Assignment 2

Bayesian Data Analysis and Models of Behavior

Wenjie Tu

Spring Semester 2022

Contents

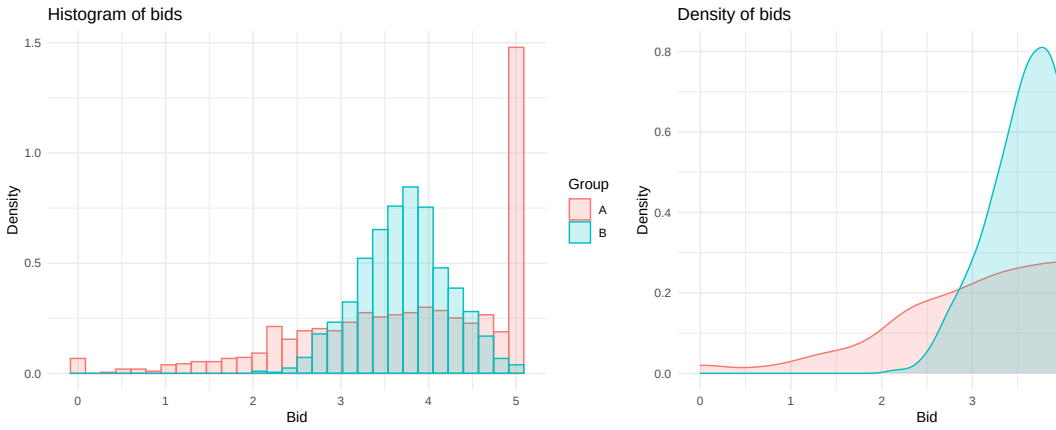
Part 1

```
# Read in data for willingness to pay
d.wtp <- read.csv("./data/wtp_data.csv")
head(d.wtp)
```

```
##   Subject Group Food  Bid
## 1       1     A    1 2.19
## 2       1     A    2 4.42
## 3       1     A    3 5.00
## 4       1     A    4 0.48
## 5       1     A    5 4.64
## 6       1     A    6 4.76
```

```
library(ggplot2)
library(GGally)
library(rjags)
library(runjags)
library(HDInterval)
```

```
ggplot(d.wtp, aes(x=Bid, y=..density.., color=Group, fill=Group)) +
  geom_histogram(position="identity", bins=30, alpha=0.2) +
  labs(title="Histogram of bids", y="Density") + theme_minimal()

ggplot(d.wtp, aes(x=Bid, y=..density.., color=Group, fill=Group)) +
  geom_density(alpha=0.2) + labs(title="Density of bids", y="Density") +
  theme_minimal()
```



Histogram and density plot

```
# Write in JAGS model
modelString <- "model{
  for (g in 1:Ngrps) {
    # group-specific priors
    grp.mu[g] ~ dnorm(0, 1.0E-5)
    grp.sig[g] ~ dunif(1.0E-6, 100)

    # prior for the standard deviation of the response
    sig[g] ~ dunif(1.0E-5, 100)

    for (s in 1:Nsubj[g]) {
      subj.mu[s, g] ~ dnorm(grp.mu[g], 1/grp.sig[g]^2)
    }
  }

  for (i in 1:Ntotal) {
    y[i] ~ dnorm(subj.mu[subIdx[i], grpIdx[i]], 1/sig[grpIdx[i]]^2)
  }
}"

writeLines(modelString, con="./models/StandardModel.txt")
```

```
y <- d.wtp$Bid
Ntotal <- length(y)
```

```
subIdx <- d.wtp$Subject
grpIdx <- as.numeric(as.factor(d.wtp$Group))
Ngrps <- length(unique(d.wtp$Group))
Nsubj <- c(length(unique(d.wtp$Subject)), length(unique(d.wtp$Subject)))
```

```
n.sample <- 2000
n.burnin <- 1000
n.thin <- 2
n.chains <- 3

dat.standard <- dump.format(list(y = y,
                                 Ntotal = Ntotal,
                                 subIdx = subIdx,
                                 grpIdx = grpIdx,
                                 Ngrps = Ngrps,
                                 Nsubj = Nsubj))

# Let JAGS initialize
inits1 <- dump.format(list(.RNG.name="base::Super-Duper", .RNG.seed=99999))
inits2 <- dump.format(list(.RNG.name="base::Wichmann-Hill", .RNG.seed=1234))
inits3 <- dump.format(list(.RNG.name="base::Mersenne-Twister", .RNG.seed=6666))

# Tell JAGS which latent variables to monitor
monitor = c("grp.mu", "grp.sig", "sig")
```

```
results.standard <- run.jags(model = "./models/StandardModel.txt",
                             monitor = monitor,
                             data = dat.standard,
                             n.chains = n.chains,
                             inits = c(inits1, inits2, inits3),
                             plots = FALSE,
                             burnin = n.burnin,
                             sample = n.sample,
                             thin = n.thin)
```

Summary tables and correlation plots

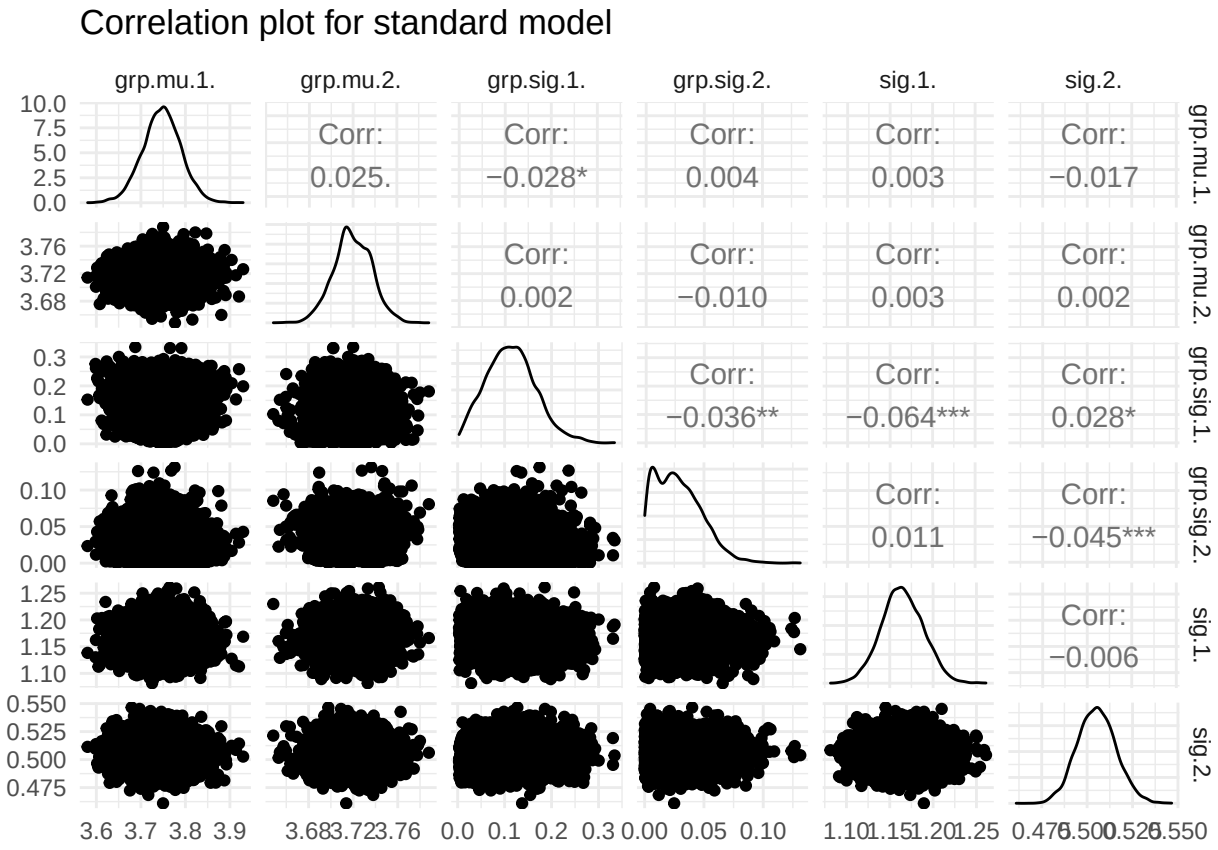
```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 1000 iterations...
## Burning in the model for 1000 iterations...
## Running the model for 4000 iterations...
## Simulation complete
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 6 variables....
## Finished running the simulation
```

```
knitr::kable(summary(results.standard), align="c", caption="Summary results for the standard model")
```

Table 1: Summary results for the standard model

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.20	psrf
grp.mu[1]	3.6643338	3.7475143	3.8329333	3.7467693	0.0430744	NA	0.0009642	2.2	1996	0.0206805	1.001651
grp.mu[2]	3.6854814	3.7188868	3.7523388	3.7193982	0.0170116	NA	0.0006991	4.1	592	0.1795134	1.002851
grp.sig[1]	0.0102595	0.1116116	0.2099225	0.1137392	0.0525471	NA	0.0023477	4.5	501	0.2258151	1.004847
grp.sig[2]	0.0000735	0.0262768	0.0630370	0.0285976	0.0193892	NA	0.0012582	6.5	237	0.4611961	1.007699
sig[1]	1.1149633	1.1641400	1.2084727	1.1646641	0.0240186	NA	0.0003263	1.4	5418	-0.0103021	1.000794
sig[2]	0.4846711	0.5051570	0.5264808	0.5052667	0.0106460	NA	0.0001439	1.4	5475	0.0125783	1.000427

```
chains.standard <- data.frame(rbind(results.standard[["mcmc"]][[1]],
                                    results.standard[["mcmc"]][[2]],
                                    results.standard[["mcmc"]][[3]]))
ggpairs(chains.standard, title="Correlation plot for standard model") + theme_minimal()
```




```
# Write in JAGS model
modelString <- "model{
  for (g in 1:Ngrps) {
    # group-specific priors
    grp.mu[g] ~ dnorm(0, 1.0E-5)
    grp.sig[g] ~ dunif(1.0E-6, 100)

    # prior for the standard deviation of the response
```

```
    sig[g] ~ dunif(1.0E-5, 100)

    for (s in 1:Nsubj[g]) {
      subj.mu[s, g] ~ dnorm(grp.mu[g], 1/grp.sig[g]^2)
    }
  }

  for (i in 1:Ntotal) {
    y[i] ~ dnorm(subj.mu[subIdx[i], grpIdx[i]], 1/sig[grpIdx[i]]^2)
    ybin[i] ~ dinterval(y[i], threshMat[i, ])
  }
}"

writeLines(modelString, con="./models/CensoredModel.txt")
```

```
y <- d.wtp$Bid
Ntotal <- length(y)
subIdx <- d.wtp$Subject
grpIdx <- as.numeric(as.factor(d.wtp$Group))
Ngrps <- length(unique(d.wtp$Group))
Nsubj <- rep(length(unique(d.wtp$Subject)), times=Ngrps)
```

```
binThresholds <- c(min(y), max(y))
ybin <- y
for (i in 1:length(y)) {
  if (y[i] == binThresholds[1]) {
    ybin[i] = 0
  } else if (y[i] == binThresholds[2]){
    ybin[i] = 2
  } else{
    ybin[i] = 1
  }
}

y[ybin == 0] <- NA
y[ybin == 2] <- NA

threshMat <- matrix(rep(binThresholds, length(y)), byrow=TRUE, nrow=length(y),
                    dimnames=list(NULL, paste0("thresh",1:length(binThresholds))))

# initial values of censored data:
yInit <- rep(NA , length(y))
for (i in 1:length(y)) {
  # if y is censored
  if (is.na(y[i])) {
    # if it is from the lowest bin
    if (ybin[i] == 0) {
      # initialize at below the lower bound of threshold
      yInit[i] <- threshMat[i, 1] - 1
      # if it is from the highest bin
    } else if (ybin[i] == 2) {
      # initialize at above the upper bound of threshold
      yInit[i] <- threshMat[i, 2] + 1
```

```
    } else {
      # initialize at middle of bin
      yInit[i] <- (threshMat[i, ybin[i]] + threshMat[i, ybin[i] + 1]) / 2
    }
  }
}
```

```
dat.censored <- dump.format(list(y = y,
                                 Ntotal = Ntotal,
                                 subIdx = subIdx,
                                 grpIdx = grpIdx,
                                 Ngrps = Ngrps,
                                 Nsubj = Nsubj,
                                 threshMat = threshMat,
                                 ybin = ybin))

inits1 <- dump.format(list(.RNG.name="base::Super-Duper", .RNG.seed=99999))
inits2 <- dump.format(list(.RNG.name="base::Wichmann-Hill", .RNG.seed=1234))
inits3 <- dump.format(list(.RNG.name="base::Mersenne-Twister", .RNG.seed=6666))

# Tell JAGS which latent variables to monitor
monitor = c("grp.mu", "grp.sig", "sig")
```

```
results.censored <- run.jags(model = "./models/CensoredModel.txt",
                             monitor = monitor,
                             data = dat.censored,
                             n.chains = n.chains,
                             inits = c(inits1, inits2, inits3),
                             plots = FALSE,
                             burnin = n.burnin,
                             sample = n.sample,
                             thin = n.thin)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 1000 iterations...
## Burning in the model for 1000 iterations...
## Running the model for 4000 iterations...
## Simulation complete
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 6 variables....
## Finished running the simulation
```

