Final Project

2024-12-17

Load Required Packages

```
library(lme4)
```

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

```
library(brms)
```

```
## Warning: package 'brms' was built under R version 4.4.2
```

```
## Loading required package: Rcpp
```

```
## Loading 'brms' package (version 2.22.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
##
## Attaching package: 'brms'
```

```
## The following object is masked from 'package:lme4':
##
##     ngrps
```

```
## The following object is masked from 'package:stats':
##
##     ar
```

```
library(ggplot2)
library(glmmTMB)
```

```
##
## Attaching package: 'glmmTMB'
```

```
## The following object is masked from 'package:brms':
##
##     lognormal
```

```r
library(broom.mixed)
library(performance)
library(DHARMa)
```

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

```r
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
```

```r
library(ggeffects)
library(splines)
```

Load the Dataset

```r
# Load the VerbAgg dataset
data("VerbAgg", package = "lme4")
```

```r
# Display structure of the dataset
str(VerbAgg)
```

```
## 'data.frame':    7584 obs. of  9 variables:
##  $ Anger : int  20 11 17 21 17 21 39 21 24 16 ...
##  $ Gender: Factor w/ 2 levels "F","M": 2 2 1 1 1 1 1 1 1 1 ...
##  $ item  : Factor w/ 24 levels "S1WantCurse",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ resp  : Ord.factor w/ 3 levels "no"<"perhaps"<..: 1 1 2 2 2 3 3 1 1 3 ...
##  $ id    : Factor w/ 316 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ btype : Factor w/ 3 levels "curse","scold",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ situ  : Factor w/ 2 levels "other","self": 1 1 1 1 1 1 1 1 1 1 ...
##  $ mode  : Factor w/ 2 levels "want","do": 1 1 1 1 1 1 1 1 1 1 ...
##  $ r2    : Factor w/ 2 levels "N","Y": 1 1 2 2 2 2 2 1 1 2 ...
```

```r
# Summarize the dataset
summary(VerbAgg)
```

```
##      Anger    Gender            item           resp            id
##  Min.   :11   F:5832   S1WantCurse: 316   no     :3973   1      :  24
##  1st Qu.:17   M:1752   S1WantScold: 316   perhaps:2081   2      :  24
##  Median :19            S1WantShout: 316   yes    :1530   3      :  24
```

```
##  Mean   :20            S2WantCurse: 316                 4      :  24
##  3rd Qu.:23            S2WantScold: 316                 5      :  24
##  Max.   :39            S2WantShout: 316                 6      :  24
##                        (Other)    :5688                 (Other):7440
##    btype         situ        mode      r2
##  curse:2528   other:3792   want:3792   N:3973
##  scold:2528   self :3792   do  :3792   Y:3611
##  shout:2528
##
##
##
##
```

Description of the Questions

Objective of Data Collection: The VerbAgg dataset was collected to investigate the determinants of verbal aggression in hypothetical scenarios. Specifically, the study aimed to understand how individual characteristics, situational contexts, and behavior types influence the likelihood of aggressive responses (`r2`):

Demographic Influences: Does `gender` affect the likelihood of verbal aggression? Are males or females more likely to respond aggressively? **Behavioral Context**: How does the type of aggressive behavior (`btype`, e.g., cursing, scolding, shouting) influence the likelihood of a participant exhibiting aggression? **Situational Context** (`situ`): Are participants more likely to respond aggressively when the situation involves themselves (`self`) compared to others (`other`)? **Mode of Response** (`mode`): Is there a difference between participants' hypothetical desires (what they want to do) and their actual behaviors (what they would do) in terms of aggression? **Inter-Individual Variability**: Is there substantial variability in aggression tendencies between individuals (`id`), and can this variability be captured through random effects?

Modeling Approach and Methods

To address these questions, the original data collectors used a **Generalized Linear Mixed Model (GLMM)** with the following components:

Response Variable:

`r2`: A binary indicator of whether the participant responded aggressively (`Y` for Yes, `N` for No).

Fixed Effects:

`gender`: A demographic predictor indicating the participant's gender (male or female). `btype`: Categorical predictor for behavior type (curse, scold, shout). `mode`: Categorical predictor for response mode (want vs. do). `situ`: Categorical predictor for situational context (self vs. other).

Random Effects:

`(1 | id)`: A random intercept for each individual (`id`), accounting for differences in baseline aggression tendencies between participants.

Conditional Distribution:

The response variable `r2` is modeled using a **binomial distribution** (since it is binary) with a **logit link function**.

Model Equation

We specify a Generalized Linear Mixed Model (GLMM) to analyze the likelihood of verbal aggression ($r2 = 1$) as a function of individual and situational predictors, accounting for inter-individual variability. The maximal model is written as:

$$\text{logit}(P(r2 = 1)) = \beta_0 + \beta_1 \cdot \text{gender} + \beta_2 \cdot \text{btype} + \beta_3 \cdot \text{mode} + \beta_4 \cdot \text{situ} + u_{\text{id}}$$

Where: - β_0: Overall intercept. - $\beta_1, \beta_2, \beta_3, \beta_4$: Coefficients for fixed effects (gender, behavior type, mode, situational context). - $u_{\text{id}} \sim \mathcal{N}(0, \sigma^2)$: Random intercept capturing inter-individual variability.

Specification and Justification of Three Packages

For the analysis of the **VerbAgg** dataset, we will use three packages: two Frequentist approaches (`lme4` and `glmmTMB`), and one Bayesian (`brms`).

1. `lme4` (Frequentist Approach)

Description

- `lme4` is a widely used R package for fitting linear and generalized linear mixed models (GLMMs) using maximum likelihood estimation (MLE).

Why Use It?

- **Efficiency**: Uses **Laplace approximation** by default, providing a balance of speed and accuracy for GLMMs.
- **Flexibility**: Supports a wide range of response distributions and link functions (e.g., binomial with logit link for binary responses like `r2`).
- **Reliability**: It is well-documented, robust, and commonly used in mixed-effects modeling.

When to Use

- Fit the maximal Frequentist model to establish baseline results.
- Use it for model comparison and diagnostics (e.g., singularity issues, fixed effect significance).

2. `glmmTMB` (Frequentist Approach)

Description `glmmTMB` is an R package for fitting generalized linear mixed models using the Template Model Builder (TMB) framework. It extends the capabilities of `lme4` by supporting additional distributions, zero-inflation, and overdispersion.

Why Use It?

- It uses **Laplace approximation** by default, similar to `lme4`.
- **Flexibility**: Supports a broader range of response distributions, including:
  - Negative Binomial (for overdispersed count data).
  - Beta and zero-inflated models.
  - Gaussian, Binomial, Poisson, and more.
- **Advanced Modeling**: Allows for zero-inflation or hurdle models to account for datasets with excessive zeros.
- **Stability**: Efficient optimization routines make it robust for larger datasets or models with more parameters.
- **Random Effects**: Supports more complex structures, including nested and crossed random effects.

When to Use

- Use when the dataset exhibits **overdispersion** or **zero-inflation** (issues that `lme4` cannot directly address).
- Fit complex GLMMs requiring advanced response distributions or mixed-effects structures.
- Use as a fallback when `lme4` runs into singularity or convergence issues, or when diagnostics suggest model inadequacy.
- Particularly useful for ecological, medical, or hierarchical data with excessive zeros or overdispersed counts.

3. `brms` (Bayesian Approach)

Description

- `brms` is an interface to Stan for Bayesian regression models, allowing flexible and user-friendly mixed model fitting.

Why Use It?

- It uses **Markov Chain Monte Carlo (MCMC)** sampling via Stan.
- **Flexibility**: Supports custom priors, complex random effects, and non-standard distributions.
- **Bayesian Inference**: Incorporates uncertainty in parameter estimates through posterior distributions.
- **Diagnostics**: Provides convergence diagnostics (e.g., R-hat values) and posterior predictive checks.
- **Extensibility**: Supports complex models, including multilevel and hierarchical Bayesian models.

When to Use

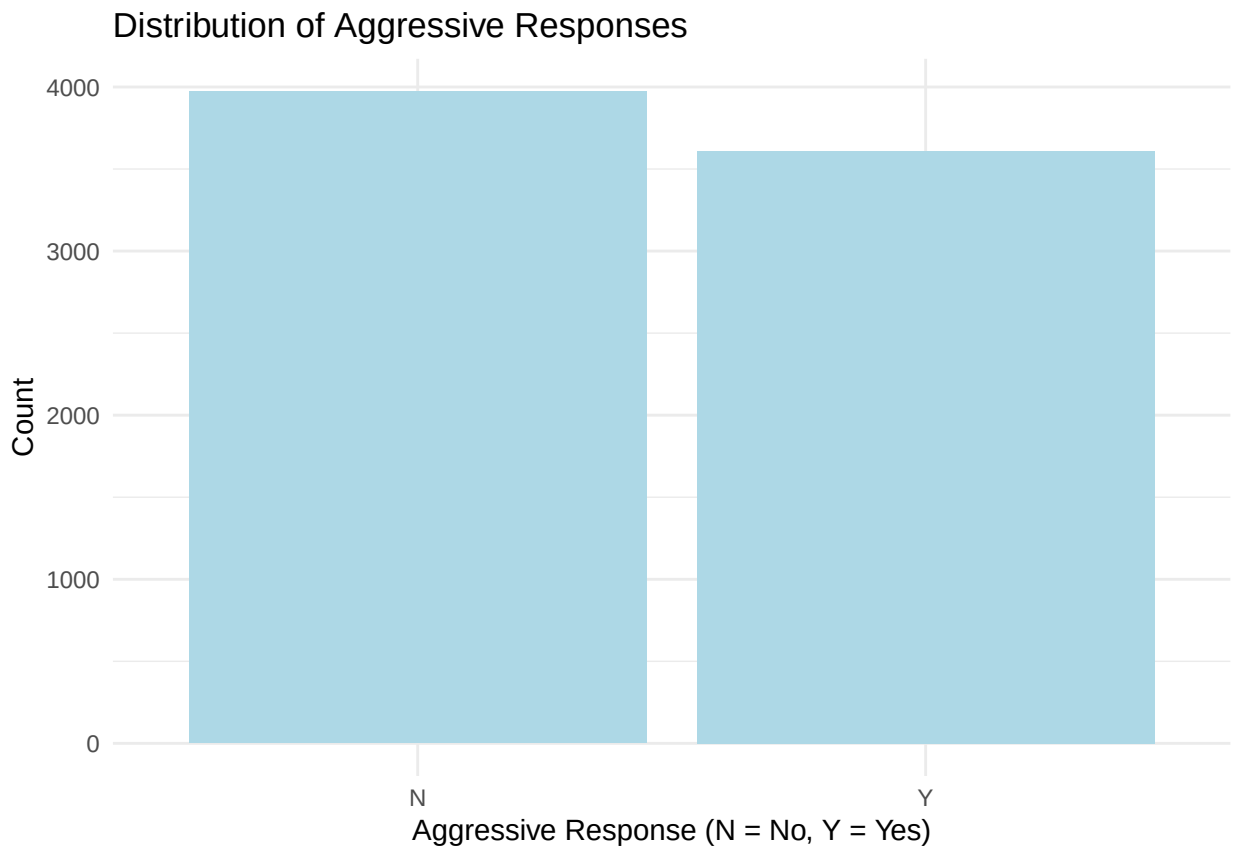
- Use to refine the analysis with prior knowledge (e.g., weakly informative priors).
- Address model uncertainty or convergence issues in the Frequentist approach.

Summary

By combining `lme4`, `glmmTMB`, and `brms`, we ensure robust modeling across frequentist and Bayesian paradigms. This approach enables flexibility to handle complex data structures (e.g., overdispersion, zero-inflation), provides rigorous diagnostics, and allows for uncertainty quantification, offering a comprehensive analytical framework.

