Tarea5

Alberto, Ivan, Sara, Valeria

Tarea 5

1. hierarchical_betaBlocker

Los datos en el archivo hierarchical_betaBlocker.csv muestran los resultados de 22 ensayos incluídos en un meta-análisis de datos de ensayos clínicos sobre el efecto de los beta-bloqueadores en la reducción de riesgo de infarto.

El objetivo de este meta-análisis es determinar un estimador robusto del efecto de los beta-bloqueadores combinando información de un rango de estudios previos.

a. Posterior

Comienza suponiendo que el número de muertes en los grupos de control (r_i^c) y de tratamiento (r_i^t) de cada ensayo están dados por distribuciones binomiales de la forma: $r_i^c \sim \mathbf{Bin}(n_i^c, p_i^c)$ y $r_i^t \sim \mathbf{Bin}n_i^t, p_i^t$, donde (n_i^c, n_i^t) son el número de individuos en los grupos de control y tratamiento respectivamente.

Adicionalmente suponer que las probabilidades de mortalidad en los conjuntos de tratamiento y control están dados por: $logit(p_i^c) = \mu_i$ y $logit(p_i^t) = \mu_i + \delta_i$. Se espera que $\delta_i < 0$ si los beta-bloqueadores tienen el efecto deseado. Se asumen las siguientes iniciales para los parámetros: $\mu_i \sim \mathcal{N}(0, 10)$ y $\delta_i \sim \mathcal{N}(0, 10)$.

Estimar la posterior para δ_i usando el modelo indicado. Notar que para este modelo no hay interdependencia entre los estudios.

El problema que estamos abordando es el de estimar la efectividad de los beta-bloqueadores en la reducción del riesgo de infarto. Para ello, tenemos datos de 22 ensayos clínicos, cada uno con información sobre el número de muertes en los grupos de tratamiento y control, así como el tamaño de la muestra en cada grupo.

Buscamos estimar la diferencia en la probabilidad de mortalidad entre los grupos de tratamiento y control, representada por δ_i. Esta diferencia se modela utilizando un enfoque

de regresión logística, donde las probabilidades de mortalidad en los grupos de tratamiento (p_i^t) y control (p_i^c) se expresan en términos de los parámetros μ_i y δ_i, respectivamente.

Sabemos que:

- μ_i representa la probabilidad logit de mortalidad en el grupo de control
- δ_i representa la diferencia en la probabilidad de mortalidad entre los grupos de tratamiento y control.

De acuerdo a los datos, especificamos el siguiente modelo en stan:

```
# Modelo en Stan
stan_code <- '
data {
  int<lower=0> J;           // Número de estudios
  int rt[J];                // Número de muertes en tratamiento
  int nt[J];                // Tamaño de la muestra en tratamiento
  int rc[J];                // Número de muertes en control
  int nc[J];                // Tamaño de la muestra en control
}

parameters {
  real delta[J];            // Efecto del tratamiento
  real mu[J];               // Efecto del control
}

model {
  for (j in 1:J) {
    mu[j] ~ normal(0, 10);      // Priori para el efecto del control
    delta[j] ~ normal(0, 10);   // Priori para el efecto del tratamiento

    // Verosimilitud de los datos
    rc[j] ~ binomial_logit(nc[j], mu[j]);                              // Verosimilitud del
    rt[j] ~ binomial_logit(nt[j], mu[j] + delta[j]);                    // Verosimilitud de
  }
}
'

# Compilar el modelo
stan_model <- stan_model(model_code = stan_code)
```

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

```
Running /usr/lib/R/bin/R CMD SHLIB foo.c
using C compiler: ‘gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0’
gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-
In file included from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-pc-linux-gnu/StanHead
                 from <command-line>:
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
  628 | namespace Eigen {
      | ^~~~~~~~~
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
  628 | namespace Eigen {
      |                 ^
In file included from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-pc-linux-gnu/StanHead
                 from <command-line>:
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
   96 | #include <complex>
      |          ^~~~~~~~~
compilation terminated.
make: *** [/usr/lib/R/etc/Makeconf:191: foo.o] Error 1


SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.8e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
```

```
Chain 1:
Chain 1:  Elapsed Time: 0.205 seconds (Warm-up)
Chain 1:                0.121 seconds (Sampling)
Chain 1:                0.326 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.4e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.19 seconds (Warm-up)
Chain 2:                0.111 seconds (Sampling)
Chain 2:                0.301 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.4e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
```

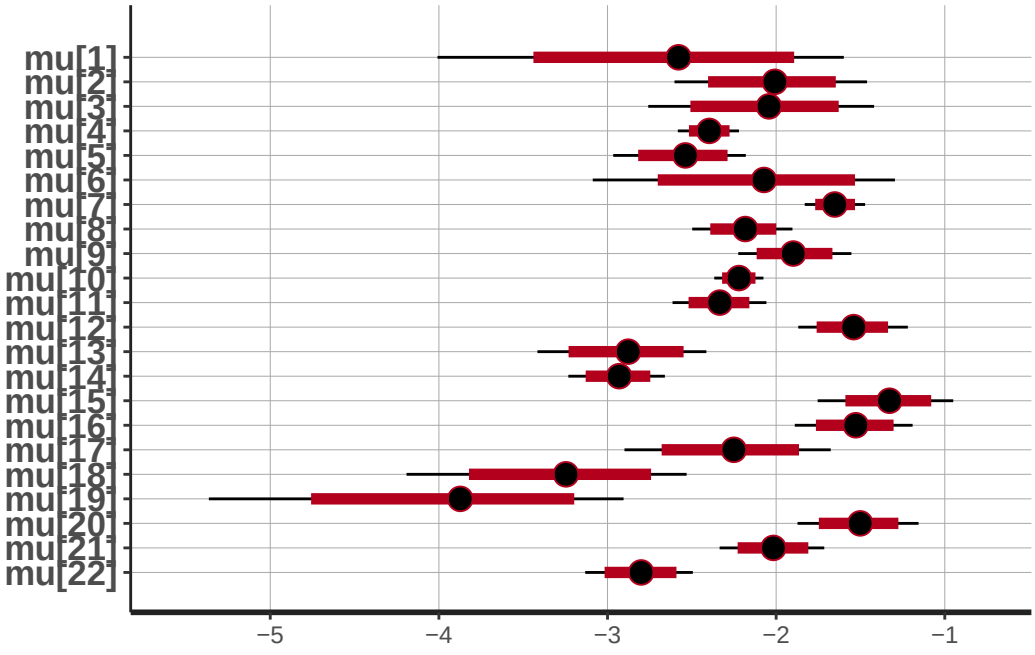
```
Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.183 seconds (Warm-up)
Chain 3:                0.146 seconds (Sampling)
Chain 3:                0.329 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.4e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.177 seconds (Warm-up)
Chain 4:                0.113 seconds (Sampling)
Chain 4:                0.29 seconds (Total)
Chain 4:
```

A continuación presentamos una gráfica para observar el rango de valores para cada μ_i

```r
plot(stan_samples, pars = c("mu"))
```

```
ci_level: 0.8 (80% intervals)
```

```
outer_level: 0.95 (95% intervals)
```