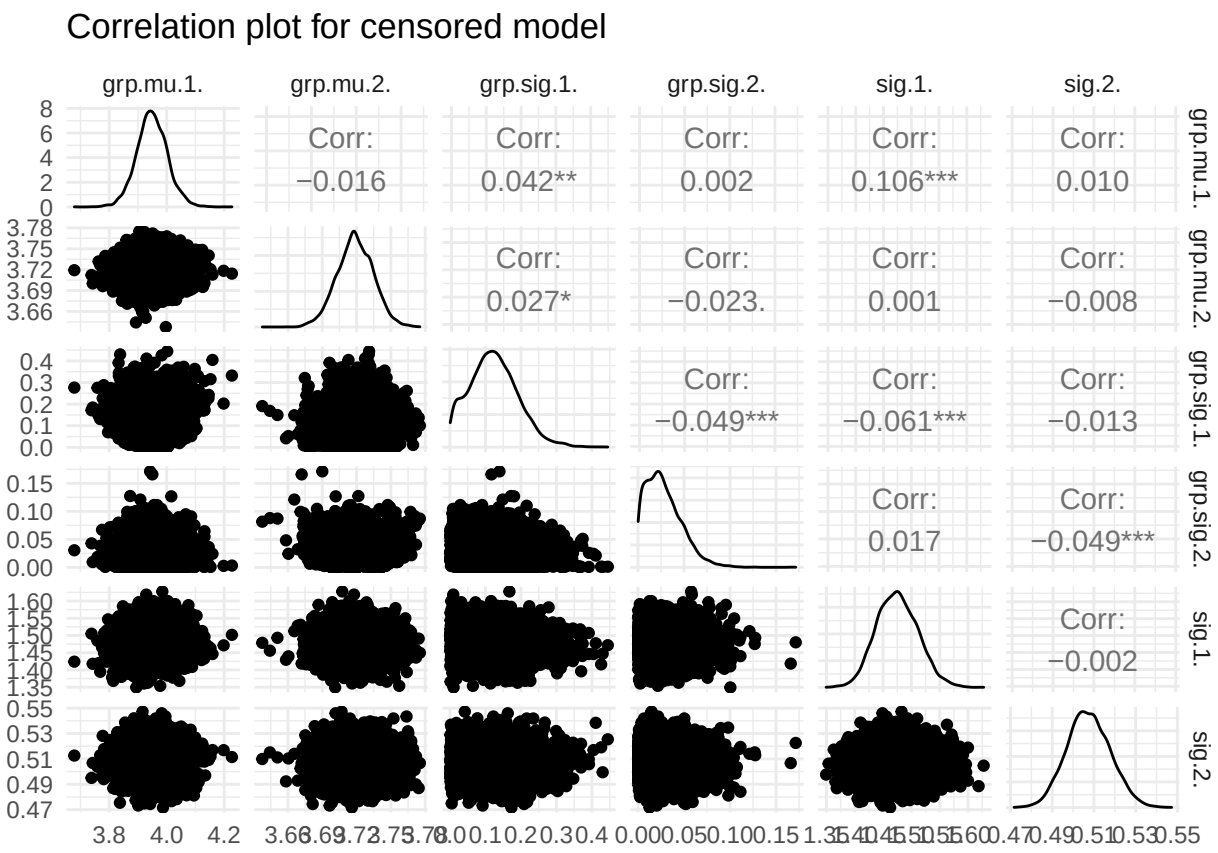
```
knitr::kable(summary(results.censored), align="c", caption="Summary results for the censored model")
```

Table 2: Summary results for the censored model

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.20	psrf
grp.mu[1]	3.8469580	3.9479165	4.0595766	3.9489758	0.0535479	NA	0.0013358	2.5	1607	0.0193494	1.0043716
grp.mu[2]	3.6851608	3.7185497	3.7473857	3.7187869	0.0159376	NA	0.0006538	4.1	594	0.1846761	1.0075262
grp.sig[1]	0.0012484	0.1206696	0.2395748	0.1237714	0.0676627	NA	0.0037974	5.6	317	0.3318470	1.0046897

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.20	psrf
grp.sig[2]	0.0001618	0.0249725	0.0633215	0.0280119	0.0194453	NA	0.0011943	6.1	265	0.4059735	1.0169655
sig[1]	1.4026036	1.4741242	1.5431247	1.4750567	0.0364873	NA	0.0006212	1.7	3450	0.0273024	1.0001064
sig[2]	0.4874418	0.5071311	0.5276976	0.5074181	0.0104176	NA	0.0001431	1.4	5299	-0.0137722	0.9998306

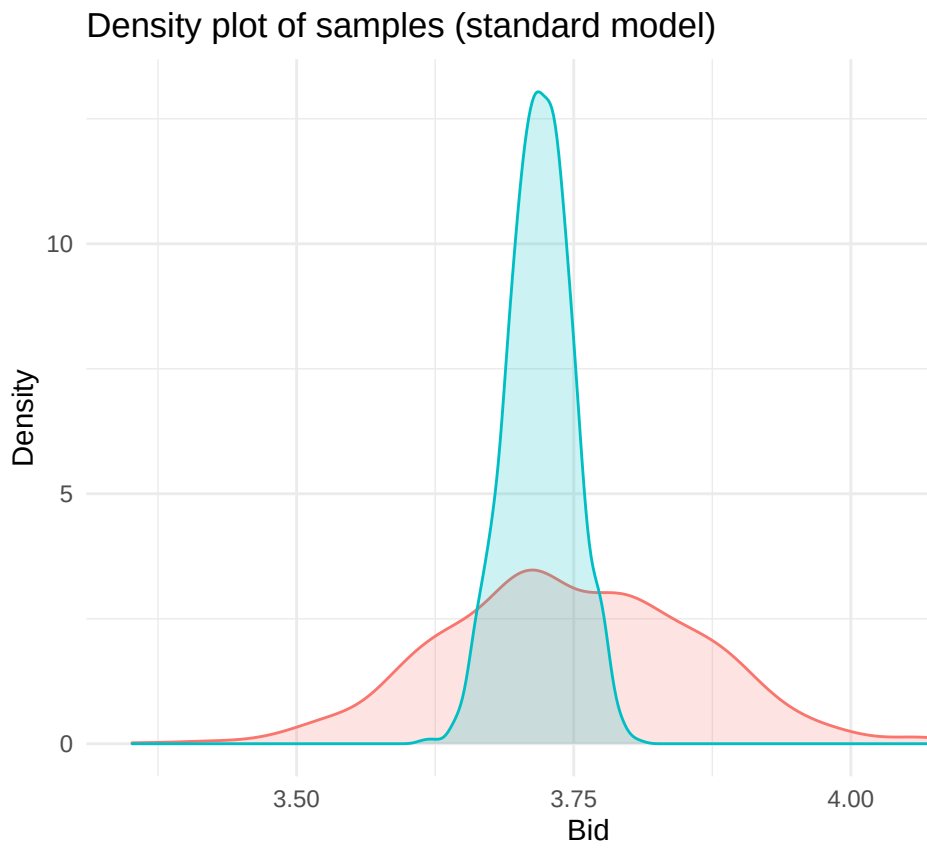
```
chains.censored <- data.frame(rbind(results.censored[["mcmc"]][[1]],
                                    results.censored[["mcmc"]][[2]],
                                    results.censored[["mcmc"]][[3]]))
ggpairs(chains.censored, title="Correlation plot for censored model") + theme_minimal()
```




```
mu1.standard <- summary(results.standard)["grp.mu[1]", "Mean"]
mu2.standard <- summary(results.standard)["grp.mu[2]", "Mean"]
sig1.standard <- summary(results.standard)["grp.sig[1]", "Mean"]
sig2.standard <- summary(results.standard)["grp.sig[2]", "Mean"]
```

```
# Set seed for reproducible results
set.seed(2022)
```

```r
# Set sample size
M <- 1000

# Generate random samples using estimated parameters
group1.standard <- rnorm(M, mean=mu1.standard, sd=sig1.standard)
group2.standard <- rnorm(M, mean=mu2.standard, sd=sig2.standard)

# Create a dataframe for ggplot
d.plot <- data.frame(Bid=c(group1.standard, group2.standard),
                     Group=rep(c("A", "B"), each=M))

ggplot(d.plot, aes(x=Bid, y=..density.., color=Group, fill=Group)) +
  geom_density(alpha=0.2) + theme_minimal() +
  labs(title="Density plot of samples (standard model)", y="Density")
```

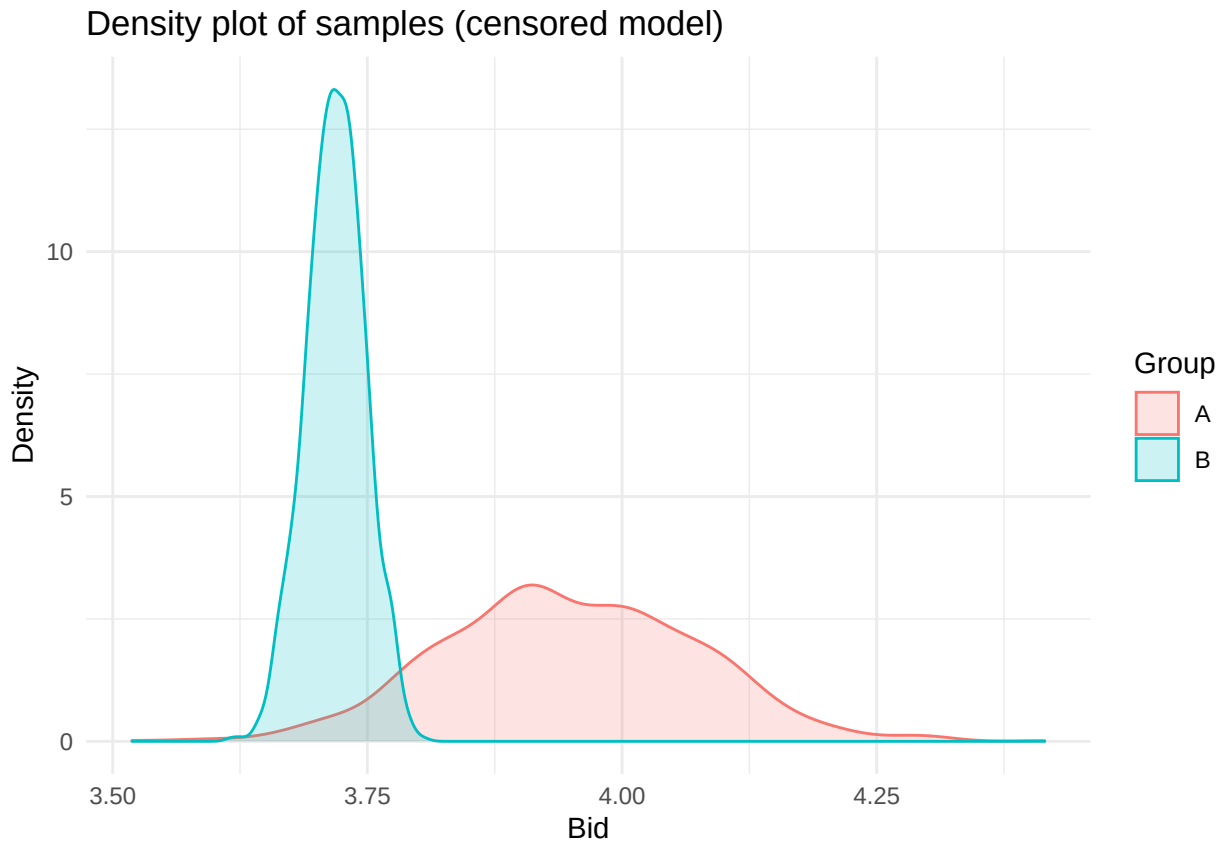


Empirical density based on estimates

```r
mu1.censored <- summary(results.censored)["grp.mu[1]", "Mean"]
mu2.censored <- summary(results.censored)["grp.mu[2]", "Mean"]
sig1.censored <- summary(results.censored)["grp.sig[1]", "Mean"]
sig2.censored <- summary(results.censored)["grp.sig[2]", "Mean"]
```

```r
# Set seed for reproducible results
set.seed(2022)
# Set sample size
M <- 1000
```

```r
# Generate random samples using estimated parameters
group1.censored <- rnorm(M, mean=mu1.censored, sd=sig1.censored)
group2.censored <- rnorm(M, mean=mu2.censored, sd=sig2.censored)

# Create a dataframe for ggplot
d.plot <- data.frame(Bid=c(group1.censored, group2.censored),
                     Group=rep(c("A", "B"), each=M))

ggplot(d.plot, aes(x=Bid, y=..density.., color=Group, fill=Group)) +
  geom_density(alpha=0.2) + theme_minimal() +
  labs(title="Density plot of samples (censored model)", y="Density")
```

