STATS 720 Assignment 4

Konstantinos Banos 400609120

2024-12-05

Table of contents

Introduction

we will apply mixed-effects modeling techniques to analyze and simulate data from various contexts. First, we will explore the Olympics dataset, using a chosen response variable such as total medals or gold medals, and incorporate country as a random-effect grouping variable. This will involve constructing a maximal model with fixed-effect predictors and random effects, considering time as both a fixed and random effect. We will also investigate the appropriateness of including a country-year interaction as a random effect, depending on the number of observations and the distribution of the response variable. Next, we will analyze the toenail dataset from the HSAUR3 package, applying various modeling approaches to predict a binary outcome and comparing methods such as pooled GLM, penalized quasi-likelihood estimation, Laplace approximation, adaptive Gauss-Hermite quadrature, and Bayesian models. In addition, we will simulate binomial data with repeated measurements to evaluate the performance of different fitting techniques, assessing metrics like bias, variance, RMSE, and coverage. Finally, we will fit a four-parameter logistic model to a dataset of treatment groups and time points, allowing specific model parameters to vary by group and individual. The analysis will involve constructing fixed- and random-effect design matrices and fitting the model using the RTMB package. The overall objective of the assignment is to compare the performance of various mixed-effects modeling techniques, evaluate their accuracy, and present the results graphically for better interpretation.

1. Analysis of the Olympics Dataset

```
# Improt and convert the data to tibble
olymp <- read.csv(url("https://raw.githubusercontent.com/bbolker/stats720/main/data/olymp1.cs
olymp <- as_tibble(olymp)

gdp_per_cap <- olymp$gdp/olymp$pop ## GDP per Capita
log_gdp <- log(olymp$gdp)          ## Logarithm of GDP
log_pop <- log(olymp$pop)          ## Logarithm of Population
log_gdp_perC <- log_gdp-log_pop    ## Logarithm of GDP per Capita
log_n <- log(olymp$n+1)            ## Logarithm of the Number of Medals
log_year <- log(olymp$year)


## Add six new columns using mutate to the olymp dataset

 olymp <- olymp %>%
```

```r
  mutate(gdp_per_cap = gdp_per_cap,,
         log_gdp = log_gdp,
         log_pop = log_pop,
         log_gdp_perC = log_gdp_perC,
         log_n = log_n,
         log_year = log_year)


mydata = olymp |> filter(medal == "Gold")       ## Filtering by "Gold" Metal and creation of
mydata <- mydata %>%
  mutate(centered_year = year - mean(year, na.rm = TRUE),
         centered_log_year = log_year - mean(log_year, na.rm = TRUE),
         centered_log_pop = log_pop - mean(log_pop, na.rm = TRUE),
         wealthy_class = case_when(
    log_gdp_perC <= 0.5 ~ 1,                      # If log_gdp_perC < 0, wealthy_class = 1
    log_gdp_perC > 0.5 & log_gdp_perC <= 2 ~ 2,   # If log_gdp_perC between 0 and 2, wealthy
    log_gdp_perC > 2 & log_gdp_perC <= 3.5 ~ 3,   # If log_gdp_perC between 2 and 4, wealthy
    log_gdp_perC > 3.5 ~ 4,                       # If log_gdp_perC > 4, wealthy_class = 4
      ))
mydata$team <- as.factor(mydata$team)
mydata$wealthy_class <- as.numeric(mydata$wealthy_class)
mydata <- mydata[!is.na(mydata$log_pop) & !is.na(mydata$log_gdp) &
                      !is.infinite(mydata$log_pop) & !is.infinite(mydata$log_gdp), ]


head(mydata,10)
```

```
# A tibble: 10 x 16
   team   year medal     n    gdp   pop gdp_per_cap log_gdp log_pop log_gdp_perC
   <fct> <int> <chr> <int>  <dbl> <dbl>       <dbl>   <dbl>   <dbl>        <dbl>
 1 Afgh~  2000 Gold      0   6.21  19.5       0.318    1.83    2.97       -
1.15
 2 Afgh~  2004 Gold      0   7.98  23.6       0.339    2.08    3.16       -
1.08
 3 Afgh~  2008 Gold      0  11.1   26.4       0.419    2.40    3.27       -
0.871
 4 Afgh~  2012 Gold      0  17.4   30.5       0.571    2.86    3.42       -
0.561
 5 Afgh~  2016 Gold      0  19.6   34.6       0.565    2.97    3.54       -
0.571
 6 Alge~  2000 Gold      1 110.    30.8       3.57     4.70    3.43        1.27
 7 Alge~  2004 Gold      0 133.    32.5       4.08     4.89    3.48        1.41
```

```
 8 Alge~  2008 Gold      0 152.    34.6       4.40     5.02    3.54        1.48
 9 Alge~  2012 Gold      1 170.    37.3       4.57     5.14    3.62        1.52
10 Alge~  2016 Gold      0 195.    40.3       4.83     5.27    3.70        1.57
# i 6 more variables: log_n <dbl>, log_year <dbl>, centered_year <dbl>,
#   centered_log_year <dbl>, centered_log_pop <dbl>, wealthy_class <dbl>
```

```
## table(mydata$wealthy_class)
## hist(log_gdp_perC)
```

In this analysis, the focus was on estimating gold medals from the *olymp1* dataset. After importing the dataset, it was transformed into a tibble format for more convenient handling. Five new variables were created by applying a logarithmic transformation to five key variables, reducing their dispersion and ensuring a smoother dataset for further analysis. Subsequently, a new tibble (*mydata*) was created, filtered to include only observations related to gold medals. This filtered dataset serves as the primary basis for the following analysis.

The `mydata` dataset is a filtered subset of the Olympic dataset, containing observations exclusively for countries that earned "Gold" medals. It comprises variables detailing economic and demographic characteristics of nations, along with their Olympic performance metrics. The dataset includes 16 variables: `team` (factor indicating the country's name), `year` (year of the Olympic event), `medal` (medal type, fixed as "Gold"), `n` (number of gold medals), `gdp` (Gross Domestic Product), `pop` (population), and derived variables such as `gdp_per_cap` (GDP per capita), `log_gdp`, `log_pop`, `log_gdp_perC` (logarithmic transformations of GDP, population, and GDP per capita), and `log_n` (logarithmic transformation of medal count). Additionally, it features time-centered variables (`centered_year` and `centered_log_year`) and a derived categorical variable, `wealthy_class`, which classifies nations into wealth tiers based on their logarithmic GDP per capita. Observations with missing or infinite values for `log_pop` or `log_gdp` were excluded to ensure data integrity.

a) Maximal Model Specification for Predicting Olympic Medal Counts with Fixed and Random Effects

Model Specification

Following the recommendations of Barr et al. (2013), we will begin with the maximal model to account for all theoretically justified random effects. This approach prioritizes capturing the structure of the experimental design to avoid anti-conservative inferences. If singularity or non-convergence issues arise, we will iteratively simplify the random-effects structure while maintaining theoretical plausibility.

Response Variable

The dependent variable for this analysis is the **number of gold medals won** (`n`), a count variable.

Fixed Effects

We will include the following fixed effects:

- **Logarithm of population** (`log_pop`): This variable accounts for the size of the population, which can influence the likelihood of winning gold medals.
- **Logarithm of GDP** (`log_gdp`): Captures economic strength, hypothesized to correlate with investment in sports.
- **Centered year** (`centered_year`): Centering the year ensures more interpretable estimates and captures variability associated with time trends.
- **Wealth class** (`wealthy_class`): Categorizes countries into economic groups to facilitate insights into disparities between economic classes.

Random Effects

The random-effects structure will model the variability attributable to **teams** (i.e., countries). Specifically, we will include:

- A **random intercept** to account for baseline differences in medal counts across teams.
- **Random slopes** for each fixed effect, allowing their effects to vary among teams.

Statistical Model

Given that the response variable is a count, we will use a **generalized linear mixed-effects model** with one of the following families:

- **Poisson regression** if the count data follows the Poisson distribution.
- **Negative binomial regression** if overdispersion is detected (variance exceeds the mean).

Justification of Model Components

- **Wealth Class as a Fixed Effect**: As noted by Gelman (2006), treating wealth class as a random effect is not appropriate given the small number of clusters (<5). Including it as a fixed effect ensures numerical stability.
- **Random Effects for Team**: This accounts for unobserved heterogeneity among teams, improving the generalizability of the model.

This approach allows for a robust analysis that accounts for both fixed and random effects while addressing the characteristics of count data

b) When to Include a Random Effects Term for Country × Year in Mixed-Effects Models: Key Considerations

Including a random effects term with the grouping variable `country:year` is appropriate when there is variability across the combinations of countries and years that cannot be explained by the fixed effects. This is especially relevant when the response variable is continuous and there are multiple observations per combination of `country:year`, such as when the data includes repeated measures of the same countries across different years. The random effects term accounts for the correlation of observations within each country-year combination, providing a more accurate estimate of the fixed effects by adjusting for unobserved heterogeneity. It is also suitable when the response distribution is Gaussian, as random effects models are robust for continuous data, where within-group correlation needs to be modeled for unbiased inference.

On the other hand, if most `country:year` combinations have only one or two observations (as in the case of countries winning medals in a particular year), the random effects cannot reliably capture variability within these groups. It doesn't make sense to include a random effect in the form of `country:year` when the distribution does not have a scale parameter because random effects are fundamentally tied to modeling variance components within a distribution. In distributions that lack a scale parameter, such as the **Poisson** or **binomial** distributions, the variance is inherently linked to the mean (in Poisson, the variance equals the mean, and in binomial, the variance is a function of both the probability of success and the number of trials). Random effects in such models are typically used to account for **unexplained variability** or **correlation** in the data. For distributions without a scale parameter, there is no independent variance to estimate, so introducing random effects may lead to model misspecification. In other words, the random effects term may not have any meaningful interpretation because the model already incorporates the necessary dependency structure through the mean-variance relationship of the distribution.

For example:

- **Poisson regression** models count data, where the variance is tied directly to the mean. Adding a random effect term like `country:year` might result in overfitting or unnecessary complexity, as the scale of the distribution is constrained by the mean.
- **Binomial regression** models proportions, where the variance also depends on the mean. Introducing a random effect term under these conditions can be redundant and could distort the model's behavior, especially when the data has relatively few observations per group or when variance already naturally scales with the mean.

In these cases, more appropriate methods like **fixed-effects models** or **generalized estimating equations (GEE)** may be preferred, as they focus on capturing correlations within clustered data without assuming a separate variance component that random effects require.

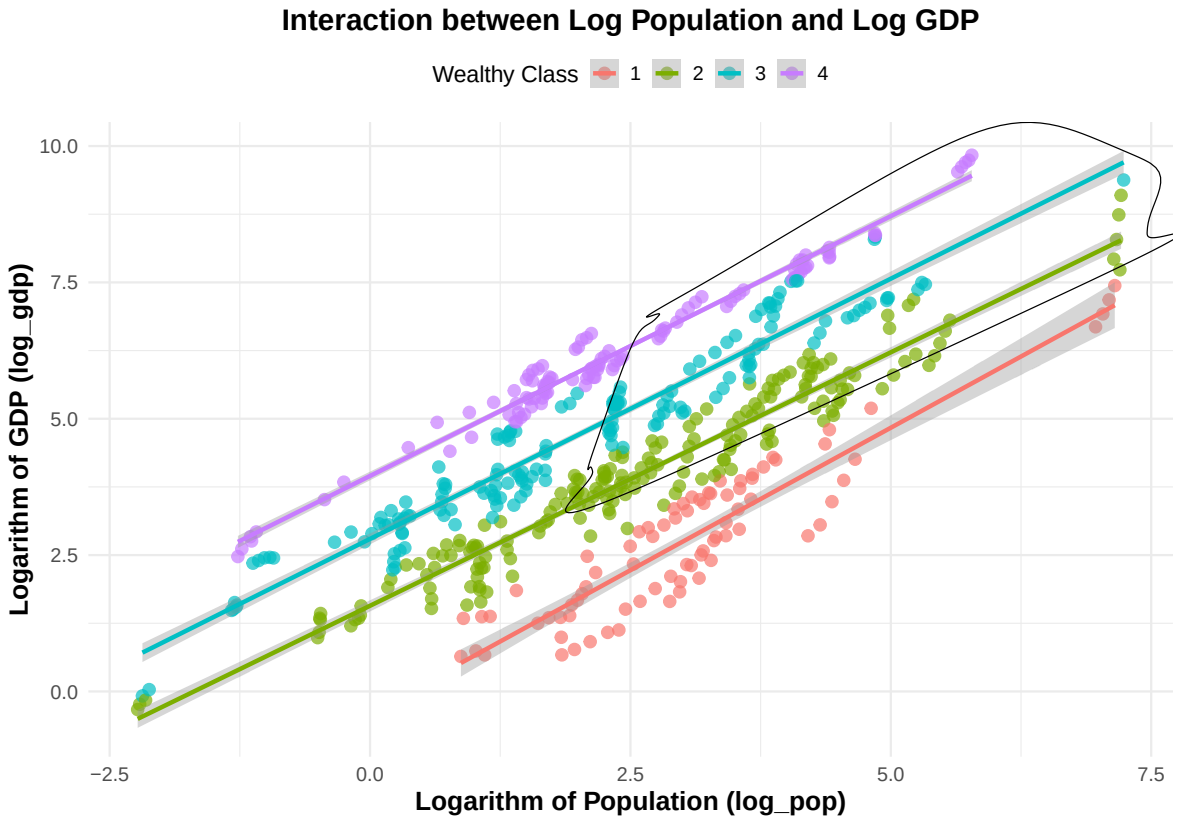
c and d) Defining the Initial Model: Balancing Complexity and Practicality and Visualizing the Dataset

Given the inherent nature of the problem, I have decided to employ a generalized mixed-effects model from either the Poisson or negative binomial family. The choice of the specific family will be guided by several key factors, including the level of dispersion in the data, the presence of zero inflation, and other distributional characteristics. I have opted not to use the maximal model due to its relative complexity and the high likelihood of encountering singularity and convergence issues. Instead, I will pursue a more practical and parsimonious approach to model specification. To inform the final decision regarding the appropriate family and structure of the model, I will conduct a series of diagnostic tests and generate visualizations to better understand the data. These preliminary analyses will provide essential insights into the response distribution, variability, and other relevant characteristics, ensuring that the chosen model is both robust and well-suited to the problem at hand.

Firstly, I will try to visualize the different relations between the variables of interest.

```
## Criterion for gold metals
high_medal <- mydata %>% filter(n > 20)


ggplot(mydata, aes(x = log_pop, y = log_gdp)) +
  geom_point(size = 3, aes(color = factor(wealthy_class)), alpha = 0.7) +  # Points colored l
  geom_smooth(aes(group = factor(wealthy_class), color = factor(wealthy_class)),
              method = "lm", se = TRUE, formula = 'y ~ x') + # Regression line for each grou
  ggalt::geom_encircle(data = high_medal, aes(x = log_pop, y = log_gdp), color = "black", siz
  labs(
    title = "Interaction between Log Population and Log GDP",
    x = "Logarithm of Population (log_pop)",
    y = "Logarithm of GDP (log_gdp)",
```

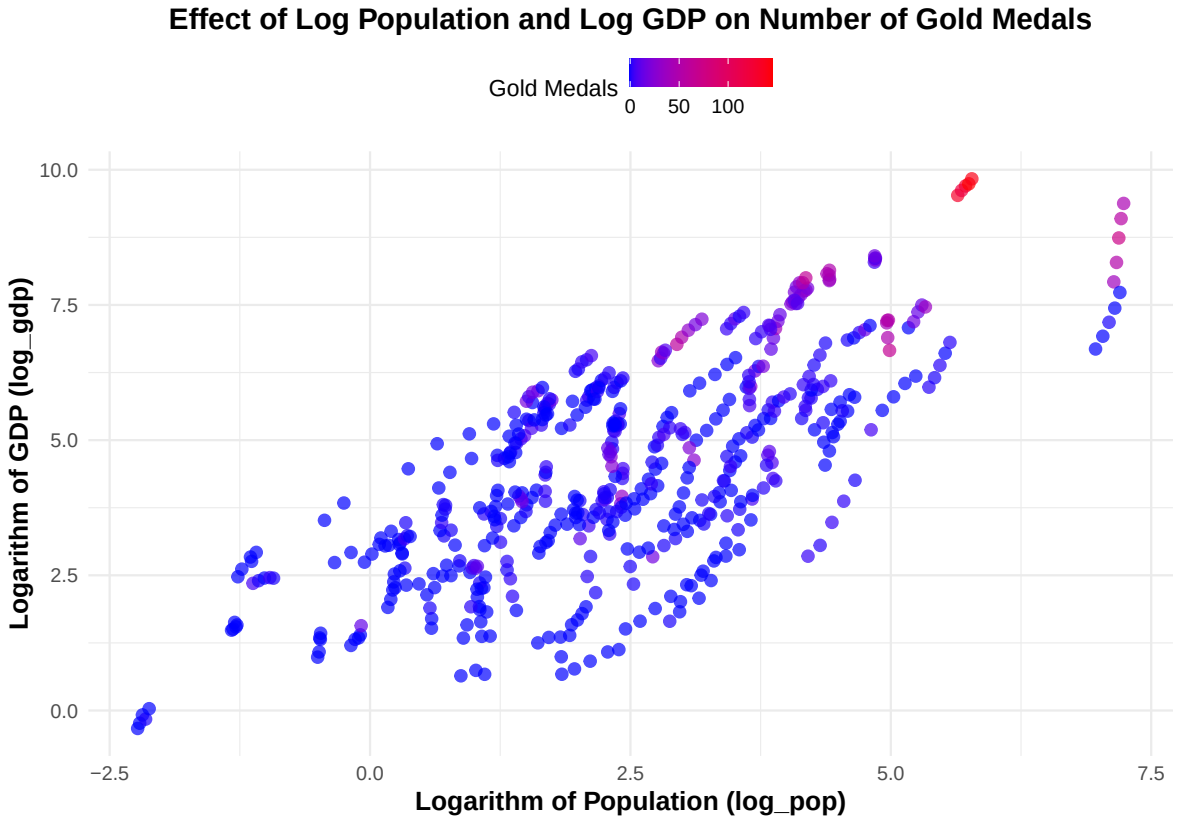
```
    color = "Wealthy Class") +
  theme_minimal(base_size = 14) +
  theme(
    axis.title = element_text(face = "bold", size = 16),
    axis.text = element_text(size = 12),
    plot.title = element_text(face = "bold", size = 18, hjust = 0.5),
    legend.position = "top",
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12))
```



The graph illustrates the interaction between the logarithm of population (`log_pop`) and the logarithm of GDP (`log_gdp`) across four wealth categories ("Wealthy Class"). A positive relationship is observed between population and GDP for all groups, with regression lines indicating that wealthier classes (e.g., classes 3 and 4) exhibit higher GDP levels for a given population size compared to less wealthy classes (e.g., classes 1 and 2). The encircled points represent countries with more than 20 medals, highlighting that high athletic success is generally associated with nations that have larger populations and higher GDPs. This suggests that economic resources and population size play a significant role in achieving athletic success,

as wealthier and more populous nations tend to dominate medal counts. The graph further underscores disparities in economic capacity and their potential influence on global sports performance.

```r
# Scatter plot with color indicating the number of gold medals
ggplot(mydata, aes(x = log_pop, y = log_gdp, color = n)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_gradient(low = "blue", high = "red") +
  labs(
    title = "Effect of Log Population and Log GDP on Number of Gold Medals",
    x = "Logarithm of Population (log_pop)",
    y = "Logarithm of GDP (log_gdp)",
    color = "Gold Medals"
  ) +
 theme_minimal(base_size = 14) +
 theme(
    axis.title = element_text(face = "bold", size = 16),
    axis.text = element_text(size = 12),
    plot.title = element_text(face = "bold", size = 18, hjust = 0.5),
    legend.position = "top",
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12))
```



The scatter plot depicts the relationship between the logarithm of population (`log_pop`) and the logarithm of GDP (`log_gdp`), with the color intensity of each point representing the number of gold medals won by a country. A positive correlation is observed between population and GDP, as countries with larger populations tend to have higher GDPs. Additionally, countries that have won more gold medals are represented by red-shaded points, predominantly located in regions of high `log_pop` and `log_gdp`. This trend suggests that nations with both substantial populations and strong economic capacity are more likely to achieve greater success in terms of gold medal counts. The gradient from blue (fewer medals) to red (more medals) further highlights the influence of economic and demographic factors on athletic performance at an international level.

