Homework 4 - Linear Models

```
# Library Setup
suppressWarnings(suppressMessages({
  library(ggplot2)
  library(dplyr)
  library(car) # For checking model assumptions
  library(MASS) # For transformations
  library(broom) # For tidying model outputs
}))
```

Q6.16

First few steps:

- Fit the model
- Check the model assumptions
- Check for multicollinearity

```
# Load the data
data <- read.table("recovery", header = TRUE)

head(data)
```

	x1	x2	y
	<dbl>	<int>	<int>
1	2.26	66	7
2	1.81	52	10
3	1.78	72	18
4	1.54	67	4
5	2.06	69	10

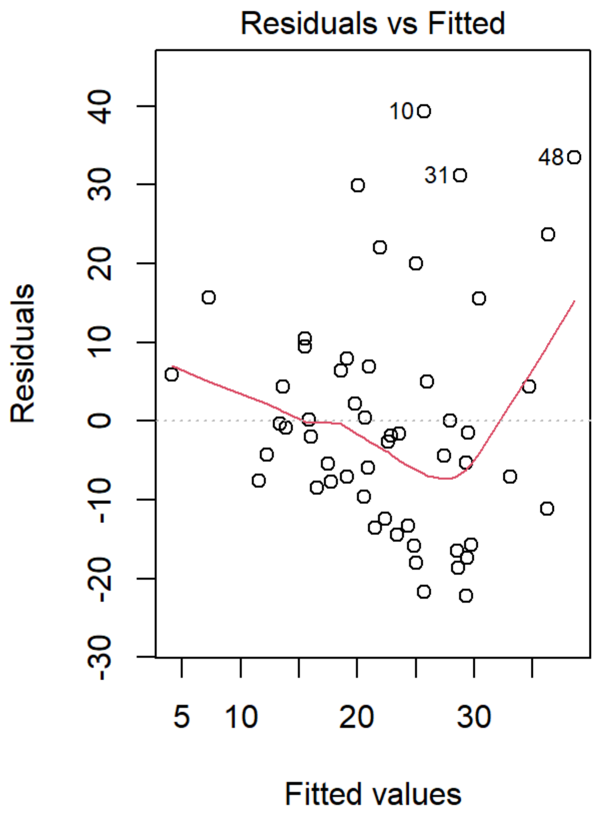
	x1 <dbl>	x2 <int>	y <int>
6	1.74	71	13

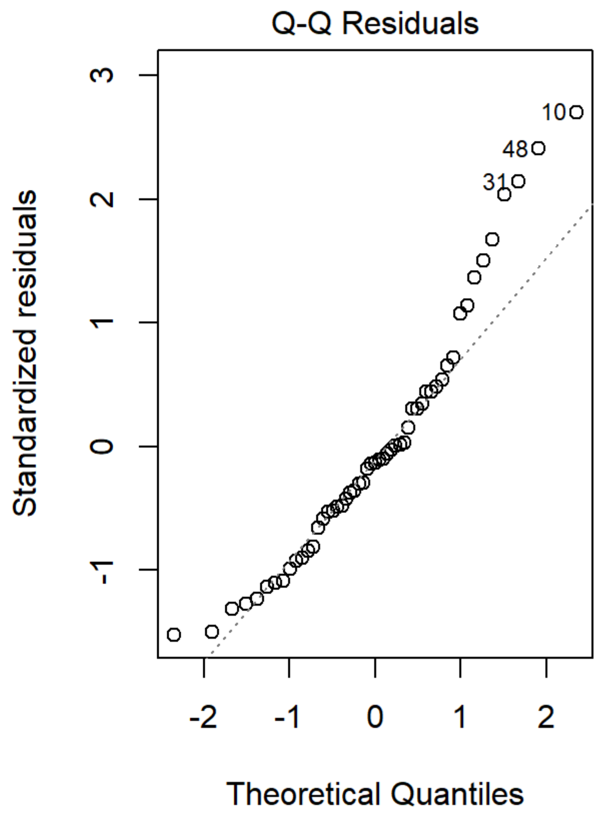
6 rows

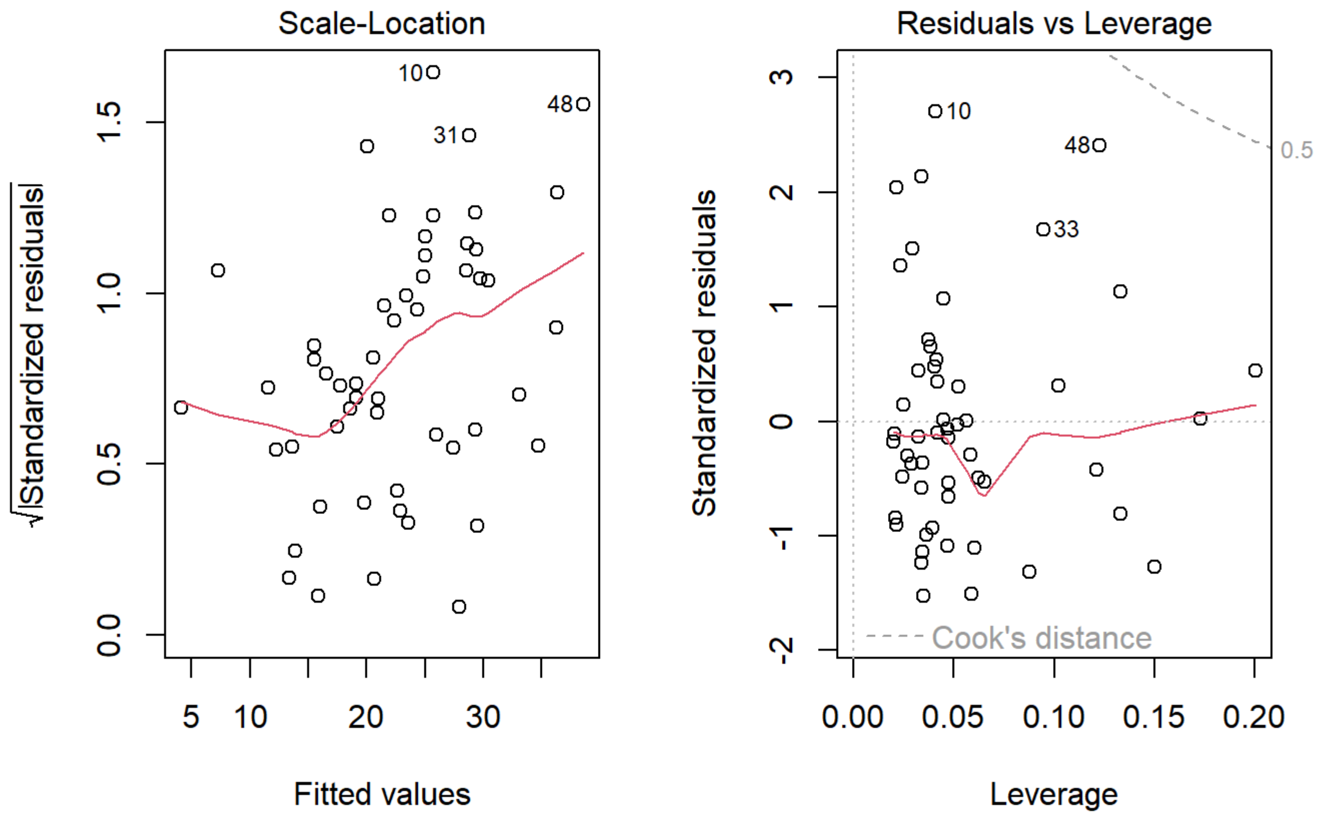
```
# Fit a linear regression model
model1 <- lm(y ~ x1 + x2, data = data)
summary(model1)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.265  -9.563  -1.916   6.405  39.319
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23.0107    18.2849   1.258  0.21407
## x1           23.6386     6.8479   3.452  0.00114 **
## x2           -0.7147     0.3014  -2.371  0.02163 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.84 on 50 degrees of freedom
## Multiple R-squared:  0.2018, Adjusted R-squared:  0.1699
## F-statistic: 6.321 on 2 and 50 DF,  p-value: 0.00357
```

```
# Check for model violations
par(mfrow = c(1, 2))
plot(model1)
```


Residuals vs Fitted
Residuals
Fitted values
10
31
48
-30
-20
-10
0
10
20
30
40
5
10
20
30


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
10
48
31
-1
0
1
2
3
-2
-1
0
1
2



We observe that the "Residuals vs Fitted" plot indicates some non-linear behaviour. This could suggest that the relationship between the predictors and the response might be better suited to a quadratic model.