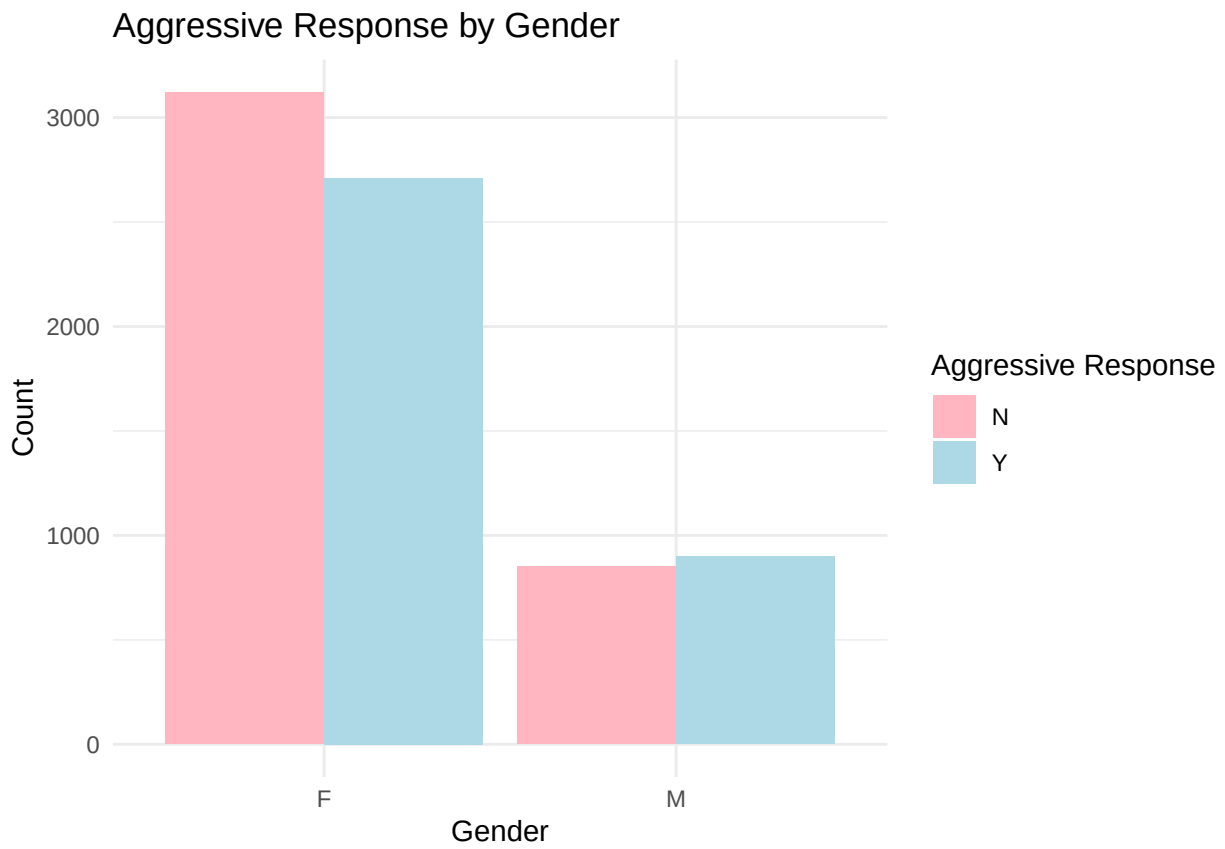
Exploratory Plots

```
# Plot 1: Distribution of Aggressive Responses
ggplot(VerbAgg, aes(x = r2)) +
  geom_bar(fill = "lightblue") +
  labs(title = "Distribution of Aggressive Responses",
       x = "Aggressive Response (N = No, Y = Yes)",
       y = "Count") +
  theme_minimal()
```



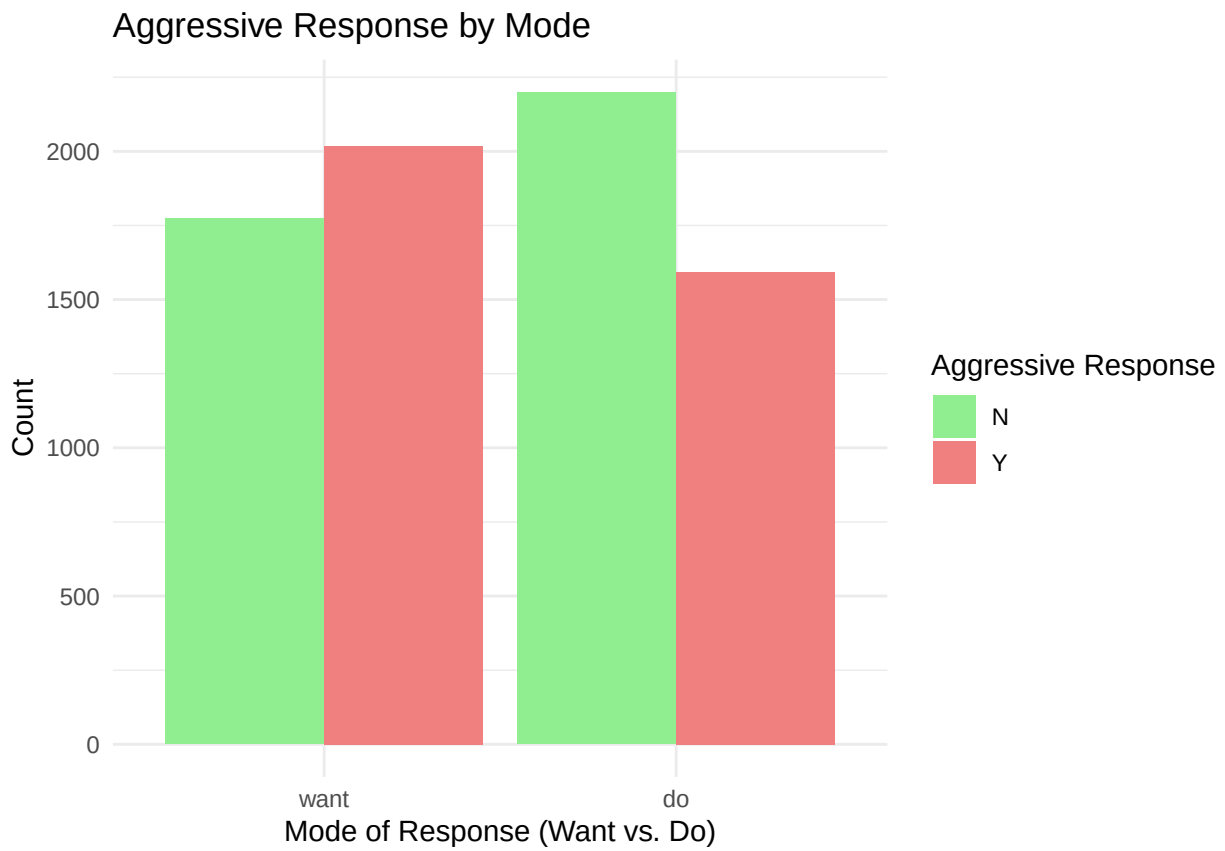
The data is relatively balanced between the two categories, with slightly more non-aggressive responses (`N`) compared to aggressive responses (`Y`), which is helpful for binary classification modeling.

```
# Plot 2: Aggressive Response by Gender
ggplot(VerbAgg, aes(x = Gender, fill = r2)) +
  geom_bar(position = "dodge") +
  labs(title = "Aggressive Response by Gender",
       x = "Gender",
       y = "Count",
       fill = "Aggressive Response") +
  scale_fill_manual(values = c("N" = "lightpink", "Y" = "lightblue")) +
  theme_minimal()
```



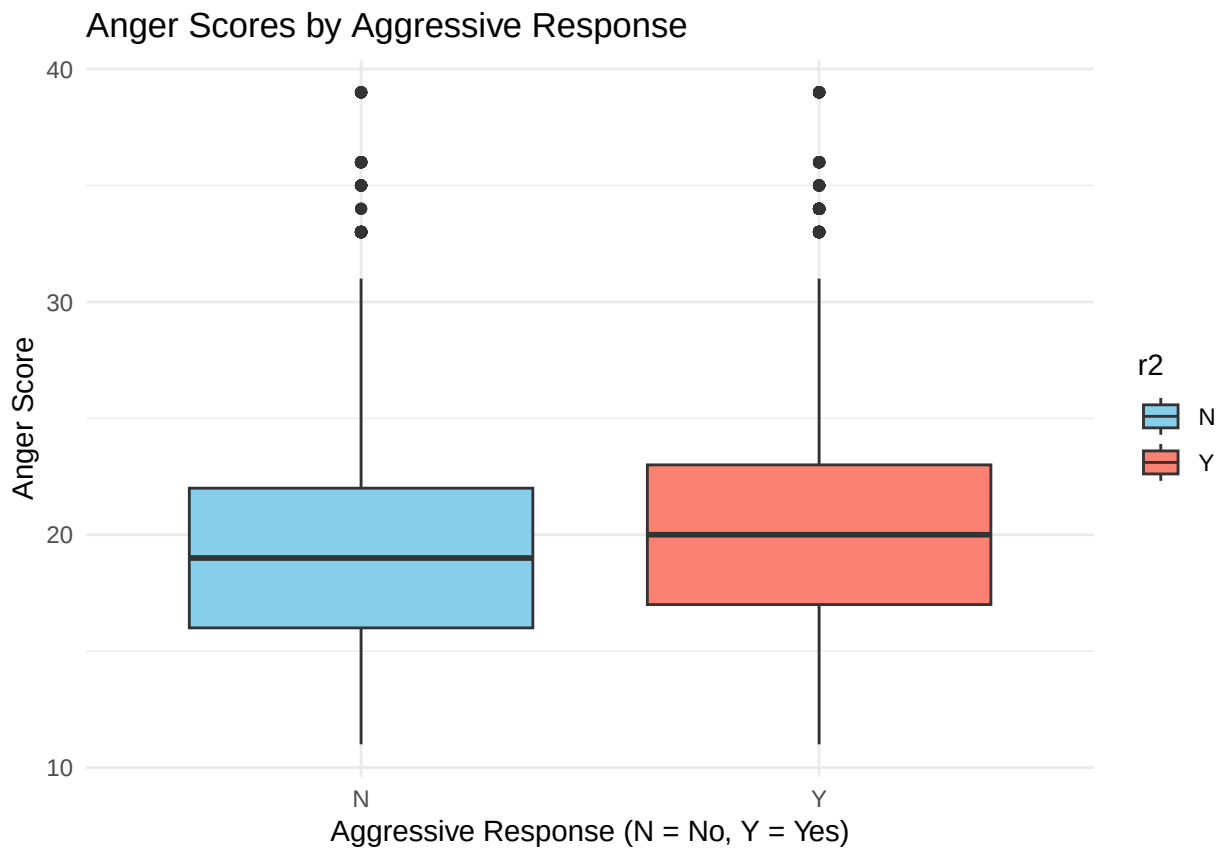
Females (`F`) have a higher number of both aggressive (`Y`) and non-aggressive (`N`) responses compared to males (`M`). Proportionally, the difference between `N` and `Y` responses appears larger for males (`M`).

```
# Plot 3: Aggressive Response by Mode
ggplot(VerbAgg, aes(x = mode, fill = r2)) +
  geom_bar(position = "dodge") +
  labs(title = "Aggressive Response by Mode",
       x = "Mode of Response (Want vs. Do)",
       y = "Count",
       fill = "Aggressive Response") +
  scale_fill_manual(values = c("N" = "lightgreen", "Y" = "lightcoral")) +
  theme_minimal()
```



In the "want" mode, there are more aggressive (Y) responses than non-aggressive (N) responses. In the "do" mode, non-aggressive responses (N) are more frequent than aggressive ones (Y).

```
# Plot 4: Anger Scores by Aggressive Response
ggplot(VerbAgg, aes(x = r2, y = Anger, fill = r2)) +
  geom_boxplot() +
  labs(title = "Anger Scores by Aggressive Response",
       x = "Aggressive Response (N = No, Y = Yes)",
       y = "Anger Score") +
  scale_fill_manual(values = c("N" = "skyblue", "Y" = "salmon")) +
  theme_minimal()
```



Median anger scores are slightly higher for aggressive (`Y`) responses than for non-aggressive (`N`) responses. There is considerable overlap in the anger score distributions between the two groups.

Fit the Models

```
# Fit the GLMM using lme4
model_lme4 <- glmer(r2 ~ Gender + btype + mode + situ + (1 | id),
                    data = VerbAgg, family = binomial(link = "logit"))

# Summarize the model
summary(model_lme4)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: r2 ~ Gender + btype + mode + situ + (1 | id)
##    Data: VerbAgg
##
##      AIC      BIC   logLik deviance df.resid
##   8249.4   8298.0  -4117.7   8235.4     7577
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -7.6060 -0.6357 -0.1974  0.6319  9.3777
##
```

```
## Random effects:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 1.777    1.333
## Number of obs: 7584, groups:  id, 316
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.67366    0.11096  15.084   <2e-16 ***
## GenderM      0.30206    0.19049   1.586    0.113
## btypescold  -1.05524    0.06899 -15.295   <2e-16 ***
## btypeshout  -2.04220    0.07455 -27.394   <2e-16 ***
## modedo      -0.67158    0.05686 -11.812   <2e-16 ***
## situself    -1.02789    0.05773 -17.805   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) GendrM btypsc btypsh modedo
## GenderM     -0.390
## btypescold -0.365 -0.008
## btypeshout -0.386 -0.013  0.528
## modedo     -0.303 -0.006  0.059  0.105
## situself   -0.321 -0.009  0.089  0.159  0.068
```

```r
# Fit the GLMM using glmmTMB
model_glmmTMB <- glmmTMB(r2 ~ Gender + btype + mode + situ + (1 | id),
                         data = VerbAgg, family = binomial(link = "logit"))

# Summarize the model
summary(model_glmmTMB)
```

```
##  Family: binomial  ( logit )
## Formula:          r2 ~ Gender + btype + mode + situ + (1 | id)
## Data: VerbAgg
##
##      AIC      BIC   logLik deviance df.resid
##   8249.4   8298.0  -4117.7   8235.4     7577
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  id     (Intercept) 1.778    1.333
## Number of obs: 7584, groups:  id, 316
##
## Conditional model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.67420    0.11120  15.055   <2e-16 ***
## GenderM      0.30219    0.19075   1.584    0.113
## btypescold  -1.05562    0.06934 -15.223   <2e-16 ***
## btypeshout  -2.04294    0.07497 -27.251   <2e-16 ***
## modedo      -0.67179    0.05713 -11.760   <2e-16 ***
## situself    -1.02823    0.05802 -17.721   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Specify priors (optional)
priors <- c(
  prior(normal(0, 5), class = "b"),  # Priors for fixed effects
  prior(normal(0, 10), class = "Intercept"),  # Prior for intercept
  prior(exponential(1), class = "sd"))  # Priors for random effects

# Fit the GLMM using brms
model_brms <- brm(r2 ~ Gender + btype + mode + situ + (1 | id),
                  data = VerbAgg, family = bernoulli(link = "logit"),
                  prior = priors, chains = 4, iter = 2000, seed = 123)
```

```
## Compiling Stan program...

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.00177 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 17.7 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 18.645 seconds (Warm-up)
## Chain 1:                13.011 seconds (Sampling)
## Chain 1:                31.656 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000725 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 7.25 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 18.433 seconds (Warm-up)
## Chain 2:                12.971 seconds (Sampling)
## Chain 2:                31.404 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000933 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 9.33 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 21.864 seconds (Warm-up)
## Chain 3:                12.9 seconds (Sampling)
## Chain 3:                34.764 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000999 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 9.99 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

```
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 20.407 seconds (Warm-up)
## Chain 4:                13.411 seconds (Sampling)
## Chain 4:                33.818 seconds (Total)
## Chain 4:
```

```r
# Summarize the model
summary(model_brms)
```

```
##  Family: bernoulli
##   Links: mu = logit
## Formula: r2 ~ Gender + btype + mode + situ + (1 | id)
##    Data: VerbAgg (Number of observations: 7584)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Multilevel Hyperparameters:
## ~id (Number of levels: 316)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     1.35      0.07     1.22     1.48 1.00     1133     1941
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      1.67      0.11     1.46     1.90 1.01      538     1658
## GenderM        0.30      0.20    -0.08     0.69 1.01      554     1102
## btypescold    -1.06      0.07    -1.19    -0.92 1.00     4866     3191
## btypeshout    -2.04      0.08    -2.19    -1.89 1.00     4416     2776
## modedo        -0.67      0.06    -0.79    -0.56 1.00     7261     3069
## situself      -1.03      0.06    -1.14    -0.91 1.00     6354     2930
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Comparison of Fixed Effects

- `GenderM` has a positive coefficient (`~0.30`), suggesting males have slightly higher odds of responding aggressively, though it is not statistically significant (`p > 0.05` or 95% CI contains `0`).
- `btypescold` and `btypeshout` have large negative coefficients (`~-1.06` and `-2.04`, respectively), indicating that scolding and shouting reduce the odds of aggression compared to cursing.
- `modedo` has a negative coefficient (`~-0.67`), suggesting that participants are less likely to act aggressively when reporting what they "would do", compared to "want to do."
- `situself` has a negative coefficient (`~-1.03`), indicating participants are less aggressive in situations involving themselves than others.

