Olympic Medal Analysis

Yuanrong Liu

2024-09-14

Packages Loaded

```r
## BMB: avoid chaff by setting message=FALSE in chunk options ...
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(splines)
library(sandwich)
library(lmtest)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(dotwhisker)
library(broom)
library(dplyr)
library(effects)
```

```
## Loading required package: carData
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```r
library(ggplot2)
```

Question 1

Introduction

In this document, we will analyze the Olympic `medal` dataset, focusing on gold medals and using `GDP` and `population` data as predictors.

Part a: Selection of Variables

- In this analysis, we are going to include GDP(`gdp`) and population(`pop`) as predictor variables, and `gold medals` as the responses variable. Gold metals tend to represent the highest achievements at the Olympics. GDP can significantly impact the training and development of athletes, infrastructure, and support systems for sports. Population size reflects the potential talent pool from which a country can draw athletes. As a result, GDP and populations have been widely used in sports research as a predictor of international success, as they are closely related to the resources available to support Olympic athletes.

```r

# Load the Olympic dataset
mydat <- read.csv(url("https://raw.githubusercontent.com/bbolker/stats720/main/data/olymp1.csv"))

# Filter for gold medals
newdata <- mydat |> filter(medal == "Gold")

# Clean the data by dropping rows with missing values for GDP and population
newdata_clean <- newdata |> drop_na(gdp, pop)

## BMB: comment on whether/how dropping incomplete cases will affect results?
```

Load and Clean the Data

Log-Transforming the Variables

- Log-transforming predictors like GDP and population can make the relationships between variables more linear and easier to model. By log-transforming, you're modeling the proportional changes in the predictors (GDP and population) rather than absolute changes.
- If the distribution of the number of gold medals is highly skewed (with many zeros and a few large values), log-transforming the response variable `n` can help to stabilize variance and make the data more suitable for linear modeling.

```r
newdata_clean <- newdata_clean |>
  ## BMB: don't really need +1 for positive-valued variables (gdp, pop) ?
  mutate(log_gdp = log(gdp + 1),
         log_pop = log(pop + 1),
         log_n = log(n + 1))
```

Splines for Non-Linear Relationships

- If the relationships between GDP, population, and gold medals are non-linear, using natural splines allows you to model these relationships in a flexible way without overfitting.

```r
# Natural spline with 5 degrees of freedom for GDP and Population
model <- lm(n ~ ns(gdp, df=5) + ns(pop, df=5), data = newdata_clean)
```

```r
model_log <- lm(log_n ~ ns(log_gdp, df = 5) + ns(log_pop, df = 5), data = newdata_clean)
summary(model)
```

```
##
## Call:
## lm(formula = n ~ ns(gdp, df = 5) + ns(pop, df = 5), data = newdata_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.255  -2.744  -0.988   0.011  56.090
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.5102     1.5869   0.322 0.747948
## ns(gdp, df = 5)1     2.6381     1.9494   1.353 0.176529
## ns(gdp, df = 5)2     2.3117     1.8024   1.283 0.200209
## ns(gdp, df = 5)3   67.6577     5.2882  12.794  < 2e-16 ***
## ns(gdp, df = 5)4 103.2563     4.6283  22.310  < 2e-16 ***
## ns(gdp, df = 5)5 137.0702     5.5250  24.809  < 2e-16 ***
## ns(pop, df = 5)1    1.8279     1.8393   0.994 0.320760
## ns(pop, df = 5)2   -0.9251     2.0050  -0.461 0.644726
## ns(pop, df = 5)3    9.0549     6.6101   1.370 0.171312
## ns(pop, df = 5)4   -3.8883     5.1048  -0.762 0.446579
## ns(pop, df = 5)5 -13.8685     3.6854  -3.763 0.000187 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.293 on 530 degrees of freedom
## Multiple R-squared:  0.7486, Adjusted R-squared:  0.7438
## F-statistic: 157.8 on 10 and 530 DF,  p-value: < 2.2e-16
```

```r
summary(model_log)
```

```
##
## Call:
## lm(formula = log_n ~ ns(log_gdp, df = 5) + ns(log_pop, df = 5),
##     data = newdata_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0379 -0.5965 -0.1928  0.4236  2.7712
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -0.01008    0.29214  -0.034 0.972498
## ns(log_gdp, df = 5)1  0.73249    0.28605   2.561 0.010722 *
## ns(log_gdp, df = 5)2  0.44136    0.34132   1.293 0.196541
## ns(log_gdp, df = 5)3  1.66490    0.26931   6.182 1.26e-09 ***
## ns(log_gdp, df = 5)4  4.35641    0.69392   6.278 7.14e-10 ***
## ns(log_gdp, df = 5)5  6.00314    0.40333  14.884  < 2e-16 ***
## ns(log_pop, df = 5)1  0.18478    0.24056   0.768 0.442756
## ns(log_pop, df = 5)2  0.05017    0.28107   0.178 0.858406
## ns(log_pop, df = 5)3  0.13257    0.27250   0.487 0.626813
## ns(log_pop, df = 5)4 -0.90083    0.54506  -1.653 0.098979 .
## ns(log_pop, df = 5)5 -1.34565    0.34611  -3.888 0.000114 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.859 on 530 degrees of freedom
## Multiple R-squared:  0.506,  Adjusted R-squared:  0.4967
## F-statistic: 54.28 on 10 and 530 DF,  p-value: < 2.2e-16
```

BMB: (1) better to incorporate numeric values programmatically (e.g. 0.749)

- The **original model** had a higher R-squared (**0.7486**), explaining 74.86% of the variability in gold medals, but with a relatively high residual standard error (RSE) of **8.293**, indicating issues with extreme values. In contrast, the **log-transformed model** had a lower R-squared (**0.506**), explaining 50.6% of the variability, but a much lower RSE (**0.859**), suggesting a more stable fit when modeling proportional changes. Significant coefficients for the log-transformed GDP splines (e.g., **6.00314** for `ns(log_gdp, df = 5)5`) highlight a strong non-linear relationship between GDP and gold medals.
- The **log-transformed model** better captures proportional changes, making it the preferred choice for predicting gold medals based on GDP and population.

Interaction

- The interaction between **GDP** and **population** might influence the number of gold medals (i.e., countries with large populations and high GDPs perform better).

```r
# Include interaction between log_gdp and log_pop
model_interaction <- lm(log_n ~ ns(log_gdp, df = 5) * ns(log_pop, df = 5), data = newdata_clean)
summary(model_interaction)
```

```
##
## Call:
## lm(formula = log_n ~ ns(log_gdp, df = 5) * ns(log_pop, df = 5),
##     data = newdata_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3876 -0.4965 -0.1575  0.3591  2.8011
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                4.092e-02  3.740e-01   0.109
## ns(log_gdp, df = 5)1                      -1.176e+00  1.254e+00  -0.937
## ns(log_gdp, df = 5)2                       9.980e+00  5.689e+00   1.754
## ns(log_gdp, df = 5)3                      -9.173e+01  5.982e+01  -1.534
## ns(log_gdp, df = 5)4                       1.046e+04  3.083e+03   3.393
## ns(log_gdp, df = 5)5                       1.894e+04  5.563e+03   3.405
## ns(log_pop, df = 5)1                       2.455e-01  1.381e+00   0.178
## ns(log_pop, df = 5)2                      -2.991e+00  3.355e+00  -0.892
## ns(log_pop, df = 5)3                       5.307e+00  2.775e+01   0.191
## ns(log_pop, df = 5)4                       2.407e+02  3.094e+02   0.778
## ns(log_pop, df = 5)5                       4.248e+02  5.552e+02   0.765
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)1  1.891e+00  1.483e+00   1.275
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)1 -6.275e+00  5.958e+00  -1.053
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)1  8.190e+01  5.958e+01   1.375
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)1 -1.037e+04  3.080e+03  -3.367
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)1 -1.877e+04  5.559e+03  -3.377
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)2  3.003e+00  3.576e+00   0.840
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)2 -6.073e+00  6.411e+00  -0.947
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)2  9.189e+01  5.980e+01   1.536
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)2 -1.045e+04  3.083e+03  -3.389
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)2 -1.893e+04  5.562e+03  -3.403
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)3  1.356e+00  2.744e+01   0.049
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)3 -9.756e+00  2.796e+01  -0.349
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)3  5.439e+01  3.987e+01   1.364
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)3 -6.384e+03  1.906e+03  -3.350
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)3 -1.152e+04  3.438e+03  -3.350
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)4 -2.429e+02  3.069e+02  -0.791
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)4 -2.667e+02  3.098e+02  -0.861
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)4  3.526e+01  2.135e+02   0.165
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)4 -2.028e+04  5.807e+03  -3.492
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)4 -3.608e+04  1.044e+04  -3.457
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)5 -4.377e+02  5.509e+02  -0.794
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)5 -4.346e+02  5.551e+02  -0.783
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)5 -2.421e+02  3.280e+02  -0.738
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)5 -5.910e+03  1.874e+03  -3.154
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)5 -9.579e+03  2.821e+03  -3.396
##                                           Pr(>|t|)
## (Intercept)                               0.912927
## ns(log_gdp, df = 5)1                      0.348967
## ns(log_gdp, df = 5)2                      0.080000 .
## ns(log_gdp, df = 5)3                      0.125775
## ns(log_gdp, df = 5)4                      0.000746 ***
## ns(log_gdp, df = 5)5                      0.000715 ***
## ns(log_pop, df = 5)1                      0.859000
## ns(log_pop, df = 5)2                      0.373070
## ns(log_pop, df = 5)3                      0.848437
## ns(log_pop, df = 5)4                      0.436956
## ns(log_pop, df = 5)5                      0.444528
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)1 0.202955
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)1 0.292751
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)1 0.169848
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)1 0.000817 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)1 0.000789 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)2 0.401445
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)2 0.344011
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)2 0.125042
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)2 0.000756 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)2 0.000718 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)3 0.960605
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)3 0.727260
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)3 0.173046
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)3 0.000869 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)3 0.000869 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)4 0.429149
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)4 0.389727
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)4 0.868883
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)4 0.000522 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)4 0.000592 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)5 0.427288
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)5 0.434091
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)5 0.460755
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)5 0.001704 **
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)5 0.000738 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7991 on 505 degrees of freedom
## Multiple R-squared:  0.5926, Adjusted R-squared:  0.5644
## F-statistic: 20.99 on 35 and 505 DF,  p-value: < 2.2e-16
```

