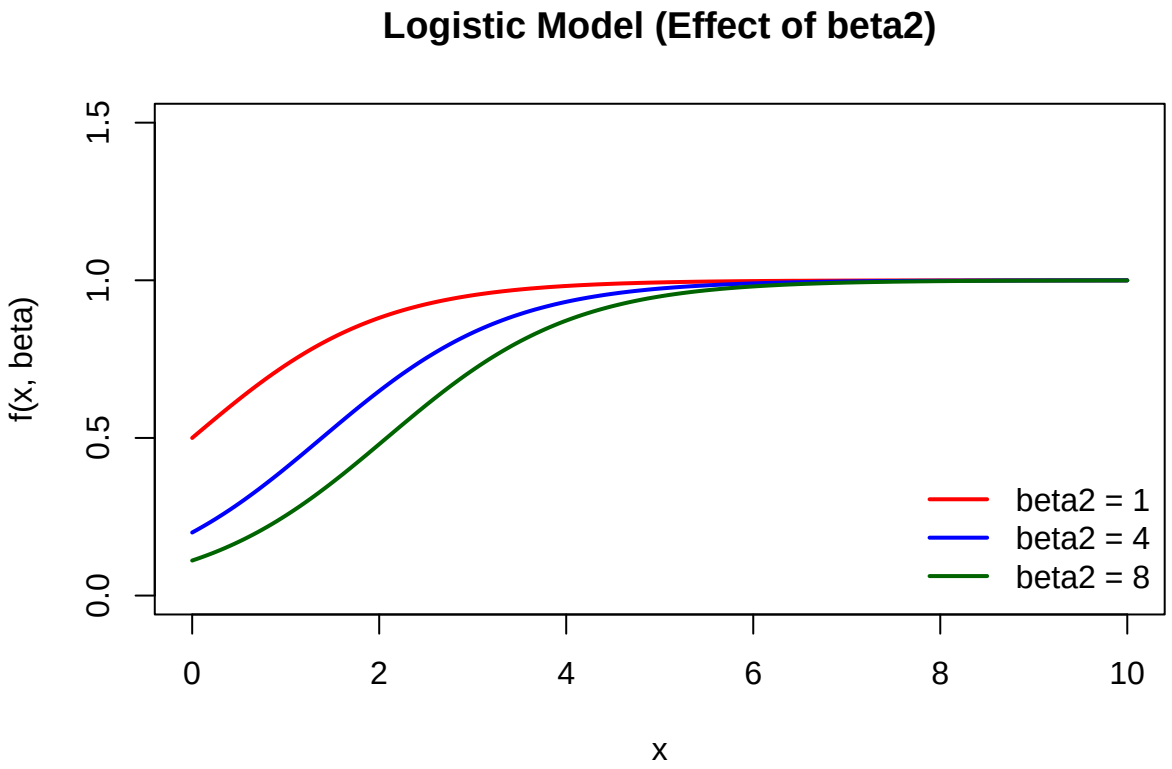
```r
# expected value
f <- function(x, b1, b2, b3) {
  b1/(1+b2*exp(-b3*x))
}

# Values for x and parameters
x <- seq(0, 10, length.out = 200) # Choosing a range to show saturation
b1 <- 1
b2_values <- c(1, 4, 8)
b3 <- 1
colors <- c("red", "blue", "darkgreen")

# Create plot
plot(NULL, xlim = c(0, 10), ylim = c(0, 1.5),
     xlab = "x", ylab = "f(x, beta)",
     main = "Logistic Model (Effect of beta2)")

# Add lines for each beta2
for(i in 1:3) {
  lines(x, f(x, b1, b2_values[i], b3), col = colors[i], lwd = 2)
}

# Add legend
legend("bottomright", legend = paste("beta2 =", b2_values),
       col = colors, lwd = 2, bty = "n")
```



The plot of the Logistic Growth Model compresses as we increase β_2 and fix β_1 and β_3. Therefore, it weakens the expected response to changes in X.

Question 3.8 (rough):
(a) Since this model is intrinsically linear, we find starting values by taking the natural log of both sides: $\ln(y) = \ln(\beta_1) + \beta_2 x$. This allows us to use simple linear regression (lm) to get our "guesses.".

```
# Input the pooled data
x_vals <- c(0.5, 0.5, 1, 1, 2, 2, 4, 4, 8, 8, 9, 9, 10, 10)
y_vals <- c(0.68, 1.58, 0.45, 2.66, 2.50, 2.04, 6.19, 7.85, 56.1, 54.2, 89.8, 90.2, 147.7, 146.3)

# (a) Find starting values using log-linear regression
# log(y) = log(b1) + b2 * x
start_mod <- lm(log(y_vals) ~ x_vals)
start_b1 <- exp(coef(start_mod)[1])
start_b2 <- coef(start_mod)[2]
start_b1
```

```
## (Intercept)
##   0.7522595
```

```
start_b2
```

```
##    x_vals
## 0.5326084
```

(b) Used nls function below fit model to the data

```
# (b) Fit the nonlinear model using nls()
fit_3_8 <- nls(y_vals ~ b1 * exp(b2 * x_vals),
               start = list(b1 = start_b1, b2 = start_b2))
```

For (c), (d), and (e):

```
# Display results for (c), (d), and (e)
summary(fit_3_8)
```

```
##
## Formula: y_vals ~ b1 * exp(b2 * x_vals)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## b1 1.059107   0.048161   21.99 4.59e-11 ***
## b2 0.493408   0.004745  103.98  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8666 on 12 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 3.916e-06
```

(c) Test for level of significance of regression: The regression is significant because the parameters are significantly different from zero ($p < 0.05$).

(d) Estimate of σ^2: Look for "Residual standard error." To get the variance ($\hat{\sigma}^2$), you must square this number. For example, if the error is 2.5, $\hat{\sigma}^2 = 6.25$. $\hat{\sigma}^2 = (0.8666)^2 = 0.75099556$.