The QQ Residual plot also indicates heavy right tailed residuals, this suggest that the data is right skewed. The data might not be normally distributed.

The scale-location plot also shows some heteroscedasticity, which suggests that the variance of the residuals is not constant. This could be due to the non-linear relationship between the predictors and the response.

The residuals vs leverage plot shows that there are no influential data points since no points lie outside of Cook's Distance. However, some points do still exhibit high leverage.

```
# Check for multi-collinearity
vif(model1)
```

```
##       x1       x2 
## 1.281743 1.281743
```

We observe that the VIF of x1 and x2 are both less than 5, which indicates that there is signigificantly low multi-collinearity between the two predictors.

Now we explore the usefulness of transformation on the response:

```
# Log transformation of the response
model2 <- lm(log(y) ~ x1 + x2, data = data)
summary(model2)
```

```
## 
## Call:
## lm(formula = log(y) ~ x1 + x2, data = data)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.58599 -0.49722  0.06988  0.48008  1.19303 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  3.09063    0.79728   3.876  0.00031 ***
## x1           0.85789    0.29859   2.873  0.00595 ** 
## x2          -0.02876    0.01314  -2.188  0.03335 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.647 on 50 degrees of freedom
## Multiple R-squared:  0.1549, Adjusted R-squared:  0.1211 
## F-statistic: 4.581 on 2 and 50 DF,  p-value: 0.0149
```

```r
# Compare models
broom::glance(model1)
```

r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.value <dbl>	df <dbl>	logLik <dbl>	AIC <dbl>	BIC <dbl>	deviance <dbl>
0.2018118	0.1698842	14.83776	6.320933	0.00356971	2	-216.6099	441.2198	449.1009	11007.95

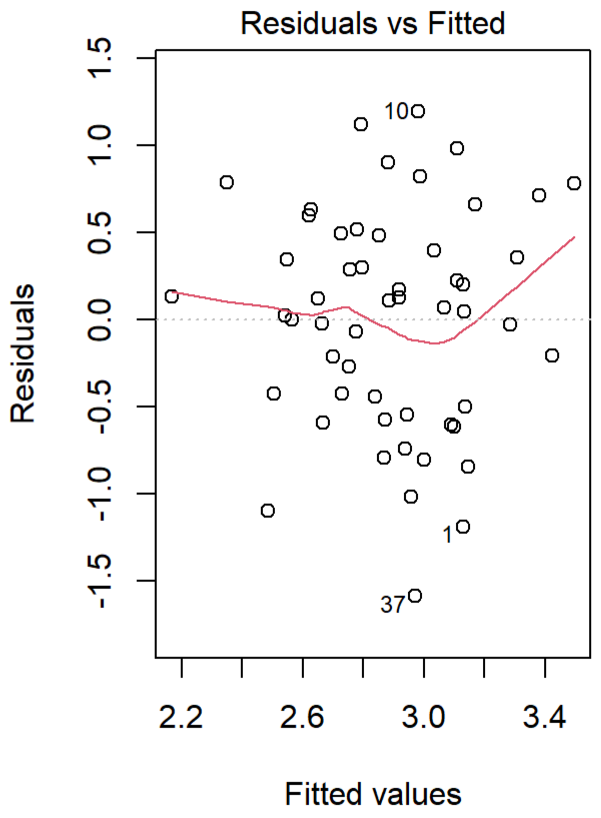
1 row | 1-10 of 12 columns

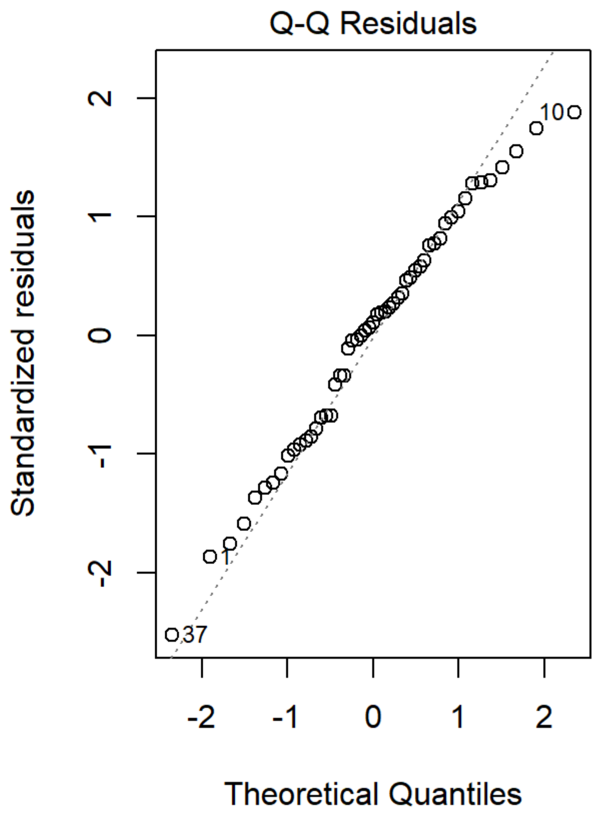
```r
broom::glance(model2)
```

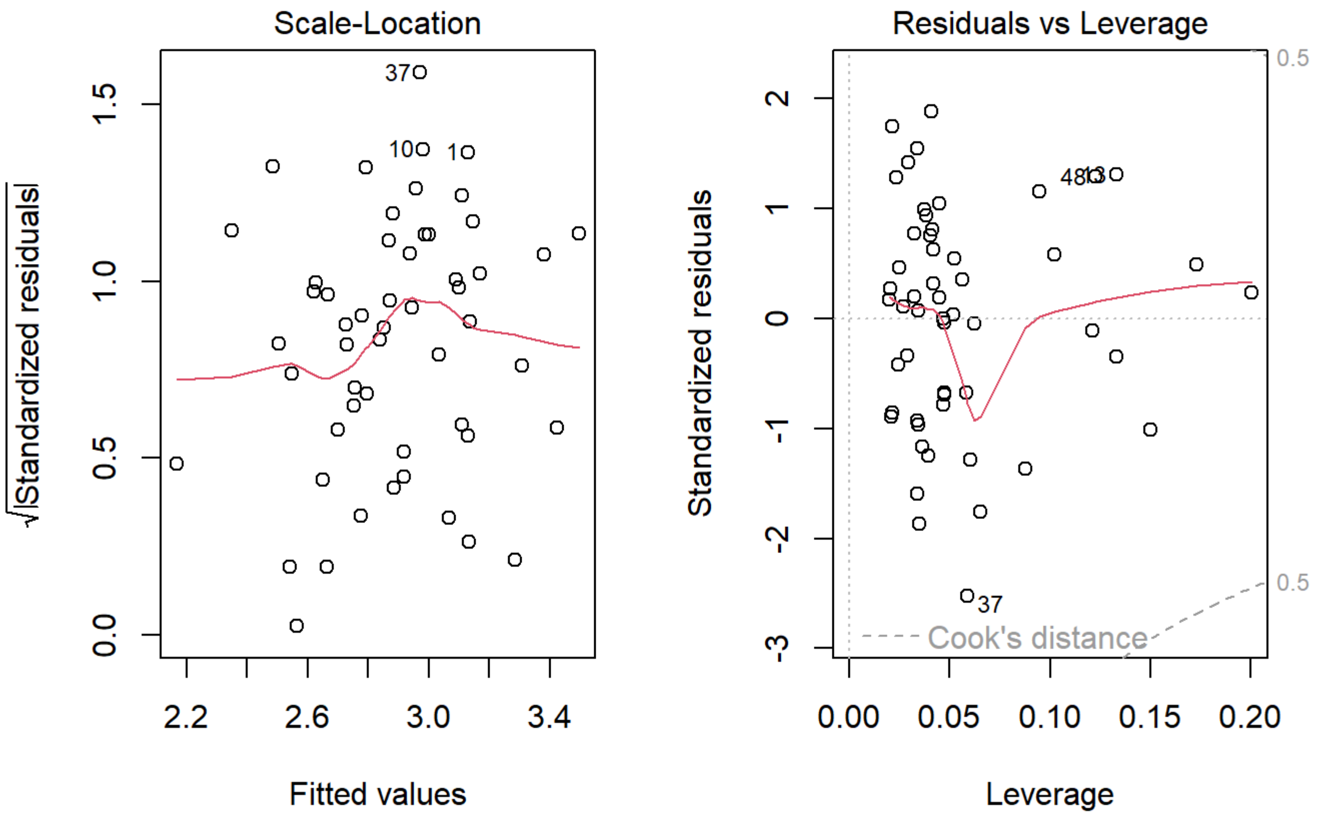
r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.value <dbl>	df <dbl>	logLik <dbl>	AIC <dbl>	BIC <dbl>	deviance <dbl>
0.1548619	0.1210564	0.6469762	4.580965	0.01490019	2	-50.58099	109.162	117.0432	20.92891