Choice of Model

- The interaction model improves on the log-transformed model, with a higher R-squared (**0.5926** vs **0.506**) and a lower residual standard error (**0.7991** vs **0.859**). This suggests that the interaction model explains more variability and provides a better fit. The significant interaction terms ($p < 0.001$) indicate non-linear interactions between GDP and population in predicting Olympic success.
- Given its better fit and ability to capture the complexity of the relationships between GDP, population, and gold medal counts, the **interaction model** is preferred for understanding how these factors jointly influence Olympic outcomes.

Part b: Units and Thresholds

Response Variable: Log of Gold Medals (`log_n`)

- **Unit**: The log of the number of gold medals (log-transformed count of medals).
- **Reasonable Threshold for Small Change**: A 10% proportional change, corresponding to 0.1 units in the log scale.

Predictor 1: Log of GDP (`log_gdp`)

- **Unit**: The log of GDP, where GDP is measured in billions of US dollars.
- **Reasonable Threshold for Small Change**: A 10% proportional change, corresponding to 0.1 units in the log scale.

Predictor 2: Log of Population (`log_pop`)

- **Unit**: The log of population, where population is measured in millions of people.
- **Reasonable Threshold for Small Change**: A 10% proportional change, corresponding to 0.1 units in the log scale.

Part c: Model Fitting

```r
# Fit the interaction model with log-transformed variables and splines
model_interaction <- lm(log_n ~ ns(log_gdp, df = 5) * ns(log_pop, df = 5), data = newdata_clean)
summary(model_interaction)
```

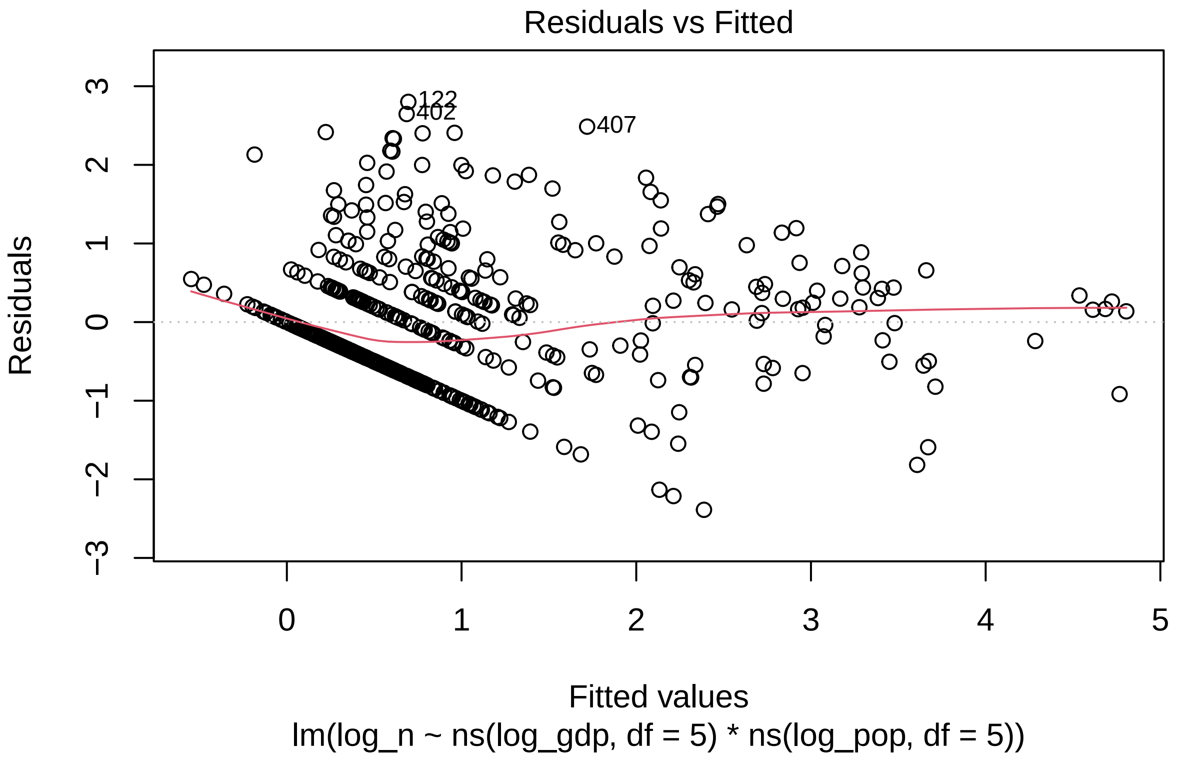
```
##
## Call:
## lm(formula = log_n ~ ns(log_gdp, df = 5) * ns(log_pop, df = 5),
##     data = newdata_clean)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3876 -0.4965 -0.1575  0.3591  2.8011
##
## Coefficients:
```

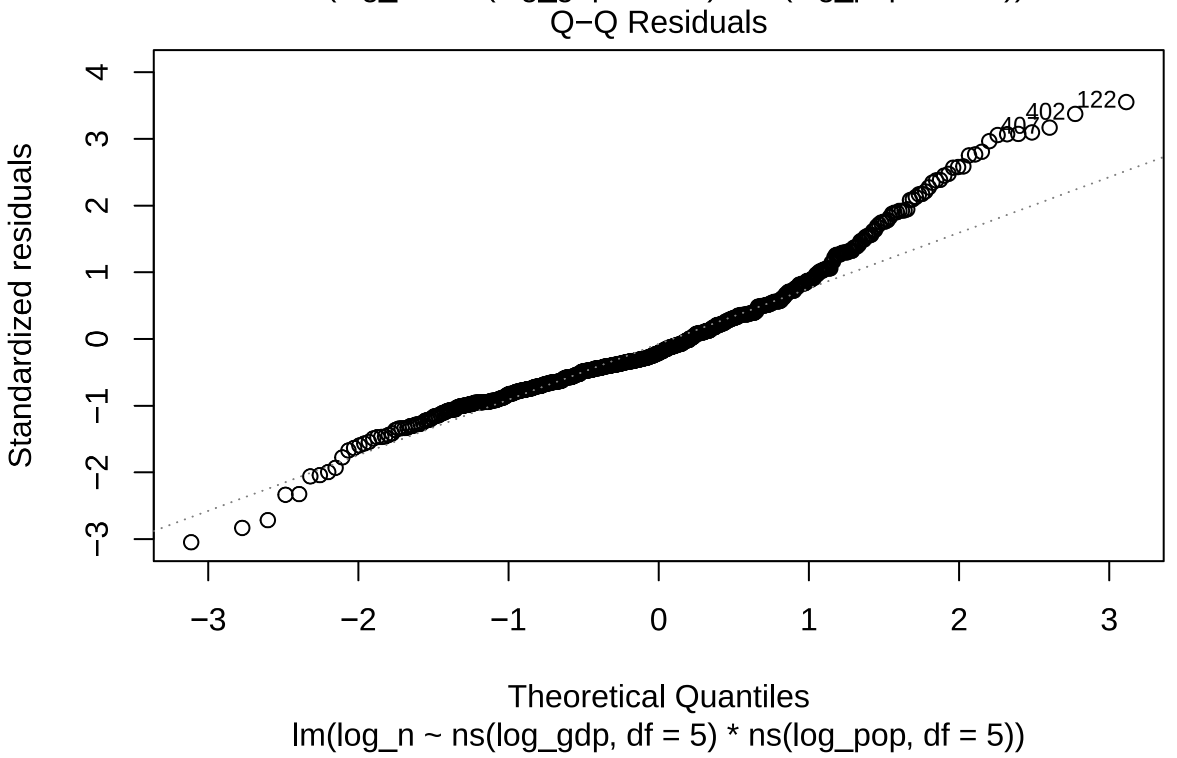
```
##                                           Estimate Std. Error t value
## (Intercept)                              4.092e-02  3.740e-01   0.109
## ns(log_gdp, df = 5)1                    -1.176e+00  1.254e+00  -0.937
## ns(log_gdp, df = 5)2                     9.980e+00  5.689e+00   1.754
## ns(log_gdp, df = 5)3                    -9.173e+01  5.982e+01  -1.534
## ns(log_gdp, df = 5)4                     1.046e+04  3.083e+03   3.393
## ns(log_gdp, df = 5)5                     1.894e+04  5.563e+03   3.405
## ns(log_pop, df = 5)1                     2.455e-01  1.381e+00   0.178
## ns(log_pop, df = 5)2                    -2.991e+00  3.355e+00  -0.892
## ns(log_pop, df = 5)3                     5.307e+00  2.775e+01   0.191
## ns(log_pop, df = 5)4                     2.407e+02  3.094e+02   0.778
## ns(log_pop, df = 5)5                     4.248e+02  5.552e+02   0.765
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)1  1.891e+00  1.483e+00   1.275
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)1 -6.275e+00  5.958e+00  -1.053
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)1  8.190e+01  5.958e+01   1.375
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)1 -1.037e+04  3.080e+03  -3.367
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)1 -1.877e+04  5.559e+03  -3.377
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)2  3.003e+00  3.576e+00   0.840
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)2 -6.073e+00  6.411e+00  -0.947
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)2  9.189e+01  5.980e+01   1.536
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)2 -1.045e+04  3.083e+03  -3.389
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)2 -1.893e+04  5.562e+03  -3.403
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)3  1.356e+00  2.744e+01   0.049
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)3 -9.756e+00  2.796e+01  -0.349
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)3  5.439e+01  3.987e+01   1.364
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)3 -6.384e+03  1.906e+03  -3.350
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)3 -1.152e+04  3.438e+03  -3.350
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)4 -2.429e+02  3.069e+02  -0.791
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)4 -2.667e+02  3.098e+02  -0.861
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)4  3.526e+01  2.135e+02   0.165
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)4 -2.028e+04  5.807e+03  -3.492
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)4 -3.608e+04  1.044e+04  -3.457
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)5 -4.377e+02  5.509e+02  -0.794
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)5 -4.346e+02  5.551e+02  -0.783
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)5 -2.421e+02  3.280e+02  -0.738
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)5 -5.910e+03  1.874e+03  -3.154
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)5 -9.579e+03  2.821e+03  -3.396
##                                           Pr(>|t|)
## (Intercept)                               0.912927
## ns(log_gdp, df = 5)1                      0.348967
## ns(log_gdp, df = 5)2                      0.080000 .
## ns(log_gdp, df = 5)3                      0.125775
## ns(log_gdp, df = 5)4                      0.000746 ***
## ns(log_gdp, df = 5)5                      0.000715 ***
## ns(log_pop, df = 5)1                      0.859000
## ns(log_pop, df = 5)2                      0.373070
## ns(log_pop, df = 5)3                      0.848437
## ns(log_pop, df = 5)4                      0.436956
## ns(log_pop, df = 5)5                      0.444528
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)1 0.202955
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)1 0.292751
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)1 0.169848
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)1 0.000817 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)1 0.000789 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)2 0.401445
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)2 0.344011
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)2 0.125042
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)2 0.000756 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)2 0.000718 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)3 0.960605
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)3 0.727260
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)3 0.173046
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)3 0.000869 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)3 0.000869 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)4 0.429149
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)4 0.389727
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)4 0.868883
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)4 0.000522 ***
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)4 0.000592 ***
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)5 0.427288
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)5 0.434091
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)5 0.460755
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)5 0.001704 **
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)5 0.000738 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7991 on 505 degrees of freedom
## Multiple R-squared:  0.5926, Adjusted R-squared:  0.5644
## F-statistic: 20.99 on 35 and 505 DF,  p-value: < 2.2e-16
```