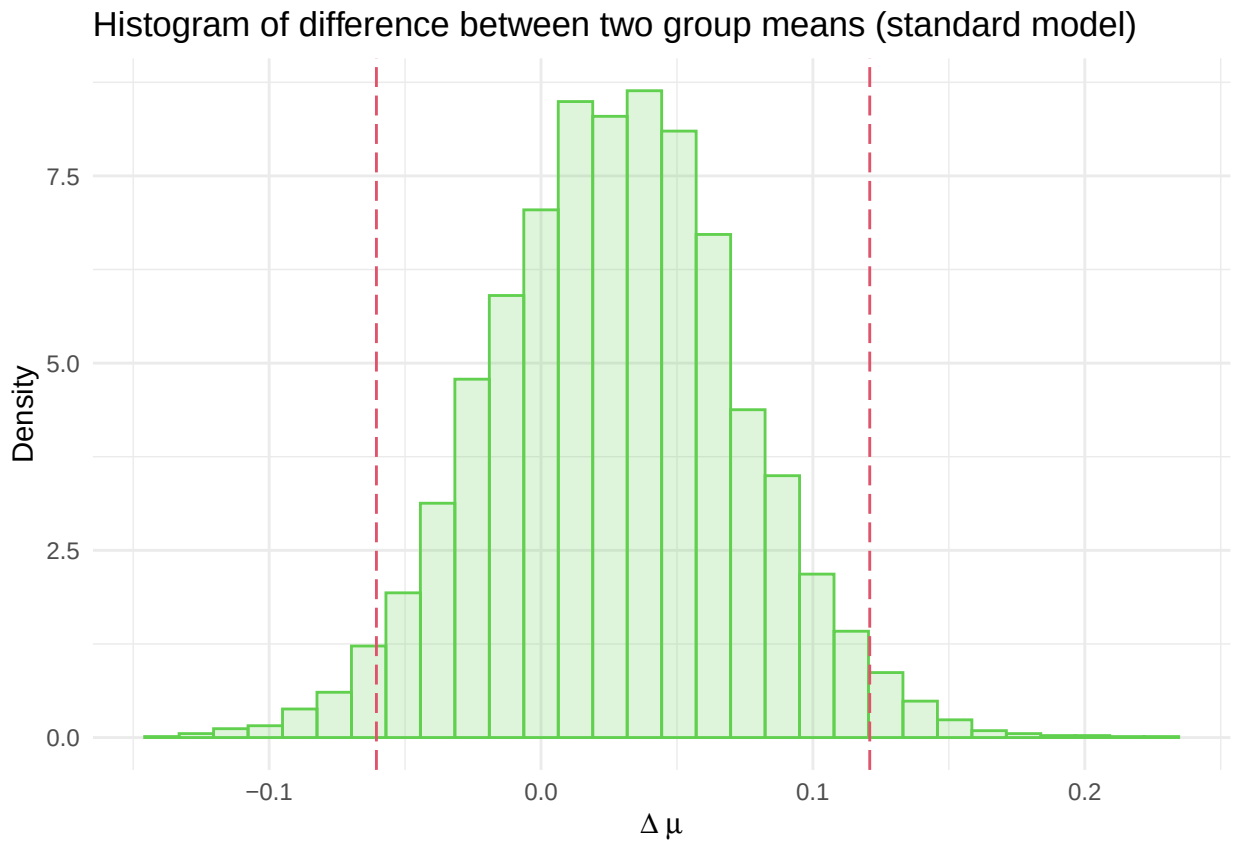


Contrasts Contrasts are based on the difference between sample posterior distributions of parameters

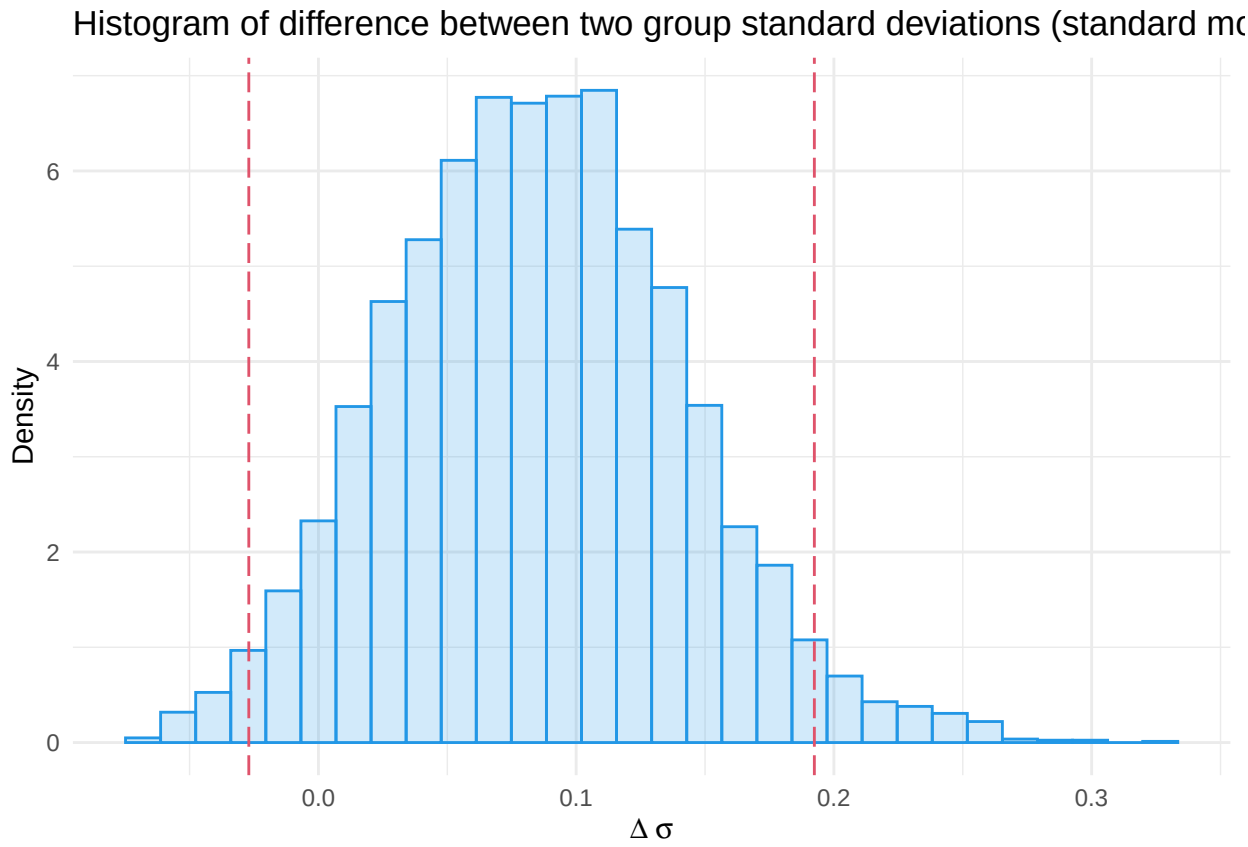
```r
contrast.mu.std <- chains.standard[, "grp.mu.1."] - chains.standard[, "grp.mu.2."]
contrast.sig.std <- chains.standard[, "grp.sig.1."] - chains.standard[, "grp.sig.2."]

contrast.mu.csd <- chains.censored[, "grp.mu.1."] - chains.censored[, "grp.mu.2."]
contrast.sig.csd <- chains.censored[, "grp.sig.1."] - chains.censored[, "grp.sig.2."]
```

```r
ggplot(data.frame(x=contrast.mu.std), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=3, fill=3) +
  geom_vline(xintercept=hdi(contrast.mu.std), color=2, lty="longdash") +
```

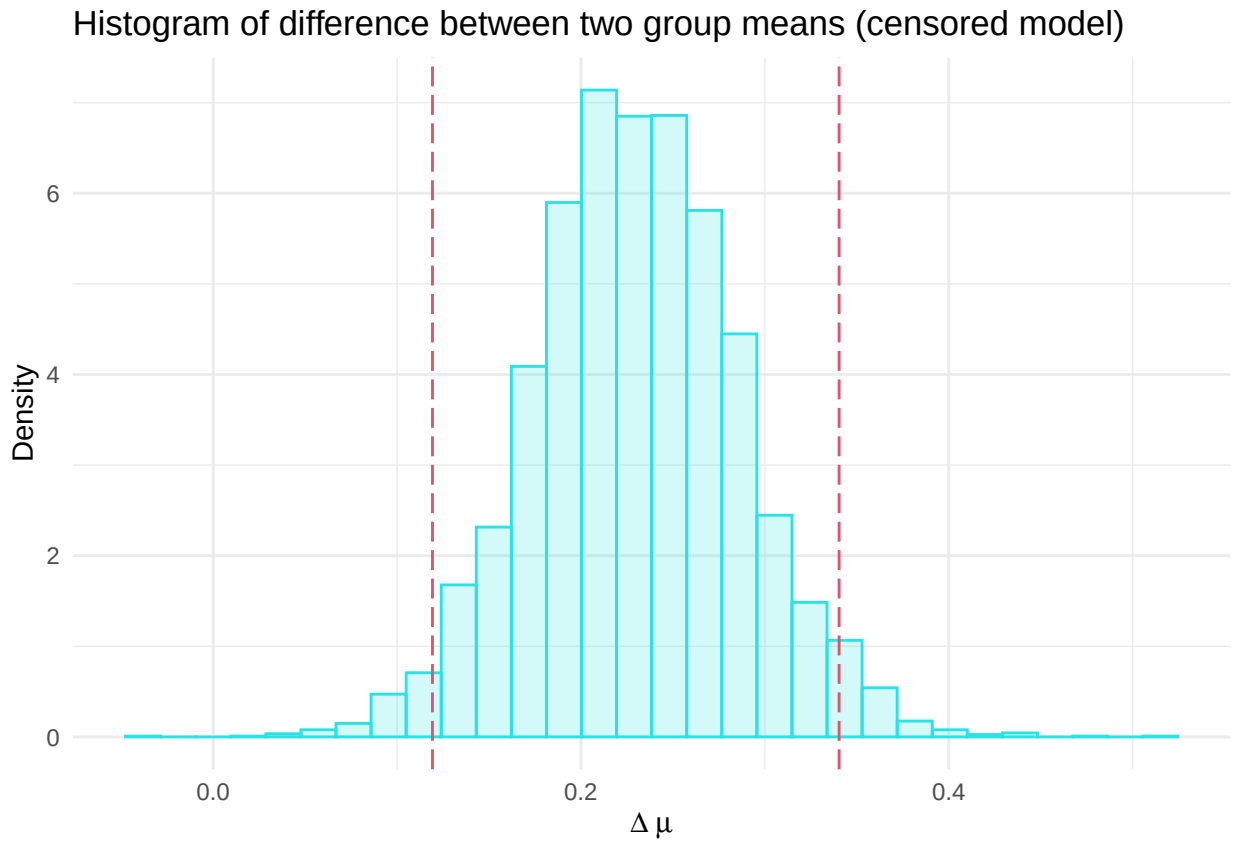
```
  labs(title="Histogram of difference between two group means (standard model)",
       x=expression(Delta~mu), y="Density") + theme_minimal()
```



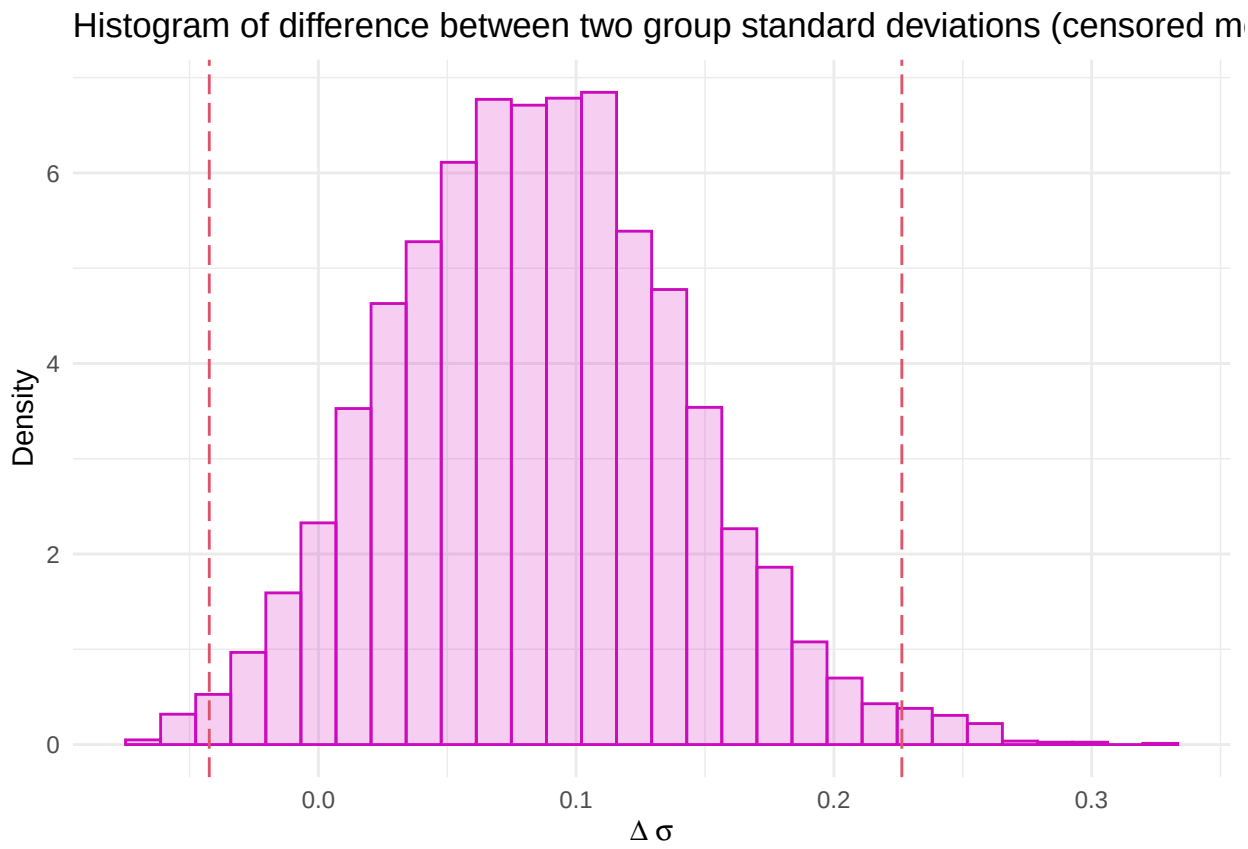

```
ggplot(data.frame(x=contrast.sig.std), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=4, fill=4) +
  geom_vline(xintercept=hdi(contrast.sig.std), color=2, lty="longdash") +
  labs(title="Histogram of difference between two group standard deviations (standard model)",
       x=expression(Delta~sigma), y="Density") + theme_minimal()
```




```
ggplot(data.frame(x=contrast.mu.csd), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=5, fill=5) +
  geom_vline(xintercept=hdi(contrast.mu.csd), color=2, lty="longdash") +
  labs(title="Histogram of difference between two group means (censored model)",
       x=expression(Delta~mu), y="Density") + theme_minimal()
```




```
ggplot(data.frame(x=contrast.sig.std), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=6, fill=6) +
  geom_vline(xintercept=hdi(contrast.sig.csd), color=2, lty="longdash") +
  labs(title="Histogram of difference between two group standard deviations (censored model)",
       x=expression(Delta~sigma), y="Density") + theme_minimal()
```