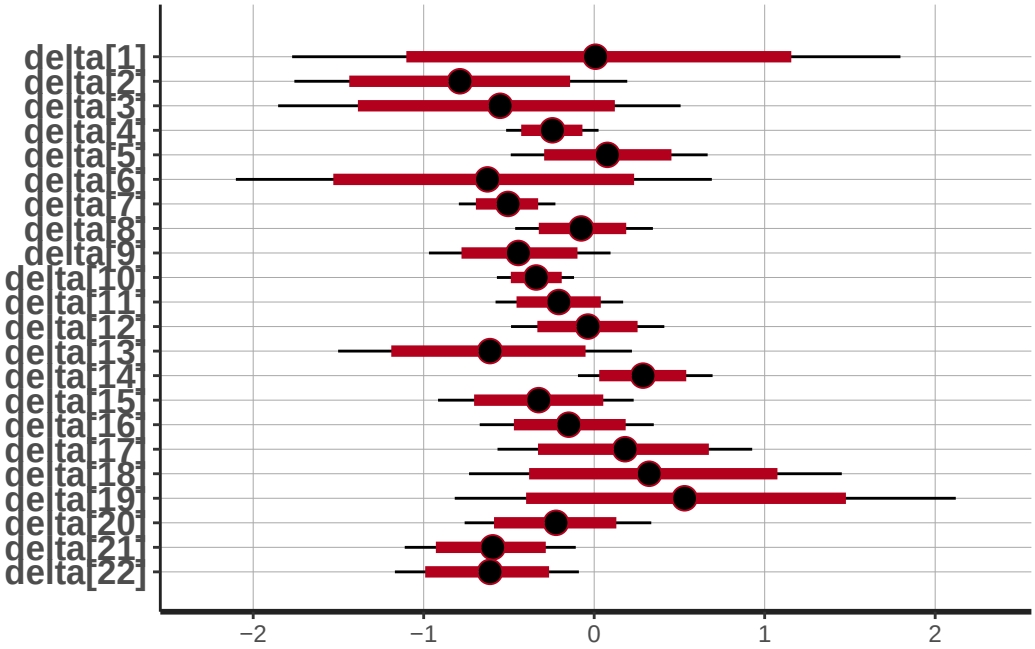


A continuación presentamos una gráfica para observar el rango de valores para cada δ_i

```
plot(stan_samples, pars = c("delta"))
```

```
ci_level: 0.8 (80% intervals)
```

```
outer_level: 0.95 (95% intervals)
```

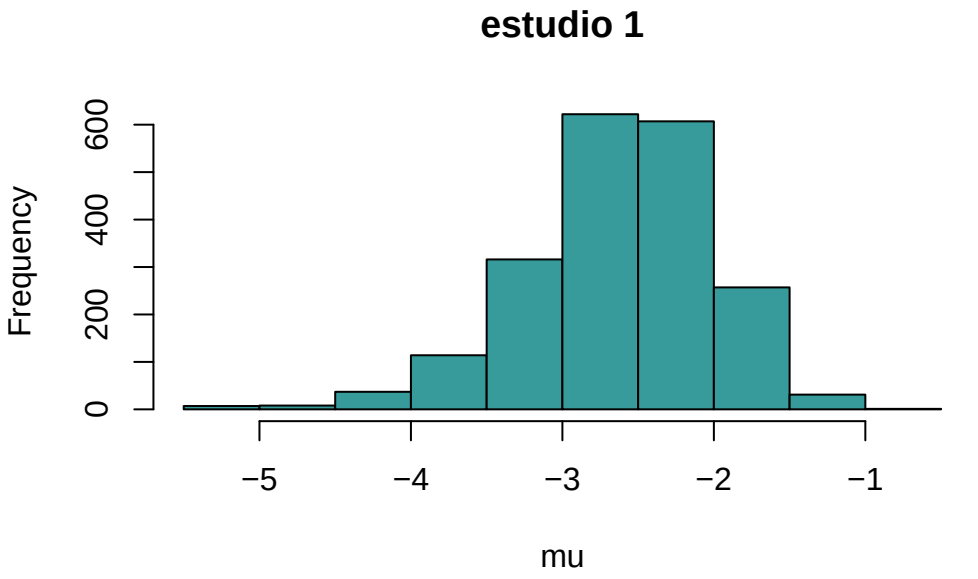


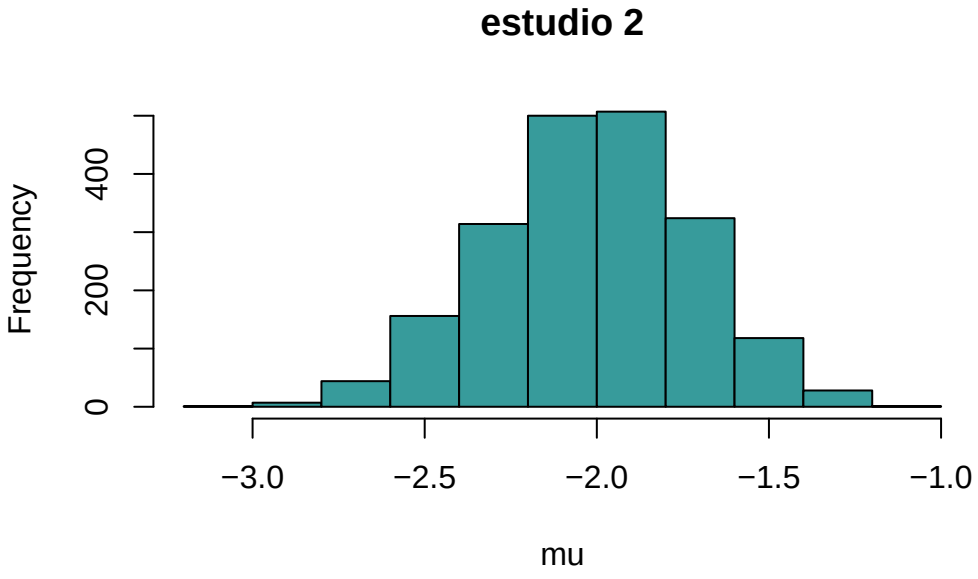
FInalmente, presentamos los histogramas para las distribuciones posteriores para cada μ_i y δ_i

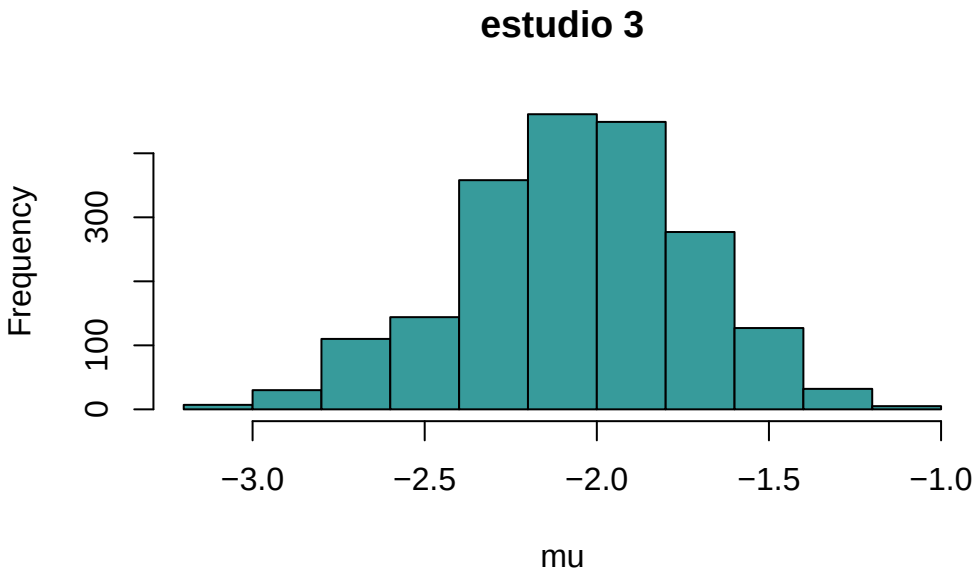
```
# Extraer muestras de la posterior para mu y delta
posterior_samples <- rstan::extract(stan_samples)

# Obtener muestras de la posterior para mu y delta
mu_samples <- posterior_samples$mu
delta_samples <- posterior_samples$delta
```

