Il significato sociale della musica in età evolutiva

Laboratorio didattico: esempio di relazione

Corrado Caudek

2022-01-30

Motivazione

Mehr, Song e Spelke (2016) discutono i significati sociali della musica in età evolutiva. Secondo questi ricercatori, la musica è capace di trasmettere significati sociali anche nel caso di bambini molto piccoli. Per valutare questa ipotesi, Mehr et al. (2016) esaminano un campione di 32 bambini di età media pari a 5.6 mesi (SD = 0.31, gamma di variazione: 5.1–6.1). Nel primo esperimento, Mehr e al. si chiedono se la familiarità con una melodia possa modulare la preferenza sociale, ovvero i tempi di fissazione dello sguardo (*looking-time*) che i bambini dirigono verso un individuo adulto sconosciuto che, cantando, produce una melodia familiare oppure non familiare.

Metodo

Nella prima fase dell'esperimento, ai genitori dei bambini veniva chiesto di imparare una nuova ninna-nanna e di ripeterla spesso al figlio/a. Nella fase test dell'esperimento, ai bambini venivano presentati due video, uno di fianco all'altro. Uno dei due video faceva vedere uno sconosciuto che cantava la melodia che, in precedenza, i genitori avevano presentato ai bambini; l'altro video faceva vedere un altro sconosciuto che cantava una ninna-nanna non conosciuta ai bambini. La variabile dipendente era il tempo di fissazione dello sguardo del bambino.

Nel pre-test, ai bambini venivano mostrati gli stessi due adulti utilizzati nella fase test; in questo caso, però, gli attori si limitavano a sorridere.

Risultati

Mediante una prima analisi statistica (relativa ai dati del pre-test) i ricercatori si sono chiesti se i bambini preferissero uno dei due sconosciuti, quando non cantavano. In una seconda analisi statistica, i ricercatori si sono chiesti se i bambini dimostrano una preferenza per lo sconosciuto che canta la ninna-nanna familiare, piuttosto che per lo sconosciuto che canta la ninna-nanna non familiare – ciò dovrebbe corrispondere a tempi di fissazione maggiri per il primo volto rispetto al secondo.

L'analisi statistica eseguita dagli autori è un test t di Student. L'articolo di Mehr et al. (2016) riporta un valore della statistica test pari a $t_{31} = 2.96$, un p-valore di 0.06 e un intervallo di confidenza al 95% pari a [0.529, 0.658]. La dimensione dell'effetto, misurata con il d di Cohen, è pari a 0.52.

Conclusioni

Mehr e al. (2016) riportano che i bambini di cinque mesi dirigono in modo preferenziale la propria attenzione verso un adulto sconosciuto che canta una melodia familiare piuttosto che verso un adulto sconosciuto che canta una melodia simile, ma sconosciuta. Questi risultati sono stati trovati quando la familiarizzazione con la ninna-nanna target veniva creata mediante l'interazione tra il bambino e un adulto che faceva parte dell'ambiente domestico. L'attenzione dei bambini, invece, non veniva modulata dalla familiarità con la melodia nel caso in cui la melodia in questione fosse stata presentata in precedenza al bambino in un modo

diverso, ovvero essendo prodotta da un giocattolo, o da un adulto che il bambino conosceva poco. In base a tali risultati, Mehr e al. (2016) concludono che le melodie prodotte nell'ambiente domestico, nell'interazione tra i bambini e gli adulti, sono dotate di un particolare significato sociale per i bambini.

Scopo del progetto

Lo scopo dell'analisi statistica che verrà qui eseguita è quello di replicare alcuni dei risultati riportati da Mehr et al. (2016) utilizzando i dati resi disponibili dagli autori. Le stesse analisi statistiche verranno eseguite usando due metodi diversi: l'approccio frequentista (quello usato nell'articolo) e l'approccio bayesiano.

Analisi dei dati

Per replicare i risultati riportati da Mehr et al. (2016), iniziamo a leggere in R i dati contenuti nel file `MehrSongSpelke_exp_1.csv`. Questo risultato si ottiene utilizzando la funzione `import()` contenuta nel pacchetto `rio`. Utilizziamo anche le funzionalità dei pacchetti `tidyverse` per manipolare i dati e creare grafici.

Dopo avere cambiato la cartella di lavoro in RStudio, carichiamo i pacchetti che verranno utilizzati mediante le seguenti istruzioni:

```
suppressPackageStartupMessages({
  library("here")
  library("tidyverse")
  library("patchwork")
  library("bayesplot")
})

theme_set(bayesplot::theme_default(base_size = 12))
bayesplot::color_scheme_set("brightblue")

options(
  digits = 3,
  width = 68,
  str = strOptions(strict.width = "cut"),
  crayon.enabled = TRUE
)

knitr::opts_chunk$set(
  comment = "#>",
  collapse = TRUE,
  message = FALSE,
  warning = FALSE,
  error = FALSE,
  width = 68,
  fig.align = "center",
  fig.width = 6,
  fig.asp = 0.618, # 1 / phi
  fig.show = "hold",
  dpi = 300,
  fig.pos = "h",
  cache.extra = knitr::rand_seed,
  tidy.opts = list(width.cutoff = 70),
  tidy = "styler"
)
```

```
mehr <- rio::import("MehrSongSpelke_exp_1.csv")
```

I nomi delle colonne dell'oggetto `mehr` vengono restituiti usando la funzione `names()` che prende come argomento il nome del data.frame:

```
names(mehr)
```