We want to test whether group-level means and standard deviations differ across group A and group B in the standard normal model and censored normal model separately using 95% highest density intervals (HDIs).

$$\mu_1 \stackrel{?}{=} \mu_2$$
$$\sigma_1 \stackrel{?}{=} \sigma_2$$

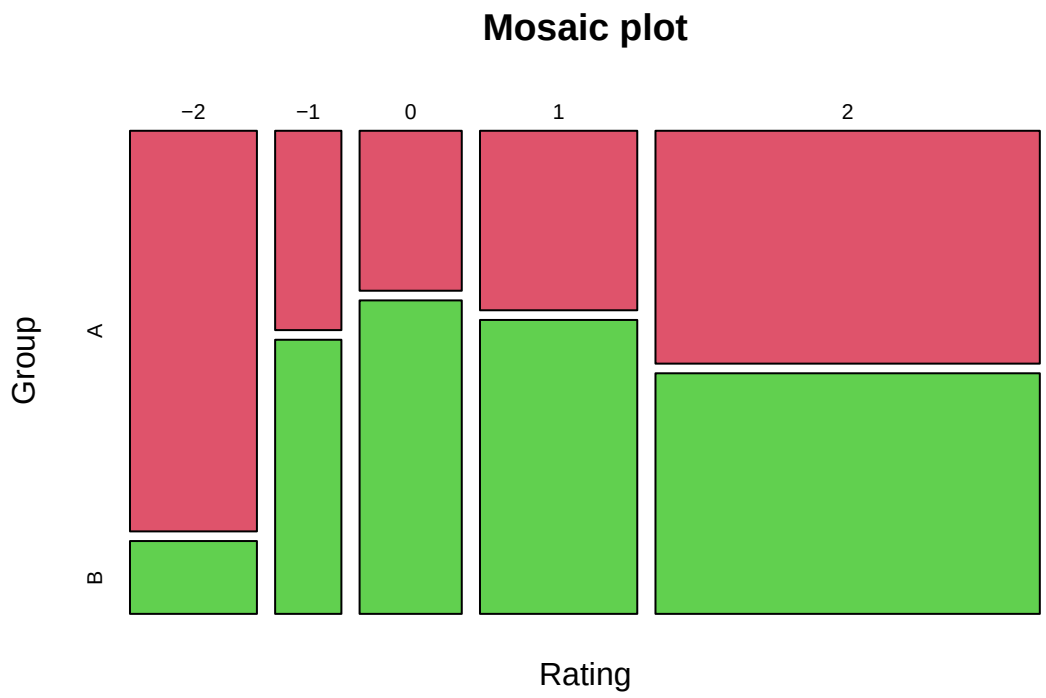
- For the standard model, we see that both $\Delta\mu = 0$ and $\Delta\sigma$ lie in their 95% HDIs.
- For the censored model, we see that the 95% HDI for $\Delta\mu$ does not include $\Delta\mu = 0$ while the 95% HDI for $\Delta\sigma$ includes $\Delta\sigma = 0$.

We therefore conclude that with 95% HDIs both the group-level means and standard deviations do not differ in the standard normal model whereas the group-level means do differ but standard deviations do not differ in the censored normal model.

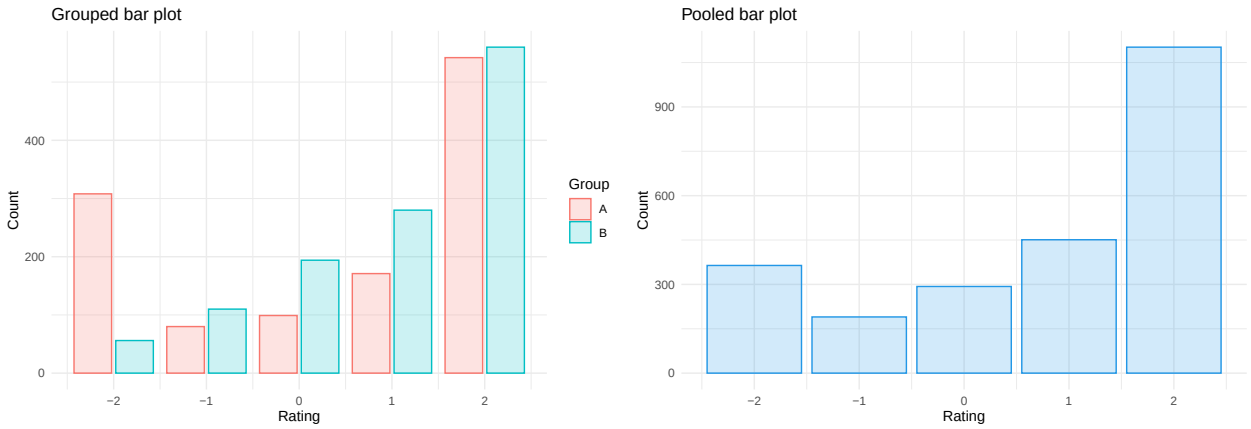
Part 2

```
d.ratings <- read.csv("./data/ratings_data.csv")
d.ratings$Group <- factor(d.ratings$Group)
```

```
mosaicplot(Rating ~ Group, data=d.ratings, col=2:3, main="Mosaic plot")
```




```
ggplot(d.ratings, aes(x=Rating, fill=Group, color=Group)) +
  geom_bar(stat="count", position=position_dodge2(preserve="single"), alpha=0.2) +
  labs(title="Grouped bar plot", y="Count") + theme_minimal()

ggplot(d.ratings, aes(x=Rating)) +
  geom_bar(stat="count", position=position_dodge2(preserve="single"), alpha=0.2, color=4, fill=4) +
  labs(title="Pooled bar plot", y="Count") + theme_minimal()
```



Bar plots

```
modelString <- "model{
  for (g in 1:Ngrps) {
    b0.mu[g] ~ dnorm(0.0, 1.0E-5) # prior intercept
    b1.mu[g] ~ dnorm(0.0, 1.0E-5) # prior coefficient1
    b2.mu[g] ~ dnorm(0.0, 1.0E-5) # prior coefficient2

    b0.sig[g] ~ dunif(1.0E-4, 100) # prior intercept error
    b1.sig[g] ~ dunif(1.0E-4, 100) # prior coefficient1 error
    b2.sig[g] ~ dunif(1.0E-4, 100) # prior coefficient2 error

    sig[g] ~ dunif(1.0E-5, 100) # prior for the sd

    for (s in 1:Nsubj[g]) {
      b0.s[s, g] ~ dnorm(b0.mu[g], 1/b0.sig[g]^2)
      b1.s[s, g] ~ dnorm(b1.mu[g], 1/b1.sig[g]^2)
      b2.s[s, g] ~ dnorm(b2.mu[g], 1/b2.sig[g]^2)
    }
  }

  for (i in 1:Ntotal) {
    mu[i] <- b0.s[subIdx[i], grpIdx[i]] +
             b1.s[subIdx[i], grpIdx[i]] * x1[i] +
             b2.s[subIdx[i], grpIdx[i]] * x2[i]
    y[i] ~ dnorm(mu[i], 1/sig[grpIdx[i]]^2)
  }
}"
writeLines(modelString, con="./models/LinearModel.txt")
```

```
# Prepare data for JAGS
x1 <- d.ratings$x1
x2 <- d.ratings$x2
y <- d.ratings$Rating
Ntotal <- length(y)
subIdx <- d.ratings$Subject
grpIdx <- as.numeric(as.factor(d.ratings$Group))
```

```
Ngrps <- length(unique(d.ratings$Group))
Nsubj <- rep(length(unique(d.ratings$Subject)), times=Ngrps)

dat.linear <- dump.format(list(x1 = x1,
                               x2 = x2,
                               y = y,
                               Ntotal = Ntotal,
                               subIdx = subIdx,
                               grpIdx = grpIdx,
                               Ngrps = Ngrps,
                               Nsubj = Nsubj))

# Let JAGS initialize
inits1 <- dump.format(list(.RNG.name="base::Super-Duper", .RNG.seed=99999 ))
inits2 <- dump.format(list(.RNG.name="base::Wichmann-Hill", .RNG.seed=1234 ))
inits3 <- dump.format(list(.RNG.name="base::Mersenne-Twister", .RNG.seed=6666 ))

monitor <- c("b0.mu", "b1.mu", "b2.mu", "sig")
```

```
# Run JAGS model
results.linear <- run.jags(model = "./models/LinearModel.txt",
                           monitor = monitor,
                           data = dat.linear,
                           n.chains = n.chains,
                           inits = c(inits1, inits2, inits3),
                           plots = FALSE,
                           burnin = n.burnin,
                           sample = n.sample,
                           thin = n.thin)
```

Summary tables and correlation plots

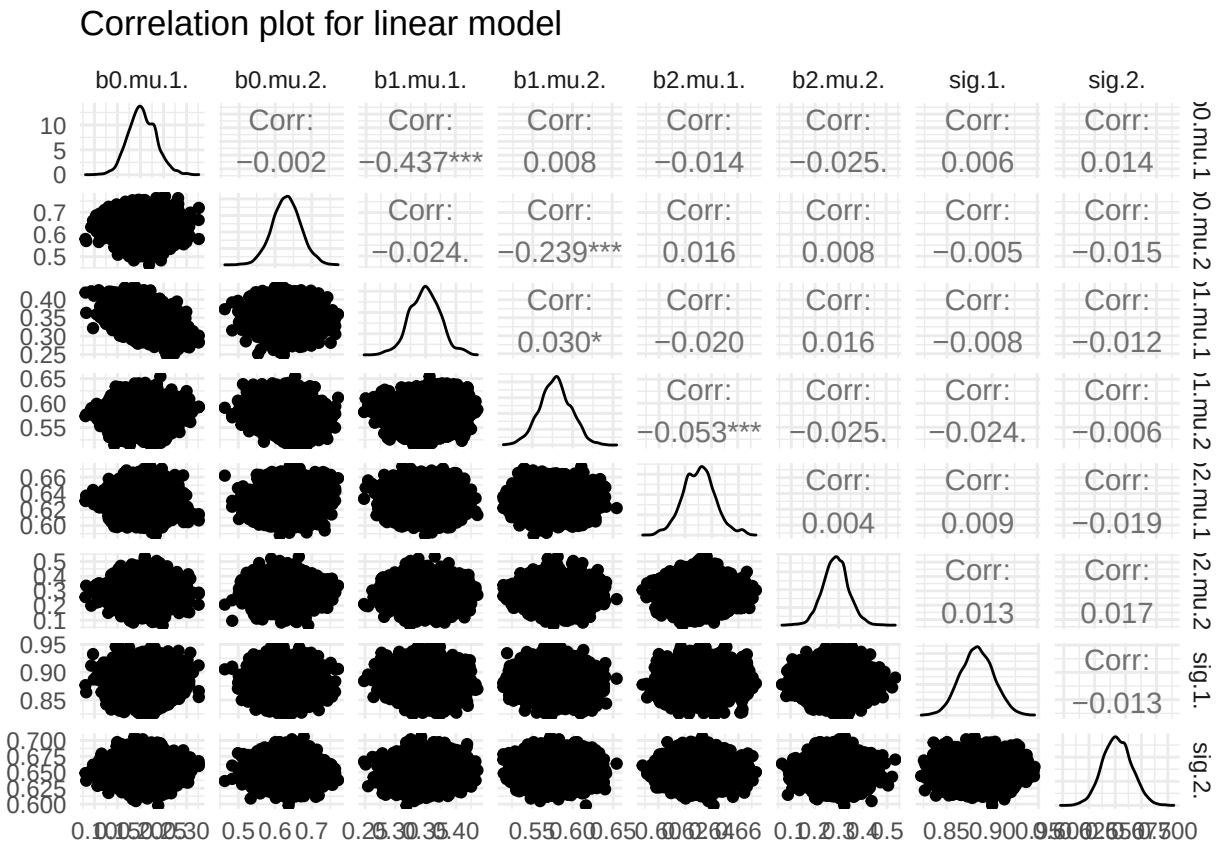
```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 1000 iterations...
## Burning in the model for 1000 iterations...
## Running the model for 4000 iterations...
## Simulation complete
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 8 variables....
## Finished running the simulation

knitr::kable(summary(results.linear), align="c", caption="Summary results for the linear model")
```