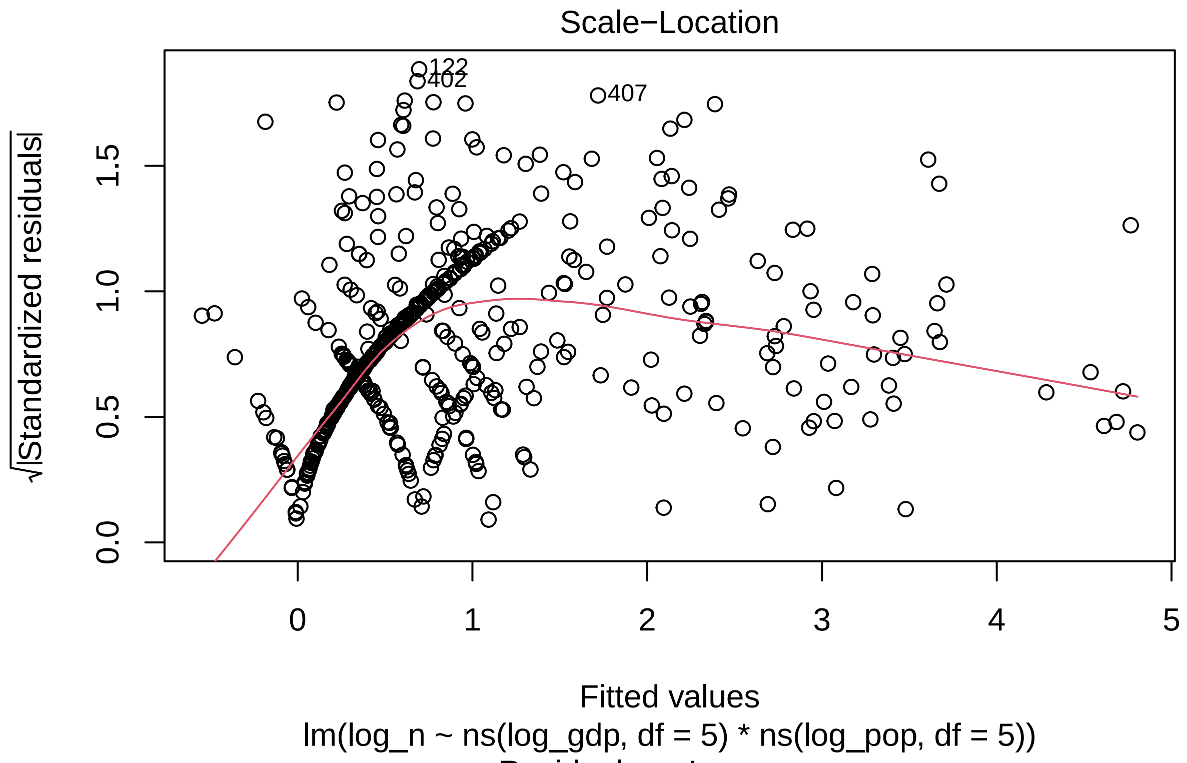
- **R-squared**: 0.5926, meaning that the model explains 59.26% of the variability in the log-transformed number of gold medals.
- **Residual Standard Error (RSE)**: 0.7991, indicating a reasonably good fit, with relatively small deviations from the observed values.
- **Significant Interaction Terms**: Several interaction terms between log_gdp and log_pop are statistically significant ($p < 0.001$), highlighting important non-linear relationships between GDP and population in predicting Olympic success.

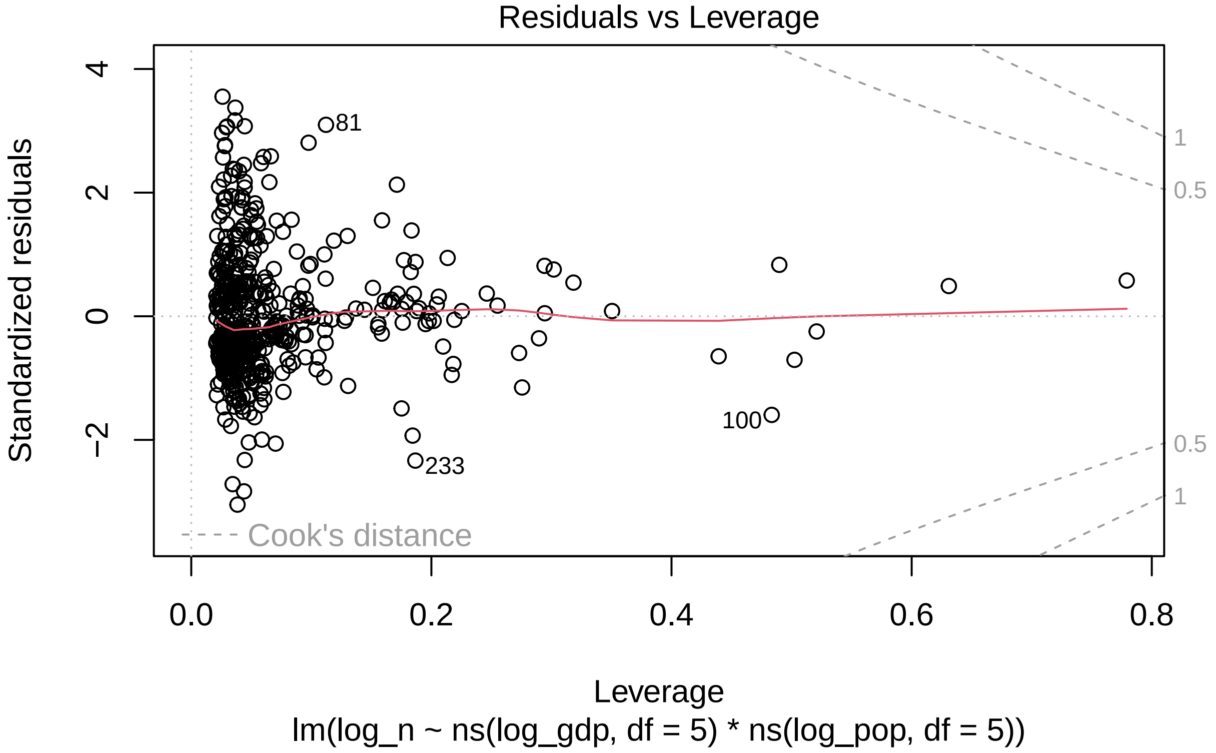
Part d: Model Diagnostics

```r
# Base R diagnostic plots
par(mar = c(5, 4, 4, 2) + 0.1)
plot(model_interaction)
```


Residuals vs Fitted
Residuals
Fitted values
lm(log_n ~ ns(log_gdp, df = 5) * ns(log_pop, df = 5))
122
402
407
3
2
1
0
−1
−2
−3
0
1
2
3
4
5


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(log_n ~ ns(log_gdp, df = 5) * ns(log_pop, df = 5))
122
402
407
4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3





- **Linearity**: The **Residuals vs Fitted plot** shows a slight curvature, suggesting that the model might not fully capture the non-linear relationship between the predictors and the response. This non-linearity indicates that some interactions or non-linear terms might still be missing from the model, despite using splines.
- **Normality of Residuals**: The **Q-Q plot** suggests that the residuals are roughly normally distributed. However, the deviations at both the lower and upper tails indicate that there are some extreme values,

which may not be well-captured by the model, potentially outliers.