Random Effects

Compare the **standard deviation** of the random intercept for `id` (individuals):

- `lme4`: Std.Dev = 1.333.
- `glmmTMB`: Std.Dev = 1.333.
- `brms`: Std.Dev = 1.35 (posterior mean with credible interval [1.22, 1.48]).

The random effects are very similar across the three models, indicating that the variability in baseline aggression among individuals is consistent.

Goodness-of-Fit Metrics

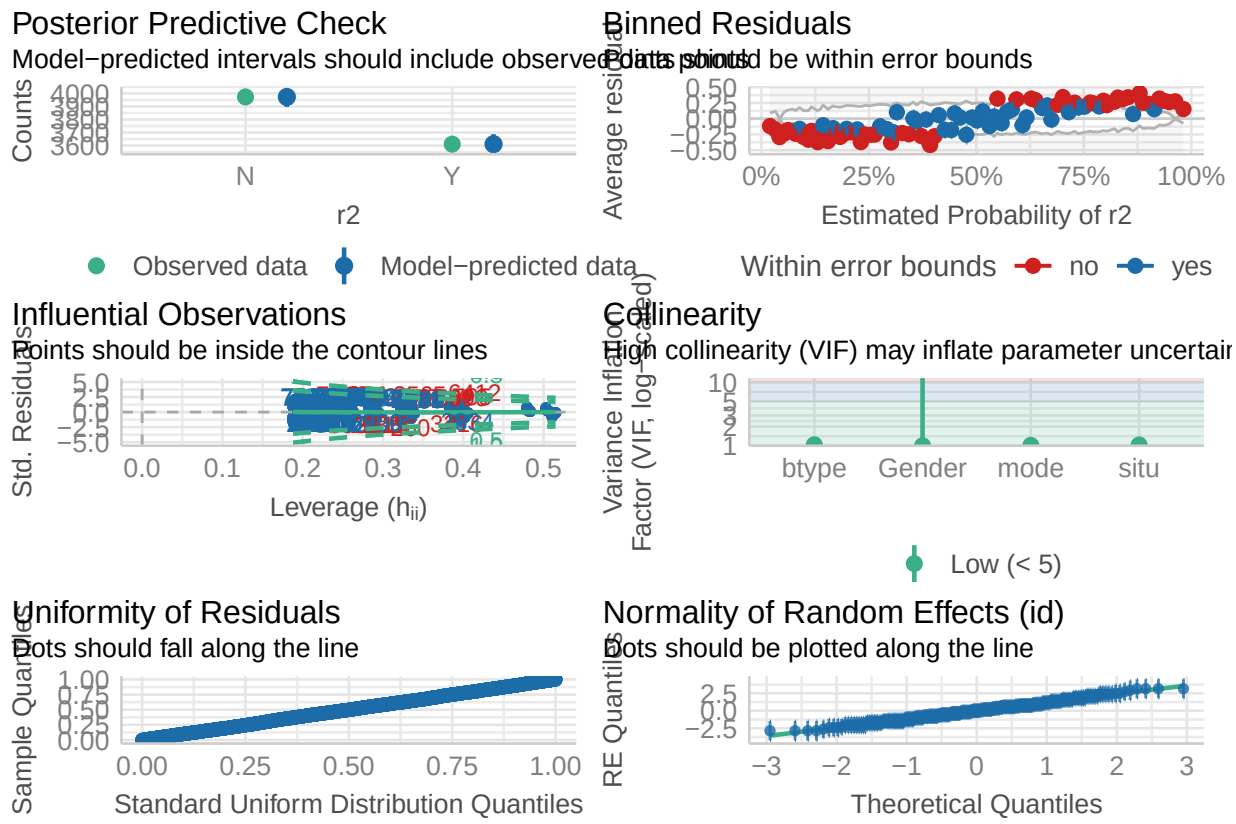
Compare **AIC** and **BIC** values from `lme4` and `glmmTMB` (Bayesian models like brms do not provide AIC/BIC by default):

`lme4`: AIC = 8249.4, BIC = 8298.0. `glmmTMB`: AIC = 8249.4, BIC = 8298.0.

Both frequentist models fit the data equally well based on AIC/BIC.

Diagnostics for Models

```
# Check model performance
check_model(model_lme4)
```

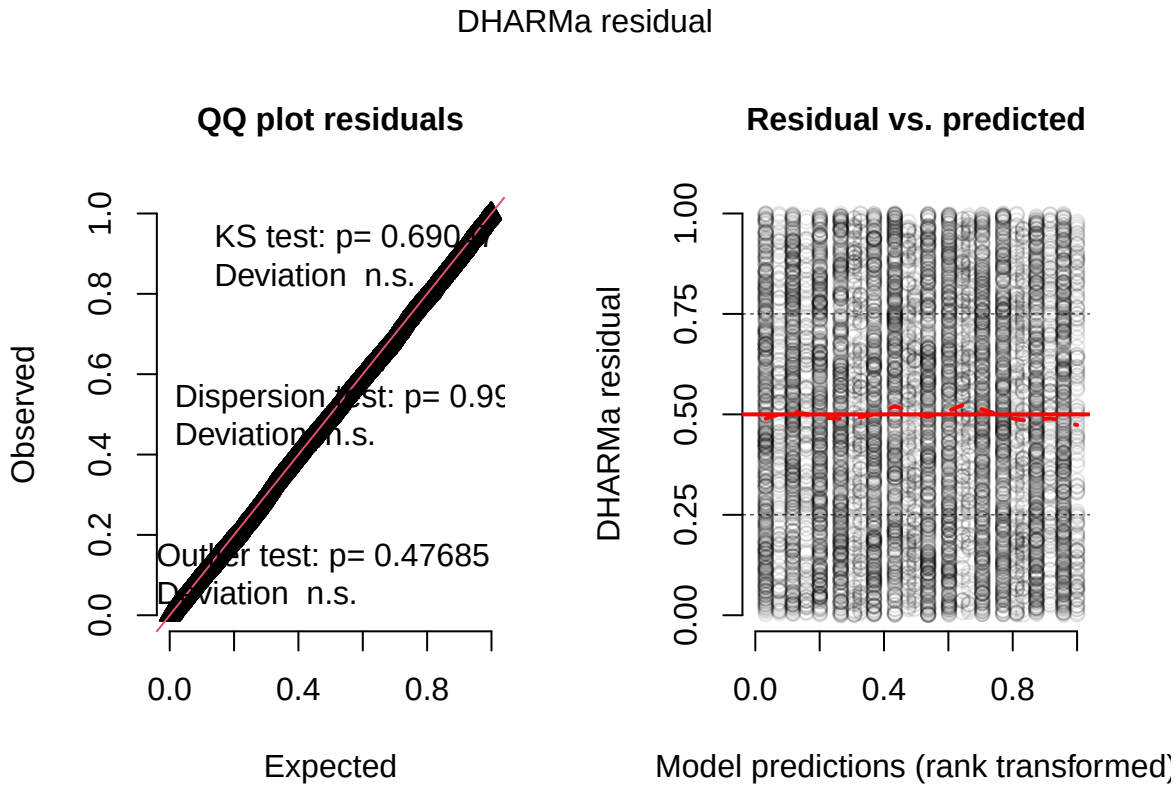



```
# Residual diagnostics
check_residuals(model_lme4)
```

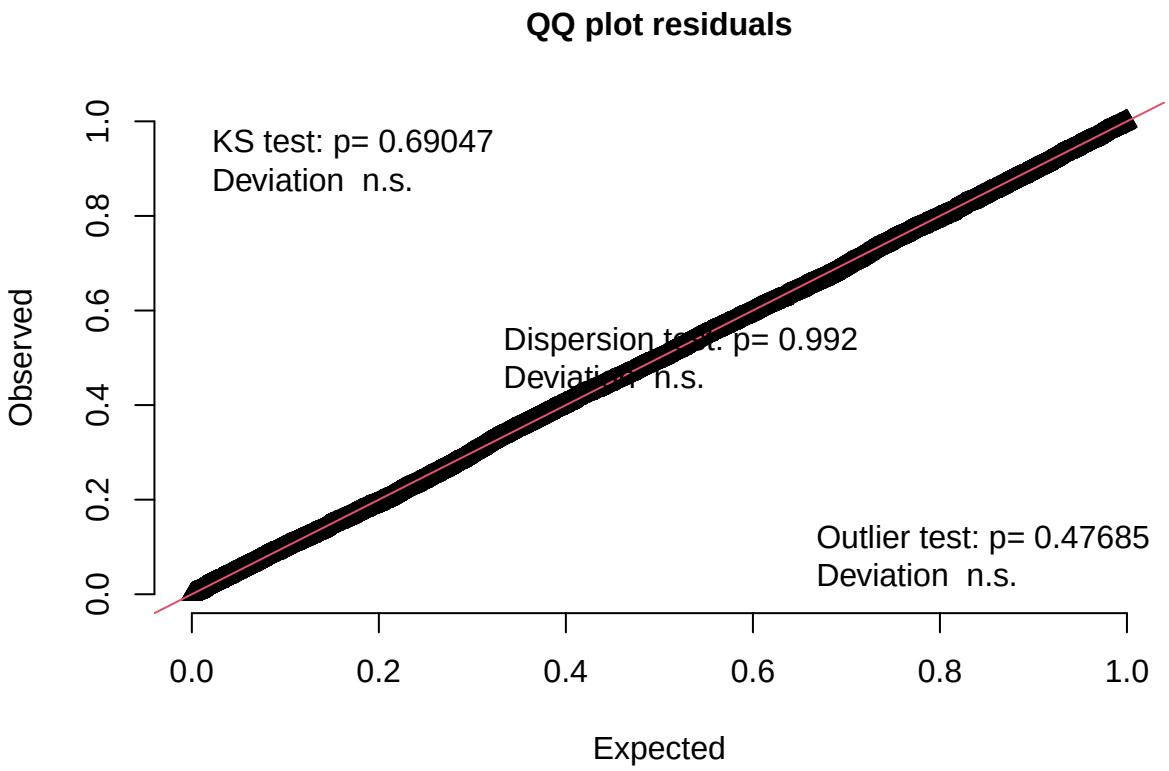
```
## OK: Simulated residuals appear as uniformly distributed (p = 0.690).
```

Posterior Predictive Check: Model-predicted intervals cover observed data points, indicating a good fit. **Binned Residuals:** Most residuals fall within the error bounds, but there are some deviations at higher probabilities of aggression. This may indicate slight misspecification at the extremes. **Influential Observations:** Points outside the contour lines suggest potential influential observations. These should be further investigated for their impact on the model. **Collinearity:** Variance Inflation Factor (`VIF`) values are low (`< 5`), indicating no significant collinearity among predictors. **Uniformity of Residuals:** Residuals align well with the expected uniform distribution, suggesting a good model fit. **Normality of Random Effects:** The random effects (`id`) follow the theoretical quantiles closely, supporting the assumption of normality.

```r
# Simulate residuals for DHARMa diagnostics
res_lme4 <- simulateResiduals(fittedModel = model_lme4)

# Plot residual diagnostics
plot(res_lme4)
```

