

Non-inferiority Test

Mohammed Al Owayyed

2025-04-11

Contents

1	Prepare and load the files	1
2	NLP	2
2.1	Add Intent Columns Based on Labels	2
2.2	Fleiss' Kappa: Interrater Reliability	2
2.3	Bayesian Modeling: LLM vs BDI Agreement	4
2.4	ROPE and posterior check	6
2.5	ROPE and non-inferiority results	6
3	NLG and Bypass	7
3.1	Process NLG Data	7
3.2	Summary Statistics for Human vs LLM	8
3.3	Bayesian Model with ROPE	9
3.4	ROPE & Posterior Results for NLG	10
3.5	ROPE and non-inferiority results	11

In this file, we will describe how we performed the Bayesian non-inferiority for the NLP and NLG components. For that, we show the following:

- Data transforming process
- Interrater Reliability
- Fitted models
- ROPE and posterior distributions
- Non-inferiority test

To conduct the analysis, we need two files:

- NLG.csv: includes the coders assessment of the generated examples (either human generated or LLM generated)
- NLP.csv: includes whether the LLM, the rule-based system, the four coders identified the correct intent from the example inputs. we also added a “mean coder” to get one human coder overall measure.

First, let's load the necessary libraries:

1 Prepare and load the files

```
# Load the CSV files
data <- read.csv("NLG.csv")
df <- read.csv("NLP.csv")
```

2 NLP

2.1 Add Intent Columns Based on Labels

```
update_with_intent <- function(df) { #change the 0,1 in each column to the intent label categorisation
  df$`BDIintent` <- ifelse(df$BDI == 1, as.character(df$GroundTruth), 0)
  df$`LLMintent` <- ifelse(df$LLM == 1, as.character(df$GroundTruth), 0)
  df$`Coder1intent` <- ifelse(df$Coder1 == 1, as.character(df$GroundTruth), 0)
  df$`Coder2intent` <- ifelse(df$Coder2 == 1, as.character(df$GroundTruth), 0)
  df$`Coder3intent` <- ifelse(df$Coder3 == 1, as.character(df$GroundTruth), 0)
  df$`Coder4intent` <- ifelse(df$Coder4 == 1, as.character(df$GroundTruth), 0)
  df$`MedianCoderintent` <- ifelse(df$MedianCoder == 1, as.character(df$GroundTruth), 0)
  return(df)
}
df_updated <- update_with_intent(df)
```

2.2 Fleiss' Kappa: Interrater Reliability

```
ratingsCoders <- data.frame( # between the 4 coders
  #LLM = factor(df_updated$LLMintent),
  #BDI = factor(df_updated$BDIintent),
  #groundTruth = factor(df_updated$GroundTruth),
  Coder1 = factor(df_updated$Coder1intent),
  Coder2 = factor(df_updated$Coder2intent),
  Coder3 = factor(df_updated$Coder3intent),
  Coder4 = factor(df_updated$Coder4intent)
)

ratingsCodersLLM <- data.frame( # between the coders and the LLM
  LLM = factor(df_updated$LLMintent),
  #BDI = factor(df_updated$BDIintent),
  #groundTruth = factor(df_updated$GroundTruth),
  Coder1 = factor(df_updated$Coder1intent),
  Coder2 = factor(df_updated$Coder2intent),
  Coder3 = factor(df_updated$Coder3intent),
  Coder4 = factor(df_updated$Coder4intent)
)

ratingsCodersRBS <- data.frame( # between the BDI (rule-based) and the coders
  #LLM = factor(df_updated$LLMintent),
  BDI = factor(df_updated$BDIintent),
  #groundTruth = factor(df_updated$GroundTruth),
  Coder1 = factor(df_updated$Coder1intent),
  Coder2 = factor(df_updated$Coder2intent),
  Coder3 = factor(df_updated$Coder3intent),
  Coder4 = factor(df_updated$Coder4intent)
)

kappam.fleiss(ratingsCoders)

## Fleiss' Kappa for m Raters
```

```

##
## Subjects = 14
## Raters = 4
## Kappa = 0.806
##
## z = 25.3
## p-value = 0
kappam.fleiss(ratingsCodersLLM)

## Fleiss' Kappa for m Raters
##
## Subjects = 14
## Raters = 5
## Kappa = 0.815
##
## z = 34.4
## p-value = 0
kappam.fleiss(ratingsCodersRBS)

## Fleiss' Kappa for m Raters
##
## Subjects = 14
## Raters = 5
## Kappa = 0.783
##
## z = 32.1
## p-value = 0

ratingsTruthMedian <- data.frame( # between the ground truth and the median coder
  #LLM = factor(df_updated$LLMintent),
  #BDI = factor(df_updated$BDIintent),
  groundTruth = factor(df_updated$GroundTruth),
  MedianCoder = factor(df_updated$MedianCoderintent)
)

ratingsTruthLLM <- data.frame( # between the ground truth and the LLM
  LLM = factor(df_updated$LLMintent),
  #BDI = factor(df_updated$BDIintent),
  groundTruth = factor(df_updated$GroundTruth),
  #MedianCoder = factor(df_updated$MedianCoderintent),
)

ratingsTruthRBS <- data.frame( # between the ground truth and the BDI
  #LLM = factor(df_updated$LLMintent),
  BDI = factor(df_updated$BDIintent),
  groundTruth = factor(df_updated$GroundTruth),
  #MedianCoder = factor(df_updated$MedianCoderintent),
)

# Calculate Cohen's Kappa