- **Homoscedasticity**: The **Scale-Location plot** shows some evidence of heteroscedasticity. The spread of the residuals appears to increase with the fitted values, which suggests that the variance of the residuals is not constant across all levels of the predicted values. This indicates that the assumption of homoscedasticity might be violated.
- **Outliers/Influential Points**: The **Residuals vs Leverage plot** highlights a few points with high leverage, particularly observations 81, 100, and 233. These points might be exerting a significant influence on the model, and it may be worthwhile to investigate them further. Additionally, these points do not seem to fall under the Cook's distance threshold, meaning that although they have leverage, they may not have an outsized effect on the model's overall fit.

Part e: Model Adjustments

Robust Standard Errors:

- Since the **Scale-Location plot** suggest **heteroscedasticity**, one option is to use **robust standard errors** to adjust for non-constant variance in the residuals. This can provide more reliable inference even if the residuals do not have constant variance.

```
# Fit the interaction model
model_interaction <- lm(log_n ~ ns(log_gdp, df = 5) * ns(log_pop, df = 5), data = newdata_clean)

# Use coeftest() with vcovHC for robust standard errors
robust_se_model <- coeftest(model_interaction, vcov = vcovHC(model_interaction, type = "HC3"))

# Display the results with robust standard errors
print(robust_se_model)
```

```
##
## t test of coefficients:
##
##                                               Estimate  Std. Error t value
## (Intercept)                                 4.0921e-02  1.7797e-01  0.2299
## ns(log_gdp, df = 5)1                       -1.1756e+00  7.6702e-01 -1.5326
## ns(log_gdp, df = 5)2                        9.9804e+00  5.1845e+00  1.9250
## ns(log_gdp, df = 5)3                       -9.1732e+01  5.7521e+01 -1.5948
## ns(log_gdp, df = 5)4                        1.0460e+04  3.5192e+03  2.9723
## ns(log_gdp, df = 5)5                        1.8939e+04  6.3374e+03  2.9885
## ns(log_pop, df = 5)1                        2.4552e-01  7.8596e-01  0.3124
## ns(log_pop, df = 5)2                       -2.9909e+00  2.1831e+00 -1.3700
## ns(log_pop, df = 5)3                        5.3068e+00  1.2197e+01  0.4351
## ns(log_pop, df = 5)4                        2.4070e+02  3.0338e+02  0.7934
## ns(log_pop, df = 5)5                        4.2479e+02  5.4055e+02  0.7859
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)1  1.8911e+00  8.4810e-01  2.2298
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)1 -6.2753e+00  5.2938e+00 -1.1854
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)1  8.1900e+01  5.7261e+01  1.4303
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)1 -1.0373e+04  3.5174e+03 -2.9490
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)1 -1.8773e+04  6.3345e+03 -2.9636
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)2  3.0030e+00  1.9962e+00  1.5044
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)2 -6.0726e+00  5.4920e+00 -1.1057
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)2  9.1885e+01  5.7423e+01  1.6001
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)2 -1.0449e+04  3.5193e+03 -2.9689
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)2 -1.8932e+04  6.3373e+03 -2.9873
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)3  1.3562e+00  1.2387e+01  0.1095
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)3 -9.7563e+00  1.2476e+01 -0.7820
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)3  5.4395e+01  3.5258e+01  1.5427
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)3 -6.3836e+03  2.1823e+03 -2.9251
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)3 -1.1517e+04  3.9303e+03 -2.9304
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)4 -2.4286e+02  3.0164e+02 -0.8051
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)4 -2.6669e+02  3.0384e+02 -0.8777
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)4  3.5257e+01  2.0974e+02  0.1681
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)4 -2.0276e+04  6.6121e+03 -3.0665
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)4 -3.6085e+04  1.1862e+04 -3.0419
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)5 -4.3765e+02  5.3789e+02 -0.8137
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)5 -4.3458e+02  5.4033e+02 -0.8043
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)5 -2.4211e+02  3.2204e+02 -0.7518
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)5 -5.9105e+03  2.0608e+03 -2.8681
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)5 -9.5787e+03  3.2311e+03 -2.9646
##                                            Pr(>|t|)
## (Intercept)                                0.818242
## ns(log_gdp, df = 5)1                       0.125989
## ns(log_gdp, df = 5)2                       0.054785 .
## ns(log_gdp, df = 5)3                       0.111389
## ns(log_gdp, df = 5)4                       0.003096 **
## ns(log_gdp, df = 5)5                       0.002940 **
## ns(log_pop, df = 5)1                       0.754877
## ns(log_pop, df = 5)2                       0.171292
## ns(log_pop, df = 5)3                       0.663685
## ns(log_pop, df = 5)4                       0.427918
## ns(log_pop, df = 5)5                       0.432318
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)1 0.026197 *
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)1 0.236412
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)1 0.153250
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)1 0.003336 **
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)1 0.003184 **
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)2 0.133114
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)2 0.269379
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)2 0.110194
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)2 0.003130 **
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)2 0.002951 **
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)3 0.912861
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)3 0.434582
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)3 0.123519
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)3 0.003598 **
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)3 0.003539 **
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)4 0.421121
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)4 0.380513
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)4 0.866576
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)4 0.002282 **
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)4 0.002473 **
## ns(log_gdp, df = 5)1:ns(log_pop, df = 5)5 0.416228
## ns(log_gdp, df = 5)2:ns(log_pop, df = 5)5 0.421605
## ns(log_gdp, df = 5)3:ns(log_pop, df = 5)5 0.452528
## ns(log_gdp, df = 5)4:ns(log_pop, df = 5)5 0.004302 **
## ns(log_gdp, df = 5)5:ns(log_pop, df = 5)5 0.003174 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Robust standard errors** adjust the estimates to account for any unevenness in the residuals (heteroscedasticity). This makes the results more reliable, especially in models with potential data issues like this one. Some previously marginally significant interactions (e.g., the combined effects of GDP and population) are now more robustly significant, confirming that these interactions are key in predicting gold medals.

Generalized Linear Models

- Given that the response variable is the **number of gold medals** (a count variable), the **Poisson regression** or **Negative Binomial regression** might be appropriate.

Poisson regression

- In Poisson regression, the log of the expected number of events is modeled as a linear combination of the predictor variables.
- The canonical link function for Poisson regression is the **log link**.

```r
# Fit a Poisson GLM
model_glm_poisson <- glm(n ~ log_gdp + log_pop, family = poisson(link = "log"), data = newdata_clean)

# Summary of the model
summary(model_glm_poisson)
```

```
##
## Call:
## glm(formula = n ~ log_gdp + log_pop, family = poisson(link = "log"),
##     data = newdata_clean)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.78128    0.07625 -36.476  < 2e-16 ***
## log_gdp      0.81007    0.01617  50.093  < 2e-16 ***
## log_pop     -0.06547    0.01808  -3.622 0.000293 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 10453.6  on 540  degrees of freedom
## Residual deviance:  3877.5  on 538  degrees of freedom
## AIC: 4799.3
##
## Number of Fisher Scoring iterations: 6
```

- **Intercept**: The estimated intercept of -2.78 means that when GDP and population are at their log-transformed value of zero (which theoretically would be when GDP and population are both 1), the expected number of gold medals is quite low.
- **log_gdp**: A 1-unit increase in log-transformed GDP is associated with an 81% increase in the expected number of gold medals ($e^{0.81007} \approx 2.25$, or a multiplicative effect of 2.25).
- **log_pop**: A 1-unit increase in log-transformed population is associated with a 6.5% decrease in the expected number of gold medals ($e^{-0.06547} \approx 0.937$).
- **Model Performance**:
  - **Residual Deviance**: 3877.5 on 538 degrees of freedom, which is quite large, suggesting that the model might not be fitting the data well.

  - **AIC**: 4799.3 (Poisson models tend to have higher AIC values when overdispersion is present).

Negative Binomial Regression (GLM)

- **Negative Binomial Regression**, relaxes the strict assumption of equal mean and variance.

