Assessed_Practical_2

2025-04-24

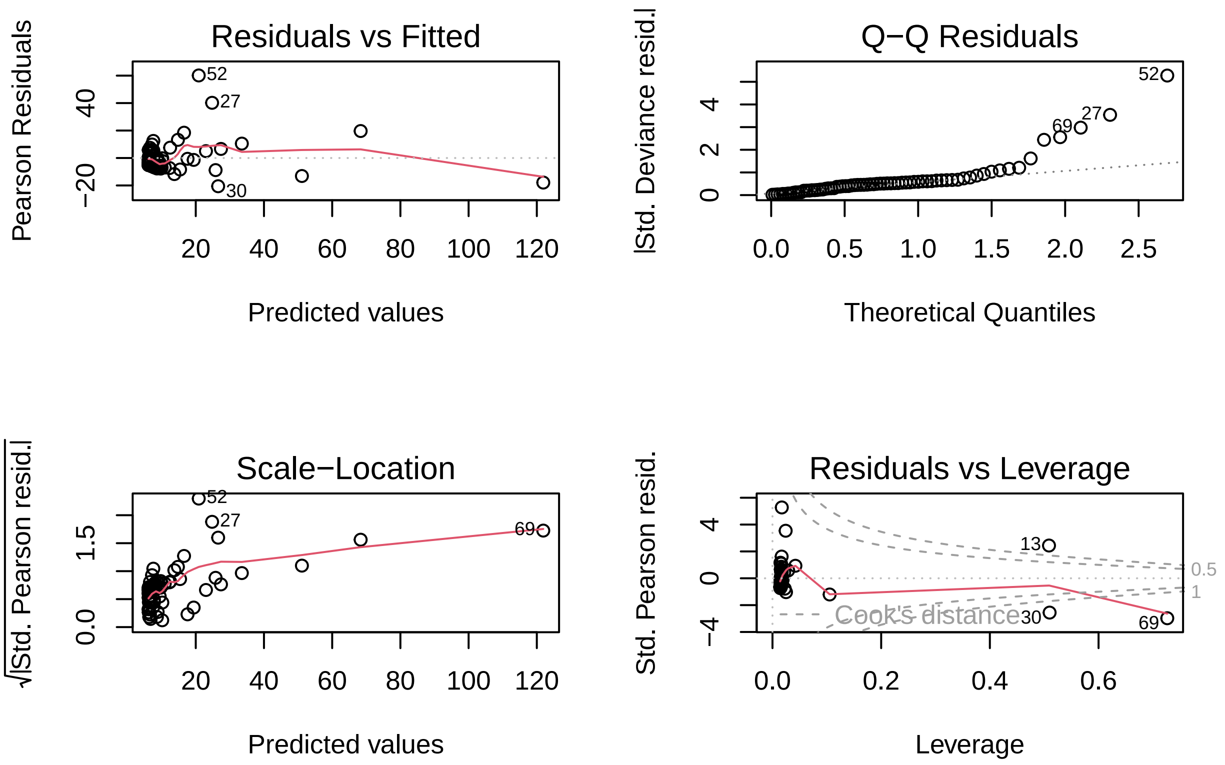
Q.18) Fit a linear regression model for the Olympic medal counts for 2012 using population and GDP as inputs and medal count as outputs. Analyse the model fit and explain if the model is appropriate for the data. Does the model assumption that the residuals follow a certain normal distribution apply for the fitted model? Do you notice that the residuals depend on the magnitude of the output values?

```
##      Country     GDP Population Medal2008 Medal2012 Medal2016
## 1    Algeria  188.68   37100000         2         1         2
## 2  Argentina  445.99   40117096         6         4         4
## 3    Armenia   10.25    3268500         6         3         4
## 4  Australia 1371.76   22880619        46        35        29
## 5 Azerbaijan   63.40    9111100         7        10        18
## 6    Bahamas    7.79     353658         2         1         2
```

```
# Fit a linear regression using glm
glm_model <- glm(Medal2012 ~ GDP + Population, family = gaussian(), data = data)

# Summarise model
summary(glm_model)
```

```
##
## Call:
## glm(formula = Medal2012 ~ GDP + Population, family = gaussian(),
##     data = data)
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.076e+00  1.500e+00   4.051 0.000133 ***
## GDP         7.564e-03  7.325e-04  10.326 1.45e-15 ***
## Population  5.247e-09  7.193e-09   0.729 0.468225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 132.1562)
##
##     Null deviance: 28402.8  on 70  degrees of freedom
## Residual deviance:  8986.6  on 68  degrees of freedom
## AIC: 553.19
##
## Number of Fisher Scoring iterations: 2
```

```r
# Check diagnostics
par(mfrow=c(2,2))
plot(glm_model)
```



The residuals vs Fitted and Scale-Location plots reveal heteroscedasticity - residual variance increases with the magnitude of predictions.

The QQ-plot shows significant deviation from the straight line, specially at higher quantiles going against the assumption of linear regression.

Q19) Consider to log transform outputs and/or inputs. Fit a linear regression model for each combination depending on whether to transform some of the inputs or not. Explain possible benefits of using the transformations.

```r
glm_logGDP <- glm(Medal2012 ~ log(GDP) + Population, family = gaussian(), data = data)
```

```r
glm_logPop <- glm(Medal2012 ~ GDP + log(Population), family = gaussian(), data = data)
```

```r
glm_logBoth <- glm(Medal2012 ~ log(GDP) + log(Population), family = gaussian(), data = data)
```

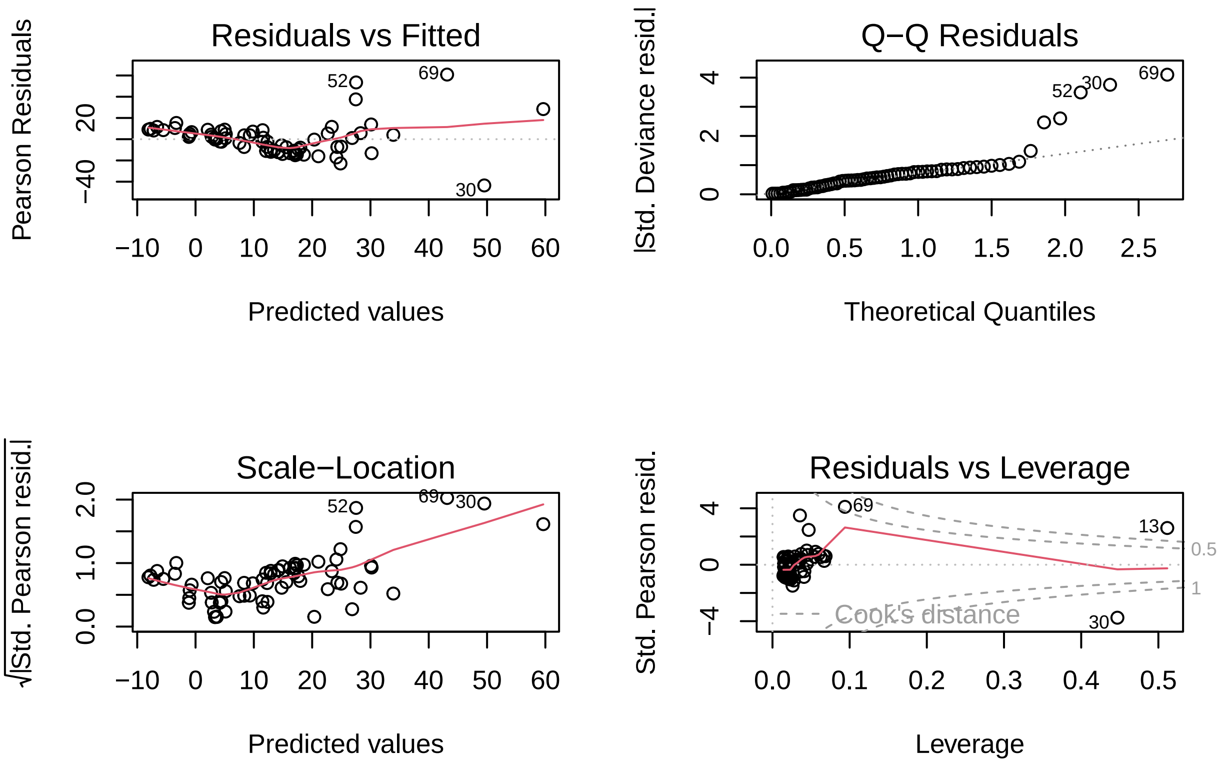
```r
glm_logOutput <- glm(log(Medal2012) ~ GDP + Population, family = gaussian(), data = data)
```

```r
glm_logOutGDP <- glm(log(Medal2012) ~ log(GDP) + Population, family = gaussian(), data = data)
```