Some additional graphs indicating this correlation are:

```
#| warning: false
#| message: false
p1 <- ggplot(mydata, aes(x = log_gdp, y = n,
                         color = factor(wealthy_class),
                         group = wealthy_class)) +
  geom_point(alpha = 0.5) +
```

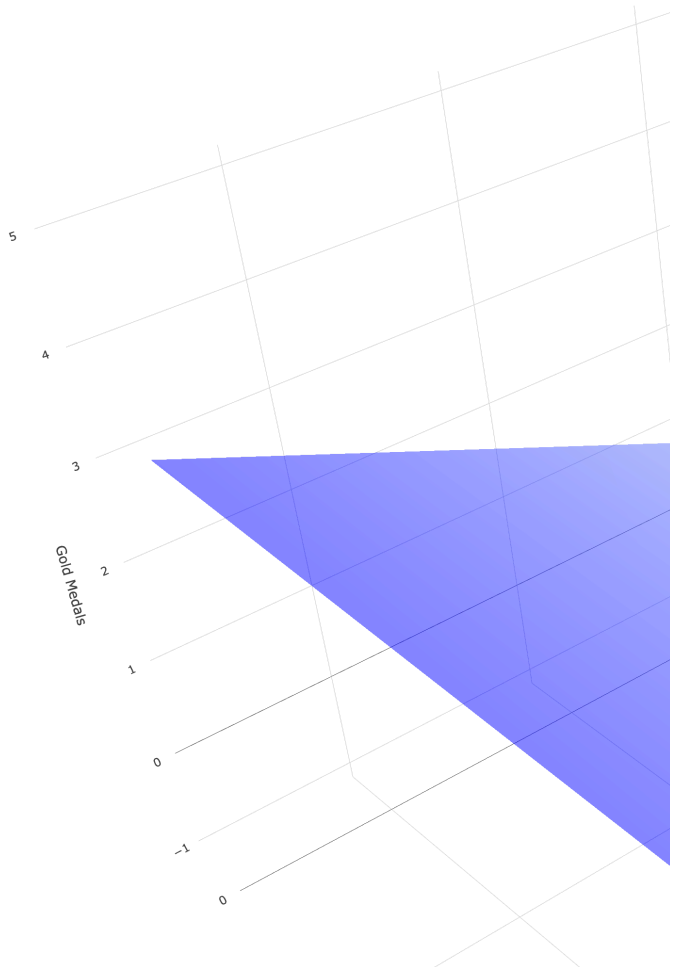
```r
  geom_smooth(
    method = "glm",
    formula = y ~ x,
    method.args = list(family = quasipoisson),
    se = FALSE
  ) +
  labs(
    title = "Gold Medals vs Log GDP by Wealth Class",
    x = "Log GDP",
    y = "Number of Gold Medals",
    color = "Wealth Class"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.title = element_text(face = "bold", size = 16),
    axis.text = element_text(size = 12),
    plot.title = element_text(face = "bold", size = 18, hjust = 0.5),
    legend.position = "top",
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12)
  )


p2 <- ggplot(mydata, aes(x = log_pop, y = n,
                         color = factor(wealthy_class),
                         group = wealthy_class)) +
  geom_point(alpha = 0.5) +
  geom_smooth(
    method = "glm",
    formula = y ~ x,
    method.args = list(family = quasipoisson),
    se = FALSE
  ) +
  labs(
    title = "Gold Medals vs Log Population by Wealth Class",
    x = "Log Population",
    y = "Number of Gold Medals",
    color = "Wealth Class"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    axis.title = element_text(face = "bold", size = 16),
```

```r
    axis.text = element_text(size = 12),
    plot.title = element_text(face = "bold", size = 18, hjust = 0.5),
    legend.position = "top",
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12)
  )


p3 <- ggplot(mydata, aes(x = centered_log_pop, y = log_gdp, colour = team)) +
  geom_point(alpha = 0.5) +
  stat_ellipse(aes(fill = team), geom = "polygon", alpha = 0.2) +
  labs(
    title = "Elliptical Clusters of GDP and Population by Team",
    x = "Centered Log(Population)",
    y = "Log(GDP)"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 18, hjust = 0.5),
    axis.title = element_text(face = "bold", size = 16),
    axis.text = element_text(size = 12),
    legend.position = "none"
  )

(p1 | p2) / p3 +
  plot_annotation(
    title = "Economic Indicators and Olympic Gold Medals",
    theme = theme(
      plot.title = element_text(face = "bold", size = 20, hjust = 0.5)
    )
  )
```

Bellow I will create an interactive 3D scatter plot capturing the effect of population and GDP on the number of gold medals:

```r
# Fit the Negative Binomial GLM model
model <- lm(log_n ~ log_pop + log_gdp, data = mydata)

# Create a grid of values for log_pop and log_gdp_perC to calculate the predicted surface
grid_data <- expand.grid(
  log_pop = seq(min(mydata$log_pop), max(mydata$log_pop), length.out = 50),
  log_gdp = seq(min(mydata$log_gdp), max(mydata$log_gdp), length.out = 50)
)

# Add predictions for the grid using the Negative Binomial model
```

```r
grid_data$n <- predict(model, newdata = grid_data, type = "response")


# 3D Scatter plot with linear surface
plot_ly() %>%
  # Add scatter plot points
  add_markers(
    data = mydata,
    x = ~log_pop,
    y = ~log_gdp,
    z = ~log_n,
    marker = list(
      size = ~log_pop * 3,  # Dynamically adjust size based on log population
      color = ~log_gdp,  # Color by log GDP
      colorscale = "Plasma",
      opacity = 0.7,
      line = list(width = 1, color = 'black')
    ),
    hoverinfo = "text",
    text = ~paste(
      "Country: ", team, "<br>",
      "Log Population: ", round(log_pop, 2), "<br>",
      "Log GDP: ", round(log_gdp, 2), "<br>",
      "Gold Medals: ", n, "<br>"
    )
  ) %>%
  # Add linear surface
  add_surface(
    x = unique(grid_data$log_pop),
    y = unique(grid_data$log_gdp),
    z = matrix(grid_data$n, nrow = 50, ncol = 50),
    opacity = 0.5,
    colorscale = list(c(0, 1), c("lightblue", "blue"))
  ) %>%
  layout(
    title = "Effect of Population and GDP on Gold Medals with Linear Surface",
    scene = list(
      xaxis = list(title = "Log Population"),
      yaxis = list(title = "Log GDP"),
      zaxis = list(title = "Gold Medals")
    )
  )
```


Gold Medals
5
4
3
2
1
0
−1
0

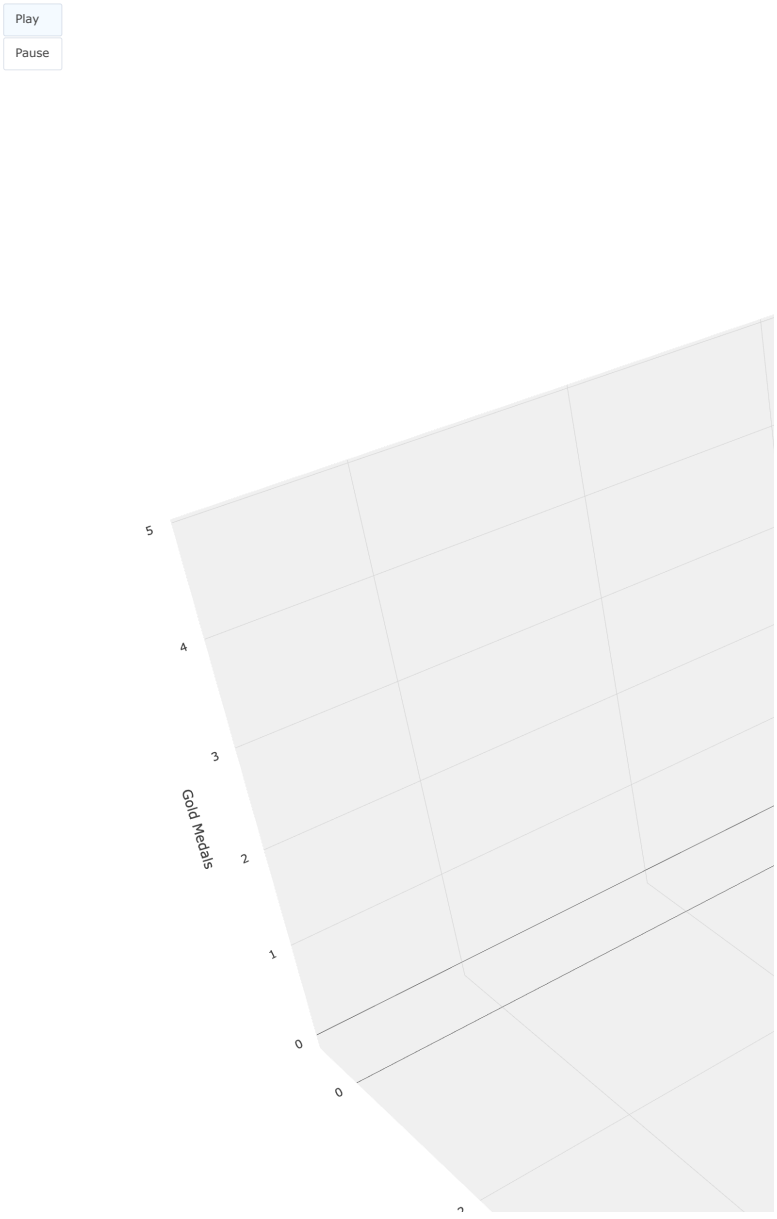
The provided plot explores the relationship between a country's population (log-transformed), GDP (log-transformed), and the number of gold medals won at the Olympics, with a linear surface included for reference. While the linear plane provides a simplified approximation, the distribution of points reveals that the relationship is not strictly linear, as evident from the variability and clustering of data points around the surface, Moreover it seem like there is heteroscedasticity within the data. This suggests that the impact of economic and demographic factors on Olympic success is influenced by non-linear dynamics, and of course that there are hidden structures and variables, such as cultural emphasis on sports, investment in athletic programs, and historical performance trends. The use of log-transformed variables aids in managing the wide disparities in GDP and population among countries, allowing for a more nuanced interpretation of the observed patterns.

For the sake of completeness, below I provide an animated version of the same graph, which displays the data over time. This animation allows for a clearer visualization of the evolving structures across different years.

```
plot_ly(
  mydata,
  x = ~log_pop,
  y = ~log_gdp,
  z = ~log_n,
  frame = ~year,  # Animate by year
  type = "scatter3d",
  mode = "markers",
  marker = list(
    size = ~log_pop * 3,  # Dynamically adjust size based on log population
    color = ~log_gdp,  # Color by log GDP
    colorscale = "Plasma",  # Plasma color scale for a vibrant effect
    colorbar = list(title = "Log GDP"),  # Color scale title
    opacity = 0.7,
    symbol = 'circle',
    line = list(width = 1, color = 'black')
  ),
  hoverinfo = "text",
  text = ~paste(
    "Country: ", team, "<br>",
    "Year: ", year, "<br>",
    "Log Population: ", round(log_pop, 2), "<br>",
    "Log GDP: ", round(log_gdp, 2), "<br>",
    "Gold Medals: ", n, "<br>",
    "Rank by Medals: ", rank(-n)
  )
) %>%
```

```
  layout(
    title = "3D Effect of Log Population and Log GDP on Gold Medals (Animated by Year)",
    scene = list(
      xaxis = list(
        title = "Log Population",
        gridcolor = "lightgray",
        showbackground = TRUE,
        backgroundcolor = "rgb(240, 240, 240)",
        zeroline = TRUE
      ),
      yaxis = list(
        title = "Log GDP",
        gridcolor = "lightgray",
        showbackground = TRUE,
        backgroundcolor = "rgb(240, 240, 240)",
        zeroline = TRUE
      ),
      zaxis = list(
        title = "Gold Medals",
        gridcolor = "lightgray",
        showbackground = TRUE,
        backgroundcolor = "rgb(240, 240, 240)",
        zeroline = TRUE
      )
    ),
    margin = list(l = 50, r = 50, b = 50, t = 100),
    paper_bgcolor = "rgb(255, 255, 255)",
    plot_bgcolor = "rgb(240, 240, 240)",
    showlegend = FALSE,
    updatemenus = list(
      list(
        type = "buttons",
        x = 0.1,
        y = 0.95,
        buttons = list(
          list(
            method = "animate",
            args = list(NULL, list(frame = list(duration = 500, redraw = TRUE), fromcurrent =
            label = "Play"
          ),
          list(
            method = "animate",
```

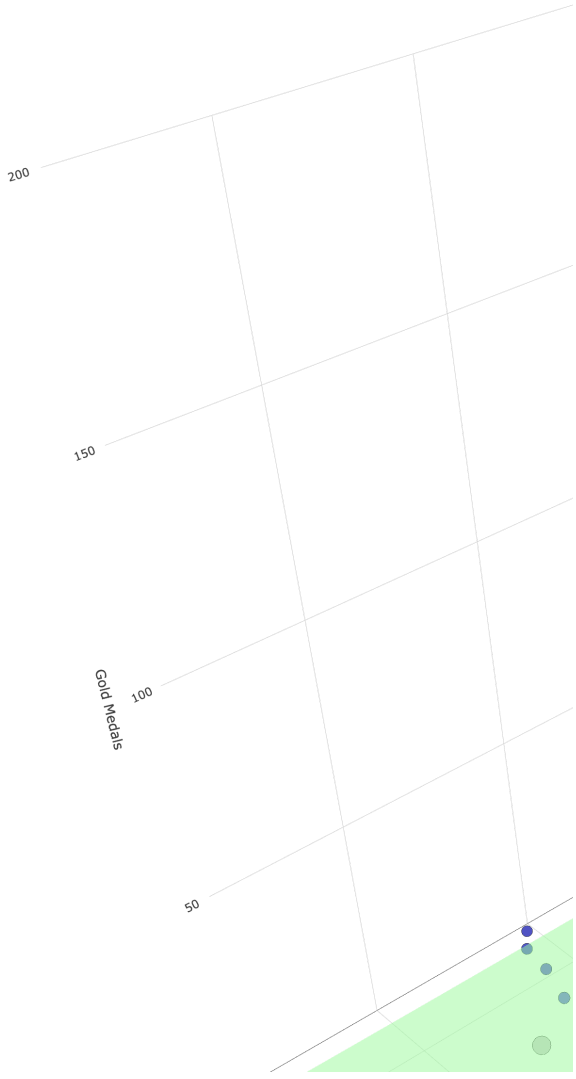
```
            args = list(NULL, list(frame = list(duration = 0, redraw = TRUE), mode = "immedi
            label = "Pause"
          )
        )
      )
    )
  )
```


Play
Pause
Gold Medals
5
4
3
2
1
0
0

Again the interpretation is similar with similar structure across the years. Bellow I will provide the same plot with the regression surface from the first analysis (for comparison reasons)

```
# Fit the Negative Binomial GLM model
nb_model <- glm.nb(n ~ log_pop + log_gdp_perC, data = mydata)

# Create a grid of values for log_pop and log_gdp_perC to calculate the predicted surface
grid_data_nb <- expand.grid(
  log_pop = seq(min(mydata$log_pop), max(mydata$log_pop), length.out = 50),
  log_gdp_perC = seq(min(mydata$log_gdp_perC), max(mydata$log_gdp_perC), length.out = 50)
)

# Add predictions for the grid using the Negative Binomial model
grid_data_nb$n <- predict(nb_model, newdata = grid_data_nb, type = "response")

# 3D scatter plot with the Negative Binomial model surface
plot_ly() %>%
  # Add scatter plot points
  add_markers(
    data = mydata,
    x = ~log_pop,
    y = ~log_gdp_perC,
    z = ~n,
    marker = list(
      size = ~log_pop * 3,  # Dynamically adjust size based on log population
      color = ~log_gdp_perC,  # Color by log GDP per capita
      colorscale = "Plasma",  # Plasma color scale for a vibrant effect
      opacity = 0.7,  # Set opacity for better clarity
      line = list(width = 1, color = 'black')  # Add black outline to markers
    ),
    hoverinfo = "text",
    text = ~paste(
      "Country: ", team, "<br>",
      "Log Population: ", round(log_pop, 2), "<br>",
      "Log GDP per Capita: ", round(log_gdp_perC, 2), "<br>",
      "Gold Medals: ", n, "<br>"
    )
  ) %>%
  # Add surface based on the Negative Binomial model predictions
  add_surface(
    x = unique(grid_data_nb$log_pop),
    y = unique(grid_data_nb$log_gdp_perC),
    z = matrix(grid_data_nb$n, nrow = 50, ncol = 50),
```

```
    opacity = 0.5,  # Set surface opacity
    colorscale = list(c(0, 1), c("lightgreen", "green"))  # Color scale for the surface
  ) %>%
  layout(
    title = "Effect of Log Population and Log GDP per Capita on Gold Medals (Negative Binomia
    scene = list(
      xaxis = list(title = "Log Population"),
      yaxis = list(title = "Log GDP per Capita"),
      zaxis = list(
        title = "Gold Medals",
        range = c(0, 200)  # Restrict Z-axis to 0-200 for better visualization
      )
    )
  )
```


200
150
100
50
Gold Medals

The graph demonstrates a reasonably good fit of the model from the initial analysis. Based on these illustrations, I have decided to include the following fixed effects in my new model: the logarithm of population, the logarithm of GDP, the centered year, and the interaction between the natural spline terms of the logarithms of population and the logarithm of GDP with 3 degrees of freedom, each. Additionally, the intercept will vary by country (team). This model setup is designed to capture both linear and non-linear relationships within the data. To determine the appropriate family (Poisson or negative binomial to account for overdispersion), I will now proceed with a test for overdispersion.