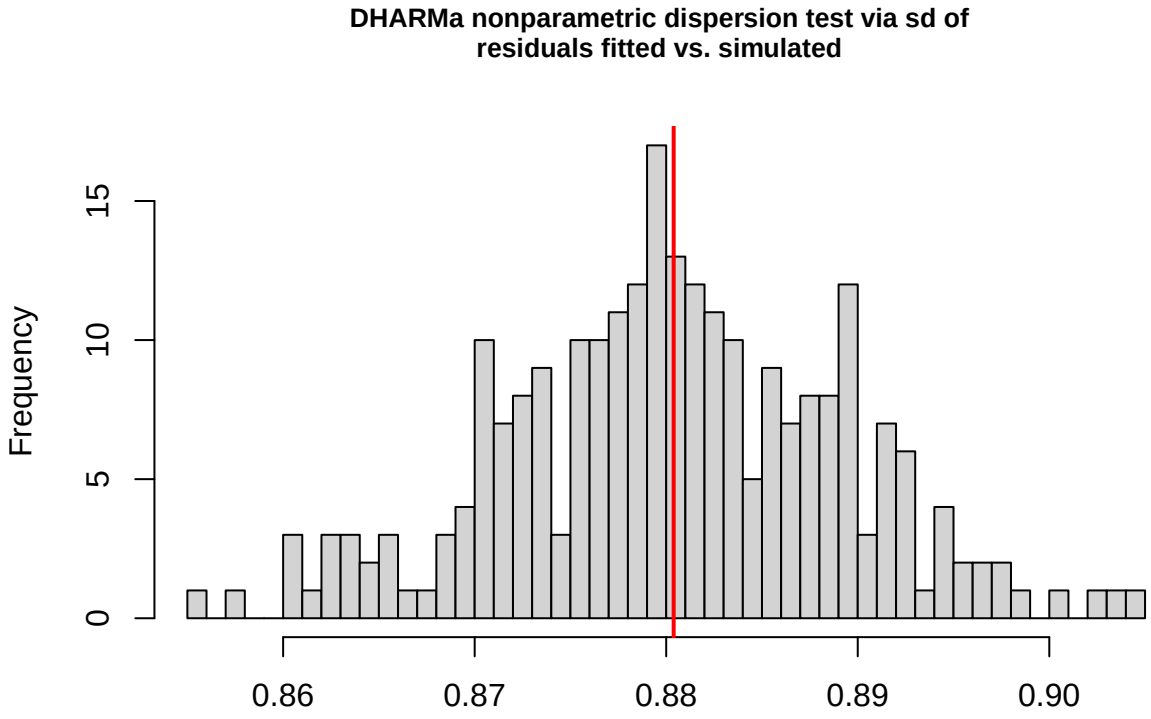



```r
# Test residuals for uniformity
testUniformity(res_lme4)
```



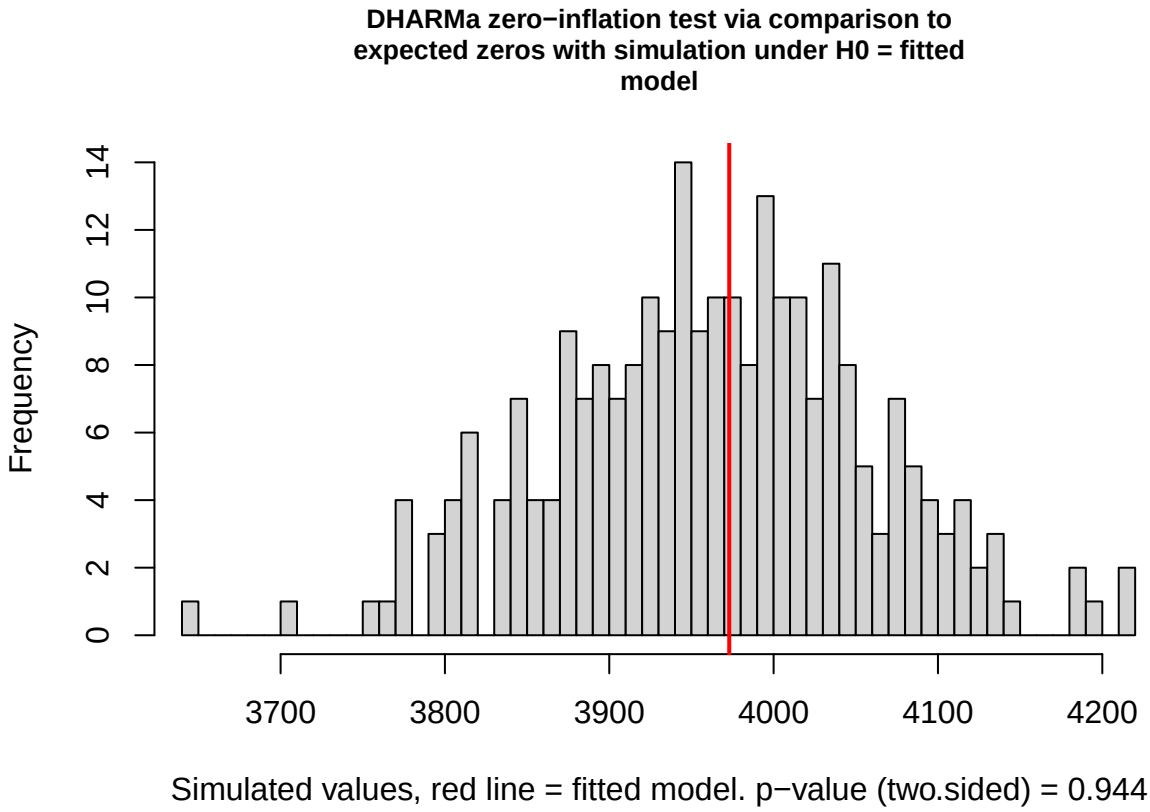

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  simulationOutput$scaledResiduals
## D = 0.0081807, p-value = 0.6905
## alternative hypothesis: two-sided
```

```
# Test for overdispersion
testDispersion(res_lme4)
```




```
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted vs.
##  simulated
##
## data:  simulationOutput
## dispersion = 1, p-value = 0.992
## alternative hypothesis: two.sided
```

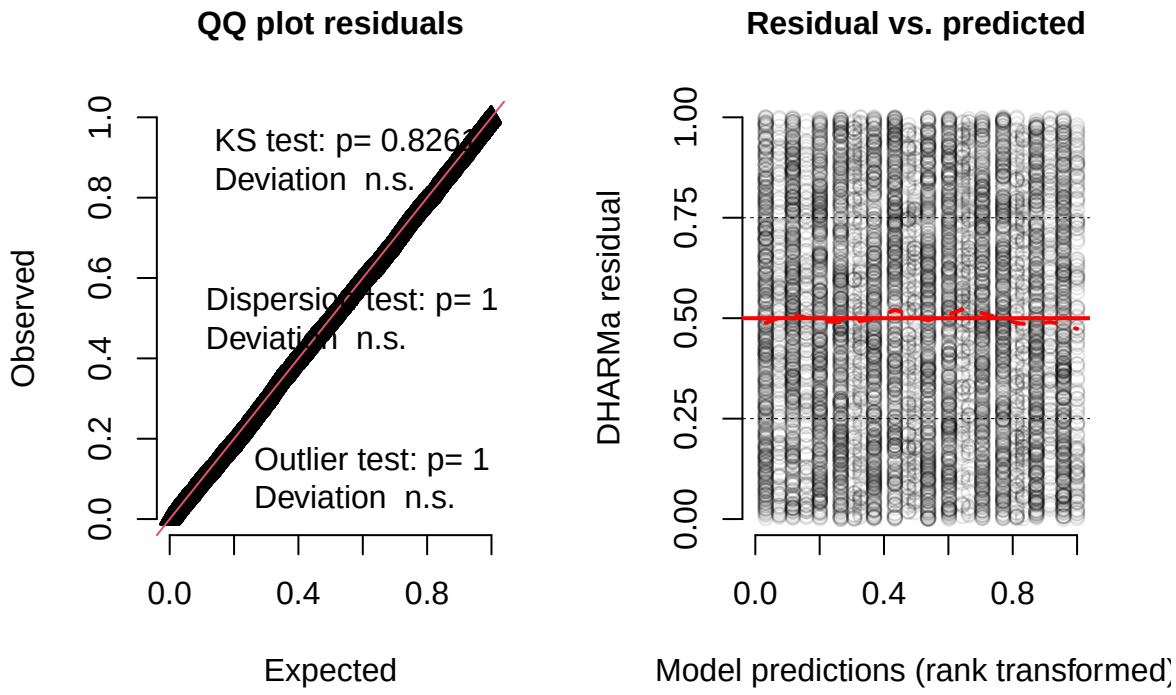
```
# Test for zero inflation
testZeroInflation(res_lme4)
```



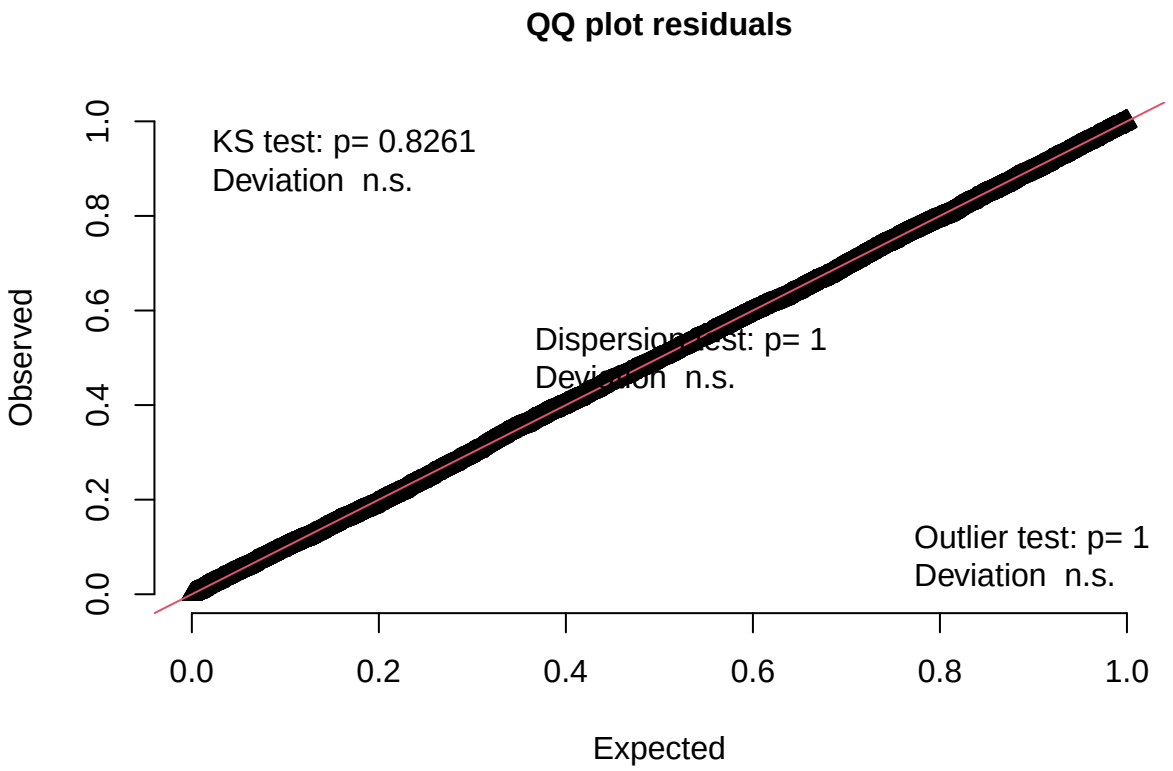

```
##
##  DHARMa zero-inflation test via comparison to expected zeros with
##  simulation under H0 = fitted model
##
## data:  simulationOutput
## ratioObsSim = 1.0025, p-value = 0.944
## alternative hypothesis: two.sided
```

QQ Plot of Residuals: The residuals follow the 1:1 line closely, indicating a good fit. The Kolmogorov-Smirnov (KS) test, dispersion test, and outlier test all show non-significant results (`p > 0.05`), confirming no substantial deviations from model assumptions. **Residuals vs. Predicted Values:** The residuals are evenly distributed around 0 across predicted probabilities, with no obvious patterns or trends. This suggests the model captures the data well.

```r
# Simulate residuals for DHARMa diagnostics
res_glmmTMB <- simulateResiduals(fittedModel = model_glmmTMB)