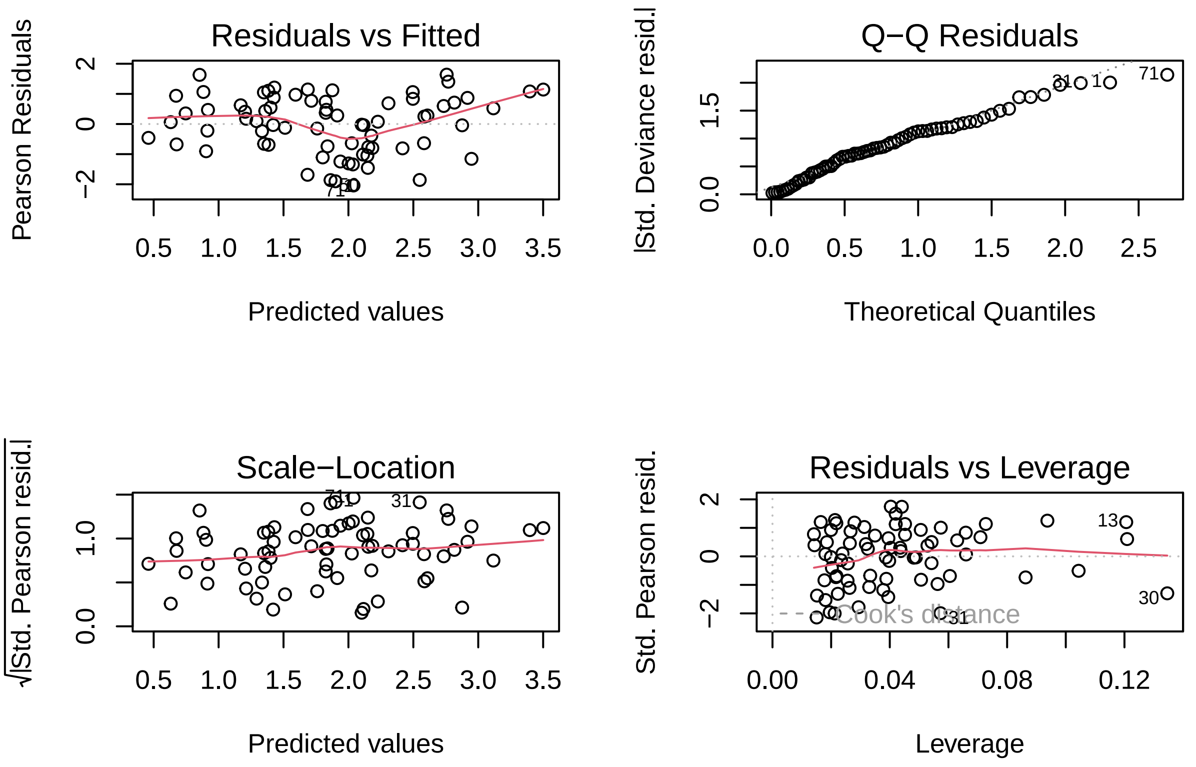
```r
glm_logOutPop <- glm(log(Medal2012) ~ GDP + log(Population), family = gaussian(), data = data)
```

```r
glm_logAll <- glm(log(Medal2012) ~ log(GDP) + log(Population), family = gaussian(), data = data)
```

```r
# Set layout to 2x2
par(mfrow = c(2, 2))

# Plot diagnostics for transformation with greatest benefit with comparison transformation
plot(glm_logGDP)
```



```r
plot(glm_logAll)
```



Log transforming can help stabilise variance, reducing heteroscedasticity - particularly if residuals previously increased with higher medal counts.

Log transformations often produce residuals closer to normality improving reliability of inference.

The glm_logAll model provides much better adherence to the assumptions of linear regression than the untransformed model:

```
- More normally distributed residuals

- More constant residual variance

- No major outliers or high-leverage points
```

This suggests that log-transforming both the output and predictors is beneficial in this context and improves model validity.

Q20) Fit Poisson regression model for the data. Consider using log transformations for the inputs. Analyse the model fit and explain if the model is appropriate for the data.

```r
# No transformation
poisson_none <- glm(Medal2012 ~ GDP + Population, family = poisson, data = data)

# Log(GDP) only
poisson_logGDP <- glm(Medal2012 ~ log(GDP) + Population, family = poisson, data = data)

# Log(Population) only
poisson_logPop <- glm(Medal2012 ~ GDP + log(Population), family = poisson, data = data)
```

```r
# Log both
poisson_logBoth <- glm(Medal2012 ~ log(GDP) + log(Population), family = poisson, data = data)
```

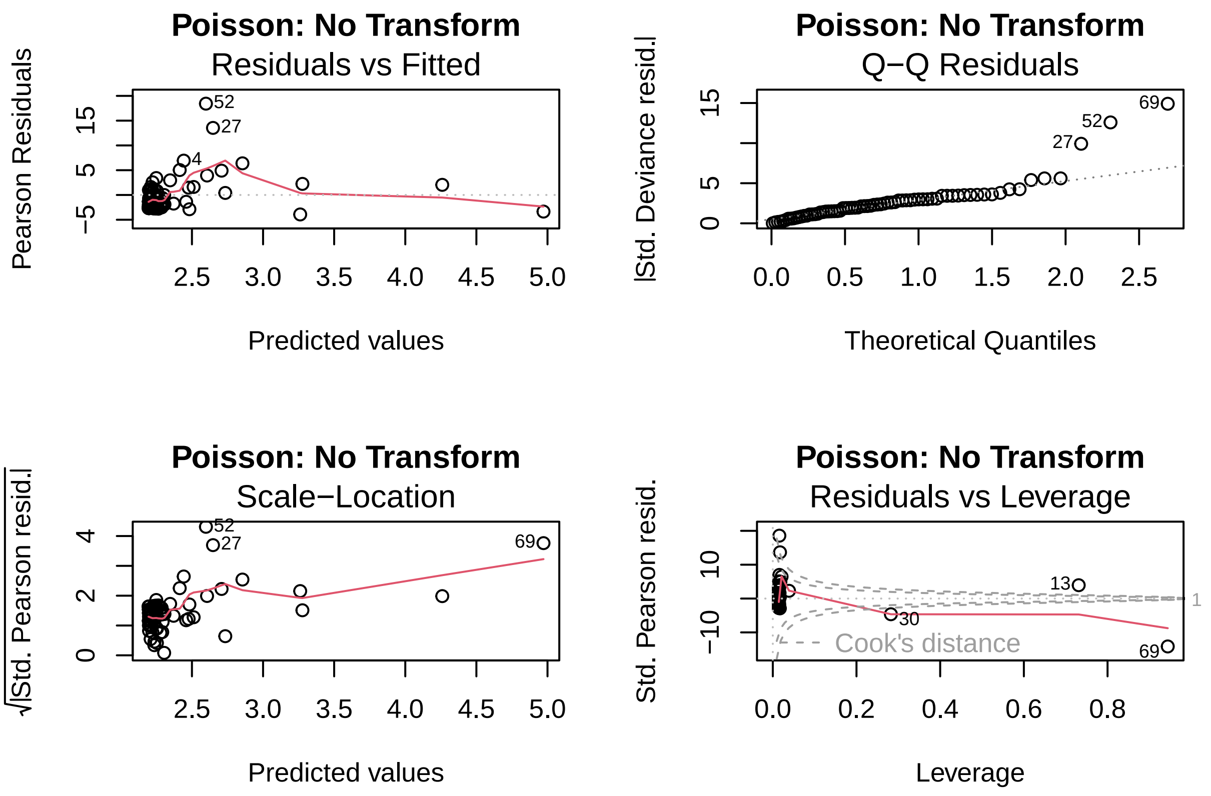
```r
summary(poisson_logBoth)
```

```
##
## Call:
## glm(formula = Medal2012 ~ log(GDP) + log(Population), family = poisson,
##     data = data)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.71251    0.42296  -4.049 5.15e-05 ***
## log(GDP)         0.48830    0.03036  16.083  < 2e-16 ***
## log(Population)  0.07388    0.03119   2.369   0.0178 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1331.81  on 70  degrees of freedom
## Residual deviance:  494.31  on 68  degrees of freedom
## AIC: 766.28
##
## Number of Fisher Scoring iterations: 5
```