Consideriamo la variabile `exp1`:

```
unique(mehr$exp1)
#> [1] 1 0
```

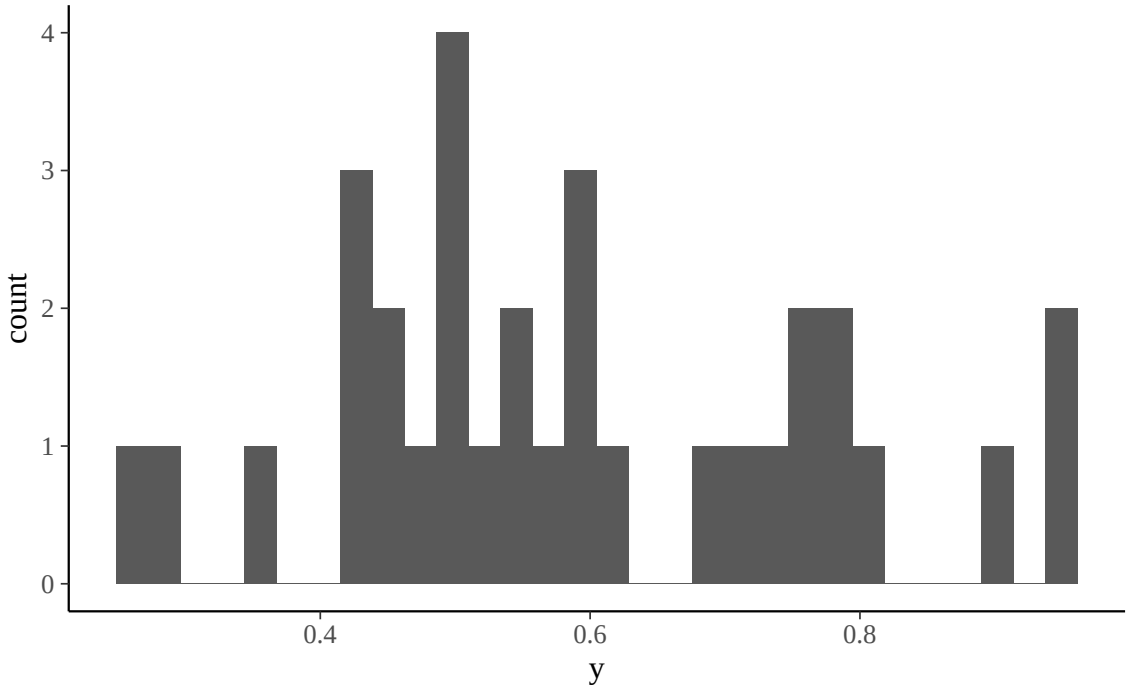
Quando `exp1` assume il valore 1, i dati appartengono al primo esperimento, quello di interesse qui. Per selezionare i dati del primo esperimento usiamo la funzione `filter()`:

```
df_exp1 <- mehr %>%
  dplyr::filter(exp1 == 1)
```

L'istruzione precedente dice che l'oggetto `mehr` (che è un DataFrame) viene passato alla funzione `filter()` contenuta nel pacchetto `dplyr`. L'argomento passato a `filter()` istruisce R a selezionare solo le righe del data.frame `mehr` nelle quali la colonna `exp` ha valore 1. Il sottoinsieme delle righe selezionate corrispone alle osservazioni che fanno parte del primo esperimento. L'operatore di attribuzione `<-` assegna al risultato che abbiamo ottenuto (ovvero, ai dati del primo esperimento) il nome `df_exp1`.

La variabile `Test_Proportion_Gaze_to_Singer` corrisponde al tempo di fissazione dello sguardo (espresso nei termini di una proporzione). Costruiamo un istogramma:

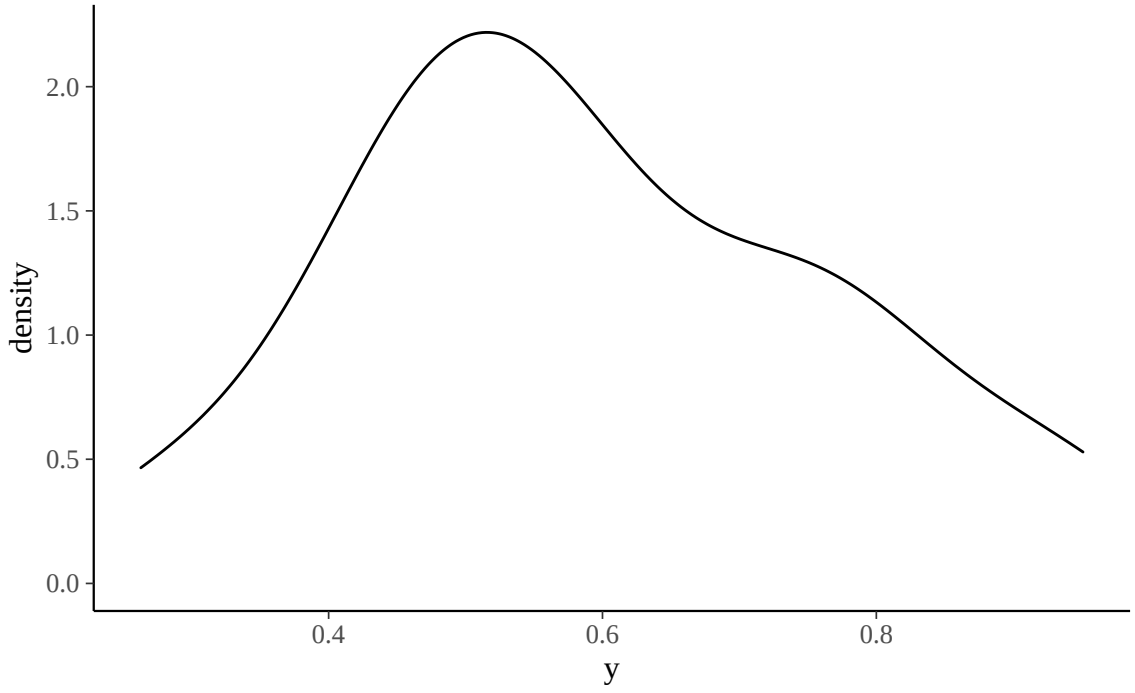
```
tibble(y = df_exp1$Test_Proportion_Gaze_to_Singer) %>%
ggplot(aes(y)) +
  geom_histogram()
```



Più utile è una stima della densità della frequenza dei dati:

```
tibble(y = df_exp1$Test_Proportion_Gaze_to_Singer) %>%
ggplot(aes(y)) +
```

```
  geom_density()
```



Media e deviazione standard delle proporzioni di tempo trascorso a fissare il volto target (sconosciuto che canta la ninna-nanna familiare) si ottengono nel modo seguente:

```
df_exp1 %>%
  summarise(
    p = mean(Test_Proportion_Gaze_to_Singer),
    std = sd(Test_Proportion_Gaze_to_Singer)
  )
#>       p   std
#> 1 0.593 0.179
```

Approccio frequentista

Test t di Student per un solo campione

Utilizziamo ora l'approccio frequentista per fare inferenza su μ, ovvero sulla proporzione media del tempo di fissazione rivolto allo sconosciuto target, piuttosto che verso lo sconosciuto che canta una ninna-nanna ignota al bambino.

