HW4

2024-11-29

Load the Packages

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
library(splines) # For spline-based trends
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: ’dplyr’

## The following objects are masked from ’package:stats’:
##
##     filter, lag

## The following objects are masked from ’package:base’:
##
##     intersect, setdiff, setequal, union
```

```
library(ggalt)
```

```
## Warning: package ’ggalt’ was built under R version 4.4.2

## Registered S3 methods overwritten by ’ggalt’:
##   method                  from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob  ggplot2
##   grobX.absoluteGrob      ggplot2
##   grobY.absoluteGrob      ggplot2
```

```
library(performance)
library(glmmTMB)
library(DHARMa)
```

```
## This is DHARMa 0.4.6. For overview type ’?DHARMa’. For recent changes, type news(package = ’DHARMa’)
```

```r
library(dotwhisker)
library(broom.mixed)
library(carData)
library(ggeffects)
library(effects)
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```r
library(HSAUR3)
```

```
## Warning: package ’HSAUR3’ was built under R version 4.4.2
```

```r
library(MASS)
```

```
##
## Attaching package: ’MASS’

## The following object is masked from ’package:dplyr’:
##
##     select
```

```r
library(MCMCglmm)
```

```
## Warning: package ’MCMCglmm’ was built under R version 4.4.2

## Loading required package: coda

## Loading required package: ape

##
## Attaching package: ’ape’

## The following object is masked from ’package:dplyr’:
##
##     where
```

```r
library(GLMMadaptive)
```

```
## Warning: package ’GLMMadaptive’ was built under R version 4.4.2

##
## Attaching package: ’GLMMadaptive’

## The following object is masked from ’package:MASS’:
##
##     negative.binomial

## The following object is masked from ’package:lme4’:
##
##     negative.binomial
```

Q1. Olympics Data

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

# Filter data for gold medals
gold_medals_data <- mydat[mydat$medal == "Gold", ]

# Scale gdp and pop
gold_medals_data$gdp_scaled <- scale(gold_medals_data$gdp)
gold_medals_data$pop_scaled <- scale(gold_medals_data$pop)
```

a. Describe the Maximal Model

- I chose "gold medals" (`n` when `medal` == `Gold`) as the response variable, `year`, `GDP` and `population` as fixed effect predictor variables, and `team (country)` as a random effect.
- The **maximal model** includes the **count of gold medals** (`n`) as the *response variable*, `year` as a *fixed effect* using natural splines, `gdp` and `population` as *fixed effects*, and `team` (country) and `year` as a *random effect*. An **interaction** between `gdp` and `population` is also included as a predictor.

b. country:year as Grouping Variable

- When does it make sense to include a random effects term with grouping variable `country:year`?

1. When there are enough observations for each `(country × year)` combination to estimate the associated variability reliably, it makes sense to include a random effects term with grouping variable `country:year`.
2. Including `country:year` can also make sense in a **hierarchical model** when `year` is nested within `country`. If each country can have **unique yearly effects** beyond the overall time trend, then modeling these as a random effect helps capture this nested dependency.

- When does it not make sense to include a random effects term with grouping variable `country:year`?

1. If there are very few observations for each `(country × year)` combination (e.g., one observation per level), estimating a random effect for each level may lead to **overfitting and numerical instability**.
2. In the model, the response follows a **Poisson distribution** (count data). The **Poisson distribution** assumes that the variance is equal to the mean, which complicates the estimation when you add many random effects levels (like `country:year`). When the response is count data, especially if the counts are mostly **low or zero**, adding a `country:year` random effect can lead to **convergence issues** or make it harder to reliably estimate the random effect variances.
3. If `year` is already included as a **fixed effect** (e.g., using splines or polynomial terms), adding `country:year` as a random effect could be redundant or lead to **multicollinearity issues**, where the model struggles to separate the effect of the fixed `year` trend from the `country:year` random intercept.

c. Initial Model to Fit

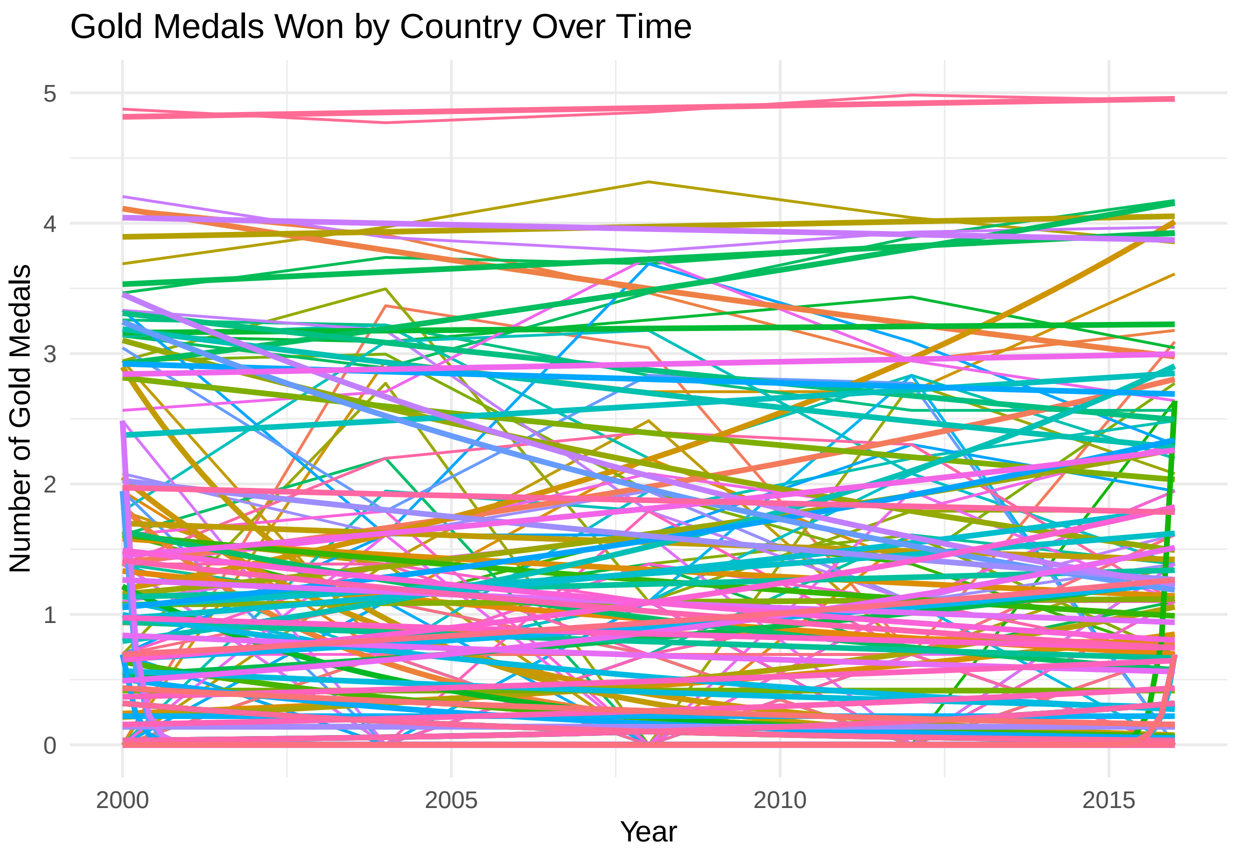
- Initially, I tried to fit the maximal model: `gold_model <- glmer(n ~ ns(year, df = 3) + gdp + pop + (1 | team) + (1 | year), data = gold_medals_data, family = poisson(link = "log"))`, where the count of gold medals won (`n`) is the response variable, `year`, `gdp`, `pop` are fixed-effect predictor variables, `team` are `year` are the random-effects grouping variables.

d. Create Exploratory Plots

Plotting Gold Medal Counts by Year and Country

```r
# Remove rows with missing values
gold_medals_data1 <- gold_medals_data %>%
  filter(!is.na(year), !is.na(n), !is.na(gdp), !is.na(pop))

# Create the plot with filtered data and set axis limits
ggplot(gold_medals_data1, aes(x = year, y = log(n+1), group = team, colour = team)) +
  geom_line() +  # Line plot to show trends over time for each country
  geom_smooth(method = "glm", method.args = list(family = quasipoisson), se = FALSE) +
  # Smoothed trend line
  theme_minimal() +
  theme(legend.position = "none") +  # Hide the legend to avoid clutter
  labs(
    title = "Gold Medals Won by Country Over Time",
    x = "Year",
    y = "Number of Gold Medals"
  ) +
  xlim(2000, 2016) + ylim(0, 5)
```

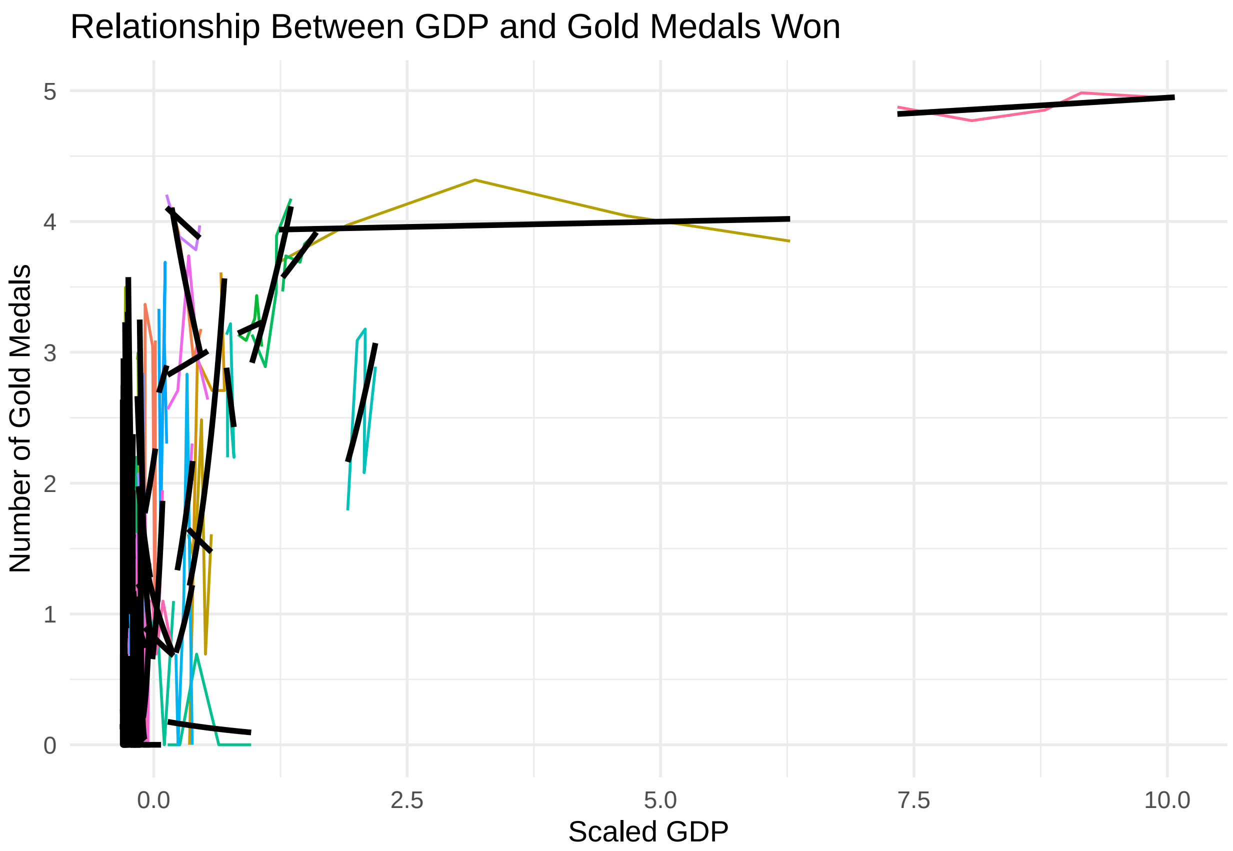
```
## `geom_smooth()` using formula = 'y ~ x'
```



GDP vs. Gold Medals (Using geom_path())

```
ggplot(gold_medals_data1, aes(x = gdp_scaled, y = log(n+1), group = team, colour = team)) +
  geom_path() +  # Connect points with a non-directed line to visualize changes
  geom_smooth(method = "glm", method.args = list(family = quasipoisson),
              se = FALSE, colour = "black") +
  # Overall trend
  theme_minimal() +
  theme(legend.position = "none") +  # Hide the legend to avoid clutter
  labs(
    title = "Relationship Between GDP and Gold Medals Won",
    x = "Scaled GDP",
    y = "Number of Gold Medals"
  )
```

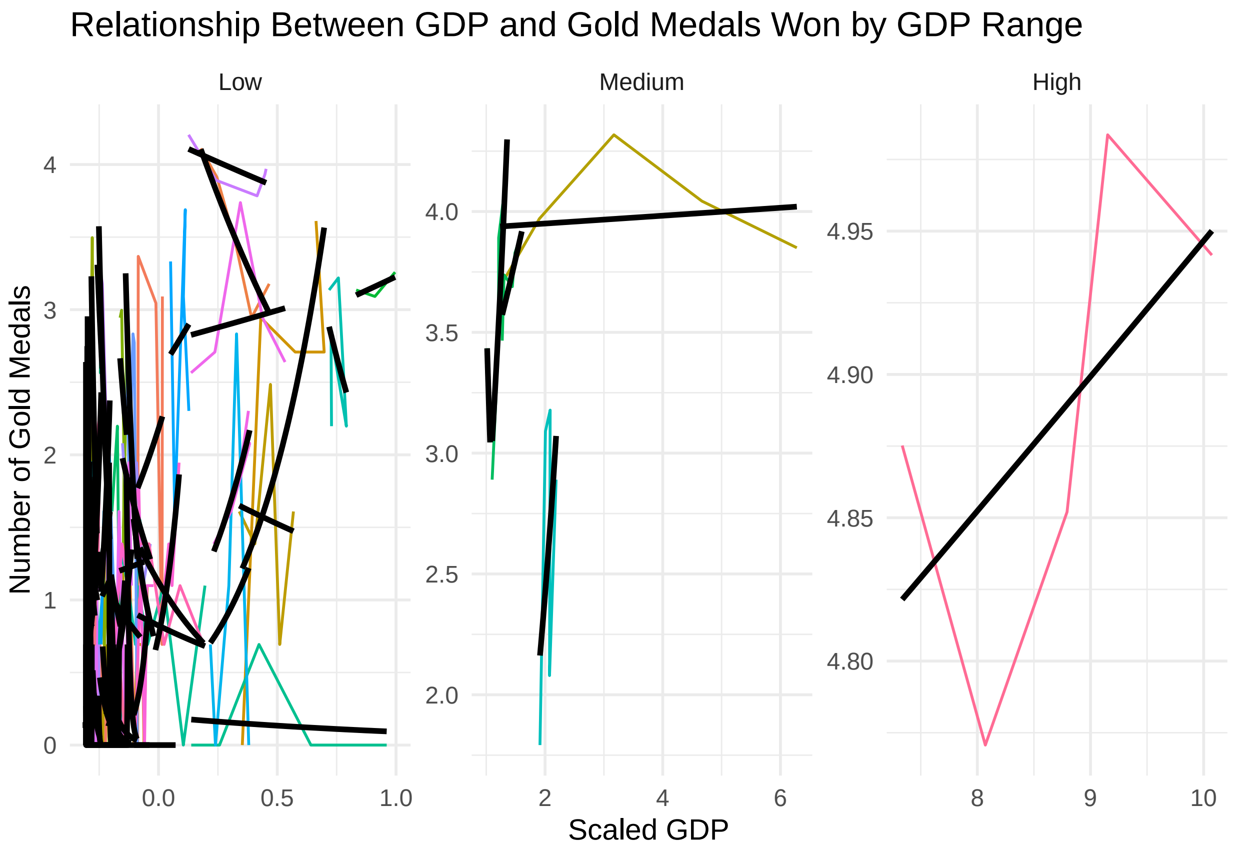
```
## 'geom_smooth()' using formula = 'y ~ x'
```