Table 3: Summary results for the linear model

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.20	psrf
b0.mu[1]	0.1468833	0.2022878	0.2648419	0.2036940	0.0304263	NA	0.0018615	6.1	267	0.4123779	1.012948
b0.mu[2]	0.5547507	0.6313626	0.7090348	0.6312211	0.0387935	NA	0.0006798	1.8	3256	0.0331856	1.000511
b1.mu[1]	0.2959213	0.3490125	0.4007270	0.3489257	0.0255967	NA	0.0015314	6.0	279	0.3985834	1.010361

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.20	psrf
b1.mu[2]	0.5389319	0.5785500	0.6148478	0.5785344	0.0194529	NA	0.0011059	5.7	309	0.3193557	1.015693
b2.mu[1]	0.6075150	0.6306419	0.6561205	0.6305341	0.0120744	NA	0.0007699	6.4	246	0.4523380	1.018678
b2.mu[2]	0.1799900	0.2875940	0.3890225	0.2875068	0.0534959	NA	0.0007037	1.3	5779	0.0203211	1.000642
sig[1]	0.8486186	0.8833906	0.9179941	0.8833592	0.0179351	NA	0.0002499	1.4	5152	-0.0115975	1.000474
sig[2]	0.6258526	0.6515066	0.6793915	0.6518467	0.0137702	NA	0.0001935	1.4	5062	0.0032623	1.000765

```
chains.linear <- data.frame(rbind(results.linear[["mcmc"]][[1]],
                                  results.linear[["mcmc"]][[2]],
                                  results.linear[["mcmc"]][[3]]))
ggpairs(chains.linear, title="Correlation plot for linear model") + theme_minimal()
```




```
# Write in JAGS model
modelString <- "model {
    for (g in 1:Ngrps) {

    for (k in 2:(nYlevels-2)) {
        thresh[k, g] ~ dnorm(k+0.5, 1/2^2)
    }

        b0.mu[g] ~ dnorm(0.0, 1.0E-5) # prior beta0
        b1.mu[g] ~ dnorm(0.0, 1.0E-5) # prior beta1
        b2.mu[g] ~ dnorm(0.0, 1.0E-5) # prior beta2
```

```
        b0.sig[g] ~ dunif(1.0E-4, 100) # prior beta0 error
        b1.sig[g] ~ dunif(1.0E-4, 100) # prior beta1 error
        b2.sig[g] ~ dunif(1.0E-4, 100) # prior beta2 error

        sig[g] ~ dunif(nYlevels/1000 , nYlevels*10)

        for (s in 1:Nsubj[g]){
            b0.s[s, g] ~ dnorm(b0.mu[g], 1/(b0.sig[g]^2))
            b1.s[s, g] ~ dnorm(b1.mu[g], 1/(b1.sig[g]^2))
            b2.s[s, g] ~ dnorm(b2.mu[g], 1/(b2.sig[g]^2))
        }
    }

    for (i in 1:Ntotal) {
        mu[i] <- b0.s[subIdx[i], grpIdx[i]] * 1.0   +
             b1.s[subIdx[i], grpIdx[i]] * x1[i] +
             b2.s[subIdx[i], grpIdx[i]] * x2[i]

        pr[i, 1] <- pnorm(thresh[1, grpIdx[i]] , mu[i] , 1/sig[grpIdx[i]]^2)

        for (k in 2:(nYlevels-1)) {
            pr[i, k] <- max(0,  pnorm(thresh[k, grpIdx[i]], mu[i], 1/sig[grpIdx[i]]^2) -
                                                pnorm(thresh[k-1, grpIdx[i]], mu[i], 1/sig[grpIdx[i]]
        }

        pr[i, nYlevels] <- 1 - pnorm(thresh[nYlevels-1, grpIdx[i]], mu[i], 1/sig[grpIdx[i]]^2)

        y[i] ~ dcat(pr[i,1:nYlevels])
    }
}"

writeLines(modelString, con="./models/OrdinalModel.txt")
```

```
# Prepare data list for JAGS
x1 <- d.ratings$x1
x2 <- d.ratings$x2
y <- as.numeric(ordered(d.ratings$Rating))
Ntotal <- length(y)
subIdx <- d.ratings$Subject
grpIdx <- as.numeric(as.factor(d.ratings$Group))
Ngrps <- length(unique(d.ratings$Group))
Nsubj <- rep(length(unique(d.ratings$Subject)), times=Ngrps)
nYlevels <- as.numeric(length(unique(d.ratings$Rating)))

thresh = matrix(NA, nrow = nYlevels-1, ncol = 2)
thresh[1, ] <- rep(1.5, 2)
thresh[nYlevels-1, ] <- rep(nYlevels - 0.5, 2)

dat <- dump.format(list(x1=x1,
                        x2=x2,
                        y=y,
                        Ntotal=Ntotal,
                        Ngrps = Ngrps,
```

```
                          subIdx = subIdx,
                          grpIdx = grpIdx,
                          Nsubj=Nsubj,
                          nYlevels = nYlevels,
                          thresh = thresh))

# Let JAGS initialize
inits1 <- dump.format(list( .RNG.name="base::Super-Duper", .RNG.seed=99999 ))
inits2 <- dump.format(list( .RNG.name="base::Wichmann-Hill", .RNG.seed=1234 ))
inits3 <- dump.format(list( .RNG.name="base::Mersenne-Twister", .RNG.seed=6666 ))

# Tell JAGS which latent variables to monitor
monitor = c("b0.mu", "b1.mu", "b2.mu", "sig", "thresh")
```

```
n.sample <- 2000
n.burnin <- 1000
n.thin <- 2
n.chains <- 3

results.ordinal <- run.jags(model = "models/ordinalHierRegr.txt",
                            monitor = monitor,
                            data = dat,
                            n.chains = 3,
                            inits = c(inits1, inits2, inits3),
                            plots = FALSE,
                            burnin = n.burnin,
                            sample = n.sample,
                            thin = n.thin)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 1000 iterations...
## Burning in the model for 1000 iterations...
## Running the model for 4000 iterations...
## Simulation complete
## Calculating summary statistics...
## Note: The monitored variables 'thresh[1,1]', 'thresh[4,1]',
## 'thresh[1,2]' and 'thresh[4,2]' appear to be non-stochastic; they will
## not be included in the convergence diagnostic
## Calculating the Gelman-Rubin statistic for 16 variables....
## Finished running the simulation
```

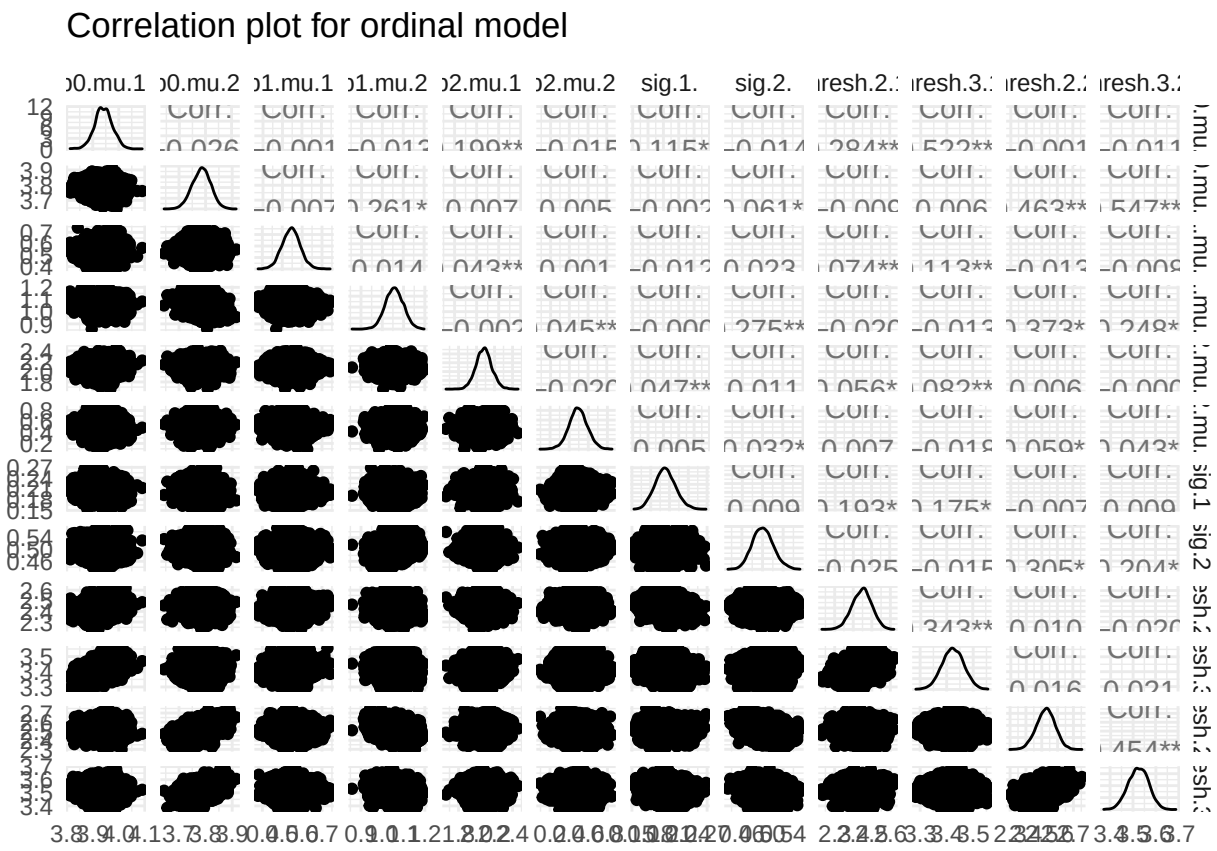
```
load("./.RData")
knitr::kable(summary(results.ordinal), align="c", digits=6, caption="Summary results for ordinal model")
```

Table 4: Summary results for ordinal model

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.20	psrf
b0.mu[1]	3.876851	3.941098	4.010390	3.941569	0.034154	NA	0.001714	5.0	397	0.261224	1.007586
b0.mu[2]	3.716983	3.785046	3.852849	3.784329	0.034773	NA	0.001648	4.7	445	0.259192	1.004263
b1.mu[1]	0.459217	0.545159	0.641325	0.544780	0.046080	NA	0.000657	1.4	4913	0.027948	0.999962
b1.mu[2]	0.975382	1.056802	1.138194	1.056810	0.042098	NA	0.001295	3.1	1057	0.071566	1.003824

	Lower95	Median	Upper95	Mean	SD	Mode	MCerr	MC%ofSD	SSeff	AC.20	psrf
b2.mu[1]	1.917548	2.081299	2.264433	2.081591	0.088364	NA	0.001499	1.7	3474	0.017077	1.001142
b2.mu[2]	0.310054	0.493605	0.692162	0.494325	0.096752	NA	0.001276	1.3	5750	-0.001365	1.000741
sig[1]	0.171358	0.202757	0.234968	0.203255	0.016302	NA	0.000295	1.8	3064	0.025564	1.000388
sig[2]	0.460416	0.500276	0.539818	0.500781	0.020171	NA	0.000388	1.9	2701	0.000990	1.002139
thresh[1,1]	1.500000	1.500000	1.500000	1.500000	0.000000	NA	NA	NA	NA	NA	NA
thresh[2,1]	2.349224	2.448203	2.550956	2.446635	0.052371	NA	0.001123	2.1	2173	0.002828	1.000110
thresh[3,1]	3.343247	3.423793	3.502723	3.423847	0.040978	NA	0.001299	3.2	995	0.117388	1.001930
thresh[4,1]	4.500000	4.500000	4.500000	4.500000	0.000000	NA	NA	NA	NA	NA	NA
thresh[1,2]	1.500000	1.500000	1.500000	1.500000	0.000000	NA	NA	NA	NA	NA	NA
thresh[2,2]	2.408680	2.526917	2.636953	2.525351	0.058944	NA	0.001954	3.3	910	0.112729	1.005235
thresh[3,2]	3.440280	3.524401	3.608588	3.523699	0.043600	NA	0.001544	3.5	798	0.130086	1.001759
thresh[4,2]	4.500000	4.500000	4.500000	4.500000	0.000000	NA	NA	NA	NA	NA	NA

```
chains.ordinal <- data.frame(rbind(results.ordinal[["mcmc"]][[1]],
                                   results.ordinal[["mcmc"]][[2]],
                                   results.ordinal[["mcmc"]][[3]]))

# Drop rows with NAs except for the mode column
ggpairs(chains.ordinal[, c(1:8, 10:11, 14:15)], title="Correlation plot for ordinal model") + theme_min
```

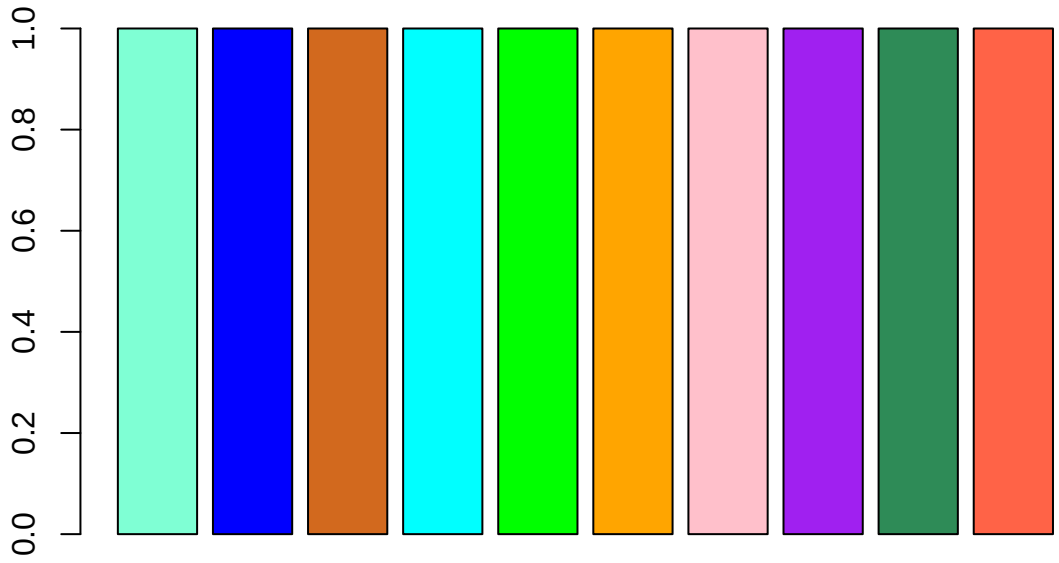


Contrasts Contrasts are based on the difference between sample posterior distributions of parameters. The posterior probability of superiority (PPS) is defined as follows:

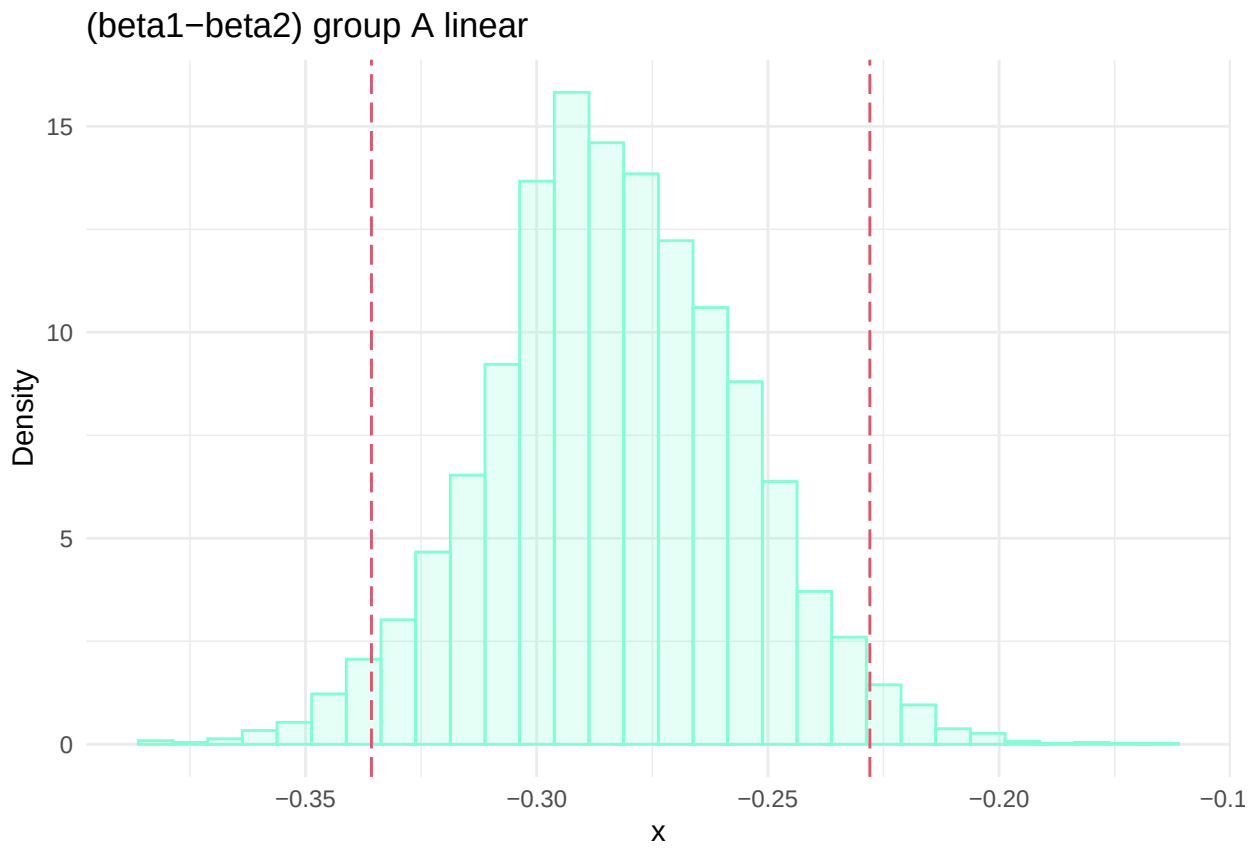
$$\text{PPS} = P(\text{contrast} > 0)$$

```
contrast.a1 <- chains.linear[, "b1.mu.1."] - chains.linear[, "b2.mu.1."]
contrast.a2 <- chains.linear[, "b1.mu.2."] - chains.linear[, "b2.mu.2."]

contrast.b1 <- chains.ordinal[, "b1.mu.1."] - chains.ordinal[, "b2.mu.1."]
contrast.b2 <- chains.ordinal[, "b1.mu.2."] - chains.ordinal[, "b2.mu.2."]

contrast.c <- chains.linear[, "b1.mu.1."] - chains.ordinal[, "b1.mu.1."]
contrast.d <- chains.linear[, "b2.mu.1."] - chains.ordinal[, "b2.mu.1."]

contrast.e <- chains.linear[, "b1.mu.2."] - chains.ordinal[, "b1.mu.2."]
contrast.f <- chains.linear[, "b2.mu.2."] - chains.ordinal[, "b2.mu.2."]

contrast.g <- (contrast.a1 - contrast.b1)
contrast.h <- (contrast.a2 - contrast.b2)
```

```
cols <- c("aquamarine", "blue", "chocolate", "cyan", "green", "orange", "pink", "purple", "seagreen", "
barplot(rep(1, length(cols)), col=cols)
```



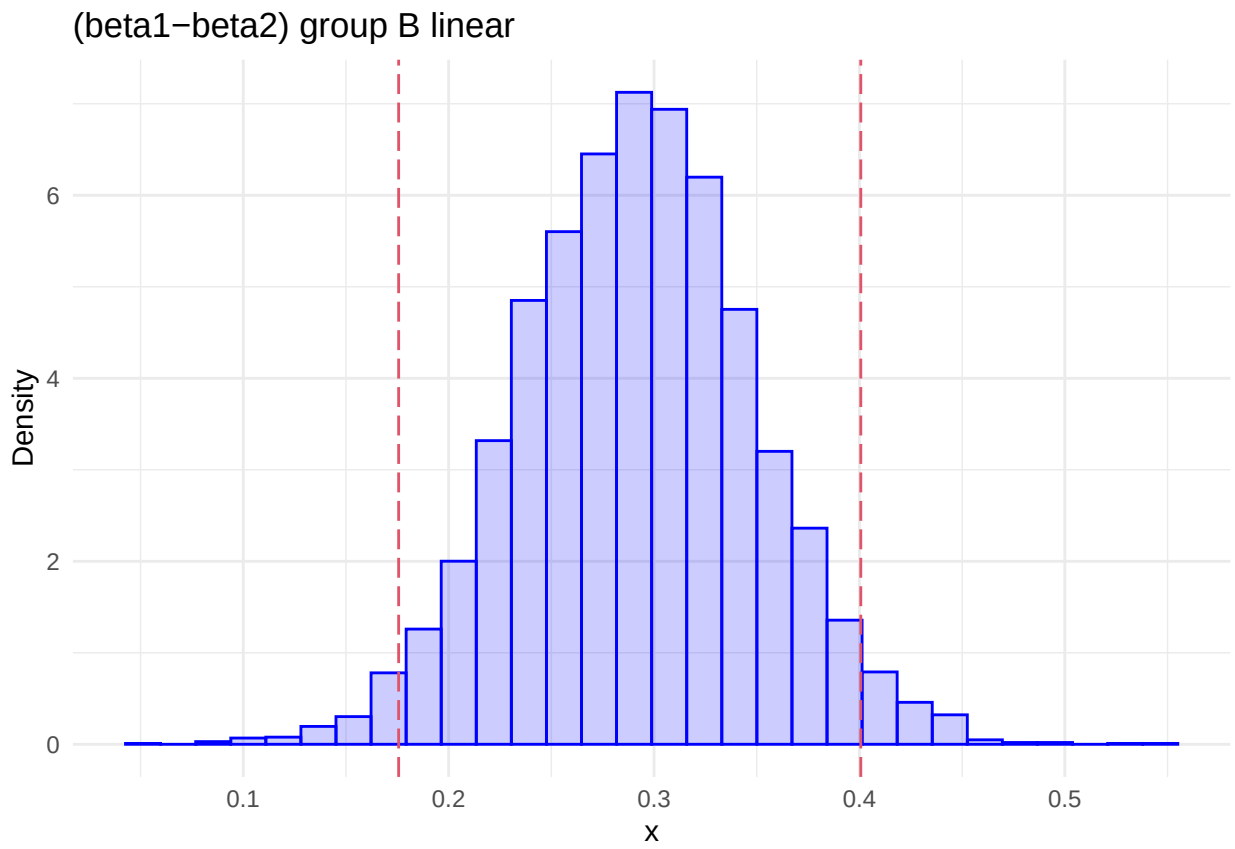

```
ggplot(data.frame(x=contrast.a1), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[1], fill=cols[1]) +
  geom_vline(xintercept=hdi(contrast.a1), color=2, lty="longdash") +
  labs(title="(beta1-beta2) group A linear", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.a1 <- mean(contrast.a1 > 0); pps.contrast.a1
```

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

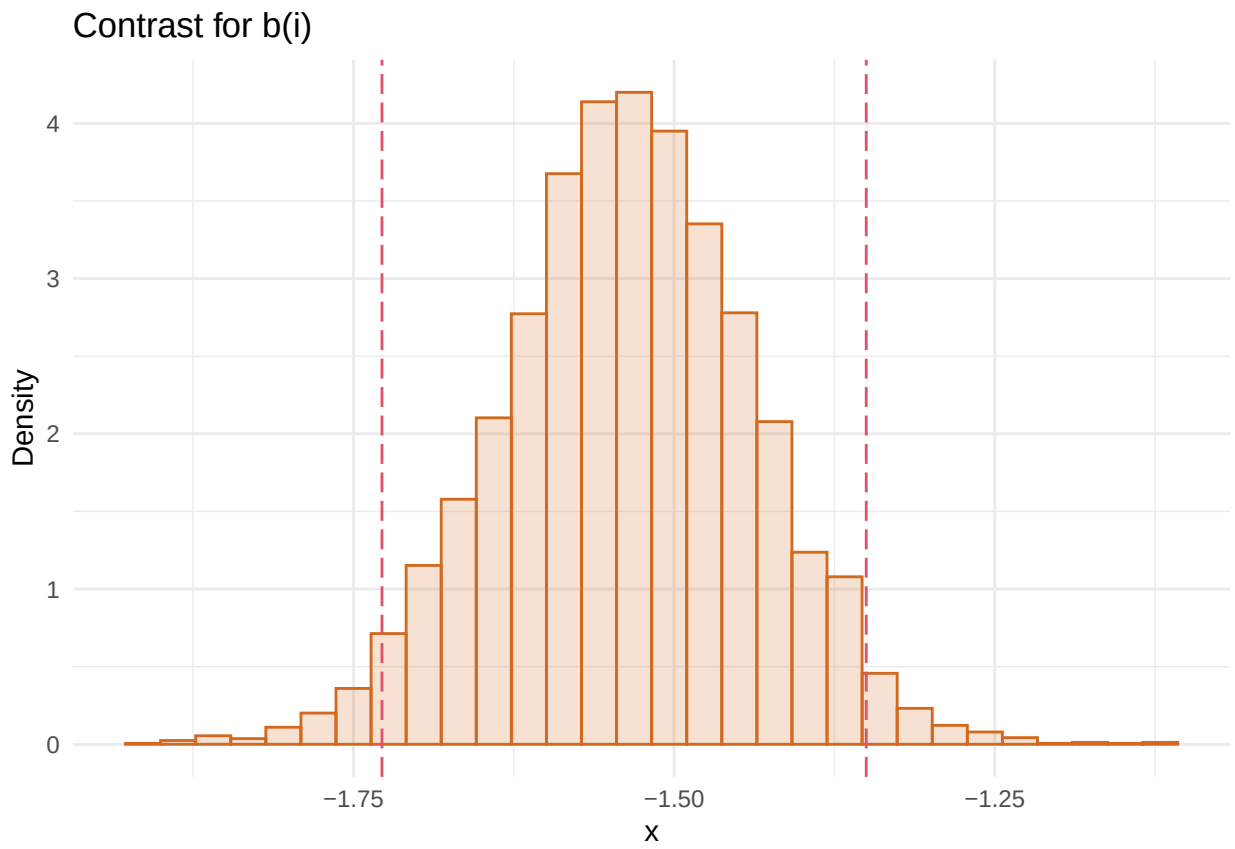
```
ggplot(data.frame(x=contrast.a2), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[2], fill=cols[2]) +
  geom_vline(xintercept=hdi(contrast.a2), color=2, lty="longdash") +
  labs(title="(beta1-beta2) group B linear", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.a2 <- mean(contrast.a2 > 0); pps.contrast.a2
```