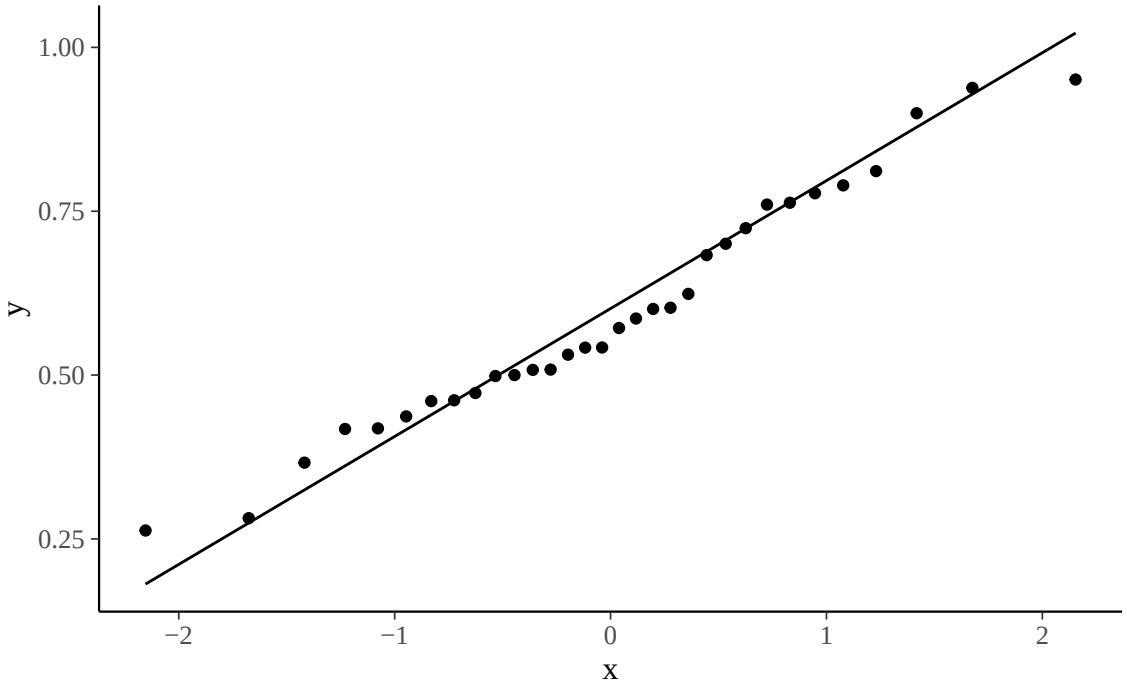
L'ipotesi nulla è

$$H_0 : \mu = 0.5.$$

Se non vi è alcuna preferenza, la variabile `Test_Proportion_Gaze_to_Singer` dovrebbe assumere il valore 0.5. Se invece i bambini preferiscono la melodia familiare, il valore medio di `Test_Proportion_Gaze_to_Singer` dovrebbe essere maggiore di 0.5; se i bambini preferiscono la melodia sconosciuta, il valore medio di `Test_Proportion_Gaze_to_Singer` dovrebbe essere minore di 0.5.

Iniziamo ad esaminare la distribuzione dei dati. Usiamo il diagramma quantile-quantile per verificare l'ipotesi di gaussianità (necessaria per eseguire il test statistico riportato dagli autori):

```
tibble(y = df_exp1$Test_Proportion_Gaze_to_Singer) %>%
  ggplot(aes(sample = y)) +
  stat_qq() +
  stat_qq_line()
```



Dato che l'assunzione di gaussianità è ragionevole, procediamo eseguendo il test t di Student. La proporzione del looking-time diretta allo sconosciuto che canta la melodia familiare è

```
m <- mean(df_exp1$Test_Proportion_Gaze_to_Singer)
m
#> [1] 0.593
```

pari a 0.59. Abiamo trovato tale valore usando la funzione `mean()` che prende, come argomento, il vettore che contiene la proporzione del looking-time diretta allo sconosciuto che canta la melodia familiare per ciascun bambino nel primo esperimento. Utilizziamo la sintassi `$` per estrarre dal DataFrame `df_exp1` la colonna di dati `Test_Proportion_Gaze_to_Singer`.

L'errore standard della media è

$$\sigma_{\bar{y}} = \frac{\sigma}{\sqrt{n}}$$

ovvero

```
se <- sd(df_exp1$Test_Proportion_Gaze_to_Singer) / sqrt(length(df_exp1$Test_Proportion_Gaze_to_Singer))
se
#> [1] 0.0316
```

La funzione `length()` ritorna il numero di elementi di un vettore ovvero, nel nostro caso, l'ampiezza del campione.

Applichiamo ora la formula per il calcolo della statistica T, ovvero

$$T = \frac{\bar{y} - \mu_0}{\hat{\sigma}_{\bar{y}}} = \frac{0.59 - 0.50}{0.032} = 2.96$$

ovvero

```
T <- (m - 0.5) / se
T
#> [1] 2.96
```

I gradi di libertà sono:

```
dof <- length(df_exp1$Test_Proportion_Gaze_to_Singer) - 1
dof
#> [1] 31
```

Per un test bidirezionale, il p-valore diventa

```
round(2 * (1 - pt(T, dof)), 3)
#> [1] 0.006
```

Nell'istruzione precedente abbiamo calcolato l'area sottesa alla ditribuzione t_{31} nell'intervallo $[2.960, \infty]$ e l'abbiamo moltiplicata per 2. Per trovare l'area nell'intervallo $[2.960, \infty]$, abbiamo calcolato il valore della funzione di ripartizione in corrispondenza del valore 2.960 (ovvero l'area nell'intervallo $[-\infty, 2.960]$) e abbiamo sottratto tale valore da 1.0.

Gli stessi risultati, che coincidono con quelli trovati sopra e con quelli riportati dagli autori, si possono facilmente ottenere usando la funzione `t.test()`

```
t.test(df_exp1$Test_Proportion_Gaze_to_Singer, mu = 0.5)
#>
#>  One Sample t-test
#>
#> data:  df_exp1$Test_Proportion_Gaze_to_Singer
#> t = 3, df = 31, p-value = 0.006
#> alternative hypothesis: true mean is not equal to 0.5
#> 95 percent confidence interval:
#>  0.529 0.658
#> sample estimates:
#> mean of x
#>     0.593
```

L'intervallo di confidenza al 95% si calcola come

$$\text{stima del parametro} \pm t \times \text{errore standard}$$

Nel caso presente, abbiamo:

```
round(m + c(-1, 1) * qt(0.975, dof) * se, 3)
#> [1] 0.529 0.658
```

Tale risultato riproduce quello riportato da Mehr et al. (2016).