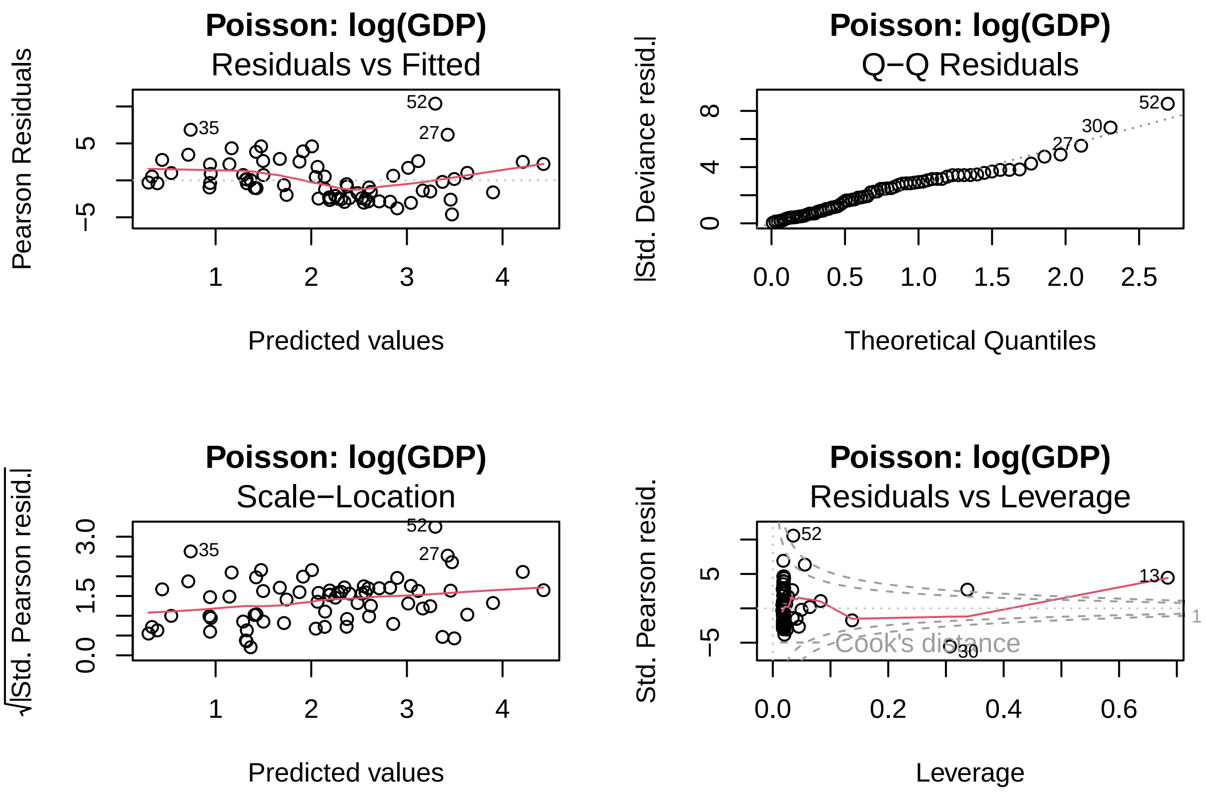
```r
# Function cal dispersion
calc_dispersion <- function(model) {
  deviance(model) / df.residual(model)
}

# Dispersion values
disp_none     <- calc_dispersion(poisson_none)
disp_logGDP   <- calc_dispersion(poisson_logGDP)
disp_logPop   <- calc_dispersion(poisson_logPop)
disp_logBoth  <- calc_dispersion(poisson_logBoth)

disp_none; disp_logGDP; disp_logPop; disp_logBoth
```

```
## [1] 10.15103

## [1] 7.309004

## [1] 8.618312

## [1] 7.26929
```

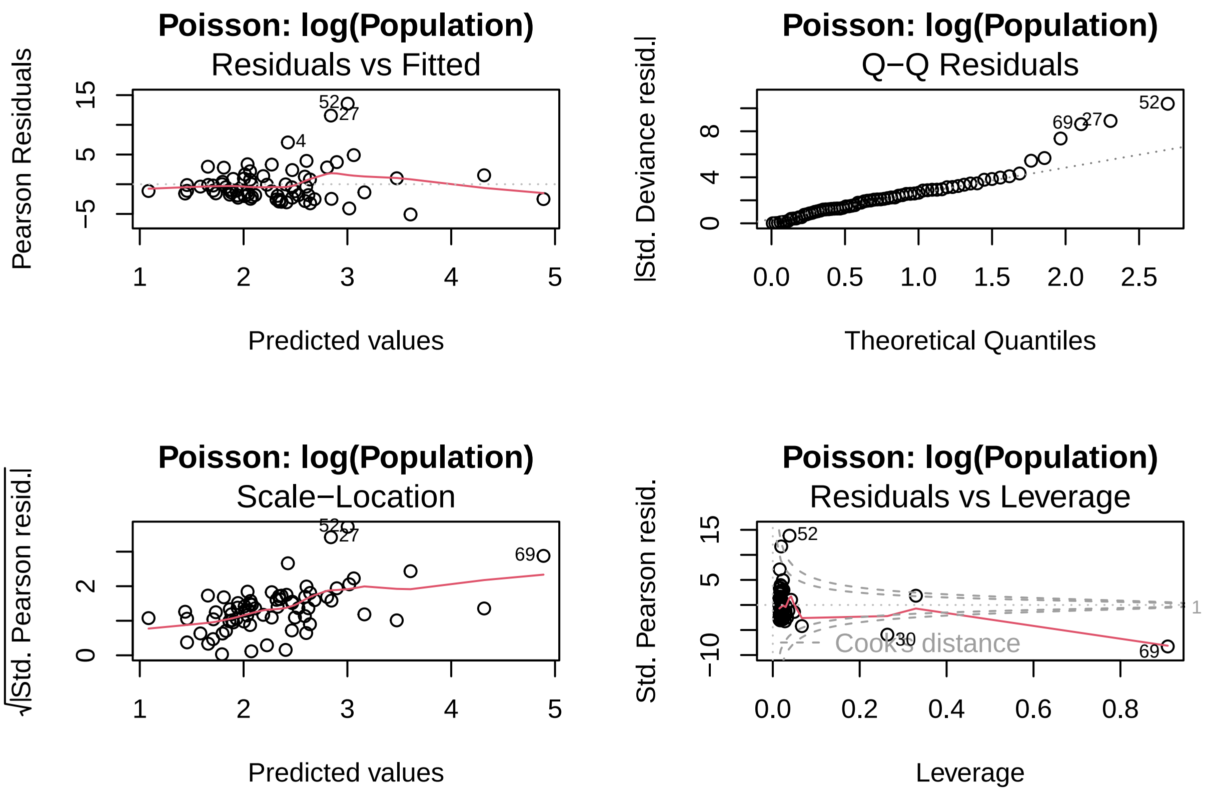
```r
# Plot for no transformation
par(mfrow = c(2, 2))
plot(poisson_none, main = "Poisson: No Transform")
```



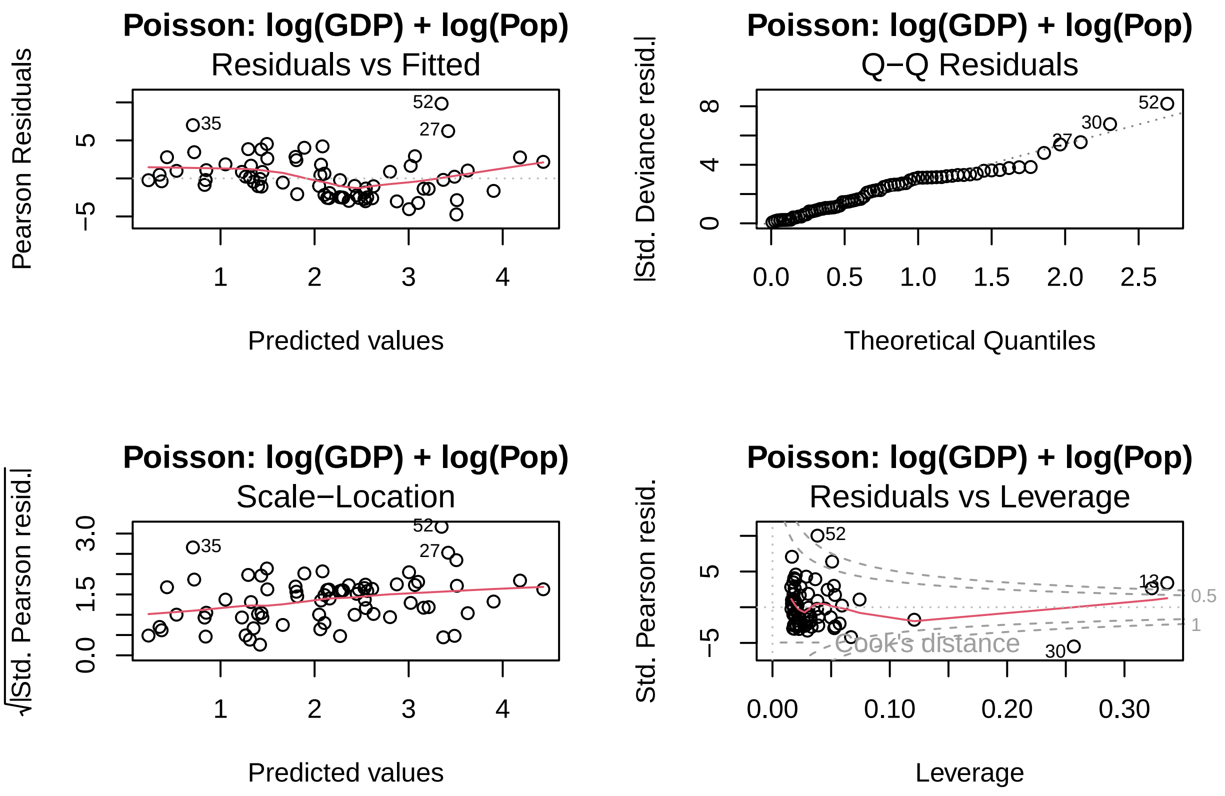
```
# Plot for log(GDP)
par(mfrow = c(2, 2))
plot(poisson_logGDP, main = "Poisson: log(GDP)")
```



```r
# Plot for log(Population)
par(mfrow = c(2, 2))
plot(poisson_logPop, main = "Poisson: log(Population)")
```



```r
# Plot for log(GDP) + log(Population)
par(mfrow = c(2, 2))
plot(poisson_logBoth, main = "Poisson: log(GDP) + log(Pop)")
```



