Final Project

Onyul Haque

4/25/2022

```
library(tidyr)
library(dplyr)
library(reshape2)
library(car)
library(compute.es)
library(effects)
library(multcomp)
library(pastecs)
library(psych)
library(ggplot2)
library(gmodels)
library(mlogit)
library(Hmisc)
library(brms)
library(here)  # for easier management of file paths
library(rstan)
rstan_options(auto_write = TRUE)  # save compiled Stan object
library(brms)  # simplify fitting Stan GLM models
library(shinystan)  # graphical exploration
library(posterior)  # for summarizing draws
library(bayesplot)  # for plotting
library(modelsummary)  # table for brms
library(ProbBayes)
theme_set(theme_classic() +
    theme(panel.grid.major.y = element_line(color = "grey92")))
```

Research Question

Do people trust less when there is a timer that can cut them off from making a and is this uniquely due to the fact that the origin of risk is a person and not nature?

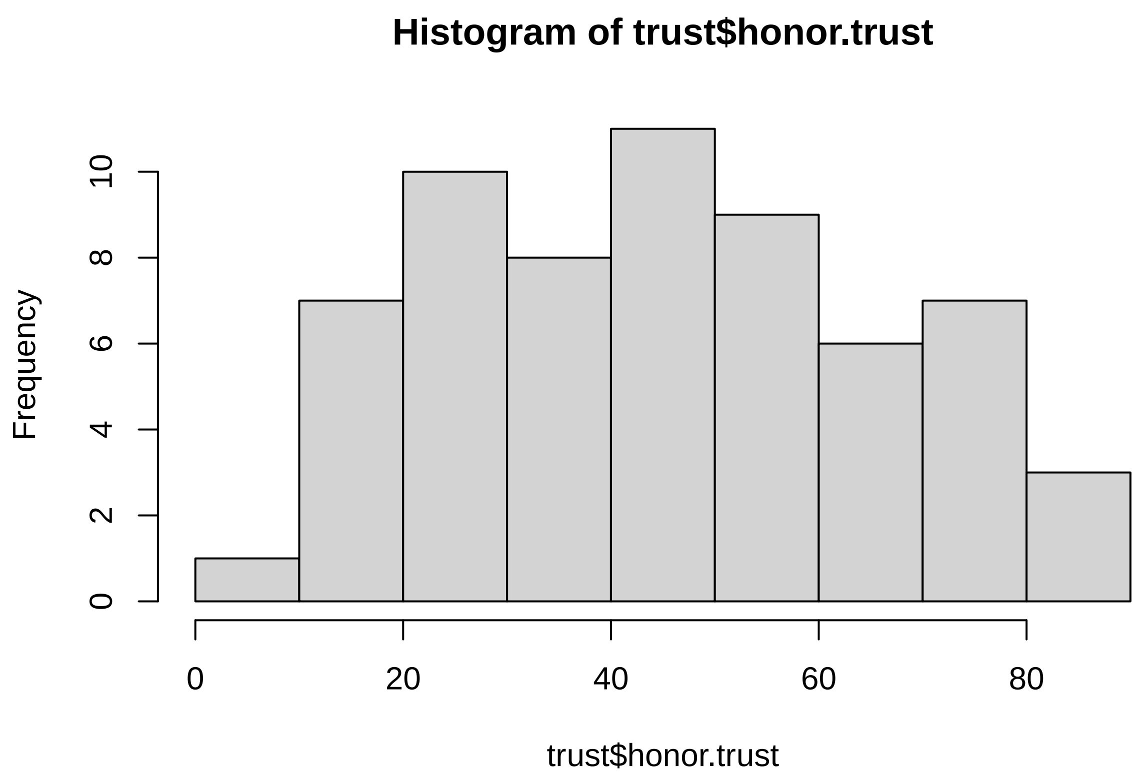
```
#Gather ID's of people who passed Lottery Timer Comp Questions
lt.pass <- trust$id[c(trust$lt.cc.duration == 30 &
                     trust$lt.cc.end == "False" &
                     trust$lt.cc.random == "True" &
                     trust$lt.cc.choice == "False")]
lt.pass <- na.omit(lt.pass)


#Gather ID's of people who passed Trust Timer Comp Questions
```

```r
#Pilot Data Collection
#tt.pass <- trust$id[c(trust$tt.cc.duration == 30  &
                   #   trust$tt.cc.end == "False" &
                    #  trust$tt.cc.random == "True" &
                    #  trust$tt.cc.choice == "False")]


## The Data COllection
tt.pass <- trust$id[c(trust$tt.cc.duration == 30  &
                      trust$tt.cc.end == "False" &
                      trust$tt.cc.random == "True" &
                      trust$tt.cc.choice == "False" &
                      trust$tt.cc.PersonB == "False" )]


tt.pass <- na.omit(tt.pass)

#Gather ID's of people who passed in No timer Conditions
lnt.pass <- trust$id[trust$lnt.decision == "Option 1" | trust$lnt.decision == "Option 2"]
lnt.pass <- na.omit(lnt.pass)

tnt.pass <- trust$id[trust$tnt.decision == "Option 1" | trust$tnt.decision == "Option 2"]
tnt.pass <- na.omit(tnt.pass)

valid.id <- c(lt.pass, tt.pass, lnt.pass, tnt.pass)

trust <- trust[c(trust$id %in% valid.id),]

#trust long subset for only the relevant variable in Analysis

trustlong <- trust[, c("lt.instruction.submit.page","lt.decision", "lt.decision.first.click",
  "lt.decision.last.click", "lt.decision.page.submit","lt.decision.click.count", "tt.instruction.submit
  "tt.decision", "tt.decision.first.click",
  "tt.decision.last.click", "tt.decision.page.submit", "tt.decision.click.count","lnt.instruction.submi
  "lnt.decision", "lnt.decision.first.click",
  "lnt.decision.last.click", "lnt.decision.page.submit", "lnt.decision.click.count","tnt.instruction.su
  "tnt.decision", "tnt.decision.first.click",
  "tnt.decision.last.click", "tnt.decision.page.submit", "tnt.decision.click.count", "id", "length")]

#Standardized Covariate
#Create standardize function
standardize <- function(variable){
  (variable - mean(variable, na.rm = TRUE))/sd(variable, na.rm = TRUE)
}

trustlong$z.lt.instruction.timer <- standardize(trustlong$lt.instruction.submit.page)
trustlong$z.tt.instruction.timer <- standardize(trustlong$tt.instruction.submit.page)
trustlong$z.lnt.instruction.timer <- standardize(trustlong$lnt.instruction.submit.page)
trustlong$z.tnt.instruction.timer <- standardize(trustlong$tnt.instruction.submit.page)

#Melt Data For Analysis
trustlong <- trustlong %>%
  gather(key = "condition", value = "decision",
```

```
        c(lt.decision, tt.decision, lnt.decision, tnt.decision)) %>%
  gather(key = "fc.condition", value = "fc.decision.sec",
         contains("decision.first")) %>%
  gather(key = "lc.condition", value = "lc.decision.sec",
         contains("decision.last")) %>%
  gather(key = "submit.condition", value = "submit.decision.sec",
         contains("decision.page")) %>%
  gather(key = "cc.condition", value = "cc.decision",
         contains("decision.click")) %>%
  gather(key = "instruction.condition", value = "instruction.time",
         contains("instruction.submit")) %>%
  gather(key = "z.covariate", value = "z.instruction.time",
         c(z.lt.instruction.timer, z.tt.instruction.timer, z.lnt.instruction.timer, z.tnt.instruction.t

#Gather complete cases
trustlong <- na.omit(trustlong)

#Re-categorize dependent for simplicity sake - anybody who took longer than 30
#seconds can be considered as Did not give - can do secondary analysis later
trustlong <- within(trustlong, decision[submit.decision.sec >= 30] <- "Cutoff")
trustlong$decision <- ifelse(trustlong$decision == "Option 2", "Give $5", "Did not Give $5")
trustlong$decision <- as.factor(trustlong$decision)
trustlong$decision <- as.numeric(trustlong$decision)

#Create Columns to track indepednet effect of Game and Timer
trustlong$game <- ifelse(trustlong$condition == "tt.decision" |
                           trustlong$condition == "tnt.decision",
                         yes = "Trust",
                         no = "Lottery")
trustlong$timer <- ifelse(trustlong$condition == "tt.decision" |
                           trustlong$condition == "lt.decision",
                         yes = "Timer",
                         no = "No Timer")
trustlong$game <- as.factor(trustlong$game)
trustlong$timer <- as.factor(trustlong$timer)

levels(trustlong$game)
levels(trustlong$timer)
```