# Plot residual diagnostics
plot(res_glmmTMB)
```

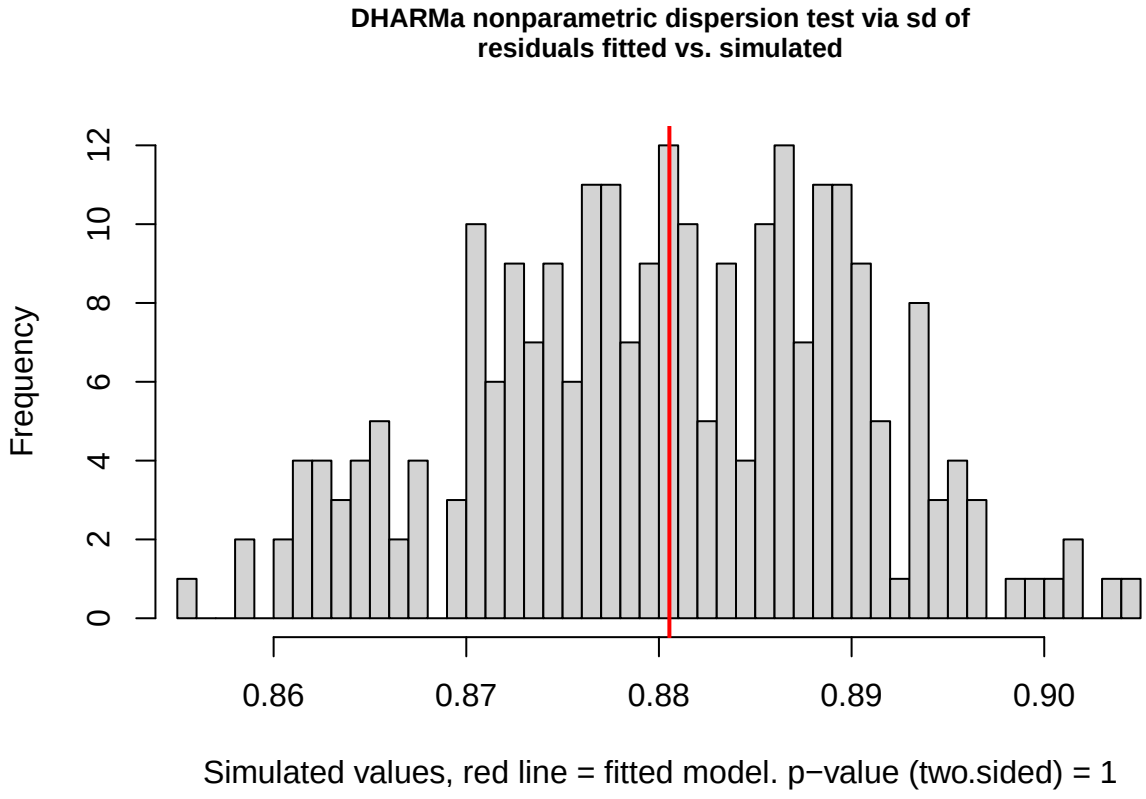



```
# Test residuals for uniformity
testUniformity(res_glmmTMB)
```



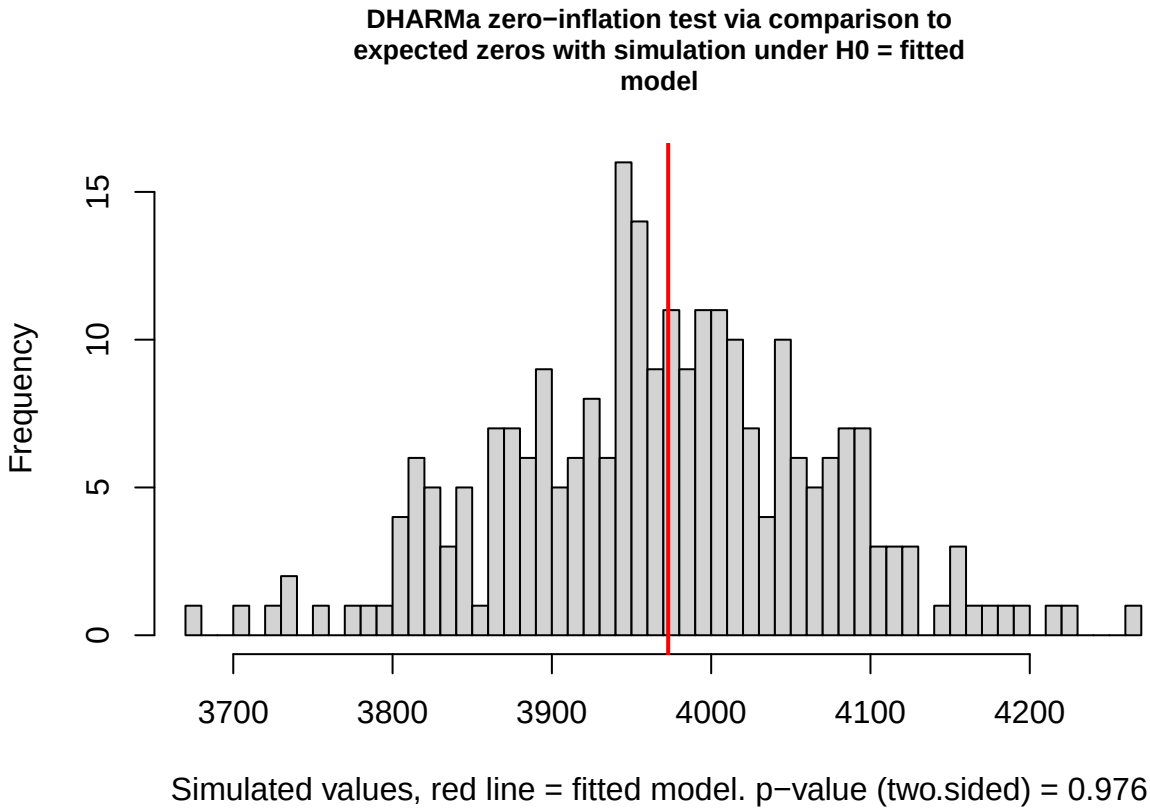

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  simulationOutput$scaledResiduals
## D = 0.0072041, p-value = 0.8261
## alternative hypothesis: two-sided
```

```
# Test for overdispersion
testDispersion(res_glmmTMB)
```




```
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted vs.
##  simulated
##
## data:  simulationOutput
## dispersion = 1.0002, p-value = 1
## alternative hypothesis: two.sided
```

```
# Test for zero inflation
testZeroInflation(res_glmmTMB)
```

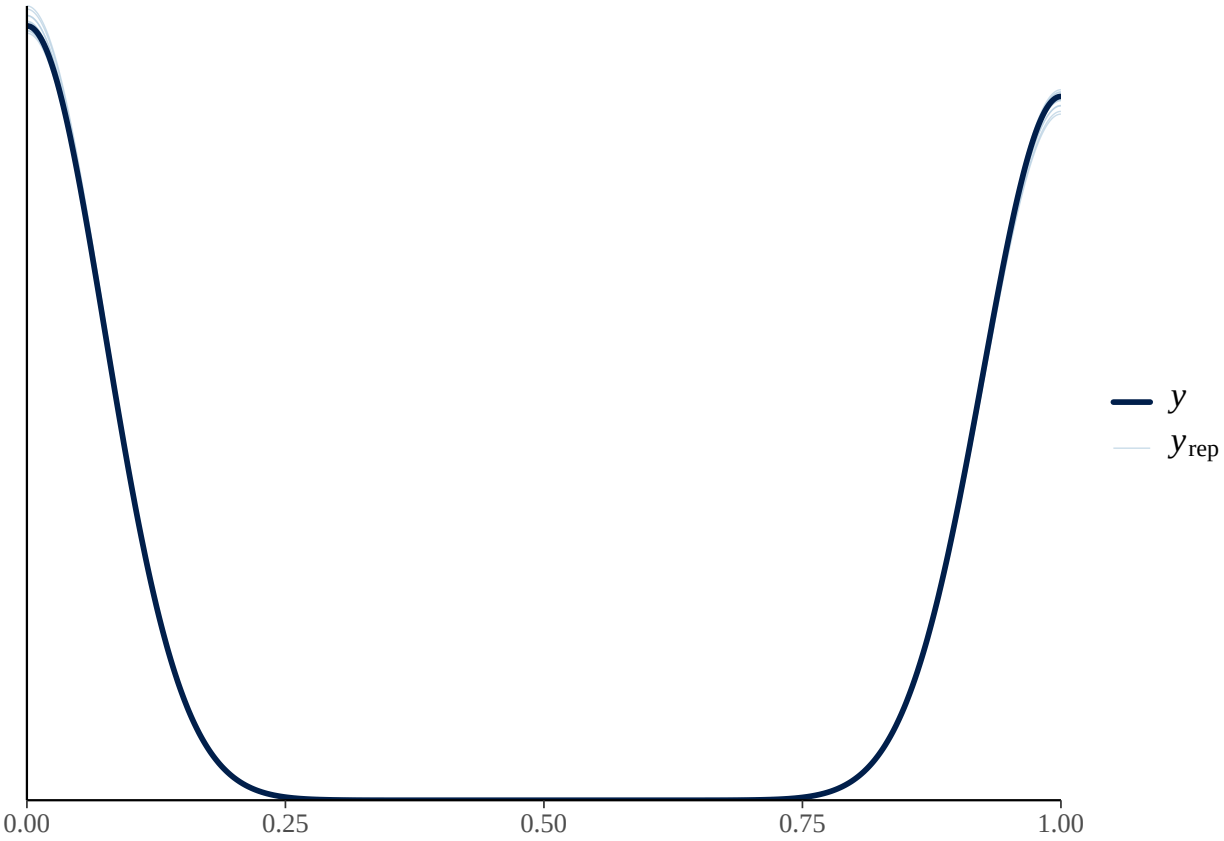



```
##
##  DHARMa zero-inflation test via comparison to expected zeros with
##  simulation under H0 = fitted model
##
## data:  simulationOutput
## ratioObsSim = 1.0009, p-value = 0.976
## alternative hypothesis: two.sided
```

Q-Q Plot Residuals: The residuals follow the 1:1 diagonal line, confirming no systematic deviation from the expected uniform distribution. Tests for uniformity (Kolmogorov-Smirnov), dispersion, and outliers are non-significant (`p > 0.05`), confirming that residuals are well-behaved. **Dispersion Test:** Dispersion ratio (`1.0002`) and `p-value = 1` indicate no overdispersion, suggesting the variance in the data is well-captured by the model. **Zero-Inflation Test:** The observed-to-expected ratio of zeros is close to 1 (`ratioObsSim = 1.0009`), with a non-significant `p-value = 0.976`. This confirms that there is no excessive zero-inflation in the model.

```
# Posterior predictive check with density overlay
pp_check(model_brms, type = "dens_overlay")
```

```
## Using 10 posterior draws for ppc type 'dens_overlay' by default.
```



The model's predicted distributions align closely with the observed data, showing that the model captures the underlying structure of the data well. The model fits the data effectively, with no visible discrepancies between observed and predicted distributions.

Adjustments and Refinements

```
# Create spline terms for a continuous predictor, e.g., "Anger"
VerbAgg$spline_Anger <- ns(VerbAgg$Anger, df = 4)
```

1. `lme4`: Addressing Deviations in Residuals at Extremes by Adding Interaction Terms

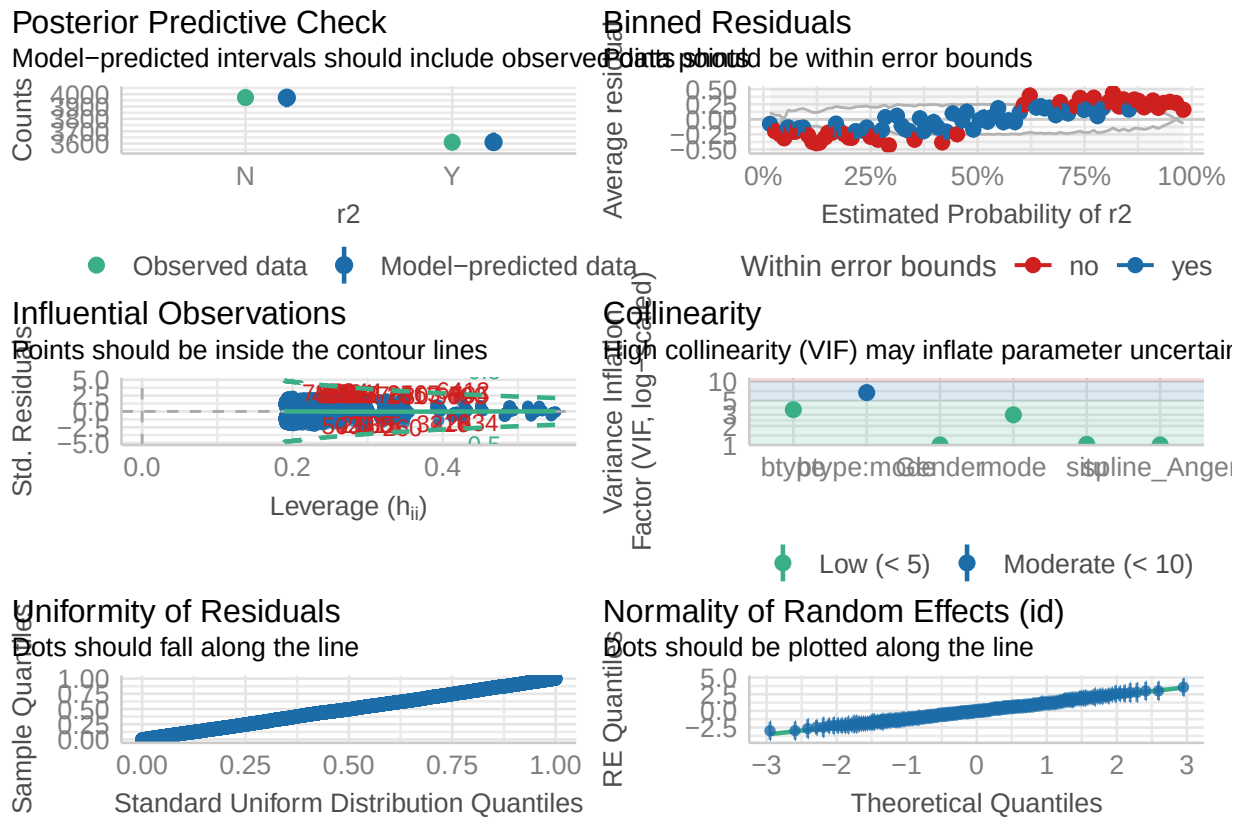
Adding the interaction `btype:mode` allows the model to capture potential variations in how the behavioral type (`btype`) affects aggressive responses (`r2`) depending on the mode of response (`mode`).

```
model_lme4_refined <- glmer(r2 ~ Gender + btype * mode + situ + spline_Anger + (1 | id),
                            data = VerbAgg, family = binomial(link = "logit"),
                            control = glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 1e5)))
```

```
# Check model performance
check_model(model_lme4_refined)
```

```
## Some of the variables were in matrix-format - probably you used
##   'scale()' on your data?
```