The current plot shows some dense clusters and disconnected lines that make interpretation challenging. Thus, I am going to divide the x-axis (Scaled GDP) into three parts, and present each seperately.

```
# Add a column indicating the GDP range
gold_medals_data_clean <- gold_medals_data1 %>%
  mutate(gdp_range = case_when(
    gdp_scaled < 1 ~ "Low",
    gdp_scaled >= 1 & gdp_scaled <= 7 ~ "Medium",
    gdp_scaled > 7 ~ "High"
  ))

# Set the order: 'Low', 'Medium', 'High'
gold_medals_data_clean <- gold_medals_data_clean %>%
  mutate(gdp_range = factor(gdp_range, levels = c("Low", "Medium", "High")))

# Plot with facet wrap
ggplot(gold_medals_data_clean, aes(x = gdp_scaled, y = log(n+1), group = team, colour = team)) +
  geom_path() +
  geom_smooth(method = "glm", method.args = list(family = quasipoisson), se = FALSE, colour = "black") +
  theme_minimal() +
  theme(legend.position = "none") +
  labs(
    title = "Relationship Between GDP and Gold Medals Won by GDP Range",
    x = "Scaled GDP",
    y = "Number of Gold Medals"
  ) +
  facet_wrap(~ gdp_range, scales = "free")
```

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

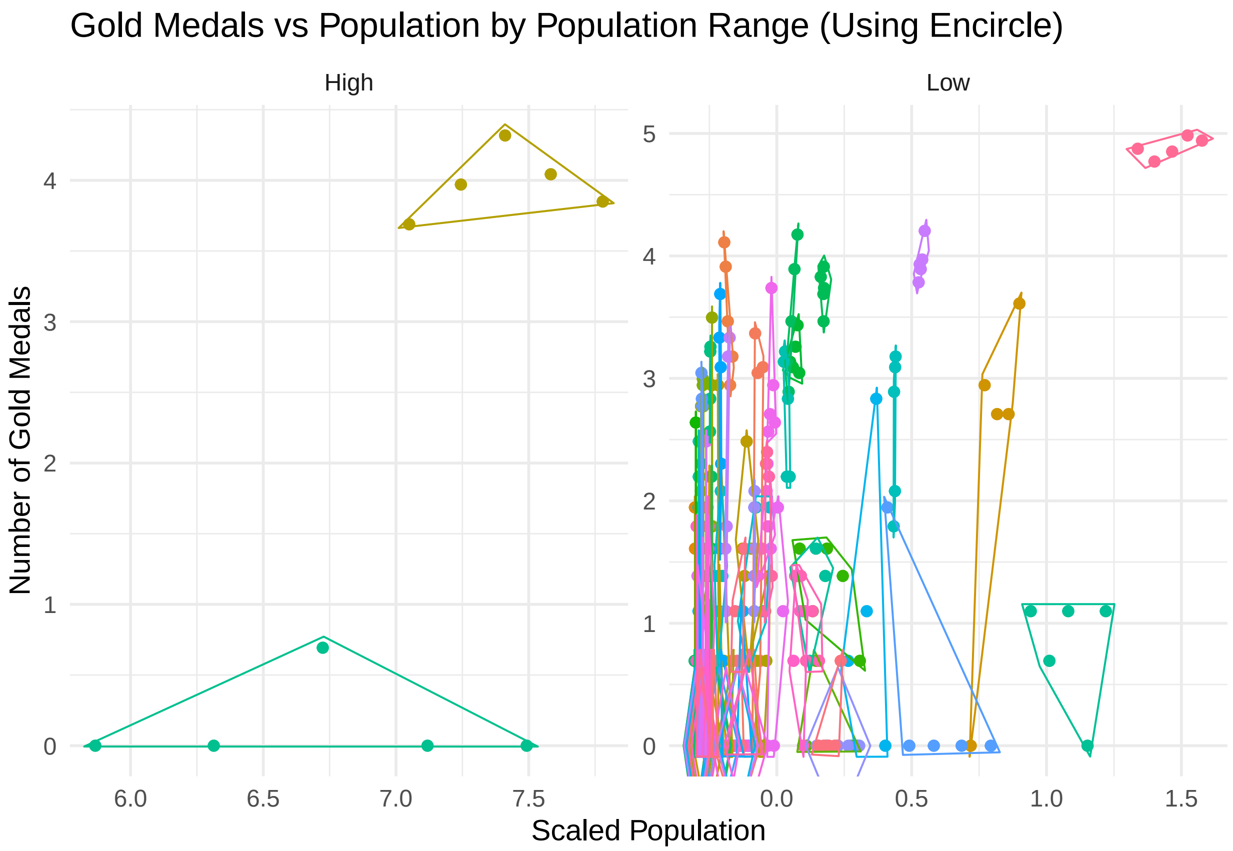


The plot shows the **relationship between GDP and the number of gold medals won**, split into three different GDP ranges: Low (`gdp_scaled < 1`), Medium (`1 <= gdp_scaled <= 7`), and High (`gdp_scaled > 7`).

- The data in the **Low GDP** range is highly clustered, with a lot of **variability and overlapping lines**. Most countries with low GDP do not win a significant number of gold medals. However, there are exceptions, where some countries show occasional spikes in their gold medal counts.
- Countries in the **Medium GDP** range show a more consistent and positive relationship between GDP and the number of gold medals won. The **black smoothed trend line** suggests that as GDP increases, the number of gold medals tends to also increase.
- Countries in the **High GDP** range tend to consistently win a **large number of gold medals**. The trend line shows a **steady increase**, indicating that even within the high GDP group, increased GDP is correlated with higher medal counts.

Population vs. Gold Medals (Using Ellipses to Show Grouping)

```
# Define breaks for splitting population into three ranges (adjust these values as needed)
gold_medals_data1 <- gold_medals_data1 %>%
  mutate(pop_range = case_when(
    pop_scaled < 2 ~ "Low",
    pop_scaled >= 2 & pop_scaled <= 5 ~ "Medium",
    pop_scaled > 5 ~ "High"
  ))

# Plot using geom_encircle for more flexible ellipses
ggplot(gold_medals_data1, aes(x = pop_scaled, y = log(n+1), colour = team)) +
  geom_point() +
  geom_encircle(aes(group = team), s_shape = 1, expand = 0.02) +  # Add flexible ellipses
  theme_minimal() +
  theme(legend.position = "none") +
  labs(
    title = "Gold Medals vs Population by Population Range (Using Encircle)",
    x = "Scaled Population",
    y = "Number of Gold Medals"
  ) +
  facet_wrap(~ pop_range, scales = "free", nrow = 1)
```



High Population Range (`pop_scaled > 5`): There are only two countries in this population range. One of these countries performs notably well, with gold medal counts around `50` or above (with `log(n+1)` around 4). The other country has a much lower medal count, close to `0`. This suggests that having a high population alone does not guarantee success in winning Olympic medals.

Low Population Range (`pop_scaled < 2`): The low population range shows the most variability, with a dense cluster of countries exhibiting a wide range of gold medal counts. Notably, one country won more than `100` gold medals (with `log(n+1)` around 5). This indicates that countries with smaller populations can still achieve great success.

Conclusion: The plot does not reveal a significant positive relationship between population size and gold medal counts. Population size is not necessarily a strong predictor of success in terms of gold medal counts.

e. Fit the model and Run Diagnostics

- To address the warning 'Some predictor variables are on very different scales', I scaled `gdp` and `pop` using standard scaling (subtracting the mean and dividing by the standard deviation). This step helps improve numerical stability and prevents issues arising from predictor variables having very different magnitudes. To keep the 'data.frame' class instead of 'matrix', I will use `datawizard::standardize()` instead of `scale()`.
- To solve the 'boundary (singular) fit' warning, which occurs when one or more variance components are estimated to be close to 0, indicating that the random effect might not be necessary, I removed `year` as the random-effects grouping variable because the random effect for `year (Intercept)` had an estimated variance of `0.000`. This resulted in a simplified model that includes year as a spline-based fixed effect and team as a random effect.
- To address the "Convergence Issue", I added `control = glmerControl(optimizer = "bobyqa")`. The `bobyqa` optimizer is often more robust for mixed models, particularly when dealing with complex random effects structures, and can help the model converge more reliably.

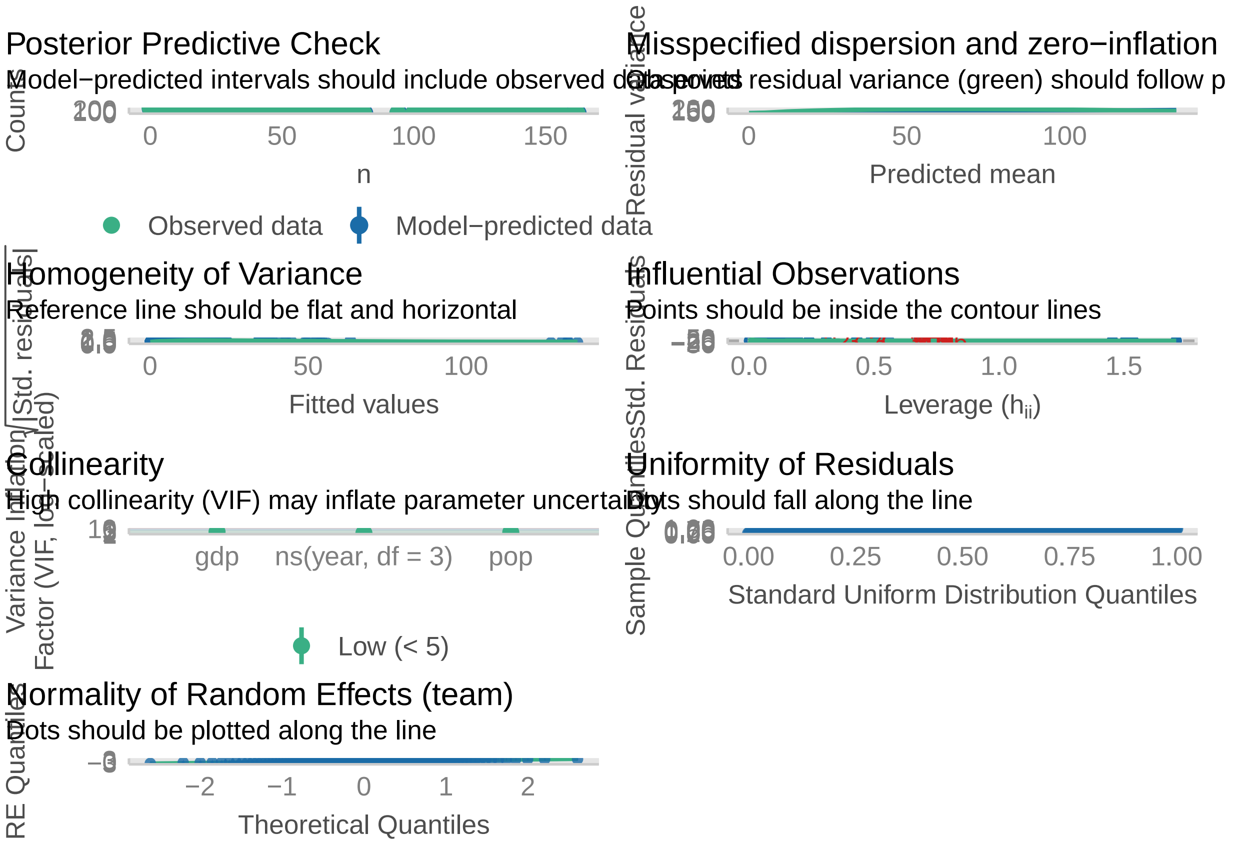
```r
# Fit the mixed-effects model

gold_medals_data <- datawizard::standardize(gold_medals_data, select = c("gdp", "pop"))

gold_model <- glmer(n ~ ns(year, df = 3) +
                      # time (year) as a fixed, continuous predictor variable using natural splines
                      gdp + pop +
                      # other fixed-effect predictor variables: gdp, pop (both scaled)
                      (1 | team),
                    # random-effects grouping variable: country (team)
                    data = gold_medals_data,
                    family = poisson(link = "log"),
                    # Poisson distribution for counts
                    control = glmerControl(optimizer = "bobyqa"))
# to resolve the convergence issues

# Summarize the model
summary(gold_model)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: n ~ ns(year, df = 3) + gdp + pop + (1 | team)
##    Data: gold_medals_data
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
##   2420.7   2450.8  -1203.4   2406.7      534
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.0431 -0.6359 -0.3039  0.3324  6.8781
##
## Random effects:
##  Groups Name        Variance Std.Dev.
##  team   (Intercept) 4.66     2.159
## Number of obs: 541, groups:  team, 109
##
## Fixed effects:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.18908    0.22645  -0.835   0.4037
## ns(year, df = 3)1 -0.11155    0.07769  -1.436   0.1510
## ns(year, df = 3)2 -0.10176    0.10572  -0.962   0.3358
## ns(year, df = 3)3 -0.11376    0.05845  -1.946   0.0516 .
## gdp                 0.02175    0.04092   0.531   0.5951
## pop                 0.47461    0.20885   2.272   0.0231 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) n(,d=3)1 n(,d=3)2 n(,d=3)3 gdp
## ns(yr,d=3)1 -0.059
## ns(yr,d=3)2 -0.163  0.045
## ns(yr,d=3)3 -0.052  0.359    0.150
## gdp          0.089 -0.163   -0.295   -0.391
## pop         -0.045  0.005    0.033    0.045   -0.654
```

```
# Check for convergence issues and model performance
check_convergence(gold_model)
```

```
## [1] TRUE
## attr(,"gradient")
## [1] 5.879716e-07
```

```
check_model(gold_model)
```



1. Posterior Predictive Check: There is a clustering of observed counts at lower values (around `0`, where the model-predicted intervals fails to include the observed data points), with a gradual tapering off as values increase. The model appears to have difficulty capturing some of the variability, especially at higher counts, suggesting that the model might not fit the extreme values well.

2. Misspecified Dispersion and Zero-inflation: The `observed variance` (green line) should ideally follow the `predicted variance` (blue line). However, there seems to be a mismatch, particularly for lower predicted means, which could indicate issues with the model's ability to handle **over-dispersion or zero-inflation** effectively.

3. Homogeneity of Variance: Ideally, the points should be scattered evenly around the horizontal line without clear patterns. Here, there seems to be a slight trend and increased spread for higher fitted values, suggesting **heteroscedasticity**. This indicates that the model may not fit all regions of the data equally well.