```
# Fit a Negative Binomial GLM
model_glm_nb <- glm.nb(n ~ log_gdp + log_pop, data = newdata_clean)

# Summary of the model
summary(model_glm_nb)
```

```
##
## Call:
## glm.nb(formula = n ~ log_gdp + log_pop, data = newdata_clean,
##     init.theta = 0.3550280463, link = log)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.97690    0.22723  -8.700   <2e-16 ***
## log_gdp      0.64947    0.06273  10.354   <2e-16 ***
## log_pop     -0.01899    0.07851  -0.242    0.809
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.355) family taken to be 1)
##
##     Null deviance: 795.68  on 540  degrees of freedom
## Residual deviance: 493.17  on 538  degrees of freedom
## AIC: 2230.8
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.3550
##           Std. Err.:  0.0313
##
##  2 x log-likelihood:  -2222.7600
```

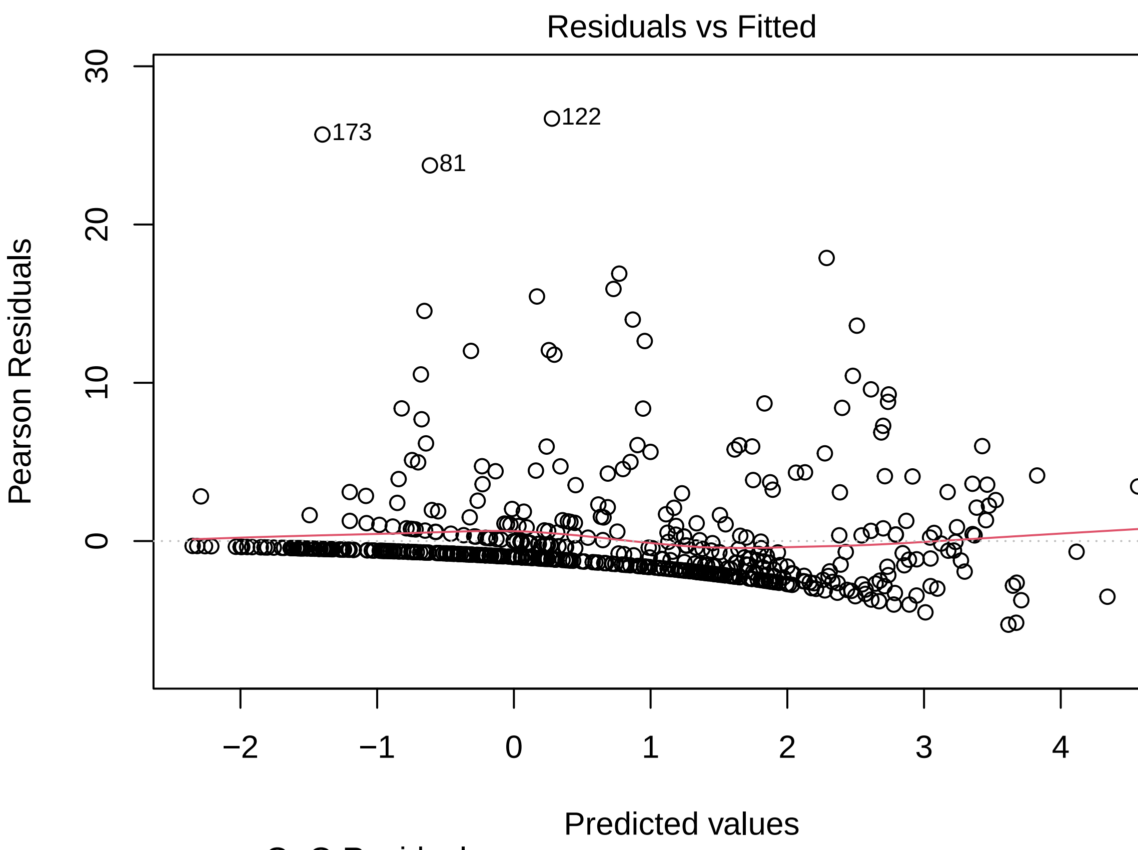
- **Intercept**: The intercept of -1.98 is slightly higher than in the Poisson model, indicating a higher baseline expected count of gold medals compared to the Poisson model.
- **log_gdp**: A 1-unit increase in log-transformed GDP is associated with a 65% increase in the expected number of gold medals ($e^{0.64947} \approx 1.91$).
- **log_pop**: The coefficient is not statistically significant ($p = 0.809$), meaning that population doesn't have a clear impact on the expected number of gold medals in this model.
- **Model Performance**:
  - **Residual Deviance**: 493.17 on 538 degrees of freedom, which is much lower than the Poisson model, suggesting a better fit.
  - **AIC**: 2230.8 (significantly lower than the Poisson model's AIC, indicating a better model).

```

# Compare the AIC of different models
AIC(model_interaction, model_glm_poisson, model_glm_nb)
```

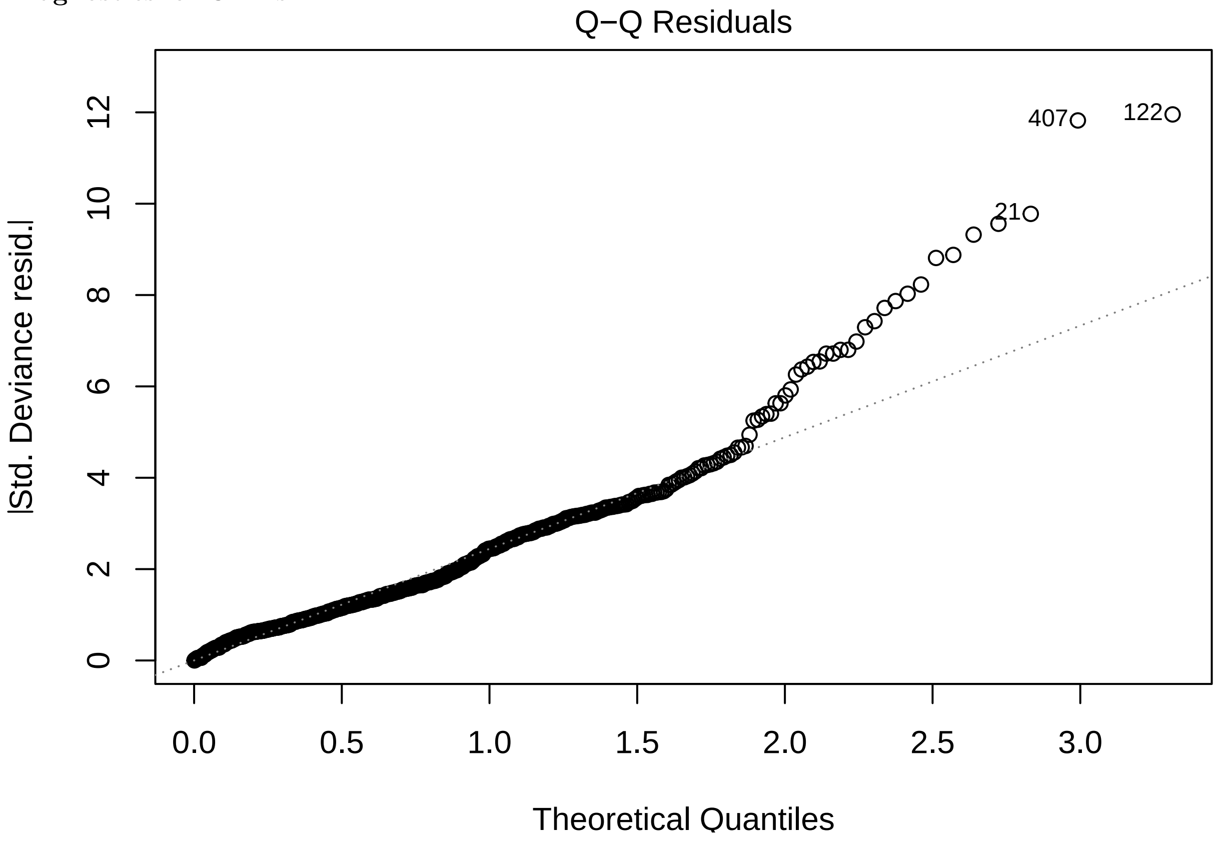
Comparing Models