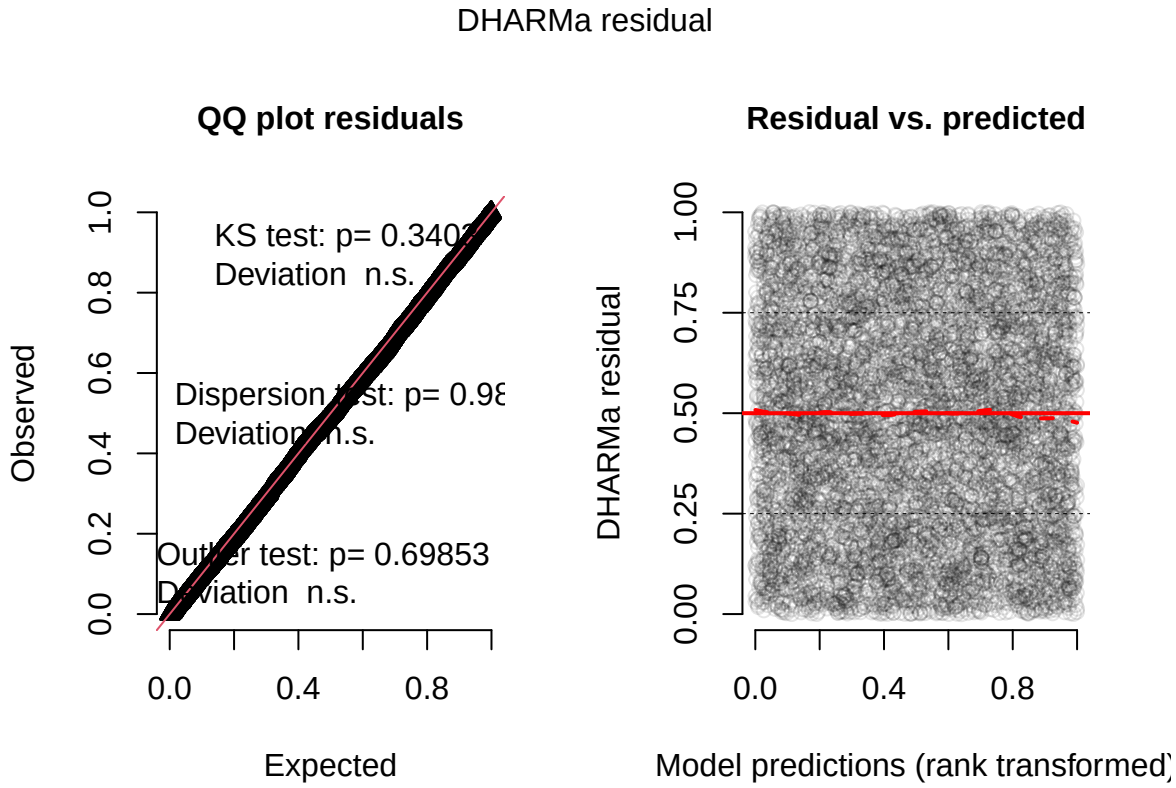
```
##   If so, and you get an error, please try 'datawizard::standardize()' to
##   standardize your data.
## Some of the variables were in matrix-format - probably you used
##   'scale()' on your data?
##   If so, and you get an error, please try 'datawizard::standardize()' to
##   standardize your data.
```




```
# Residual diagnostics
check_residuals(model_lme4_refined)
```

```
## OK: Simulated residuals appear as uniformly distributed (p = 0.340).
```

```
# Simulate residuals for DHARMa diagnostics
res_lme4_refined <- simulateResiduals(fittedModel = model_lme4_refined)

# Plot residual diagnostics
plot(res_lme4_refined)
```

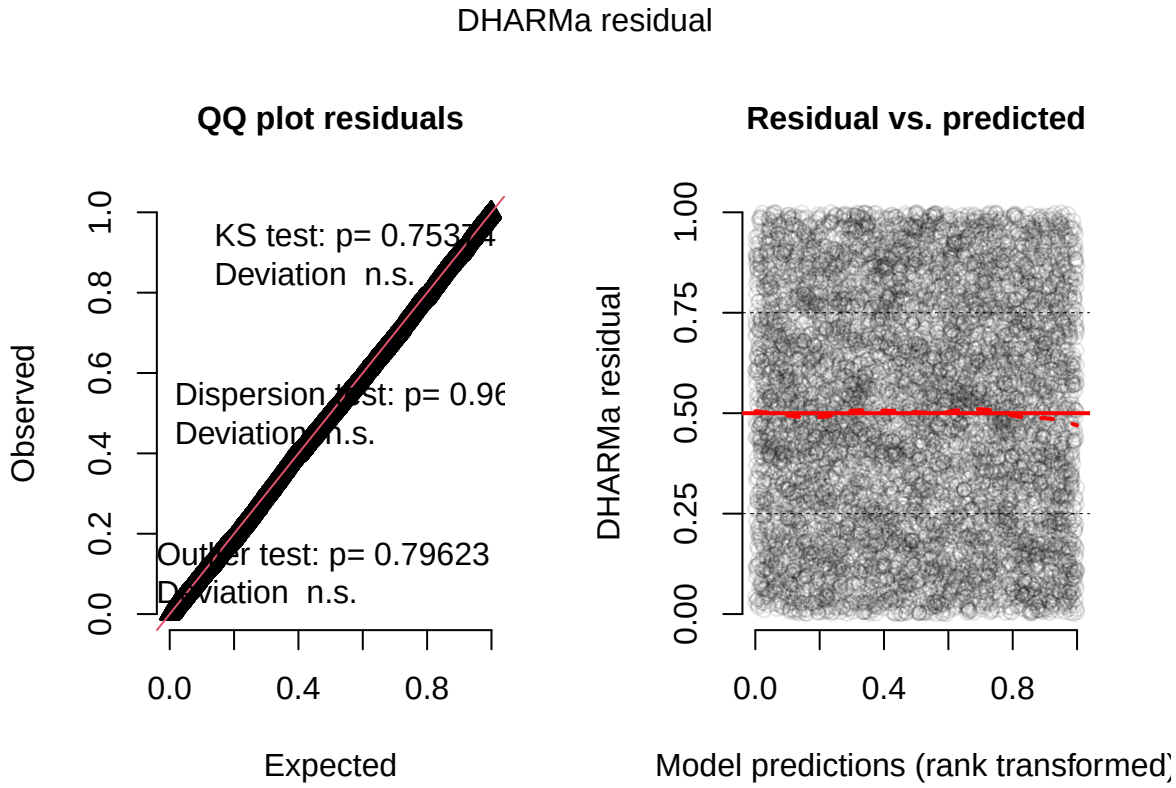


Based on the diagnostic results, the refined model with the interaction term (`btype:mode`) including spline terms (`spline_Anger`) does not improve significantly compared to the simpler model (`model_lme4`). Therefore, I would revert to the simpler model (`model_lme4`).

2. `glmmTMB`: Refit with Alternative Link Functions

```
model_glmmTMB_probit <- glmmTMB(r2 ~ Gender + btype + mode + situ + spline_Anger + (1 | id),
                                data = VerbAgg, family = binomial(link = "probit"))
```

```
# Simulate residuals for DHARMa diagnostics
res_glmmTMB_probit <- simulateResiduals(fittedModel = model_glmmTMB_probit)

# Plot residual diagnostics
plot(res_glmmTMB_probit)
```



The diagnostics for the `probit` model (`model_glmmTMB_probit`) including spline terms (`spline_Anger`) do not show significant improvement over the simpler `logit` model (`model_lme4`). Thus, I will keep using the `logit` model (`model_lme4`).

3. `brms` Model with spline

```
model_brms_spline <- brm(
  r2 ~ Gender + btype + mode + situ + spline_Anger + (1 | id),
  data = VerbAgg, family = bernoulli(link = "logit"),
  prior = set_prior("normal(0, 5)", class = "b"))
```

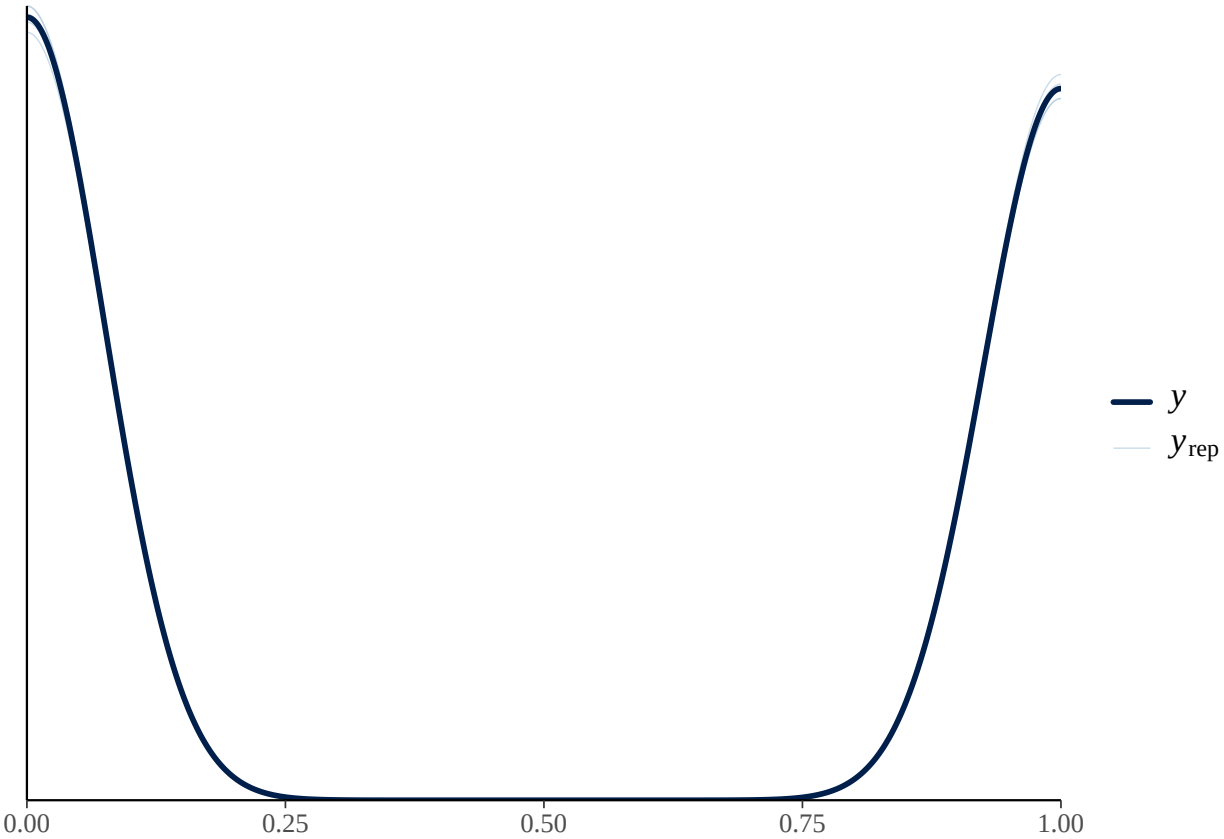
```
## Compiling Stan program...

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.002908 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 29.08 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 67.29 seconds (Warm-up)
## Chain 1:                68.527 seconds (Sampling)
## Chain 1:                135.817 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.002133 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 21.33 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 72.861 seconds (Warm-up)
## Chain 2:                70.732 seconds (Sampling)
## Chain 2:                143.593 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.002256 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 22.56 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 84.835 seconds (Warm-up)
## Chain 3:                69.065 seconds (Sampling)
## Chain 3:                153.9 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.002148 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 21.48 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 73.336 seconds (Warm-up)
## Chain 4:                69.803 seconds (Sampling)
## Chain 4:                143.139 seconds (Total)
## Chain 4:
```

```
pp_check(model_brms_spline, type = "dens_overlay")
```

```
## Using 10 posterior draws for ppc type 'dens_overlay' by default.
```



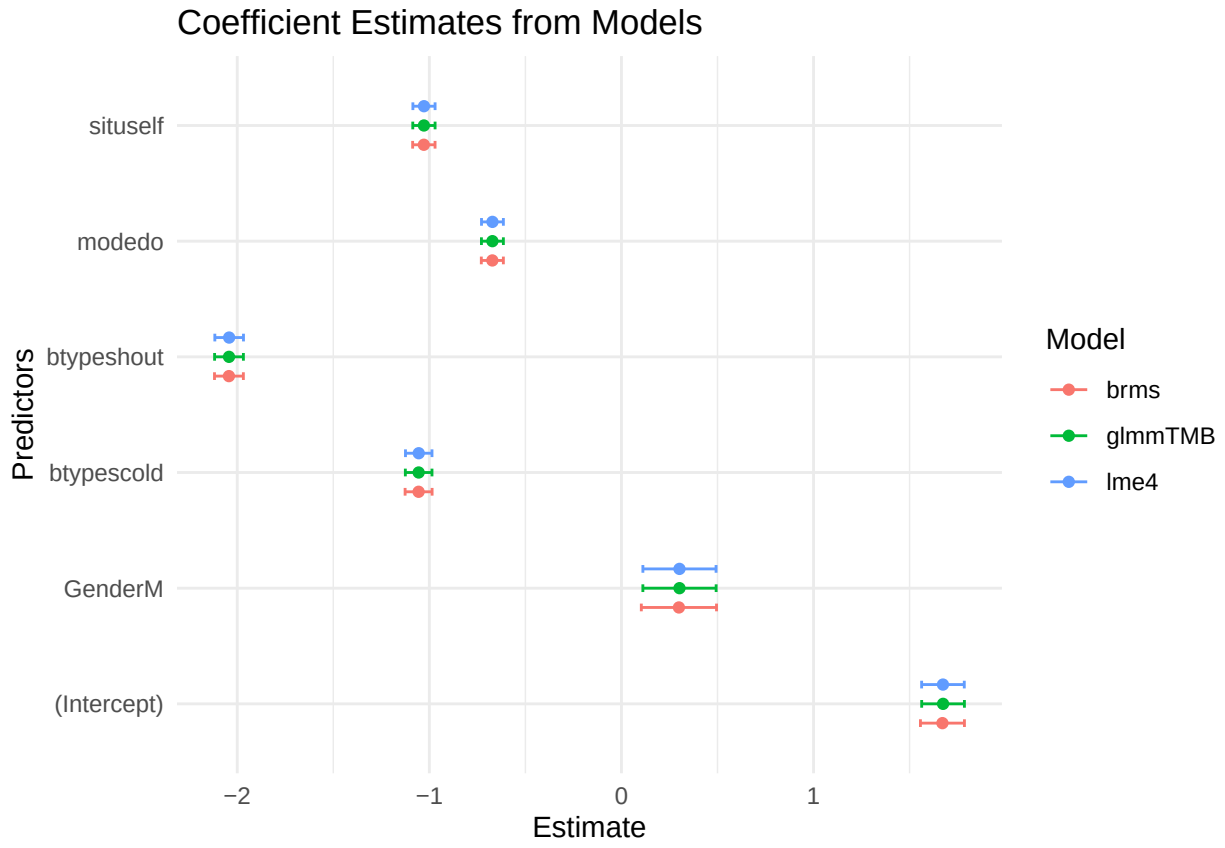
The diagnostics for the model (`model_brms_spline`) including spline terms (`spline_Anger`) do not show significant improvement over the simpler model (`model_brms`). Thus, I will keep using the simpler model (`model_brms`).