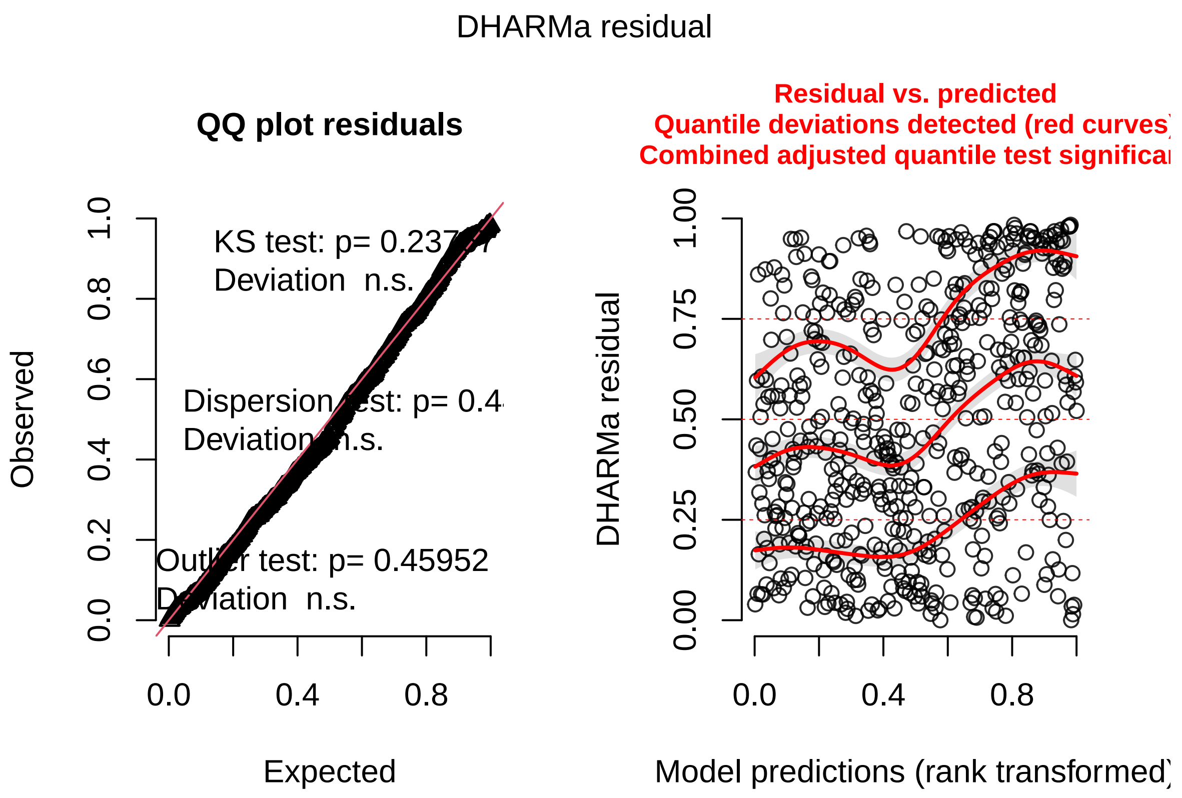
4. Influential Observations: The highlighted points (like `92, 325`) have higher leverage and residuals, suggesting they may be influential data points. These should be investigated further to determine if they represent outliers or data quality issues.

5. Collinearity: `gdp`, `ns(year, df = 3)` and `pop` all show acceptable VIF values (less than `5`), indicating that collinearity may not be a big issue for this model.

6. Uniformity of Residuals: The dots generally fall along the line. This indicates that the residuals are generally uniformly distributed.

7. Normality of Random Effects (team): There are deviations, especially low values of theoretical quantiles, indicating that the random effects may not be normally distributed, which could affect the model fit.

```r
residuals <- simulateResiduals(gold_model)
plot(residuals)
```



1. QQ Plot Residuals: The **QQ plot** shows that the residuals generally align well with the expected distribution, with the p-values from the KS, dispersion, and outlier tests all being non-significant (`p > 0.05`). This suggests that the model does not deviate significantly from the assumptions about the distribution of residuals.

2. Residual vs. Predicted: The **red curves** indicate deviations from the expected quantiles, and the adjusted quantile test is significant, meaning there is still some systematic deviation between residuals and predicted values. The deviations in the residuals indicate that the model could miss certain key interactions or effects, or that the fixed effects may not be fully capturing the structure in the data.

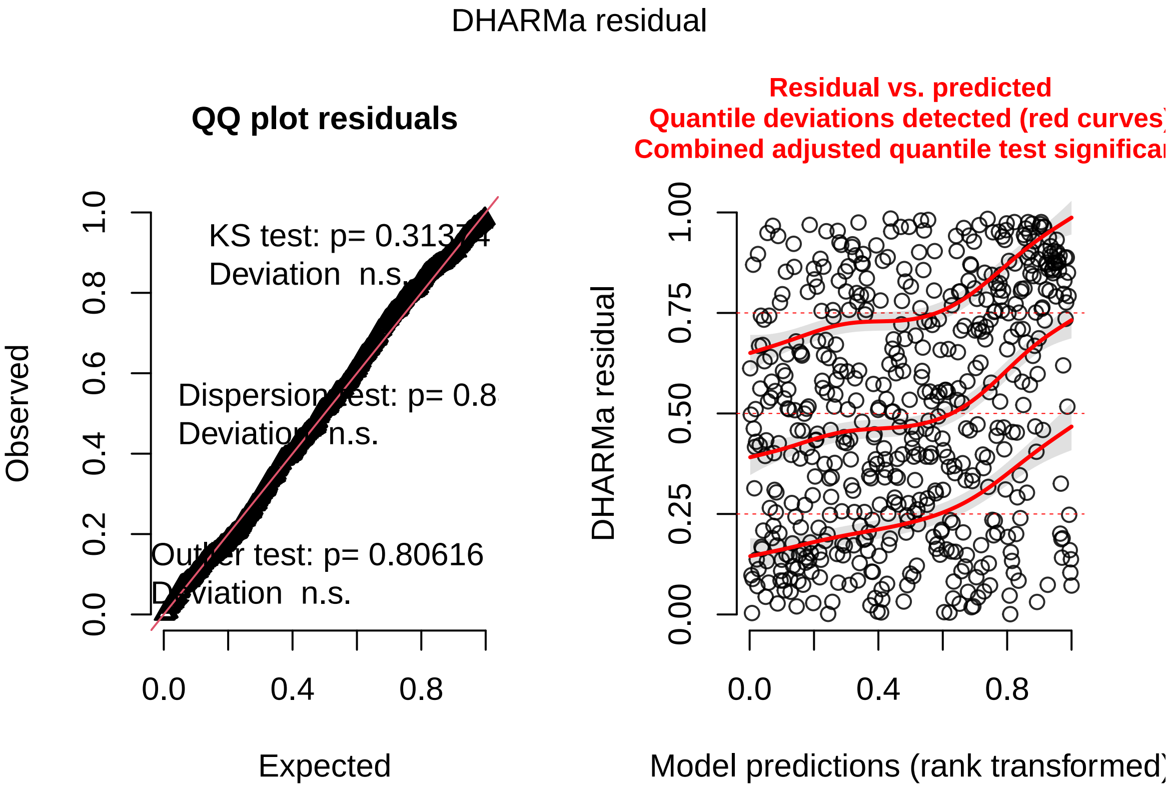
Address Overdispersion and Zero-inflation: we are going to switch to a **negative binomial** or zero-inflated model using `glmmTMB` to handle the overdispersion and excess zeros.

Interaction Term (`gdp_scaled:pop_scaled`): Adding an interaction between `gdp_scaled` and `pop_scaled` may be helpful in capturing combined effects.

```r
# Fit a negative binomial mixed-effects model
gold_model_nb <- glmmTMB(
  n ~ ns(year, df = 3) + gdp + pop + gdp:pop + (1 | team),
  data = gold_medals_data,
  family = nbinom2(link = "log") # Using the negative binomial family with a log link
)

# Summary of the model to assess the fit
summary(gold_model_nb)
```

```
##  Family: nbinom2  ( log )
## Formula:          n ~ ns(year, df = 3) + gdp + pop + gdp:pop + (1 | team)
## Data: gold_medals_data
##
##      AIC      BIC   logLik deviance df.resid
##   2043.6   2082.3  -1012.8   2025.6      532
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  team   (Intercept) 3.521    1.876
## Number of obs: 541, groups:  team, 109
##
## Dispersion parameter for nbinom2 family (): 1.81
##
## Conditional model:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.12013    0.23133   0.519 0.603554
## ns(year, df = 3)1 -0.33597    0.22988  -1.461 0.143879
## ns(year, df = 3)2 -0.43172    0.29829  -1.447 0.147813
## ns(year, df = 3)3 -0.20725    0.15623  -1.327 0.184665
## gdp                 1.04544    0.23066   4.532 5.83e-06 ***
## pop                 0.27405    0.20437   1.341 0.179950
## gdp:pop            -0.13188    0.03952  -3.337 0.000846 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Run DHARMa diagnostics
residuals_nb <- simulateResiduals(gold_model_nb)
plot(residuals_nb)
```

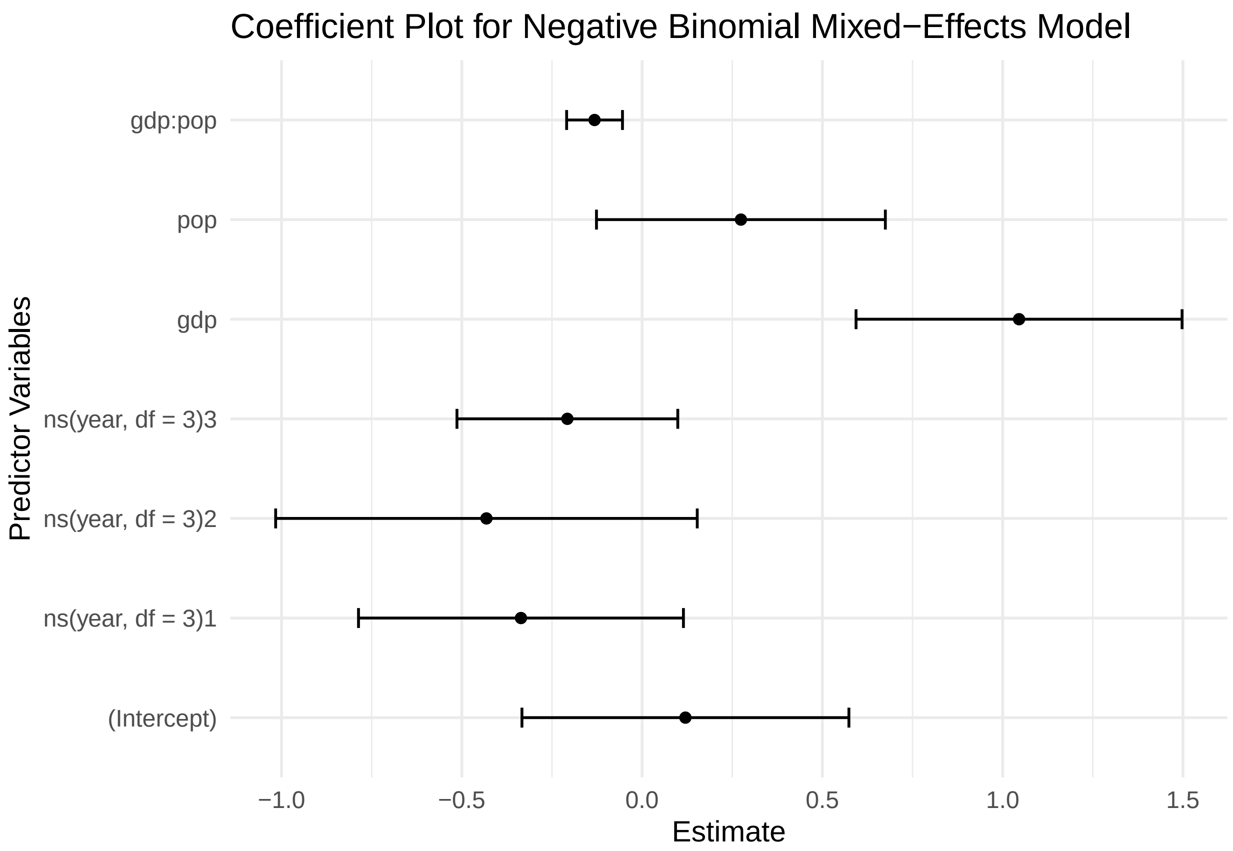


QQ Plot Residuals: The QQ plot shows a **fairly good alignment** with the diagonal line, suggesting that the residuals follow a reasonable distribution. The KS test indicates that the deviation is not significant (`p = 0.31374`), which is a positive sign.

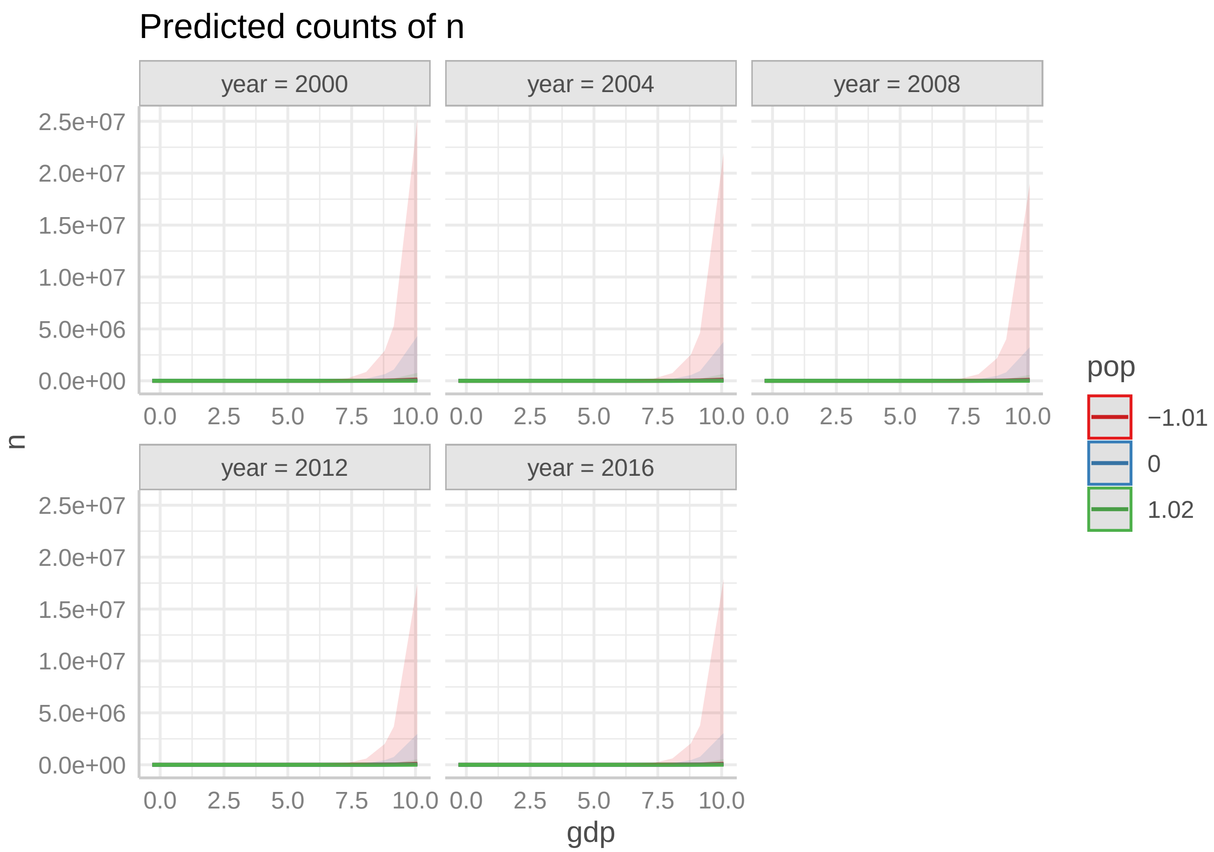
Residual vs. Predicted Plot: There are still quantile deviations detected, which means that there are some areas where the model isn't perfectly capturing the variability in the data, especially at higher values of model predictions.

f. Present the Results Graphically

```
# Create coefficient plot
coefficients <- tidy(gold_model_nb, effects = "fixed")
coefficients$term <- factor(coefficients$term, levels = coefficients$term) # Maintain the order of term

# Plot coefficients with 95% confidence intervals
coef_plot <- ggplot(coefficients, aes(x = estimate, y = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = estimate - 1.96 * std.error, xmax = estimate + 1.96 * std.error), height =
  theme_minimal() +
  labs(
    title = "Coefficient Plot for Negative Binomial Mixed-Effects Model",
    x = "Estimate",
    y = "Predictor Variables"
  )

print(coef_plot)
```



```r
# Use ggeffects to create effects plots
effects_nb_ggeffects <- ggpredict(gold_model_nb, terms = c("gdp [all]", "pop", "year"))
plot(effects_nb_ggeffects)
```



Q2. Toenail Data

```r
# Load the toenail data
data("toenail", package = "HSAUR3")
# force(toenail)
toenail$outcome_binary <- as.numeric(toenail$outcome == "moderate or severe")
```

a. Describe the Maximal Model

- The *response variable* is `outcome` (**binary**: `moderate or severe` or `none or mild`).
- *Fixed effects*: `treatment` (**binary**: `terbinafine` or `itraconazole`), `time` (a continuous variable) and 'visit' (an integer).
- *Random effect*: `patientID` to account for repeated measures within each patient.