```

```

kappa_result1 <- kappa2(ratingsTruthMedian)
kappa_result2 <- kappa2(ratingsTruthLLM)
kappa_result3 <- kappa2(ratingsTruthRBS)

# View the results
print(kappa_result1)

## Cohen's Kappa for 2 Raters (Weights: unweighted)
##
## Subjects = 14
## Raters = 2
## Kappa = 0.923
##
## z = 13
## p-value = 0
print(kappa_result2)

## Cohen's Kappa for 2 Raters (Weights: unweighted)
##
## Subjects = 14
## Raters = 2
## Kappa = 1
##
## z = 13.5
## p-value = 0
print(kappa_result3)

## Cohen's Kappa for 2 Raters (Weights: unweighted)
##
## Subjects = 14
## Raters = 2
## Kappa = 0.848
##
## z = 12.6
## p-value = 0

```

Fleiss' Kappa indicated almost perfect agreement among human coders alone ($\kappa = 0.81$), as well as between coders and LLM ($\kappa = 0.82$); agreement was substantial between coders and the rule-based model ($\kappa = 0.78$). Cohen's Kappa indicated almost perfect agreement between the LLM and ground truth ($\kappa = 1.00$), the median coder and ground truth ($\kappa = 0.92$), and the rule-based model and ground truth ($\kappa = 0.85$).

2.3 Bayesian Modeling: LLM vs BDI Agreement

We will first fit a bayesian model to predict the agreement between each group (i.e., LLM or BDI) and the ground truth

```

df_filtered <- df %>% select(GroundTruth, LLM, BDI)

df_transformed <- df_filtered %>%
  pivot_longer(cols = c(LLM, BDI), names_to = "Origin", values_to = "Agreement")

colnames(df_transformed)[1] <- "GroundTruth"
df_transformed$Origin <- as.factor(df_transformed$Origin)