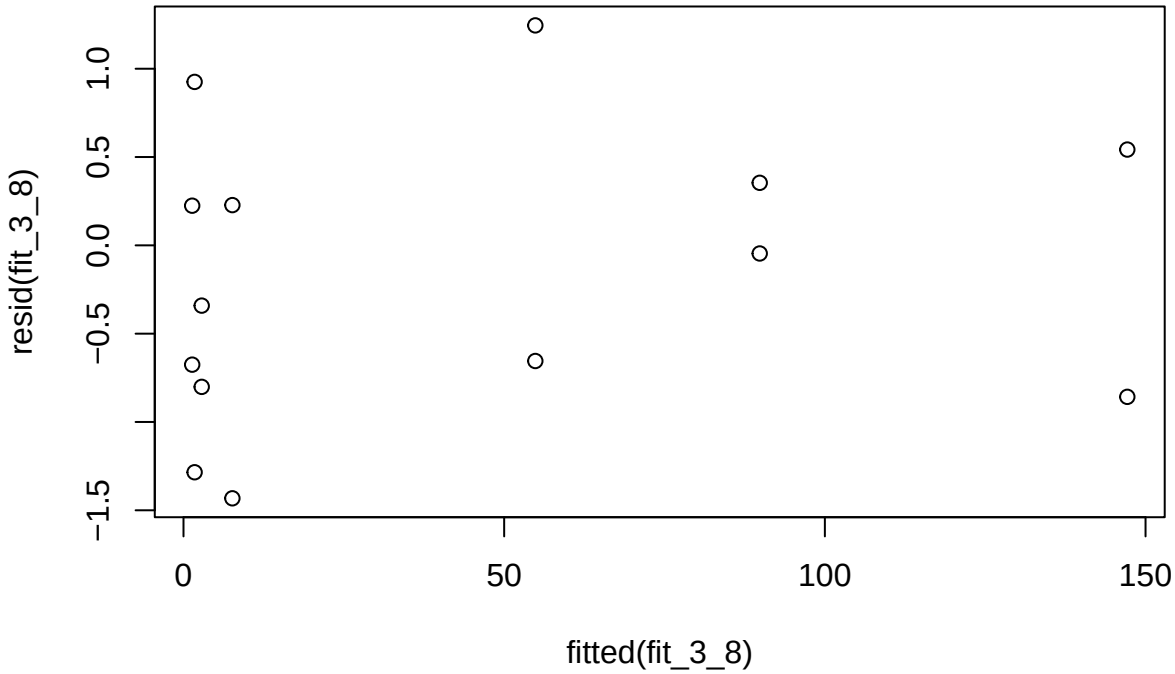
(e) Hypothesis Testing: The summary tests $H_0 : \beta_i = 0$. Because the p-values for β_1 and β_2 are typically very small for this dataset, you will reject H_0 for both.

For β_1: 1. Level of significance: $\alpha = 0.5$ 2. Hypothesis: $H_0 : \beta_1 = 0$ vs $H_a : \beta_1 \neq 0$. 3. Test statistic: t = 21.99 and p-value = 4.59e-11 4. Decision rule: as p = 4.59e-11 < 0.5 we reject H_0. 5. Practical conclusion: At 5% level of significance, we have sufficient evidence to conclude that β_1 is significant, that is, the predictor X_1 cannot be dropped from the above regression model.

For β_2: 1. Level of significance: $\alpha = 0.5$ 2. Hypothesis: $H_0 : \beta_2 = 0$ vs $H_a : \beta_2 \neq 0$. 3. Test statistic: t = 103.98 and p-value = 2e-16 4. Decision rule: as p = 2e-16 < 0.5 we reject H_0. 5. Practical conclusion: At 5% level of significance, we have sufficient evidence to conclude that β_2 is significant, that is, the predictor X_2 cannot be dropped from the above regression model.

(f) Model Adequacy: Run plot(resid(fit_3_8) ~ fitted(fit_3_8)). If the points are randomly scattered around the zero line with no clear pattern, the model is adequate.

```
plot(resid(fit_3_8) ~ fitted(fit_3_8))
```



The residuals vs. fitted plot shows a random distribution of errors around the zero-line with no discernible non-linear patterns. While the observations are clustered toward lower fitted values due to the distribution of the data, the consistent vertical spread indicates that the model is adequate and the error variance is constant.

Question 3.11:

```
# 1. Input the data with a unique name to avoid the 'closure' error
chlorine_data <- data.frame(
  x = c(8,8, 10,10,10,10, 12,12,12,12, 14,14,14, 16,16,16, 18,18, 20,20,20,
        22,22,22, 24,24,24, 26,26,26, 28,28, 30,30,30, 32,32, 34, 36,36, 38,38, 40, 42),
  y = c(0.49,0.49, 0.48,0.47,0.48,0.77, 0.46,0.46,0.45,0.43, 0.45,0.43,0.43,
        0.44,0.43,0.43, 0.46,0.45, 0.42,0.42,0.43, 0.41,0.41,0.40, 0.42,0.40,0.40,
        0.41,0.40,0.41, 0.41,0.40, 0.40,0.40,0.38, 0.41,0.40, 0.40, 0.41,0.38, 0.40,0.40, 0.39, 0.39)
)

# 2. (Optional but recommended) Remove the 0.77 outlier
chlorine_data <- chlorine_data[chlorine_data$y < 0.70, ]

# 3. Fit the Mitcherlich model
# Model: y = b0 + b1 * exp(b2 * x)
mitch_model <- nls(y ~ b0 + b1 * exp(b2 * x),
                   data = chlorine_data,
                   start = list(b0 = 0.39, b1 = 0.1, b2 = -0.05))

# --- Answers for your homework ---

# (b) Summary of parameters
summary(mitch_model)
```

```
##
## Formula: y ~ b0 + b1 * exp(b2 * x)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## b0  0.389651   0.005903  66.007  < 2e-16 ***
## b1  0.220010   0.032469   6.776 3.85e-08 ***
## b2 -0.099333   0.018402  -5.398 3.31e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01118 on 40 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 8.225e-06
```