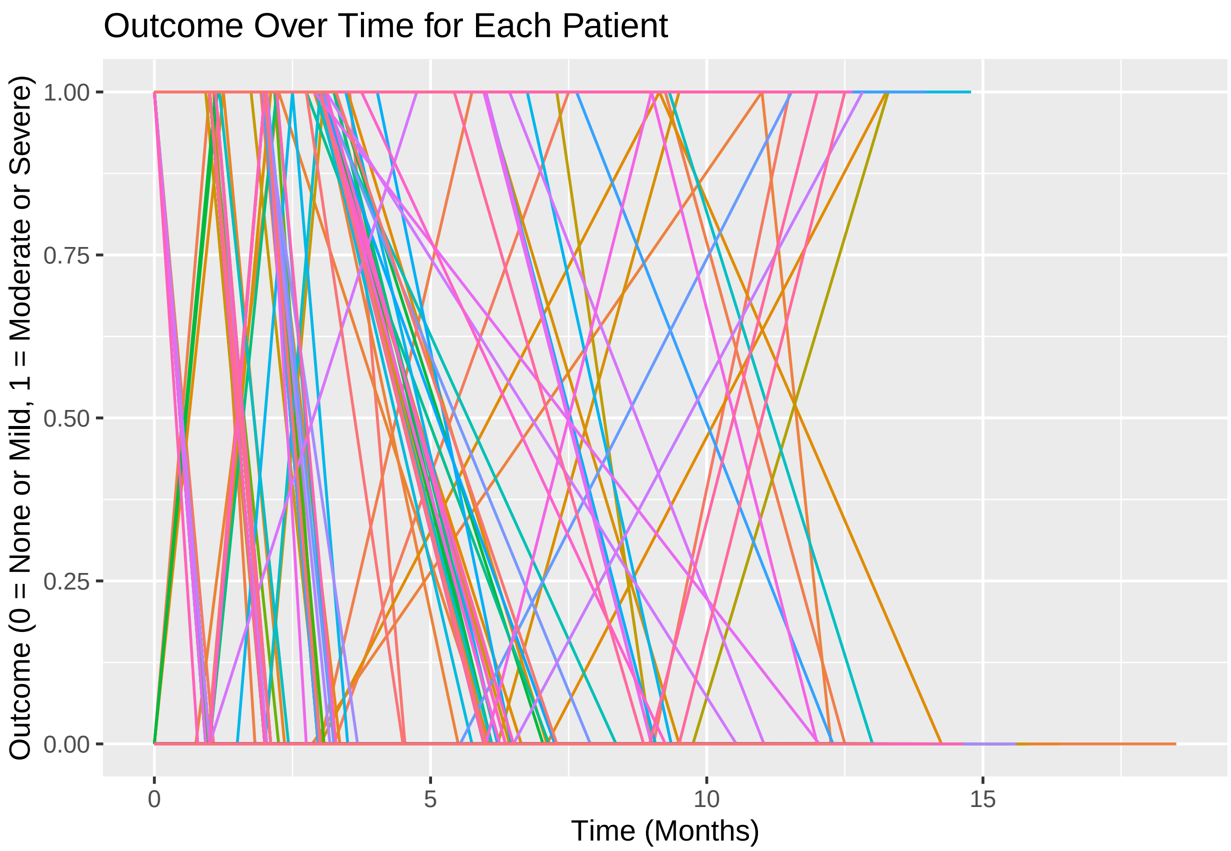
c. Specify the initial model to fit

- The model I would like to fit initially is `toenail_model <- glmmTMB(outcome_binary ~ treatment + time + treatment:time + (1 | patientID), data = toenail, family = binomial(link = "logit"))`, a mixed-effects logistic regression model (binomial family) to account for repeated measures.

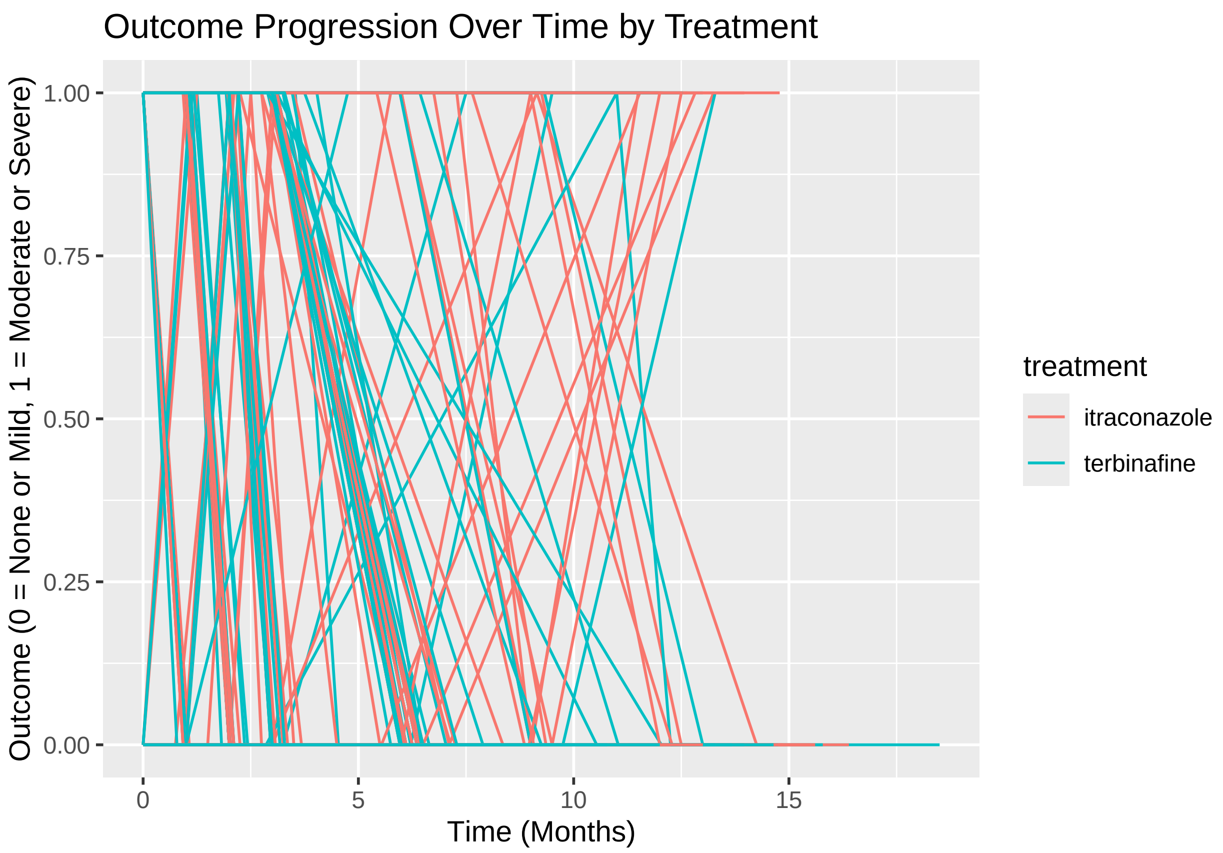
- I decided not to include `visit` as a predictor in the model because it is closely related to `time`, which already provides a **continuous** measure of treatment progression. Including both `visit` and `time` would introduce redundancy and unnecessary complexity without adding substantial new information to the model.

d. Create Exploratory Plots

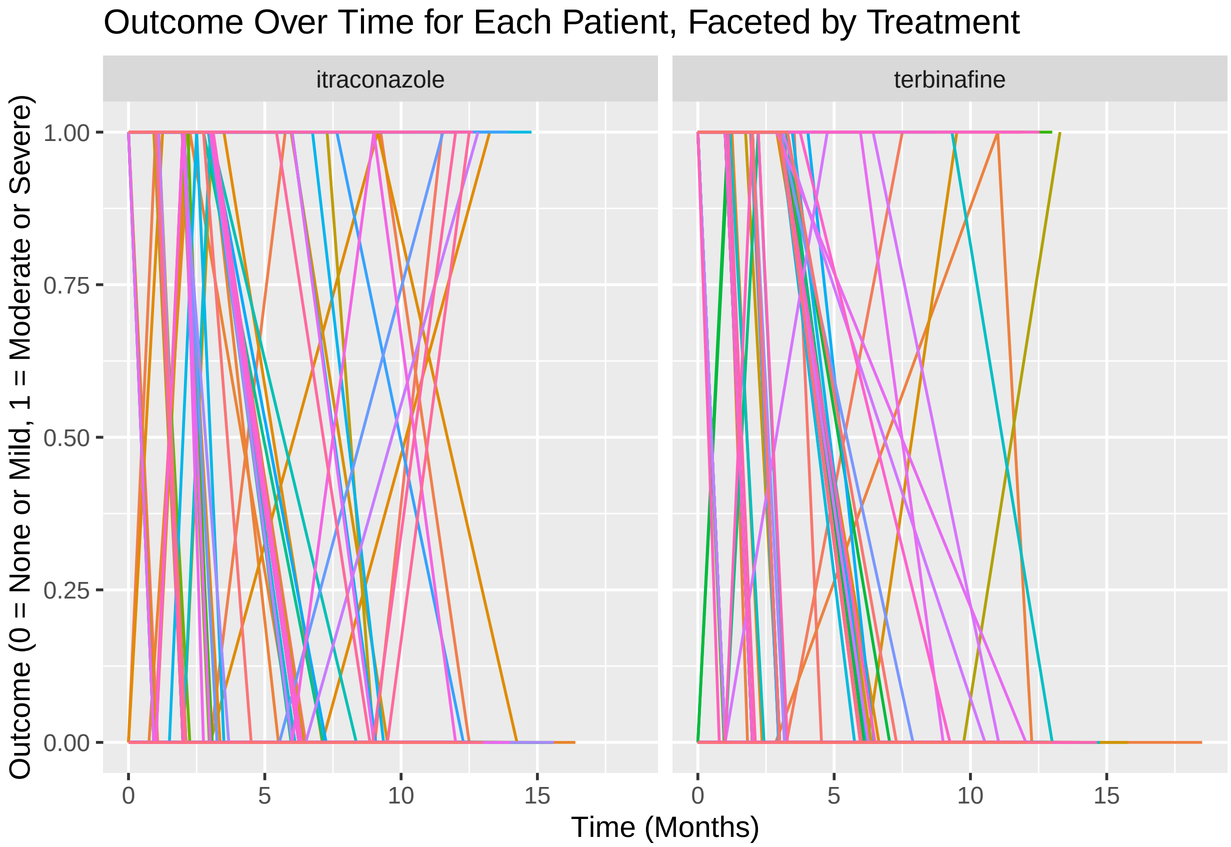
```
# Line plot for time vs outcome with patientID grouping
ggplot(toenail, aes(x = time, y = outcome_binary, group = patientID, color = factor(patientID))) +
  geom_line() +
  theme(legend.position = "none") +
  labs(title = "Outcome Over Time for Each Patient",
       x = "Time (Months)",
       y = "Outcome (0 = None or Mild, 1 = Moderate or Severe)")
```



```
# Path plot to show progression by treatment for each patient
ggplot(toenail, aes(x = time, y = outcome_binary, group = patientID, color = treatment)) +
  geom_path() +
  theme(legend.position = "right") +
  labs(title = "Outcome Progression Over Time by Treatment",
       x = "Time (Months)",
       y = "Outcome (0 = None or Mild, 1 = Moderate or Severe)")
```

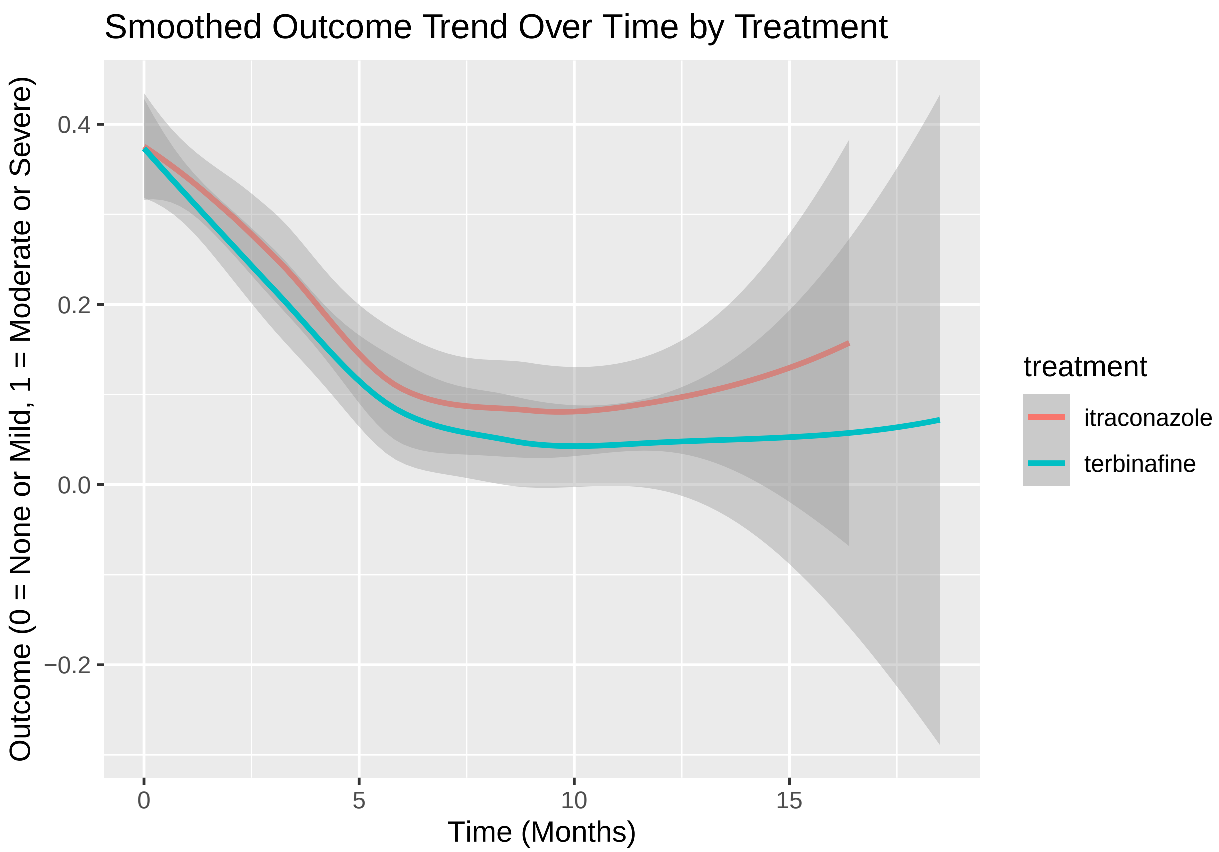


```
# Faceted plot by treatment group to see different trends
ggplot(toenail, aes(x = time, y = outcome_binary, group = patientID, color = patientID)) +
  geom_line() +
  facet_wrap(~ treatment) +
  theme(legend.position = "none") +
  labs(title = "Outcome Over Time for Each Patient, Faceted by Treatment",
       x = "Time (Months)",
       y = "Outcome (0 = None or Mild, 1 = Moderate or Severe)")
```



```
# Smoothed trend line using geom_smooth, stratified by treatment
ggplot(toenail, aes(x = time, y = outcome_binary, color = treatment)) +
  geom_smooth(method = "loess") +
  labs(title = "Smoothed Outcome Trend Over Time by Treatment",
       x = "Time (Months)",
       y = "Outcome (0 = None or Mild, 1 = Moderate or Severe)")
```