Manipulation Check

```
hist(trust$honor.trust)
```




```r
t.test(trust$honor.trust, mu = 45)
```

```
##
##  One Sample t-test
##
## data:  trust$honor.trust
## t = 1.1813, df = 61, p-value = 0.2421
## alternative hypothesis: true mean is not equal to 45
## 95 percent confidence interval:
##  42.76529 53.68633
## sample estimates:
## mean of x
##  48.22581
```

```r
#Break it down by timer conditions
tt.df <-  trust[complete.cases(trust$tt.id),]
tnt.df <-  trust[complete.cases(trust$tnt.id),]
mean(tt.df$honor.trust, na.rm = T)
```

```
## [1] 49.5
```

```r
mean(tnt.df$honor.trust, na.rm = T)
```

```
## [1] 47.30556
```

```r
t.test(tt.df$honor.trust, mu = 45)
```

```
##
##  One Sample t-test
##
## data:  tt.df$honor.trust
## t = 1, df = 25, p-value = 0.3269
## alternative hypothesis: true mean is not equal to 45
## 95 percent confidence interval:
##  40.23208 58.76792
## sample estimates:
## mean of x
##      49.5
```

```r
t.test(tnt.df$honor.trust, mu = 45)
```

```
##
##  One Sample t-test
##
## data:  tnt.df$honor.trust
## t = 0.66894, df = 35, p-value = 0.5079
## alternative hypothesis: true mean is not equal to 45
## 95 percent confidence interval:
##  40.30858 54.30253
## sample estimates:
## mean of x
##  47.30556
```

```r
#How many people were cutoff by game
lt.df <-  trust[complete.cases(trust$lt.decision.page.submit),]
sum(lt.df$lt.decision.page.submit >= 30, na.rm = T)
```

```
## [1] 0
```

```r
sum(tt.df$tt.decision.page.submit >= 30, na.rm = T)/length(tt.df$tt.decision.page.submit)
```

```
## [1] 0.1071429
```

```r
prior.logit.transformation <- function(lower.bound, upperbound){
  logit.p <- qlogis(c(lower.bound, upperbound))
new.m <- round(max(logit.p)-(abs(diff(logit.p))/2), digits = 2)
new.sd <- round((abs(diff(logit.p))/4), digits = 2)
return(list("m" = new.m, "sd" = new.sd))
}
```

```r
#Prior For Intercept- Most people are risk averse (prospect theory).
#This suggest people will not gamble too much in the lottery game
#I believe the prior probability will be around 35% with sd of .10

prior.logit.transformation(.15,.55)
```

```
## $m
## [1] -0.77
##
## $sd
## [1] 0.48
```

```
#Prior For Beta 1 - effect of game
#Expect a 30% increase when trust game because prior studies suggest that about
#65% of people trust. Ranges from 60-70
prior.logit.transformation(.60,.70)
```

```
## $m
## [1] 0.63
##
## $sd
## [1] 0.11
```

```
#Prior for beta 2 - Effect of time
#Expect a decrease of about %10 [-.20, 0.0]
prior.logit.transformation(.15,.35)
```

```
## $m
## [1] -1.18
##
## $sd
## [1] 0.28
```

```
#Prior for beta 3 - interaction
#Expect a decrease of about %10 [-20,0.0]
prior.logit.transformation(.40,.60)
```

```
## $m
## [1] 0
##
## $sd
## [1] 0.2
```

Bayesian Analysis - Give vs Do not Give

Model Equation:

$$Decision_i = Bin(N, \theta)$$
$$\theta = \beta_o + \beta_1 game_i + \beta_2 timer_i + \beta_3 game_i * timer_i$$

Prior:

Lottery No Timer

$$\beta_0 = N(-.77, .48)$$

Difference between Lottery No timer + Trust No Timer

$$\beta_1 = N(.63, .11)$$

Average difference between Lottery Timer + Lottery no Timer conditions

$$\beta_2 = N(-1.18, .28)$$

How the effect of game and timer changes when both together (trust timer)

$$\beta_3 = N(0, .2)$$

```
m1 <-brm(
    # Y (vote) = beta0 + beta1 (growth)
    decision ~ game * timer,
    data = trustlong,
    # Normal distribution with identity link
    family = binomial,
    # Overwrite the default priors
    prior = c(
        # prior for beta0
        prior(normal(-.77,.48), class = "Intercept"),
        # prior for beta1 game
        prior(normal(.63,.11), class = "b", coef = "gameTrust"),
        #prior Beta 2 Time
        prior(normal(-1.18,.28), class = "b", coef = "timerTimer"),
        #Beta 3 Interaction Term
        prior(normal(0,.2), class = "b", coef = "gameTrust:timerTimer")
    ),
    sample_prior = TRUE,  # also sample the prior distributions
    iter = 4000,  # default is 4 chains, 2000 iterations each
    seed = 21
)
```

```
## Using the maximum response value as the number of trials.

## Compiling Stan program...

## Start sampling

##
## SAMPLING FOR MODEL '30eef63e7c9a47fb51416bd86f2de8a2' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
```

```
## Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.407 seconds (Warm-up)
## Chain 1:                0.412 seconds (Sampling)
## Chain 1:                0.819 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '30eef63e7c9a47fb51416bd86f2de8a2' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.407 seconds (Warm-up)
## Chain 2:                0.411 seconds (Sampling)
## Chain 2:                0.818 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '30eef63e7c9a47fb51416bd86f2de8a2' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3:
```

```
## Chain 3:  Elapsed Time: 0.413 seconds (Warm-up)
## Chain 3:                0.474 seconds (Sampling)
## Chain 3:                0.887 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '30eef63e7c9a47fb51416bd86f2de8a2' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.417 seconds (Warm-up)
## Chain 4:                0.493 seconds (Sampling)
## Chain 4:                0.91 seconds (Total)
## Chain 4:
```

```
#Exponentiate Values for Odds Ratios
##Model Summary
msummary(m1, statistic = "conf.int", fmt = 2, exponentiate = T)
```

```
## Using the maximum response value as the number of trials.
```

```
## Using the maximum response value as the number of trials.
## Using the maximum response value as the number of trials.
## Using the maximum response value as the number of trials.
## Using the maximum response value as the number of trials.
## Using the maximum response value as the number of trials.
## Using the maximum response value as the number of trials.
```