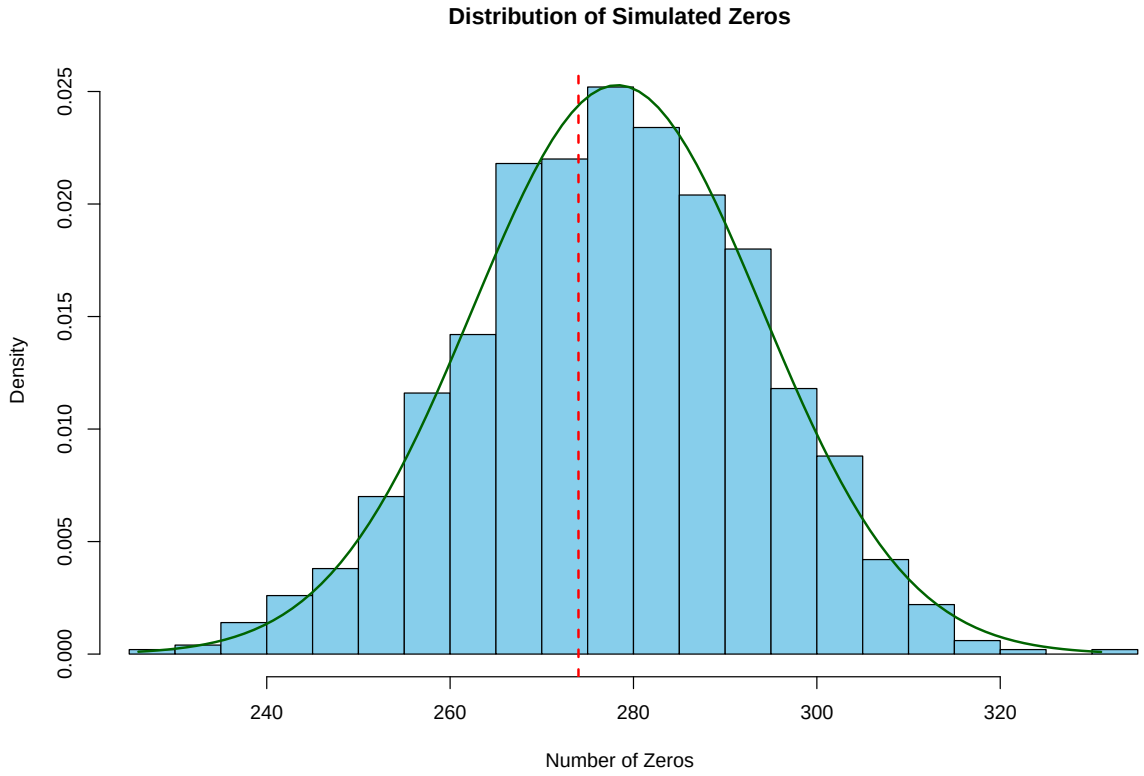
```
m <- glmmTMB(
  n ~ 1 + log_pop + log_gdp + ns(log_pop, df = 3):ns(log_gdp, df = 3) + centered_year + (1|te
  data = mydata,
 family = poisson(link = "log")
 )

overdisp_fun <- function(m) {
    rdf <- df.residual(m)
    rp <- residuals(m,type="pearson")
    Pearson.chisq <- sum(rp^2)
    prat <- Pearson.chisq/rdf
    pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=FALSE)
    c(chisq=Pearson.chisq,ratio=prat,rdf=rdf,p_value=pval)
}
overdisp_fun(m)
```

```
       chisq        ratio          rdf      p_value
1.007970e+03 1.912656e+00 5.270000e+02 1.571927e-32
```

To assess the adequacy of the Poisson regression model fitted using the `glmmTMB` package, we evaluated overdispersion using the Pearson chi-squared statistic. The computed Pearson chi-squared value was 1007.97, with 527 degrees of freedom, resulting in a dispersion ratio of 1.91. A dispersion ratio greater than 1 indicates the presence of overdispersion, suggesting that the variability in the data exceeds what is expected under the Poisson distribution. The associated p-value was extremely small ($p < 2 \times 10-16$), further confirming that overdispersion is statistically significant. These results suggest that the Poisson model may not adequately capture the observed data's variability, and alternative models, such as those accounting for overdispersion (e.g., Negative Binomial models), should be considered.

To further investigate the characteristics of the data, I will run a simulation to assess the presence of an unexpectedly high number of zeros. This will help determine whether a zero-inflation model is necessary to properly account for an excess of zero counts in the data. The results of this simulation will guide the decision on whether incorporating zero inflation is warranted in the modeling process.

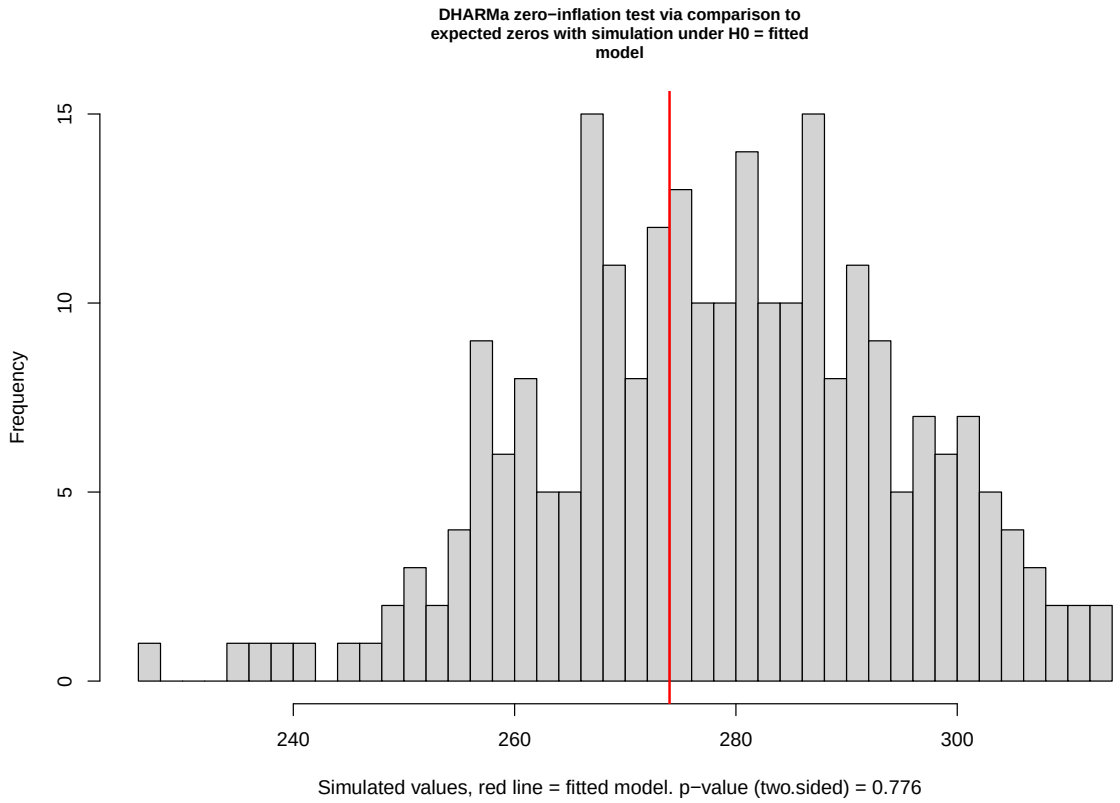
```r
set.seed(123)

model <- glmmTMB(
  n ~ 1 + log_pop + log_gdp + ns(log_pop, df = 3):ns(log_gdp, df = 3) + centered_year + (1|t
  ziformula = ~ 0,
  dispformula = ~ 1,
  data = mydata,
 family = nbinom12(link = "log")
 )

# Simulate new responses from the negative binomial model
sim_vals <- simulate(model, nsim = 1000)
# Calculate the number of zeros in each simulation and in the observed data
zeros_sim <- colSums(sim_vals == 0)
zeros_obs <- sum(mydata$n == 0)

# Create a histogram of the number of zeros in the simulated data
hist(zeros_sim, breaks = 30, main = "Distribution of Simulated Zeros",
     xlab = "Number of Zeros", col = "skyblue", border = "black",
     xlim = range(c(zeros_sim, zeros_obs), na.rm = TRUE),
     freq = FALSE)

# Add a vertical line representing the observed number of zeros
abline(v = zeros_obs, col = "red", lwd = 2, lty = 2)

# Overlay the normal distribution curve for reference
curve(dnorm(x, mean = mean(zeros_sim), sd = sd(zeros_sim)),
      add = TRUE, col = "darkgreen", lwd = 2)
```




```
# Calculate p-value based on simulation results
p_value <- mean(zeros_sim >= zeros_obs)
cat("P-value from simulation: ", p_value, "\n")
```

```
P-value from simulation:  0.626
```

```
# Perform DHARMa test for zero-inflation
simulation_output <- simulateResiduals(fittedModel = model)
dh_test <- testZeroInflation(simulation_output)
```




```
cat("P-value from DHARMa test: ", dh_test$p.value, "\n")
```

```
P-value from DHARMa test:  0.776
```

Based on the exploratory analysis and data characteristics, I have determined that a zero-inflation model is not necessary for this dataset. Consequently, I will employ a negative binomial regression mixed model. This choice is motivated by the following considerations:

1. **Count Data with Overdispersion**: The dependent variable represents count data, and preliminary analysis indicates the presence of overdispersion (variance exceeding the mean), making the negative binomial model more appropriate than a Poisson model.
2. **Lack of Zero Inflation**: An inspection of the response variable distribution shows no substantial excess of zeros that would warrant the use of a zero-inflated model.
3. **Incorporating Random Effects**: The hierarchical nature of the data, with observations nested within groups (e.g., teams or countries), makes a sensible decision the inclusion of random effects to account for group-level variability.

The negative binomial regression mixed model will enable a robust analysis of the relationship between predictors and the response variable while appropriately addressing the overdispersion and hierarchical structure of the data.

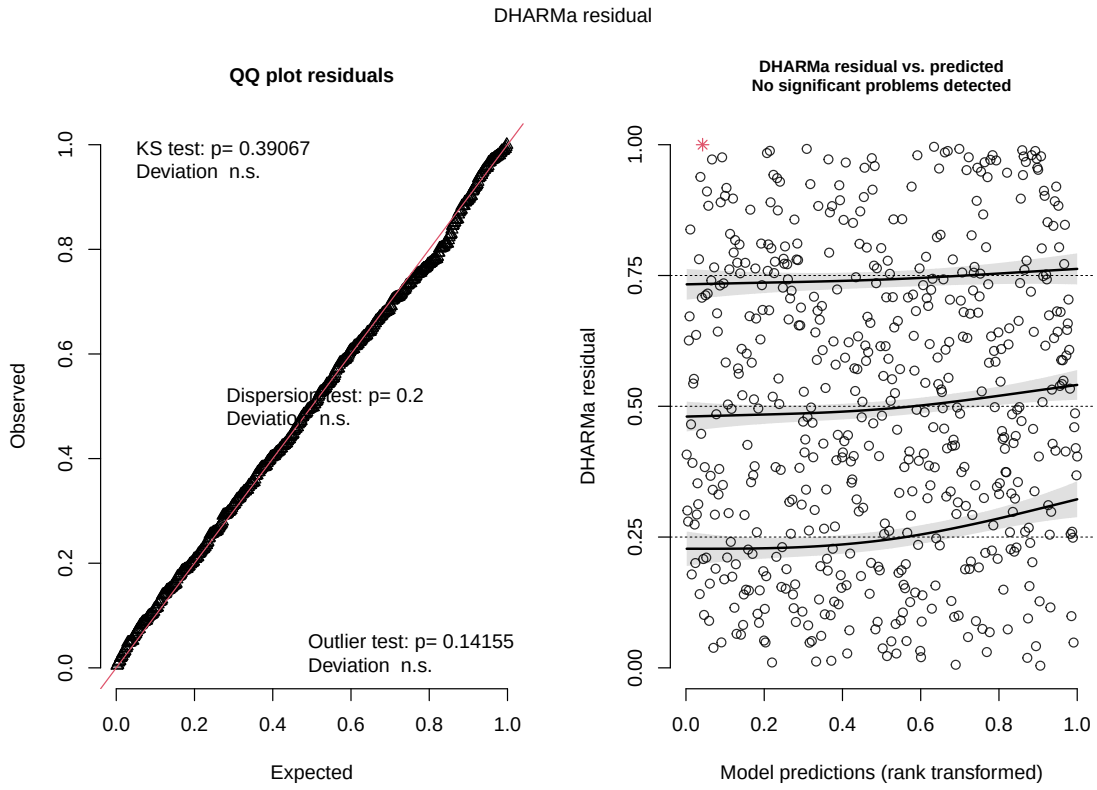
e) Model Fit and Diagnostics

```r
# Model fit
model <- glmmTMB(
  n ~ 1 + log_pop + log_gdp + ns(log_pop, df = 3):ns(log_gdp, df = 3) + centered_year + (1|te
  ziformula = ~ 0,
  dispformula = ~1,
  data = mydata,
 family = nbinom1(link = "log")
 )

summary(model)
```