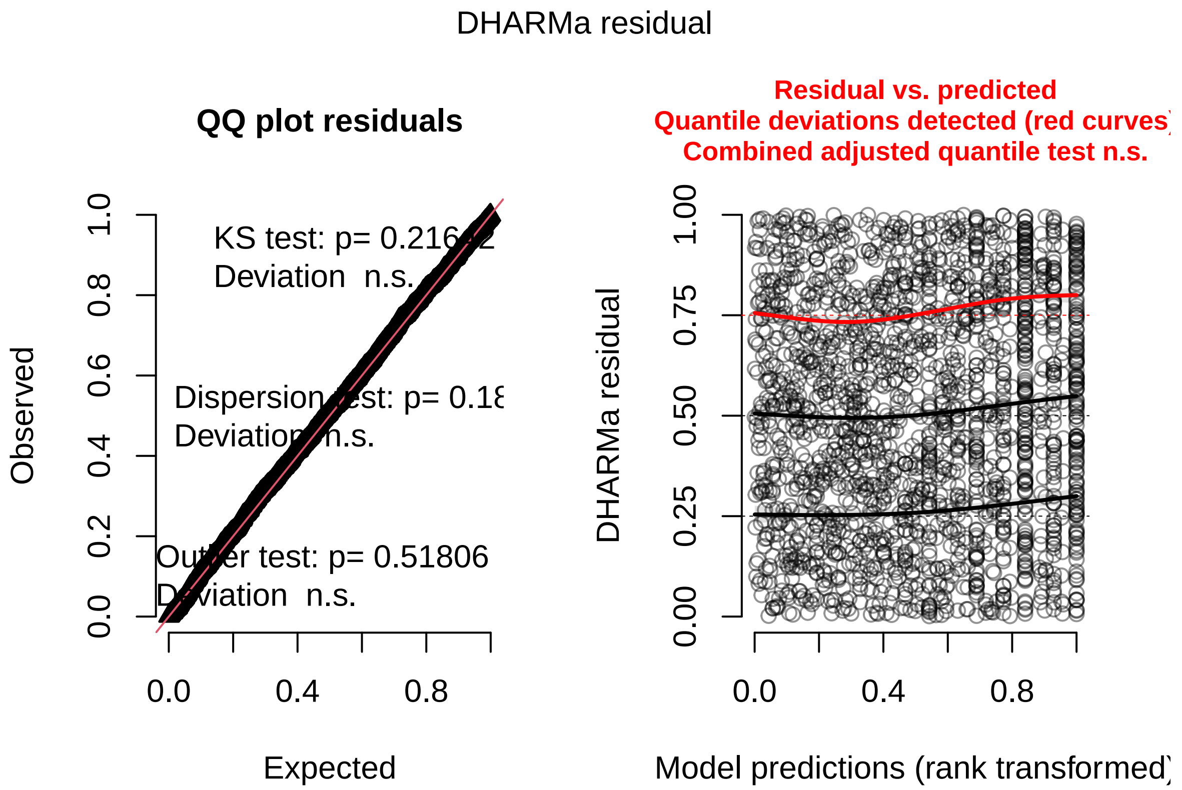
```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```



e. Fit the model and Run Diagnostics

```
# Fit a mixed-effects probit regression model (binomial family) to account for repeated measures
toenail_model <- glmmTMB(
  outcome_binary ~ treatment + time + (1 | patientID),
  data = toenail,
  family = binomial(link = "probit"))
```

```
residuals <- simulateResiduals(toenail_model)
plot(residuals)
```

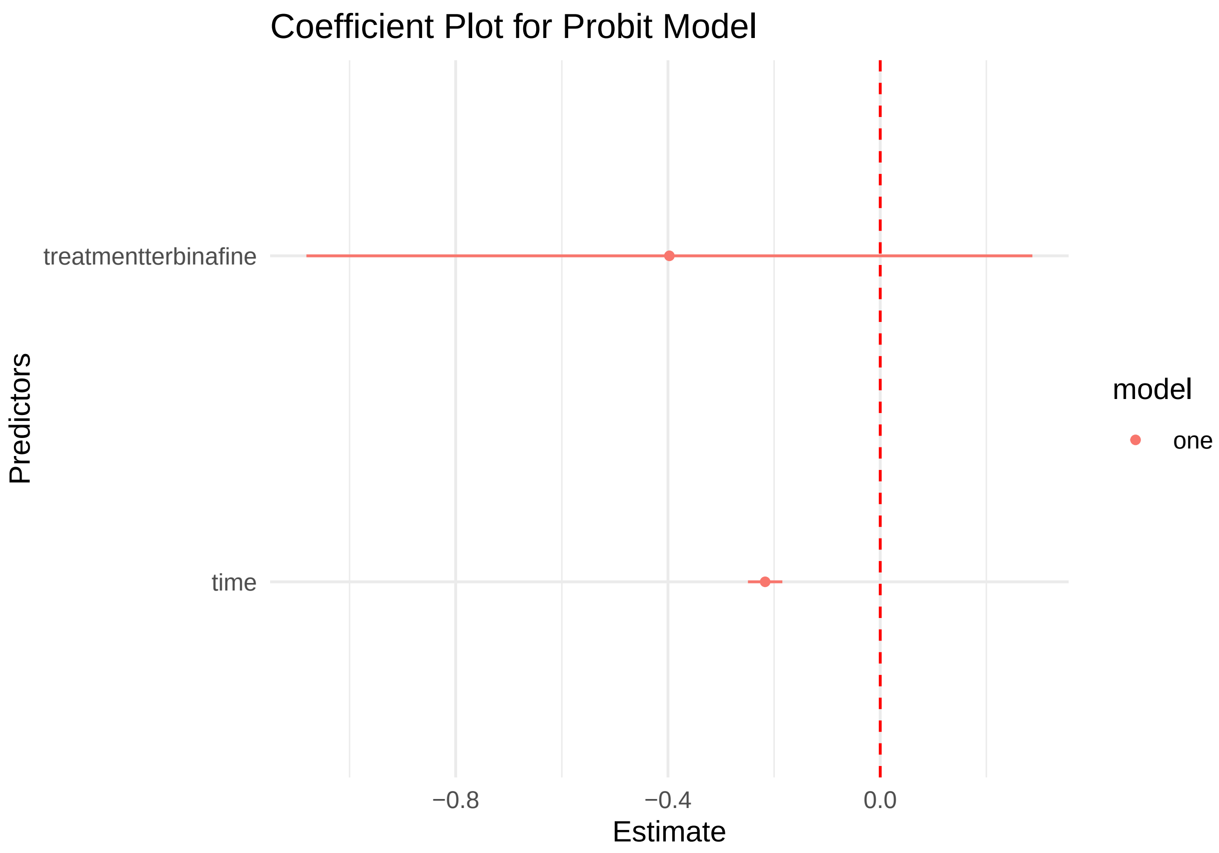


The **QQ Plot residuals** shows that the `p-value` for *KS test* (`0.21642`), *Dispersion test* (`0.184`), and *outlier test* (`0.51806`) are all non-significant (`< 0.05`). The residuals align closely with the diagonal line, indicating a good match between the observed and expected distribution.

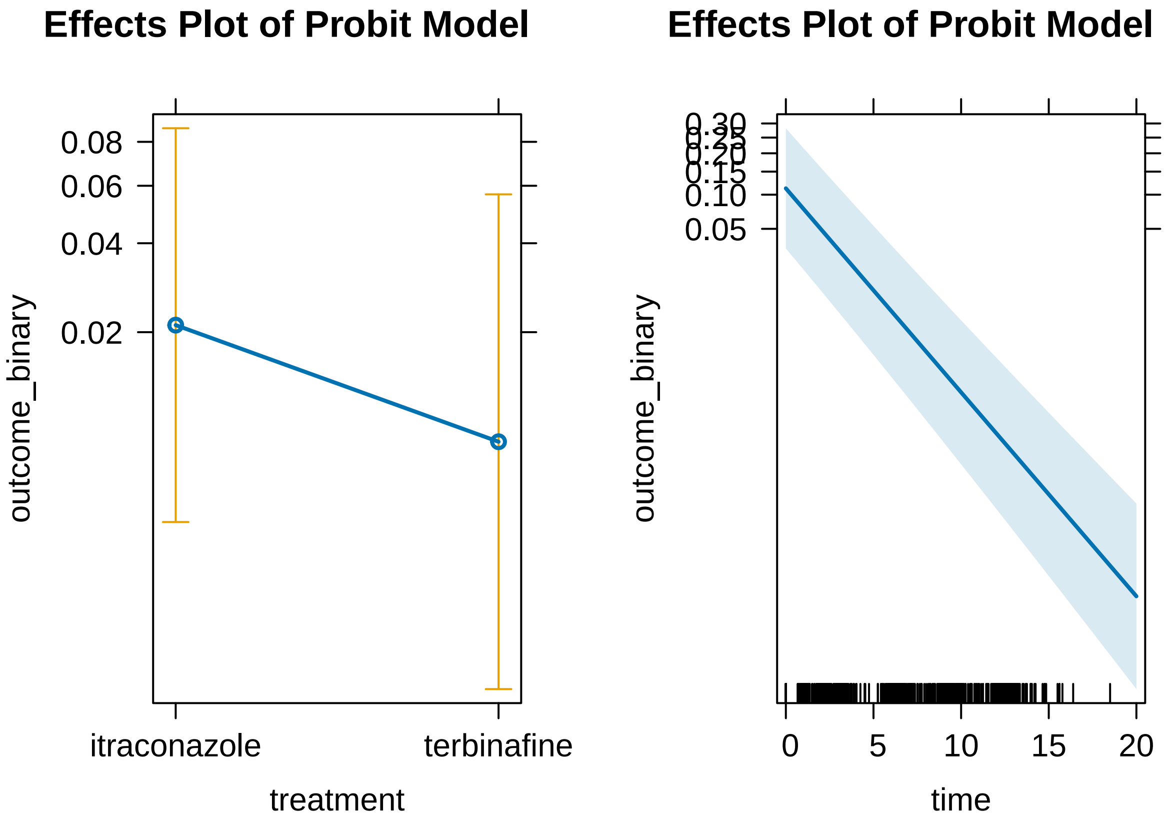
The **residuals vs. predicted values plot** shows that the combined quantile test is non-significant, suggesting that the model captures the quantiles reasonably well. The red curve indicates a slight deviation, but it is not significant overall, suggesting a reasonable fit.

f. Present the Results Graphically

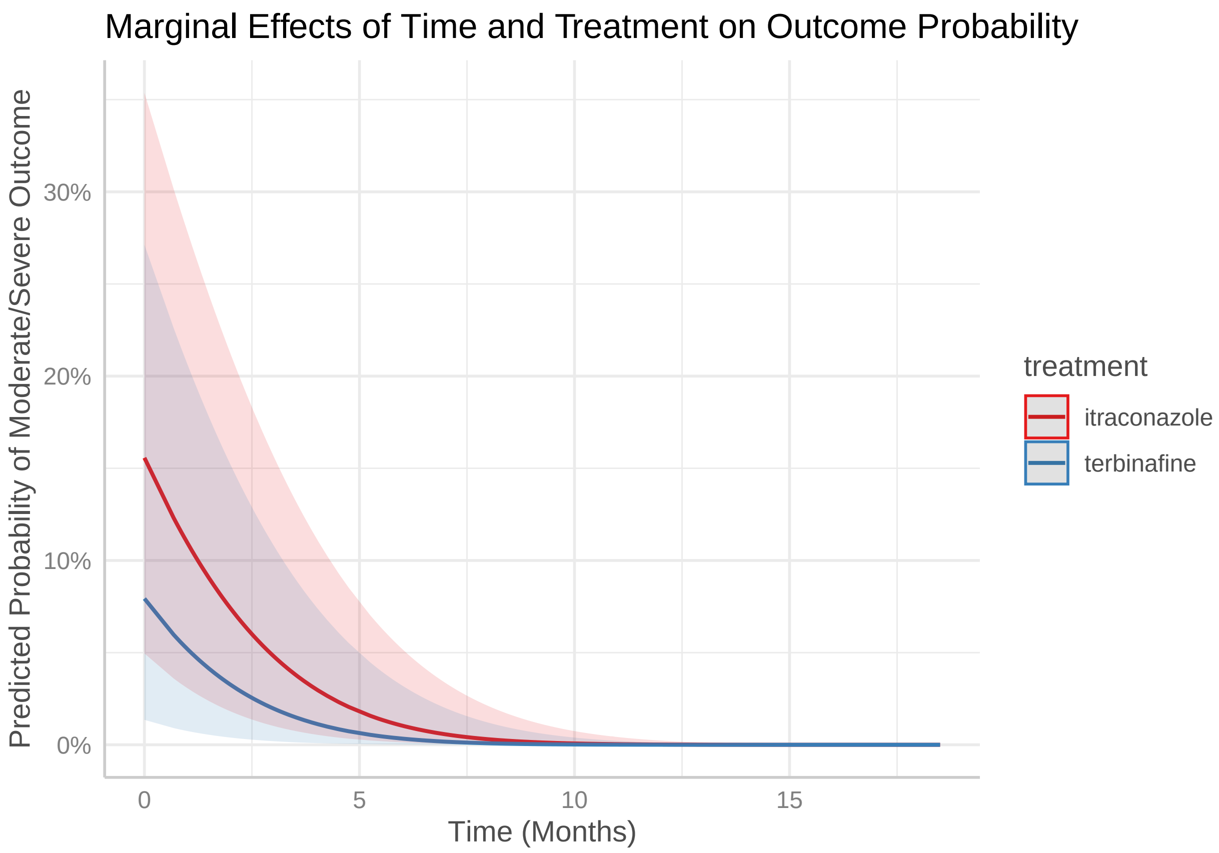
```r
# Use tidy from broom.mixed to get coefficients
model_tidy <- tidy(toenail_model, effects = "fixed")

# Create the coefficient plot using dotwhisker
dwplot(model_tidy) +
  theme_minimal() +
  labs(title = "Coefficient Plot for Probit Model",
       x = "Estimate",
       y = "Predictors") +
  geom_vline(xintercept = 0, linetype = "dashed", color = "red")  # Adds a vertical line at zero
```



```r
# Generate effects plot for the model
effect_plot <- allEffects(toenail_model)
plot(effect_plot, main = "Effects Plot of Probit Model")
```



```r
# Generate marginal effects for time and treatment
marginal_effects <- ggpredict(toenail_model, terms = c("time [all]", "treatment"))

# Plot marginal effects using ggplot2
plot(marginal_effects) +
  labs(title = "Marginal Effects of Time and Treatment on Outcome Probability",
       x = "Time (Months)",
       y = "Predicted Probability of Moderate/Severe Outcome")
```



g. Compares the Fixed-effect Parameters

(a) a completely pooled analysis (using `glm`)