```

```

modelNLP <- brm(Agreement ~ Origin, data = df_transformed, family = bernoulli(link = "logit"),
  prior = c(set_prior("normal(0, 1)", class = "b")),
  iter = 10000, warmup = 1000, chains = 4, cores = 4, seed = 1234)

## Compiling Stan program...
## Trying to compile a simple C file
## Running /usr/local/lib/R/bin/R CMD SHLIB foo.c
## using C compiler: 'gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0'
## gcc -I"/usr/local/lib/R/include" -DNDEBUG -I"/usr/local/lib/R/site-library/Rcpp/include/" -I"/usr/
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
## from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
## from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:
## from <command-line>:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown t
## 628 | namespace Eigen {
## | ^~~~~~
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected
## 628 | namespace Eigen {
## | ^
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
## from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:
## from <command-line>:
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file
## 96 | #include <complex>
## | ^~~~~~
## compilation terminated.
## make: *** [/usr/local/lib/R/etc/Makeconf:191: foo.o] Error 1

## Start sampling
summary(modelNLP)

## Family: bernoulli
## Links: mu = logit
## Formula: Agreement ~ Origin
## Data: df_transformed (Number of observations: 28)
## Draws: 4 chains, each with iter = 10000; warmup = 1000; thin = 1;
## total post-warmup draws = 36000
##
## Population-Level Effects:
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept 2.34 0.80 0.96 4.09 1.00 28016 20825
## OriginLLM 0.67 0.83 -0.92 2.31 1.00 23953 22684
##
## 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).
hypothesis(modelNLP, "OriginLLM > 0")

## Hypothesis Tests for class b:
## Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob
## 1 (OriginLLM) > 0 0.67 0.83 -0.67 2.04 3.82 0.79
## Star

```

```
## 1
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

2.4 ROPE and posterior check

```
describe_posterior(modelNLP, ci = 0.95)
```

```
## Summary of Posterior Distribution
```

```
##
```

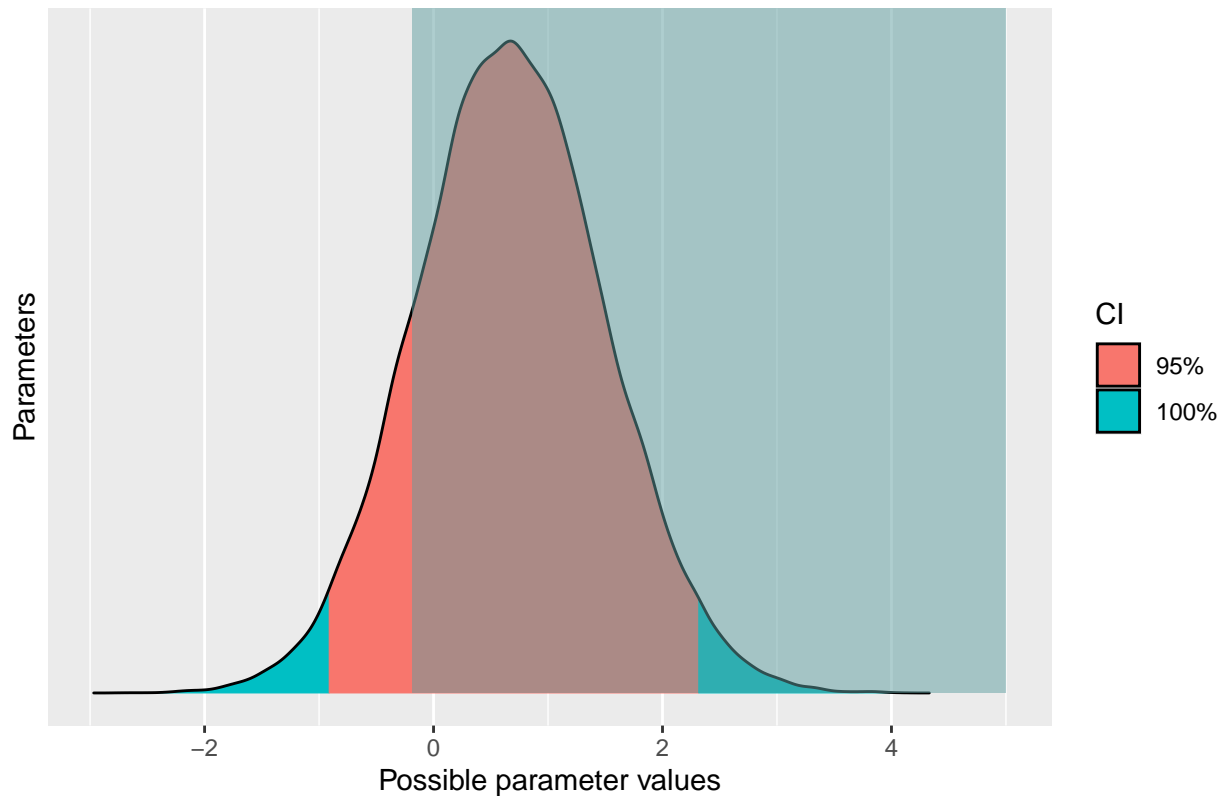
## Parameter	Median	95% CI	pd	ROPE	% in ROPE	Rhat	ESS
## (Intercept)	2.28	[0.96, 4.09]	99.98%	[-0.18, 0.18]	0%	1.000	25861.00
## OriginLLM	0.67	[-0.92, 2.31]	79.25%	[-0.18, 0.18]	13.16%	1.000	23893.00

2.5 ROPE and non-inferiority results

We defined a non-inferiority threshold just below the null value. We then assessed the proportion of the posterior distribution of the parameter estimate that fell above this threshold, which would indicate practical non-inferiority. The threshold was set based on half of Cohen's threshold for a small effect size to define negligible or better effects. In this case, the rang is posteriors > -0.18 . We defined the upper range as 5 as all values are lower than this number.