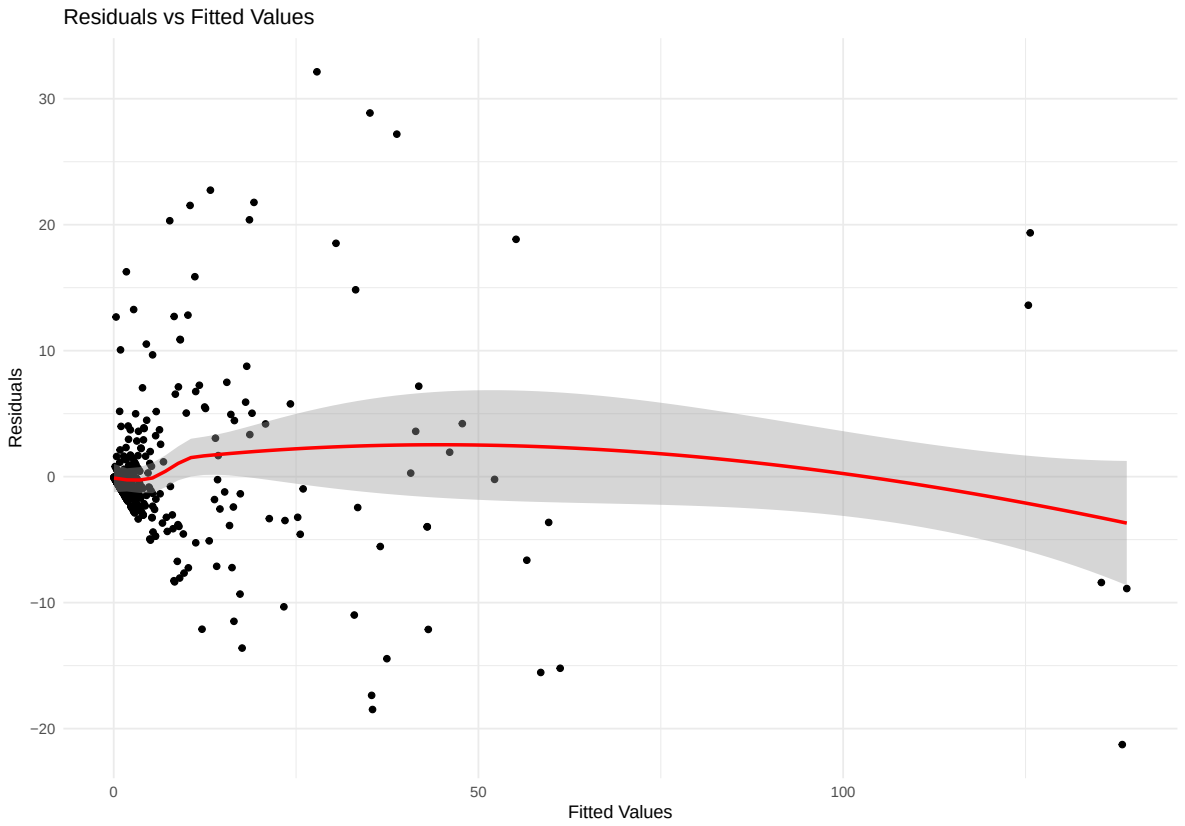
```
 Family: nbinom1  ( log )
Formula:
n ~ 1 + log_pop + log_gdp + ns(log_pop, df = 3):ns(log_gdp, df = 3) +
    centered_year + (1 | team)
Data: mydata

      AIC       BIC    logLik -2*log(L)  df.resid
   1996.3    2060.7    -983.1    1966.3       526

Random effects:

Conditional model:
 Groups Name        Variance Std.Dev.
 team   (Intercept) 1.225    1.107
Number of obs: 541, groups:  team, 109

Dispersion parameter for nbinom1 family ():  3.5

Conditional model:
                                        Estimate Std. Error z value Pr(>|z|)
(Intercept)                            -3.424196   1.030880  -3.322 0.000895
log_pop                                -0.940909   0.571611  -1.646 0.099750
log_gdp                                -0.446034   0.770755  -0.579 0.562793
```

```
centered_year                                -0.022103   0.007238  -3.054 0.002261
ns(log_pop, df = 3)1:ns(log_gdp, df = 3)1  3.725862   3.816835   0.976 0.328983
ns(log_pop, df = 3)2:ns(log_gdp, df = 3)1 12.902521   8.743490   1.476 0.140032
ns(log_pop, df = 3)3:ns(log_gdp, df = 3)1  8.522550   5.297929   1.609 0.107691
ns(log_pop, df = 3)1:ns(log_gdp, df = 3)2 16.498703   8.073483   2.044 0.040996
ns(log_pop, df = 3)2:ns(log_gdp, df = 3)2 41.255653  22.997115   1.794 0.072821
ns(log_pop, df = 3)3:ns(log_gdp, df = 3)2 15.200190  12.141897   1.252 0.210614
ns(log_pop, df = 3)1:ns(log_gdp, df = 3)3  8.517357   6.975405   1.221 0.222065
ns(log_pop, df = 3)2:ns(log_gdp, df = 3)3 26.126432  14.718697   1.775 0.075890
ns(log_pop, df = 3)3:ns(log_gdp, df = 3)3  5.381917   6.622150   0.813 0.416382

(Intercept)                                ***
log_pop                                    .
log_gdp
centered_year                              **
ns(log_pop, df = 3)1:ns(log_gdp, df = 3)1
ns(log_pop, df = 3)2:ns(log_gdp, df = 3)1
ns(log_pop, df = 3)3:ns(log_gdp, df = 3)1
ns(log_pop, df = 3)1:ns(log_gdp, df = 3)2 *
ns(log_pop, df = 3)2:ns(log_gdp, df = 3)2 .
ns(log_pop, df = 3)3:ns(log_gdp, df = 3)2
ns(log_pop, df = 3)1:ns(log_gdp, df = 3)3
ns(log_pop, df = 3)2:ns(log_gdp, df = 3)3 .
ns(log_pop, df = 3)3:ns(log_gdp, df = 3)3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Check for singularity
# vc <- VarCorr(model)$cond$team; eigen(vc)
# performance::check_model(model)
# Create a DHARMa residuals object
residuals_dharma <- simulateResiduals(model)
# Plot the residuals
plot(residuals_dharma)
```




```r
# Extract fitted values and residuals
fitted_values <- fitted(model)
residuals <- residuals(model)

# Plot residuals vs fitted values
ggplot(data.frame(fitted = fitted_values, residuals = residuals), aes(x = fitted, y = residua
  geom_point() +
  geom_smooth(method = "loess", color = "red", formula = 'y ~ x') +
  labs(title = "Residuals vs Fitted Values", x = "Fitted Values", y = "Residuals") +
  theme_minimal()
```




```r
r2.corr.mer <- function(m) {
   lmfit <-  lm(model.response(model.frame(m)) ~ fitted(m))
   summary(lmfit)$r.squared
}

cat("R^2 = ", r2.corr.mer(model), "\n")
```

```
R^2 =  0.9080412
```

```r
performance::r2(model)
```

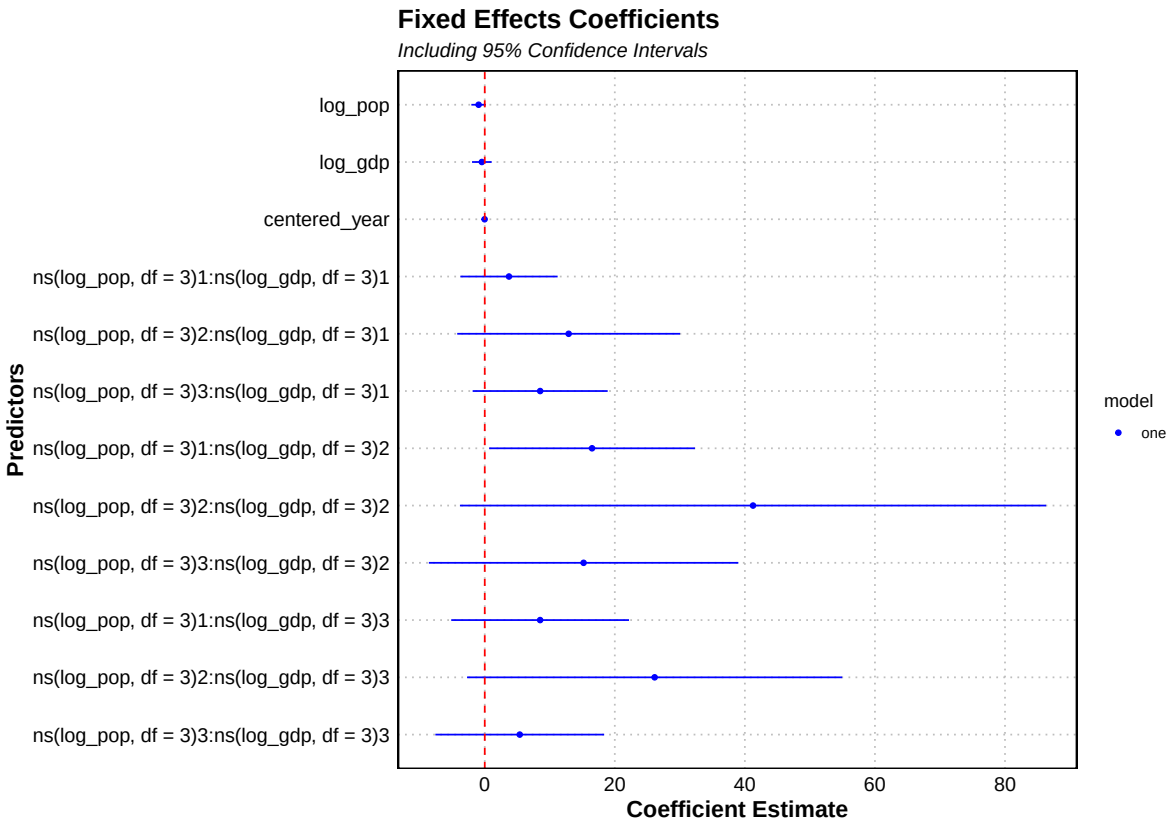
```
# R2 for Mixed Models

  Conditional R2: 0.886
     Marginal R2: 0.503
```

The diagnostic assessment of the fitted mixed-effects model using DHARMa residual diagnostics reveals no significant deviations from model assumptions. The QQ plot of residuals indicates

no substantial departures from normality, with p-values for the Kolmogorov-Smirnov (KS) test, dispersion test, and outlier test all being nonsignificant ($p > 0.1$). Similarly, the residuals versus predicted values plot shows no discernible patterns, suggesting homoscedasticity. The conditional R^2 value for the model is 0.886, indicating that the full model (fixed and random effects combined) explains approximately 88.6% of the variance. The marginal R^2 value of 0.503 reflects that fixed effects alone account for 50.3% of the variance. The calculated R^2 using the auxiliary `r2.corr.mer` function is 0.908, corroborating the high explanatory power of the model. These results suggest the model provides a robust fit to the data without significant violations of underlying assumptions.

f) Visualizing the Results

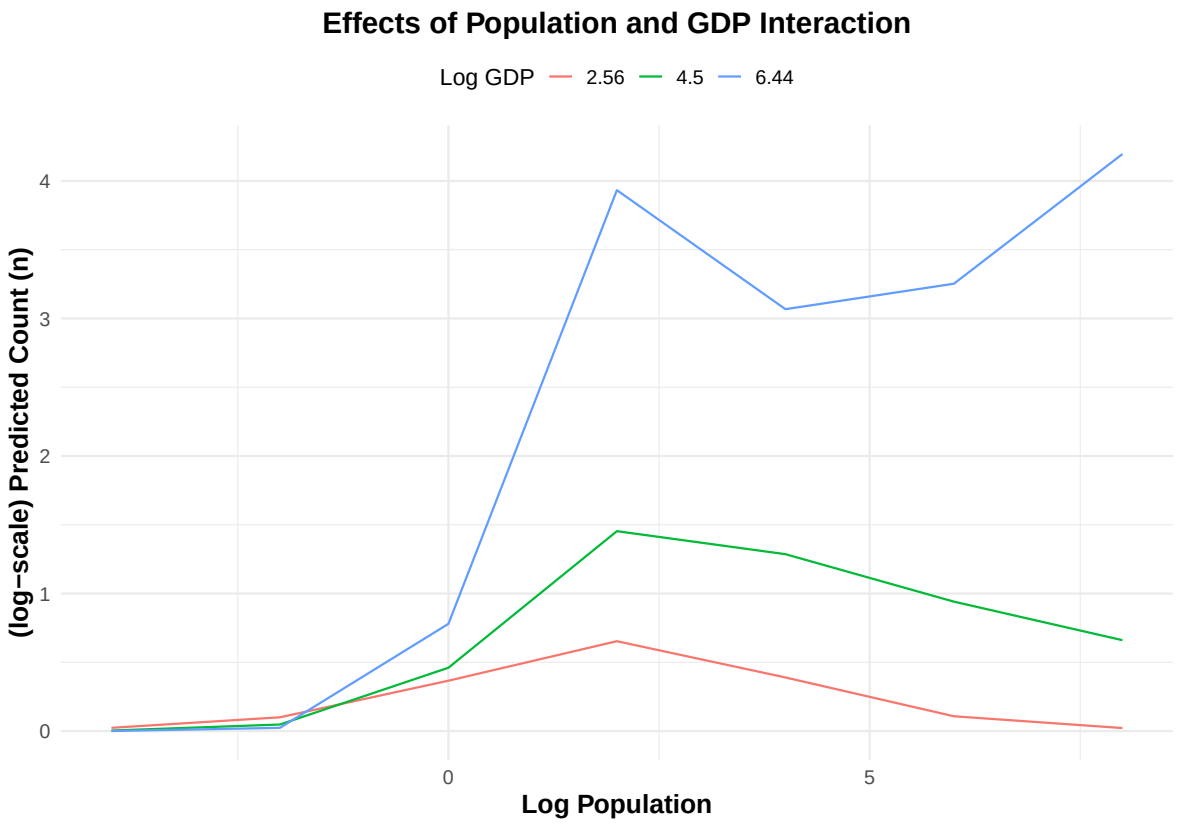
```r
# Extract fixed effects with confidence intervals
fixed_effects <- broom.mixed::tidy(model, effects = "fixed", conf.int = TRUE)


dwplot(fixed_effects) +
  theme_minimal() +
  labs(
    title = "Fixed Effects Coefficients",
    subtitle = "Including 95% Confidence Intervals",
    x = "Coefficient Estimate",
    y = "Predictors"
  ) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red") +
  theme(
    axis.text = element_text(size = 12, color = "black"),
    axis.title = element_text(size = 14, face = "bold"),
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic"),
    panel.grid.major = element_line(color = "gray", linetype = "dotted"),
    panel.grid.minor = element_blank(),
    panel.border = element_rect(color = "black", fill = NA, linewidth = 0.8)  # Updated size
  ) +
  scale_color_manual(values = c("blue", "red"))  # Customize the color scheme for better visu
```



The plot above visualizes the fixed-effect coefficients from a regression model, along with their 95% confidence intervals. The predictors include log-transformed population size (`log_pop`), log-transformed GDP (`log_gdp`), a centered measure of time (`centered_year`), and interaction terms involving natural splines of `log_pop` and `log_gdp`. While the coefficients for `log_pop`, `log_gdp`, and `centered_year` are relatively small with tight confidence intervals, the interaction terms involving spline components exhibit larger coefficient estimates with wider confidence intervals, indicating greater variability and potential non-linear relationships. Notably, many confidence intervals overlap the reference line at zero, suggesting that some predictors may not have statistically significant effects. This highlights the complex interplay between predictors and the importance of assessing interaction effects in modeling.

```
# Generate predictions for the interaction effect
effect_data <- ggpredict(model, terms = c("log_pop", "log_gdp"))

# Effects plot
ggplot(effect_data, aes(x = x, y = predicted, color = group)) +
  geom_line() +
  theme_minimal() +
  labs(title = "Effects of Population and GDP Interaction",
```

```
       x = "Log Population",
       y = "(log-scale) Predicted Count (n)",
       color = "Log GDP")+
  theme_minimal(base_size = 14) +
 theme(
    axis.title = element_text(face = "bold", size = 16),
    axis.text = element_text(size = 12),
    plot.title = element_text(face = "bold", size = 18, hjust = 0.5),
    legend.position = "top",
    legend.title = element_text(size = 14),
    legend.text = element_text(size = 12))
```



The plot illustrates the interaction effects between log-transformed population size (`log_pop`) and log-transformed GDP (`log_gdp`) on the predicted response variable, measured on a log scale. The curves represent predictions for different levels of `log_gdp` (2.56, 4.5, and 6.44), showing how the predicted count varies with changes in population size. At lower values of `log_pop`, the predicted counts are relatively small across all GDP levels. As `log_pop` increases, the predicted response rises significantly, especially for higher GDP levels (e.g., `log_gdp = 6.44`),

demonstrating a synergistic interaction. However, the trajectories differ; while predictions for the highest GDP level continue to rise at higher population sizes, predictions for lower GDP levels plateau or decline. This suggests a complex interplay between economic and population factors in shaping the outcome.

2. Analysis of toenail Dataset from HSAUR3 Package

Description of the Dataset

The **Toenail Infection Dataset** consists of data from a clinical trial designed to compare the effectiveness of two oral antifungal treatments, **itraconazole** and **terbinafine**, for treating toenail infections caused by dermatophytes (onychomycosis). The dataset includes 1,908 observations across 378 patients, each uniquely identified by a `patientID`. The key variables include `outcome`, which represents the degree of separation between the nail plate and the nail bed (onycholysis), and is categorized into moderate or severe versus none or mild; `treatment`, a factor indicating the administered medication (itraconazole or terbinafine); `time`, the actual visit time in months; and `visit`, indicating the number of the visit. The study followed patients over seven visits, initially scheduled at weeks 0, 4, 8, 12, 24, 36, and 48, although the timing of actual visits varied. The data is unbalanced, as not all patients attended all seven visits, reflecting real-world deviations from the planned schedule. This dataset provides valuable insights into the treatment's impact over time on the severity of onycholysis in toenail infection patients.

```r
# Load the toenail data
data("toenail")

mydata <- toenail
mydata <- mydata %>%
  mutate(outcome_binary = ifelse(outcome %in% c("none or mild"), 0, 1))
mydata$patientID <- as.factor(mydata$patientID)


head(mydata)
```

```
  patientID            outcome   treatment       time visit outcome_binary
1         1 moderate or severe terbinafine  0.0000000     1              1
2         1 moderate or severe terbinafine  0.8571429     2              1
3         1 moderate or severe terbinafine  3.5357140     3              1
4         1       none or mild terbinafine  4.5357140     4              0
5         1       none or mild terbinafine  7.5357140     5              0
6         1       none or mild terbinafine 10.0357100     6              0
```

```r
# ?toenail
```

a) Maximal Model Specification for Predicting the Sensitivity Outcome with Fixed and Random Effects

To create a maximal model specification for predicting the binary sensitivity outcome (`outcome_binary`) in the toenail data, we'll need to account for both fixed and random effects. Since this is a repeated measures dataset (patients have multiple visits), a mixed-effects model is appropriate, where we include both patient-level random effects and treatment and time as fixed effects.

Maximal Model Specification

1. **Fixed Effects:**
    - **Treatment**: The type of treatment (`terbinafine`, `itraconazole`) can influence the outcome, so it will be a fixed effect.
    - **Time**: Time is a continuous predictor, representing the progression of treatment. We may expect a nonlinear relationship, so this might be modeled using natural splines or polynomials.
    - **Visit**: The visit number could also be included as a fixed effect, especially if it interacts with treatment or time.
2. **Random Effects:**
    - **Random Intercept for Patient**: Each patient could have a different baseline level of the outcome, so we include a random intercept for `patientID`.
    - **Random Slope for Time within Patient**: It's likely that the effect of time (or treatment progression) varies between patients. Hence, we may model a random slope for time within patients.