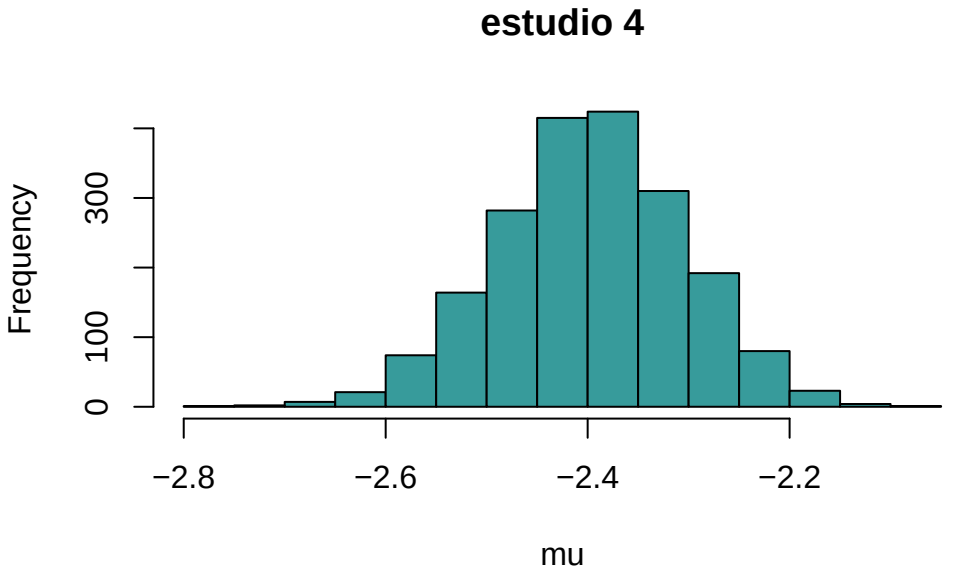
Histogramas de las posteriores de μ_i

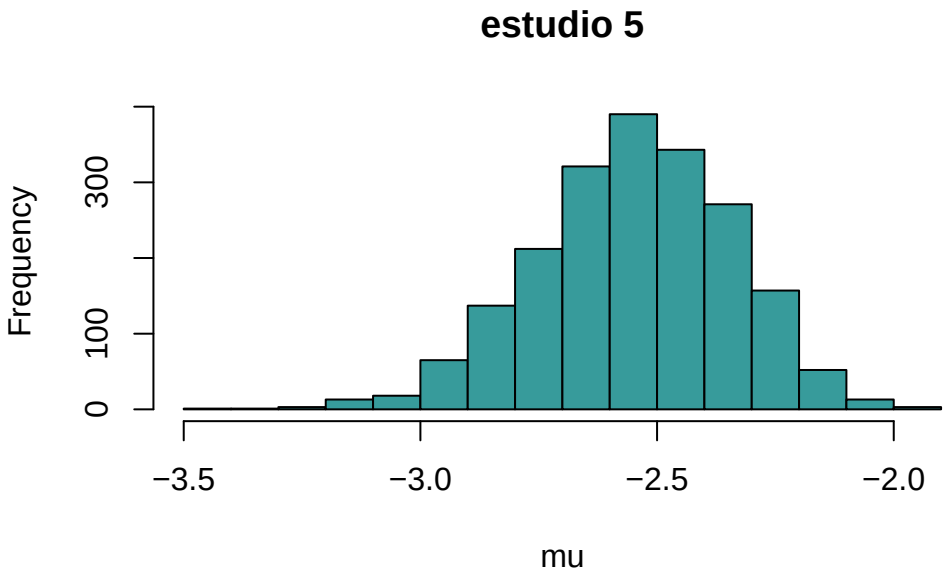
```
# Graficar las distribuciones posteriores de mu y delta para cada estudio

# par(mfrow = c(5, 5))

for (i in 1:22) {

  # Histograma de mu para el estudio i
  hist(mu_samples[, i], main = paste("estudio", i), xlab = "mu", col = "#379b9b")

}
```