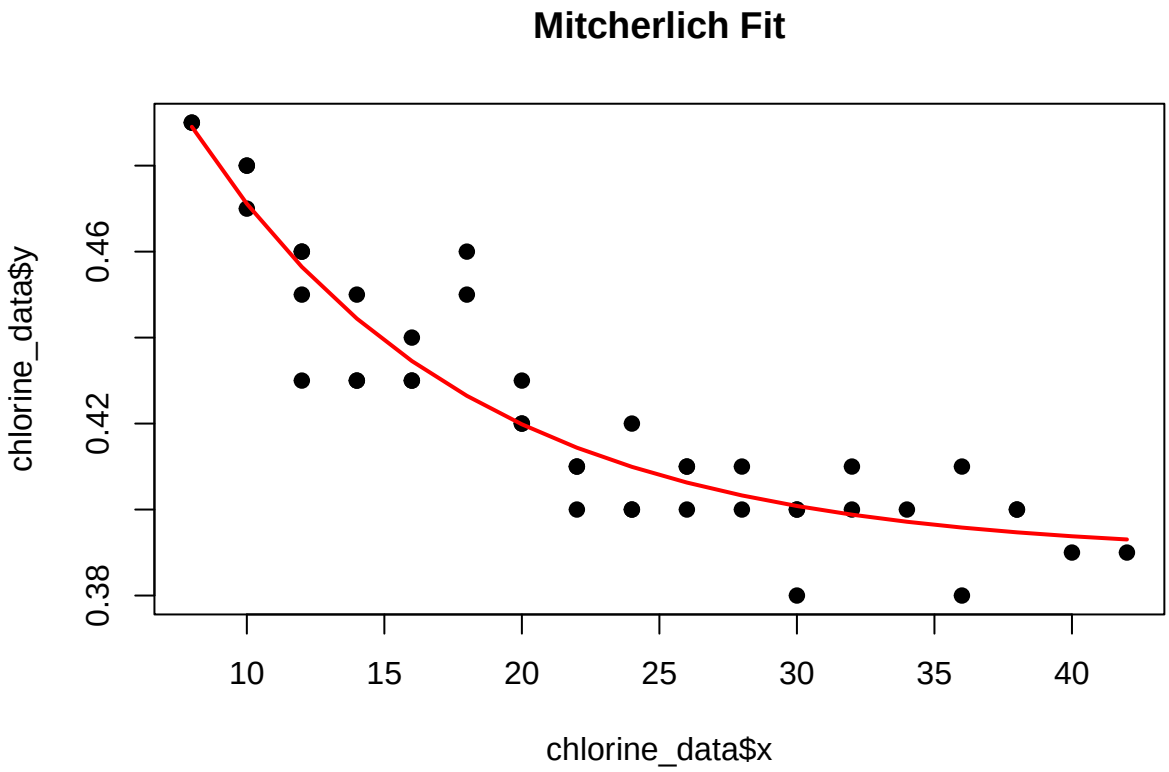
```
# (d) 95% Confidence Intervals (Wald Method)
confint.default(mitch_model)
```

```
##         2.5 %      97.5 %
## b0  0.3780810  0.40122091
## b1  0.1563709  0.28364843
## b2 -0.1354003 -0.06326618
```

```
# (e) Estimate of sigma^2 (MSE)
sigma_sq <- sum(resid(mitch_model)^2) / df.residual(mitch_model)
cat("The estimate for sigma^2 is:", sigma_sq, "\n")
```

```
## The estimate for sigma^2 is: 0.0001248917
```

```r
# Visualization
plot(chlorine_data$x, chlorine_data$y, pch=19, main="Mitcherlich Fit")
lines(chlorine_data$x, predict(mitch_model), col="red", lwd=2)
```