Moreover, we could include different interactions:

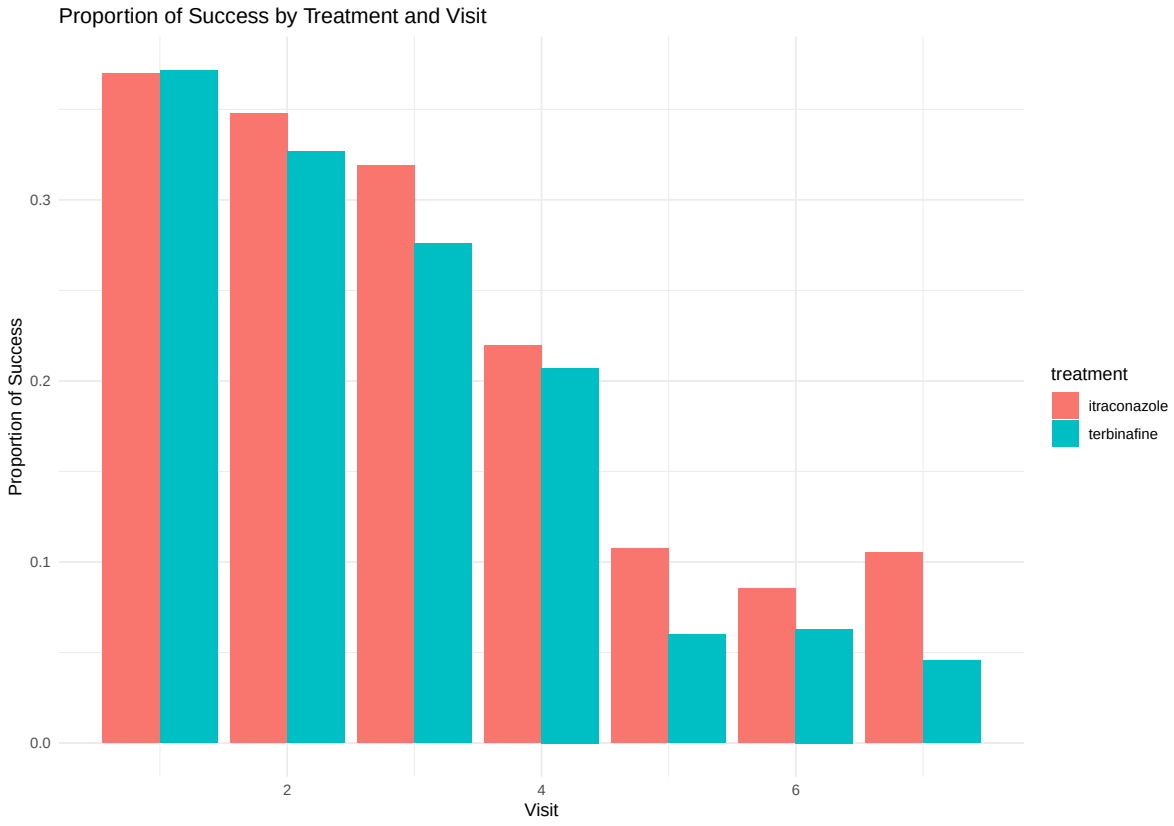
- **Treatment x Time Interaction**: The effect of time may differ by treatment, so an interaction term between treatment and time is included.
- **Treatment x Visit Interaction**: The effect of treatment may vary across different visits.

Given the binary nature of the response variable is completely sensible to use a logistic regression mixed model from the binomial family.

b and c)

Initially, I will not employ the maximal model. Instead, I plan to visualize the data to gain insights into its underlying structure. This exploratory step will inform the development of the final model by helping to identify patterns and relationships within the data.

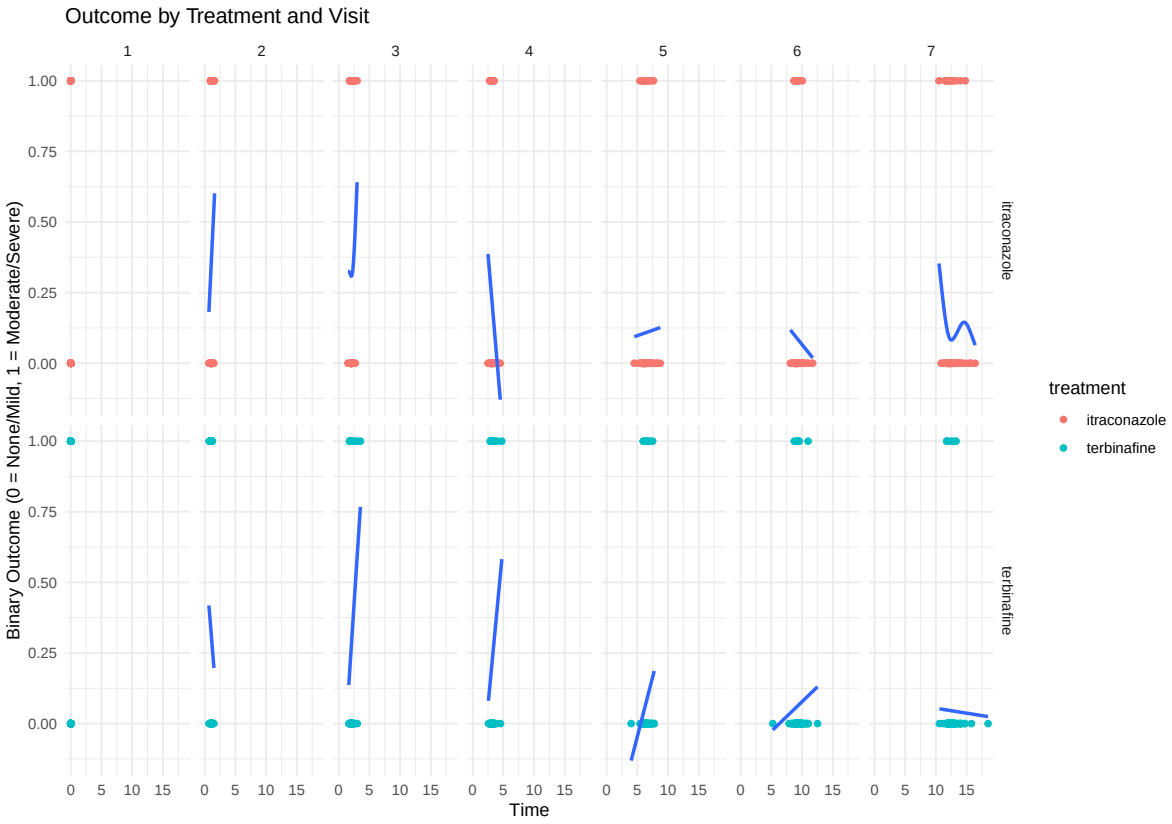
```r
mydata %>%
  group_by(treatment, visit) %>%
  summarise(success_rate = mean(outcome_binary), .groups = "drop") %>%  # Ungroup the data
  ggplot(aes(x = visit, y = success_rate, fill = treatment)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Proportion of Success by Treatment and Visit",
       x = "Visit",
       y = "Proportion of Success") +
  theme_minimal()
```



Success by Treatment and Visit (Bar Chart): This plot compares the proportion of successful outcomes between two treatments, itraconazole and terbinafine, over a series of visits. At visit 2, itraconazole demonstrates a marginally higher proportion of success compared to terbinafine,

with the gap narrowing by visit 4. By visits 6 and beyond, both treatments show a decline in success rates, with little distinction between the two treatments. This trend suggests that the efficacy of both treatments may diminish over time, potentially indicating a need for further investigation into long-term effectiveness or other influencing factors.

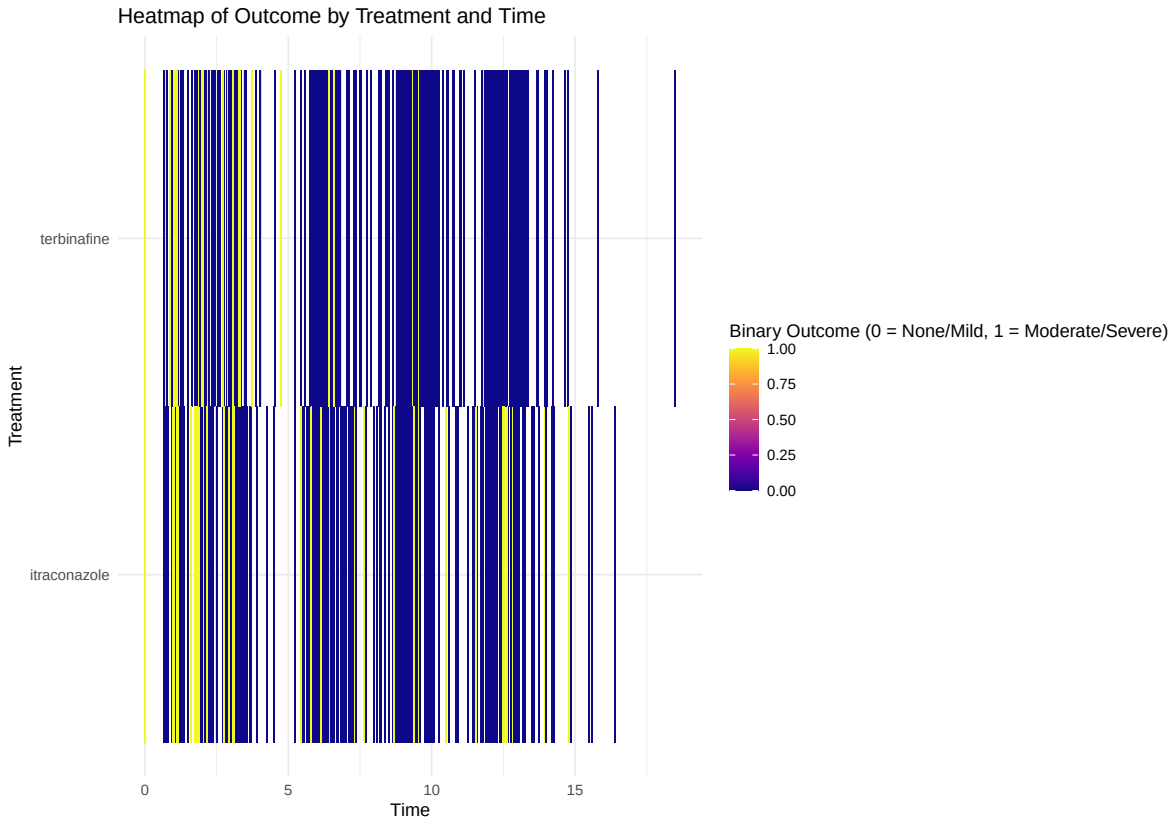
```
# Facet grid to show outcome_binary by treatment and time
ggplot(mydata, aes(x = time, y = outcome_binary)) +
  geom_point(aes(color = treatment)) +
geom_smooth(method = "gam", formula = y ~ s(x), aes(group = treatment), se = FALSE)+
  facet_grid(treatment ~ visit) + # Facet by treatment and visit
  labs(title = "Outcome by Treatment and Visit",
       x = "Time",
       y = "Binary Outcome (0 = None/Mild, 1 = Moderate/Severe)") +
  theme_minimal()
```



Outcome by Treatment and Visit (Scatter Plot Matrix): This figure provides a detailed temporal view of binary outcomes (0 = None/Mild, 1 = Moderate/Severe) across time points and visits for both itraconazole and terbinafine. The data reveals a higher prevalence of moderate/severe outcomes (outcome = 1) in the itraconazole group, particularly at earlier visits, with variability

decreasing over time. In contrast, terbinafine shows a more consistent distribution of mild outcomes (outcome = 0). The temporal clustering indicates differing response patterns to the two treatments, with itraconazole exhibiting greater variability in efficacy during earlier phases of treatment.

```r
# Heatmap of outcome_binary by time and treatment
ggplot(mydata, aes(x = time, y = treatment, fill = outcome_binary)) +
  geom_tile() +
  scale_fill_viridis(option = "C") +
  labs(title = "Heatmap of Outcome by Treatment and Time",
       x = "Time",
       y = "Treatment",
       fill = "Binary Outcome (0 = None/Mild, 1 = Moderate/Severe)") +
  theme_minimal()
```



Heatmap of Outcome by Treatment and Time: This heatmap visualizes the distribution of binary outcomes (0 = None/Mild, 1 = Moderate/Severe) across time for both treatments. Terbinafine predominantly shows outcomes concentrated in the lower range (None/Mild), indicating better overall efficacy. Itraconazole exhibits more frequent occurrences of moderate/severe outcomes,

represented by warmer colors (yellow). The temporal and treatment-specific clustering suggests that terbinafine consistently maintains better outcomes over time, while itraconazole shows more fluctuation and less favorable results in terms of severity.

The analysis reveals clear correlations between treatment type, time, and outcome severity. Terbinafine consistently shows better efficacy, with a higher proportion of mild outcomes (binary outcome = 0) over time, as reflected inthe plots. In contrast, itraconazole exhibits a higher frequency of moderate to severe outcomes (binary outcome = 1), particularly at earlier visits, indicating less consistent performance. The bar chart supports this trend, showing a slight early advantage for itraconazole in success rates at visit 2, but this diminishes over time, aligning with terbinafine's more stable efficacy. These findings indicate a strong negative correlation between time and treatment efficacy for both drugs, but the decline is more pronounced for itraconazole. Overall, terbinafine demonstrates a more robust and sustained correlation with favorable outcomes, suggesting it is the superior treatment option.

After conducting an extensive graphical analysis, a logistic regression mixed model was chosen for the analysis. The model will use a logit link function to capture the binary nature of the response variable. The fixed effects in the model will include **time**, **treatment**, and their interaction (**time** $\times$ **treatment**), allowing for the assessment of how the treatment effect evolves over time. Additionally, a random intercept will be included to account for variability among individual patients, identified by **patientID**.

d) Model Fit and Diagnostics

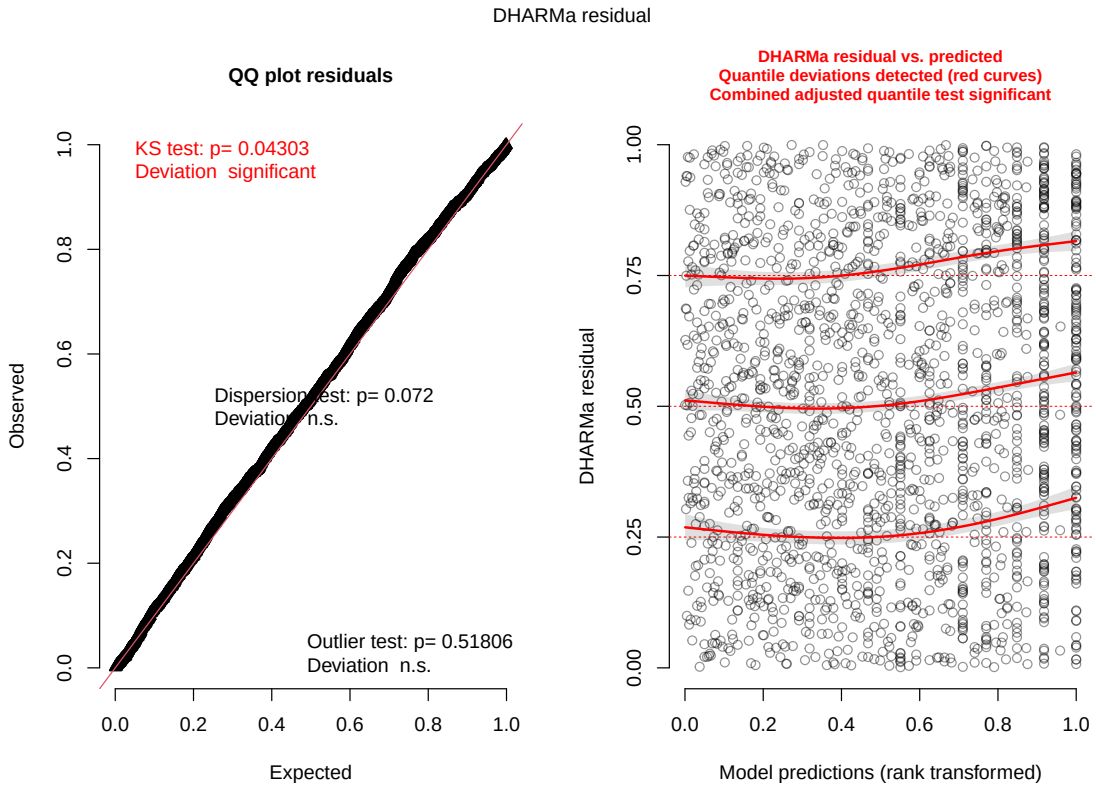
```r
# Model fit
model <- glmer(outcome_binary ~ time*treatment + (1 | patientID),
               data = mydata,
               family = binomial(link = "logit"))
summary(model)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: outcome_binary ~ time * treatment + (1 | patientID)
   Data: mydata

      AIC       BIC    logLik -2*log(L)  df.resid
   1265.6    1293.4    -627.8    1255.6      1903

Scaled residuals:
   Min     1Q Median     3Q    Max
```

```
-3.290 -0.149 -0.071 -0.006 47.215

Random effects:
 Groups    Name        Variance Std.Dev.
 patientID (Intercept) 20.76    4.557
Number of obs: 1908, groups:  patientID, 294

Fixed effects:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                -2.50986    0.76394  -3.285  0.00102 **
time                       -0.39973    0.04679  -8.543  < 2e-16 ***
treatmentterbinafine       -0.30483    0.68708  -0.444  0.65729
time:treatmentterbinafine -0.13714    0.06949  -1.973  0.04846 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) time   trtmnt
time         0.130
trtmnttrbnf -0.374  0.220
tm:trtmnttr  0.181 -0.541 -0.265
```

```r
#isSingular(model)
```

```r
# Check for singularity
# vc <- VarCorr(model); eigen(vc)
# performance::check_model(model)
# Create a DHARMa residuals object
residuals_dharma <- simulateResiduals(model)
# Plot the residuals
plot(residuals_dharma)
```



The DHARMa residual diagnostics provide a comprehensive assessment of model fit by comparing observed and expected residual distributions and evaluating potential patterns in the residuals. The QQ plot (left panel) reveals a slight deviation from the expected distribution, supported by a significant Kolmogorov-Smirnov (KS) test ($p = 0.04303$). However, given the asymptotic nature of these results and the visual inspection of the QQ plot, the deviations appear to be within acceptable limits.