Una stima della dimensione dell'effetto si ottiene come il rapporto tra la differenza tra le medie (in questo caso, proporzioni) e la relativa deviazione standard:

```
round(
  (m - 0.5) / sd(df_exp1$Test_Proportion_Gaze_to_Singer),
  3
)
#> [1] 0.523
```

Anche questo risultato coincide con il valore riportato dagli autori.

Analisi Bayesiana

Ci poniamo ora il problema di ripetere le analisi statiche precedenti utilizzando l'approccio bayesiano. Carichiamo i pacchetti necessari:

```
suppressPackageStartupMessages({
  library("brms")
  library("rstan")
  library("cmdstanr")
  library("posterior")
  library("loo")
})

rstan_options(auto_write = TRUE) # avoid recompilation of models
options(mc.cores = parallel::detectCores()) # parallelize across all CPUs
Sys.setenv(LOCAL_CPPFLAGS = "-march=native") # improve execution time
rstan_options(auto_write = TRUE) # avoid recompilation of models
options(mc.cores = parallel::detectCores()) # parallelize across all CPUs
Sys.setenv(LOCAL_CPPFLAGS = "-march=native") # improve execution time
```

I dati sono i seguenti:

```
y <- sort(df_exp1$Test_Proportion_Gaze_to_Singer)
y
#>  [1] 0.263 0.282 0.366 0.418 0.419 0.437 0.460 0.462 0.473 0.499
#> [11] 0.500 0.508 0.508 0.531 0.542 0.542 0.572 0.586 0.601 0.603
#> [21] 0.624 0.683 0.700 0.724 0.760 0.763 0.777 0.789 0.811 0.900
#> [31] 0.938 0.951
```

Considereremo y (così come hanno fatto gli autori) come una variabile continua per la quale è sensato assumere un meccanismo generatore dei dati gaussiano di media e deviazione standard ignote. L'inferenza riguarda il parametro μ di tale distribuzione gaussiana.

Funzione `brm()`

Utilizzando delle distribuzioni a priori per i parametri debolmente informative, facciamo inferenza utilizzando la funzione `brm()` del pacchetto `brms`:

```
fit <- brm(
  data = df_exp1,
  family = gaussian(),
  Test_Proportion_Gaze_to_Singer ~ 1,
  prior = c(
    prior(normal(0, 2.0), class = Intercept),
    prior(cauchy(0, 2.0), class = sigma)
  ),
  iter = 4000,
  refresh = 0,
  chains = 4,
  backend = "cmdstanr"
)
#> Running MCMC with 4 chains, at most 8 in parallel...
#>
#> Chain 1 finished in 0.1 seconds.
```

```
#> Chain 2 finished in 0.0 seconds.
#> Chain 3 finished in 0.1 seconds.
#> Chain 4 finished in 0.1 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 0.1 seconds.
#> Total execution time: 0.3 seconds.
```

Le stime a posteriori dei parametri si recuperano nel modo seguente:

```
fixef(fit)
#>           Estimate Est.Error  Q2.5 Q97.5
#> Intercept    0.593     0.033 0.527 0.657
```

Nell'output, `Estimate` corrisponde alla stima della media a posteriori, mentre `Q2.5` e `Q97.5` forniscono gli estremi dell'intervallo di credibilità al 95%. I risultati sono simili a quelli riportati dagli autori.

Modello in linguaggio Stan

Ripetiamo ora l'analisi precedente utilizzano un modello scritto in linguaggio Stan. Il modello riportato di seguito verrà salvato nella cartella `code` con il nome `normalmodel.stan`:

```
modelString = "
data {
  int<lower=0> N;
  vector[N] y;
}
parameters {
  real mu;
  real<lower=0> sigma;
}
model {
  mu ~ normal(0.5, 0.5);
  sigma ~ cauchy(0, 1);
  y ~ normal(mu, sigma);
}
"
writeLines(modelString, con = "code/normalmodel.stan")
```

Sistemiamo i dati nel formato appropriato per Stan:

```
data_list <- list(
  N = length(df_exp1$Test_Proportion_Gaze_to_Singer),
  y = df_exp1$Test_Proportion_Gaze_to_Singer
)
data_list
#> $N
#> [1] 32
#>
#> $y
#>  [1] 0.603 0.683 0.724 0.282 0.499 0.951 0.418 0.938 0.500 0.586
#> [11] 0.473 0.508 0.811 0.572 0.777 0.263 0.508 0.437 0.542 0.601
#> [21] 0.419 0.789 0.760 0.624 0.366 0.462 0.900 0.531 0.542 0.700
#> [31] 0.763 0.460
```

Leggiamo il file in cui abbiamo salvato il codice Stan

```
file <- file.path("code", "normalmodel.stan")
```

compiliamo il modello

```
mod <- cmdstan_model(file)
```

ed eseguiamo il campionamento MCMC:

```
fit <- mod$sample(
  data = data_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = 123,
  chains = 4L,
  parallel_chains = 2L,
  refresh = 0,
  thin = 1
)
```