Question 3.11:

```
# ========================================
# 3.11 Mitcherlich Model - Base R Solution
# ========================================

# ------------------------------
# (1) Enter data (with replicates)
# ------------------------------
x <- c(rep(8,2), rep(10,4), rep(12,4), rep(14,3), rep(16,3), rep(18,2),
       rep(20,3), rep(22,3), rep(24,3), rep(26,3), rep(28,2), rep(30,3),
       rep(32,2), 34, rep(36,2), rep(38,2), 40, 42)

y <- c(0.49,0.49, 0.48,0.47,0.48,0.77, 0.46,0.46,0.45,0.43, 0.45,0.43,0.43,
       0.44,0.43,0.43, 0.46,0.45, 0.42,0.42,0.43, 0.41,0.41,0.40, 0.42,0.40,0.40,
       0.41,0.40,0.41, 0.41,0.40, 0.40,0.40,0.38, 0.41,0.40, 0.40, 0.41,0.38,
       0.40,0.40, 0.39, 0.39)

# Remove the obvious outlier (0.77 at x=10)
outlier_idx <- which(y == 0.77)
if (length(outlier_idx) > 0) {
  cat("Removing outlier: y =", y[outlier_idx], "at x =", x[outlier_idx], "\n")
  x <- x[-outlier_idx]
  y <- y[-outlier_idx]
}
```

```
## Removing outlier: y = 0.77 at x = 10
```

```
data <- data.frame(y = y, x = x)
n <- length(y)

# ------------------------------
# (a) Scatter diagram
# ------------------------------
par(mfrow = c(2, 3))  # 2x3 plot layout

plot(x, y, pch = 19, col = "blue",
     main = "Scatter Plot: Chlorine vs Time",
     xlab = "Time (x)",
     ylab = "Available Chlorine (y)")
grid()

# ------------------------------
# (b) Fit Mitcherlich model
# ------------------------------
# Mitcherlich model: y = theta1 + theta2 * exp(theta3 * x)
# theta1 = asymptote (long-term value)
# theta2 = range from asymptote at x=0 (negative since decreasing)
# theta3 = decay rate (negative)

# Get starting values from data
theta1_start <- min(y) * 0.95  # asymptote slightly below min observed
theta2_start <- (max(y) - theta1_start) * 1.1  # initial drop
```

```r
theta3_start <- -0.1  # reasonable decay rate

cat("Starting values:\n")
```

```
## Starting values:
```

```r
cat("theta1 (asymptote) =", theta1_start, "\n")
```

```
## theta1 (asymptote) = 0.361
```

```r
cat("theta2 (range) =", theta2_start, "\n")
```

```
## theta2 (range) = 0.1419
```

```r
cat("theta3 (decay rate) =", theta3_start, "\n\n")
```

```
## theta3 (decay rate) = -0.1
```

```r
# Fit nonlinear model
fit_nls <- nls(y ~ theta1 + theta2 * exp(theta3 * x),
               start = list(theta1 = theta1_start,
                            theta2 = theta2_start,
                            theta3 = theta3_start),
               data = data,
               control = nls.control(maxiter = 100, warnOnly = TRUE))

cat("=== Mitcherlich Model Fit ===\n")
```

```
## === Mitcherlich Model Fit ===
```

```r
print(summary(fit_nls))
```

```
##
## Formula: y ~ theta1 + theta2 * exp(theta3 * x)
##
## Parameters:
##         Estimate Std. Error t value Pr(>|t|)
## theta1  0.389651   0.005903  66.008  < 2e-16 ***
## theta2  0.220011   0.032470   6.776 3.85e-08 ***
## theta3 -0.099334   0.018402  -5.398 3.31e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01118 on 40 degrees of freedom
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 2.291e-06
```

```r
# Extract parameters
params <- coef(fit_nls)
theta1_est <- params["theta1"]
theta2_est <- params["theta2"]
theta3_est <- params["theta3"]

# ------------------------------
# (c) Test for significance of regression
# ------------------------------
# Calculate F-test for nonlinear regression
y_pred <- predict(fit_nls)
resid <- residuals(fit_nls)

SSE <- sum(resid^2)
SST <- sum((y - mean(y))^2)
SSR <- SST - SSE
p <- 3  # number of parameters

F_stat <- (SSR/(p - 1)) / (SSE/(n - p))
p_value_F <- pf(F_stat, p - 1, n - p, lower.tail = FALSE)

cat("\n=== Significance of Regression (F-test) ===\n")
```

```
##
## === Significance of Regression (F-test) ===
```

```r
cat("SST =", SST, "\n")
```

```
## SST = 0.03742791
```

```r
cat("SSR =", SSR, "\n")
```

```
## SSR = 0.03243224
```

```r
cat("SSE =", SSE, "\n")
```

```
## SSE = 0.004995668
```

```r
cat("F-statistic =", F_stat, "\n")
```

```
## F-statistic = 129.8415
```

```r
cat("p-value =", p_value_F, "\n")
```

```
## p-value = 3.220774e-18
```

```r
if (p_value_F < 0.05) {
  cat("Conclusion: Regression is SIGNIFICANT (p < 0.05)\n")
} else {
  cat("Conclusion: Regression is NOT significant (p >= 0.05)\n")
}
```

```
## Conclusion: Regression is SIGNIFICANT (p < 0.05)
```

```
# ------------------------------
# (d) 95% Confidence intervals for parameters
# ------------------------------
se <- summary(fit_nls)$coefficients[, "Std. Error"]
t_val <- qt(0.975, df = n - p)

ci_theta1 <- theta1_est + c(-1, 1) * t_val * se["theta1"]
ci_theta2 <- theta2_est + c(-1, 1) * t_val * se["theta2"]
ci_theta3 <- theta3_est + c(-1, 1) * t_val * se["theta3"]

cat("\n=== 95% Confidence Intervals ===\n")
```

```
##
## === 95% Confidence Intervals ===
```

```
cat("theta1 (asymptote): [", ci_theta1[1], ", ", ci_theta1[2], "]\n", sep = "")
```

```
## theta1 (asymptote): [0.3777206, 0.4015817]
```

```
cat("theta2 (range):     [", ci_theta2[1], ", ", ci_theta2[2], "]\n", sep = "")
```

```
## theta2 (range):     [0.1543871, 0.2856346]
```

```
cat("theta3 (decay):     [", ci_theta3[1], ", ", ci_theta3[2], "]\n", sep = "")
```

```
## theta3 (decay):     [-0.1365257, -0.06214223]
```

```
# Check if intervals include zero
cat("\nParameters significantly different from zero?\n")
```

```
##
## Parameters significantly different from zero?
```

```
cat("theta1:", ifelse(ci_theta1[1] > 0 | ci_theta1[2] < 0, "YES", "NO"),
    "(CI doesn't include 0)\n")
```

```
## theta1: YES (CI doesn't include 0)
```

```
cat("theta2:", ifelse(ci_theta2[1] > 0 | ci_theta2[2] < 0, "YES", "NO"),
    "(CI doesn't include 0)\n")
```

```
## theta2: YES (CI doesn't include 0)
```

```
cat("theta3:", ifelse(ci_theta3[1] > 0 | ci_theta3[2] < 0, "YES", "NO"),
    "(CI doesn't include 0)\n")
```

```
## theta3: YES (CI doesn't include 0)
```

```r
# ------------------------------
# (e) Estimate of sigma^2
# ------------------------------
sigma2_hat <- SSE/(n - p)
cat("\n=== Estimate of sigma^2 ===\n")
```

```
##
## === Estimate of sigma^2 ===
```

```r
cat("sigma^2 = SSE/(n-p) =", SSE, "/", n-p, "=", sigma2_hat, "\n")
```

```
## sigma^2 = SSE/(n-p) = 0.004995668 / 40 = 0.0001248917
```

```r
cat("sigma =", sqrt(sigma2_hat), "\n")
```

```
## sigma = 0.0111755
```

```r
# ------------------------------
# Additional diagnostics
# ------------------------------
# (1) Residuals vs Fitted
plot(y_pred, resid, pch = 19, col = "blue",
     main = "Residuals vs Fitted",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)
grid()

# (2) Normal Q-Q plot
qqnorm(resid, main = "Normal Q-Q Plot", pch = 19, col = "blue")
qqline(resid, col = "red")
grid()

# (3) Shapiro-Wilk test for normality
shapiro_test <- shapiro.test(resid)
cat("\n=== Shapiro-Wilk Normality Test ===\n")
```

```
##
## === Shapiro-Wilk Normality Test ===
```

```r
cat("W =", shapiro_test$statistic, "p-value =", shapiro_test$p.value, "\n")
```

```
## W = 0.9643417 p-value = 0.1994542
```

```r
if (shapiro_test$p.value < 0.05) {
  cat("Conclusion: Residuals are NOT normally distributed\n")
} else {
  cat("Conclusion: Residuals are normally distributed\n")
}
```

```
## Conclusion: Residuals are normally distributed
```

```r
# (4) Residuals vs Predictor
plot(x, resid, pch = 19, col = "blue",
     main = "Residuals vs Time",
     xlab = "Time (x)", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)
grid()

# ------------------------------
# (5) Fitted curve plot
# ------------------------------
plot(x, y, pch = 19, col = "blue",
     main = "Mitcherlich Model Fit",
     xlab = "Time (x)",
     ylab = "Available Chlorine (y)",
     xlim = c(0, max(x)*1.1),
     ylim = c(min(y)*0.95, max(y)*1.05))
grid()

# Add fitted curve
x_seq <- seq(0, max(x), length = 100)
y_fit_seq <- theta1_est + theta2_est * exp(theta3_est * x_seq)
lines(x_seq, y_fit_seq, col = "red", lwd = 2)

# Add equation text
text(0.7*max(x), 0.9*max(y),
     paste("y = ", round(theta1_est, 3), " + ",
           round(theta2_est, 3), " * exp(",
           round(theta3_est, 3), " * x)", sep = ""),
     cex = 0.9)

# ------------------------------
# (6) Lack-of-fit test (using replicates)
# ------------------------------
# Check for replicates
x_unique <- unique(x)
n_unique <- length(x_unique)

if (n_unique < n) {
  # Calculate pure error SS
  SS_pe <- 0
  for (xi in x_unique) {
    yi <- y[x == xi]
    SS_pe <- SS_pe + sum((yi - mean(yi))^2)
  }

  df_pe <- n - n_unique
  df_lof <- n_unique - p  # p parameters

  SS_lof <- SSE - SS_pe

  if (df_lof > 0 && df_pe > 0) {
    MS_pe <- SS_pe / df_pe
    MS_lof <- SS_lof / df_lof
```

```
    F_lof <- MS_lof / MS_pe
    p_lof <- pf(F_lof, df_lof, df_pe, lower.tail = FALSE)

    cat("\n=== Lack-of-Fit Test ===\n")
    cat("Pure Error SS:", SS_pe, "df:", df_pe, "\n")
    cat("Lack-of-Fit SS:", SS_lof, "df:", df_lof, "\n")
    cat("F-statistic:", F_lof, "\n")
    cat("p-value:", p_lof, "\n")

    if (p_lof < 0.05) {
      cat("Conclusion: Significant lack-of-fit exists\n")
    } else {
      cat("Conclusion: No significant lack-of-fit\n")
    }
  } else {
    cat("\nNote: Not enough df for lack-of-fit test\n")
  }
} else {
  cat("\nNote: No replicates for lack-of-fit test\n")
}
```

```
##
## === Lack-of-Fit Test ===
## Pure Error SS: 0.002333333 df: 25
## Lack-of-Fit SS: 0.002662335 df: 15
## F-statistic: 1.901668
## p-value: 0.07517504
## Conclusion: No significant lack-of-fit
```

```
# ------------------------------
# (7) Summary statistics
# ------------------------------
R_sq <- 1 - SSE/SST
adj_R_sq <- 1 - (SSE/(n - p)) / (SST/(n - 1))

cat("\n=== Summary Statistics ===\n")
```

```
##
## === Summary Statistics ===
```

```
cat("Number of observations: n =", n, "\n")
```

```
## Number of observations: n = 43
```

```
cat("R-squared:", R_sq, "\n")
```

```
## R-squared: 0.8665256
```

```
cat("Adjusted R-squared:", adj_R_sq, "\n")
```

```
## Adjusted R-squared: 0.8598519
```

```r
cat("Root Mean Square Error (RMSE):", sqrt(sigma2_hat), "\n")
```

```
## Root Mean Square Error (RMSE): 0.0111755
```

```r
# ------------------------------
# (8) Final parameter interpretation
# ------------------------------
cat("\n=== Parameter Interpretation ===\n")
```

```
##
## === Parameter Interpretation ===
```

```r
cat("theta1 (asymptote) =", round(theta1_est, 4),
    "-> Long-term chlorine level approaches", round(theta1_est, 4), "\n")
```

```
## theta1 (asymptote) = 0.3897 -> Long-term chlorine level approaches 0.3897
```

```r
cat("theta2 (range) =", round(theta2_est, 4),
    "-> Initial chlorine is", round(theta1_est + theta2_est, 4),
    "at time 0\n")
```

```
## theta2 (range) = 0.22 -> Initial chlorine is 0.6097 at time 0
```