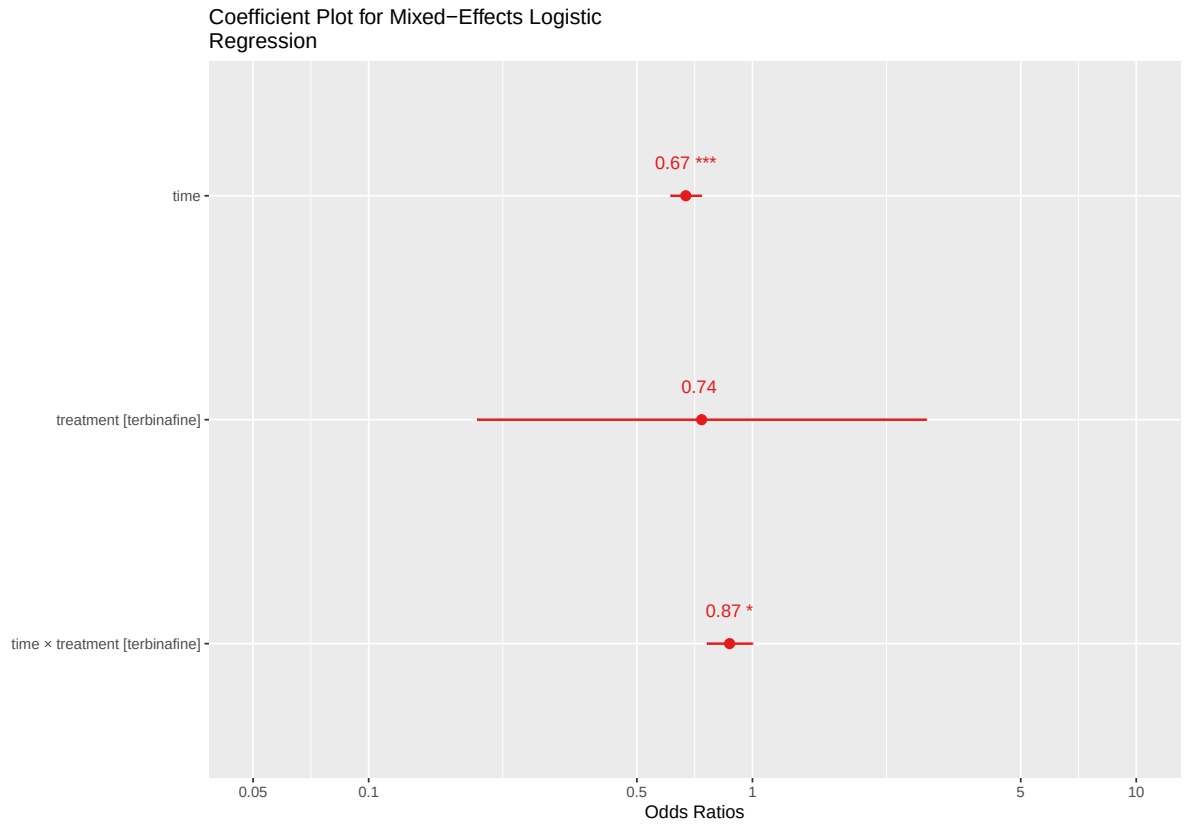
The dispersion and outlier tests indicate no significant departures from the expected distribution, with p-values of 0.072 and 0.51806, respectively, suggesting no evidence of overdispersion or the presence of influential outliers. The residuals vs. predicted values plot (right panel) shows some quantile deviations, highlighted by the red curves, which may suggest a potential non-linear relationship. The combined adjusted quantile test is significant, indicating possible model misspecifications. However, no indications of heteroscedasticity are observed.

The conditional `R2` value of 0.884 reflects the variance explained by both fixed and random effects, while the marginal `R2` value of 0.154 highlights the contribution of fixed effects alone.

In summary, while minor issues related to non-linearity and residual deviations are present, they are deemed acceptable for the purposes of this analysis.

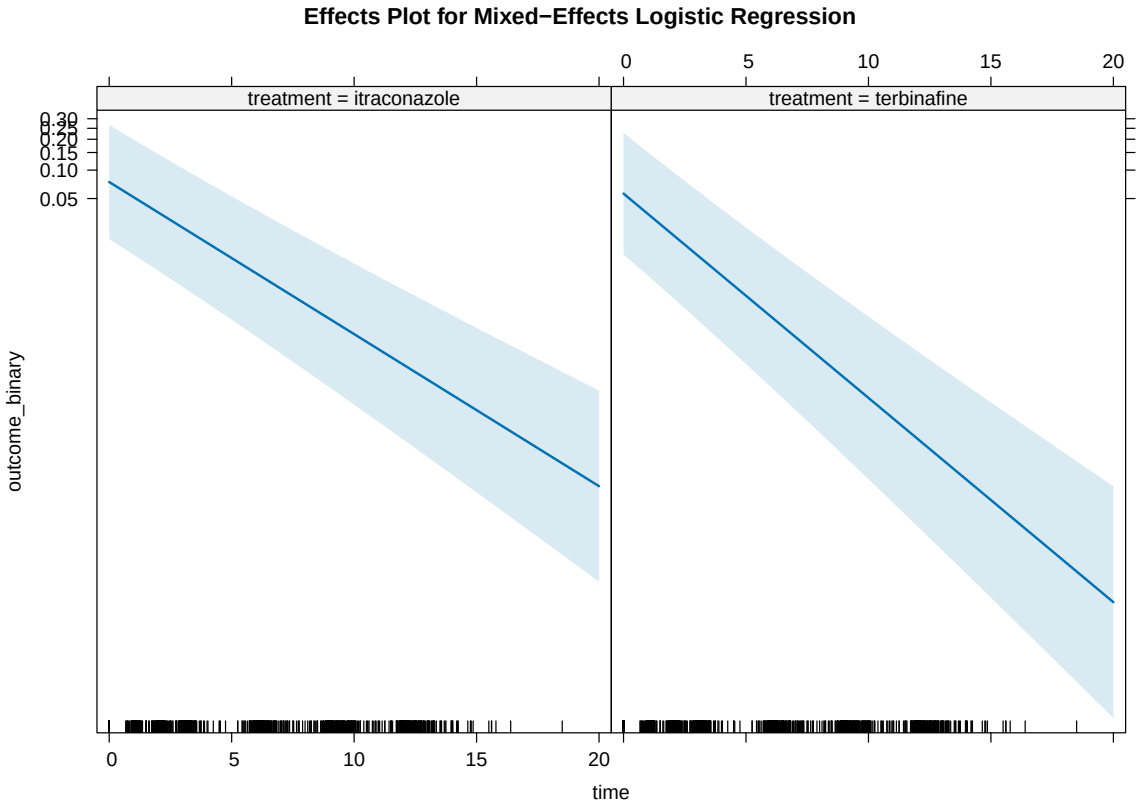
e) Visualizing the Results

```
sjPlot::plot_model(model, type = "est", show.values = TRUE, show.p = TRUE,
                   title = "Coefficient Plot for Mixed-Effects Logistic Regression")
```



The coefficient plot presents the odds ratios (ORs) from a mixed-effects logistic regression model assessing the effect of time, treatment (terbinafine), and their interaction on a binary outcome. The odds ratio for time is 0.67 (95% CI excludes 1, $p < 0.001$), indicating a significant decrease in the odds of the outcome over time. The odds ratio for treatment (terbinafine) is 0.74, with a wide confidence interval crossing 1, suggesting no statistically significant effect of treatment alone. The interaction between time and treatment (OR = 0.87, $p < 0.05$) implies that the effect of time on the outcome differs significantly depending on the treatment, with the decrease in odds over time being slightly attenuated in the terbinafine group.

```
# Generate effects for the interaction term
eff <- allEffects(model)
plot(eff, main = "Effects Plot for Mixed-Effects Logistic Regression")
```

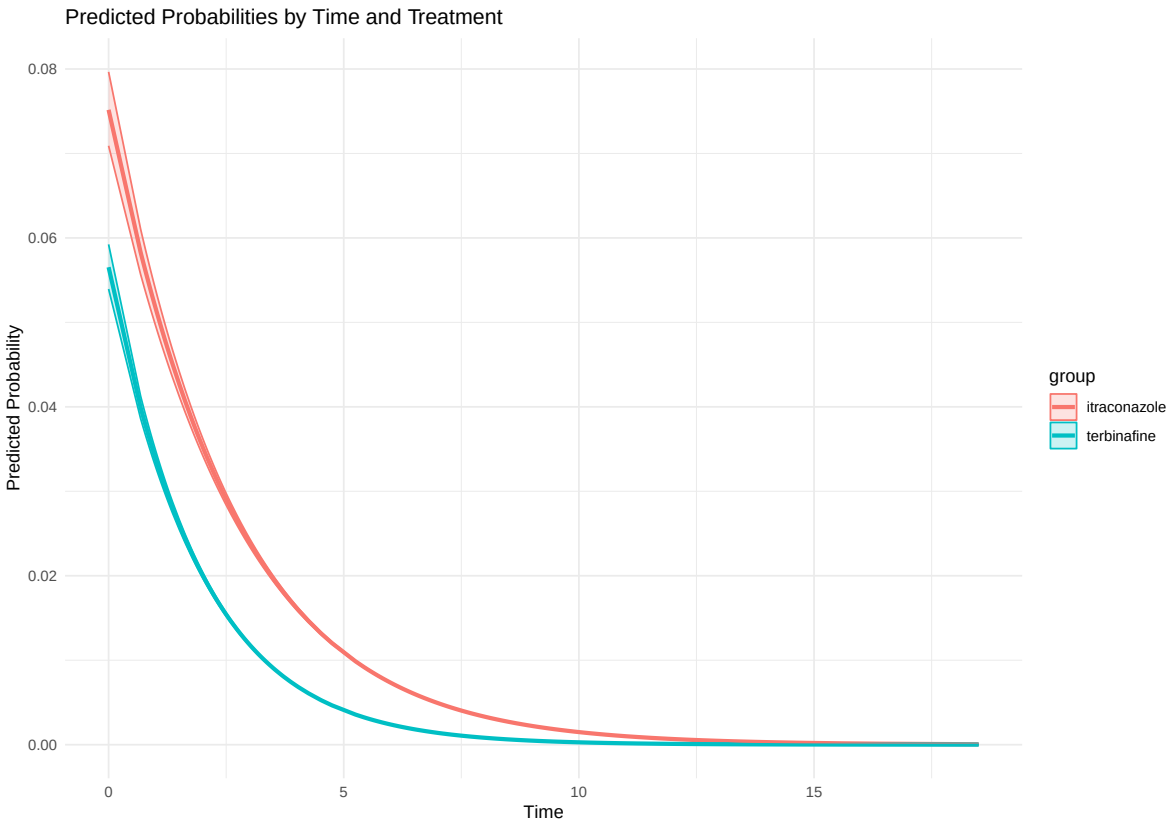


The effects plot depicts the predicted probabilities of the binary outcome over time for two treatment groups (itraconazole and terbinafine), derived from the mixed-effects logistic regression model. For both treatments, the probability of the outcome decreases over time, as indicated by the downward slope of the lines. However, the rate of decrease appears slightly less pronounced for the terbinafine group compared to itraconazole, consistent with the significant time × treatment interaction observed in the coefficient plot. The shaded regions represent 95% confidence intervals, highlighting the model's uncertainty, particularly at later time points where data sparsity is evident in the rug plot of observations along the x-axis.

Similar is the interpretation for the following diagram as well.

```
# Get predicted probabilities
pred <- ggpredict(model, terms = c("time [all]", "treatment"))

# Plot the interaction effects
ggplot(pred, aes(x = x, y = predicted, color = group)) +
  geom_line(size = 1.2) +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high, fill = group), alpha = 0.2) +
  labs(title = "Predicted Probabilities by Time and Treatment",
```

```
       x = "Time",
       y = "Predicted Probability") +
  theme_minimal()
```



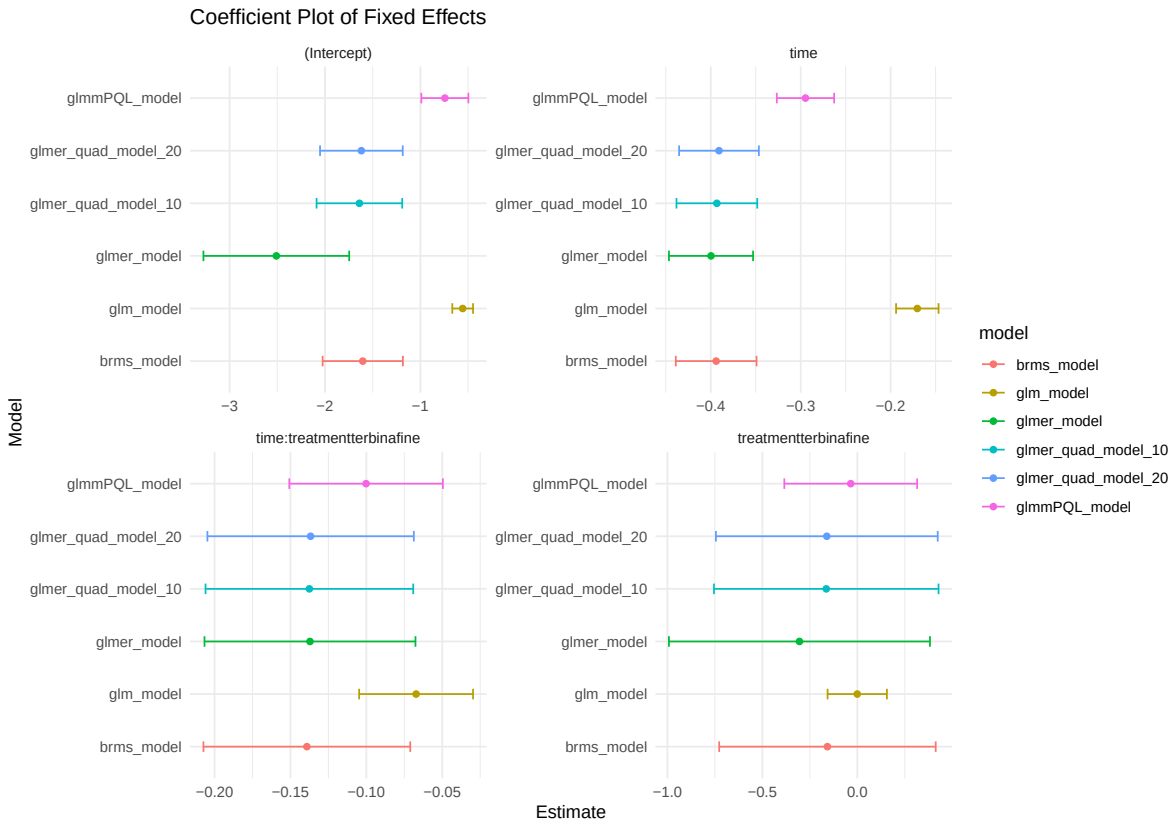
f) Fixed Effects Comparison from Different packages

```
models_list <- list(
  glm_model = glm_model,
  glmmPQL_model = glmmPQL_model,
  glmer_model = glmer_model,
  glmer_quad_model_10 = glmer_quad_model_10,
  glmer_quad_model_20 = glmer_quad_model_20,
  brms_model = model_brms
)

# Extract and arrange fixed effect coefficients for comparison
```

```r
fixed_effects_comparison <- purrr::map_dfr(models_list, ~tidy(., effects = "fixed", conf.int=
  dplyr::arrange(term)

# View the comparison
head(fixed_effects_comparison)
```

```
# A tibble: 6 x 11
  model    term  estimate std.error statistic  p.value conf.low conf.high effect
  <chr>    <chr>    <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl> <chr>
1 glm_mod~ (Int~   -0.557     0.109     -5.11  3.25e-7   -0.772    -0.344 <NA>
2 glmmPQL~ (Int~   -0.743     0.247     -3.01  2.62e-3   -1.23     -0.260 fixed
3 glmer_m~ (Int~   -2.51      0.764     -3.29  1.02e-3   -4.01     -1.01  fixed
4 glmer_q~ (Int~   -1.64      0.449     -3.65  2.60e-4   -2.52     -0.760 fixed
5 glmer_q~ (Int~   -1.62      0.433     -3.73  1.88e-4   -2.47     -0.769 fixed
6 brms_mo~ (Int~   -1.60      0.421     NA    NA         -2.48     -0.806 fixed
# i 2 more variables: df <dbl>, component <chr>
```

```r
# Plot fixed effects using ggplot2
ggplot(fixed_effects_comparison, aes(x = estimate, y = model, color = model)) +
  geom_point() +
  geom_errorbarh(aes(xmin = estimate - std.error, xmax = estimate + std.error), height = 0.2
  facet_wrap(~term, scales = "free") +
  labs(title = "Coefficient Plot of Fixed Effects",
       x = "Estimate",
       y = "Model") +
  theme_minimal()
```



The fixed-effect estimates across the six modeling approaches—`glm`, `glmmPQL`, `glmer`, `glmer_quad_10`, `glmer_quad_20`, and `brms`—show varying degrees of agreement. For the intercept, the estimates range from -0.56 (`glm_model`) to -2.51 (`glmer_model`), indicating differences in how each approach estimates baseline probabilities. The `brms` and `glmer_quad` models yield estimates closer to each other (-1.59 to -1.64), while `glm` and `glmmPQL` models differ slightly or moderately from these. For the effect of `time`, most models produce very similar estimates around -0.39, except for `glm_model`, which shows a lower magnitude of -0.17, suggesting a methodological distinction. The interaction term `time:treatmentterbinafine` is consistent across the mixed-effects and Bayesian models, with estimates clustering around -0.137, while the `glm` model produces a smaller effect (-0.067). For the treatment-only effect, variability is more pronounced, with estimates ranging from near-zero (`glm_model`) to -0.30 (`glmer_model`), though this term appears to lack statistical significance across models. Overall, the estimates for `time` and its interaction are very similar across models, while the intercept and treatment-only effects show more variation.

```
credible_intervals <- posterior_interval(model_brms, prob = 0.95)
# View the intervals
head(credible_intervals,4)
```

```
                                2.5%      97.5%
b_Intercept                -2.4826372 -0.8060465
b_time                     -0.4859797 -0.3095742
b_treatmentterbinafine     -1.2783963  0.9874379
b_time:treatmentterbinafine -0.2722888 -0.0088164
```

```r
# prior_summary(model_brms)
```

In this analysis, a Bayesian logistic regression model was fitted using the `brms` package to examine the effects of time, treatment with terbinafine, and their interaction on a binary outcome. Credible intervals (95%) for the fixed effects were derived using posterior samples. The intercept had a credible interval of [-2.51, -0.81], while the effect of time showed a credible interval of [-0.48, -0.31]. The treatment effect of terbinafine had a wider credible interval of [-1.28, 0.98], suggesting greater uncertainty around its estimate. Notably, the interaction term between time and terbinafine treatment had a credible interval of [-0.28, -0.005], indicating a likely small but significant interaction effect. Default priors were utilized for fixed effects, which were flat (uninformative) priors, allowing the posterior distributions to be driven predominantly by the observed data. For random effects (e.g., varying intercepts by patient ID), weakly informative Student's t-distribution priors were employed with three degrees of freedom, a mean of 0, and a scale parameter of 2.5, ensuring stability in estimating group-level variance while avoiding overly restrictive assumptions. These results highlight both the effectiveness of Bayesian modeling in quantifying uncertainty and the importance of considering prior choices in model interpretation.