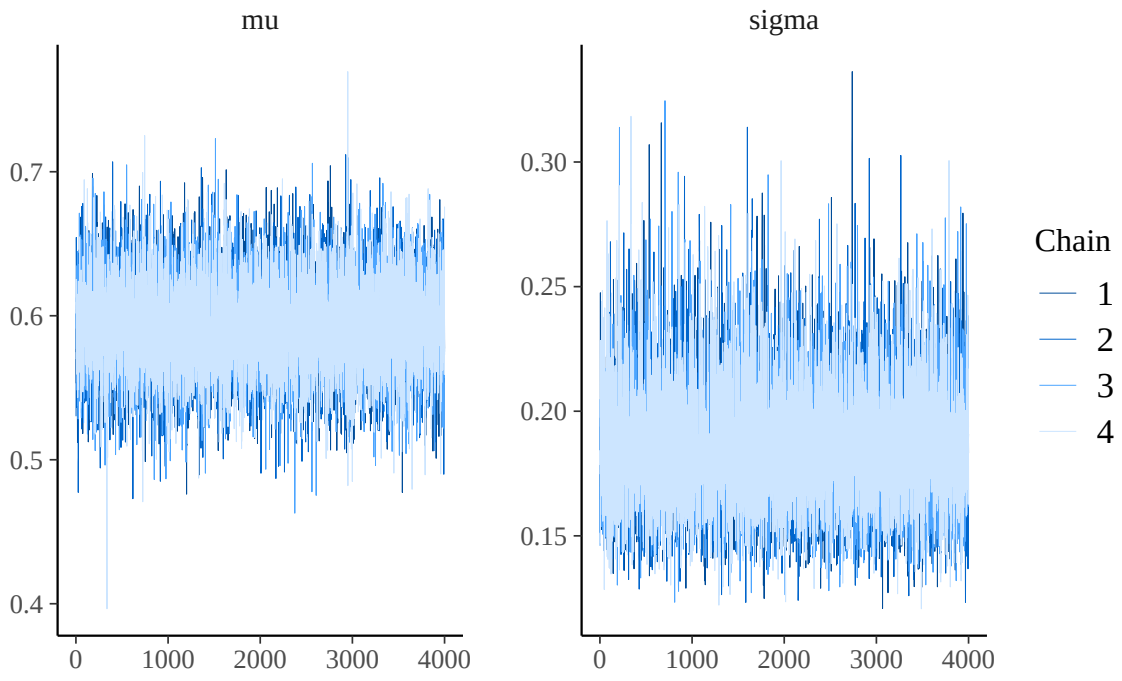
La convergenza e il "mixing" del campionamanto MCMC possono essere controllate mediante il *trace plot* che mostra l'andamento delle simulazioni e ci dice se stiamo effettivamente utilizzando una distribuzione limite:

Convertiamo l'oggetto `fit` in formato `stanfit`,

```
stanfit <- rstan::read_stan_csv(fit$output_files())
```

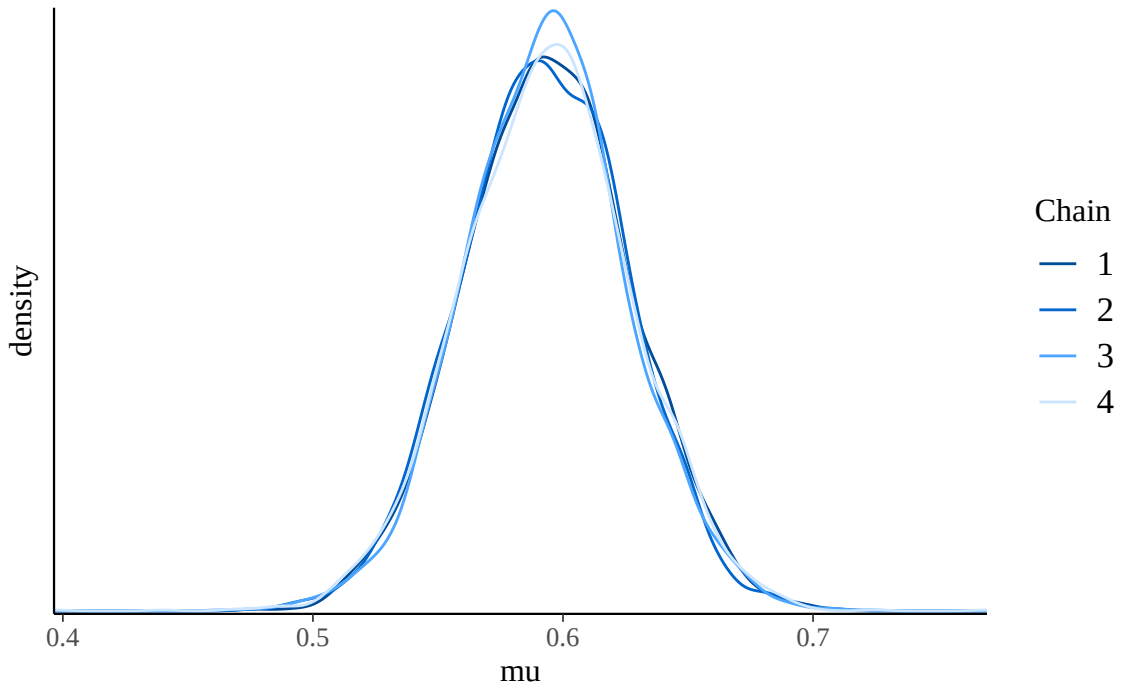
Esaminiamo i trace-plot:

```
stanfit %>%
  mcmc_trace(pars = c("mu", "sigma"), size = 0.1)
```



La stima della distribuzione a posteriori per μ per ciascuna delle quattro catene usate può essere rappresentata graficamente nel modo seguente:

```
mcmc_dens_overlay(stanfit, pars = "mu") +
  ylab("density")
```



Esaminiamo la statistica $\hat{R}$:

```
bayesplot::rhat(stanfit, pars = c("mu", "sigma"))
#>    mu sigma
#>     1     1
```

Utilizzando l'oggetto `stanfit`, possiamo recuperare la statistica di Geweke nel modo seguente:

```
fit_mcmc <- As.mcmc.list(
  stanfit,
  pars = c("mu", "sigma")
)
coda::geweke.diag(fit_mcmc, frac1 = .1, frac2 = .5)
#> [[1]]
#>
#> Fraction in 1st window = 0.1
#> Fraction in 2nd window = 0.5
#>
#>    mu sigma
#>  1.21  1.60
#>
#>
#> [[2]]
#>
#> Fraction in 1st window = 0.1
#> Fraction in 2nd window = 0.5
#>
#>    mu sigma
#> 0.631 0.874
```

```
#>
#>
#> [[3]]
#>
#> Fraction in 1st window = 0.1
#> Fraction in 2nd window = 0.5
#>
#>     mu  sigma
#> -0.271 -0.589
#>
#>
#> [[4]]
#>
#> Fraction in 1st window = 0.1
#> Fraction in 2nd window = 0.5
#>
#>     mu  sigma
#> -0.276  0.180
```

La statistica di Geweke è uguale a zero quando le medie delle due porzioni della catena di Markov sono uguali. Valori maggiori di | 2 | suggeriscono che la catena non ha ancora raggiunto una distribuzione stazionaria.

Le stime a posteriori dei parametri si ottengono con

```
s <- summary(
  stanfit,
  c("mu", "sigma"),
  probs = c(0.025, 0.50, 0.975),
  use_cache = TRUE
  )
s$summary  # all chaines merged
#>        mean  se_mean     sd  2.5%   50% 97.5% n_eff Rhat
#> mu    0.594 0.000309 0.0332 0.529 0.594 0.659 11504    1
#> sigma 0.186 0.000235 0.0251 0.144 0.183 0.241 11427    1
```

I risultati replicano quelli ottenuti in precedenza.