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

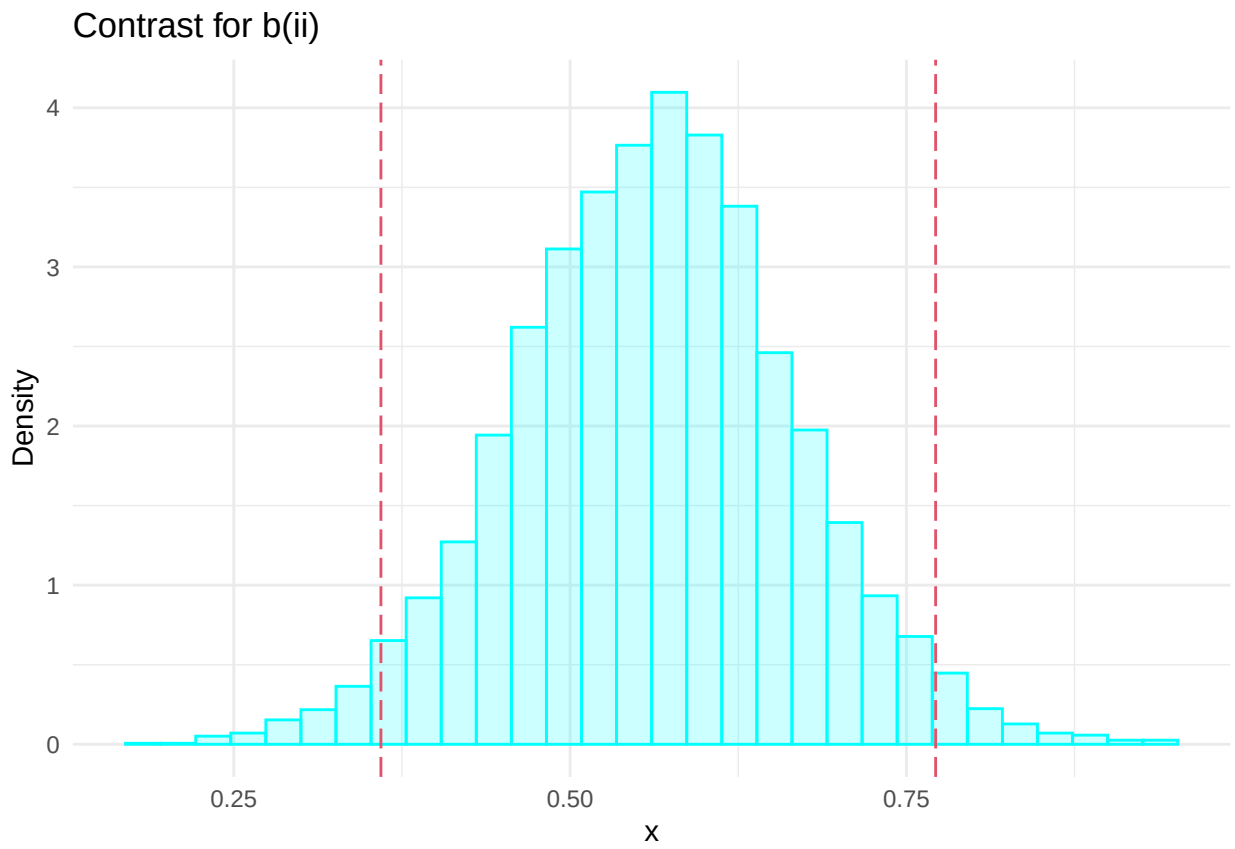
```
ggplot(data.frame(x=contrast.b1), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[3], fill=cols[3]) +
  geom_vline(xintercept=hdi(contrast.b1), color=2, lty="longdash") +
  labs(title="Contrast for b(i)", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.b1 <- mean(contrast.b1 > 0); pps.contrast.b1
```

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

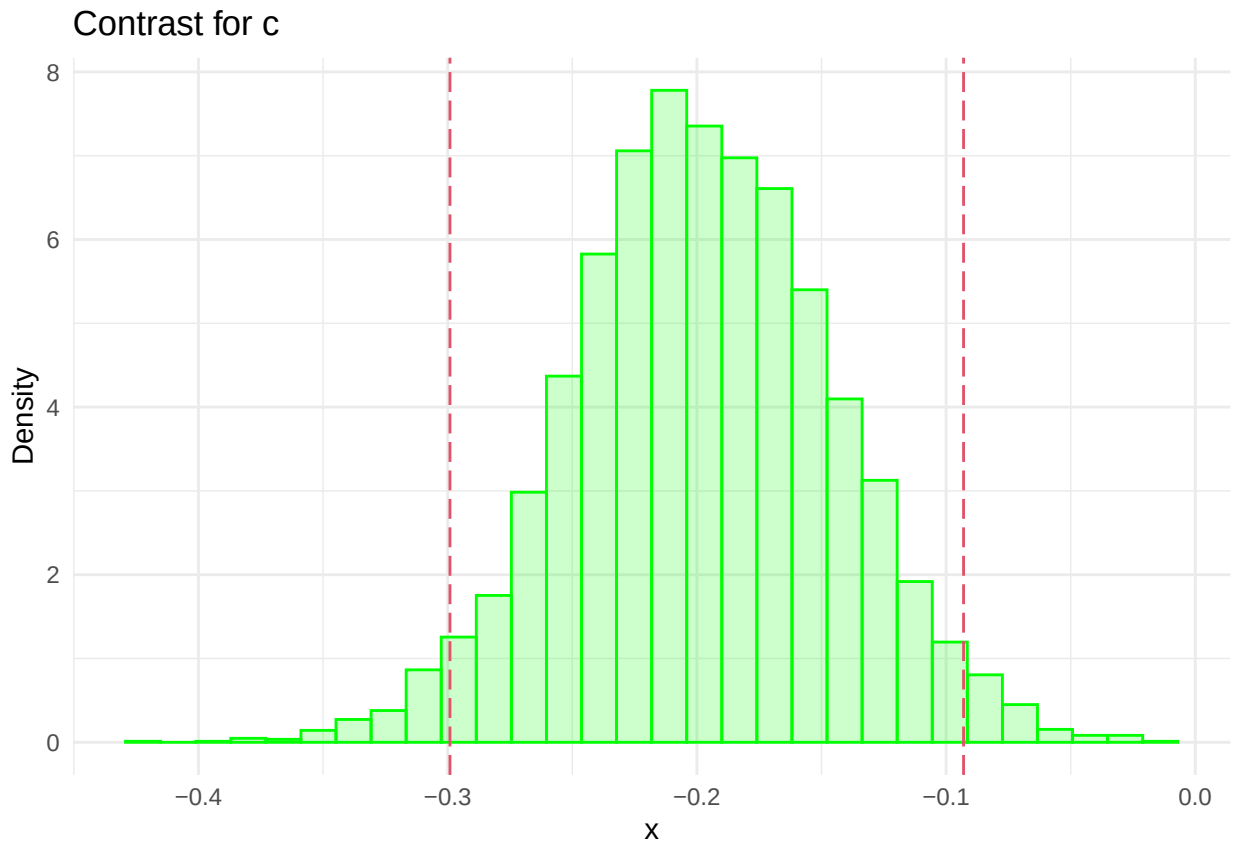
```
ggplot(data.frame(x=contrast.b2), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[4], fill=cols[4]) +
  geom_vline(xintercept=hdi(contrast.b2), color=2, lty="longdash") +
  labs(title="Contrast for b(ii)", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.b2 <- mean(contrast.b2 > 0); pps.contrast.b2
```

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

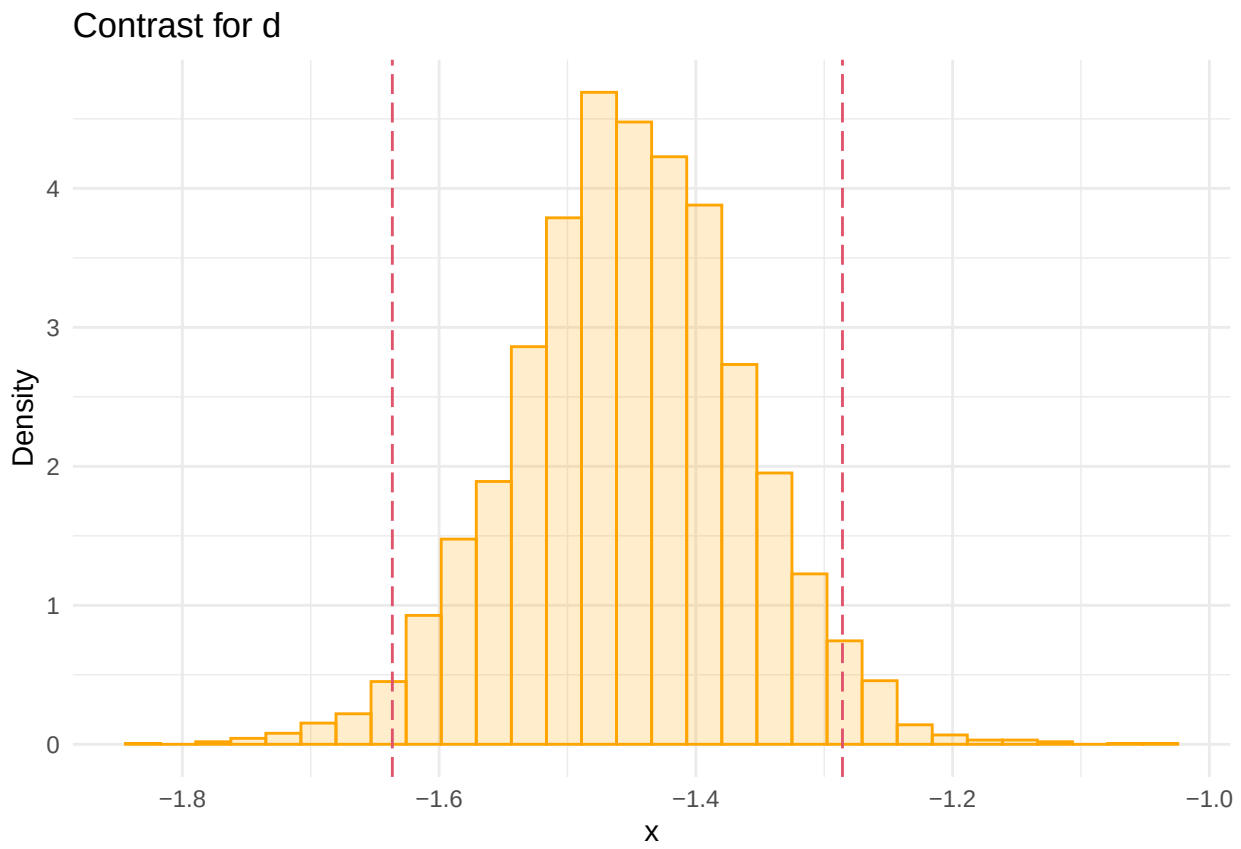
```
ggplot(data.frame(x=contrast.c), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[5], fill=cols[5]) +
  geom_vline(xintercept=hdi(contrast.c), color=2, lty="longdash") +
  labs(title="Contrast for c", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.c <- mean(contrast.c > 0); pps.contrast.c
```

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

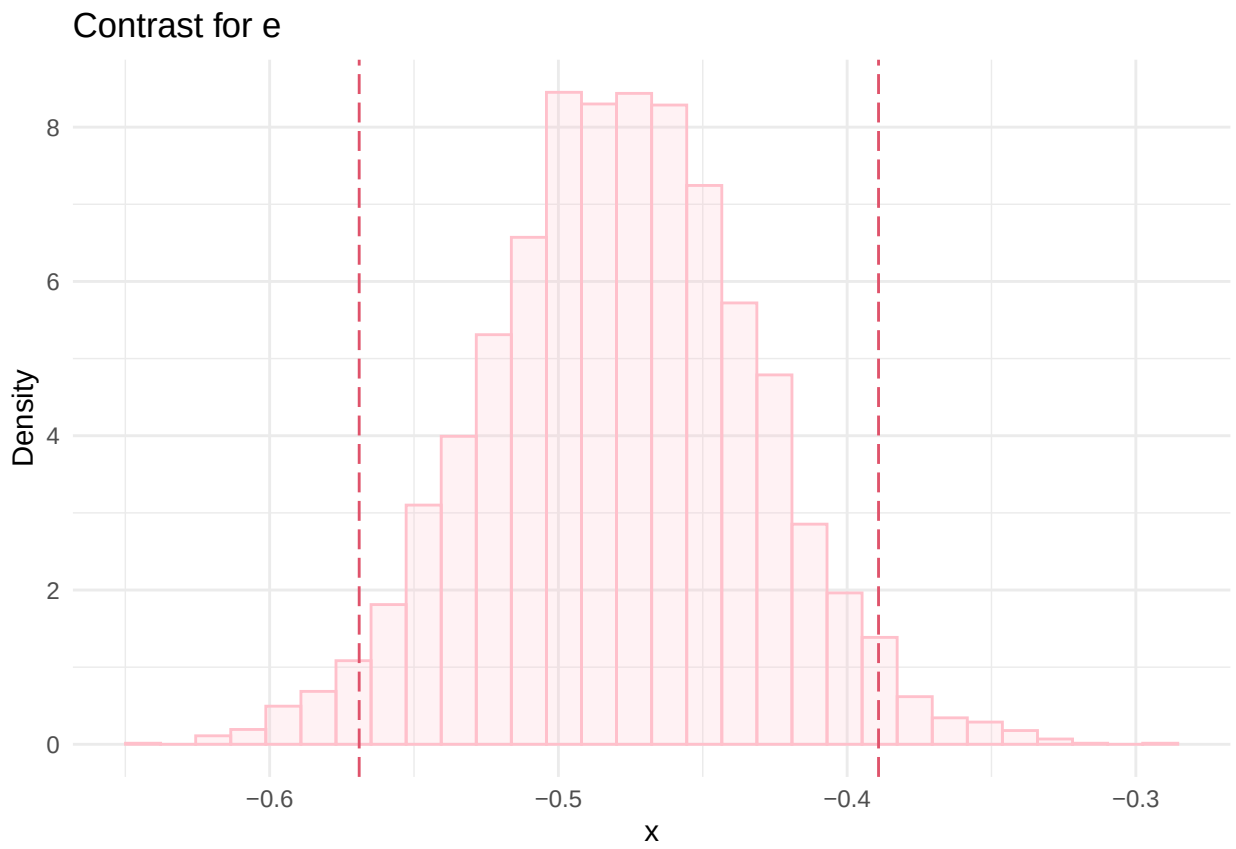
```
ggplot(data.frame(x=contrast.d), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[6], fill=cols[6]) +
  geom_vline(xintercept=hdi(contrast.d), color=2, lty="longdash") +
  labs(title="Contrast for d", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.d <- mean(contrast.d > 0); pps.contrast.d
```