3. Simulation Study

```r
simfun <- function(beta, theta, n_t, n_id) {
  # Create the dataset structure
  data <- expand.grid(
    id = factor(seq_len(n_id)),      # n_id levels for the grouping variable
    time = seq_len(n_t) - 1         # Integer time variable from 0 to n_t - 1
  )

  # Randomly assign individuals to two treatment groups
  data$ttt <- factor(sample(c("A", "B"), n_id, replace = TRUE))[as.numeric(data$id)]

  # Define the model formula
  formula <- ~ 1 + ttt * time + (1 | id)

  # Specify the parameters
```

```
  params <- list(
    beta = beta,                    # Fixed-effect parameters
    theta = theta                   # Random-effect parameters (on log scale)
  )

  # Simulate response data using glmmTMB::simulate_new
  sim_data <- simulate_new(
    object = formula,
    newdata = data,
    family = binomial(link = "logit"),         # Bernoulli response
    newparams = params
  )

  # Add the simulated response to the dataset
  data$response <- sim_data[[1]]

  return(data)
}
```

```
fitfun <- function(data, nAGQ) {
  # Fit the model based on the value of nAGQ
  if (nAGQ == -2) {
    # GLM (pooled model)
    model <- glm(response ~ 1 + ttt * time, data = data, family = binomial(link = "logit"))
    result <- broom.mixed::tidy(model, effects = "fixed", conf.int = TRUE)
    } else if (nAGQ == -1) {
    model <- MASS::glmmPQL(response ~ 1 + ttt * time,
                    random = ~ 1 | id,
                    family = binomial(link = "logit"),
                    data = data,
                    verbose = FALSE)
    result <- broom.mixed::tidy(model, effects = "fixed", conf.int = TRUE)
  } else if (nAGQ >= 1) {
    model <- glmer(response ~ 1 + ttt * time + (1 | id), data = data, family = binomial(link
    result <- broom.mixed::tidy(model, effects = "fixed", conf.int = TRUE)
    } else {
      stop("Invalid value of nAGQ. Use -2, -1, or an integer >= 1.")
      }

    results <- result[, c("term", "estimate", "conf.low", "conf.high")]
    return(results)
}
```

```r
# Function that fun sim_n number of simulations
run_sim <- function(sim_n, nAGQ, beta, theta, n_t, n_id){
  output <- replicate(sim_n, fitfun(simfun(beta = beta, theta = theta, n_t = n_t, n_id = n_id
  return(output)
}

# Intergrate the data into a data frame
sum.data <- function(data){
  result_df <- do.call(rbind, lapply(data, function(sim) {
  data.frame(
    intercept = sim$estimate[sim$term == "(Intercept)"],
    conf.low.intercept = sim$conf.low[sim$term == "(Intercept)"],
    conf.high.intercept = sim$conf.high[sim$term == "(Intercept)"],
    tttB = sim$estimate[sim$term == "tttB"],
    conf.low.tttB = sim$conf.low[sim$term == "tttB"],
    conf.high.tttB = sim$conf.high[sim$term == "tttB"],
    time = sim$estimate[sim$term == "time"],
    conf.low.time = sim$conf.low[sim$term == "time"],
    conf.high.time = sim$conf.high[sim$term == "time"],
    interaction = sim$estimate[sim$term == "tttB:time"],
    conf.low.interaction = sim$conf.low[sim$term == "tttB:time"],
    conf.high.interaction = sim$conf.high[sim$term == "tttB:time"]
  )}))
  return(result_df)
}

metrics_b2 <- function(true_values, data) {
  # Create an empty list to store results for each beta
  results <- list()

  # For b_2
  bias <- mean(data[,4]) - true_values[2]
  var <- var(data[,4])
  scaled_rmse <- sqrt(mean((data[,4] / true_values[2] - 1)^2))
  coverage <- mean(true_values[2] >= data[,5] & true_values[2] <= data[,6])

  # Store the results in the list
  results$bias <- bias
  results$variance <- var
  results$scaled_rmse <- scaled_rmse
  results$coverage <- coverage

```

```r
  # Convert the results list to a data frame
  return(as.data.frame(results))
}

metrics_b4 <- function(true_values, data) {
  # Create an empty list to store results for each beta
  results <- list()

  # For b_4
  bias <- mean(data[,10]) - true_values[4]
  var <- var(data[,10])
  scaled_rmse <- sqrt(mean((data[,10] / true_values[4] - 1)^2))
  coverage <- mean(true_values[4] >= data[,11] & true_values[4] <= data[,12])

  # Store the results in the list
  results$bias <- bias
  results$variance <- var
  results$scaled_rmse <- scaled_rmse
  results$coverage <- coverage

  # Convert the results list to a data frame
  return(as.data.frame(results))
}
```

```r
## options(timeout = 600)
## Simulation Using
set.seed(123)
beta <- c(-0.6, 0, -0.2, -0.05)
theta <- log(0.2)
n_id <- 300
n_sim <- 100


############### 100 Simulations for n_t = 5 ########################################
n_t <- 5
# Initialize empty lists to store the metrics for b4
b2_metrics <- list()

# Simulations for nAGQ = -2 (glm model)
nAGQ <- -2
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_glm_5 <- sum.data(sim_output)
```

```r


# Simulations for nAGQ = -1 (PQL model)
nAGQ <- -1
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_PQL_5 <- sum.data(sim_output)


# Simulations for nAGQ = 1 (Laplace model)
nAGQ <- 1
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_LAPLACE_5 <- sum.data(sim_output)


# Simulations for nAGQ = 5 (AGHQ model)
nAGQ <- 5
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_AGHQ_5 <- sum.data(sim_output)
```

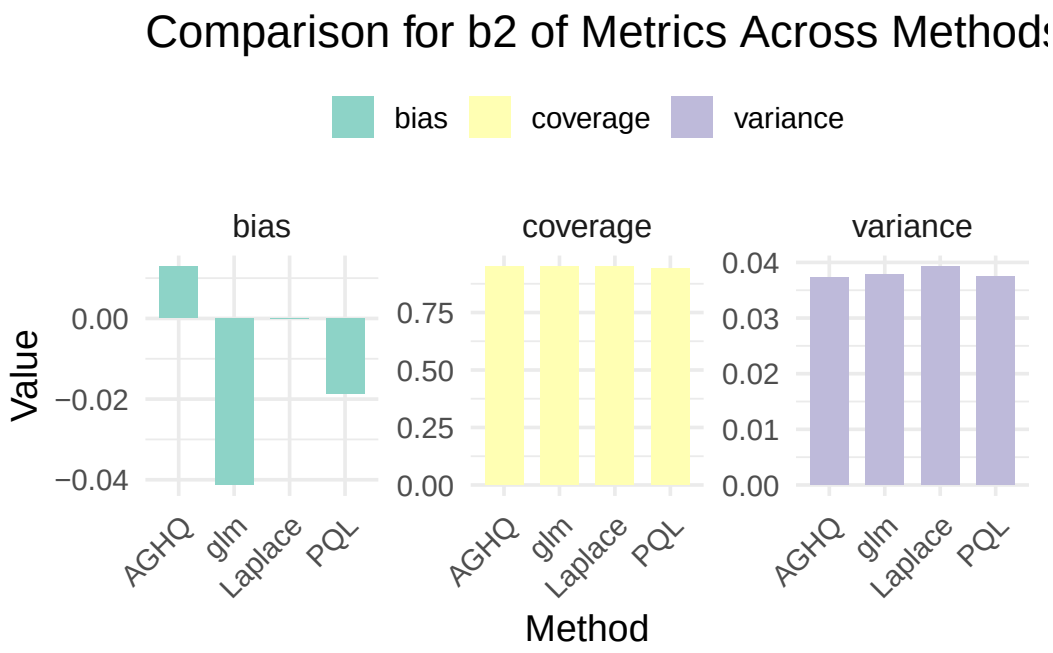
```r
############### 100 Simulations for n_t = 5 for b2 ###################################

# Initialize empty lists to store the metrics for b2
b2_metrics <- list()

# Simulations for nAGQ = -2 (glm model)
b2_metrics_glm <- metrics_b2(beta, output_glm_5)
# Store the results for glm model
b2_metrics$glm <- b2_metrics_glm

# Simulations for nAGQ = -1 (PQL model)
b2_metrics_PQL <- metrics_b2(beta, output_PQL_5)
# Store the results for PQL model
b2_metrics$PQL <- b2_metrics_PQL

# Simulations for nAGQ = 1 (Laplace model)
b2_metrics_LAPLACE <- metrics_b2(beta, output_LAPLACE_5)
# Store the results for Laplace model
b2_metrics$Laplace <- b2_metrics_LAPLACE

# Simulations for nAGQ = 5 (AGHQ model)
b2_metrics_AGHQ <- metrics_b2(beta, output_AGHQ_5)
# Store the results for AGHQ model
```

```r
b2_metrics$AGHQ <- b2_metrics_AGHQ

# Convert the lists of metrics to data frames
b2_metrics_df <- do.call(rbind, b2_metrics)

# Optionally, name the rows for easy reference
rownames(b2_metrics_df) <- c("glm", "PQL", "Laplace", "AGHQ")

# Transpose the data frame to switch rows and columns
b2_metrics_df_transposed <- t(b2_metrics_df)

# Return the transposed data frame
b2_metrics_df_transposed
```

```
                   glm         PQL       Laplace       AGHQ
bias        -0.04129691 -0.01875058 -0.0002026514 0.01287910
variance     0.03784005  0.03739414  0.0392600328 0.03737139
scaled_rmse         Inf         Inf           Inf        Inf
coverage     0.95000000  0.94000000  0.9500000000 0.95000000
```

```r
methods <- c("glm", "PQL", "Laplace", "AGHQ")
metrics <- data.frame(
  method = rep(methods, times = 3),
  bias = b2_metrics_df_transposed[1, ],   # assuming these rows are correct
  variance = b2_metrics_df_transposed[2, ],
  coverage = b2_metrics_df_transposed[4, ],
  row.names = NULL  # Ensure no row names are carried over
)


# Pivot the data longer for ggplot
metrics_long <- metrics %>%
  pivot_longer(cols = c(bias, variance, coverage), names_to = "metric", values_to = "value")

# Plotting
ggplot(metrics_long, aes(x = method, y = value, fill = metric)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +
  facet_wrap(~metric, scales = "free_y", nrow = 1) +
  scale_fill_brewer(palette = "Set3") +  # Set a nice color palette
  labs(
    title = "Comparison for b2 of Metrics Across Methods (n_t = 5)",
```

```
    x = "Method",
    y = "Value",
    fill = "Metric"
  ) +
  theme_minimal(base_size = 14) +  # Increase base size for better readability
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis text for clarity
    strip.text = element_text(size = 12),  # Adjust facet labels
    legend.position = "top",  # Place legend at the top
    legend.title = element_blank()  # Remove legend title
  )
```



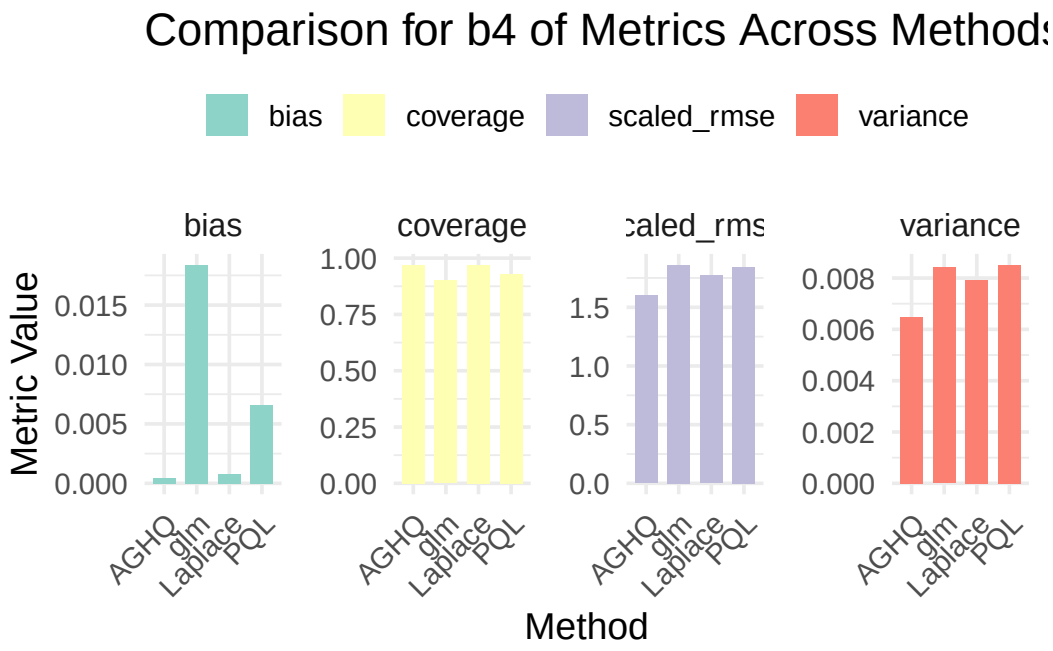
This plot compares the performance of four methods (AGHQ, glm, Laplace, PQL) for parameter b2 across three metrics: bias, coverage, and variance, with n_t = 5. The bias plot indicates that the glm method exhibits the largest negative bias, while AGHQ and PQL show minor positive bias. Coverage rates are consistently high across all methods, hovering near 1.0, demonstrating reliable interval performance. In terms of variance, all methods have comparable values, with AGHQ showing slightly higher variance compared to the others. These findings suggest that AGHQ and PQL provide relatively unbiased estimates while maintaining acceptable coverage and variance levels.

```r
############### 100 Simulations for n_t = 5 for b4 ################################
# Initialize empty lists to store the metrics for b4
b4_metrics <- list()

# Simulations for nAGQ = -2 (glm model)
b4_metrics_glm <- metrics_b4(beta, output_glm_5)
# Store the results for glm model
b4_metrics$glm <- b4_metrics_glm


# Simulations for nAGQ = -1 (PQL model)
b4_metrics_PQL <- metrics_b4(beta, output_PQL_5)
# Store the results for PQL model
b4_metrics$PQL <- b4_metrics_PQL


# Simulations for nAGQ = 1 (Laplace model)
b4_metrics_LAPLACE <- metrics_b4(beta, output_LAPLACE_5)
# Store the results for Laplace model
b4_metrics$Laplace <- b4_metrics_LAPLACE


# Simulations for nAGQ = 5 (AGHQ model)
b4_metrics_AGHQ <- metrics_b4(beta, output_AGHQ_5)
# Store the results for AGHQ model
b4_metrics$AGHQ <- b4_metrics_AGHQ

# Convert the lists of metrics to data frames
b4_metrics_df <- do.call(rbind, b4_metrics)

# Optionally, name the rows for easy reference
rownames(b4_metrics_df) <- c("glm", "PQL", "Laplace", "AGHQ")

# Transpose the data frame to switch rows and columns
b4_metrics_df_transposed <- t(b4_metrics_df)

# Return the transposed data frame
b4_metrics_df_transposed
```

```
                  glm         PQL      Laplace         AGHQ
bias         0.018385112 0.006606813 0.0007789964 0.0004380779
variance     0.008408592 0.008513147 0.0079322476 0.0064752050
```

```
scaled_rmse 1.861453077 1.840833064 1.7724031072 1.6013300489
coverage    0.900000000 0.930000000 0.9700000000 0.9700000000
```

```r
# Define methods
methods <- c("glm", "PQL", "Laplace", "AGHQ")

# Create the metrics data frame
metrics <- data.frame(
  metric = rep(c("bias", "variance", "scaled_rmse", "coverage"), each = 4),
  method = rep(methods, times = 4),
  value = c(b4_metrics_df_transposed[1,],
            b4_metrics_df_transposed[2,],
            b4_metrics_df_transposed[3,],
            b4_metrics_df_transposed[4,])
)

# Plotting
ggplot(metrics, aes(x = method, y = value, fill = metric)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +  # Adjust bar width for bett
  facet_wrap(~metric, scales = "free_y", nrow = 1) +  # Arrange metrics in a single row
  scale_fill_brewer(palette = "Set3") +  # Apply a custom color palette
  labs(
    title = "Comparison for b4 of Metrics Across Methods (n_t = 5)",  # Main plot title
    x = "Method",  # X-axis label
    y = "Metric Value",  # Y-axis label
    fill = "Metric"  # Legend label for fill
  ) +
  theme_minimal(base_size = 14) +  # Use a clean theme and set base font size
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels for readabilit
    strip.text = element_text(size = 12),  # Increase size of facet labels
    legend.position = "top",  # Move the legend to the top
    legend.title = element_blank(),  # Remove legend title for simplicity
    panel.spacing = unit(1, "lines")  # Increase space between facets
  )
```