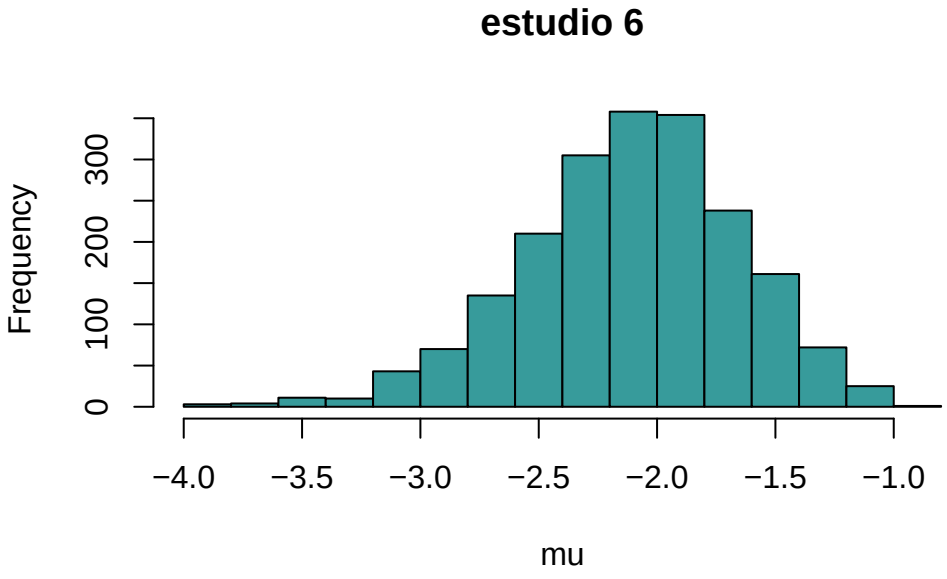

estudio 1
Frequency
0
200
400
600
−5
−4
−3
−2
−1
mu

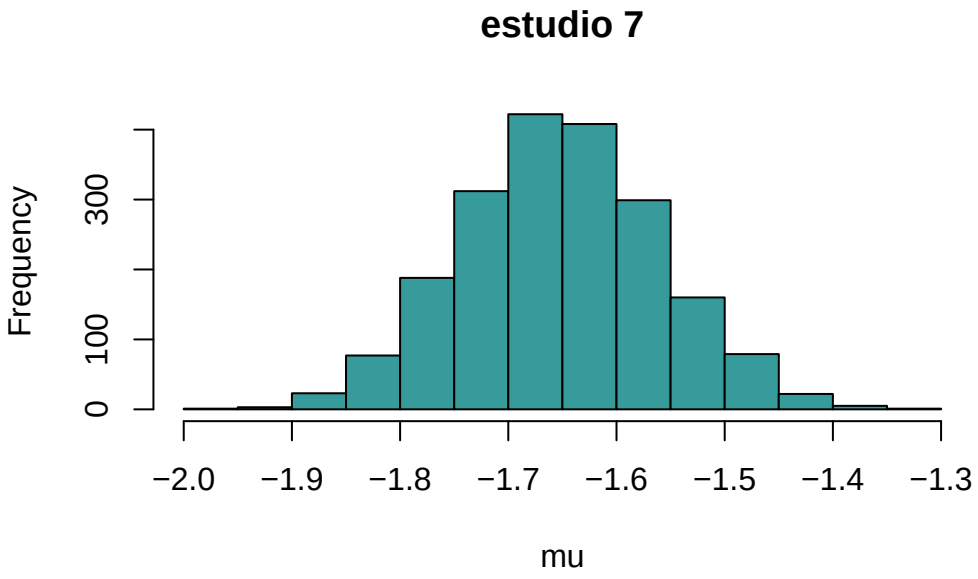

estudio 2
Frequency
0
200
400
−3.0
−2.5
−2.0
−1.5
−1.0
mu

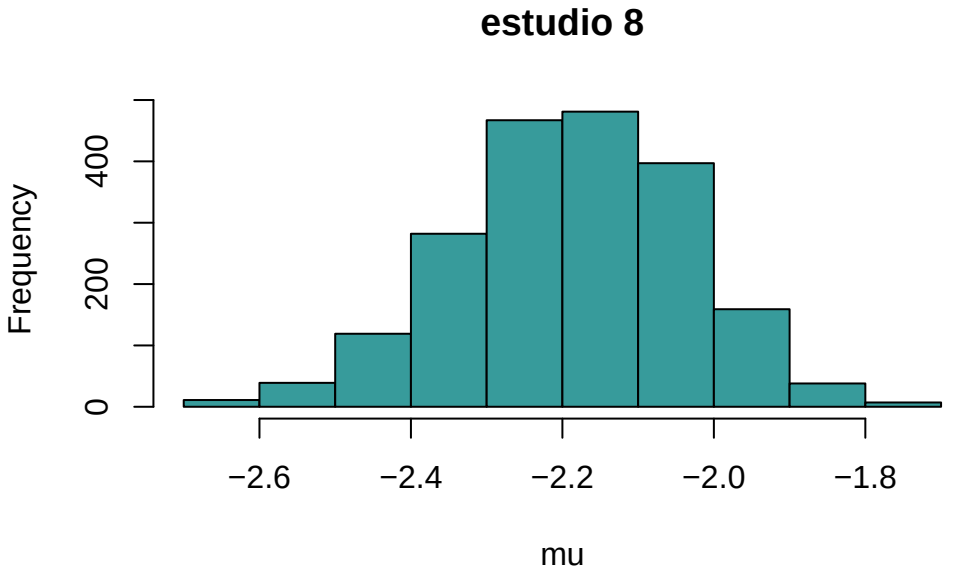

estudio 3
Frequency
0
100
300
−3.0
−2.5
−2.0
−1.5
−1.0
mu

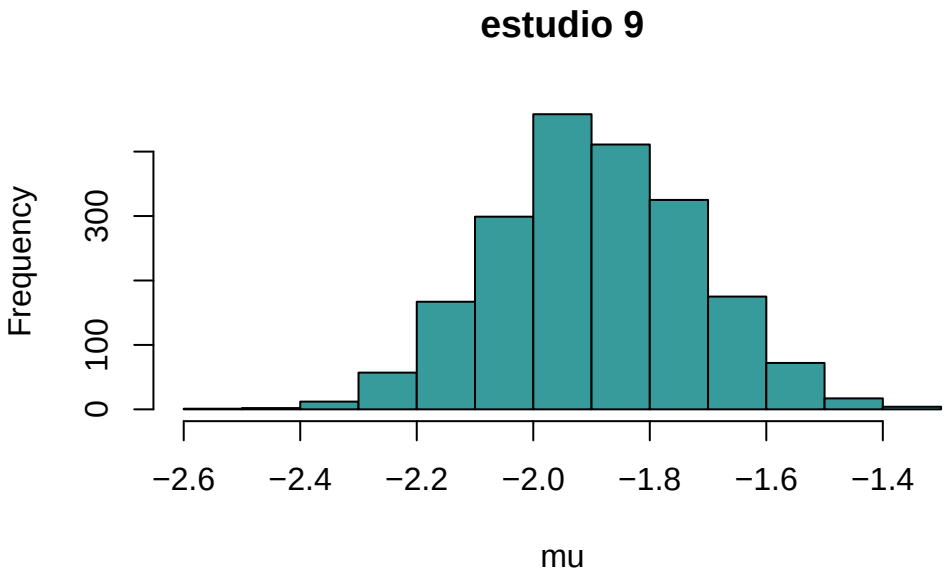

estudio 4
Frequency
0
100
300
−2.8
−2.6
−2.4
−2.2
mu

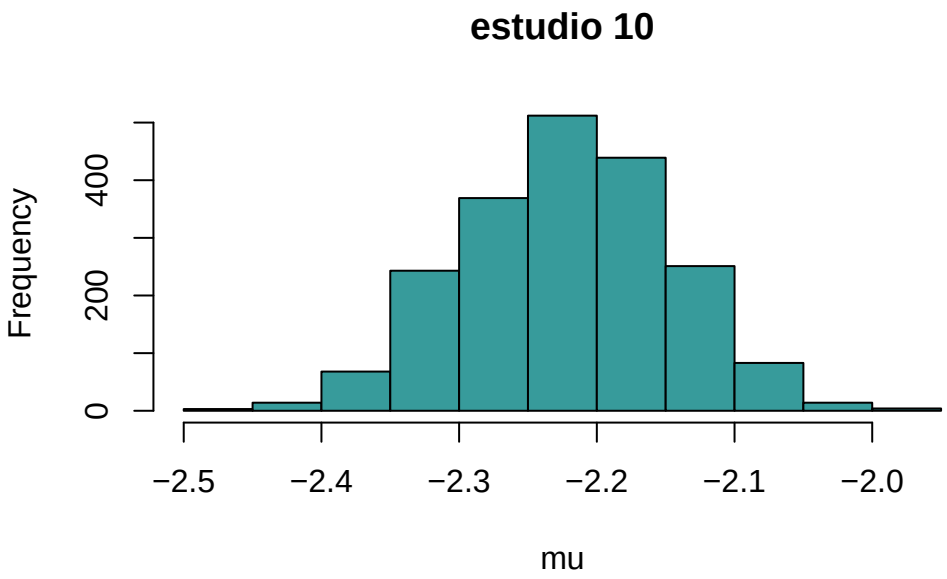

estudio 5
Frequency
0
100
300
−3.5
−3.0
−2.5
−2.0
mu

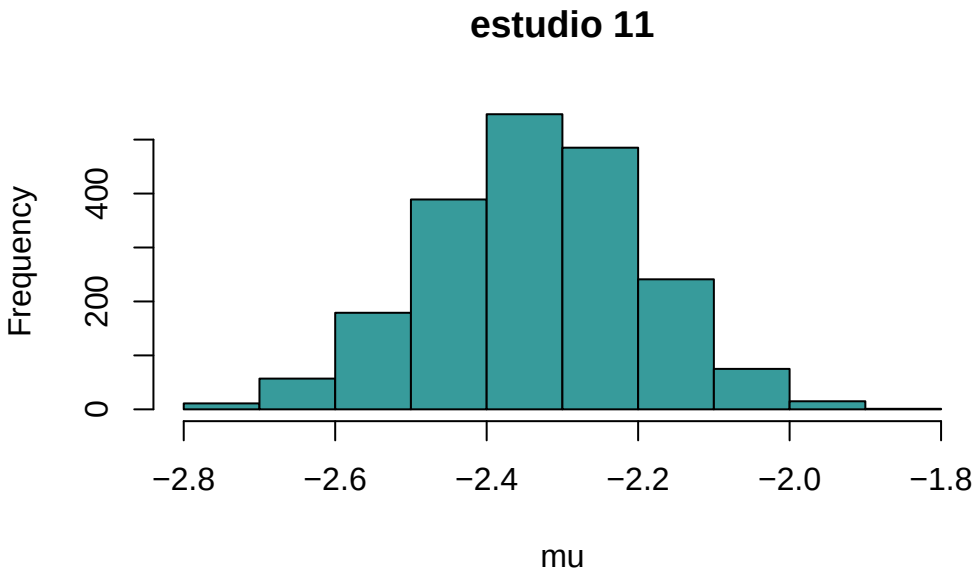

estudio 6
Frequency
0
100
200
300
−4.0
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
mu