```
##                   df      AIC
## model_interaction 37 1329.386
## model_glm_poisson  3 4799.314
## model_glm_nb       4 2230.760
```

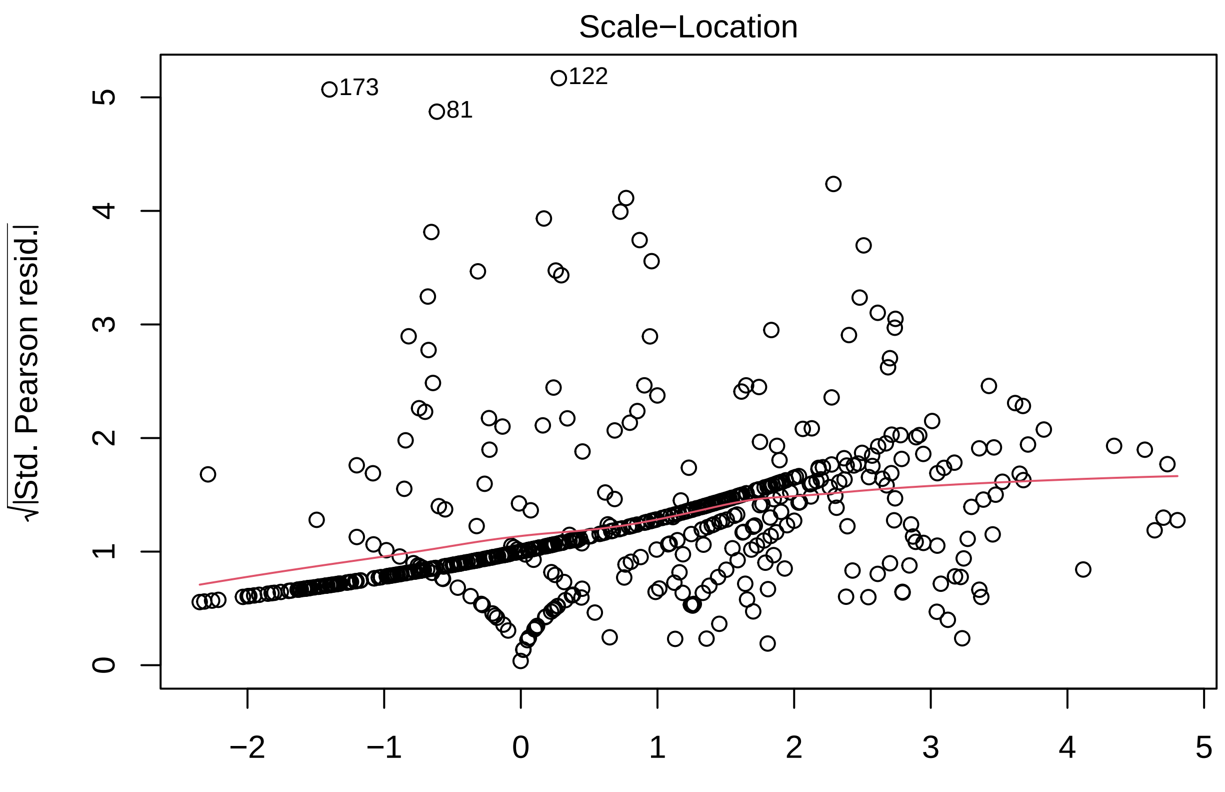
```r
# Set the margins smaller to avoid the error
par(mar = c(4, 4, 2, 1)) # Adjust as needed

# Now, create the diagnostic plot
plot(model_glm_poisson)
```

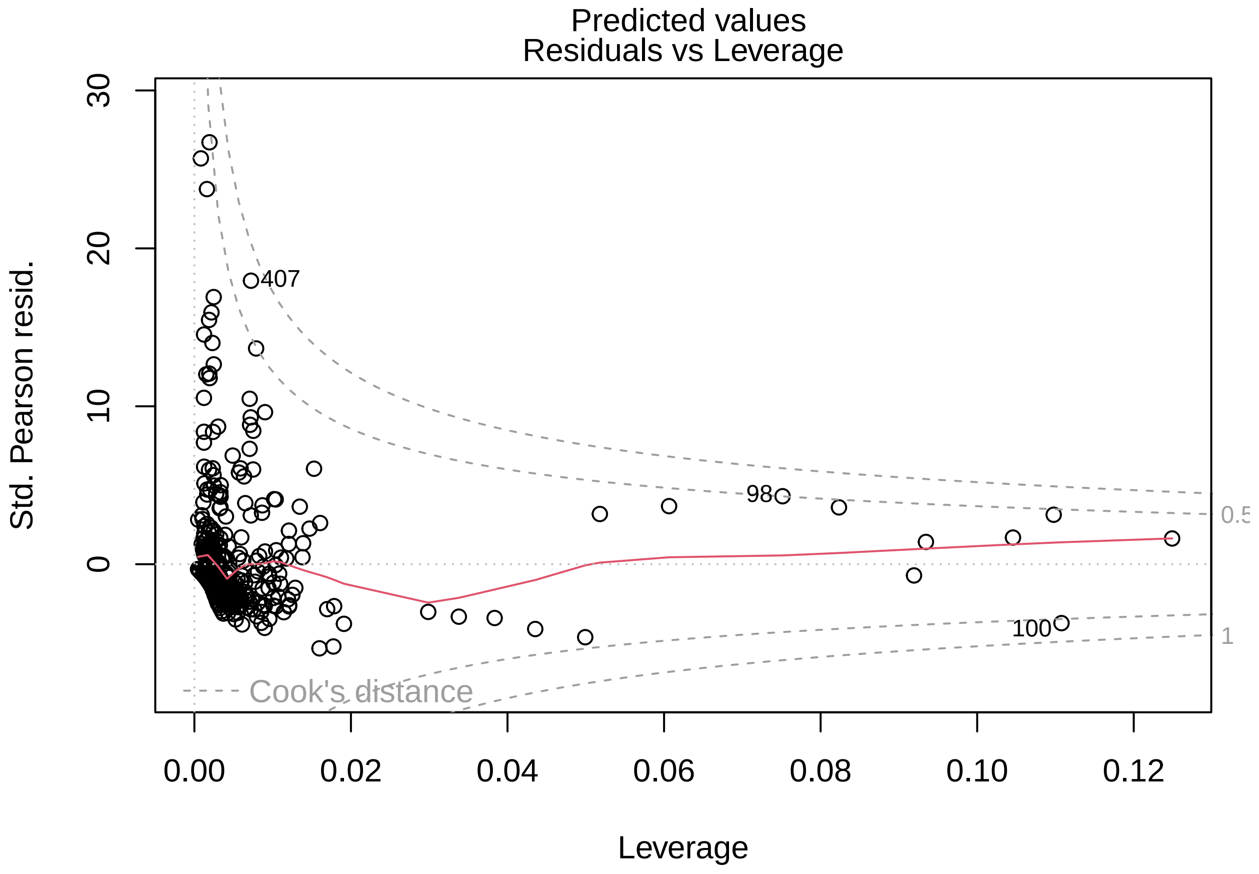


Residuals vs Fitted
Pearson Residuals
Predicted values
173
81
122
30
20
10
0
−2
−1
0
1
2
3
4



Q−Q Residuals
|Std. Deviance resid.|
Theoretical Quantiles
407
122
21
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
2
4
6
8
10
12



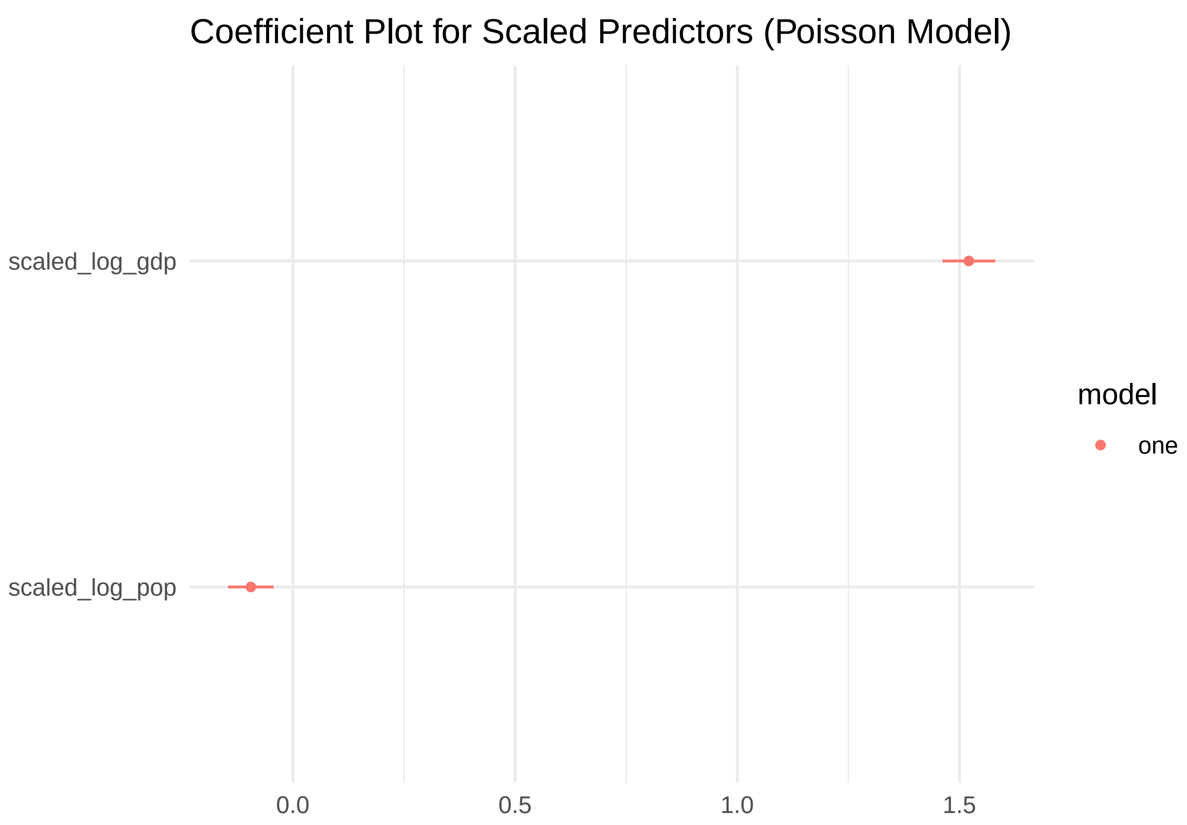


- Residuals vs Fitted (Heteroscedasticity): The plot suggests some heteroscedasticity, as the residuals tend to fan out as fitted values increase. This is not ideal, as it indicates that the variance of the residuals changes with the predicted values. This behavior could be a sign that the Poisson model does not perfectly capture the variance structure of the data.

- Q-Q Plot (Normality of Residuals): The Q-Q plot shows a significant deviation from normality, particularly in the tails. The residuals are far from the line, indicating that the assumption of normally distributed residuals is violated. This is a common occurrence with count data in Poisson models, which may not always have normally distributed residuals.
- Scale-Location Plot (Homoscedasticity): The scale-location plot also shows heteroscedasticity, as the residuals appear to follow a non-constant spread. This further indicates that the model's assumptions about variance are not well met.
- Residuals vs Leverage (Influential Points): There are several points with high leverage and significant influence, suggesting that a few data points may disproportionately impact the model's fit. Points like 407 and 122 may need closer examination, as they may unduly influence the model's parameters.
- Conclusion: it is clear that the Poisson model may not be the best fit. The heteroscedasticity, non-normal residuals, and high-leverage points indicate some issues.

Part f: Coefficient Plot

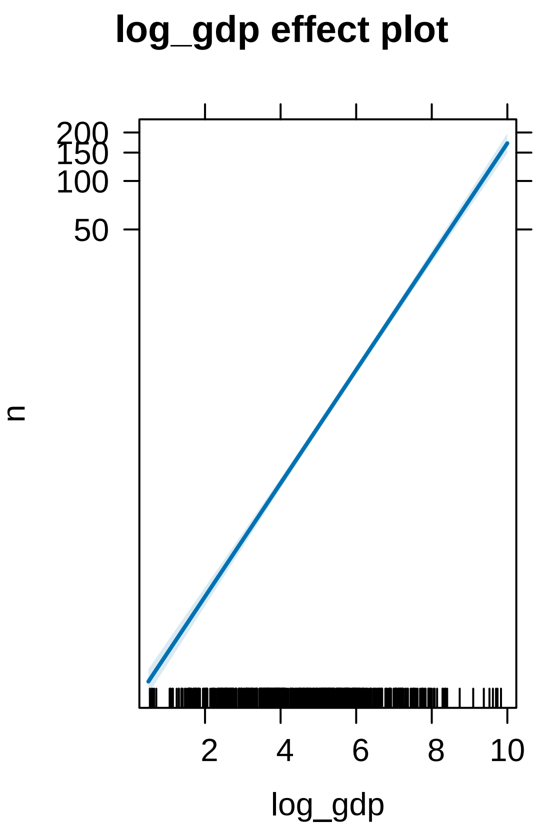
```
# Scale and center the predictors
newdata_clean <- newdata_clean %>%
  mutate(scaled_log_gdp = scale(log_gdp, center = TRUE, scale = TRUE),
         scaled_log_pop = scale(log_pop, center = TRUE, scale = TRUE))