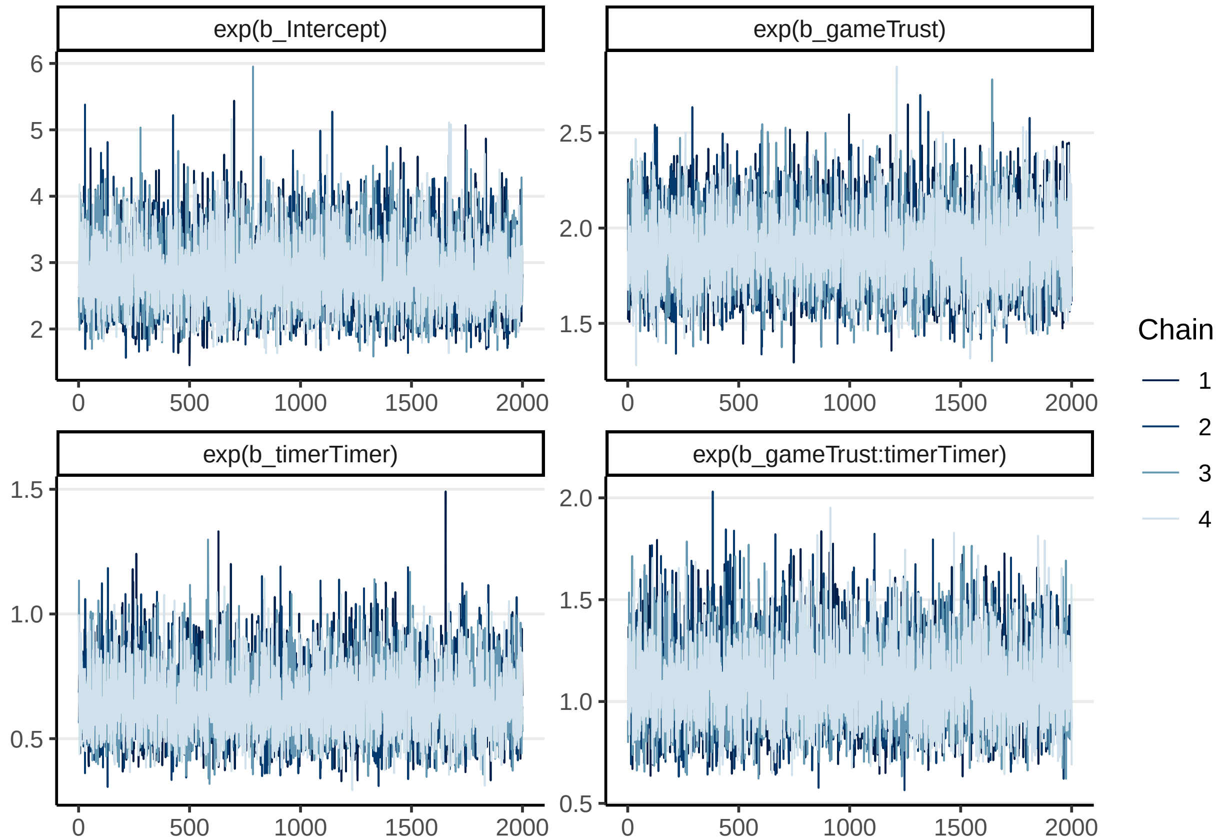
```
m1
```

```
##  Family: binomial
##   Links: mu = logit
## Formula: decision ~ game * timer
##    Data: trustlong (Number of observations: 132)
##   Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
##          total post-warmup draws = 8000
##
```

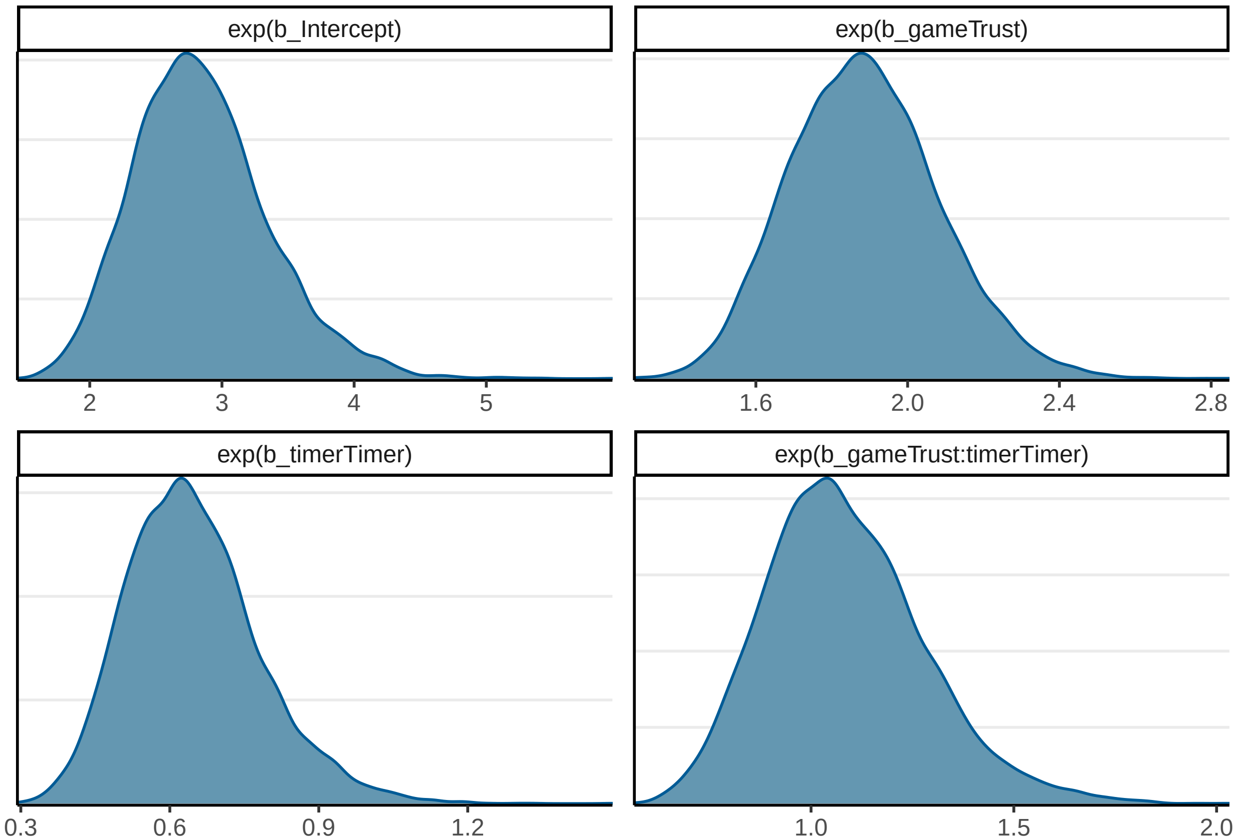
	Model 1
b_Intercept	2.80
	[1.96, 3.95]
b_gameTrust	1.88
	[1.54, 2.30]
b_timerTimer	0.64
	[0.42, 0.95]
b_gameTrust × timerTimer	1.06
	[0.76, 1.53]
Num.Obs.	132
R2	0.084
ELPD	−100.9
ELPD s.e.	3.4
LOOIC	201.9
LOOIC s.e.	6.8
WAIC	201.9
RMSE	1.04

```
## Population-Level Effects:
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept               1.03      0.18     0.68     1.38 1.00     8489
## gameTrust               0.63      0.10     0.43     0.83 1.00     7979
## timerTimer             -0.45      0.21    -0.86    -0.05 1.00     7921
## gameTrust:timerTimer    0.06      0.18    -0.29     0.42 1.00     7858
##                     Tail_ESS
## Intercept               6265
## gameTrust               6355
## timerTimer              5903
## gameTrust:timerTimer    6484
##
## 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).
```

```r
#Convergent Plot
mcmc_trace(m1,
           pars = c("b_Intercept", "b_gameTrust",
                    "b_timerTimer", "b_gameTrust:timerTimer"),
           transformations = "exp")
```