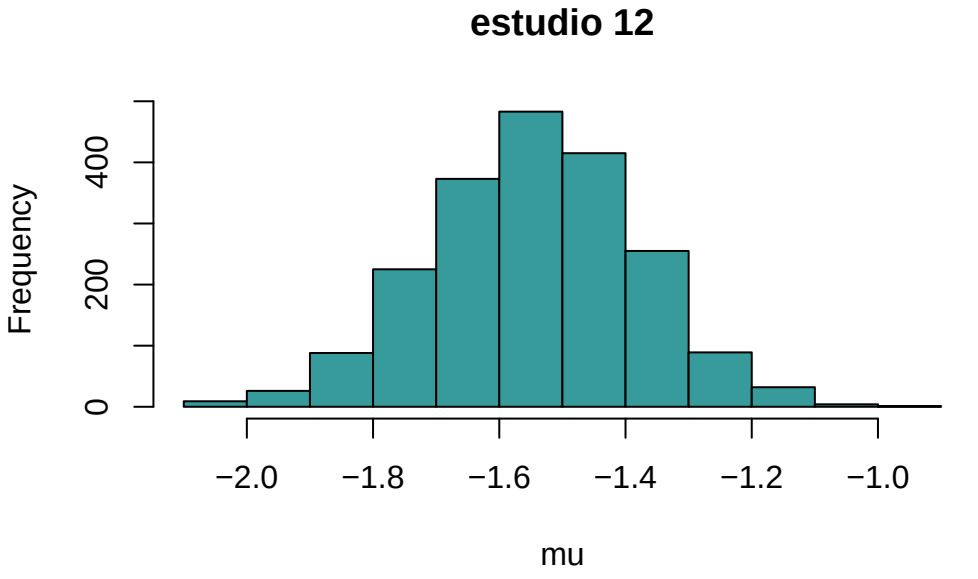

estudio 7
Frequency
0
100
300
−2.0
−1.9
−1.8
−1.7
−1.6
−1.5
−1.4
−1.3
mu

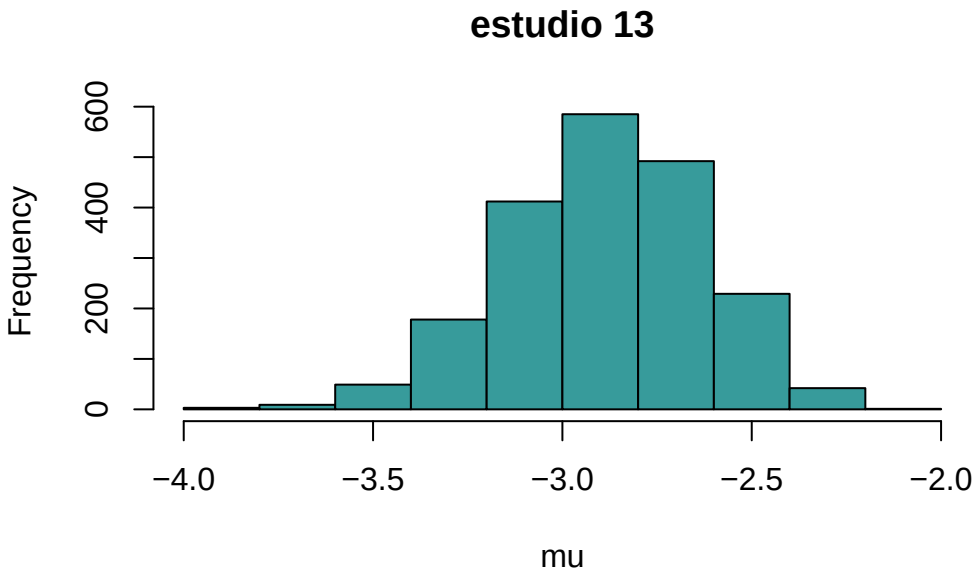

estudio 8
Frequency
0
200
400
−2.6
−2.4
−2.2
−2.0
−1.8
mu

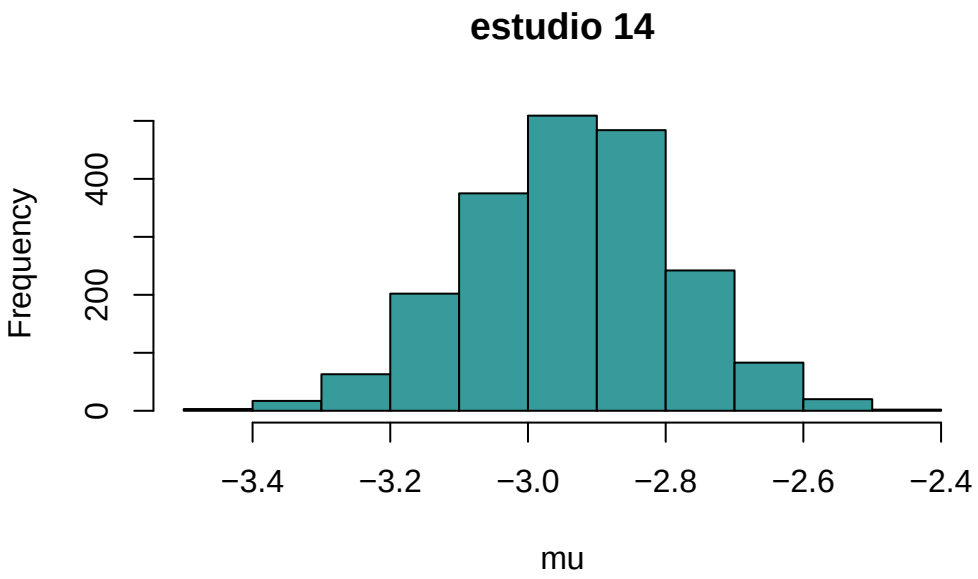

estudio 9
Frequency
0
100
300
−2.6
−2.4
−2.2
−2.0
−1.8
−1.6
−1.4
mu