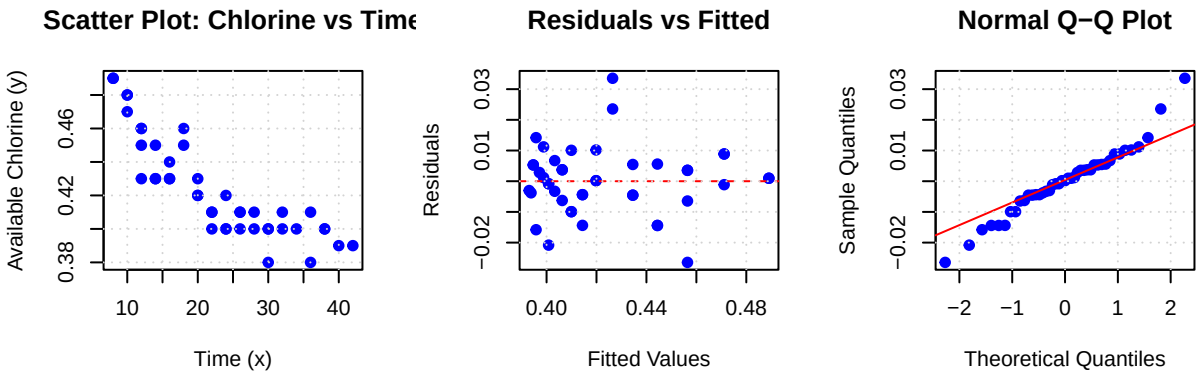
```r
cat("theta3 (decay rate) =", round(theta3_est, 4),
    "-> Negative value indicates decay\n")
```