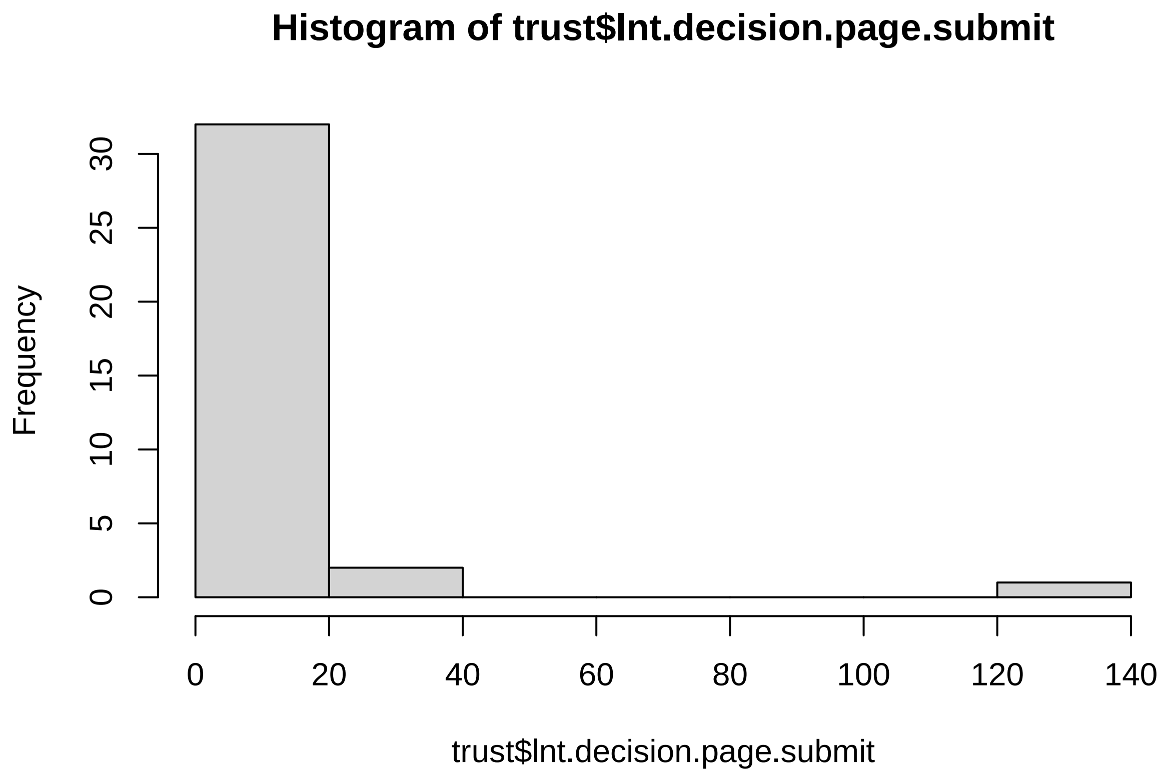

```r
#Posterior Distribution
mcmc_dens(m1,
          pars = c("b_Intercept", "b_gameTrust",
                   "b_timerTimer", "b_gameTrust:timerTimer"),
           transformations = "exp")
```



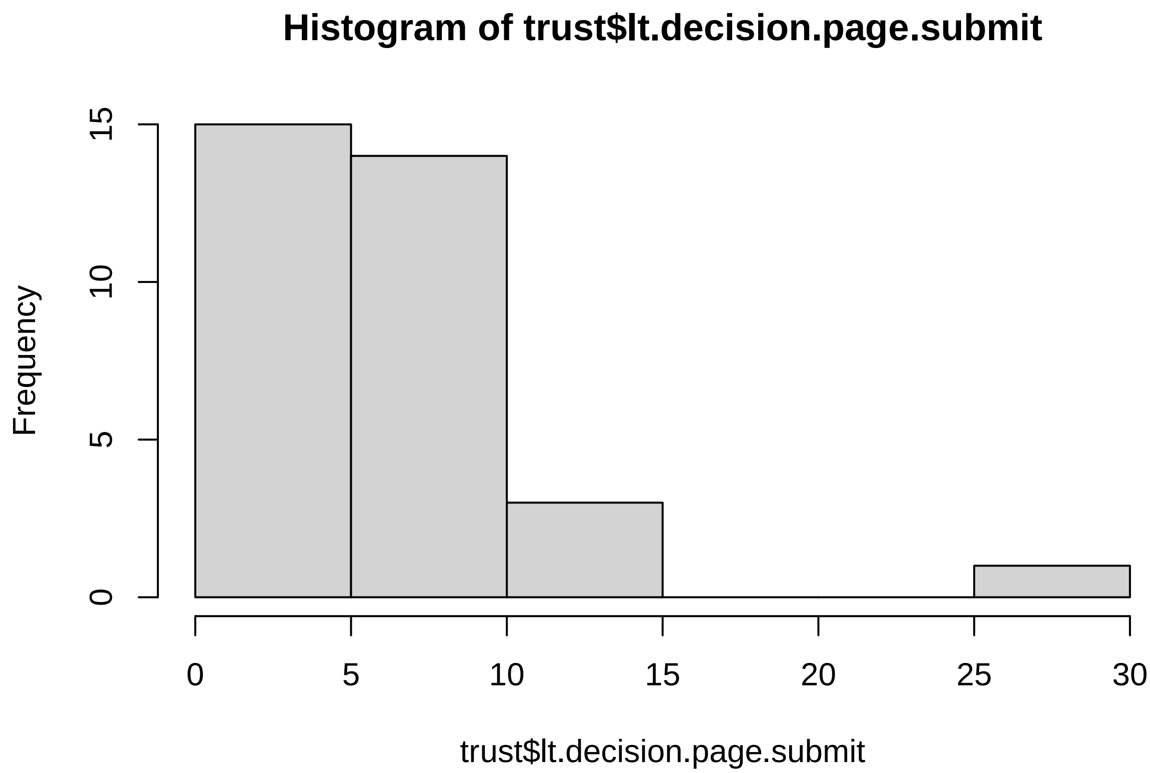
I exponetiated the results of the model to obtain the odds ratio. The only the only paramters to obtain a confidence interval not including 1 are intercept and the game effect. Thus, the odds that somebody in the trust game would give 5 dollars is 2.32 (95 confidence interval 1.17,4.17) higher than somebody in the lottery game.