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

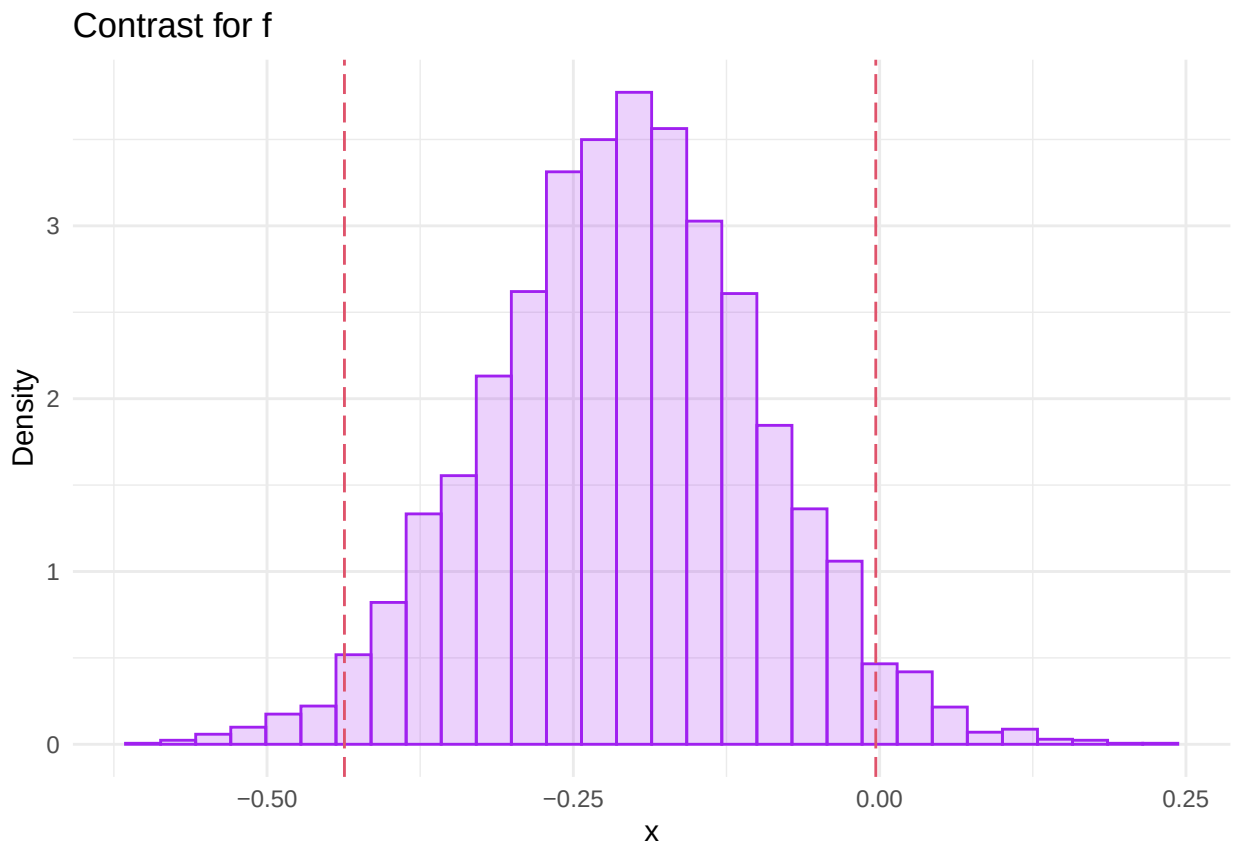
```
ggplot(data.frame(x=contrast.e), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[7], fill=cols[7]) +
  geom_vline(xintercept=hdi(contrast.e), color=2, lty="longdash") +
  labs(title="Contrast for e", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.e <- mean(contrast.e > 0); pps.contrast.e
```

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

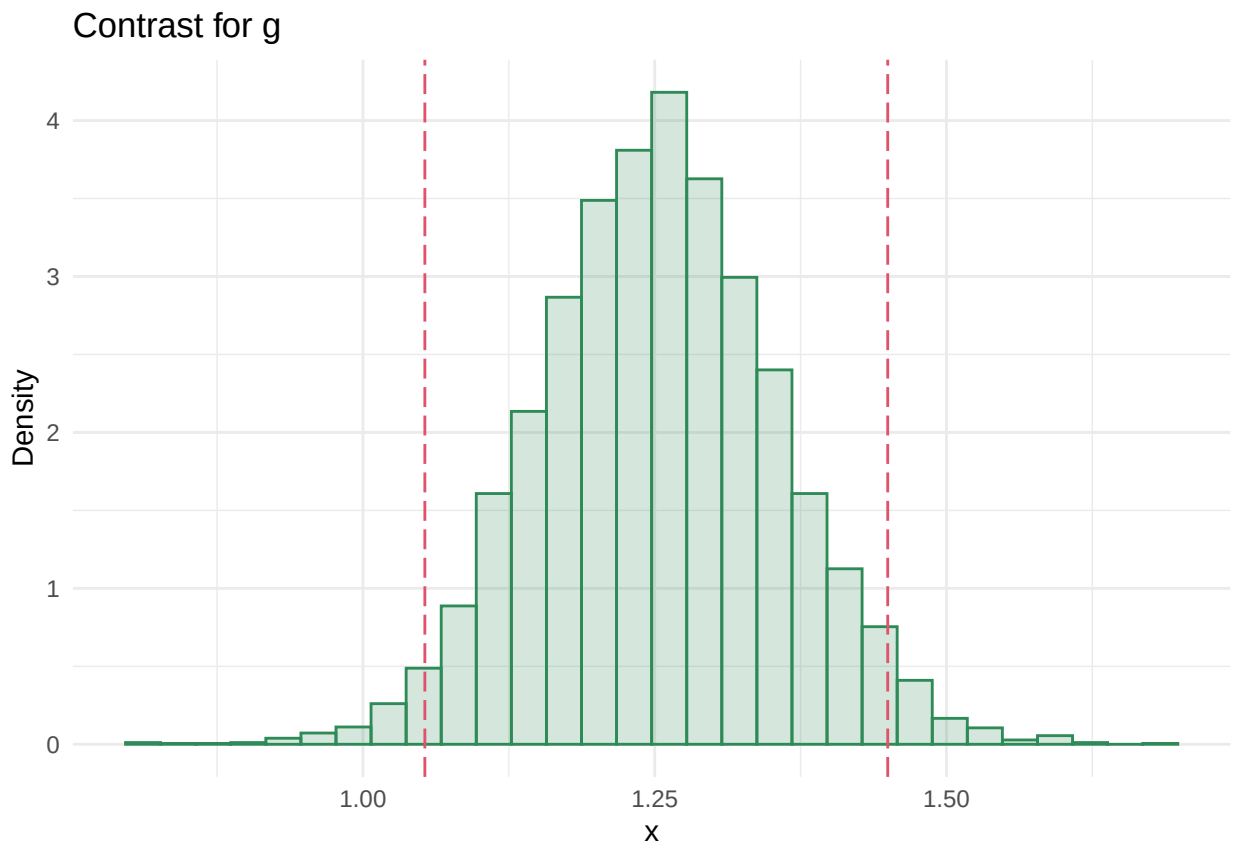
```
ggplot(data.frame(x=contrast.f), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[8], fill=cols[8]) +
  geom_vline(xintercept=hdi(contrast.f), color=2, lty="longdash") +
  labs(title="Contrast for f", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.f <- mean(contrast.f > 0); pps.contrast.f
```

```
## [1] 0.03033333
```

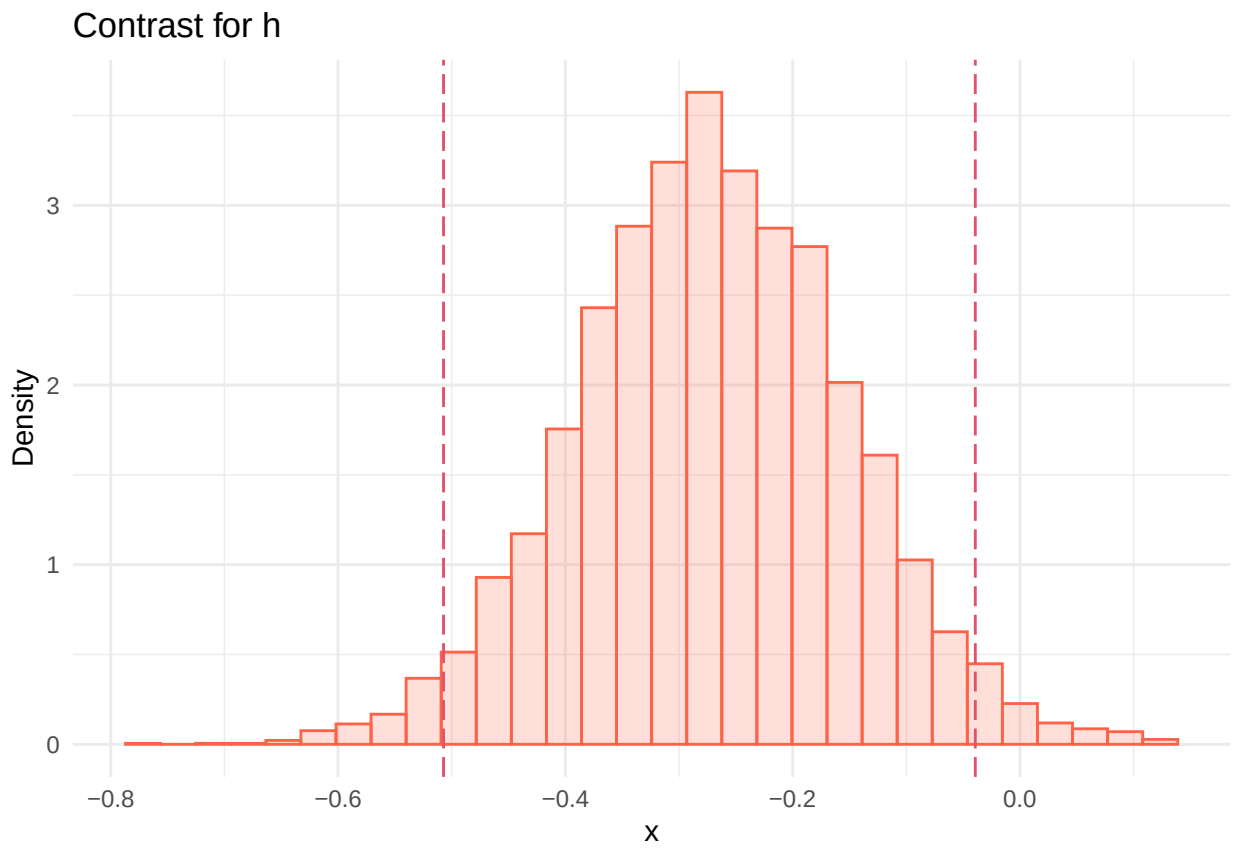
```
ggplot(data.frame(x=contrast.g), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[9], fill=cols[9]) +
  geom_vline(xintercept=hdi(contrast.g), color=2, lty="longdash") +
  labs(title="Contrast for g", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.g <- mean(contrast.g > 0); pps.contrast.g
```

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

```
ggplot(data.frame(x=contrast.h), aes(x=x, y=..density..)) +
  geom_histogram(bins=30, alpha=0.2, color=cols[10], fill=cols[10]) +
  geom_vline(xintercept=hdi(contrast.h), color=2, lty="longdash") +
  labs(title="Contrast for h", y="Density") + theme_minimal()
```




```
## Calculate posterior probability of superiority
pps.contrast.h <- mean(contrast.h > 0); pps.contrast.h
```

```
## [1] 0.01266667
```

```
pps.summary <- data.frame(
  c(pps.contrast.a1, pps.contrast.a2, pps.contrast.b1, pps.contrast.b2, pps.contrast.c,
    pps.contrast.d, pps.contrast.e, pps.contrast.f, pps.contrast.g, pps.contrast.h)
)
colnames(pps.summary) <- "PPS"
rownames(pps.summary) <- c("Contrast a(i)", "Contrast a(ii)",
                           "Contrast b(i)", "Contrast b(ii)",
                           "Contrast c", "Contrast d",
                           "Contrast e", "Contrast f",
                           "Contrast g", "Contrast h")
knitr::kable(pps.summary, align="c", digits=4)
```

	PPS
Contrast a(i)	0.0000
Contrast a(ii)	1.0000
Contrast b(i)	0.0000
Contrast b(ii)	1.0000
Contrast c	0.0000
Contrast d	0.0000

	PPS
Contrast e	0.0000
Contrast f	0.0303
Contrast g	1.0000
Contrast h	0.0127