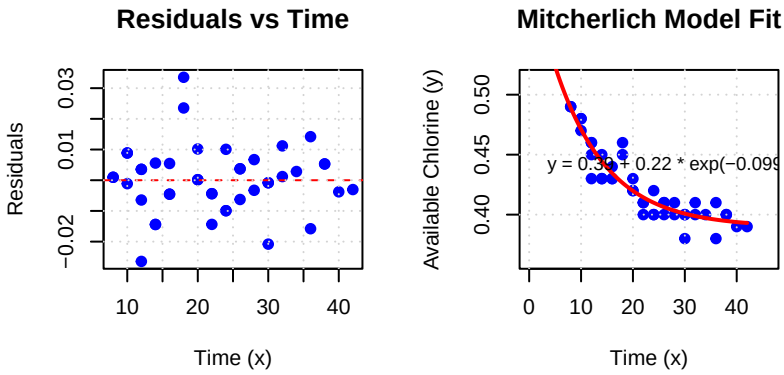
```
## theta3 (decay rate) = -0.0993 -> Negative value indicates decay
```

```r
# Half-life approximation (time for effect to reduce by half)
# For exponential decay: t_half = ln(0.5)/theta3
if (theta3_est < 0) {
  t_half <- log(0.5)/theta3_est
  cat("Approximate half-life of decay:", round(t_half, 2), "time units\n")
}
```

```
## Approximate half-life of decay: 6.98 time units
```

```r
# ------------------------------
# Reset plot layout
# ------------------------------
par(mfrow = c(1, 1))
```


Scatter Plot: Chlorine vs Time
Available Chlorine (y)
0.38
0.42
0.46
10
20
30
40
Time (x)
Residuals vs Fitted
Residuals
−0.02
0.01
0.03
0.40
0.44
0.48
Fitted Values
Normal Q−Q Plot
Sample Quantiles
−0.02
0.01
0.03
−2
−1
0
1
2
Theoretical Quantiles


Residuals vs Time
Residuals
−0.02
0.01
0.03
10
20
30
40
Time (x)
Mitcherlich Model Fit
Available Chlorine (y)
0.40
0.45
0.50
0
10
20
30
40
Time (x)