Modifichiamo ora il modello inserendo una distribuzione a priori più sensata per il parametro μ. Essendo una proporzione, deve essere un numero compreso tra 0 e 1. Pertanto, quale distribuzione a priori, è sensato usare una distribuzione Beta. Utilizzeremo qui una Beta(2,2), ovvero una distribuzione a priori debolmente informativa.

```
modelString = "
data {
  int<lower=0> N;
  vector[N] y;
}
parameters {
  real mu;
  real<lower=0> sigma;
}
model {
  mu ~ beta(2, 2);
  sigma ~ cauchy(0, 1);
  y ~ normal(mu, sigma);
}
```

```
"
writeLines(modelString, con = "code/normalmodel2.stan")
```

Leggiamo il file in cui abbiamo salvato il codice Stan

```
file2 <- file.path("code", "normalmodel2.stan")
```

compiliamo il modello

```
mod2 <- cmdstan_model(file2)
```

ed eseguiamo il campionamento MCMC:

```
fit2 <- mod2$sample(
  data = data_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = 123,
  chains = 4L,
  parallel_chains = 2L,
  refresh = 0,
  thin = 1
)
```

Troviamo le stime a posteriori dei parametri:

```
stanfit2 <- rstan::read_stan_csv(fit2$output_files())

s <- summary(
  stanfit2,
  c("mu", "sigma"),
  probs = c(0.025, 0.50, 0.975),
  use_cache = TRUE
  )
s$summary  # all chaines merged
#>        mean  se_mean     sd  2.5%   50% 97.5% n_eff Rhat
#> mu    0.593 0.000293 0.0334 0.527 0.594 0.659 13016    1
#> sigma 0.186 0.000227 0.0250 0.145 0.183 0.242 12073    1
```

I risultati sono quasi identici a quelli trovati in precedenza.

Possiamo dunque concludere, con un grado di certezza soggettiva del 95%, che siamo sicuri che la proporzione del tempo complessivo di fissazione dello sguardo verso lo sconosciuto che cantava la melodia familiare (piuttosto che verso lo sconosciuto che cantava una ninn-nanna ignota), è compresa nell'intervallo [0.527, 0.659].

Confronto tra due gruppi

Approccio frequentista (test t di Student per campioni appaiati)

In una diversa analisi statistica, Mehr e al. (2016) hanno confrontato il tempo di fissazione dello sguardo diretto verso lo sconosciuto che sorride soltanto e verso lo sconosciuto che canta la ninna-nanna familiare. Ci dovremmo aspettare un tempo di fissazione maggiore nel secondo caso rispetto al primo.

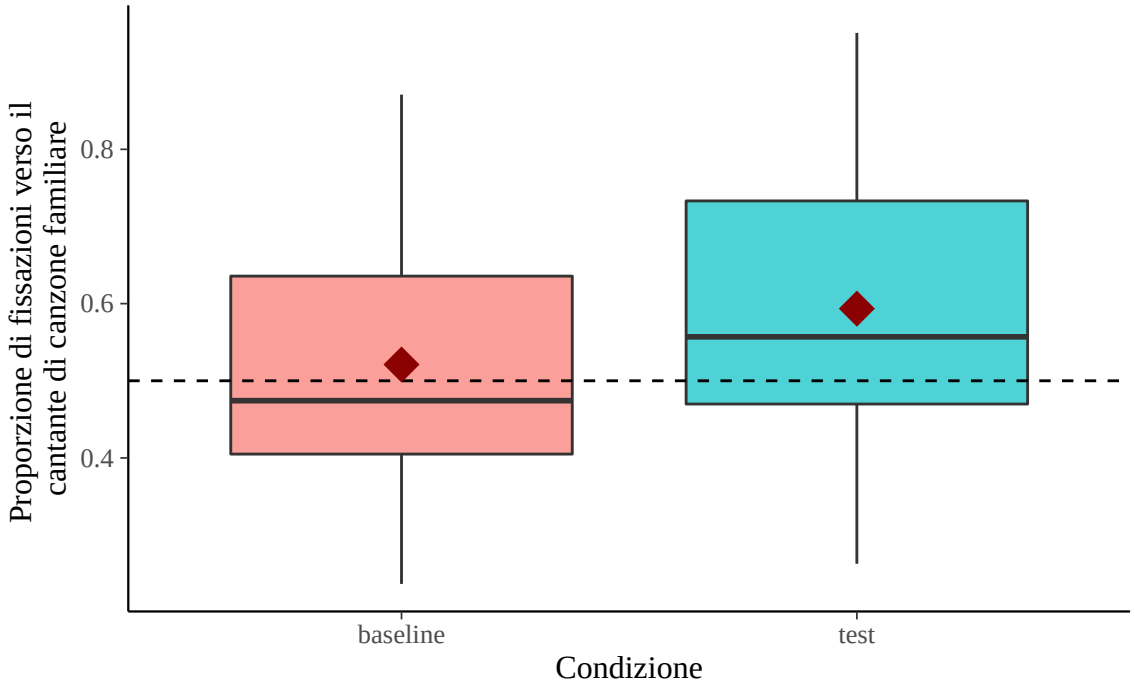
La distribuzione dei tempi di fissazione dello sguardo nelle due condizioni è presentata nel pannello a della Figura 2 del loro articolo. Riproduciamo qui sotto la figura. I tempi di fissazione dello sguardo nelle condizioni baseline e test corrispondono, rispettivamente, alle variabili `Baseline_Proportion_Gaze_to_Singer` e `Test_Proportion_Gaze_to_Singer`.