1 row | 1-10 of 12 columns

```r
# Check residuals of the log-transformed model
par(mfrow = c(1, 2))
plot(model2)
```


Residuals vs Fitted
Residuals
Fitted values
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
2.2
2.6
3.0
3.4
10
1
37


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
2
1
0
-1
-2
-2
-1
0
1
2
10
1
37



We can observe that after the log-transformation, the residuals vs fitted plot shows a more linear relationship. The QQ plot also shows that the residuals are more normally distributed - as can be seen from the less heavy right tail.

The scale-location plot also shows that the variance of the residuals is more constant. This suggests that the log-transformation of the response variable has improved the model fit.

The residuals vs leverage plot also shows that there are still no influential data points since no points lie outside of Cook's Distance. However, it seems like the points with high leverage before still have high leverage.

Interpretation of Results:

```
# Coefficients of model1
coefficients(model1)
```

```
## (Intercept)          x1          x2
##   23.010668   23.638558   -0.714675
```

```
# Coefficients of model2
coefficients(model2)
```

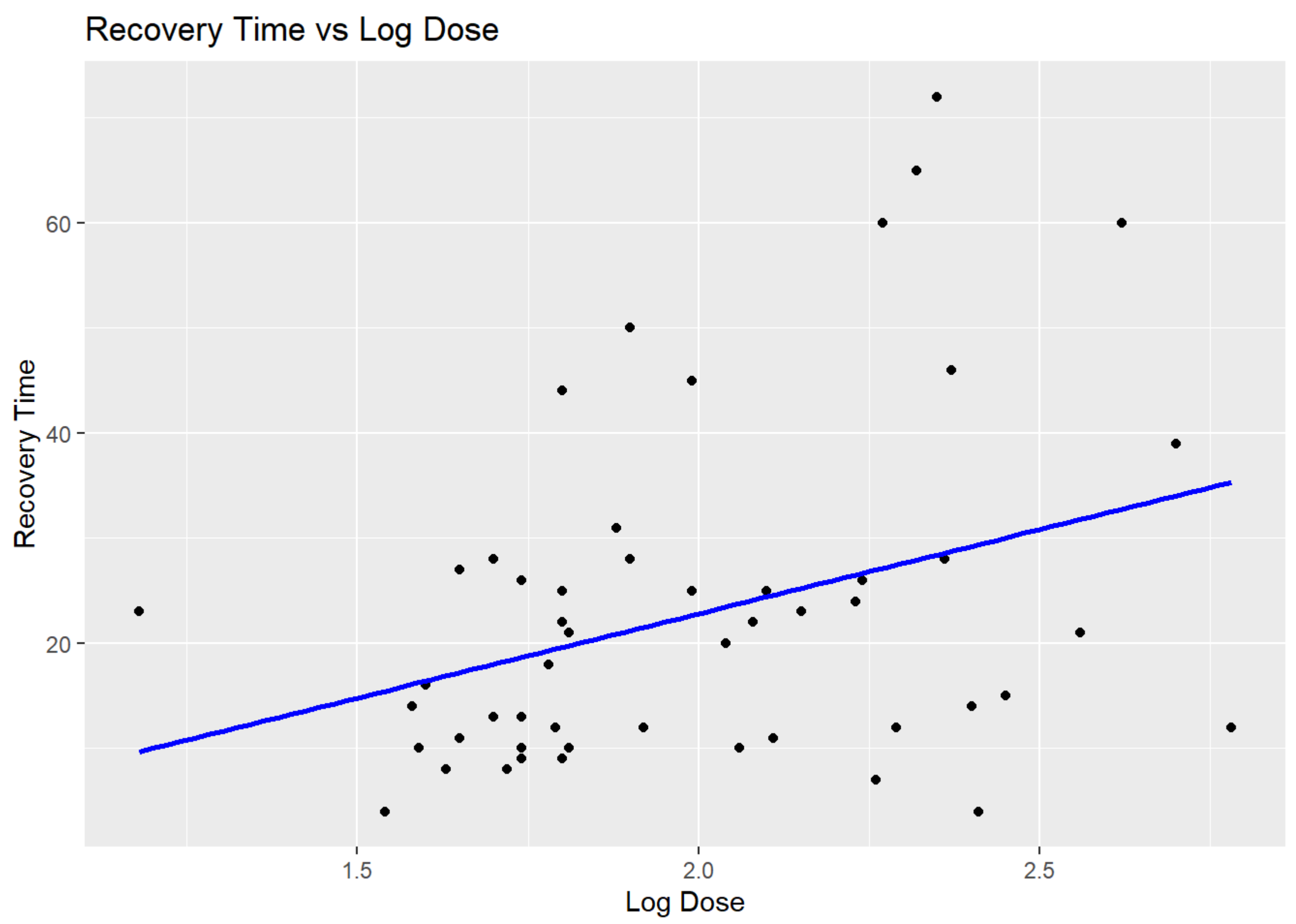
```
## (Intercept)          x1          x2
##   3.0906321   0.8578867  -0.0287612
```

```
# Predictions and confidence intervals
predictions <- predict(model1, newdata = data, interval = "confidence")
predictions_log <- predict(model2, newdata = data, interval = "confidence")

# Add predictions to the original data
data <- data %>%
  mutate(predicted_recovery = predictions[, "fit"],
         predicted_recovery_log = exp(predictions_log[, "fit"]))

# Visualize the relationships
ggplot(data, aes(x = x1, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, col = "blue") +
  labs(title = "Recovery Time vs Log Dose", x = "Log Dose", y = "Recovery Time")
```