# Fit the Poisson model with scaled and centered predictors
model_glm_scaled <- glm(n ~ scaled_log_gdp + scaled_log_pop, family = poisson(link = "log"), data = newd

# Tidy the model coefficients
tidy_model <- tidy(model_glm_scaled)

# Plot the coefficients
dwplot(tidy_model) +
  ggtitle("Coefficient Plot for Scaled Predictors (Poisson Model)") +
  theme_minimal()
```

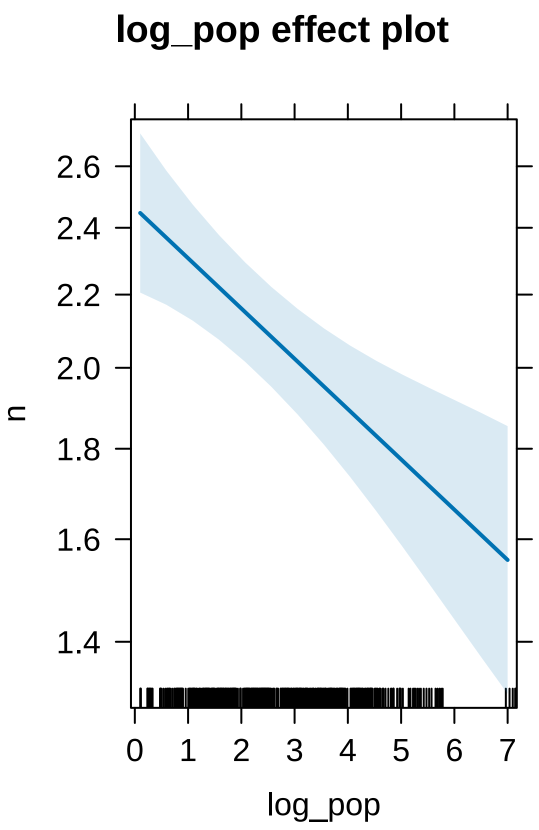


- **Scaled_log_gdp**: The coefficient for scaled log_gdp is positive and significant, indicating that countries with higher GDPs (in the log scale) are more likely to win more medals.
- **Scaled_log_pop**: The coefficient for scaled log_pop is near zero, indicating a weaker and less significant relationship between population size and the number of medals.
- This coefficient plot confirms that **GDP** plays a much stronger role in predicting Olympic success compared to population size.

Part g: Effects Plot

```r
# Calculate the effects for the poisson model
effects_poisson <- allEffects(model_glm_poisson)

# Plot the effects
plot(effects_poisson)
```





- **Log GDP Effect**: The plot on the left shows a strong positive linear relationship between `log_gdp` and the predicted number of medals (`n`). As `log_gdp` increases, the number of predicted medals increases sharply. This suggests that GDP is a strong predictor of the number of medals a country wins. The confidence band is narrow, indicating high certainty around the prediction for the effect of GDP.
- **Log Population Effect**: The plot on the right shows a weak negative relationship between `log_pop` and the predicted number of medals. As `log_pop` increases, the number of predicted medals decreases slightly. This might seem counterintuitive but reflects the negative (though weak) effect observed in the model coefficients. The confidence interval is wider here, indicating less certainty in the effect of population on medal counts, particularly as population increases.
- **Summary**: **GDP** is a much stronger predictor of Olympic success (in terms of medal counts) than population, which aligns with the model coefficients. The large effect of GDP suggests that wealthier countries are more likely to win more medals. **Population** does not appear to have a substantial positive effect on medal counts, and there may even be a slight negative relationship when accounting for GDP.

Question 2

Introduction

In this question, we assess the impact of three increasing treatment levels (I, II, III) compared to a control group (C). Our analysis focuses on two main aspects:

1. The overall effect of the treatments by comparing the control group with the average response of the treatment groups.
2. The incremental effects between the treatment levels, specifically between I and II, and II and III.

Data

```r
# Create a data frame with one observation per treatment level
df <- data.frame(
  Group = factor(c("C", "I", "II", "III"), levels = c("C", "I", "II", "III")),
  Response = c(10, 12, 15, 22)  # Hypothetical response values
)
```

Methodology

In this analysis, we use two types of contrasts:

1. **Control vs. Average of Treatments**: A *custom contrast* is used to compare the control group (C) to the average of the treatment groups (I, II, III).
2. **Successive Differences Among Treatments**: *Successive-difference contrasts* are applied to compare Treatment II with Treatment I, and Treatment III with Treatment II.

```r
C <- matrix(c(
  -1,  1/3,  1/3,  1/3,  # Control vs Average of I, II, III (custom contrast)
   0,  1,   -1,    0,    # I vs II (Successive-difference contrast)
   0,  0,    1,   -1     # II vs III (Successive-difference contrast)
), nrow = 4, byrow = TRUE)

contrasts(df$Group) <- C
```

Analysis

```r
# Fit the linear model
model <- lm(Response ~ Group, data = df)
summary(model)
```

```
##
## Call:
## lm(formula = Response ~ Group, data = df)
##
## Residuals:
## ALL 4 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   19.364        NaN     NaN      NaN
## Group1         4.364        NaN     NaN      NaN
## Group2        -6.182        NaN     NaN      NaN
## Group3        -8.818        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 3 and 0 DF,  p-value: NA
```

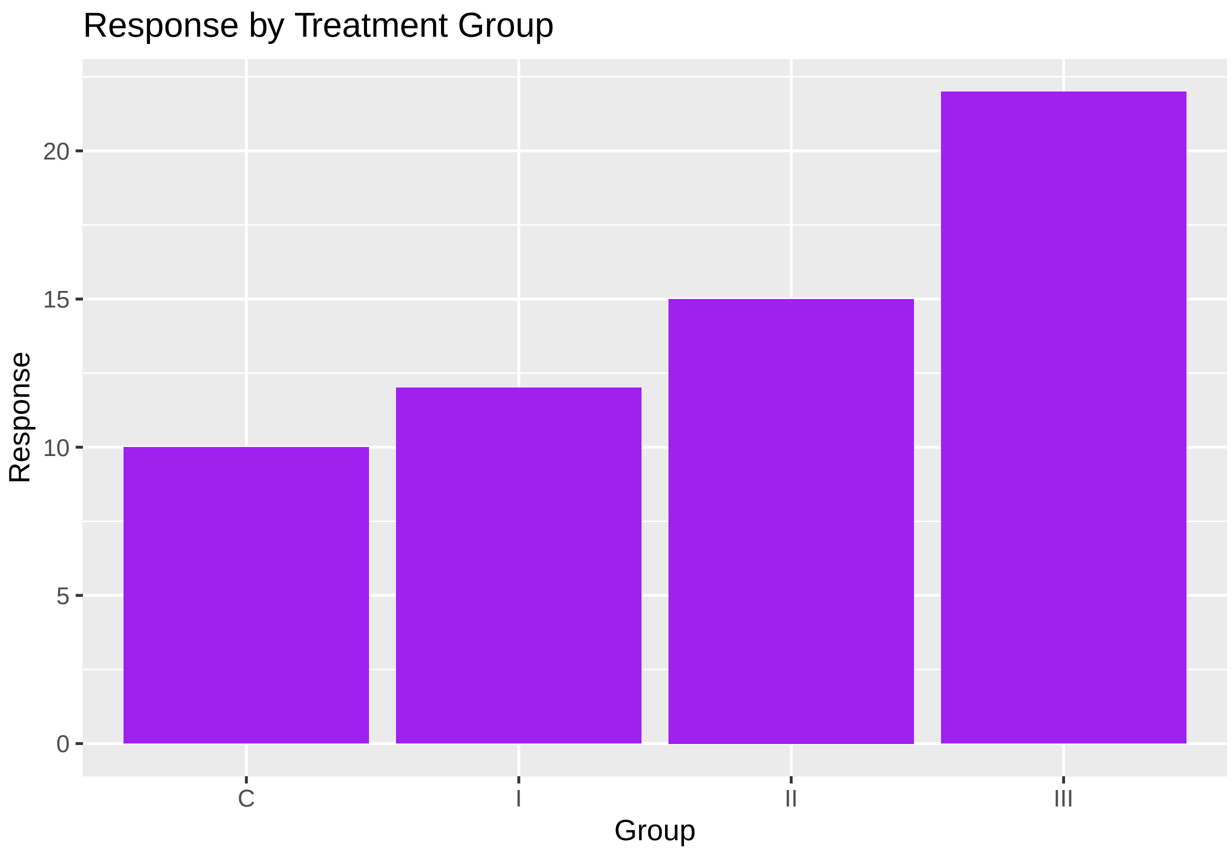
- Explanation:

1. **NaN Values**: There are `NaN` values in the model summary, for the standard errors, t-values, and p-values, due to the lack of residual degrees of freedom.
2. **Perfect Fit**: There are 4 observations (one for each level: C, I, II, III), and the linear model is trying to estimate 4 parameters (the intercept and three group contrasts), leaving no room for residual variation. This causes the standard errors of the coefficients to be `NaN`, as there is no uncertainty left to estimate.

3. **Focus on Coefficients**: The goal of the analysis is to interpret the **contrast estimates** (the model's coefficients), rather than relying on p-values or standard errors, which cannot be calculated here.

Visualization