```
rope_result3 <- rope(modelNLP, range = c(-0.18, 5))
plot(rope_result3)
```

Region of Practical Equivalence (ROPE)



```
print(rope_result3)
```

```
## # Proportion of samples inside the ROPE [-0.18, 5.00]:
##
## Parameter | inside ROPE
## -----
## Intercept |    100.00 %
## OriginLLM |     86.82 %
```

The non-inferiority results indicate strong support for the non-inferiority of the LLM, with the probability of 86.82% being non-inferior.

3 NLG and Bypass

3.1 Process NLG Data

We first transform the data so that we can fit a model

```
transformed_data <- data %>%
  mutate(
    textNumber = as.numeric(gsub("Q", "", textNumber)),
    Score = rowMeans(select(., S1, S2, S3, S4, S5), na.rm = TRUE),
    Route = sub(":.*", "", Route),
    GeneratedBy = sub(".*: ", "", data$Route),
    GeneratedBy = ifelse(grepl("Bypass:LLM", data$Route), "LLM", GeneratedBy)
  ) %>%
  select(Coder, textNumber, Score, Route, GeneratedBy)
```

```
head(transformed_data)
```

```
##   Coder textNumber Score Route GeneratedBy
## 1     1           1  7.0   NLG          LLM
## 2     1           2  5.8   NLG          LLM
## 3     1           3  7.0   NLG          LLM
## 4     1           4  6.4   NLG          LLM
## 5     1           5  5.6   NLG          LLM
## 6     1           6  7.0   NLG          LLM
```

3.2 Summary Statistics for Human vs LLM

```
#statistics for the human generated responses
summary_stats1 <- transformed_data %>%
  filter(GeneratedBy == "Human") %>%
  summarise(n = n(), mean_score = mean(Score), sd_score = sd(Score), median = median(Score))

#statistics for the LLM generated responses
summary_stats2 <- transformed_data %>%
  filter(GeneratedBy == "LLM") %>%
  summarise(n = n(), mean_score = mean(Score), sd_score = sd(Score), median = median(Score))

#statistics for the coder 1 assessments
coder1 <- transformed_data %>%
  filter(Coder == 1) %>%
  summarise(n = n(), mean_score = mean(Score), sd_score = sd(Score), median = median(Score))

#statistics for the coder 2 assessments
coder2 <- transformed_data %>%
  filter(Coder == 2) %>%
  summarise(n = n(), mean_score = mean(Score), sd_score = sd(Score), median = median(Score))

#statistics for the coder 3 assessments
coder3 <- transformed_data %>%
  filter(Coder == 3) %>%
  summarise(n = n(), mean_score = mean(Score), sd_score = sd(Score), median = median(Score))

#statistics for the coder 4 assessments
coder4 <- transformed_data %>%
  filter(Coder == 4) %>%
  summarise(n = n(), mean_score = mean(Score), sd_score = sd(Score), median = median(Score))

print(summary_stats1)

##   n mean_score sd_score median
## 1 32      6.275 0.7799917    6.5

print(summary_stats2)

##   n mean_score sd_score median
## 1 64      6.303125 0.8880099     7

print(coder1)