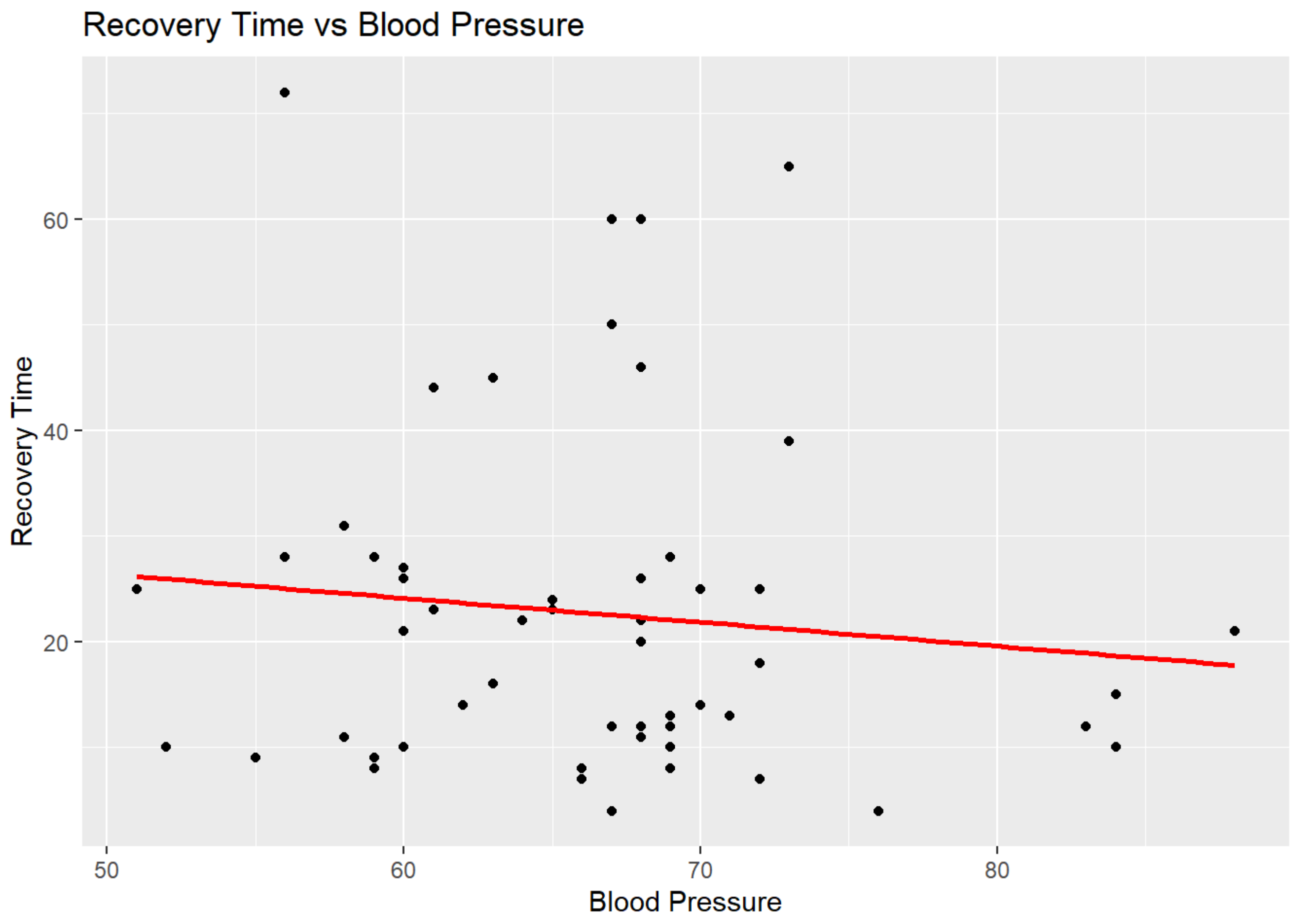
```
## `geom_smooth()` using formula = 'y ~ x'
```



The scatterplots indicate a positive linear relationship between log Dose and Recovery Time.

```
# plot for recovery vs blood pressure
ggplot(data, aes(x = x2, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(title = "Recovery Time vs Blood Pressure", x = "Blood Pressure", y = "Recovery Time")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

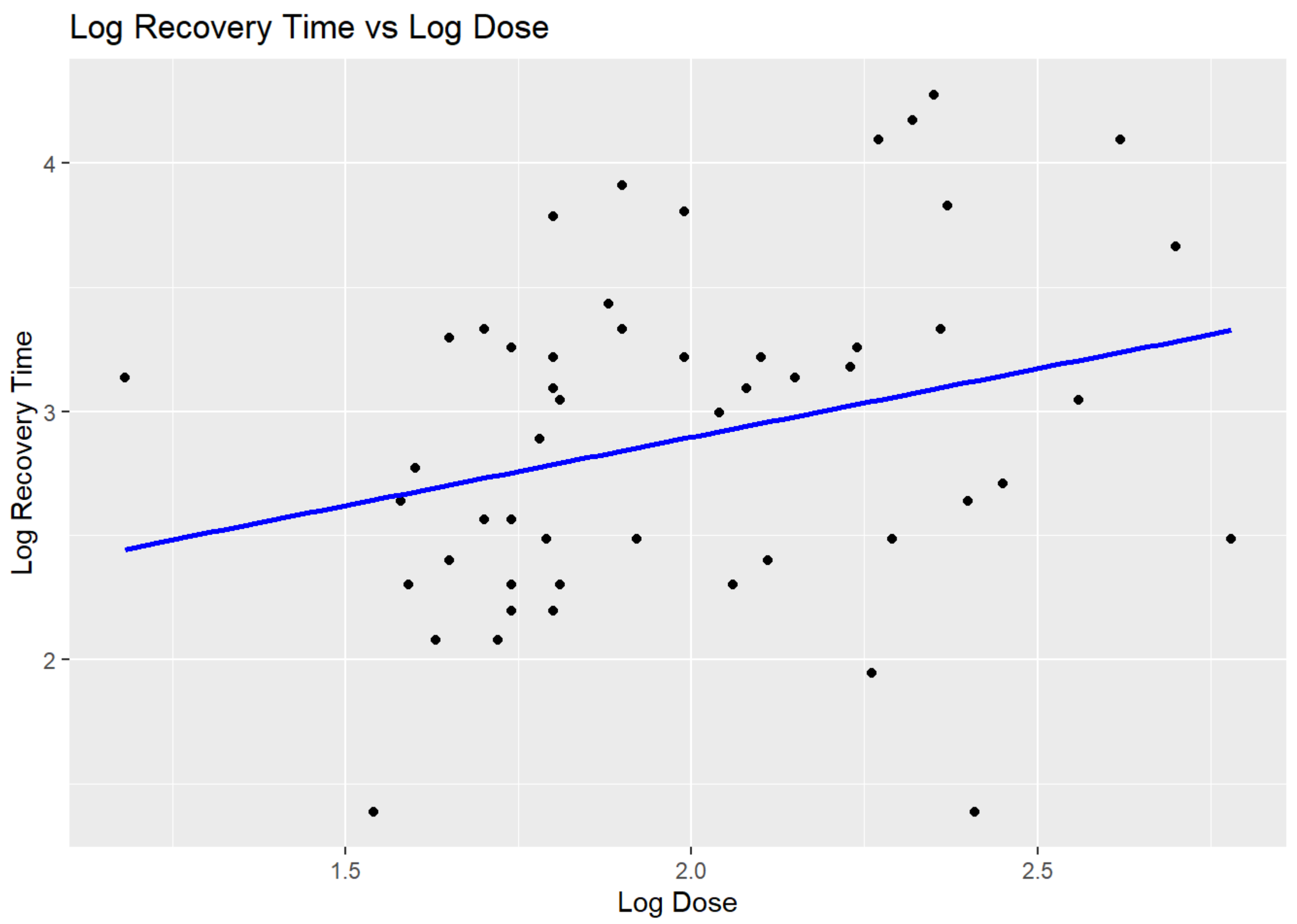


The relationship between Blood Pressure and Recovery Time is less clear, but it seems exhibit a slightly negative linear behaviour.

We can go one step further to observe the effect of the log transformation on the response variable: - compare the effect of log transformation on the response variable on the high leverage points

```
# plot a log(recovery) vs log dose graph
ggplot(data, aes(x = x1, y = log(y))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, col = "blue") +
  labs(title = "Log Recovery Time vs Log Dose", x = "Log Dose", y = "Log Recovery Time")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



We see that as compared to the “Recovery Time vs Log Dose” plot, the “Log Recovery Time vs Log Dose” plot shows that our transformation has helped (to some extent) in reducing the high leverage of those points, thereby helping to reduce non-constant variance and non-linearity.