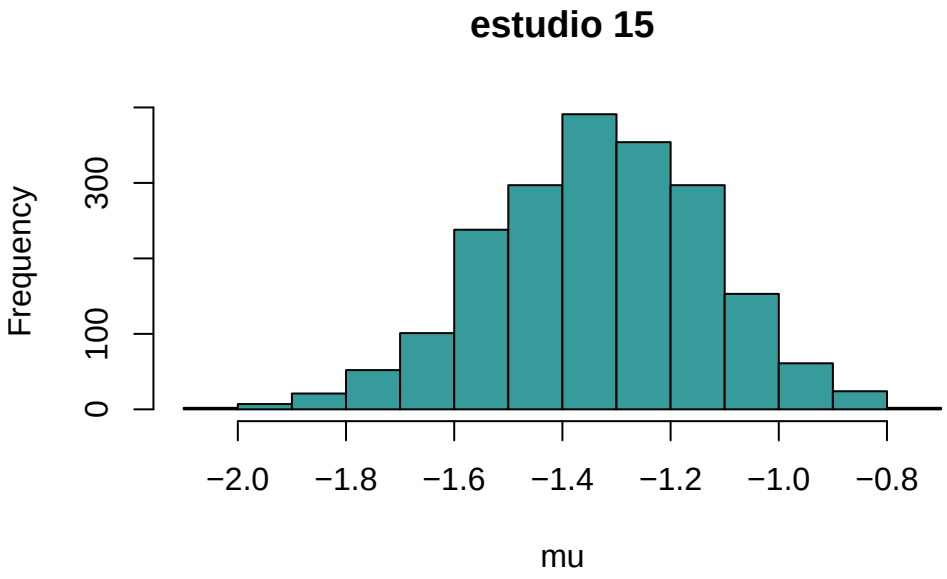

estudio 10
Frequency
0
200
400
−2.5
−2.4
−2.3
−2.2
−2.1
−2.0
mu

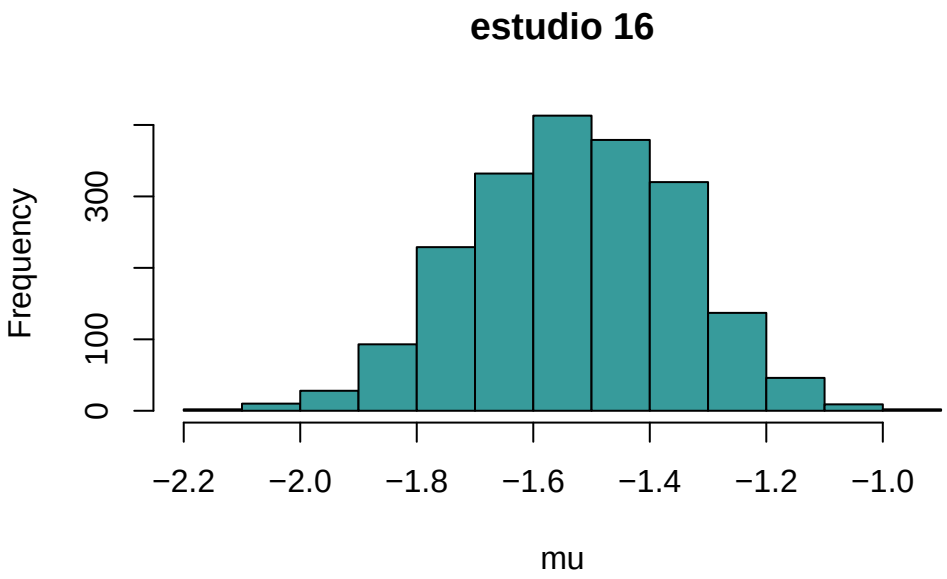

estudio 11
Frequency
0
200
400
−2.8
−2.6
−2.4
−2.2
−2.0
−1.8
mu

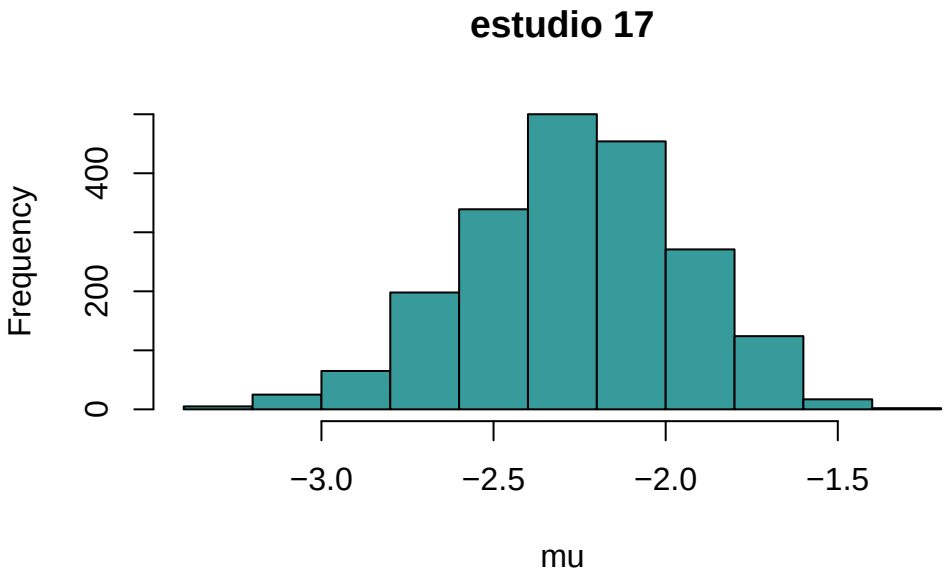

estudio 12
Frequency
0
200
400
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