Although the Poisson regression model using log-transformed GDP and population gave the best visual diagnostics among the candidate models, the dispersion statistic was 7.27, which is considerably higher than the expected value of 1.

This indicates substantial over dispersion, meaning the model underestimates the variability in the data. As a result, standard errors may be underestimated, and inference may not be reliable.

So, despite its improved structure and visual fit, this Poisson model is not appropriate for the data.

Q21) Inspect what `quasipoisson` family argument for `glm` means. Use that family choice to fit a quasi Poisson regression model for the data. Consider using log transformations for the inputs. Analyse and explain how this model differs from Poisson regression.

A quasi-Poisson model is used when data are counts, but the variance exceeds the mean — when the data are over dispersed.The dispersion parameter is not fixed at one, so they can model over-dispersion.

This means standard errors are adjusted to account for over dispersion, making the model more robust.

```r
glm_quasi <- glm(Medal2012 ~ GDP + Population,
                 family = quasipoisson(), data = data)

summary(glm_quasi)
```

```
##
## Call:
## glm(formula = Medal2012 ~ GDP + Population, family = quasipoisson(),
##     data = data)
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2.193e+00  1.490e-01  14.718  < 2e-16 ***
## GDP         1.715e-04  2.464e-05   6.960 1.67e-09 ***
## Population  6.049e-10  3.372e-10   1.794   0.0773 .  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 13.64172)
##
##     Null deviance: 1331.81  on 70  degrees of freedom
## Residual deviance:  690.27  on 68  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

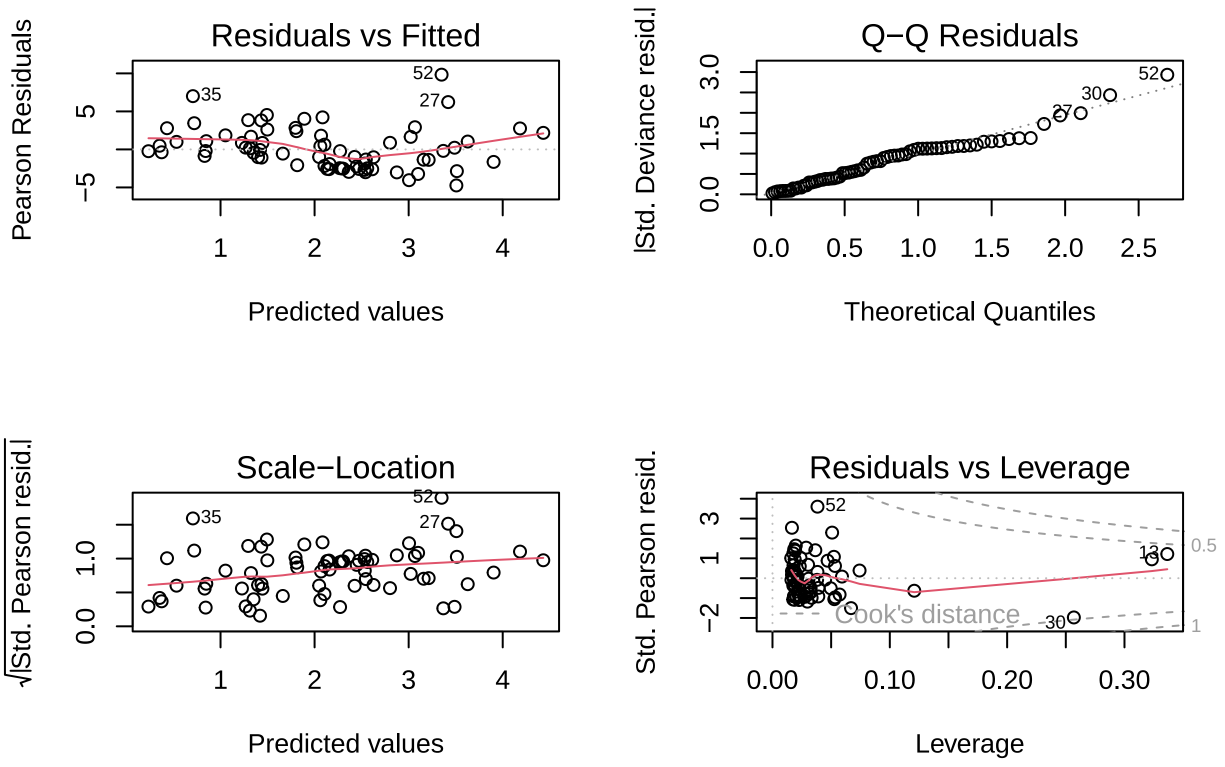
```r
glm_quasi_log <- glm(Medal2012 ~ log(GDP) + log(Population),
                 family = quasipoisson(), data = data)

summary(glm_quasi_log)
```

```
##
## Call:
## glm(formula = Medal2012 ~ log(GDP) + log(Population), family = quasipoisson(),
##     data = data)
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)    
## (Intercept)     -1.71251    1.17782  -1.454    0.151    
## log(GDP)         0.48830    0.08455   5.776 2.09e-07 ***
## log(Population)  0.07388    0.08685   0.851    0.398    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 7.754632)
##
##     Null deviance: 1331.81  on 70  degrees of freedom
## Residual deviance:  494.31  on 68  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

```r
disp_quasi_log <- summary(glm_quasi_log)$dispersion
disp_quasi_log
```

```
## [1] 7.754632
```

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



Coefficient estimates remain unchanged from the Poisson model, but the standard errors are inflated to reflect the extra variability in the data, making statistical inference (p-values, confidence intervals) more reliable.

The quasi-Poisson model is more appropriate than the standard Poisson model for this dataset. It corrects for overdispersion while preserving the count-based interpretation of medal counts and improves the reliability of inferential results.

Q22) Finally, consider using Negative Binomial regression. Consider using log transformations for the inputs. You need to use the MASS package and specify the family argument for the glm function using `family=negative.binomial(theta = thetaVal)`. Read the corresponding help files and explain in your own words the model. How the model differs from Poisson and quasi-Poisson regression models? For the predict function remember to specify type="response". In addition, try different values for `thetaVal` ranging from small (≈ 0.001) to large (≈ 1000), and comment on how the results change respectively. Find an optimal value for `thetaVal` that gives the best predictive performance.

The Negative Binomial regression model generalises Poisson regression to allow for over dispersion — the variance in the data exceeds the mean.

In this model, the variance is defined as:

$$\mathrm{Var}(Y_i) = \mu_i + \frac{\mu_i^2}{\theta}$$

This means the variance grows with the square of the mean, which is useful in count data like the Olympic medal counts where some countries dominate.

Comparing to other models:

- Poisson regression assumes:

$$\text{Var}(Y_i) = \mu_i$$

  This is often unrealistic in real-world count data.