Q7.5

```
# Load the data
rain_data <- read.table("rainseeding", header = TRUE)

rain_data
```

Seed <int>	time <int>	suit <dbl>	Echocov <dbl>	echomot <int>	wet <dbl>	rain <dbl>
0	0	1.75	13.4	2	0.274	12.85
1	1	2.70	37.9	1	1.267	5.52
1	3	4.10	3.9	2	0.198	6.29
0	4	2.35	5.3	1	0.526	6.11
1	6	4.25	7.1	1	0.250	2.45
0	9	1.60	6.9	2	0.018	3.61
0	18	1.30	4.6	1	0.307	0.47
0	25	3.35	4.9	1	0.194	4.56
0	27	2.85	12.1	1	0.751	6.35
1	28	2.20	5.2	1	0.084	5.06

1-10 of 24 rows   Previous **1** 2 3 Next

```r
# All possible regression

# Get all possible models
results <- list()
num_predictors <- ncol(rain_data) - 1  # Number of predictors

for (i in 1:num_predictors) {
  combn_names <- combn(names(rain_data)[-ncol(rain_data)], i, simplify = FALSE)  # Get combinations of predictors
  for (combo in combn_names) {
    metrics <- list()
    formula_str <- paste("rain ~", paste(combo, collapse = " + "))
    model <- lm(as.formula(formula_str), data = rain_data)
    metrics$r_squared <- summary(model)$r.squared # Get R-squared
    metrics$p_value <- broom::glance(model)$p.value # Get p-value
    results[[formula_str]] <- metrics # Store metrics in results
  }
}
```

```r
# Store results in a data frame
results_df <- do.call(rbind, lapply(results, as.data.frame))
results_df$formula <- rownames(results_df)
results_df <- results_df[, c("formula", "r_squared", "p_value")]

results_df
```

```
rain ~ Seed
rain ~ time
rain ~ suit
rain ~ Echocov
rain ~ echomot
```

rain ~ wet
rain ~ Seed + time
rain ~ Seed + suit
rain ~ Seed + Echocov
rain ~ Seed + echomot

1-10 of 63 rows | 1-1 of 4 columns Previous **1** 2 3 4 5 6 7 Next

```
# Get the best model combination based on max p-value
best_model <- results_df[which.min(results_df$p_value), ]

best_model
```

	formula <chr>	r_squared <dbl>	p_value <dbl>
rain ~ time	rain ~ time	0.2462519	0.01364925

1 row

```
# Perform backward elimination model selection
# Full model
model_full <- lm(rain ~ ., data = rain_data)
AIC_full <- AIC(model_full)
rsqr_full <- summary(model_full)$r.squared
p_val_full <- broom::glance(model_full)$p.value

cat("Full Model: AIC =", AIC_full, "R-squared =", rsqr_full, "p-value =", p_val_full, "\n")
```

```
## Full Model: AIC = 125.8717 R-squared = 0.3849316 p-value = 0.1647245
```

```r
# Backward elimination
model_backward <- stepAIC(model_full, direction = "backward", scope = formula(model_full), trace = FALSE)

AIC_backward <- AIC(model_backward)
rsqr_backward <- summary(model_backward)$r.squared
p_val_backward <- broom::glance(model_backward)$p.value

cat("Backward Elimination: AIC =", AIC_backward, "R-squared =", rsqr_backward, "p-value =", p_val_backward, "\n")
```

```
## Backward Elimination: AIC = 120.7515 R-squared = 0.2462519 p-value = 0.01364925
```

```r
model_backward$anova
```

Step <chr>	Df <dbl>	Deviance <dbl>	Resid. Df <dbl>	Resid. Dev <dbl>	AIC <dbl>
	NA	NA	17	136.7512	55.76262
- Echocov	1	0.0194813	18	136.7706	53.76604
- Seed	1	5.9708223	19	142.7415	52.79155
- wet	1	5.5874825	20	148.3290	51.71309
- suit	1	7.5906719	21	155.9196	50.91088
- echomot	1	11.6648700	22	167.5845	50.64241

6 rows

```r
# Stepwise selection
model_stepwise <- stepAIC(model_full, direction = "both", trace = FALSE)

AIC_stepwise <- AIC(model_stepwise)
rsqr_stepwise <- summary(model_stepwise)$r.squared
p_val_stepwise <- broom::glance(model_stepwise)$p.value

cat("Stepwise Selection: AIC =", AIC_stepwise, "R-squared =", rsqr_stepwise, "p-value =", p_val_stepwise, "\n")
```

```
## Stepwise Selection: AIC = 120.7515 R-squared = 0.2462519 p-value = 0.01364925
```

```r
model_stepwise$anova
```

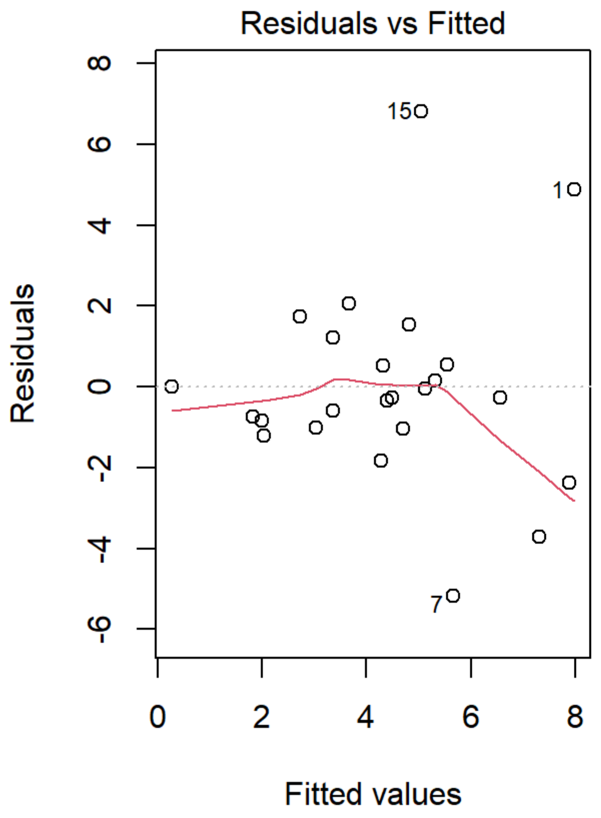
Step <chr>	Df <dbl>	Deviance <dbl>	Resid. Df <dbl>	Resid. Dev <dbl>	AIC <dbl>
	NA	NA	17	136.7512	55.76262
- Echocov	1	0.0194813	18	136.7706	53.76604
- Seed	1	5.9708223	19	142.7415	52.79155
- wet	1	5.5874825	20	148.3290	51.71309
- suit	1	7.5906719	21	155.9196	50.91088
- echomot	1	11.6648700	22	167.5845	50.64241

6 rows

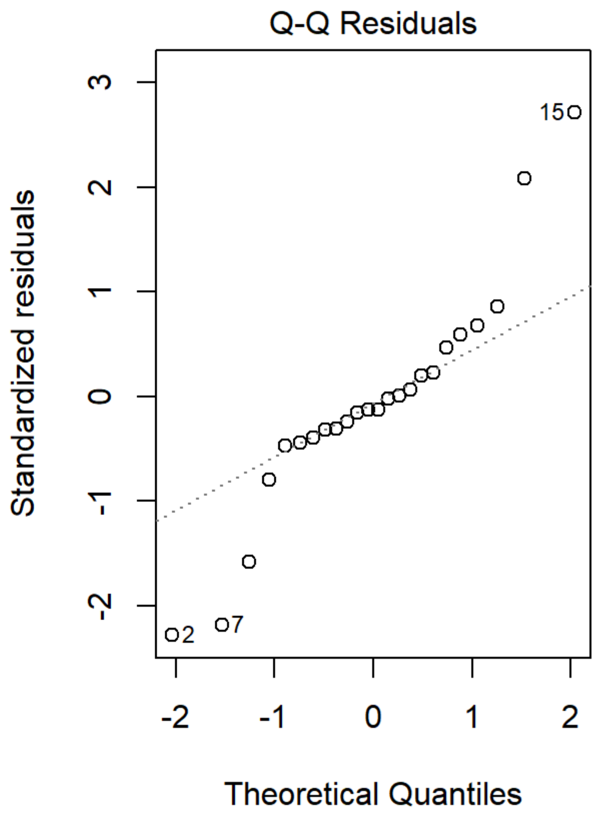
Assess the effectiveness of cloud seeding via full model, backward elimination and stepwise selection:

First: Check for unusual cases (influential points)