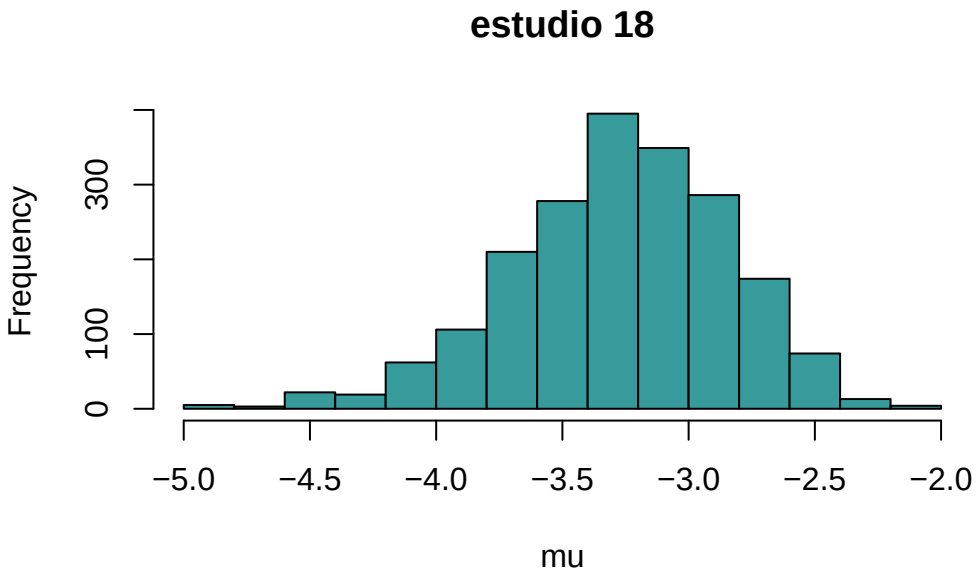

estudio 13
Frequency
0
200
400
600
−4.0
−3.5
−3.0
−2.5
−2.0
mu

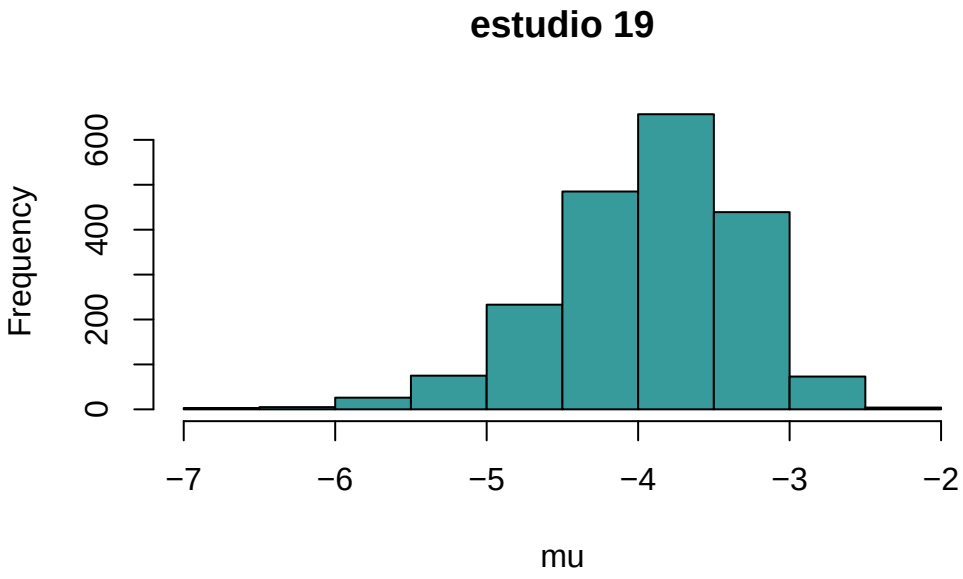

estudio 14
Frequency
0
200
400
−3.4
−3.2
−3.0
−2.8
−2.6
−2.4
mu

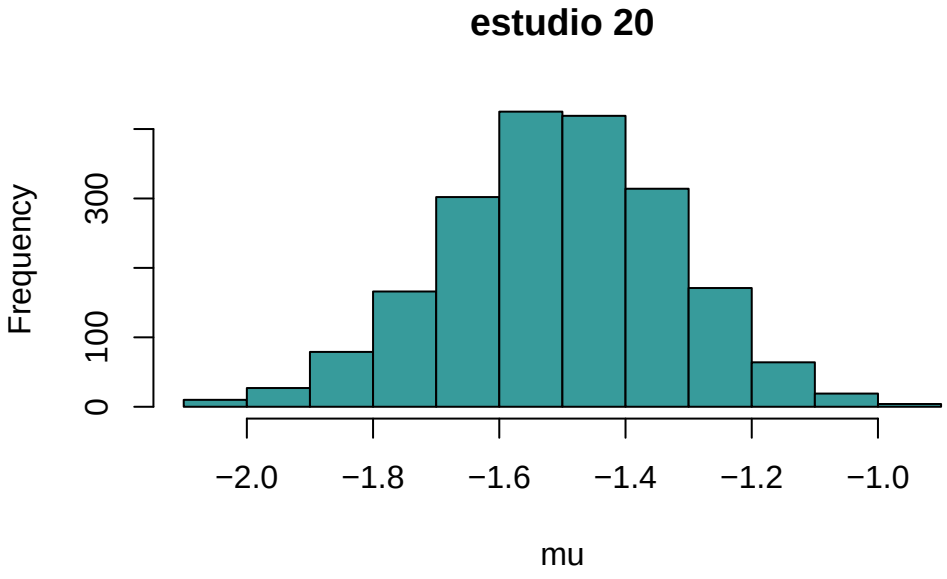

estudio 15
Frequency
0
100
300
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
mu

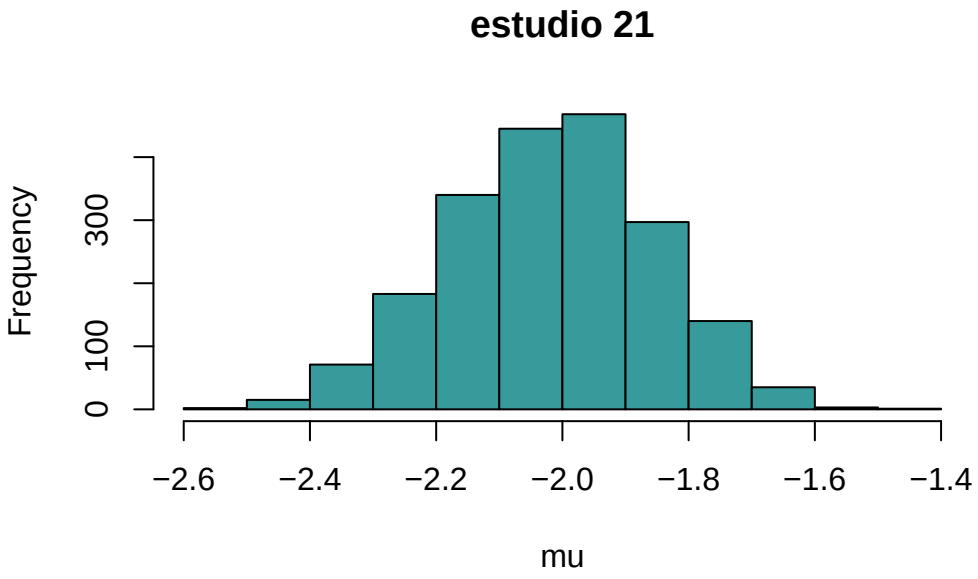

estudio 16
Frequency
0
100
300
−2.2
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