- Quasi-Poisson regression adjusts the variance by a constant factor:

$$\text{Var}(Y_i) = \phi \cdot \mu_i$$

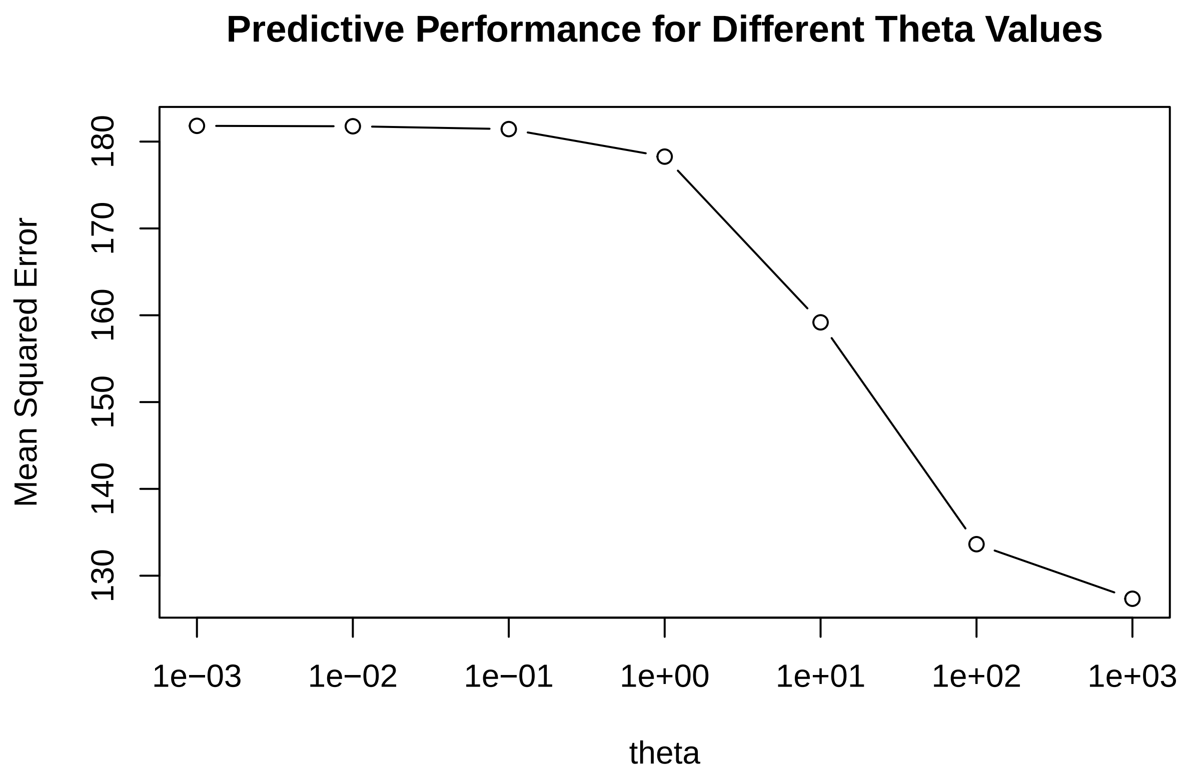
  but still assumes it scales linearly with the mean.

- Negative Binomial regression is more flexible, as the variance increases quadratically with the mean, allowing it to naturally accommodate over dispersion.

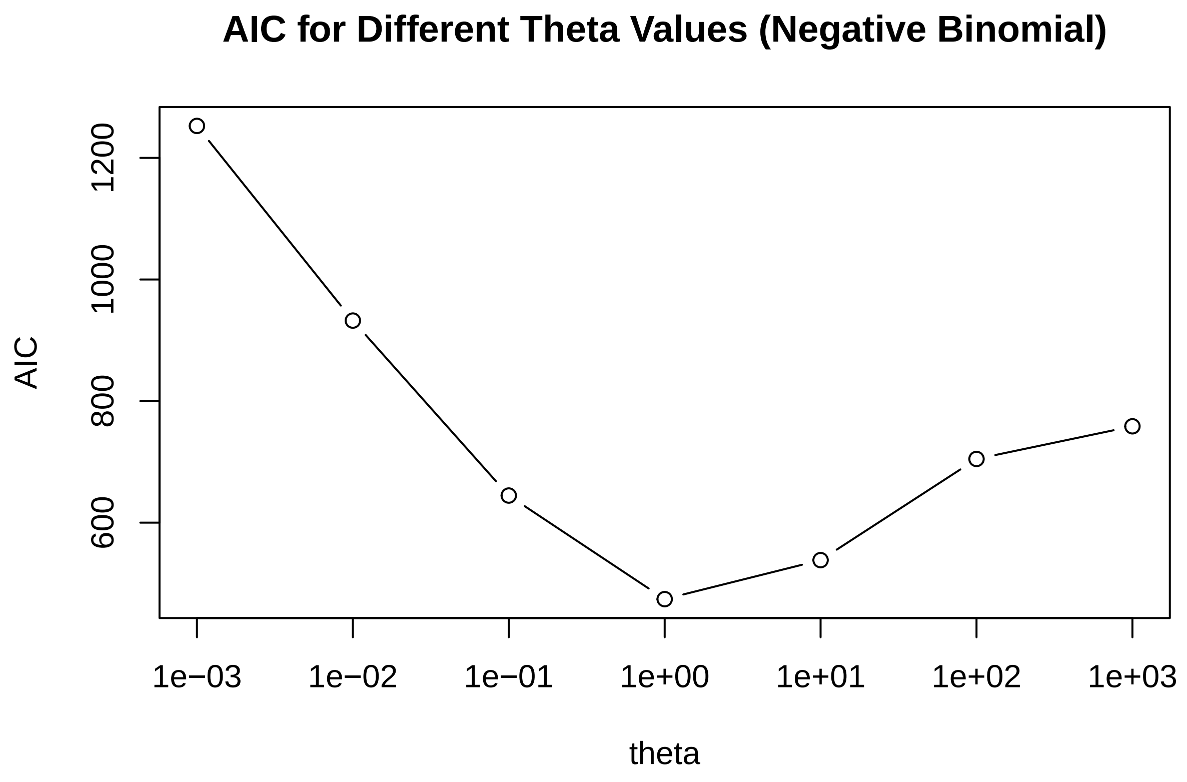
```r
# theta manually set
theta_vals <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)

# list to store models
nb_models <- list()

# Loop through theta values
for (theta in theta_vals) {
  nb_models[[as.character(theta)]] <- glm(Medal2012 ~ log(GDP) + log(Population),
                                          family = negative.binomial(theta),
                                          data = data)
}
```

```r
# Compare predictions to actual counts
pred_errors <- sapply(nb_models, function(model) {
  preds <- predict(model, type = "response")
  mean((data$Medal2012 - preds)^2)  # Mean squared error
})

# Print performance by theta
pred_errors
```

```
##    0.001     0.01      0.1        1       10      100     1000
## 181.8158 181.7688 181.4408 178.2718 159.1825 133.6303 127.3473
```



```r
# Compute AIC for each model
nb_aics <- sapply(nb_models, AIC)

# Convert theta values back to numeric for plotting
theta_numeric <- as.numeric(names(nb_aics))
```



```r
# Fit Negative Binomial regression with log-transformed inputs
nb_model <- glm.nb(Medal2012 ~ log(GDP) + log(Population), data = data)

summary(nb_model)
```

```
##
## Call:
## glm.nb(formula = Medal2012 ~ log(GDP) + log(Population), data = data,
##     init.theta = 1.797804086, link = log)
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.99881    1.28470  -1.556    0.120
## log(GDP)         0.31493    0.08079   3.898 9.68e-05 ***
## log(Population)  0.15236    0.09288   1.640    0.101
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.7978) family taken to be 1)
##
##     Null deviance: 151.926  on 70  degrees of freedom
## Residual deviance:  72.961  on 68  degrees of freedom
## AIC: 467.39
##
## Number of Fisher Scoring iterations: 1
```

```
##
##
##               Theta:  1.798
##           Std. Err.:  0.350
##
##  2 x log-likelihood:  -459.391
```

To evaluate the performance of the Negative Binomial regression model, both predictive accuracy (mean squared error) and model fit (AIC) were considered across a range of dispersion values θ.

- The MSE plot shows that prediction error decreased steadily as θ increased, with the lowest MSE observed at $\theta = 1000$. This suggests that a model with very low overdispersion performs best for prediction.
- In contrast, the AIC plot revealed a clear minimum at approximately $\theta = 1$, indicating that a moderate level of overdispersion yields the best statistical fit when balancing goodness-of-fit and model complexity.

Additionally, the `glm.nb()` function estimated the dispersion parameter to be approximately $\theta = 1.80$, which is close to the AIC-optimal value. This model adjusts for over dispersion while preserving valid statistical inference through corrected standard errors.

Summary:

- Best for inference and model fit:
  $\theta \approx 1$ (supported by AIC and `glm.nb()`)
- Best for prediction:
  $\theta = 1000$, where the model approaches Poisson behaviour

This highlights the trade-off between statistical accuracy and predictive power: larger θ improves predictions, while smaller θ better reflects the data's dispersion structure.