This plot illustrates the metrics bias, coverage, scaled RMSE, and variance for parameter b4 across four methods (AGHQ, glm, Laplace, PQL) with n_t = 5. Bias results indicate AGHQ is nearly unbiased, while PQL has a notable positive bias. Coverage is consistently high across all methods, remaining close to 1.0. Scaled RMSE values are slightly higher for Laplace and PQL compared to AGHQ and glm, reflecting differences in accuracy. Variance is lowest for AGHQ and highest for PQL, highlighting AGHQ's balance between precision and accuracy. Overall, AGHQ demonstrates the most favorable performance with low bias, high coverage, and moderate variance.

```r
## options(timeout = 600)
## Simulation Using
set.seed(123)
beta <- c(-0.6, 0, -0.2, -0.05)
theta <- log(0.2)
n_id <- 300
n_sim <- 100


############### 100 Simulations for n_t = 10 ######################################
n_t <- 10
# Initialize empty lists to store the metrics for b4
b2_metrics <- list()

```

```r
# Simulations for nAGQ = -2 (glm model)
nAGQ <- -2
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_glm_10 <- sum.data(sim_output)


# Simulations for nAGQ = -1 (PQL model)
nAGQ <- -1
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_PQL_10 <- sum.data(sim_output)


# Simulations for nAGQ = 1 (Laplace model)
nAGQ <- 1
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_LAPLACE_10 <- sum.data(sim_output)


# Simulations for nAGQ = 5 (AGHQ model)
nAGQ <- 5
sim_output <- run_sim(n_sim, nAGQ, beta, theta, n_t, n_id)
output_AGHQ_10 <- sum.data(sim_output)
```

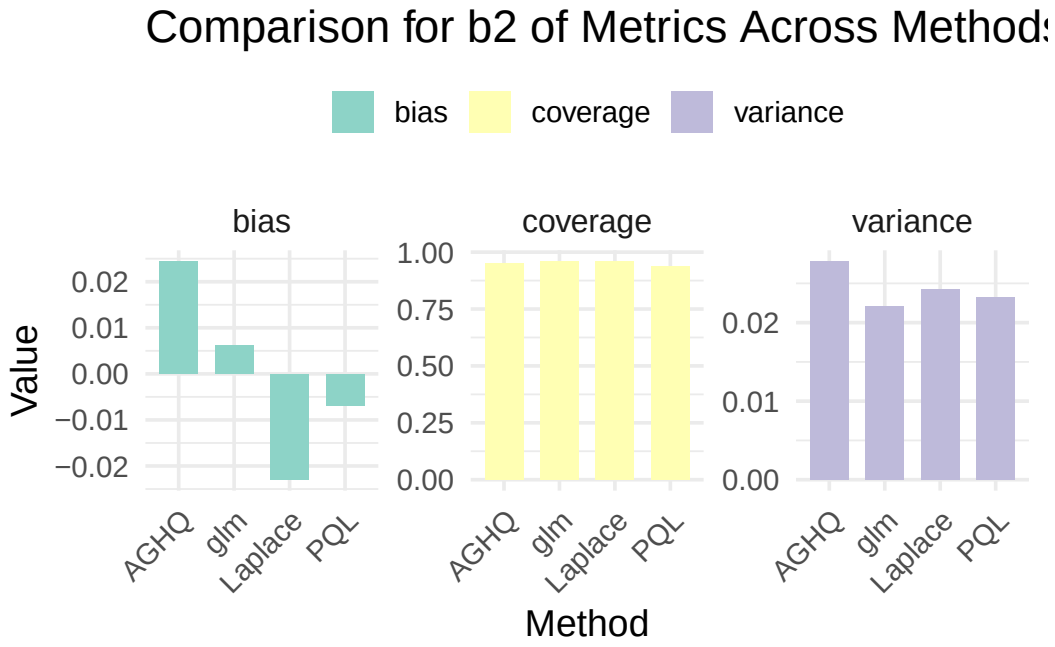
```r
############### 100 Simulations for n_t = 10 for b2 ################################

# Initialize empty lists to store the metrics for b2
b2_metrics <- list()

# Simulations for nAGQ = -2 (glm model)
b2_metrics_glm <- metrics_b2(beta, output_glm_10)
# Store the results for glm model
b2_metrics$glm <- b2_metrics_glm


# Simulations for nAGQ = -1 (PQL model)
b2_metrics_PQL <- metrics_b2(beta, output_PQL_10)
# Store the results for PQL model
b2_metrics$PQL <- b2_metrics_PQL


# Simulations for nAGQ = 1 (Laplace model)
b2_metrics_LAPLACE <- metrics_b2(beta, output_LAPLACE_10)
```

```r
# Store the results for Laplace model
b2_metrics$Laplace <- b2_metrics_LAPLACE


# Simulations for nAGQ = 5 (AGHQ model)
b2_metrics_AGHQ <- metrics_b2(beta, output_AGHQ_10)
# Store the results for AGHQ model
b2_metrics$AGHQ <- b2_metrics_AGHQ


# Convert the lists of metrics to data frames
b2_metrics_df <- do.call(rbind, b2_metrics)

# Optionally, name the rows for easy reference
rownames(b2_metrics_df) <- c("glm", "PQL", "Laplace", "AGHQ")

# Transpose the data frame to switch rows and columns
b2_metrics_df_transposed <- t(b2_metrics_df)

# Return the transposed data frame
b2_metrics_df_transposed
```

```
                  glm          PQL     Laplace       AGHQ
bias        0.006258034 -0.006838937 -0.02297344 0.02445127
variance    0.022098072  0.023205626  0.02426724 0.02782000
scaled_rmse         Inf          Inf         Inf        Inf
coverage    0.960000000  0.940000000  0.96000000 0.95000000
```

```r
methods <- c("glm", "PQL", "Laplace", "AGHQ")
metrics <- data.frame(
  method = rep(methods, times = 3),
  bias = b2_metrics_df_transposed[1, ],   # assuming these rows are correct
  variance = b2_metrics_df_transposed[2, ],
  coverage = b2_metrics_df_transposed[4, ],
  row.names = NULL  # Ensure no row names are carried over
)


# Pivot the data longer for ggplot
metrics_long <- metrics %>%
  pivot_longer(cols = c(bias, variance, coverage), names_to = "metric", values_to = "value")
```

```r

# Plotting
ggplot(metrics_long, aes(x = method, y = value, fill = metric)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +
  facet_wrap(~metric, scales = "free_y", nrow = 1) +
  scale_fill_brewer(palette = "Set3") +  # Set a nice color palette
  labs(
    title = "Comparison for b2 of Metrics Across Methods (n_t = 10)",
    x = "Method",
    y = "Value",
    fill = "Metric"
  ) +
  theme_minimal(base_size = 14) +  # Increase base size for better readability
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis text for clarity
    strip.text = element_text(size = 12),  # Adjust facet labels
    legend.position = "top",  # Place legend at the top
    legend.title = element_blank()  # Remove legend title
  )
```



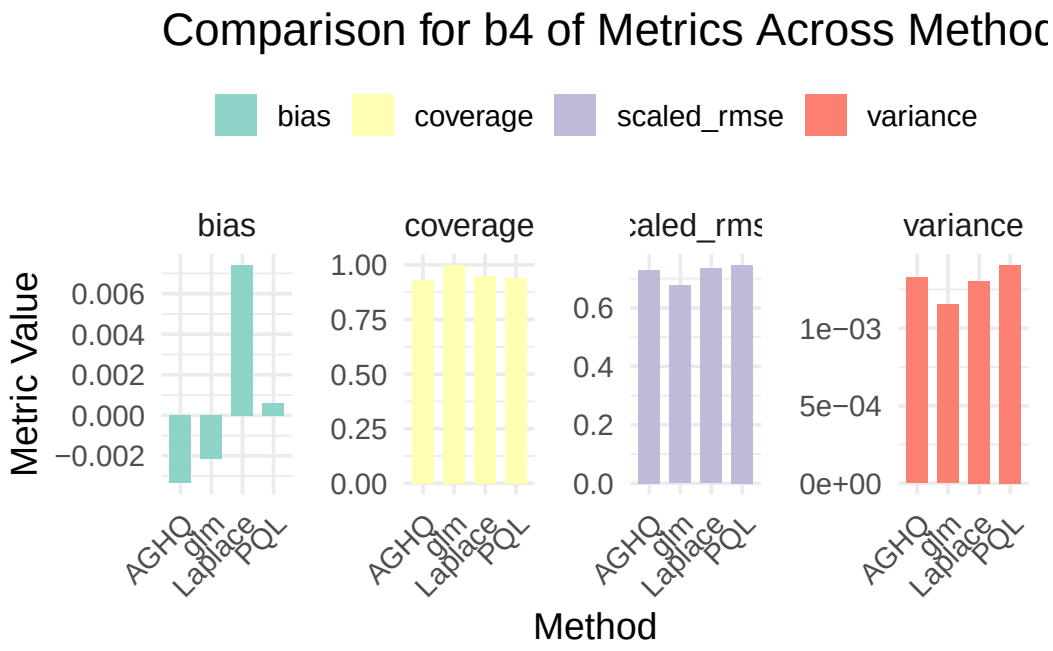
This plot evaluates the performance of four methods (AGHQ, glm, Laplace, PQL) for parameter b2 across bias, coverage, and variance metrics, with n_t = 10. AGHQ shows a slight positive bias, while glm exhibits minimal bias, and Laplace has a small negative bias. Coverage rates

remain consistently high across all methods, indicating robust interval estimation. Variance is highest for AGHQ and lowest for PQL, with glm and Laplace falling in between. These results suggest that glm offers a favorable trade-off between low bias and moderate variance while maintaining excellent coverage.

```
############### 100 Simulations for n_t = 5 for b4 ################################
# Initialize empty lists to store the metrics for b4
b4_metrics <- list()

# Simulations for nAGQ = -2 (glm model)
b4_metrics_glm <- metrics_b4(beta, output_glm_10)
# Store the results for glm model
b4_metrics$glm <- b4_metrics_glm


# Simulations for nAGQ = -1 (PQL model)
b4_metrics_PQL <- metrics_b4(beta, output_PQL_10)
# Store the results for PQL model
b4_metrics$PQL <- b4_metrics_PQL


# Simulations for nAGQ = 1 (Laplace model)
b4_metrics_LAPLACE <- metrics_b4(beta, output_LAPLACE_10)
# Store the results for Laplace model
b4_metrics$Laplace <- b4_metrics_LAPLACE


# Simulations for nAGQ = 5 (AGHQ model)
b4_metrics_AGHQ <- metrics_b4(beta, output_AGHQ_10)
# Store the results for AGHQ model
b4_metrics$AGHQ <- b4_metrics_AGHQ

# Convert the lists of metrics to data frames
b4_metrics_df <- do.call(rbind, b4_metrics)

# Optionally, name the rows for easy reference
rownames(b4_metrics_df) <- c("glm", "PQL", "Laplace", "AGHQ")

# Transpose the data frame to switch rows and columns
b4_metrics_df_transposed <- t(b4_metrics_df)

# Return the transposed data frame
b4_metrics_df_transposed
```

```
                    glm           PQL     Laplace          AGHQ
bias        -0.002159583 0.0006114208 0.007422869 -0.003354739
variance     0.001152805 0.0014086545 0.001306414  0.001328255
scaled_rmse  0.677034799 0.7469783954 0.734424599  0.728347903
coverage     1.000000000 0.9400000000 0.950000000  0.930000000
```

```r
# Define methods
methods <- c("glm", "PQL", "Laplace", "AGHQ")

# Create the metrics data frame
metrics <- data.frame(
  metric = rep(c("bias", "variance", "scaled_rmse", "coverage"), each = 4),
  method = rep(methods, times = 4),
  value = c(b4_metrics_df_transposed[1,],
            b4_metrics_df_transposed[2,],
            b4_metrics_df_transposed[3,],
            b4_metrics_df_transposed[4,])
)

# Plotting
ggplot(metrics, aes(x = method, y = value, fill = metric)) +
  geom_bar(stat = "identity", position = "dodge", width = 0.7) +  # Adjust bar width for bett
  facet_wrap(~metric, scales = "free_y", nrow = 1) +  # Arrange metrics in a single row
  scale_fill_brewer(palette = "Set3") +  # Apply a custom color palette
  labs(
    title = "Comparison for b4 of Metrics Across Methods (n_t = 10)",  # Main plot title
    x = "Method",  # X-axis label
    y = "Metric Value",  # Y-axis label
    fill = "Metric"  # Legend label for fill
  ) +
  theme_minimal(base_size = 14) +  # Use a clean theme and set base font size
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels for readabilit
    strip.text = element_text(size = 12),  # Increase size of facet labels
    legend.position = "top",  # Move the legend to the top
    legend.title = element_blank(),  # Remove legend title for simplicity
    panel.spacing = unit(1, "lines")  # Increase space between facets
  )
```



The graph displays a comprehensive comparison of four different statistical metrics (bias, coverage, scaled_rmse, and variance) across four methods (AGHQ, glm, Laplace, and PQL) for a parameter b4 with n_t = 10. Notably, the Laplace method shows the highest positive bias around 0.006, while AGHQ and glm demonstrate slight negative bias. Coverage rates are consistently high across all methods, maintaining values above 0.75. The scaled root mean square error (scaled_rmse) exhibits similar values across methods, ranging approximately between 0.4 and 0.7. Variance measurements are relatively small across all methods, with values falling in the range of 0.0005 to 0.0015, though PQL shows marginally higher variance compared to the other methods. This visualization effectively illustrates the relative performance and trade-offs between these different statistical approaches.

References

- Barr, Dale J., Roger Levy, Christoph Scheepers, and Harry J. Tily. 2013. "Random Effects Structure for Confirmatory Hypothesis Testing: Keep It Maximal." Journal of Memory and Language 68 (3): 255–78. https://doi.org/10.1016/j.jml.2012.11.001.
- Bates, Douglas, Reinhold Kliegl, Shravan Vasishth, and Harald Baayen. 2015. "Parsimonious Mixed Models." arXiv:1506.04967 [Stat], June. https://arxiv.org/abs/1506.04967.
- Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. "Fitting Linear MixedEffects Models Using Lme4." Journal of Statistical Software 67 (October): 1–48. https://doi. org/10.18637/jss.v067.i01.

- Biswas, Keya. 2015. "Performances of Different Estimation Methods for Generalized Linear Mixed Models." Master's thesis, McMaster University. https://macsphere.mcmaster.ca/ bitstream/11375/17272/2/M.Sc_Thesis_final_Keya_Biswas.pdf.
- Bolker, Benjamin M. 2015. "Generalized Linear Mixed Models." In Ecological Statistics: Contemporary Theory and Application, edited by Gordon A. Fox, Simoneta Negrete-Yankelevich, and Vinicio J. Sosa. Oxford University Press.
- Booth, James G., and James P. Hobert. 1999. "Maximizing Generalized Linear Mixed Model Likelihoods with an Automated Monte Carlo EM Algorithm." Journal of the Royal Statistical Society. Series B 61 (1): 265–85. https://doi.org/10.1111/1467-9868.00176.
- Breslow, N. E. 2004. "Whither PQL?" In Proceedings of the Second Seattle Symposium in Biostatistics: Analysis of Correlated Data, edited by Danyu Y. Lin and P. J. Heagerty, 1–22. Springer.
- Crawley, Michael J. 2002. Statistical Computing: An Introduction to Data Analysis Using S-PLUS. John Wiley & Sons.
- Gelman, Andrew. 2005. "Analysis of Variance: Why It Is More Important Than Ever." Annals of Statistics 33 (1): 1–53. https://doi.org/doi:10.1214/009053604000001048.
- Matuschek, Hannes, Reinhold Kliegl, Shravan Vasishth, Harald Baayen, and Douglas Bates.

  2017. "Balancing Type I Error and Power in Linear Mixed Models." Journal of Memory and Language 94 (June): 305–15. https://doi.org/10.1016/j.jml.2017.01.001.
- Murtaugh, Paul A. 2007. "Simplicity and Complexity in Ecological Data Analysis." Ecology 88 (1): 56–62. http://www.esajournals.org/doi/abs/10.1890/0012-9658%282007%2988%5B5 6%3ASACIED%5D2.0.CO%3B2.
- Pinheiro, José C., and Douglas M. Bates. 1996. "Unconstrained Parametrizations for VarianceCovariance Matrices." Statistics and Computing 6 (3): 289–96. https://doi.org/10.1007/BF

140873.
- Ponciano, José Miguel, Mark L. Taper, Brian Dennis, and Subhash R. Lele. 2009. "Hierarchical Models in Ecology: Confidence Intervals, Hypothesis Testing, and Model Selection Using Data Cloning." Ecology 90 (2): 356–62. http://www.jstor.org/stable/27650990.
- Stroup, Walter W. 2014. "Rethinking the Analysis of Non-Normal Data in Plant and Soil Science." Agronomy Journal 106: 1–17. https://doi.org/10.2134/agronj2013.0342.
- Sung, Yun Ju, and Charles J. Geyer. 2007. "Monte Carlo Likelihood Inference for Missing Data Models." The Annals of Statistics 35 (3): 990–1011. https://doi.org/10.1214/009053606000

1389.

- I read assignments for previous years in order to gain some great code ideas and identify previous mistakes inorder to avoid them.
- I used Chat Gpt extensively to create reports (I was writting my conclusions and after that, I was asking Chat Gpt to write it in a formal way as a report).
- I used chat-gpt to code some plots I didnt know and create a more professional output.