##   n mean_score sd_score median
```



```
## 1 24 6.233333 0.8760915 6.5
```

```
print(coder2)
```

```
##      n mean_score sd_score median
## 1 24 6.808333 0.3658423 7
```

```
print(coder3)
```

```
##      n mean_score sd_score median
## 1 24 6.633333 0.6062477 7
```

```
print(coder4)
```

```
##      n mean_score sd_score median
## 1 24 5.5 0.8086489 5.4
```

3.3 Bayesian Model with ROPE

We will fit a Bayesian model to predict the score given to each group (i.e., LLM or human generates)

```
BayesianModel <- brm(Score ~ GeneratedBy + (1 + GeneratedBy | Coder),
  data = transformed_data, family = gaussian(),
  iter = 10000, warmup = 1000, chains = 4, cores = 4, seed = 1234)
```

```
## Compiling Stan program...
```

```
## Trying to compile a simple C file
```

```
## Running /usr/local/lib/R/bin/R CMD SHLIB foo.c
```

```
## using C compiler: 'gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0'
```

```
## gcc -I"/usr/local/lib/R/include" -DNDEBUG -I"/usr/local/lib/R/site-library/Rcpp/include/" -I"/usr/
```

```
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
```

```
## from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
```

```
## from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
```

```
## from <command-line>:
```

```
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error: unknown t
```

```
## 628 | namespace Eigen {
```

```
## | ~~~~~
```

```
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:17: error: expected
```

```
## 628 | namespace Eigen {
```

```
## | ^
```

```
## In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
```

```
## from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp
```

```
## from <command-line>:
```

```
## /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file
```

```
## 96 | #include <complex>
```

```
## | ~~~~~
```

```
## compilation terminated.
```

```
## make: *** [/usr/local/lib/R/etc/Makeconf:191: foo.o] Error 1
```

```
## Start sampling
```

```
## Warning: There were 262 divergent transitions after warmup. See
```

```
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
summary(BayesianModel)
```

```
## Warning: There were 262 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Family: gaussian
## Links: mu = identity; sigma = identity
## Formula: Score ~ GeneratedBy + (1 + GeneratedBy | Coder)
## Data: transformed_data (Number of observations: 96)
## Draws: 4 chains, each with iter = 10000; warmup = 1000; thin = 1;
## total post-warmup draws = 36000
##
## Group-Level Effects:
## ~Coder (Number of levels: 4)
##
```

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat
sd(Intercept)	0.87	0.57	0.22	2.41	1.00
sd(GeneratedByLLM)	0.42	0.45	0.01	1.62	1.00
cor(Intercept,GeneratedByLLM)	0.17	0.56	-0.90	0.97	1.00

```
## Bulk_ESS Tail_ESS
## sd(Intercept) 5533 3753
## sd(GeneratedByLLM) 3085 1262
## cor(Intercept,GeneratedByLLM) 8191 7153
##
## Population-Level Effects:
##
```

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
Intercept	6.29	0.49	5.24	7.32	1.00	3452	2499
GeneratedByLLM	0.04	0.32	-0.58	0.71	1.00	2365	1437

```
##
## Family Specific Parameters:
##
```

	Estimate	Est.Error	1-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sigma	0.71	0.05	0.61	0.82	1.00	19262	21978

```
##
## 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).
```

```
hypothesis(BayesianModel, "GeneratedByLLM > 0")
```

```
## Hypothesis Tests for class b:
##
```

	Hypothesis	Estimate	Est.Error	CI.Lower	CI.Upper	Evid.Ratio
## 1	(GeneratedByLLM) > 0	0.04	0.32	-0.43	0.51	1.24

```
## Post.Prob Star
## 1 0.55
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

3.4 ROPE & Posterior Results for NLG

```
describe_posterior(BayesianModel, ci = 0.95)
```

```
## Summary of Posterior Distribution
```