Question 23) Which model out of Q.18-22 would you choose for the data based on carrying out model selection using both AIC and cross-validation using test set log likelihood as performance criterion? When computing the AIC or the log likelihood for test data for cross-validation, remember to consider the (full) likelihood function of original medal counts potentially using inverse transforms for the outputs if necessary. Note: inspect the probability density function for the log-normal distribution. For Negative Binomial regression, use the model with your optimal value for `theta`.*

Table 1: AIC comparison of selected models

Model	AIC
Linear (No Transform)	553.19
Linear (log inputs and output)	200.46
Poisson (log inputs)	766.28
Negative Binomial (glm.nb)	467.39

```r
set.seed(123)

n <- nrow(data)
test_idx <- sample(1:n, size = floor(0.3 * n))
train <- data[-test_idx, ]
test <- data[test_idx, ]
```

```r
# Refit models on training data
glm_model_cv       <- glm(Medal2012 ~ GDP + Population, family = gaussian(), data = train)
glm_logAll_cv      <- glm(log(Medal2012) ~ log(GDP) + log(Population), family = gaussian(), data = trai
poisson_logBoth_cv <- glm(Medal2012 ~ log(GDP) + log(Population), family = poisson(), data = train)
quasipoisson_logBoth_cv <- glm(Medal2012 ~ log(GDP) + log(Population), family = quasipoisson(), data = 
nb_model_cv        <- MASS::glm.nb(Medal2012 ~ log(GDP) + log(Population), data = train)
```

```r
# Create helper to get test data predictions
pred_gauss       <- predict(glm_model_cv, newdata = test)
pred_logNorm     <- predict(glm_logAll_cv, newdata = test)
pred_pois        <- predict(poisson_logBoth_cv, newdata = test, type = "response")
pred_quasi       <- predict(quasipoisson_logBoth_cv, newdata = test, type = "response")
pred_nb          <- predict(nb_model_cv, newdata = test, type = "response")
```

```r
# For Gaussian (linear)
ll_glm <- sum(dnorm(test$Medal2012, mean = pred_gauss, sd = sd(residuals(glm_model_cv)), log = TRUE))

# For log-normal: back-transform predictions
log_mean <- pred_logNorm
log_sd <- sd(residuals(glm_logAll_cv))
ll_lognorm <- sum(dlnorm(test$Medal2012 + 1, meanlog = log_mean, sdlog = log_sd, log = TRUE))

# For Poisson
ll_pois <- sum(dpois(test$Medal2012, lambda = pred_pois, log = TRUE))

# For Quasi-Poisson (use same as Poisson for likelihood comparison, as they share the mean structure)
ll_quasi <- sum(dpois(test$Medal2012, lambda = pred_quasi, log = TRUE))

# For Negative Binomial
ll_nb <- sum(dnbinom(test$Medal2012, mu = pred_nb, size = nb_model_cv$theta, log = TRUE))
```

Table 2: Cross-Validation: Test Set Log-Likelihood Comparison

Model	Test_LogLikelihood
Linear (No Transform)	-135.33
Linear (log inputs + output)	-81.79
Poisson (log inputs)	-186.77
Quasi-Poisson (log inputs)	-186.77
Negative Binomial (glm.nb)	-81.75

```r
library(MASS)

crossval_loglik_summary <- function(data, n_repeats = 100, test_prop = 0.3, seed = 123) {
  set.seed(seed)

  results <- matrix(NA, nrow = n_repeats, ncol = 5)
  colnames(results) <- c("Linear", "Log-Linear", "Poisson", "Quasi-Poisson", "NegBin")

  for (i in 1:n_repeats) {
    # Split data
    n <- nrow(data)
    test_idx <- sample(1:n, size = floor(test_prop * n))
    train <- data[-test_idx, ]
    test  <- data[test_idx, ]

    # Fit models on training data
    m1 <- glm(Medal2012 ~ GDP + Population, family = gaussian(), data = train)
    m2 <- glm(log(Medal2012 + 1) ~ log(GDP) + log(Population), family = gaussian(), data = train)
    m3 <- glm(Medal2012 ~ log(GDP) + log(Population), family = poisson(), data = train)
    m4 <- glm(Medal2012 ~ log(GDP) + log(Population), family = quasipoisson(), data = train)
    m5 <- glm.nb(Medal2012 ~ log(GDP) + log(Population), data = train)

    # Predict on test data
    pred1 <- predict(m1, newdata = test)
    pred2 <- predict(m2, newdata = test)
    pred3 <- predict(m3, newdata = test, type = "response")
    pred4 <- predict(m4, newdata = test, type = "response")
    pred5 <- predict(m5, newdata = test, type = "response")

    # Log-likelihood calculations
    ll1 <- sum(dnorm(test$Medal2012, mean = pred1, sd = sd(residuals(m1)), log = TRUE))
    ll2 <- sum(dlnorm(test$Medal2012 + 1, meanlog = pred2, sdlog = sd(residuals(m2)), log = TRUE))
    ll3 <- sum(dpois(test$Medal2012, lambda = pred3, log = TRUE))
    ll4 <- sum(dpois(test$Medal2012, lambda = pred4, log = TRUE))  # quasi-poisson uses same lambda
    ll5 <- sum(dnbinom(test$Medal2012, mu = pred5, size = m5$theta, log = TRUE))

    # Store results
    results[i, ] <- c(ll1, ll2, ll3, ll4, ll5)
  }

  # Summarise results
  summary_df <- data.frame(
    Model = colnames(results),
    Mean_LogLikelihood = apply(results, 2, mean),
    SD_LogLikelihood = apply(results, 2, sd)
  )

  return(summary_df)
}
```

Table 3: Repeated Cross-Validation: Mean and SD of Test Log-Likelihoods

	Model	Mean_LogLikelihood	SD_LogLikelihood
Linear	Linear	-100.84	33.68
Log-Linear	Log-Linear	-69.96	4.25
Poisson	Poisson	-124.67	24.93
Quasi-Poisson	Quasi-Poisson	-124.67	24.93
NegBin	NegBin	-69.55	4.11

To determine the most suitable model for explaining Olympic medal counts, I considered all candidate models (Q18–Q22) using two performance criteria:

1. Akaike Information Criterion (AIC) — assessing in-sample model fit.

2. Repeated cross-validation test set log-likelihood — evaluating predictive accuracy and robustness across random train/test splits.

Test Set Log-Likelihood (100 Repeats)

Table 4: Repeated Cross-Validation: Mean and SD of Test Log-Likelihoods

	Model	Mean_LogLikelihood	SD_LogLikelihood
Linear	Linear	-100.84	33.68
Log-Linear	Log-Linear	-69.96	4.25
Poisson	Poisson	-124.67	24.93
Quasi-Poisson	Quasi-Poisson	-124.67	24.93
NegBin	NegBin	-69.55	4.11

The best mean predictive performance was achieved by the Negative Binomial model with log-transformed inputs (mean log-likelihood = -69.55).

The log-linear model (linear regression on log-transformed inputs and output) performed almost as well (-69.96), and had similarly low variability across splits (SD ~ 4.25).

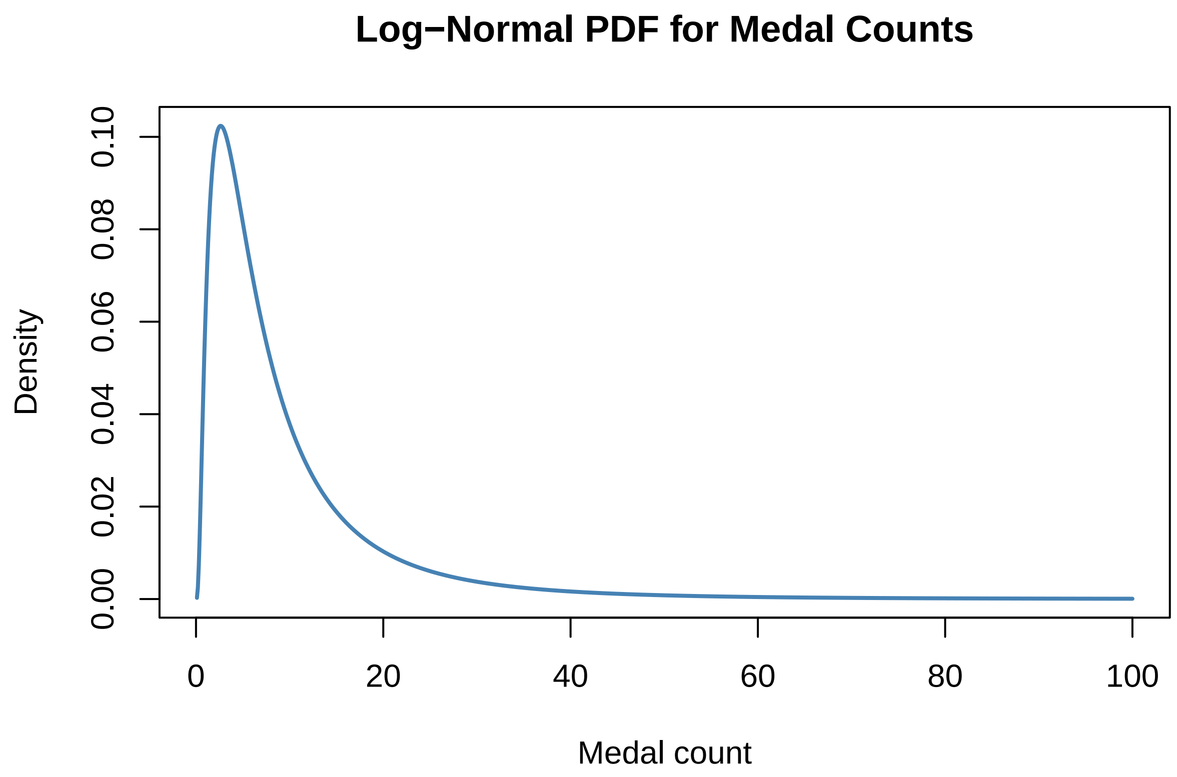
Both models outperformed Poisson, Quasi-Poisson, and untransformed linear regression by a significant margin.