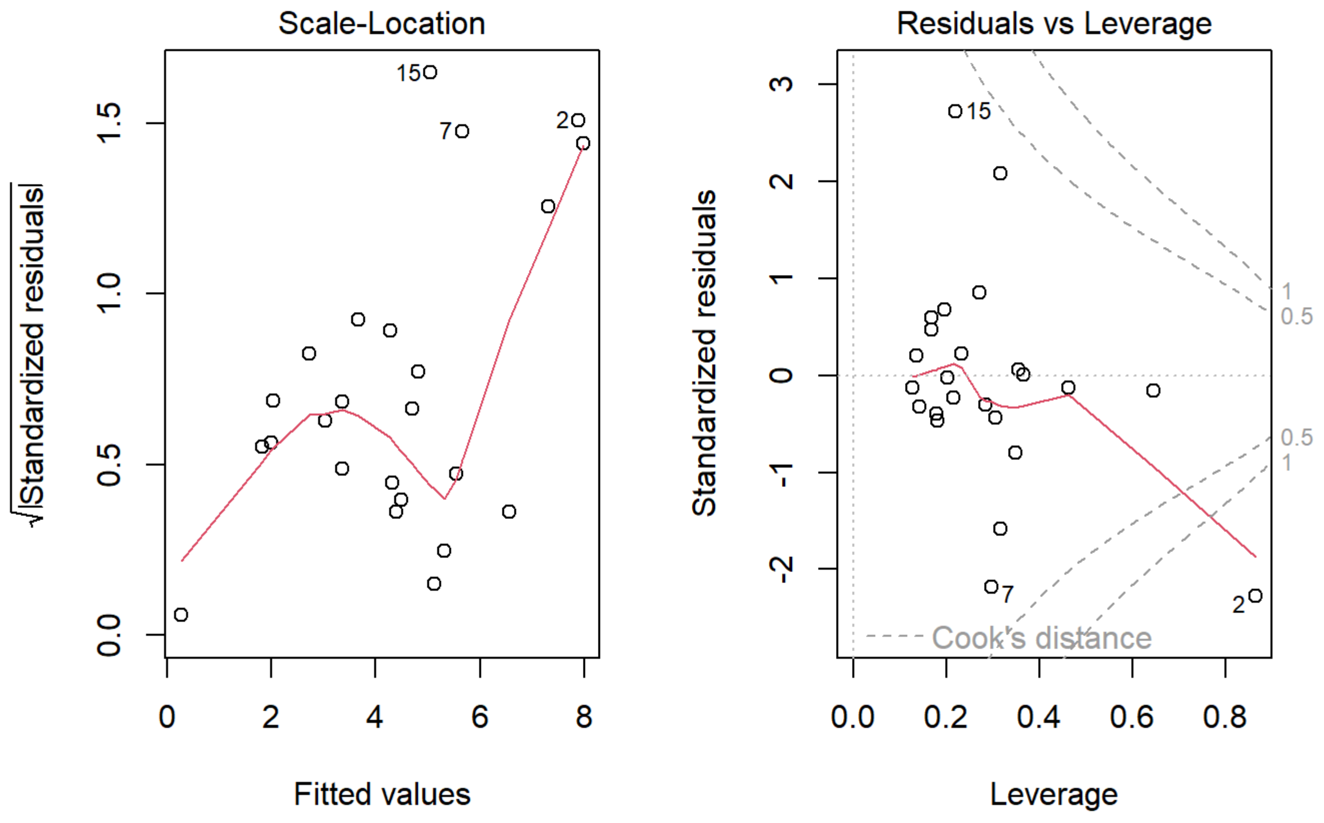
```r
# Full model
par(mfrow = c(1, 2))
plot(model_full)
```



Residuals vs Fitted
Residuals
Fitted values
15
10
7



Q-Q Residuals
Standardized residuals
Theoretical Quantiles
15
2
7



```
# Cook's distance to identify influential points
cooksd_full <- cooks.distance(model_full)
influential_points_full <- which(cooksd_full > (4 / length(cooksd_full)))  # threshold set in lecture slides

cat("Influential points:", influential_points_full, "\n")
```

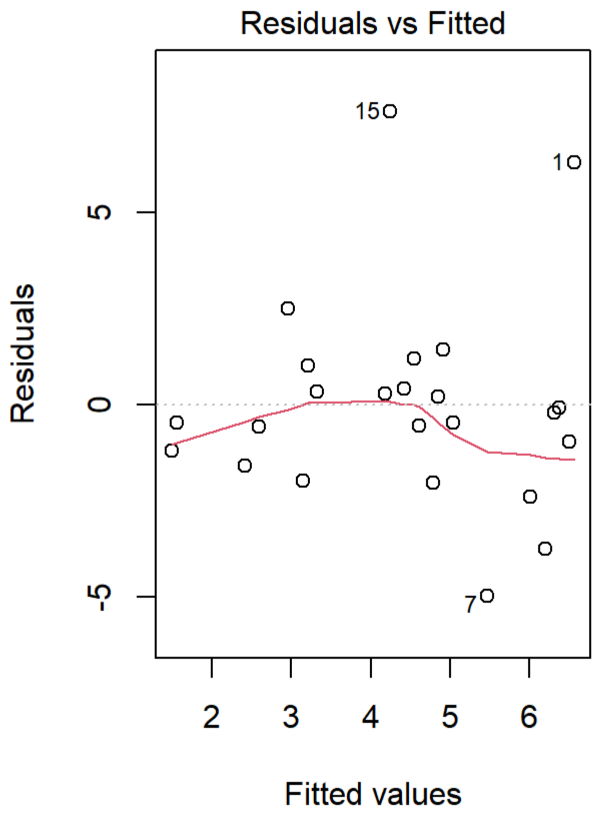
```
## Influential points: 1 2 7 15
```

We observe the Residuals vs Fitted plot shows some signs of nonlinearity, while the QQ plot shows some heavy tailed residuals.

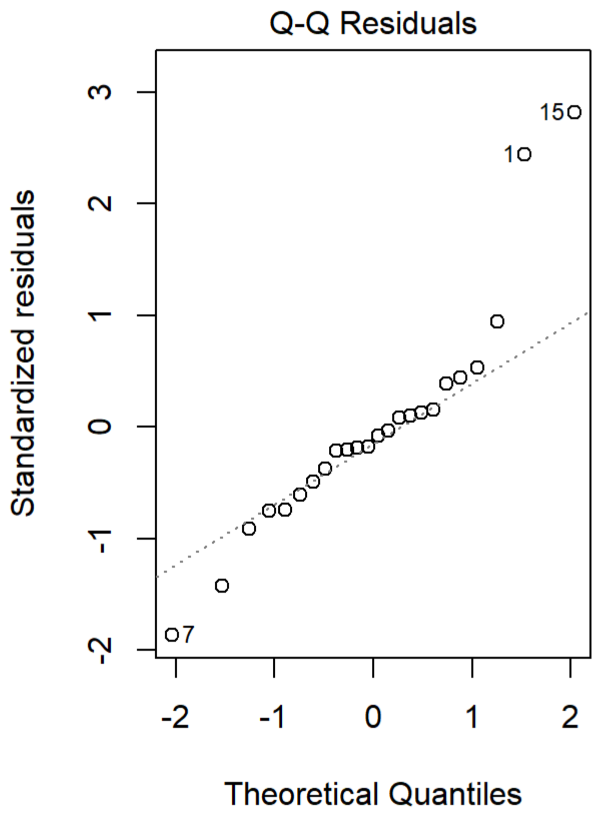
The Scale-Location plot also shows some heteroscedasticity, which suggests that the variance of the residuals is not constant.

The Residuals vs Leverage plot shows that there are some influential data points, as indicated by the points outside of Cook’s Distance.