Per replicare la Figura 2a riportata nell'articolo dobbiamo sistemare i dati nel formato "long' ', ovvero nel formato in cui a ciascuna colonna del DataFrame corrisponde una variabile. Nel caso presente, possiamo creare una variabile chiamata *condizione*, con modalità *baseline* e *test*, e una seconda variabile, *y*, che corrisponde ai tempi di fissazione dello sguardo. Per creare un DataFrame che contiene queste due variabili, procediamo come segue.

```
condizione <- factor(
  rep(
    c("baseline", "test"),
    each = length(df_exp1$Baseline_Proportion_Gaze_to_Singer)
    )
  )
y <- c(df_exp1$Baseline_Proportion_Gaze_to_Singer, df_exp1$Test_Proportion_Gaze_to_Singer)
df2 <- data.frame(condizione, y)
head(df2)
#>   condizione     y
#> 1   baseline 0.437
#> 2   baseline 0.413
#> 3   baseline 0.754
#> 4   baseline 0.439
#> 5   baseline 0.475
#> 6   baseline 0.871
dim(df2)
#> [1] 64  2
```

Il diagramma a scatola (simile a quello riportato nell'articolo) si crea con le seguenti istruzioni:

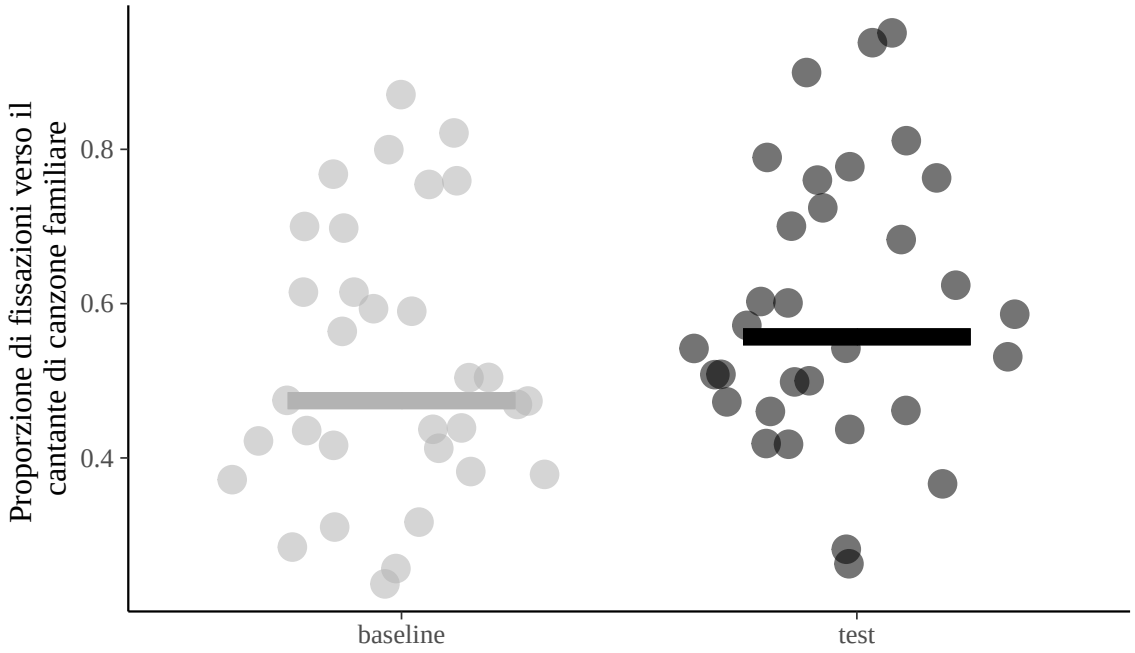
```
p <- df2 %>%
  ggplot(aes(x = condizione, y = y, fill = condizione)) +
  geom_boxplot(alpha = 0.7) +
  scale_y_continuous(
    name = "Proporzione di fissazioni verso il \ncantante di canzone familiare"
  ) +
  scale_x_discrete(name = "Condizione") +
  theme(legend.position = "none") +
  geom_hline(yintercept = 0.5, linetype = "dashed") +
  stat_summary(
    fun.y = mean, colour = "darkred", geom = "point",
    shape = 18, size = 6
  )
p
```



Una rappresentazione più efficace dei dati si ottiene riportando, per ciascuna condizione, tutte le singole osservazioni, insieme alla mediana di ciascun gruppo:

```
mehr_summary <- df2 %>%
  group_by(condizione) %>%
  summarize(
    lt_mean = mean(y),
    lt_sd = sd(y),
    lt_median = median(y)
  ) %>%
  ungroup()
```

```
df2 %>%
  ggplot(
    aes(x = condizione, y = y, color = condizione)
  ) +
  ggforce::geom_sina(aes(color = condizione, size = 3, alpha = .5)) +
  geom_errorbar(
    aes(y = lt_median, ymin = lt_median, ymax = lt_median),
    data = mehr_summary, width = 0.5, size = 3
  ) +
  labs(
    x = "",
    y = "Proporzione di fissazioni verso il \ncantante di canzone familiare"
  ) +
  theme(legend.position = "none") +
  scale_colour_grey(start = 0.7, end = 0)
```