Conclusions

Coefficient plot

```
# Extract tidy summaries
tidy_lme4 <- tidy(model_lme4)
tidy_glmmTMB <- tidy(model_glmmTMB)
tidy_brms <- tidy(model_brms, effects = "fixed")

# Add model names
tidy_lme4$model <- "lme4"
tidy_glmmTMB$model <- "glmmTMB"
tidy_brms$model <- "brms"

# Combine into one data frame
coefficients <- bind_rows(tidy_lme4, tidy_glmmTMB, tidy_brms)

# Filter for fixed effects only
coefficients <- coefficients %>%
  filter(effect == "fixed") %>%
  mutate(term = factor(term, levels = unique(term)))
```

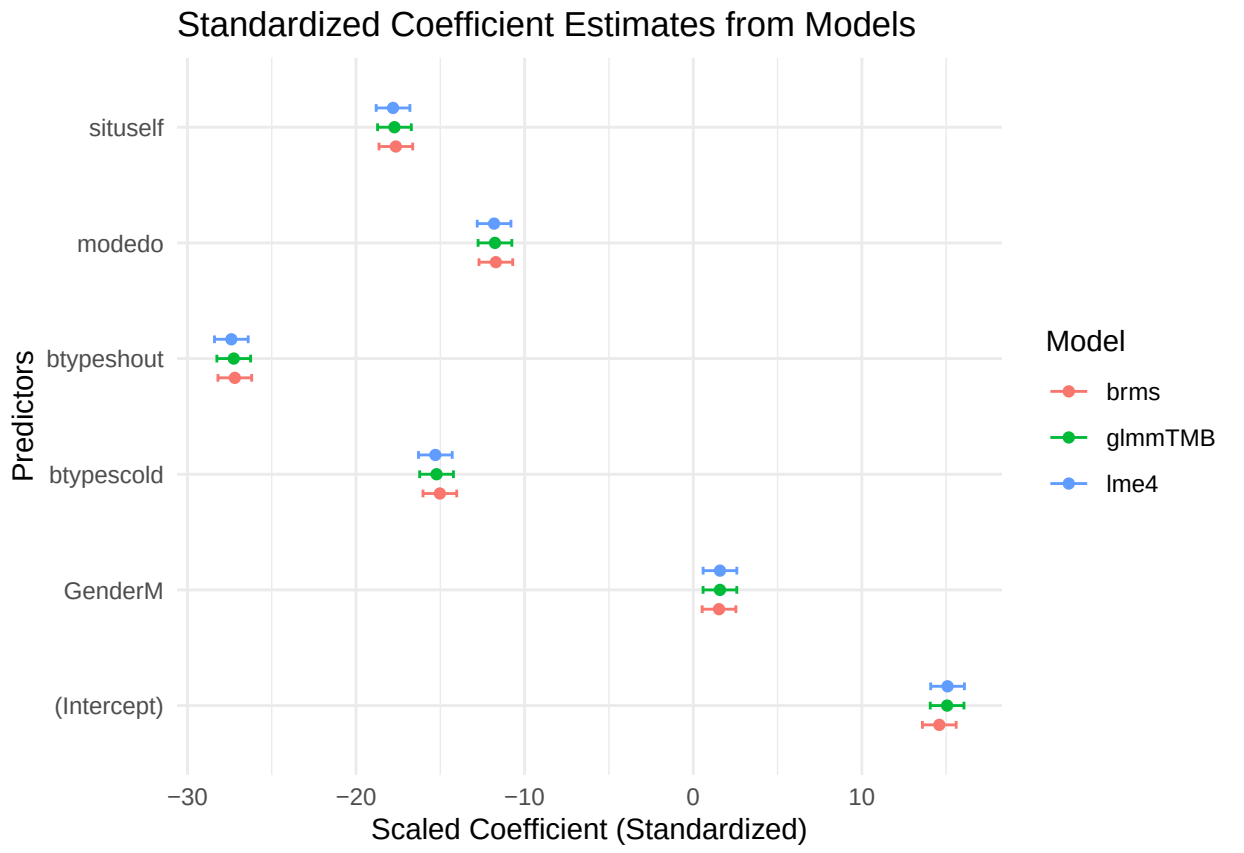
```r
# Coefficient plot
ggplot(coefficients, aes(x = estimate, y = term, color = model)) +
  geom_point(position = position_dodge(width = 0.5)) +
  geom_errorbar(aes(xmin = estimate - std.error, xmax = estimate + std.error),
                position = position_dodge(width = 0.5), width = 0.2) +
  theme_minimal() +
  labs(
    title = "Coefficient Estimates from Models",
    x = "Estimate",
    y = "Predictors",
    color = "Model")
```



Since the predictors have different units, the coefficient plot should be scaled.

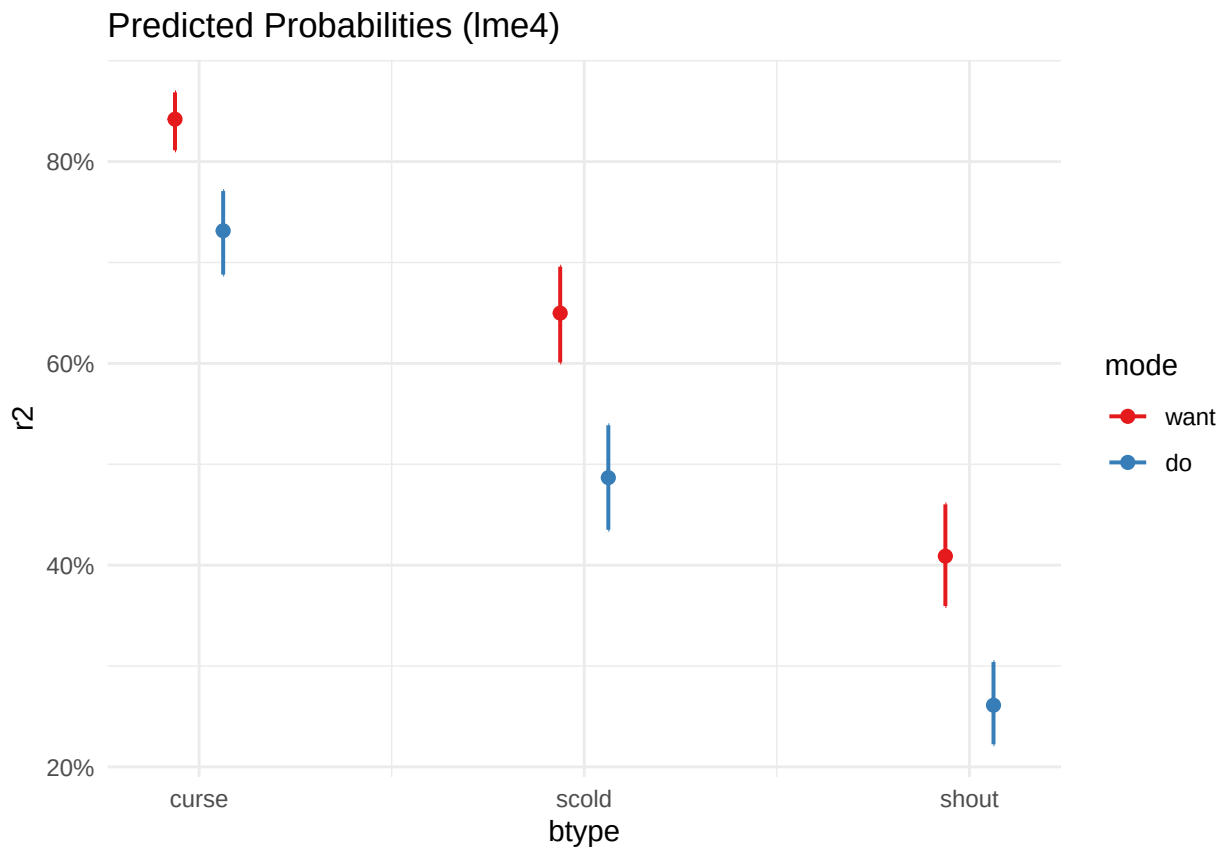
```r
coefficients <- coefficients %>%
  mutate(scaled_estimate = estimate / std.error)  # Scale by standard error for comparability

ggplot(coefficients, aes(x = scaled_estimate, y = term, color = model)) +
  geom_point(position = position_dodge(width = 0.5)) +
  geom_errorbar(aes(xmin = scaled_estimate - 1, xmax = scaled_estimate + 1),
                position = position_dodge(width = 0.5), width = 0.2) +
  theme_minimal() +
  labs(
    title = "Standardized Coefficient Estimates from Models",
    x = "Scaled Coefficient (Standardized)",
    y = "Predictors",
```

```
    color = "Model")
```