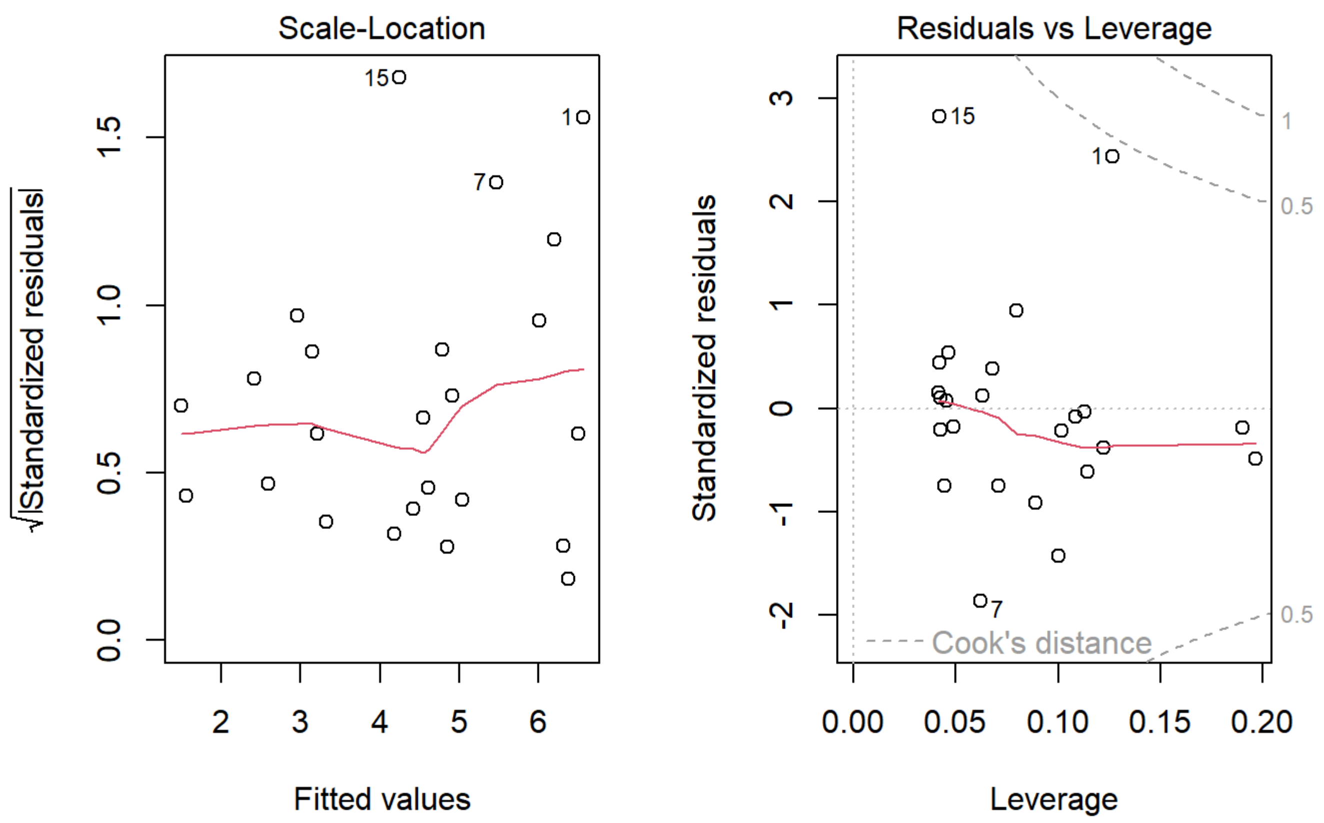
```r
# Backward elimination
par(mfrow = c(1, 2))
plot(model_backward)
```



Residuals vs Fitted
Residuals
Fitted values
15
1
7
5
0
-5
2
3
4
5
6



Q-Q Residuals
Standardized residuals
Theoretical Quantiles
15
1
7
3
2
1
0
-1
-2
-2
-1
0
1
2



```
# Cook's distance to identify influential points
cooksd_backward <- cooks.distance(model_backward)
influential_points_backward <- which(cooksd_backward > (4 / length(cooksd_backward)))  # threshold set in lecture slides

cat("Influential points:", influential_points_backward, "\n")
```

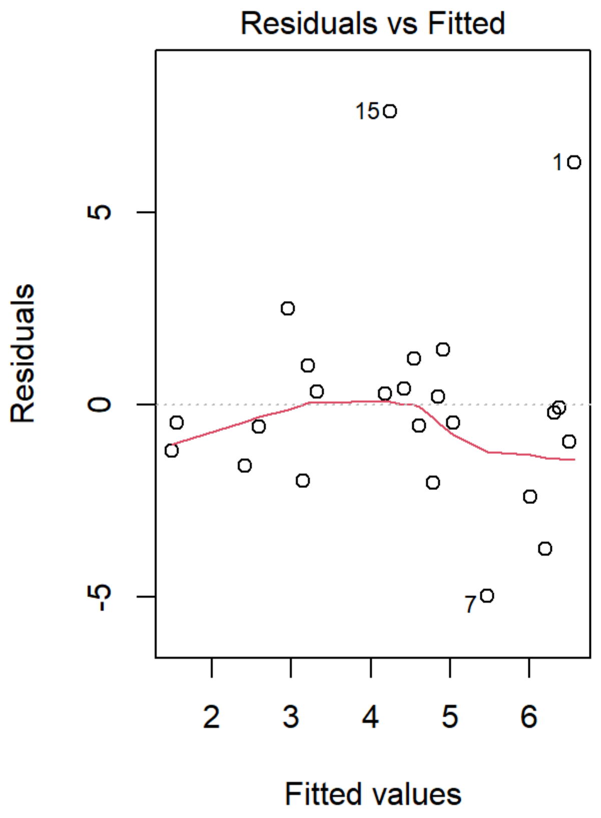
```
## Influential points: 1 15
```

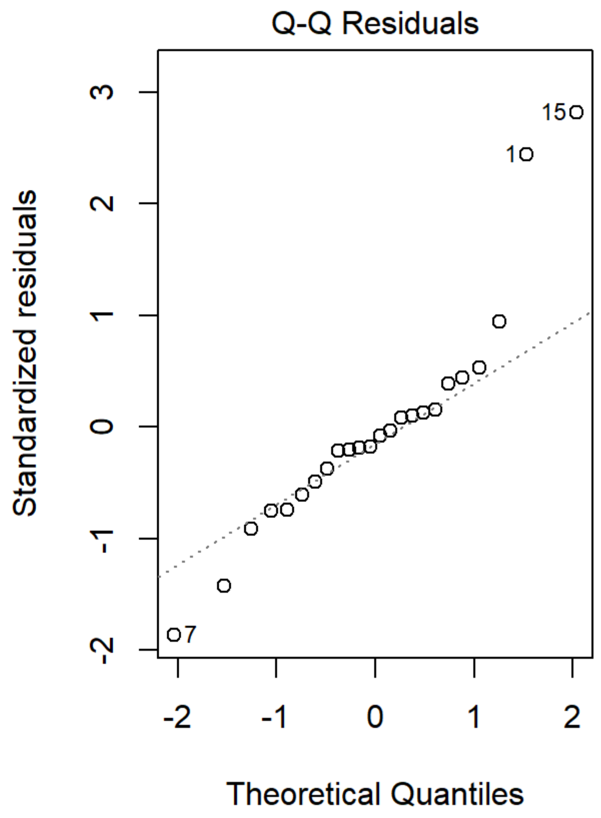
After running the backward elimination model, observations did not change much from the full model but we notice a more horizontal line in the Residuals vs Fitted plot. This suggests that the new model can capture the linearity of the data better. The QQ plot also shows that the residuals are more normally distributed, as indicated by the less heavy tails.