Question 3.16:

```
# ========================================
# 3.16 Michaelis-Menten Model - Base R Solution
# ========================================

# ------------------------------
# (1) Enter data
# ------------------------------
y <- c(0.0773895, 0.0688714, 0.0819351, 0.0737034, 0.0738753,
       0.0712396, 0.0650420, 0.0547667, 0.0497128, 0.0642727,
       0.0613005, 0.0643576, 0.0393892)
x <- c(0.417, 0.417, 0.417, 0.833, 0.833, 0.833, 1.670, 1.670,
       3.750, 3.750, 6.250, 6.250, 6.250)

data <- data.frame(y = y, x = x)
n <- length(y)

# ------------------------------
# (2) Scatter plot (a)
# ------------------------------
par(mfrow = c(2, 3))  # Set up 2x3 plot layout

# Scatter plot
plot(x, y, pch = 19, col = "blue",
     main = "Scatter Plot: y vs x",
     xlab = "Concentration (x)",
     ylab = "Velocity (y)")
grid()

# ------------------------------
# (3) Fit Michaelis-Menten model (b)
# ------------------------------
# Michaelis-Menten: y = (Vmax * x) / (Km + x)

# Get starting values
Vmax_start <- max(y) * 1.1
Km_start <- median(x)
cat("Starting values: Vmax =", Vmax_start, "Km =", Km_start, "\n\n")
```

```
## Starting values: Vmax = 0.09012861 Km = 1.67
```

```
# Fit nonlinear model
fit_nls <- nls(y ~ (Vmax * x) / (Km + x),
               start = list(Vmax = Vmax_start, Km = Km_start),
               data = data)

cat("=== Michaelis-Menten Model Fit ===\n")
```

```
## === Michaelis-Menten Model Fit ===
```

```r
print(summary(fit_nls))
```

```
##
## Formula: y ~ (Vmax * x)/(Km + x)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## Vmax  0.056955   0.003259  17.479 2.26e-09 ***
## Km   -0.113480   0.028294  -4.011  0.00205 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00863 on 11 degrees of freedom
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 8.44e-07
```

```r
# Extract parameters
params <- coef(fit_nls)
Vmax_est <- params["Vmax"]
Km_est <- params["Km"]

# ------------------------------
# (4) Confidence intervals
# ------------------------------
se <- summary(fit_nls)$coefficients[, "Std. Error"]
t_val <- qt(0.975, df = n - 2)  # 2 parameters

ci_Vmax <- Vmax_est + c(-1, 1) * t_val * se["Vmax"]
ci_Km <- Km_est + c(-1, 1) * t_val * se["Km"]

cat("\n=== 95% Confidence Intervals ===\n")
```

```
##
## === 95% Confidence Intervals ===
```

```r
cat("Vmax: [", ci_Vmax[1], ", ", ci_Vmax[2], "]\n", sep = "")
```

```
## Vmax: [0.04978326, 0.06412724]
```

```r
cat("Km:   [", ci_Km[1], ", ", ci_Km[2], "]\n", sep = "")
```

```
## Km:   [-0.1757539, -0.05120529]
```

```r
# Check if CIs include zero
cat("\nParameters different from zero?\n")
```

```
##
## Parameters different from zero?
```

```r
cat("Vmax:", ifelse(ci_Vmax[1] > 0, "YES (CI > 0)", "NO"), "\n")
```

```
## Vmax: YES (CI > 0)
```

```r
cat("Km:  ", ifelse(ci_Km[1] > 0, "YES (CI > 0)", "NO"), "\n")
```

```
## Km:   NO
```

```r
# ------------------------------
# (5) Model adequacy - Residual analysis
# ------------------------------
# Get predictions and residuals
y_pred <- predict(fit_nls)
resid <- residuals(fit_nls)

# (a) Residuals vs Fitted
plot(y_pred, resid, pch = 19, col = "blue",
     main = "Residuals vs Fitted",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)
grid()

# (b) Normal Q-Q plot
qqnorm(resid, main = "Normal Q-Q Plot", pch = 19, col = "blue")
qqline(resid, col = "red")
grid()

# (c) Shapiro-Wilk test
shapiro_test <- shapiro.test(resid)
cat("\n=== Shapiro-Wilk Normality Test ===\n")
```

```
##
## === Shapiro-Wilk Normality Test ===
```

```r
cat("W =", shapiro_test$statistic, "p-value =", shapiro_test$p.value, "\n")
```

```
## W = 0.8492754 p-value = 0.02789471
```

```r
# (d) Residuals vs Predictor
plot(x, resid, pch = 19, col = "blue",
     main = "Residuals vs Concentration",
     xlab = "Concentration (x)", ylab = "Residuals")
abline(h = 0, col = "red", lty = 2)
grid()

# ------------------------------
# (6) Goodness of fit
# ------------------------------
SSE <- sum(resid^2)
SST <- sum((y - mean(y))^2)
```

```r
R_sq <- 1 - SSE/SST
sigma2_hat <- SSE/(n - 2)

cat("\n=== Goodness of Fit ===\n")
```

```
##
## === Goodness of Fit ===
```

```r
cat("SSE:", SSE, "\n")
```

```
## SSE: 0.0008192453
```

```r
cat("SST:", SST, "\n")
```

```
## SST: 0.001657643
```

```r
cat("R-squared:", R_sq, "\n")
```

```
## R-squared: 0.5057771
```

```r
cat("sigma^2 estimate:", sigma2_hat, "\n")
```

```
## sigma^2 estimate: 7.447684e-05
```

```r
cat("RMSE:", sqrt(sigma2_hat), "\n")
```

```
## RMSE: 0.008629997
```

```r
# ------------------------------
# (7) Fitted curve plot
# ------------------------------
# Plot data
plot(x, y, pch = 19, col = "blue",
     main = "Michaelis-Menten Fit",
     xlab = "Concentration (x)",
     ylab = "Velocity (y)",
     xlim = c(0, max(x)*1.1),
     ylim = c(0, max(y)*1.1))
grid()

# Add fitted curve
x_seq <- seq(0, max(x), length = 100)
y_fit_seq <- (Vmax_est * x_seq) / (Km_est + x_seq)
lines(x_seq, y_fit_seq, col = "red", lwd = 2)

# Add equation text
text(0.7*max(x), 0.9*max(y),
     paste("y = (", round(Vmax_est, 4), "*x) / (",
           round(Km_est, 4), " + x)", sep = ""),
```

```r
     cex = 0.9)

# ------------------------------
# (8) Lack-of-fit test (using replicates)
# ------------------------------
# Check for replicates
x_unique <- unique(x)
n_unique <- length(x_unique)

if (n_unique < n) {
  # Calculate pure error SS
  SS_pe <- 0
  for (xi in x_unique) {
    yi <- y[x == xi]
    SS_pe <- SS_pe + sum((yi - mean(yi))^2)
  }

  df_pe <- n - n_unique
  df_lof <- n_unique - 2  # 2 parameters

  SS_lof <- SSE - SS_pe

  if (df_lof > 0 && df_pe > 0) {
    MS_pe <- SS_pe / df_pe
    MS_lof <- SS_lof / df_lof
    F_lof <- MS_lof / MS_pe
    p_lof <- pf(F_lof, df_lof, df_pe, lower.tail = FALSE)

    cat("\n=== Lack-of-Fit Test ===\n")
    cat("Pure Error SS:", SS_pe, "df:", df_pe, "\n")
    cat("Lack-of-Fit SS:", SS_lof, "df:", df_lof, "\n")
    cat("F-statistic:", F_lof, "\n")
    cat("p-value:", p_lof, "\n")

    if (p_lof < 0.05) {
      cat("Conclusion: Significant lack-of-fit exists\n")
    } else {
      cat("Conclusion: No significant lack-of-fit\n")
    }
  }
} else {
  cat("\nNote: No replicates for lack-of-fit test\n")
}
```

```
##
## === Lack-of-Fit Test ===
## Pure Error SS: 0.0006220527 df: 8
## Lack-of-Fit SS: 0.0001971925 df: 3
## F-statistic: 0.8453411
## p-value: 0.5066697
## Conclusion: No significant lack-of-fit
```

```r
# ------------------------------
# (9) Compare with Lineweaver-Burk linearization
# ------------------------------
inv_y <- 1/y
inv_x <- 1/x

# Fit linear model
fit_lm <- lm(inv_y ~ inv_x)
cat("\n=== Lineweaver-Burk Linearized Model ===\n")
```

```
##
## === Lineweaver-Burk Linearized Model ===
```

```r
print(summary(fit_lm))
```

```
##
## Call:
## lm(formula = inv_y ~ inv_x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.549 -1.872 -1.403  1.288  7.301
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.4941     1.2312  15.021 1.12e-08 ***
## inv_x        -2.5440     0.9356  -2.719     0.02 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.884 on 11 degrees of freedom
## Multiple R-squared:  0.4019, Adjusted R-squared:  0.3476
## F-statistic: 7.393 on 1 and 11 DF,  p-value: 0.01997
```

```r
# Extract parameters from linear fit
intercept <- coef(fit_lm)[1]
slope <- coef(fit_lm)[2]

Vmax_lb <- 1/intercept
Km_lb <- slope/intercept

cat("\nParameters from Lineweaver-Burk:\n")
```

```
##
## Parameters from Lineweaver-Burk:
```