AIC Results: The log-linear model (glm_logAll) had the lowest AIC, indicating it provides the best trade-off between model fit and complexity.

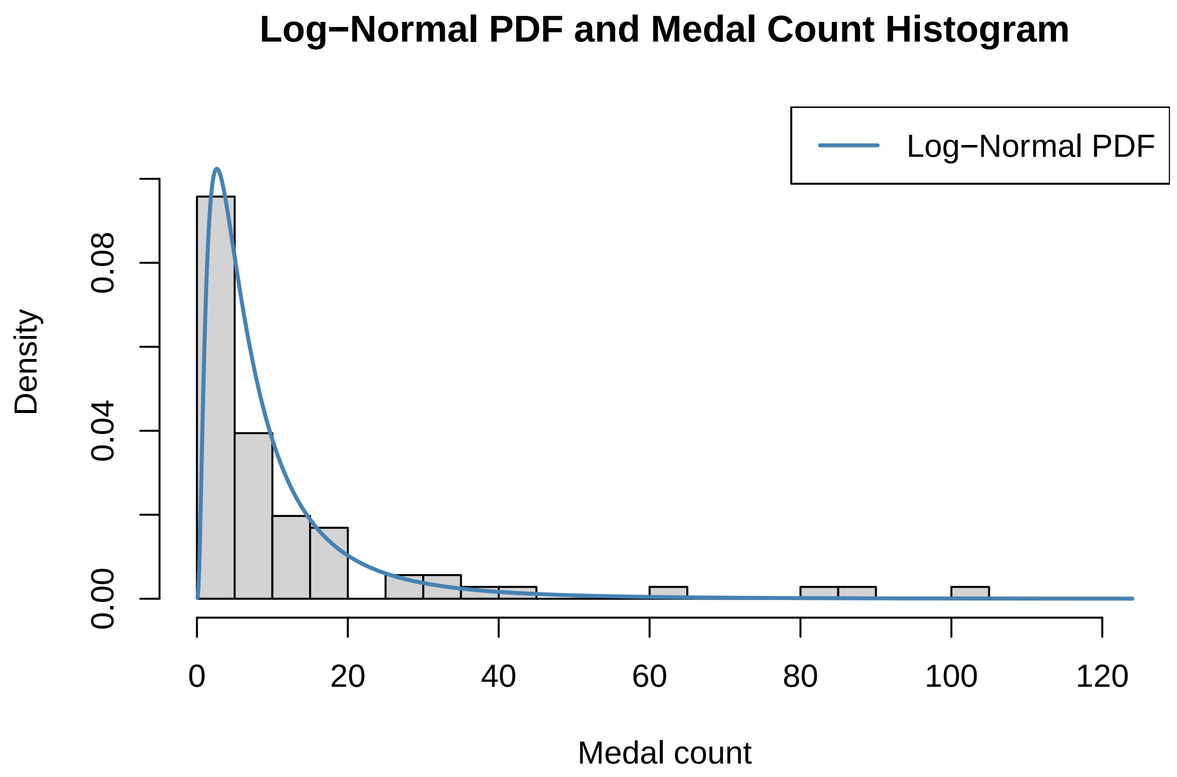
Final Conclusion: If the focus is on predictive performance, the Negative Binomial model is slightly better, with the highest mean log-likelihood and lowest variance.

However, given that the log-linear model achieved nearly identical results and had the lowest AIC, it may be the better overall choice for both interpretability and performance.

The recommended model is the log-linear regression with log-transformed GDP, population, and medal count — it provides strong predictive accuracy, excellent model fit, and a simpler, interpretable structure.



```r
# Extract model predictions and residuals (on log scale)
meanlog <- mean(predict(glm_logAll))
sdlog <- sd(residuals(glm_logAll))

# Sequence of x-values for the density curve
x_vals <- seq(0.1, max(data$Medal2012) + 20, by = 0.1)
pdf_vals <- dlnorm(x_vals, meanlog = meanlog, sdlog = sdlog)

# Plot histogram of actual medal counts
hist(data$Medal2012, breaks = 20, freq = FALSE, col = "lightgray",
     xlim = c(0, max(x_vals)), ylim = c(0, max(pdf_vals) * 1.1),
     xlab = "Medal count", main = "Log-Normal PDF and Medal Count Histogram")

# Overlay log-normal PDF
lines(x_vals, pdf_vals, col = "steelblue", lwd = 2)
legend("topright", legend = "Log-Normal PDF", col = "steelblue", lwd = 2)
```



The log-normal model approximates the actual distribution pretty well — especially for the lower medal counts (which are the majority).

It underestimates some of the extreme medal counts (like the few countries with 60–100+ medals).

Question 24) In addition to quantitative model selection, analyse and inspect the models to determine which model would you choose to best represent the data. Do you agree with the results from quantitative model selection? Justify your answer.

Log-linear model:

- Residual plots were generally well-behaved after log-transforming both the response and predictors.
- The Q-Q plot and Scale-Location plots indicated approximately normal residuals, satisfying linear regression assumptions.
- The log-normal PDF overlay showed that the predicted distribution closely matched the true medal count distribution.
- The model is simple, interpretable, and transformations are statistically justifiable.

Negative Binomial model:

- This model appropriately handles overdispersion and uses a distribution tailored for count data.
- Residual plots showed modest improvements over the Poisson model but were still influenced by high-leverage points.
- While it provided the best predictive performance on average, it is slightly more complex and less interpretable compared to the log-linear model.

The log-linear model is the best overall choice: - It aligns with linear model assumptions after log-transformations. - It is simpler and more interpretable, especially for understanding the relationship between GDP, population, and medal counts. - Its performance was comparable or better than more complex models both quantitatively and qualitatively.

Q25) Consider using both mean squared error and mean absolute error instead of test set log likelihood for cross-validation criteria. Do the results change respectively? Explain reasons for any potential differences.

```r
library(MASS)

crossval_mse_mae_summary <- function(medal2012, n_repeats = 100, test_prop = 0.3, seed = 123) {
  set.seed(seed)

  models <- c("Linear", "Log-Linear", "Poisson", "Quasi-Poisson", "NegBin")
  mse_mat <- matrix(NA, nrow = n_repeats, ncol = length(models))
  mae_mat <- matrix(NA, nrow = n_repeats, ncol = length(models))

  for (i in 1:n_repeats) {
    # Split
    n <- nrow(medal2012)
    test_idx <- sample(seq_len(n), size = floor(test_prop * n))
    train <- medal2012[-test_idx, ]
    test  <- medal2012[test_idx, ]

    # Fit models
    m1 <- glm(Medal2012 ~ GDP + Population, family = gaussian(), data = train)
    m2 <- glm(log(Medal2012) ~ log(GDP) + log(Population), family = gaussian(), data = train)
    m3 <- glm(Medal2012 ~ log(GDP) + log(Population), family = poisson(), data = train)
    m4 <- glm(Medal2012 ~ log(GDP) + log(Population), family = quasipoisson(), data = train)
    m5 <- glm.nb(Medal2012 ~ log(GDP) + log(Population), data = train)

    # Predict on test data
    pred1 <- predict(m1, newdata = test)

    # Log-normal with bias correction
    log_preds <- predict(m2, newdata = test)
    log_sd <- sd(residuals(m2))  # SD of residuals on log scale
    pred2 <- exp(log_preds + 0.5 * log_sd^2)  # bias-corrected inverse transform

    pred3 <- predict(m3, newdata = test, type = "response")
    pred4 <- predict(m4, newdata = test, type = "response")
    pred5 <- predict(m5, newdata = test, type = "response")

    preds <- list(pred1, pred2, pred3, pred4, pred5)

    for (j in 1:length(preds)) {
      mse_mat[i, j] <- mean((test$Medal2012 - preds[[j]])^2)
      mae_mat[i, j] <- mean(abs(test$Medal2012 - preds[[j]]))
    }
  }
  # Combine and summarise
  summary_df <- data.frame(
    Model = models,
    Mean_MSE = round(colMeans(mse_mat), 2),
    SD_MSE = round(apply(mse_mat, 2, sd), 2),
    Mean_MAE = round(colMeans(mae_mat), 2),
    SD_MAE = round(apply(mae_mat, 2, sd), 2)
  )

  return(summary_df)
}
```

```r
mse_mae_summary <- crossval_mse_mae_summary(data)
knitr::kable(mse_mae_summary, caption = "MSE and MAE Cross validation", digits = 2)
```

Summary of Results:

Table 5: MSE and MAE Cross validation

Model	Mean_MSE	SD_MSE	Mean_MAE	SD_MAE
Linear	278.19	258.72	8.74	2.83
Log-Linear	209.90	129.16	8.90	2.12
Poisson	158.93	91.50	8.17	1.82
Quasi-Poisson	158.93	91.50	8.17	1.82
NegBin	195.46	116.20	8.56	2.05

The Poisson and Quasi-Poisson models had the lowest MSE and MAE, suggesting strong point prediction performance.