Bayes Analysis Model 2 - Time to Decide

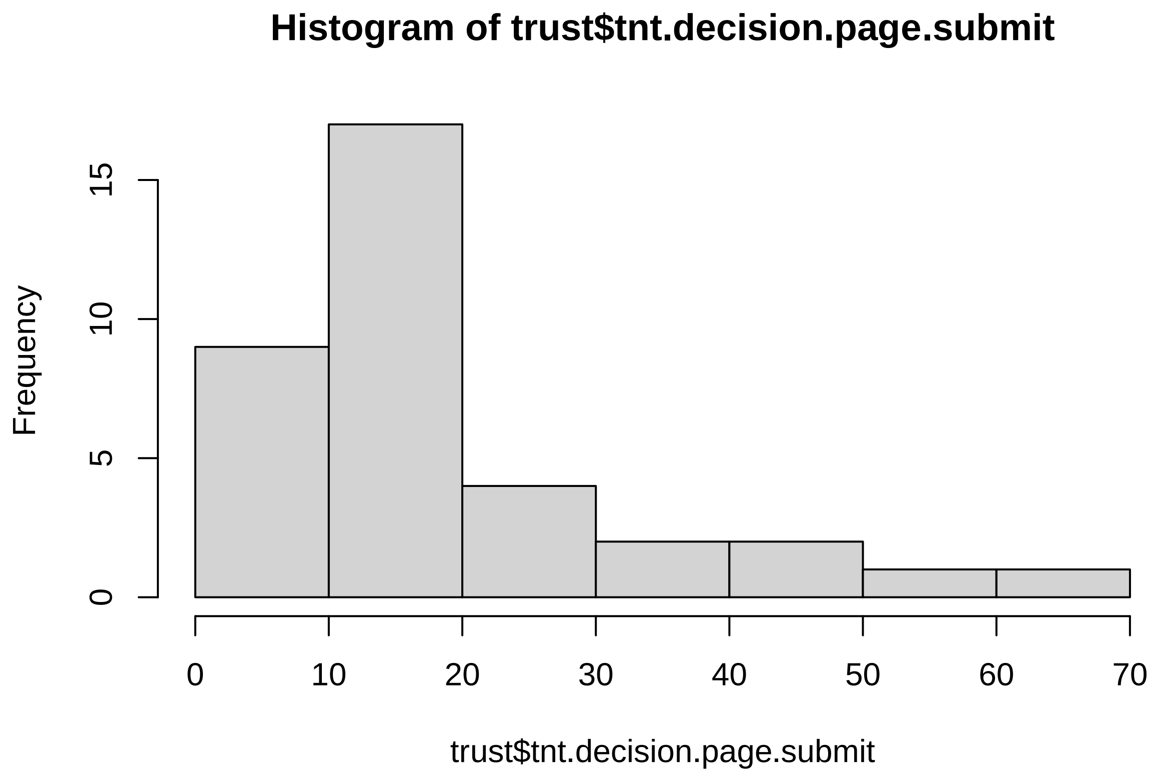
```
#HIstograms Raw
hist(trust$lnt.decision.page.submit)
```




```
hist(trust$lt.decision.page.submit)
```

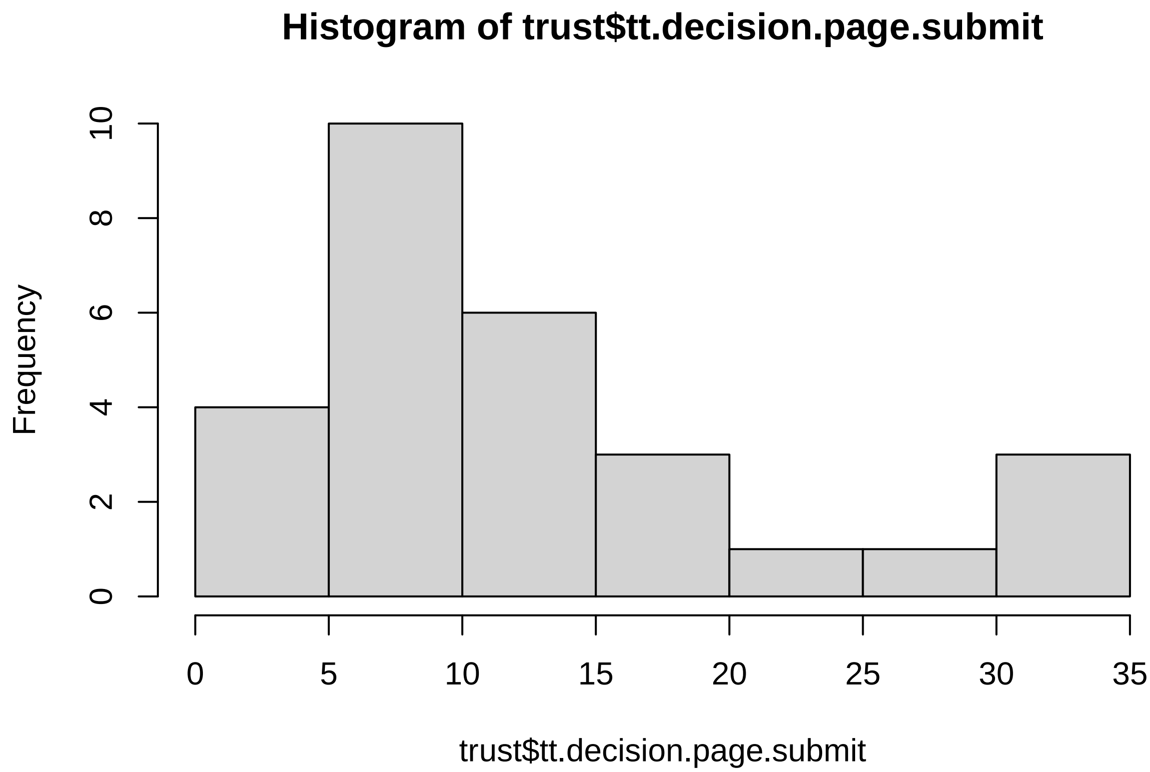



```
hist(trust$tnt.decision.page.submit)
```




```
hist(trust$tt.decision.page.submit)
```

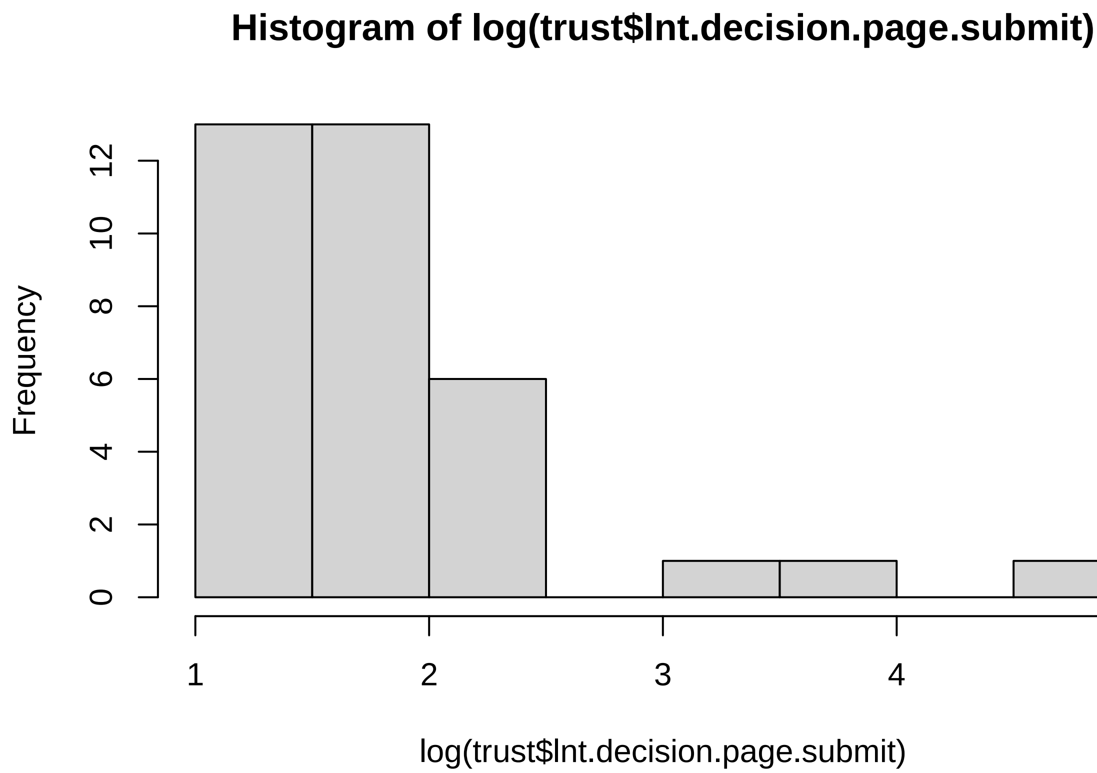
	lnt.decision		lt.decision		tnt.decision		tt.decision	
	mean	sd	mean	sd	mean	sd	mean	sd
submit.decision.sec	10.36	22.04	6.38	4.15	18.94	14.21	13.19	8.27