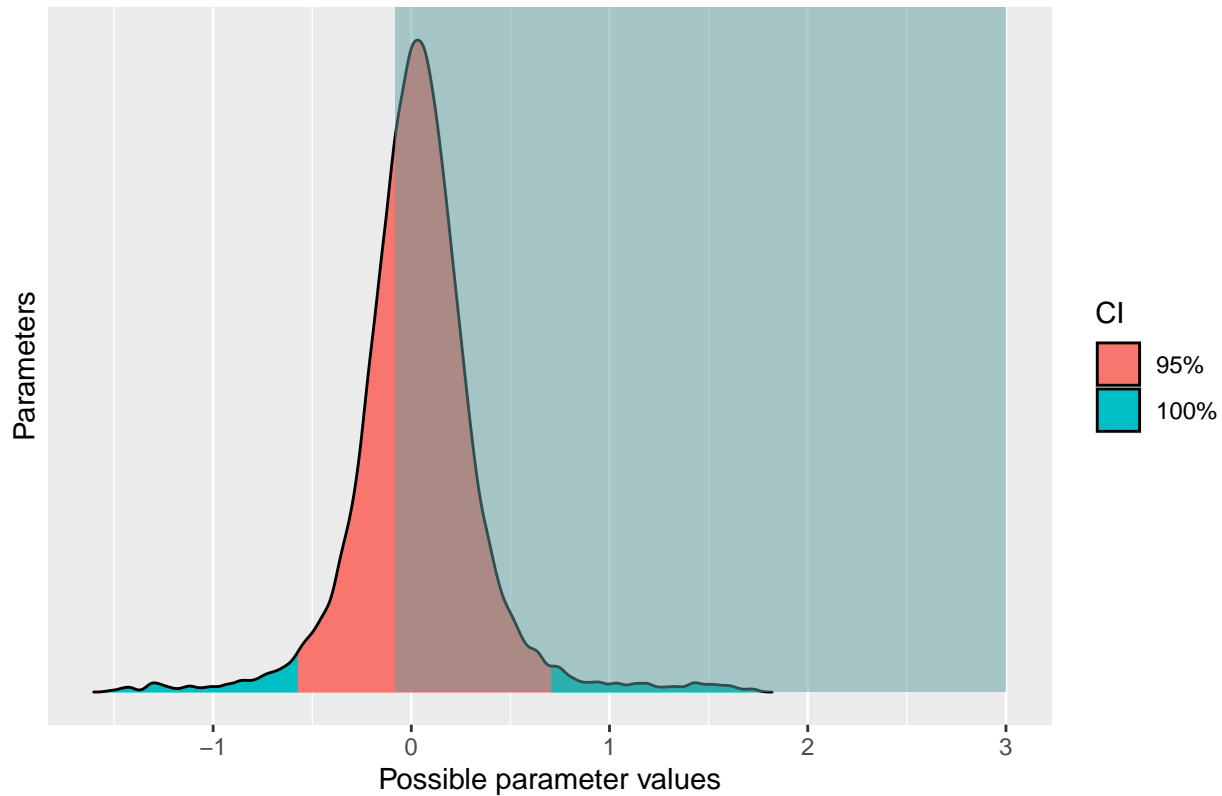
```
##
## Parameter      | Median |      95% CI |    pd |      ROPE | % in ROPE | Rhat |    ESS
## -----
## (Intercept)    |  6.28 | [ 5.24, 7.32] | 100% | [-0.08, 0.08] |      0% | 1.002 | 1958.00
## GeneratedByLLM |  0.03 | [-0.58, 0.71] | 55.31% | [-0.08, 0.08] | 30.70% | 1.002 | 1004.00
```

3.5 ROPE and non-inferiority results

We defined the non-inferiority similar to the NLP above. In this case, the range is posteriors > -0.08 . We defined the upper range as 3 as all values are lower than this number.

```
rope_result <- rope(BayesianModel, range = c(-0.08,3), ci = 0.95) # Define a practical equivalence range
plot(rope_result)
```

Region of Practical Equivalence (ROPE)



```
print(rope_result)
```

```
## # Proportion of samples inside the ROPE [-0.08, 3.00]:
```

```
##
## Parameter      | inside ROPE
## -----
## Intercept      |      0.00 %
## GeneratedByLLM |     69.60 %
```

69.60% of the posterior distribution falling within this range—indicating support for the non-inferiority of LLM-generated content relative to human-generated text.