```
# Fit a completely pooled model using glm
toenail_glm <- glm(
  outcome_binary ~ treatment + time,
  data = toenail,
  family = binomial(link = "probit"))

# Extract coefficients from glm model
toenail_glm_tidy <- tidy(toenail_glm, conf.int = TRUE) %>%
  rename(conf.low = conf.low, conf.high = conf.high) %>%
  mutate(model = "GLM (Pooled)")
```

(b) an analysis using penalized quasi-likelihood (with `MASS::glmmPQL`)

```
# Fit a penalized quasi-likelihood model using glmmPQL
toenail_pql <- glmmPQL(
  fixed = outcome_binary ~ treatment + time,
  random = ~ 1 | patientID,
  family = binomial(link = "probit"),
  data = toenail)
```

```
## iteration 1
## iteration 2

## iteration 3

## iteration 4

## iteration 5

## iteration 6

## iteration 7

## iteration 8

## iteration 9

## iteration 10
```

```r
# Extract coefficients from glmmPQL model
toenail_pql_tidy <- tidy(toenail_pql, effects = "fixed", conf.int = TRUE) %>%
  rename(conf.low = conf.low, conf.high = conf.high) %>%
  mutate(model = "glmmPQL (PQL)")
```

(c) Laplace approximation

```r
# Fit a model using Laplace approximation with glmer
toenail_glmer <- glmer(
  outcome_binary ~ treatment + time + (1 | patientID),
  data = toenail,
  family = binomial(link = "probit"))

# Extract coefficients from glmer model
toenail_glmer_tidy <- tidy(toenail_glmer, effects = "fixed", conf.int = TRUE) %>%
  rename(conf.low = conf.low, conf.high = conf.high) %>%
  mutate(model = "glmer (Laplace)")
```

(d) adaptive Gauss-Hermite quadrature

```r
# Fit model using glmer with 10 quadrature points
toenail_glmer_10 <- glmer(
  outcome_binary ~ treatment + time + (1 | patientID),
  data = toenail,
  family = binomial(link = "probit"),
  nAGQ = 10)

# Fit model using glmer with 20 quadrature points
toenail_glmer_20 <- glmer(
  outcome_binary ~ treatment + time + (1 | patientID),
  data = toenail,
  family = binomial(link = "probit"),
  nAGQ = 20)

# Extract coefficients from the model with 10 quadrature points
glmer_10_tidy <- tidy(toenail_glmer_10, effects = "fixed", conf.int = TRUE) %>%
  rename(conf.low = conf.low, conf.high = conf.high) %>%
  mutate(model = "glmer (AGQ 10)")

# Extract coefficients from the model with 20 quadrature points
glmer_20_tidy <- tidy(toenail_glmer_20, effects = "fixed", conf.int = TRUE) %>%
  rename(conf.low = conf.low, conf.high = conf.high) %>%
  mutate(model = "glmer (AGQ 20)")
```

(e) credible intervals from a Bayesian model (using `MCMCglmm`)

```
# Fit a Bayesian model using MCMCglmm
prior_spec <- list(
  R = list(V = 1, nu = 0.002),  # Prior for residual variance
  G = list(G1 = list(V = 1, nu = 0.002))  # Prior for random effect variance
  )

toenail_mcmc <- MCMCglmm(
  outcome_binary ~ treatment + time,
  random = ~ patientID,
  family = "categorical",
  data = toenail,
  prior = prior_spec,
  verbose = FALSE,
  nitt = 13000, burnin = 3000, thin = 10)
```

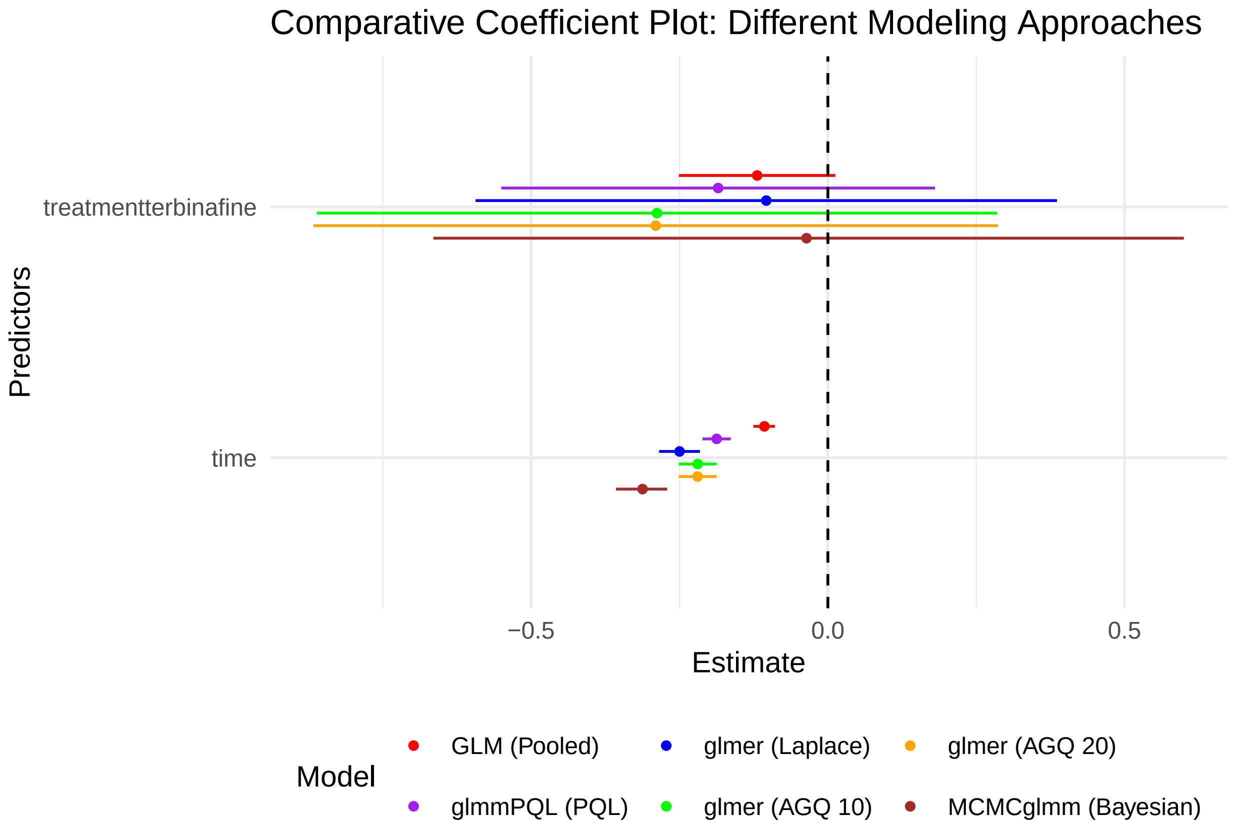
```
# Extract posterior means and 95% credible intervals
posterior_means <- summary(toenail_mcmc)$solutions
posterior_CI <- HPDinterval(toenail_mcmc$Sol)

# Combine results into a data frame
mcmc_results <- data.frame(
  term = rownames(posterior_means),
  estimate = posterior_means[, "post.mean"],
  lower = posterior_CI[, "lower"],
  upper = posterior_CI[, "upper"])

mcmc_tidy <-  mcmc_results %>%
  rename(estimate = estimate, conf.low = lower, conf.high = upper) %>%
  mutate(model = "MCMCglmm (Bayesian)")
```

Combine All Coefficients for Plotting

```
combined_results <- bind_rows(
  toenail_glm_tidy,
  toenail_pql_tidy,
  toenail_glmer_tidy,
  glmer_10_tidy,
  glmer_20_tidy,
  mcmc_tidy)
```

Create Coefficient Plot

```r
# Comparative coefficient plot using dotwhisker
dwplot(combined_results, dodge_size = 0.3) +
  theme_minimal() +
  labs(title = "Comparative Coefficient Plot: Different Modeling Approaches",
       x = "Estimate",
       y = "Predictors") +
  theme(legend.position = "bottom") +
  scale_color_manual(
    name = "Model",
    values = c("GLM (Pooled)" = "red",
               "glmmPQL (PQL)" = "purple",
               "glmer (Laplace)" = "blue",
               "glmer (AGQ 10)" = "green",
               "glmer (AGQ 20)" = "orange",
               "MCMCglmm (Bayesian)" = "brown")) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "black")
```



Priors used: In the Bayesian model, the following priors were specified: `prior_spec <- list(R = list(V = 1, nu = 0.002), G = list(G1 = list(V = 1, nu = 0.002)) )`.

Residual variance (R): The initial variance (`V`) is set to `1`, with degrees of freedom (`nu`) of `0.002`. This choice represents a **weakly informative prior**, indicating minimal prior information about the residual variability.

Random effect variance (G): Similar to the residual variance, the random effects variance (`G`) is also set with `V = 1` and `nu = 0.002`. This **weakly informative prior** allows the data to primarily determine the estimates while avoiding extreme values.

By setting both priors as weakly informative, the model maintains flexibility, letting the data play a significant role in determining parameter estimates, while providing slight regularization to aid in convergence and avoid overfitting.

Q3. Simulation and Fit Functions

```r
# Function to simulate a dataset similar to toenail data
simfun <- function(beta, theta, n_t, n_id) {
  # Arguments:
  #   beta: a vector of fixed effects coefficients (e.g., for intercept, treatment, time interaction)
  #   theta: a scalar for the random intercept standard deviation (random effect variance)
  #   n_t: number of time points for each individual
  #   n_id: number of individuals

  # Create individual IDs
  id <- factor(rep(1:n_id, each = n_t))

  # Assign two treatment groups ('A' or 'B')
  ttt <- factor(rep(sample(c("A", "B"), n_id, replace = TRUE), each = n_t))

  # Create time variable, an integer
  time <- rep(0:(n_t - 1), times = n_id)

  # Create a data frame with the necessary variables
  sim_data <- data.frame(id = id, ttt = ttt, time = time)

  # Add a placeholder for outcome_binary to fit the initial model
  sim_data$outcome_binary <- rbinom(n = nrow(sim_data), size = 1, prob = 0.5)

  # Fit a model to simulate from
  fit <- glmmTMB(outcome_binary ~ ttt * time + (1 | id),
                 family = binomial(link = "logit"),
                 data = sim_data)

  # Simulate new response values using glmmTMB::simulate
  sim_data$outcome_binary <- as.numeric(simulate(fit,
                                                  newdata = sim_data,
                                                  allow.new.levels = TRUE)[[1]])[1:nrow(sim_data)]

  return(sim_data)}
```

```r
# Function to fit a model to the simulated dataset
fitfun <- function(data, nAGQ) {
  # Arguments:
  #   data: a data frame containing the variables id, ttt, time, outcome_binary
  #   nAGQ: determines the fitting method (-2 for glm, -1 for glmmPQL, >=1 for glmer/GLMMadaptive)

  fit <- NULL  # Initialize an empty fit variable
  tryCatch({
    if (nAGQ == -2) {
      # Fit with glm (pooled)
      fit <- glm(outcome_binary ~ ttt * time, data = data, family = binomial(link = "logit"))
    } else if (nAGQ == -1) {
      # Fit with MASS::glmmPQL
      fit <- glmmPQL(fixed = outcome_binary ~ ttt * time, random = ~ 1 | id, family = binomial(link = "
    } else if (nAGQ >= 1) {
      if (nAGQ == 1) {
        # Fit with Laplace approximation using glmer, suppressing singular fit warnings
        fit <- glmer(outcome_binary ~ ttt * time + (1 | id), data = data, family = binomial(link = "log
      } else {
        # Fit with Adaptive Gauss-Hermite Quadrature using GLMMadaptive
        fit <- mixed_model(
          fixed = outcome_binary ~ ttt * time,
          random = ~ 1 | id,
          family = binomial(link = "logit"),
          data = data,
          nAGQ = nAGQ)}}},
    error = function(e) {
    message("Error in model fitting: ", e$message)
    fit <<- NULL
  })

  if (is.null(fit)) {
    return(data.frame(term = NA, estimate = NA, conf.low = NA, conf.high = NA, stringsAsFactors = FALSE
  }

  # Extract fixed-effect coefficients with confidence intervals
  results <- tryCatch({
    broom.mixed::tidy(fit, effects = "fixed", conf.int = TRUE)
  }, error = function(e) {
    message("Error in extracting coefficients: ", e$message)
    return(data.frame(term = NA, estimate = NA, conf.low = NA, conf.high = NA, stringsAsFactors = FALSE
  })

  return(results)}
```

```
# Function to run simulations and collect results
run_simulations <- function(n_t, nAGQ, n_sim = 100) {
  results_list <- vector("list", n_sim)

  for (i in 1:n_sim) {
    data <- simfun(beta, theta, n_t, n_id)
    fit_result <- tryCatch({
      fitfun(data, nAGQ)
    }, error = function(e) {
      message(sprintf("Error in iteration %d: %s", i, e$message))
      return(NULL)
    })

    if (!is.null(fit_result)) {
      results_list[[i]] <- fit_result
    }
  }

  # Remove NULL entries from results
  results_list <- results_list[!sapply(results_list, is.null)]

  # Combine all data frames
  if (length(results_list) > 0) {
    results_df <- bind_rows(results_list, .id = "simulation")
  } else {
    stop("No successful simulations were run.")
  }

  return(results_df)}
```

```
# Set parameters for simulation and fitting
beta <- c(-0.6, 0, -0.2, -0.05)
theta <- log(0.2)  # Random effect standard deviation for glmmTMB::simulate
n_id <- 300
```

```
# Run simulations for each method for n_t = 5
results_glm_5 <- run_simulations(n_t = 5, nAGQ = -2, n_sim = 100)
```

```
results_glmmPQL_5 <- run_simulations(n_t = 5, nAGQ = -1, n_sim = 100)
```

```
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3
```

The following 6 pages are useless.

```
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1
## iteration 2

## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2
## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2
## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 1
## iteration 1
```

```
results_glmer_5 <- run_simulations(n_t = 5, nAGQ = 1, n_sim = 100)
```

```
results_AGHQ_5 <- run_simulations(n_t = 5, nAGQ = 7, n_sim = 100)  # Example with nAGQ = 7

# Run simulations for each method for n_t = 10
results_glm_10 <- run_simulations(n_t = 10, nAGQ = -2, n_sim = 100)
```

```r
results_glmmPQL_10 <- run_simulations(n_t = 10, nAGQ = -1, n_sim = 100)
```

The following 6 pages are useless.