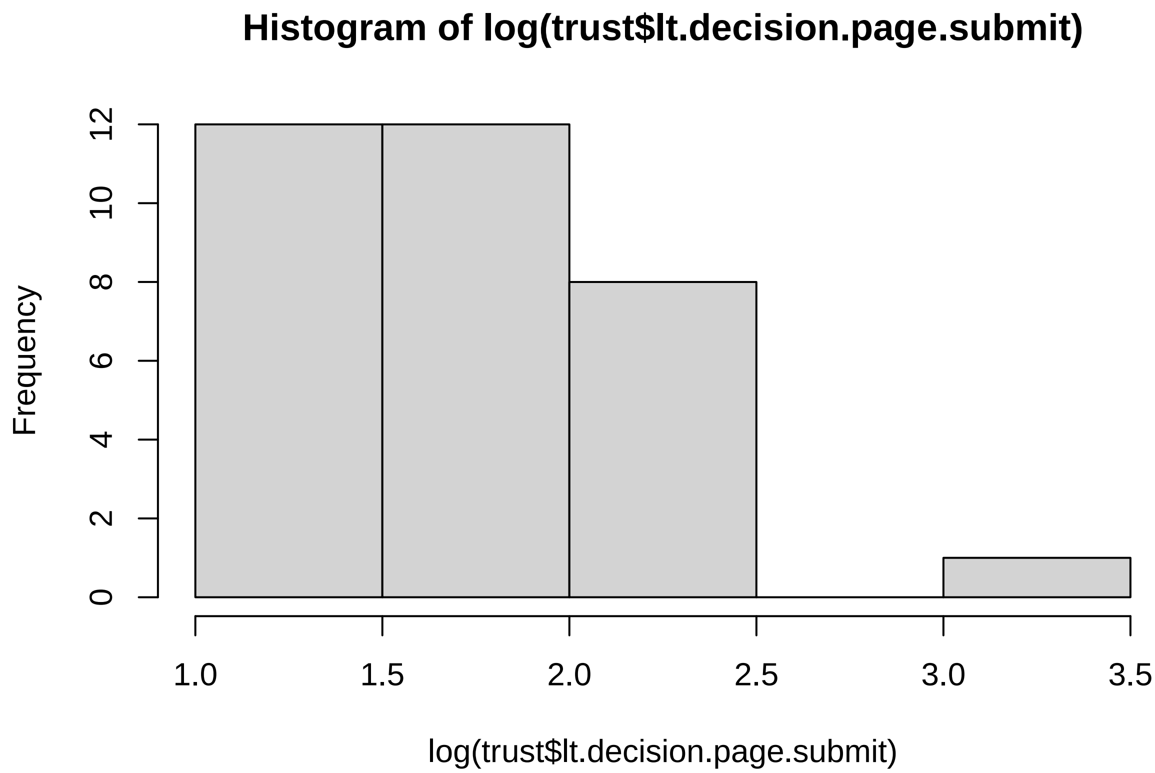

```
# Data summary for DV ~ conditions
datasummary(submit.decision.sec ~ condition * (mean + sd), data = trustlong)
```

Since the data is heavily skewed we need to log transform it

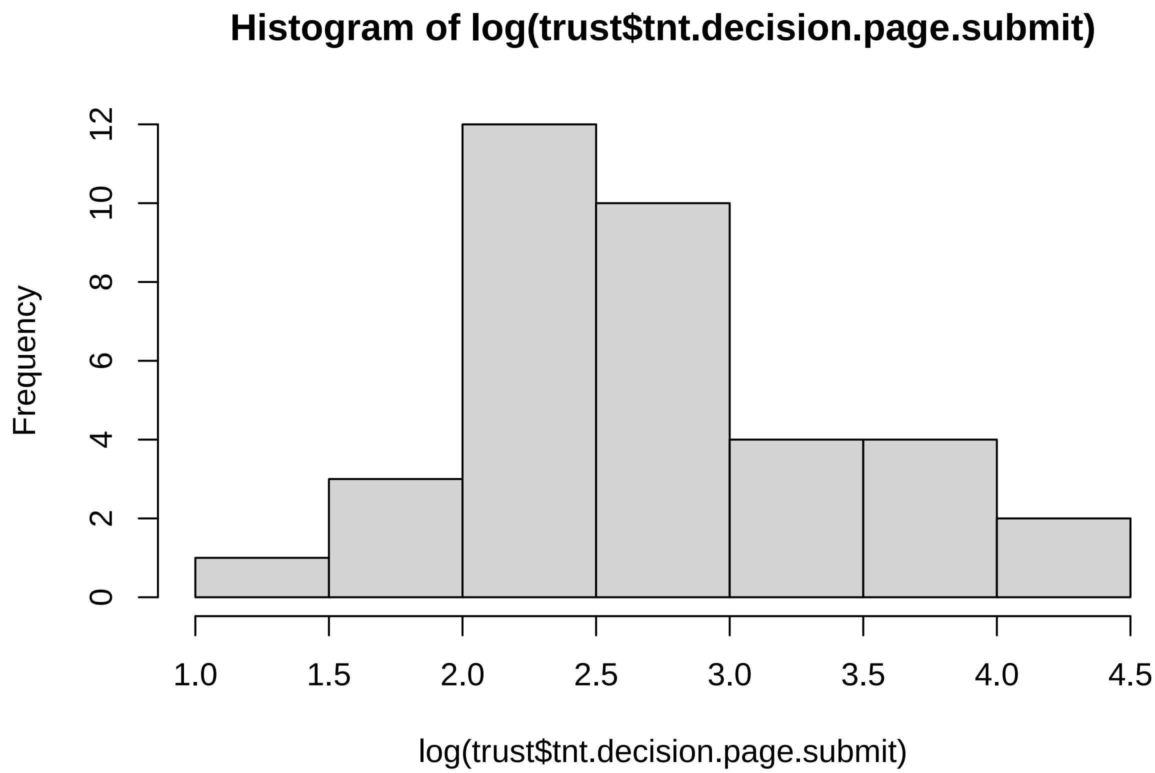
```
#HIstograms after log transformation
hist(log(trust$lnt.decision.page.submit))
```




```
hist(log(trust$lt.decision.page.submit))
```