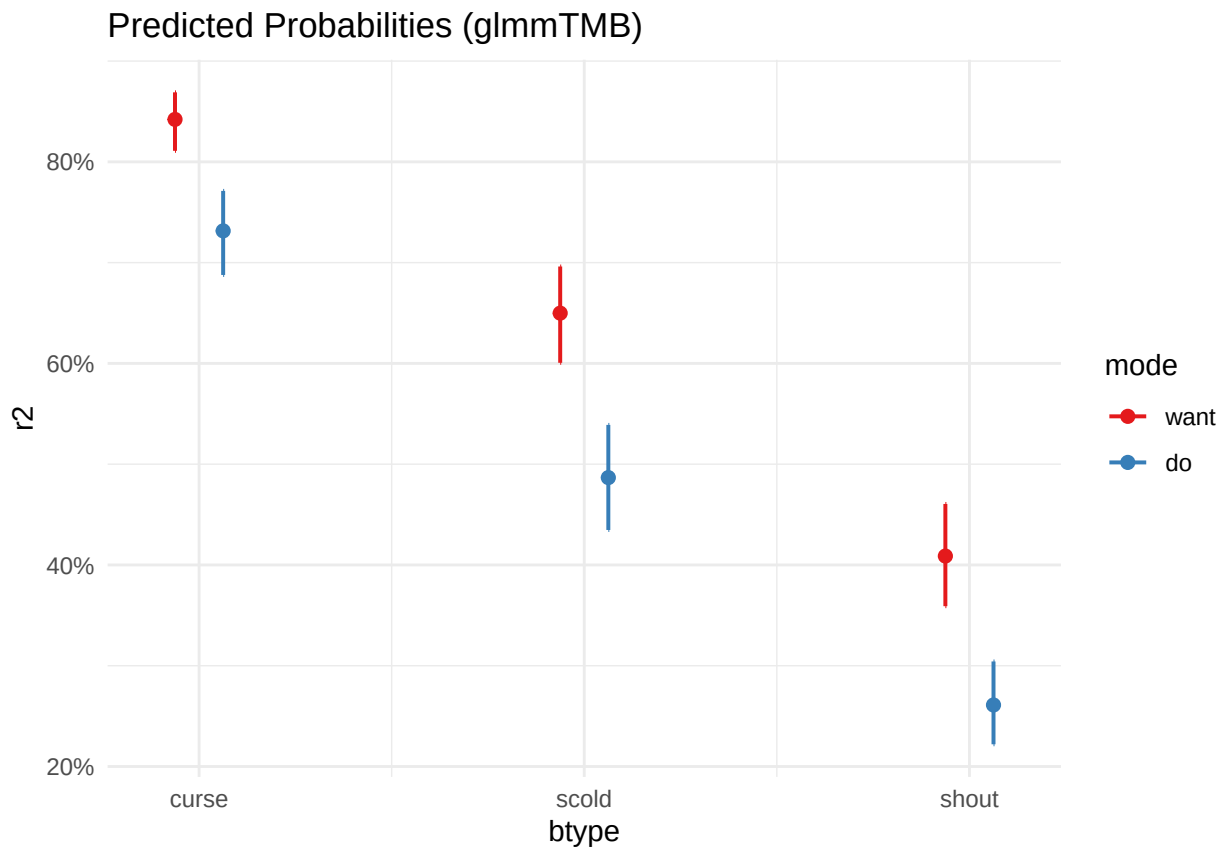


Effects Plots

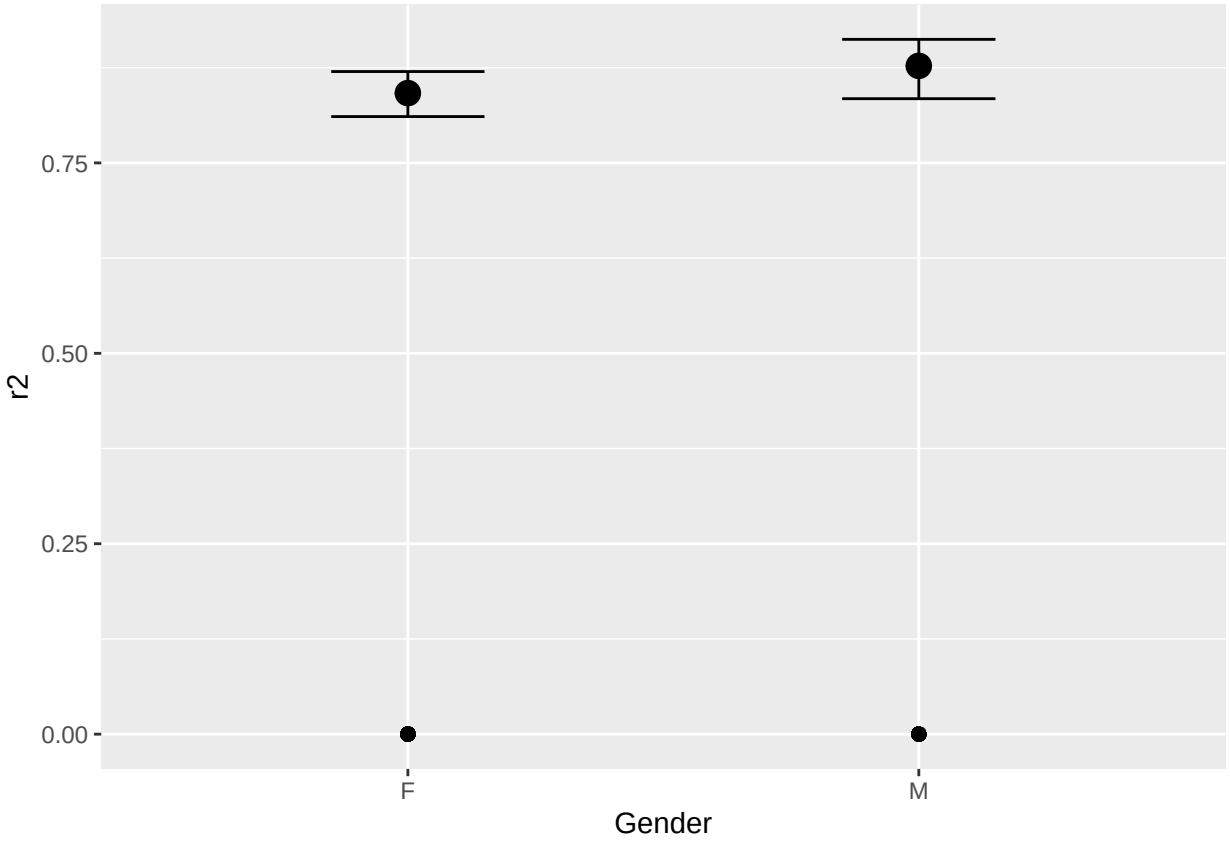
```
# Predicted Probabilities (for Frequentist Models)

# Generate predicted effects for lme4 and glmmTMB models
pred_lme4 <- ggeffects::ggpredict(model_lme4, terms = c("btype", "mode"))
pred_glmmTMB <- ggeffects::ggpredict(model_glmmTMB, terms = c("btype", "mode"))

# Plot predicted probabilities for lme4
plot(pred_lme4) +
  ggtitle("Predicted Probabilities (lme4)") +
  theme_minimal()
```

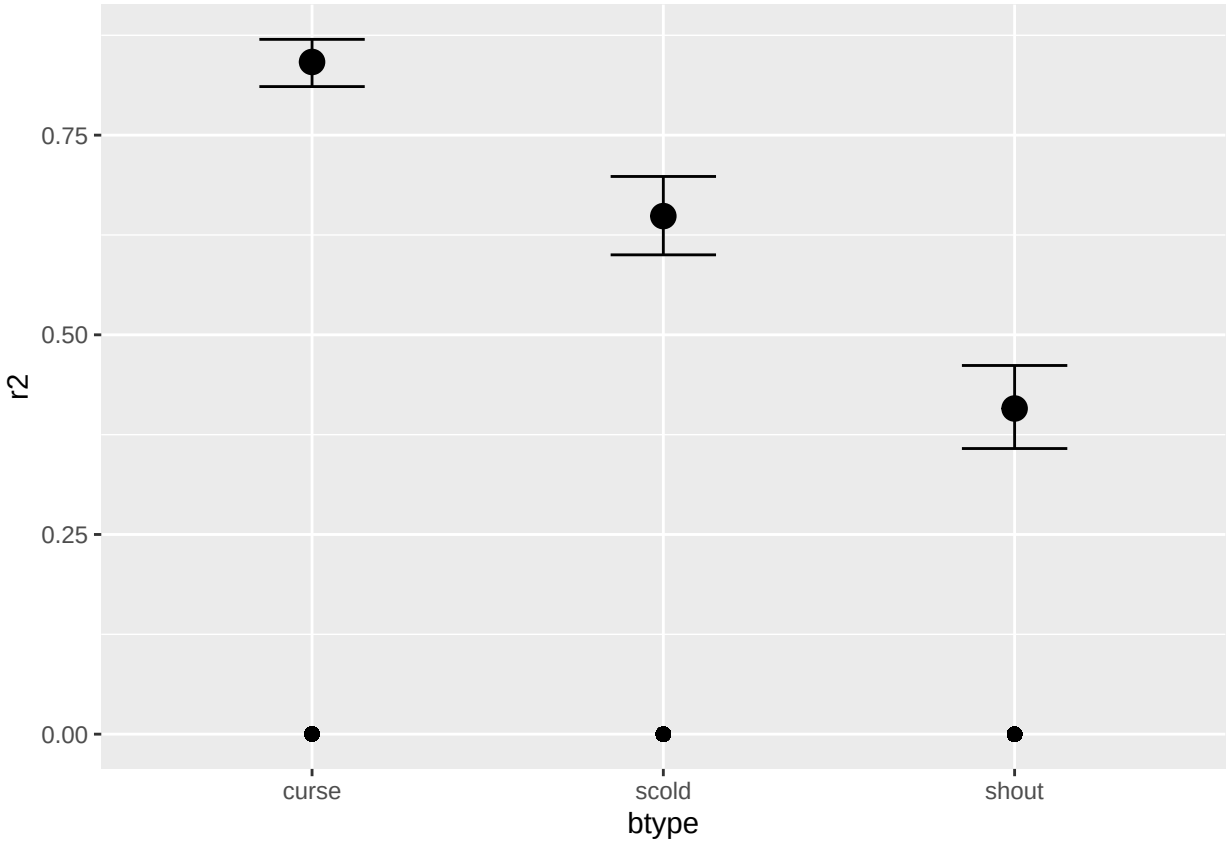


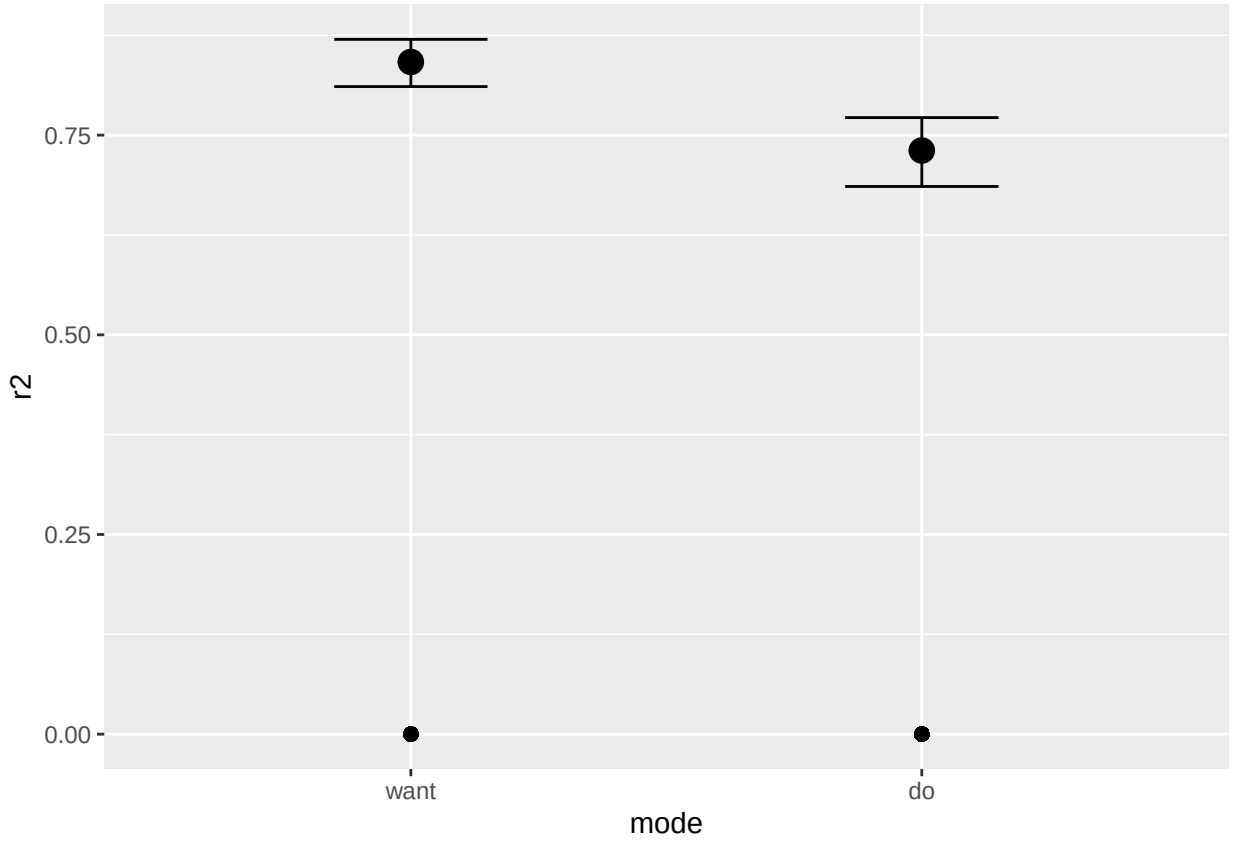

```
# Plot predicted probabilities for glmmTMB
plot(pred_glmmTMB) +
  ggtitle("Predicted Probabilities (glmmTMB)") +
  theme_minimal()
```

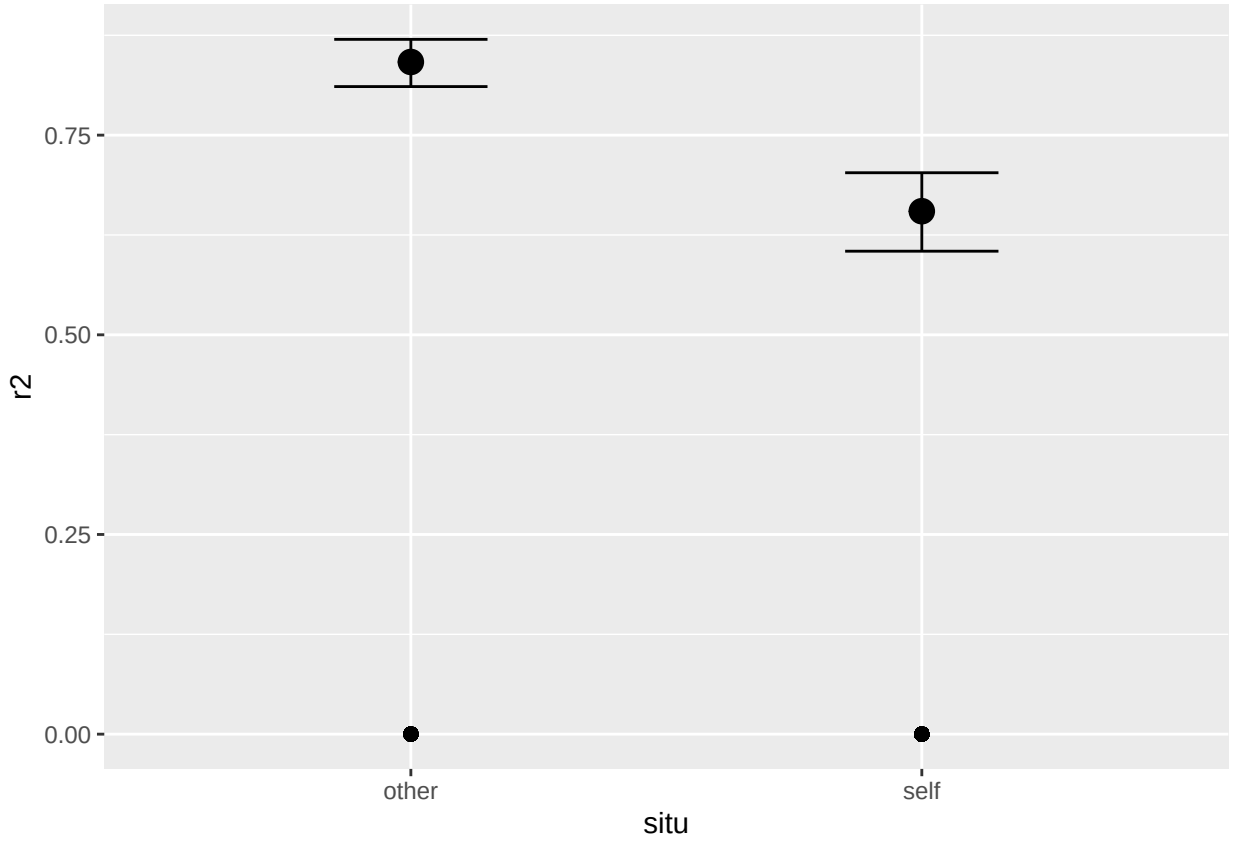



```
# Generate conditional effects
conditional_effects_brms <- conditional_effects(model_brms)

# Plot conditional effects
plot(conditional_effects_brms, points = TRUE)
```


0.75
0.50
0.25
0.00
r2
F
M
Gender


0.75
0.50
0.25
0.00
r2
curse
scold
shout
btype


0.75
0.50
0.25
0.00
r2
want
do
mode


0.75
0.50
0.25
0.00
r2
other
self
situ