```r
# Visualize the response for each group using a bar plot
ggplot(df, aes(x = Group, y = Response)) +
  geom_bar(stat = "identity", fill = "purple") +
  labs(title = "Response by Treatment Group", x = "Group", y = "Response")
```



Question 3

Introduction

- We are tasked with evaluating how violations of the assumption of conditional normality affect the `bias`, root mean squared error (`RMSE`), `power`, and `coverage` of linear regression models.
- Specifically, we will generate data where the errors are not normally distributed by sampling from a t-distribution with varying degrees of freedom (`df`).
- We will assess the effect of different sample sizes (`n = 10, 20, 100`) and degrees of freedom (`df = seq(2, 50, by = 6)`) on model performance.
- We will also evaluate the power of the **Shapiro-Wilk** test for detecting non-normality.

Simulating Data with t-distribution

```r
# simulate data for a linear model with **t-distributed** errors
sim_fun_t <- function(n = 100,
                      slope = 1,
                      sd = 1,
                      intercept = 0,
                      df = 10) {
  x <- runif(n)
  errors <- sd * rt(n, df = df)
  y <- intercept + slope * x + errors
  data.frame(x, y)
}
```

- This function generates `n` data points with a linear relationship between `x` and `y`, but with errors sampled from a **t-distribution** rather than a normal distribution. The degrees of freedom `df` control the extent of deviation from normality.

Evaluating Bias, RMSE, Power, and Coverage

- We define a function `run_simulation()` to run a specified number of simulations (`n_sim`) and calculate the `bias`, `RMSE`, `power`, and `coverage` of the linear regression mode

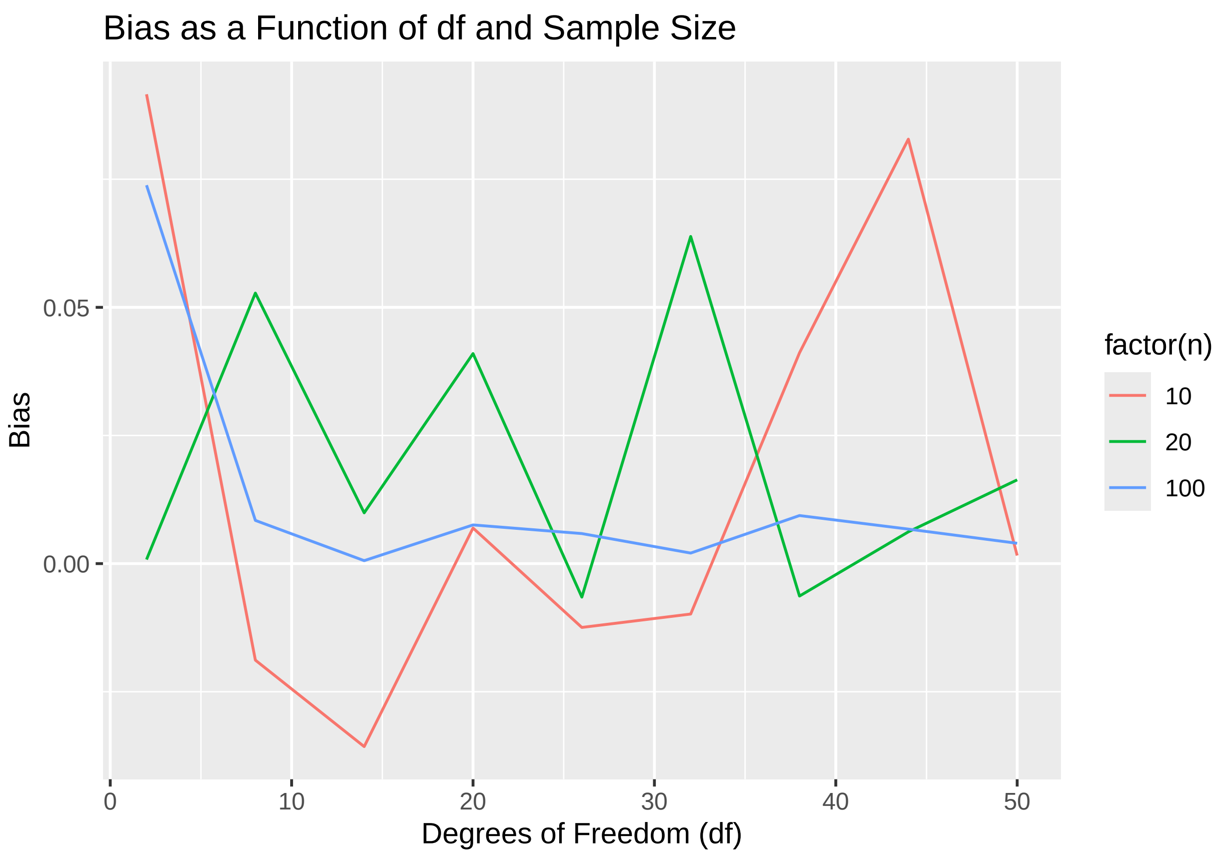
```r
run_simulation <- function(n = 100,
                           true_slope = 1,
                           sd = 1,
                           intercept = 0,
                           df = 10,
                           alpha = 0.05,
                           n_sim = 1000) {
  slopes <- numeric(n_sim)
  p_values <- numeric(n_sim)
  coverage <- numeric(n_sim)

  for (i in 1:n_sim) {
    # Simulate data with t-distributed errors
    data <- sim_fun_t(n, slope = true_slope, sd = sd, intercept = intercept, df = df)

    # Fit a linear regression model
    m <- lm(y ~ x, data = data)

    # Extract the estimated slope, p-value, and confidence interval for the slope
    slopes[i] <- coef(m)[2]
    p_values[i] <- coef(summary(m))[2, "Pr(>|t|)"]
    conf_int <- confint(m)[2, ]

    # Check whether the confidence interval contains the true slope (for coverage calculation)
    coverage[i] <- (conf_int[1] < true_slope & true_slope < conf_int[2])
  }

  # # Compute the bias (average difference between estimated and true slope)
  bias <- mean(slopes - true_slope)

  # Compute the root mean squared error (RMSE) of the slope estimates
  rmse <- sqrt(mean((slopes - true_slope)^2))

  # Compute the power (proportion of times p-value is less than alpha)
  power <- mean(p_values < alpha)

  # Compute the coverage (proportion of times the confidence interval contains the true slope)
  coverage_prob <- mean(coverage)

  # Return a data frame with the results for the current simulation
  data.frame(df = df, n = n, bias = bias, rmse = rmse, power = power, coverage = coverage_prob)
}
```

Simulations Running

```
df_values <- seq(2, 50, by = 6)
n_values <- c(10, 20, 100)

# Run the simulation for all combinations of df and n
results <- do.call(rbind, lapply(n_values, function(n) {
  do.call(rbind, lapply(df_values, function(df) {
    run_simulation(n = n, df = df)
  }))
}))
```

Visualization

```
# Bias as a Function of Degrees of Freedom and Sample Size
ggplot(results, aes(x = df, y = bias, color = factor(n))) +
  geom_line() +
  labs(title = "Bias as a Function of df and Sample Size", x = "Degrees of Freedom (df)", y = "Bias")
```



Reference

- OpenAI, ChatGPT (2024). Assistance with R programming and output analysis. Accessed on September 14, 2024.
- Dr. Ben Bolker, Lecture Notes for "Statistical Modeling", McMaster University. Accessed on September 10, 2024.