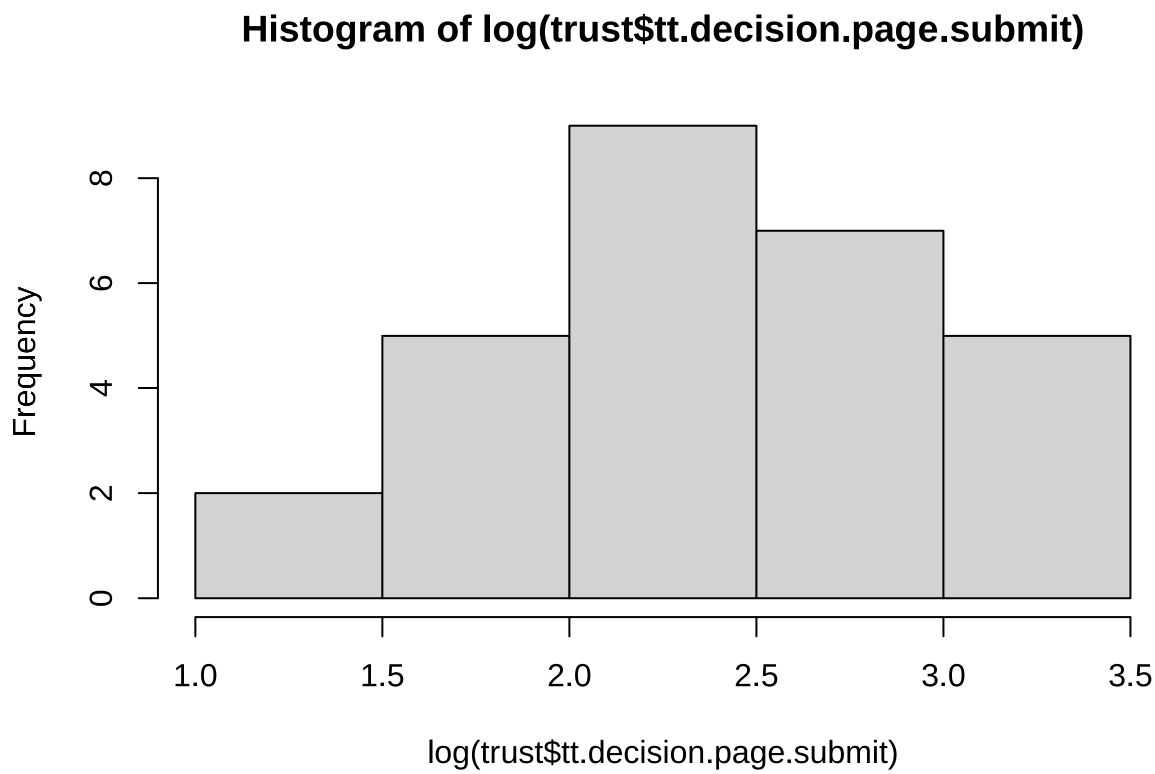



```
hist(log(trust$tnt.decision.page.submit))
```




```
hist(log(trust$tt.decision.page.submit))
```

	lnt.decision		lt.decision		tnt.decision		tt.decision	
	mean	sd	mean	sd	mean	sd	mean	sd
log(submit.decision.sec)	1.79	0.78	1.72	0.49	2.71	0.67	2.39	0.63




```r
# Data summary for DV ~ conditions
datasummary(log(submit.decision.sec) ~ condition * (mean + sd), data = trustlong)
```

Model Equation:

$$Time_i = Lognormal(\mu, \sigma)$$
$$\mu = \beta_o + \beta_1 game_i + \beta_2 timer_i + \beta_3 game_i * timer_i$$

Prior:

Lottery No Timer

$$\beta_0 = N(0, 1)$$

Difference between Lottery No timer and Trust No Timer

$$\beta_1 = N(0, 1)$$

Difference between Lottery No Timer - Lottery Timer

$$\beta_2 = N(0, 1)$$

Difference between Lottery Timer - Trust Timer beyond the difference of Lottery no Timer and Trust No Timer

$$\beta_3 = N(0, 1)$$

$$\sigma = t_5(0, .2)$$

```
m2 <-brm(
    # Y (vote) = beta0 + beta1 (growth)
    log(submit.decision.sec) ~ game * timer,
    data = trustlong,
    # Normal distribution with identity link
    family = gaussian(link = "identity"),
    # Overwrite the default priors
    prior = c(
        # prior for beta0
        prior(normal(0,1), class = "Intercept"),
        # prior for beta1 game
        prior(normal(0,1), class = "b", coef = "gameTrust"),
        #prior Beta 2 Time
        prior(normal(0,1), class = "b", coef = "timerTimer"),
        #Beta 3 Interaction Term
        prior(normal(0,1), class = "b", coef = "gameTrust:timerTimer"),
        # Residuals
        prior(student_t(5,0,.5), class = "sigma")
    ),
    sample_prior = TRUE,  # also sample the prior distributions
    iter = 4000,  # default is 4 chains, 2000 iterations each
    seed = 21
)
```

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

```
## Start sampling
```

```
##
## SAMPLING FOR MODEL '119c63018d1f60b6faae0faf90e1ac8e' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 1: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 1: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 1: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 1: Iteration: 3200 / 4000 [ 80%]  (Sampling)
```

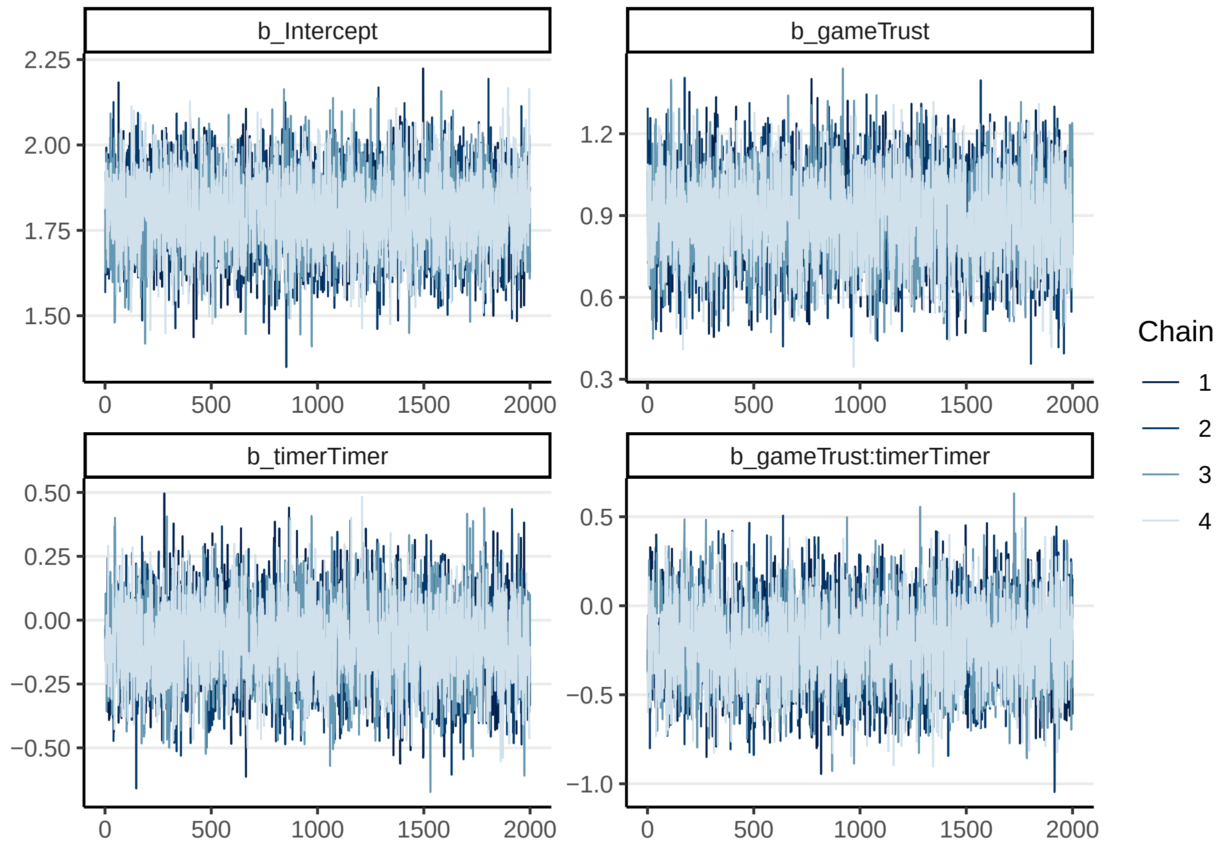
```
## Chain 1: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 1: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.062 seconds (Warm-up)
## Chain 1:                0.065 seconds (Sampling)
## Chain 1:                0.127 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '119c63018d1f60b6faae0faf90e1ac8e' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 2: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 2: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 2: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 2: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 2: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 2: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 2: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 2: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.063 seconds (Warm-up)
## Chain 2:                0.062 seconds (Sampling)
## Chain 2:                0.125 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '119c63018d1f60b6faae0faf90e1ac8e' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 3: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 3: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 3: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 3: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 3: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 3: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 3: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 3: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 3: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 3: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.064 seconds (Warm-up)
```