```
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 2

## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 2
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 1
## iteration 1
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
```

```
## iteration 1
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 1
## iteration 1
## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 1

## iteration 2

## iteration 3

## iteration 1

## iteration 2

## iteration 3

## iteration 1
## iteration 2
```

```
## iteration 3
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 3
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 1
## iteration 1
```

```
## iteration 2
```

```
## iteration 1
```

```
## iteration 2
```

```
## iteration 1
```

```
## iteration 2
```

```r
results_glmer_10 <- run_simulations(n_t = 10, nAGQ = 1, n_sim = 100)
```

```r
results_AGHQ_10 <- run_simulations(n_t = 10, nAGQ = 10, n_sim = 100)  # Example with nAGQ = 10
```

```r
# Analysis to compute bias, variance, RMSE, and coverage
analyze_results <- function(results_df, true_values) {
  results_summary <- results_df %>%
    filter(term %in% c("tttB", "tttB:time"), !is.na(estimate)) %>%
    group_by(term) %>%
    summarize(
      bias = mean(estimate - true_values[term], na.rm = TRUE),
      variance = var(estimate, na.rm = TRUE),
      rmse = sqrt(mean((estimate / true_values[term] - 1)^2, na.rm = TRUE)),
      coverage = mean(conf.low <= true_values[term] & conf.high >= true_values[term], na.rm = TRUE))
  return(results_summary)}

true_values <- setNames(beta, c("(Intercept)", "tttB", "time", "tttB:time"))
```

```r
# Analyze results for n_t = 5
summary_glm_5 <- analyze_results(results_glm_5, true_values)
summary_glmmPQL_5 <- analyze_results(results_glmmPQL_5, true_values)
summary_glmer_5 <- analyze_results(results_glmer_5, true_values)
summary_AGHQ_5 <- analyze_results(results_AGHQ_5, true_values)

# Analyze results for n_t = 10
summary_glm_10 <- analyze_results(results_glm_10, true_values)
summary_glmmPQL_10 <- analyze_results(results_glmmPQL_10, true_values)
summary_glmer_10 <- analyze_results(results_glmer_10, true_values)
summary_AGHQ_10 <- analyze_results(results_AGHQ_10, true_values)
```

```r
# Print summaries
print(summary_glm_5)
```

```
## # A tibble: 2 x 5
##   term         bias variance   rmse coverage
##   <chr>       <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      -0.0172  0.0587  Inf         0.84
## 2 tttB:time  0.0546  0.00981   2.25     0.83
```

```r
print(summary_glmmPQL_5)
```

```
## # A tibble: 2 x 5
##   term         bias variance   rmse coverage
##   <chr>       <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      0.00402   0.0467 Inf         0.9
## 2 tttB:time 0.0494    0.0111   2.32     0.74
```

```r
print(summary_glmer_5)
```

```
## # A tibble: 2 x 5
##   term        bias variance   rmse coverage
##   <chr>      <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      0.0355  0.0727   Inf         0.77
## 2 tttB:time 0.0411  0.00984   2.14     0.81
```

```r
print(summary_AGHQ_5)
```

```
## # A tibble: 2 x 5
##   term         bias variance   rmse coverage
##   <chr>       <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      -0.0338   0.0611 Inf         0.88
## 2 tttB:time  0.0516   0.0106   2.30     0.79
```

```r
print(summary_glm_10)
```

```
## # A tibble: 2 x 5
##   term        bias variance   rmse coverage
##   <chr>      <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      0.0121  0.0356   Inf         0.81
## 2 tttB:time 0.0498  0.00107   1.19     0.46
```

```r
print(summary_glmmPQL_10)
```

```
## # A tibble: 2 x 5
##   term        bias variance   rmse coverage
##   <chr>      <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      0.0215  0.0454   Inf         0.81
## 2 tttB:time 0.0478  0.00131    1.20     0.44
```

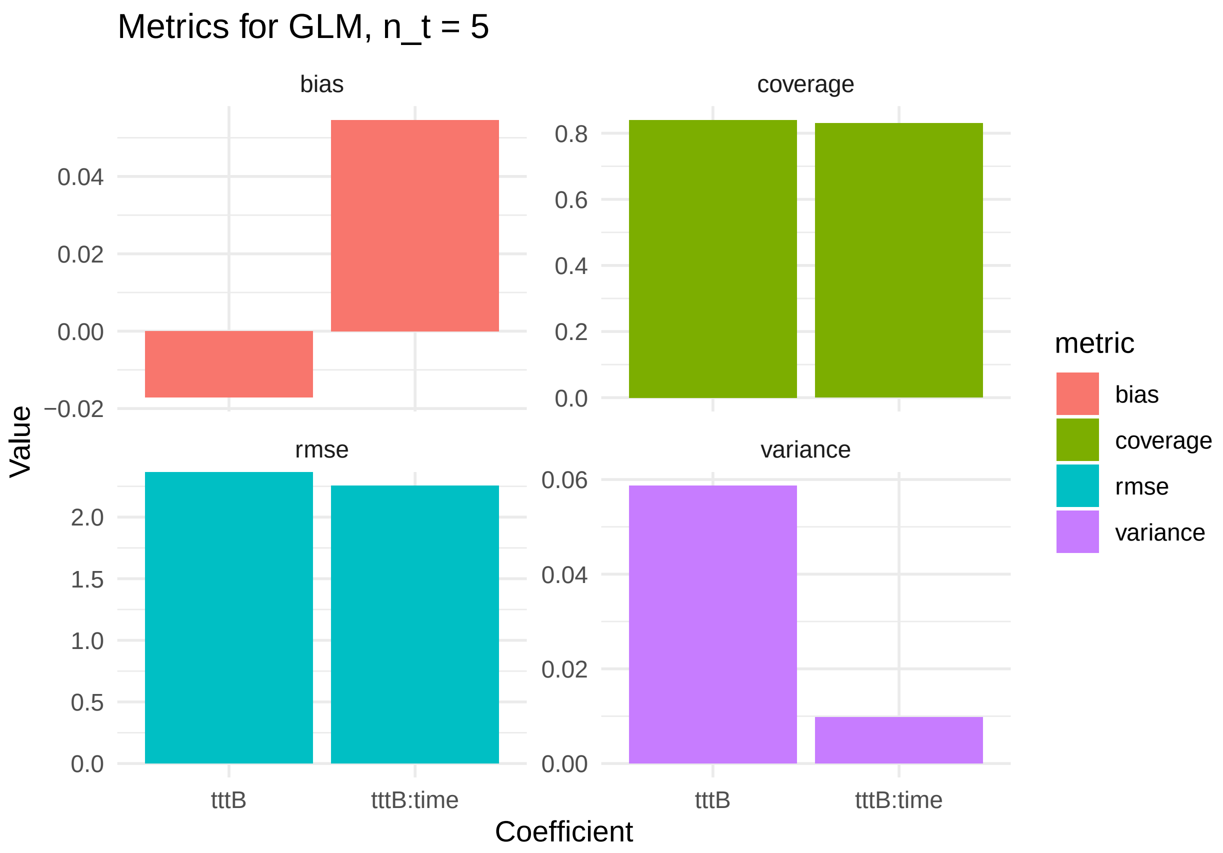
```r
print(summary_glmer_10)
```

```
## # A tibble: 2 x 5
##   term         bias variance   rmse coverage
##   <chr>       <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      0.00808 0.0279   Inf         0.9
## 2 tttB:time 0.0496  0.000834   1.15     0.53
```

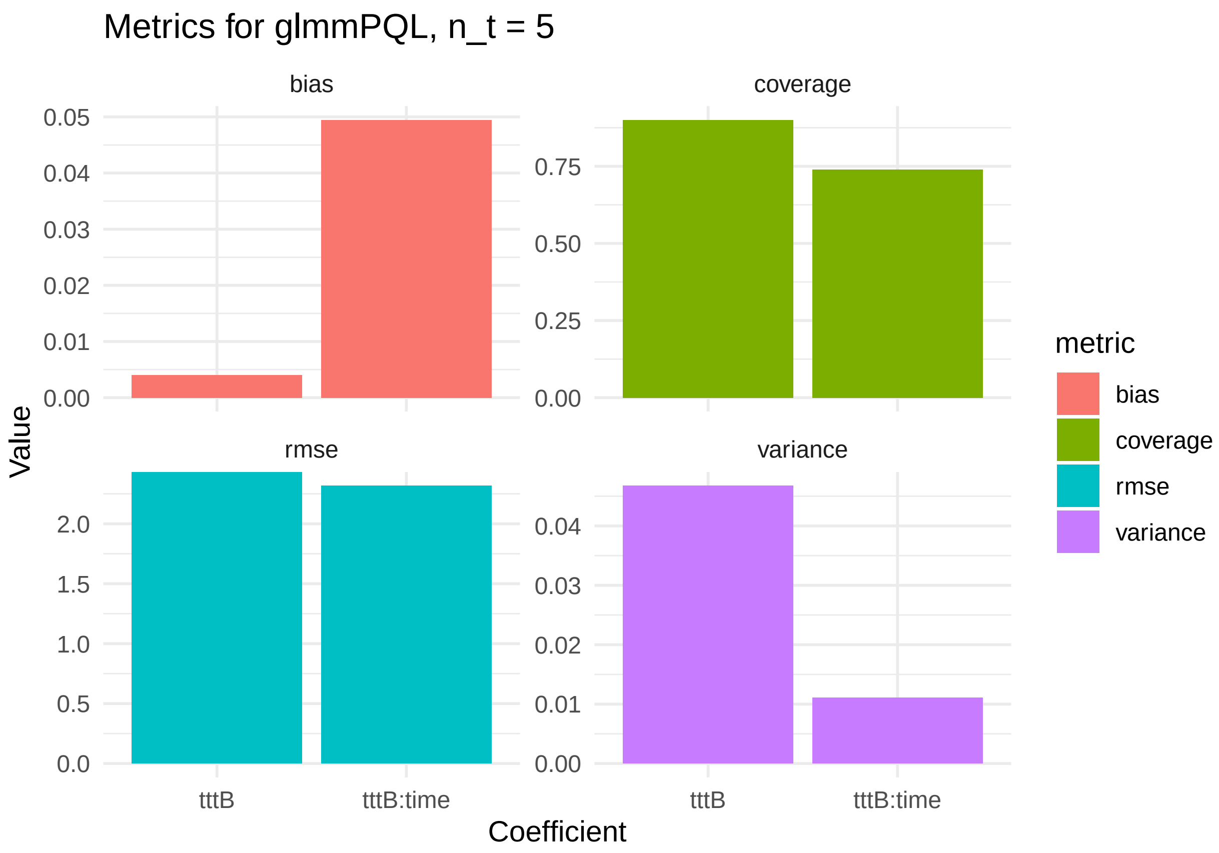
```r
print(summary_AGHQ_10)
```

```
## # A tibble: 2 x 5
##   term          bias variance   rmse coverage
##   <chr>        <dbl>    <dbl>  <dbl>    <dbl>
## 1 tttB      -0.00548  0.0380   Inf         0.84
## 2 tttB:time  0.0523   0.00126   1.26     0.44
```

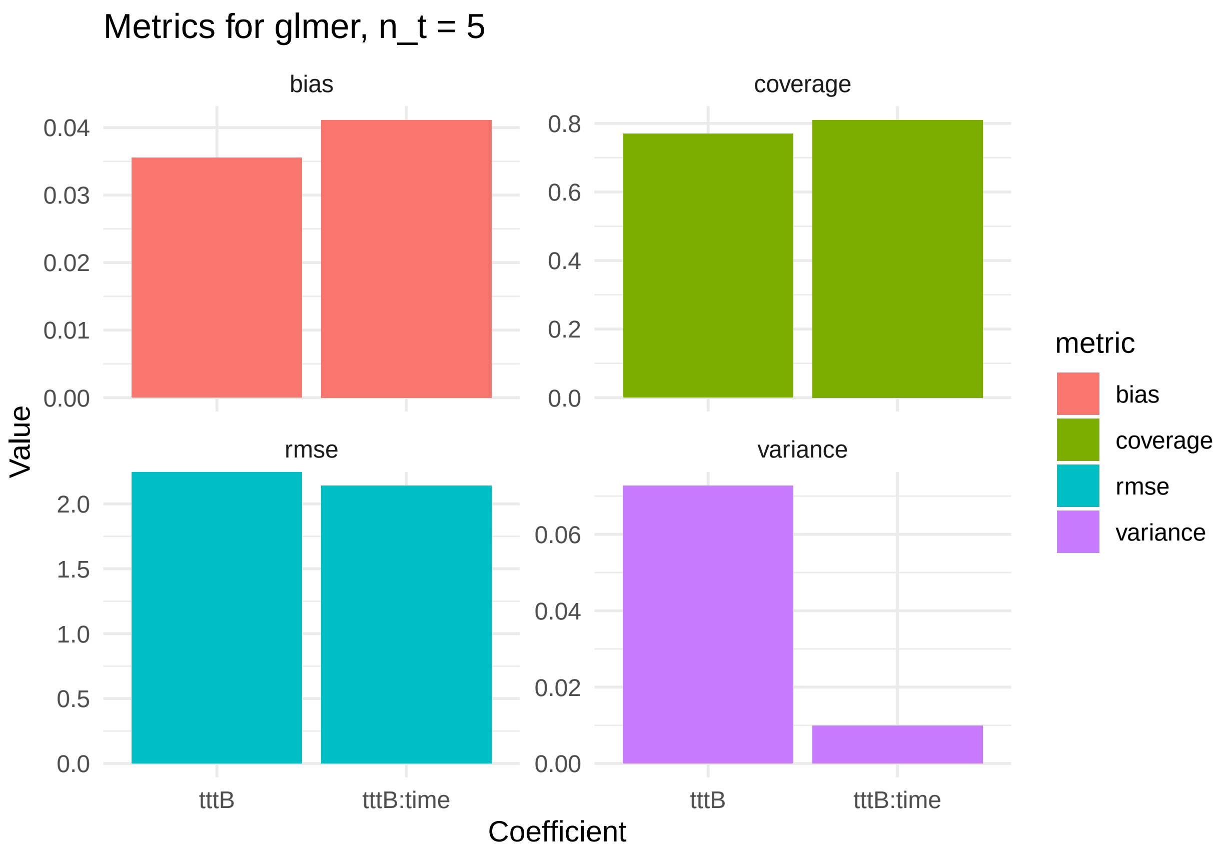
```r
plot_results <- function(summary_df, title) {
  metrics <- summary_df %>% tidyr::pivot_longer(cols = c(bias, variance, rmse, coverage), names_to = "me
  ggplot(metrics, aes(x = term, y = value, fill = metric)) +
    geom_bar(stat = "identity", position = "dodge") +
    theme_minimal() +
    labs(title = title, y = "Value", x = "Coefficient") +
    facet_wrap(~ metric, scales = "free_y")
}