The Log-Linear model performed worse than Poisson but better than Linear.

The Negative Binomial model had intermediate performance — better than Linear, but worse than Poisson on average.

The Linear model had both the highest error and largest variability across folds.

Log-likelihood considers not just point predictions but also distributional fit (e.g. variance structure and uncertainty).

MSE/MAE are purely about distance from the actual value, and may favour simpler models that under report uncertainty.

For example, Poisson assumes that the mean = variance, which may lead to sharper predictions (lower MSE) even if it underfits overdispersion.

While MSE/MAE favour Poisson and Quasi-Poisson models, these metrics do not fully capture the count nature of the data or its variance. The Log-Linear and Negative Binomial models still provide better distributional assumptions and likelihood fit, as shown in earlier questions.

So in conclusion, I acknowledge the value of MSE/MAE, but they support slightly different conclusions from log-likelihood-based selection. The overall model choice should balance both predictive error and distributional appropriateness.

Q26) Derive and compute the probability that GB wins at least one medal given the estimated model parameters using all data for the regression models. You may approximate your solution by generating data from the model. Do the models provide similar answers?

```r
library(MASS)


gb_row <- data[data$Country == "Great Britain", ]

# Fit all models on the full dataset
m1 <- glm(Medal2012 ~ GDP + Population, family = gaussian(), data = data)
m2 <- glm(log(Medal2012) ~ log(GDP) + log(Population), family = gaussian(), data = data)
m3 <- glm(Medal2012 ~ log(GDP) + log(Population), family = poisson(), data = data)
m4 <- glm(Medal2012 ~ log(GDP) + log(Population), family = quasipoisson(), data = data)
m5 <- glm.nb(Medal2012 ~ log(GDP) + log(Population), data = data)


mu1 <- as.numeric(predict(m1, newdata = gb_row))
sigma1 <- sqrt(sum(residuals(m1, type = "deviance")^2) / df.residual(m1))
sim_gauss <- rnorm(10000, mean = mu1, sd = sigma1)
p_gauss <- mean(sim_gauss >= 1)


log_mu <- as.numeric(predict(m2, newdata = gb_row))
log_sd <- sd(residuals(m2))
sim_lognorm <- rlnorm(10000, meanlog = log_mu, sdlog = log_sd)
p_lognorm <- mean(sim_lognorm >= 1)


lambda_pois <- as.numeric(predict(m3, newdata = gb_row, type = "response"))
p_pois <- 1 - dpois(0, lambda_pois)


lambda_qpois <- as.numeric(predict(m4, newdata = gb_row, type = "response"))
dispersion_qp <- summary(m4)$dispersion

sim_lambda_qp <- rnorm(10000, mean = lambda_qpois, sd = sqrt(dispersion_qp * lambda_qpois))
sim_lambda_qp <- pmax(sim_lambda_qp, 0)  # avoid negative lambdas
sim_qpois <- rpois(10000, lambda = sim_lambda_qp)
p_qpois <- mean(sim_qpois >= 1)


mu_nb <- as.numeric(predict(m5, newdata = gb_row, type = "response"))
theta_nb <- m5$theta
p_nb <- 1 - dnbinom(0, size = theta_nb, mu = mu_nb)


prob_table <- data.frame(
  Model = c("Gaussian", "Log-Normal", "Poisson", "Quasi-Poisson", "Negative Binomial"),
  `P(Y   1)` = c(p_gauss, p_lognorm, p_pois, p_qpois, p_nb)
)

knitr::kable(prob_table, digits = 4, caption = "Estimated Probability GB Wins   1 Medal (Q26)")
```

Table 6: Estimated Probability GB Wins 1 Medal (Q26)

Model	P.Y...1.
Gaussian	0.9823
Log-Normal	0.9986
Poisson	1.0000
Quasi-Poisson	0.9753
Negative Binomial	0.9919

All models estimated a very high probability (97%) that GB would win at least one medal.

The Poisson model predicted a probability of 1, reflecting its discrete nature and the relatively high expected count for GB.

The Log-Normal model estimated the highest among the continuous models (0.9985), followed closely by Gaussian and Negative Binomial.

The Quasi-Poisson model, despite accounting for over dispersion, gave the lowest probability (0.9723), though still very high.

All models provide consistent estimates of a high probability, differing by no more than ~3%. This agreement suggests that GB's medal count is far enough from zero that model choice does not drastically affect this particular probability estimate.

The models all support the conclusion that GB winning at least one medal was almost certain in 2012, regardless of the modelling approach.

Question 27) Note that countries that do not have any medals were excluded from the dataset. Consider how this affects making inferences from the data. What would you expect to change if you included all participating countries in the model? Discuss how log transformations could be used for this full dataset.

By excluding countries that won zero medals this introduces a form of selection bias.

Effects on Inference:

- The models are effectively fitted to countries that were already successful.
- This leads to overestimation of baseline medal counts: the intercept and coefficients are biased upwards, because the model is only trained on countries that earned at least one medal.
- As a result, predictions for smaller countries or those with lower GDP/population will be less reliable, as these situations were underrepresented or excluded altogether.

Including All Countries: If all participating countries were included, including those with 0 medals: - It's possible that the model may learn that low-GDP and low-population countries often win 0 medals, improving prediction accuracy in the lower range. - The distribution of outcomes would become more skewed toward 0, especially for Poisson-like models. - For linear regression, the inclusion of many 0s would likely violate homoscedasticity and normality assumptions even more severely.

Log Transformations with 0 Values: A issue arises: you cannot take the log of 0.

To get around this challenge you could: 1. Add a small constant - It allows including zeros but may distort the scale for small counts. 2. Use models that naturally handle zeros, like Poisson or Negative Binomial - These models are more robust when modeling count data that includes zeros. - They do not require transforming the response.

Conclusion: Including countries with 0 medals would improve generalisability and reduce bias, but requires careful handling of zeros. Count-based models like Poisson or Negative Binomial are better suited for such data sets, while log-transformations need adjustments like `log(y + 1)` to remain valid.

Question 28) Are the medal counts independent of each other over different countries? Is the total number of medals a random variable or fixed to some value? How the competition aspect of winners and losers affects the medal counts over different countries? Based on your answers explain how suitable the aforementioned regression models are for the data. Are some of the model assumptions violated? Consider both interpretability and quality of predictions.

Independence of Medal Counts In the models, we assumed that each country's medal count is independent of others. However, this assumption is likely violated in reality: - The total number of medals is fixed in the Olympics — if one country wins more, others must win fewer. - This creates a competitive constraint that induces negative dependence between countries.

This dependency structure means that treating countries' medal counts as independent may lead to overconfident predictions and underestimation of uncertainty.

Random or Fixed Total?

- The total number of medals awarded is fixed, it is based on the number of events and categories.
- While individual event outcomes may be probabilistic, the overall sum of medals is constrained.
- Therefore, modelling each country's medal count as an independent random variable overlooks this zero-sum nature.

Effects of Competition

- The Olympics is inherently competitive: only one gold, silver, and bronze can be awarded per event.
- Models that assume individual countries accumulate medals independently ignore this zero-sum competition.
- This may bias predictions, especially for dominant countries whose performance suppresses that of others.

Suitability of Models

- Despite these theoretical issues, models like Poisson and Negative Binomial have provided reasonable predictive performance, especially when covariates (e.g., GDP, population) explain much of the variation.
- Linear and log-linear models, however, may struggle with heteroscedasticity, interpretability, and overprediction for smaller nations.

Violated Assumptions:

- Independence: Violated due to competition.
- Fixed total: Ignored in models
- Normality of residuals: Violated for linear models.

Interpretability vs Accuracy

- Linear/log-linear models are easy to interpret but perform poorly.
- Negative Binomial models are less interpretable but better capture overdispersion and count structure.
- Poisson models often underestimate variance but are conceptually appropriate for count data.

Conclusion: While regression models provide useful approximations, they simplify the true structure of Olympic medal allocation. The independence and fixed-total assumptions are violated, which limits the models' theoretical validity. Still, Negative Binomial regression strikes a balance between predictive accuracy and realism, especially when paired with well-chosen covariates.