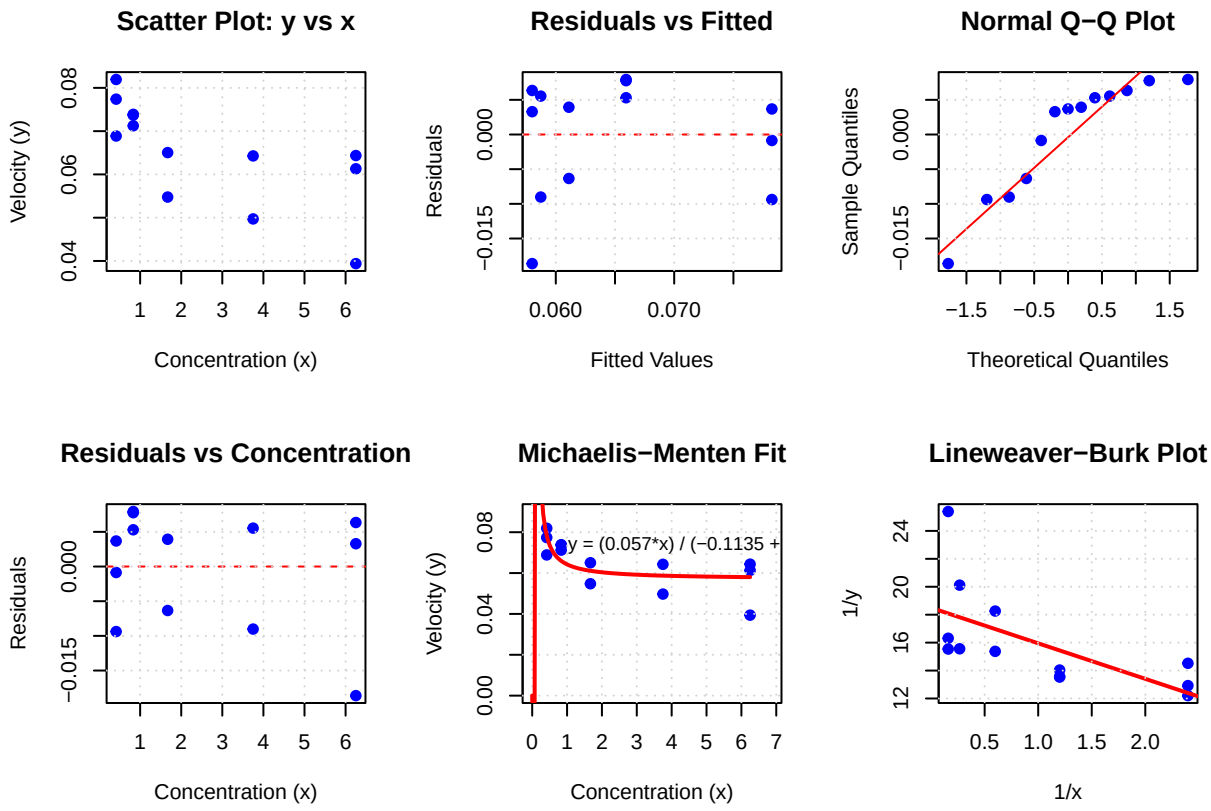
```r
cat("Vmax =", Vmax_lb, "\n")
```

```
## Vmax = 0.0540713
```

```r
cat("Km =", Km_lb, "\n")
```

```
## Km = -0.1375562
```

```r
# Plot Lineweaver-Burk
plot(inv_x, inv_y, pch = 19, col = "blue",
     main = "Lineweaver-Burk Plot",
     xlab = "1/x", ylab = "1/y")
abline(fit_lm, col = "red", lwd = 2)
grid()
```




```r
# ------------------------------
# (10) Overall model assessment
# ------------------------------
cat("\n=== MODEL ADEQUACY SUMMARY ===\n")
```

```
##
## === MODEL ADEQUACY SUMMARY ===
```

```r
cat("1. Parameters: Vmax =", round(Vmax_est, 5),
    "Km =", round(Km_est, 5), "\n")
```

```
## 1. Parameters: Vmax = 0.05696 Km = -0.11348
```

```r
cat("2. R-squared:", round(R_sq, 4), "\n")
```

```
## 2. R-squared: 0.5058
```

```r
cat("3. Residual normality (Shapiro-Wilk): p =",
    round(shapiro_test$p.value, 4), "\n")
```

```
## 3. Residual normality (Shapiro-Wilk): p = 0.0279
```

```r
cat("4. Residual plots: Check for patterns\n")
```

```
## 4. Residual plots: Check for patterns
```

```r
if (exists("p_lof")) {
  cat("5. Lack-of-fit: p =", round(p_lof, 4), "\n")
}
```

```
## 5. Lack-of-fit: p = 0.5067
```

```r
# Check for issues
issues <- 0
if (shapiro_test$p.value < 0.05) {
  cat("WARNING: Residuals may not be normally distributed\n")
  issues <- issues + 1
}
```

```
## WARNING: Residuals may not be normally distributed
```

```r
if (R_sq < 0.7) {
  cat("WARNING: R-squared is relatively low\n")
  issues <- issues + 1
}
```

```
## WARNING: R-squared is relatively low
```

```r
if (exists("p_lof") && p_lof < 0.05) {
  cat("WARNING: Significant lack-of-fit detected\n")
  issues <- issues + 1
}

if (issues == 0) {
  cat("\nCONCLUSION: Michaelis-Menten model appears adequate.\n")
} else {
  cat("\nCONCLUSION: Model has", issues, "potential issue(s).\n")
}
```

```
##
## CONCLUSION: Model has 2 potential issue(s).
```

```r
# Reset plot layout
par(mfrow = c(1, 1))
```