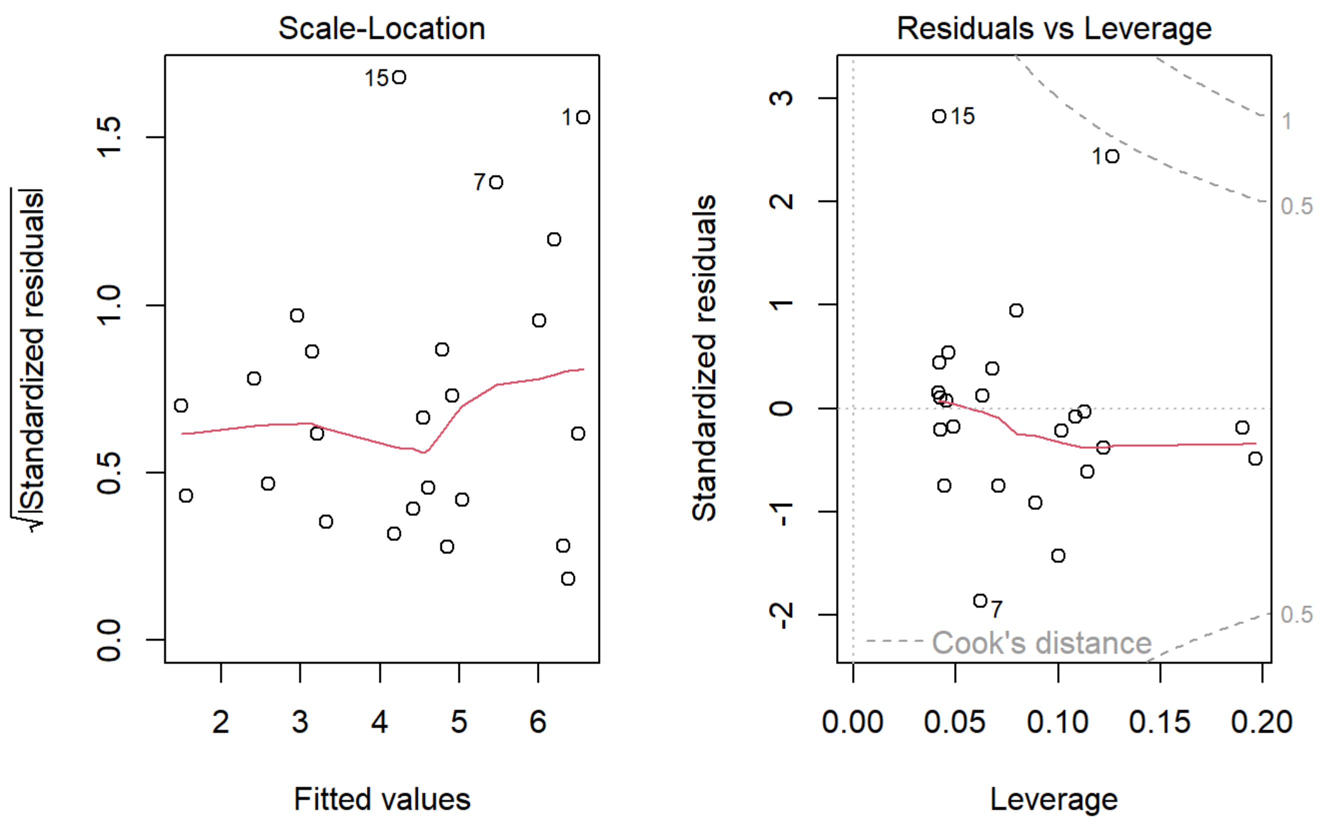
The Scale-Location plot also shows that the variance of the residuals is more constant. This suggests that the backward elimination model has improved the model fit.

The Residuals vs Leverage plot shows that there are still some influential data points, as indicated by the points outside of Cook’s Distance.

```r
# Stepwise selection
par(mfrow = c(1, 2))
plot(model_stepwise)
```


Residuals vs Fitted
Residuals
Fitted values
15
1
7
5
0
-5
2
3
4
5
6


Q-Q Residuals
Standardized residuals
Theoretical Quantiles
15
1
7
3
2
1
0
-1
-2
-2
-1
0
1
2



```
# Cook's distance to identify influential points
cooksd_stepwise <- cooks.distance(model_stepwise)
influential_points_stepwise <- which(cooksd_stepwise > (4 / length(cooksd_stepwise)))  # threshold set in lecture slides

cat("Influential points:", influential_points_stepwise, "\n")
```

```
## Influential points: 1 15
```

The observations here are similar to that of backward elimination.

Second: Sensitivy Analysis (Compare models with and without influential points)

```r
# data without influential points (full model)
rain_data_no_influential <- rain_data[-influential_points_full, ]

# data without influential points (backward elimination)
rain_data_no_influential_backward <- rain_data[-influential_points_backward, ]

# data without influential points (stepwise selection)
rain_data_no_influential_stepwise <- rain_data[-influential_points_stepwise, ]

# Full model without influential points
model_full_no_influential <- lm(rain ~ ., data = rain_data_no_influential)

# Backward elimination without influential points
model_backward_no_influential <- stepAIC(model_full_no_influential, direction = "backward", scope = formula(model_full_no_in
fluential), trace = FALSE)

# Stepwise selection without influential points
model_stepwise_no_influential <- stepAIC(model_full_no_influential, direction = "both", trace = FALSE)

# Compare models AIC, rsquared, and p-values
AIC_full_no_influential <- AIC(model_full_no_influential)
rsqr_full_no_influential <- summary(model_full_no_influential)$r.squared
p_val_full_no_influential <- broom::glance(model_full_no_influential)$p.value

AIC_backward_no_influential <- AIC(model_backward_no_influential)
rsqr_backward_no_influential <- summary(model_backward_no_influential)$r.squared
p_val_backward_no_influential <- broom::glance(model_backward_no_influential)$p.value

AIC_stepwise_no_influential <- AIC(model_stepwise_no_influential)
rsqr_stepwise_no_influential <- summary(model_stepwise_no_influential)$r.squared
p_val_stepwise_no_influential <- broom::glance(model_stepwise_no_influential)$p.value

# Create a dataframe to display results of sensitivity analysis with and without influential points
sensitivity_results <- data.frame(
  Model = c("Full Model", "Backward Elimination", "Stepwise Selection"),
  # AICs
  AIC = c(AIC_full, AIC_backward, AIC_stepwise),
  AIC_no_influential = c(AIC_full_no_influential, AIC_backward_no_influential, AIC_stepwise_no_influential),
  # R-squared
  R_squared = c(rsqr_full, rsqr_backward, rsqr_stepwise),
  R_squared_no_influential = c(rsqr_full_no_influential, rsqr_backward_no_influential, rsqr_stepwise_no_influential),

  # p-values
  p_value = c(p_val_full, p_val_backward, p_val_stepwise),
  p_value_no_influential = c(p_val_full_no_influential, p_val_backward_no_influential, p_val_stepwise_no_influential)
)

sensitivity_results
```

Model <chr>	AIC <dbl>	AIC_no_influential <dbl>	R_squared <dbl>	R_squared_no_influential <dbl>	p_value <dbl>
Full Model	125.8717	72.38115	0.3849316	0.7199831	0.16472453
Backward Elimination	120.7515	71.56935	0.2462519	0.6715909	0.01364925
Stepwise Selection	120.7515	71.56935	0.2462519	0.6715909	0.01364925

3 rows | 1-6 of 7 columns

Third: Review the sensitivity of the results based on the unsual cases

1. Trend Across Models:

- The AIC values (with & without influential points) are higher for the Full Model as compared to the Backward Elimination and Stepwise Selection models.
- The R-squared values (with & without influential points) are higher for the Full Model as compared to the Backward Elimination and Stepwise Selection models.
- The p-values (with & without influential points) are higher for the Full Model as compared to the Backward Elimination and Stepwise Selection models.
- We also notice that the Full model considers points 1, 2, 7, 15 influential points while both Backward Elimination and Stepwise Selection only considers points 1, 15 as influential points.

Overall, the Backward Elimination and Stepwise Selection models seem to perform better than the Full Model. This suggests that the Full model is overfitting the data. Using model selection procedures can help us identify the most important predictors and reduce the complexity of the model, thereby improving the model’s prediction performance.

2. Trend Across Sensitivity Analysis:

- Removing influential points seems to have a positive effect on the AIC values, R-squared values, and p-values for all models.

This suggests that the influential points were affecting the model's performance and removing them has helped improve the model's fit. It is important to identify and address influential points in the data to ensure that the model is not biased or overfitting the data.

3. Conclusion:

- Using model selection procedures, the results suggest that the best model for predicting rain seeding `Rain ~ Time` as indicated by iterating through all possible combinations of predictors, Backwards Elimination and Stepwise Selection. The Full Model seems to be overfitting the data and including unnecessary predictors, which can lead to poor prediction performance.