	Model 1
b_Intercept	1.80
	[1.60, 2.02]
b_gameTrust	0.89
	[0.61, 1.20]
b_timerTimer	−0.09
	[−0.39, 0.21]
b_gameTrust × timerTimer	−0.21
	[−0.65, 0.22]
sigma	0.65
	[0.58, 0.73]
Num.Obs.	132
R2	0.299
ELPD	−134.9
ELPD s.e.	14.1
LOOIC	269.7
LOOIC s.e.	28.2
WAIC	269.6
RMSE	18.12

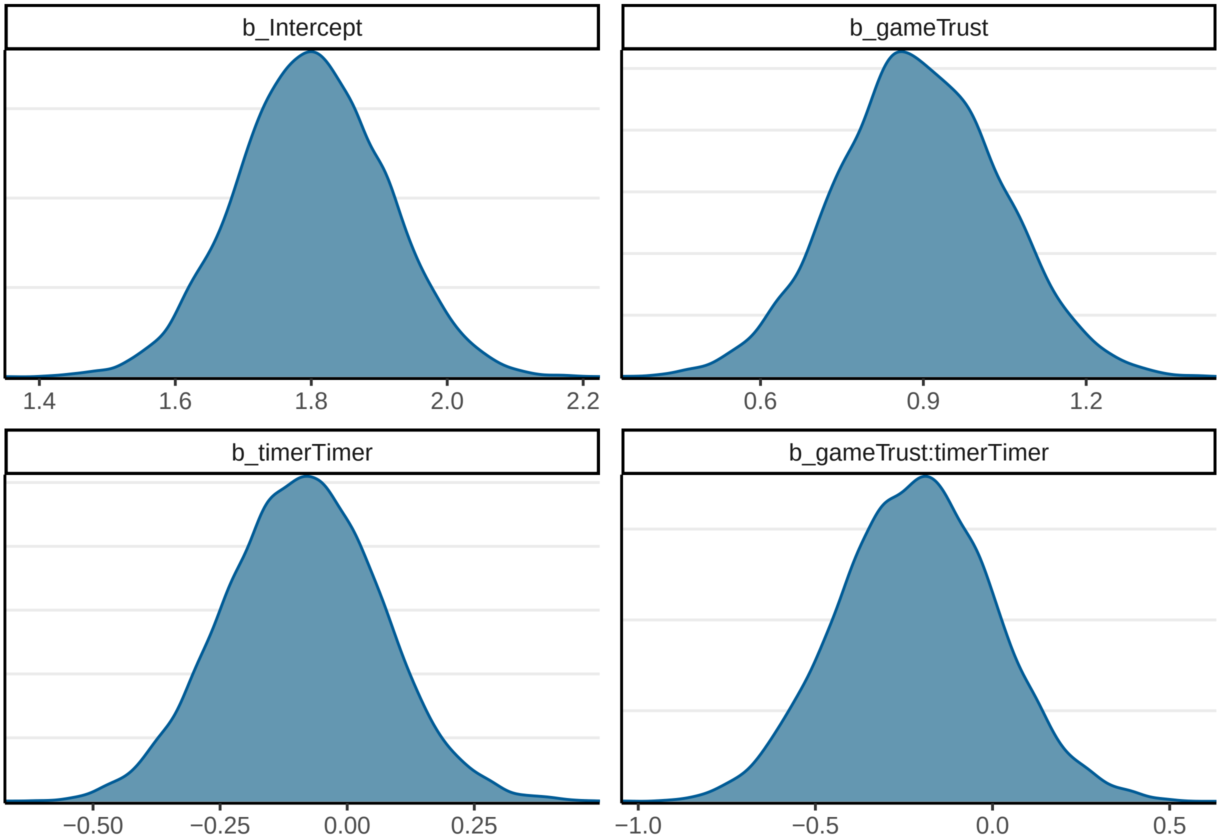
```
## Chain 3:                0.069 seconds (Sampling)
## Chain 3:                0.133 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL '119c63018d1f60b6faae0faf90e1ac8e' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 4: Iteration:  400 / 4000 [ 10%]  (Warmup)
## Chain 4: Iteration:  800 / 4000 [ 20%]  (Warmup)
## Chain 4: Iteration: 1200 / 4000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1600 / 4000 [ 40%]  (Warmup)
## Chain 4: Iteration: 2000 / 4000 [ 50%]  (Warmup)
## Chain 4: Iteration: 2001 / 4000 [ 50%]  (Sampling)
## Chain 4: Iteration: 2400 / 4000 [ 60%]  (Sampling)
## Chain 4: Iteration: 2800 / 4000 [ 70%]  (Sampling)
## Chain 4: Iteration: 3200 / 4000 [ 80%]  (Sampling)
## Chain 4: Iteration: 3600 / 4000 [ 90%]  (Sampling)
## Chain 4: Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 0.062 seconds (Warm-up)
## Chain 4:                0.064 seconds (Sampling)
## Chain 4:                0.126 seconds (Total)
## Chain 4:
```

```
# Summary of Model
msummary(m2, statistic = "conf.int", fmt = 2,)
```

```
#Convergence Plot
mcmc_trace(m2,
           pars = c("b_Intercept", "b_gameTrust",
                    "b_timerTimer", "b_gameTrust:timerTimer"))
```




```
#Posterior Distribution
mcmc_dens(m2,
          pars = c("b_Intercept", "b_gameTrust",
                   "b_timerTimer", "b_gameTrust:timerTimer"))
```



Only significant effect was that of game.