plot_results(summary_glm_5, "Metrics for GLM, n_t = 5")
```



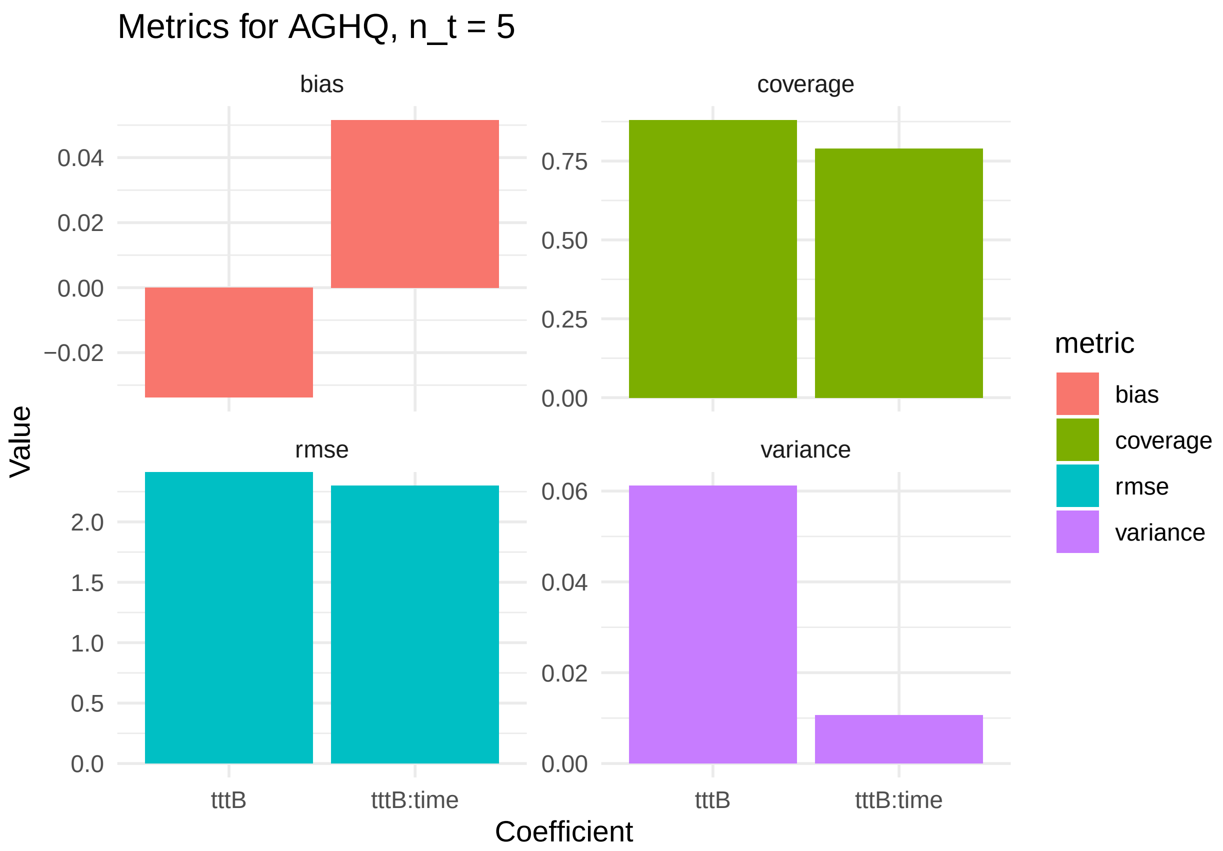
```r
plot_results(summary_glmmPQL_5, "Metrics for glmmPQL, n_t = 5")
```



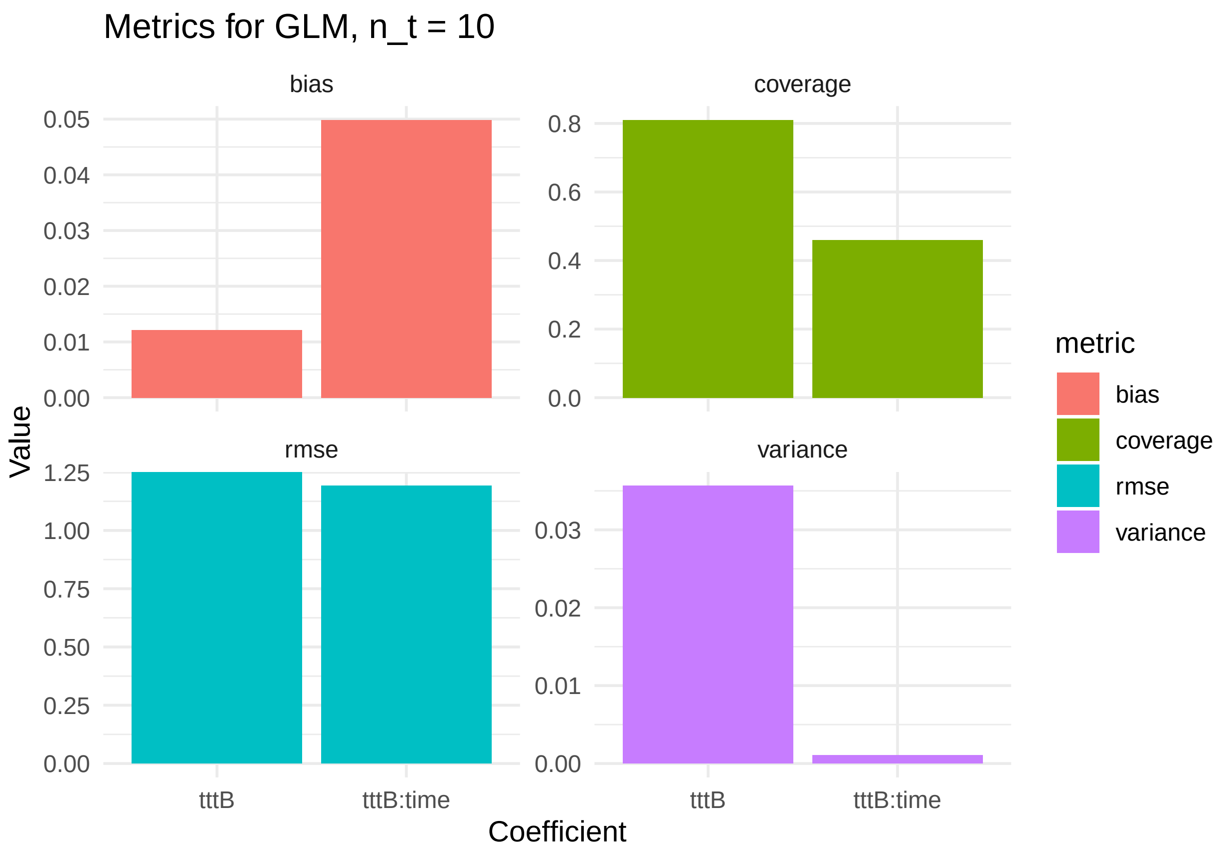
```r
plot_results(summary_glmer_5, "Metrics for glmer, n_t = 5")
```



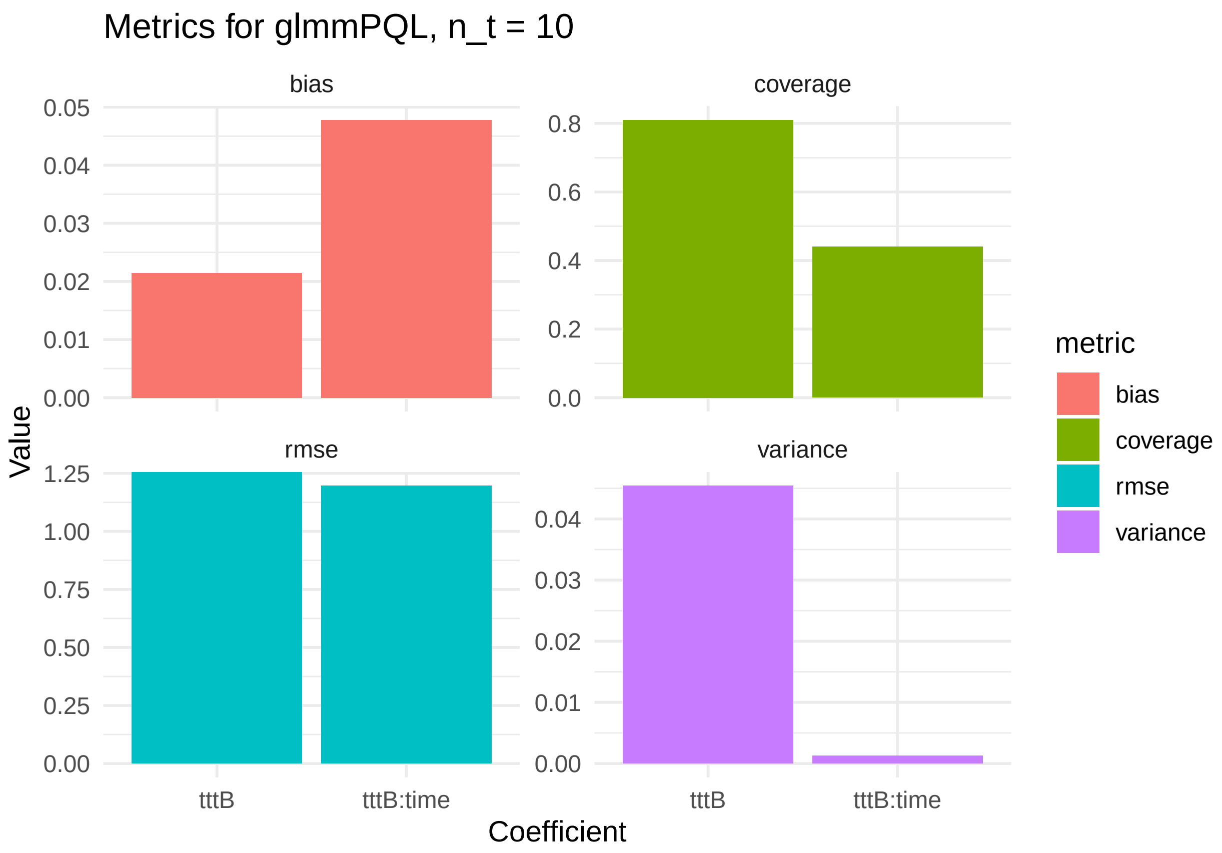
```r
plot_results(summary_AGHQ_5, "Metrics for AGHQ, n_t = 5")
```



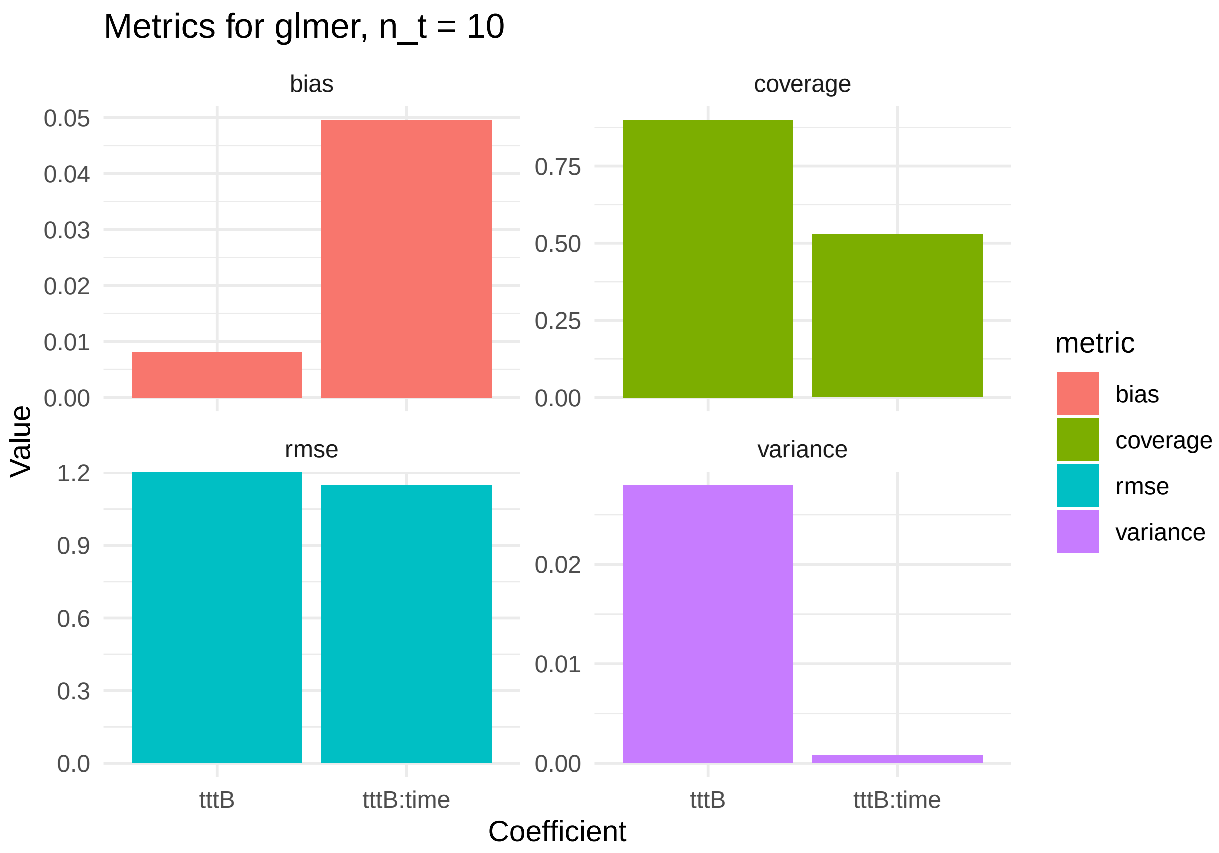
```r
plot_results(summary_glm_10, "Metrics for GLM, n_t = 10")
```



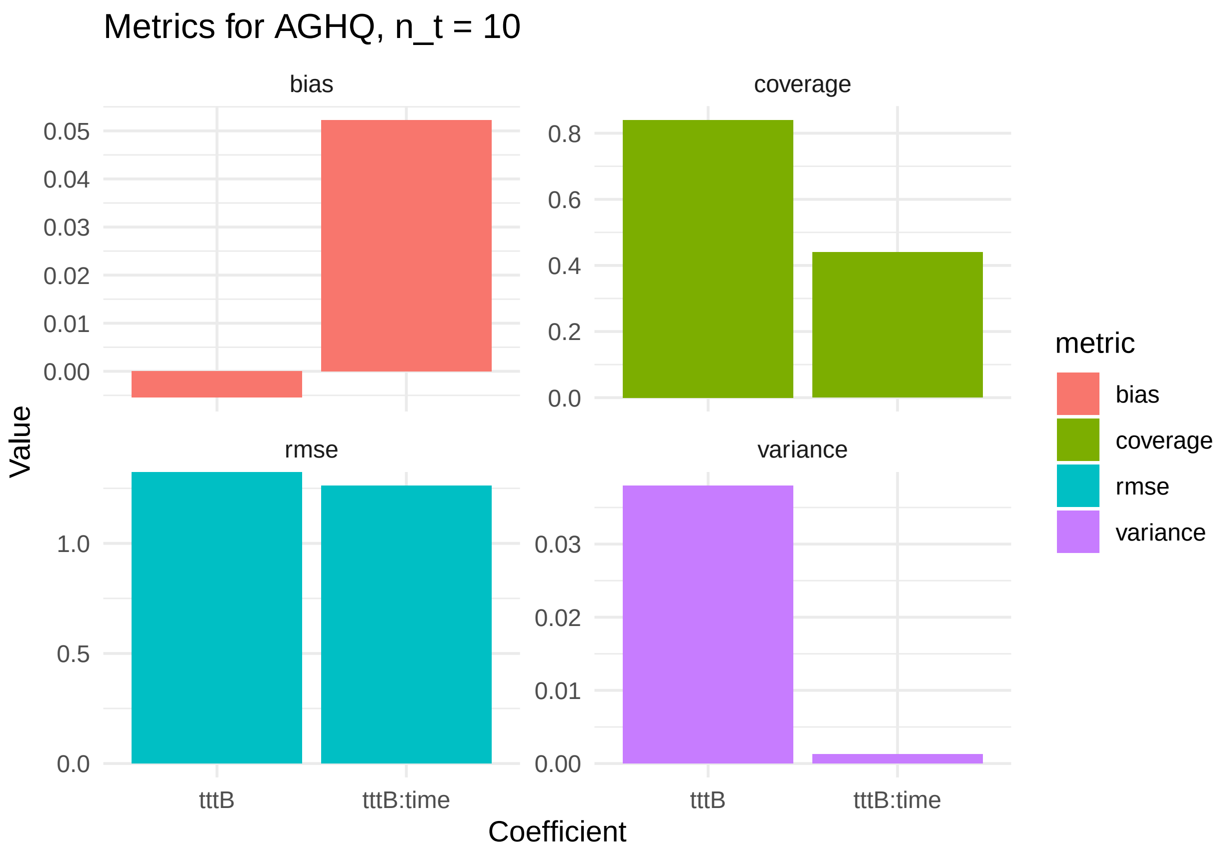
```r
plot_results(summary_glmmPQL_10, "Metrics for glmmPQL, n_t = 10")
```



```r
plot_results(summary_glmer_10, "Metrics for glmer, n_t = 10")
```



```r
plot_results(summary_AGHQ_10, "Metrics for AGHQ, n_t = 10")
```


Metrics for AGHQ, n_t = 10
bias
coverage
rmse
variance
0.05
0.04
0.03
0.02
0.01
0.00
0.8
0.6
0.4
0.2
0.0
1.0
0.5
0.0
0.03
0.02
0.01
0.00
Value
tttB
tttB:time
tttB
tttB:time
Coefficient
metric
bias
coverage
rmse
variance