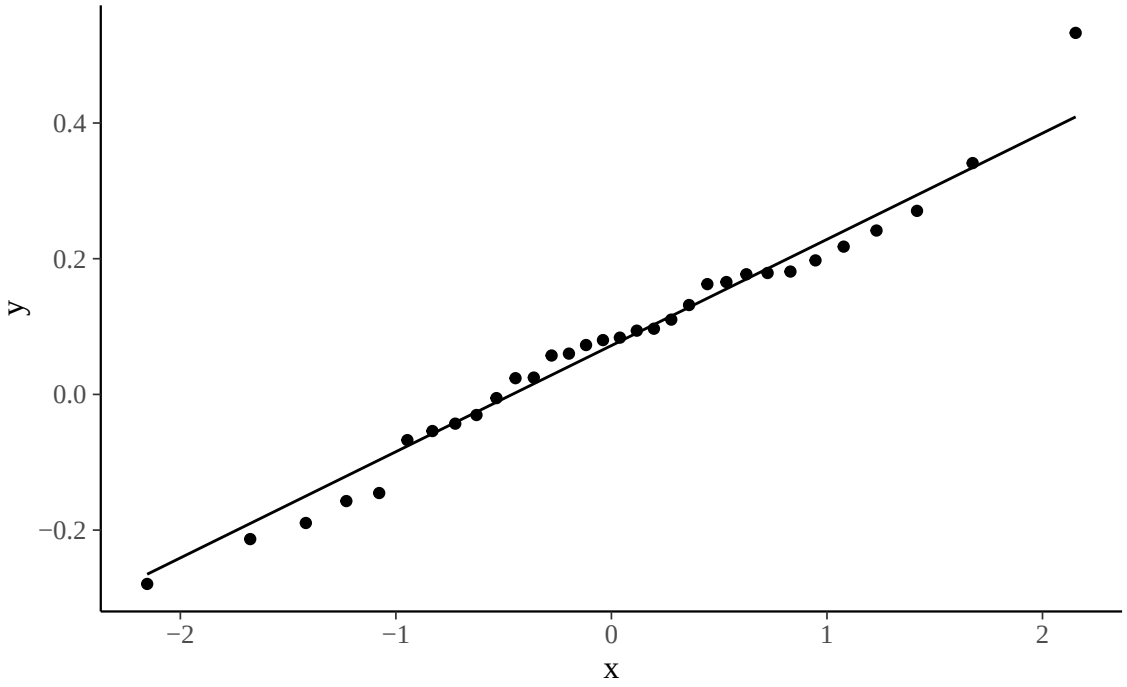


Un confronto tra queste due condizioni viene effettuato con un test t di Student per campioni appaiati. Ovvero

```
diff <- df_exp1$Test_Proportion_Gaze_to_Singer - df_exp1$Baseline_Proportion_Gaze_to_Singer
diff
#>  [1]  0.16561  0.27049 -0.03035 -0.15722  0.02390  0.08002  0.18104
#>  [8]  0.17894  0.08367 -0.21324  0.09395 -0.18951  0.21778 -0.04311
#> [15]  0.16254 -0.05399  0.19752 -0.06739  0.07277  0.09682 -0.14536
#> [22]  0.53284  0.06011  0.24154 -0.00545  0.17707  0.13171  0.05731
#> [29] -0.27932  0.11023  0.34093  0.02479
```

Verifichiamo la gaussianità dei dati:

```
tibble(y = diff) %>%
  ggplot(aes(sample = y)) +
  stat_qq() +
  stat_qq_line()
```



La media delle differenze è:

```
m <- mean(diff)
m
#> [1] 0.0724
```

L'errore standard delle differenze è:

```
se <- sd(diff) / sqrt(length(diff))
se
#> [1] 0.03
```

La statistica T è:

```
T <- (m - 0) / se
T
#> [1] 2.42
```

I gradi di libertà sono:

```
nu <- length(diff) - 1
nu
#> [1] 31
```

Il p-valore è:

```
2 * (1 - pt(T, nu))
#> [1] 0.0218
```

Usando `t.test()` otteniamo:

```
t.test(diff, mu = 0)
#>
#>  One Sample t-test
#>
```

```
#> data:  diff
#> t = 2, df = 31, p-value = 0.02
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  0.0113 0.1335
#> sample estimates:
#> mean of x
#>    0.0724
```

Questi risultati replicano i valori riportati dagli autori: $t_{31} = 2.42$, $p = 0.22$.

L'intervallo di confidenza è

```
mean(diff) + c(-1, 1) * qt(.975, nu) * se
#> [1] 0.0113 0.1335
```

La dimensione dell'effetto è

```
d <- mean(diff) / sd(diff)
d
#> [1] 0.427
```

Le stime dell'intervallo di confidenza e della dimensione dell'effetto coincidono con i valori riportati da Mehr e al. (2016).

Analisi bayesiana

Dato che, nel caso di campioni appaiati, abbiamo un unico valore della varibile dipendente per ciascuna "coppia", l'analisi statistica è sostanzialmente identica a quella già descritta sopra. In questo caso, la variabile dipendente è chiamata `diff`.

```
data2_list <- list(
  N = length(diff),
  y = diff
)
data2_list
#> $N
#> [1] 32
#>
#> $y
#>  [1]  0.16561  0.27049 -0.03035 -0.15722  0.02390  0.08002  0.18104
#>  [8]  0.17894  0.08367 -0.21324  0.09395 -0.18951  0.21778 -0.04311
#> [15]  0.16254 -0.05399  0.19752 -0.06739  0.07277  0.09682 -0.14536
#> [22]  0.53284  0.06011  0.24154 -0.00545  0.17707  0.13171  0.05731
#> [29] -0.27932  0.11023  0.34093  0.02479
```

Usiamo nuovamente il modello `normalmodel2.stan` che abbiamo già compilato in precedenza ed eseguiamo il campionamento MCMC:

```
fit3 <- mod2$sample(
  data = data2_list,
  iter_sampling = 4000L,
  iter_warmup = 2000L,
  seed = 123,
  chains = 4L,
  parallel_chains = 2L,
  refresh = 0,
```

```
  thin = 1
)
```

Le stime a posteriori dei parametri si ottengono con:

```
stanfit3 <- rstan::read_stan_csv(fit3$output_files())
```

```
s <- summary(
  stanfit3,
  c("mu", "sigma"),
  probs = c(0.025, 0.50, 0.975),
  use_cache = TRUE
  )
s$summary  # all chaines merged
#>         mean  se_mean     sd   2.5%   50% 97.5% n_eff Rhat
#> mu    0.085 0.000281 0.0286 0.0322 0.084 0.146 10406    1
#> sigma 0.176 0.000233 0.0238 0.1374 0.174 0.231 10392    1
```

I risultati sono molto simili ai precedenti.

Dall'output precedente possiamo ottenere una stima della dimensione dell'effetto:

```
s$summary[1, 1] /   s$summary[2, 1]
#> [1] 0.482
```

Anche in questo caso, il risultato è simile a quello ottenuto sopra.

Conclusioni

Utilizzando i dati grezzi messi a disposizione dagli autori siamo stati in grado di replicare i risultati riportati dell'articolo di Mehr e al. (2016). In questa relazione ci siamo focalizzati su due problemi: l'inferenza su una media e l'inferenza su due medie nel caso di dati appaiati. Nel caso presente, i risultati dell'analisi bayesiana sono praticamente identici a quelli prodotti dall'approccio frequentista, se vengono usate distribuzioni a priori debolmente informative. Utilizzando `ggplot()` siamo anche stati in grado di replicare una delle figure riportate nell'articolo di Mehr e al. (2016). Abbiamo anche proposto una nuova versione di tale figura che sembra più informativa di quella proposta dagli autori.

Le analisi statistiche descritte da Mehr e al. (2016) sono convincenti: i dati non mostrano anomalie degne di nota e le assunzioni alla base dei test statistici svolti dagli autori – ovvero, la gaussianità dei dati – sono sensate. È dunque sensato concludere che la musica sembra possedere un importante valore sociale anche nel caso di bambini molto piccoli.

Riferimenti bibliografici

Mehr, S. A., Song, L. A., & Spelke, E. S. (2016). For 5-month-old infants, melodies are social. *Psychological Science*, *27*(4), 486-501.