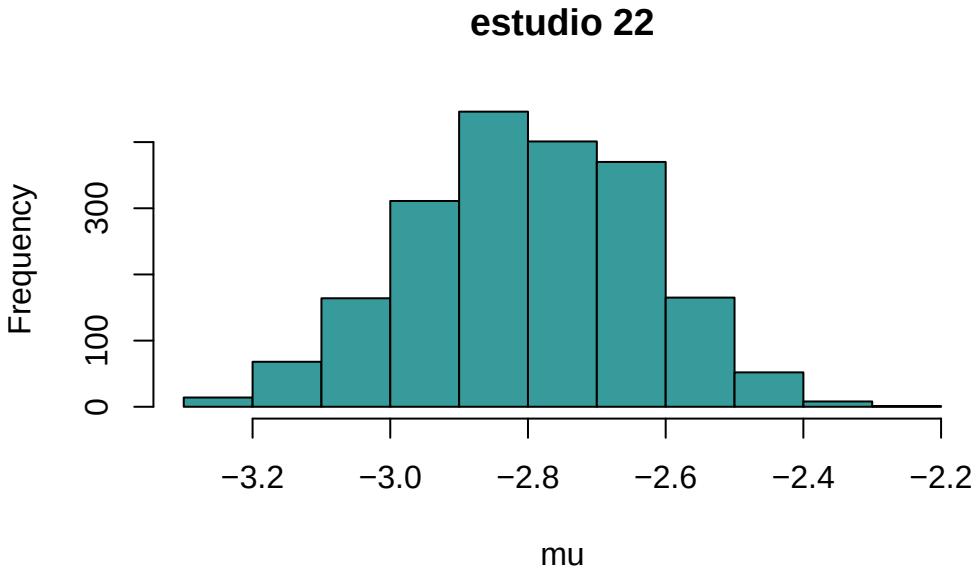

estudio 17
Frequency
0
200
400
−3.0
−2.5
−2.0
−1.5
mu


estudio 18
Frequency
0
100
300
−5.0
−4.5
−4.0
−3.5
−3.0
−2.5
−2.0
mu


estudio 19
Frequency
0
200
400
600
−7
−6
−5
−4
−3
−2
mu

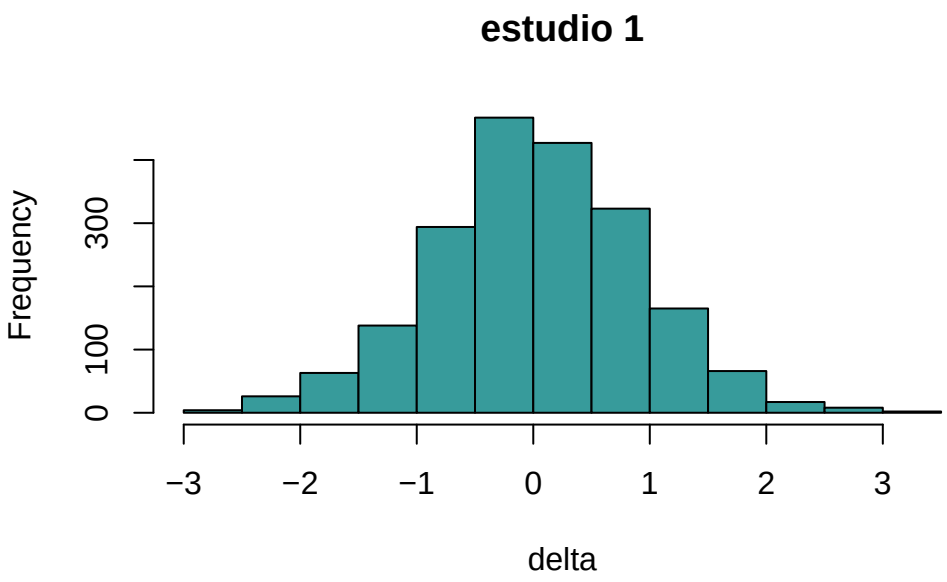

estudio 20
Frequency
0
100
300
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

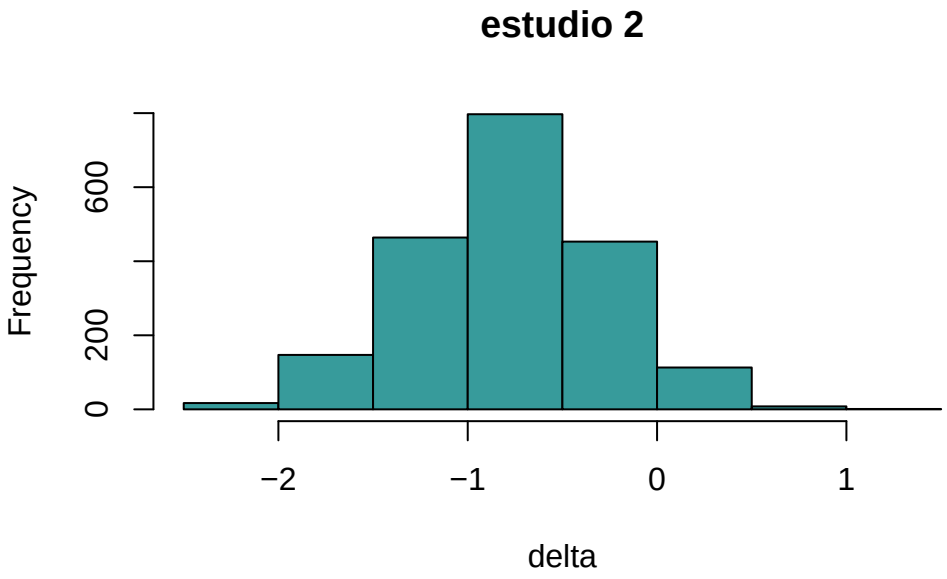


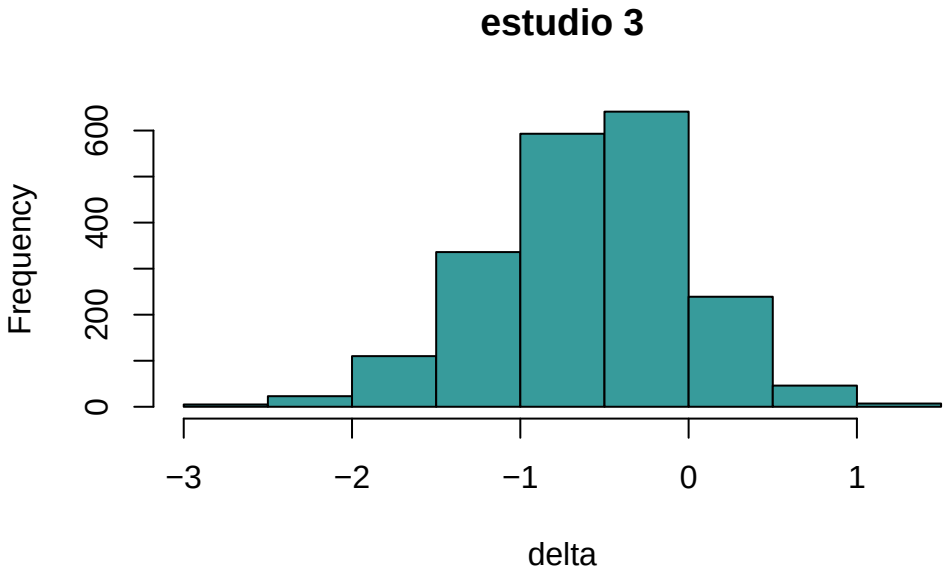


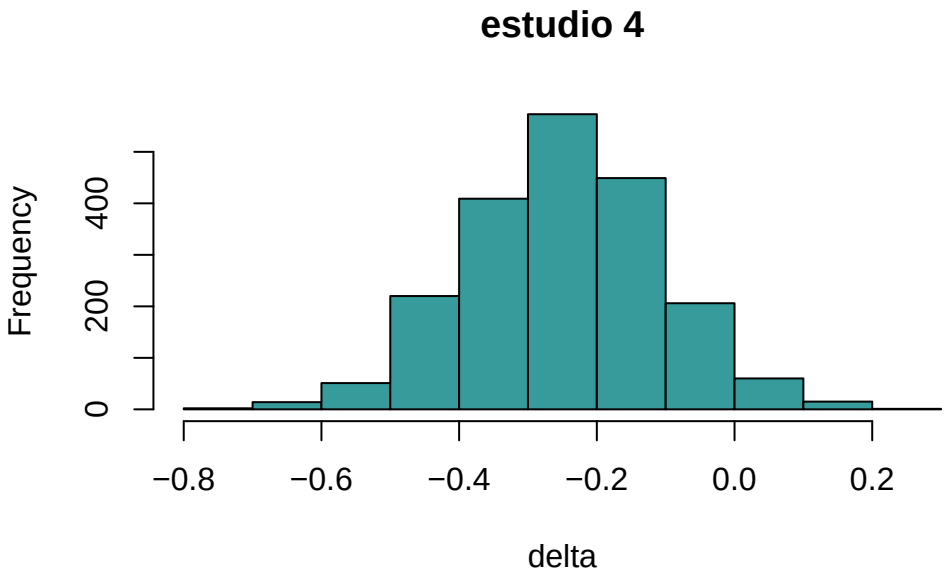
Histogramas de las posteriores de δ_i

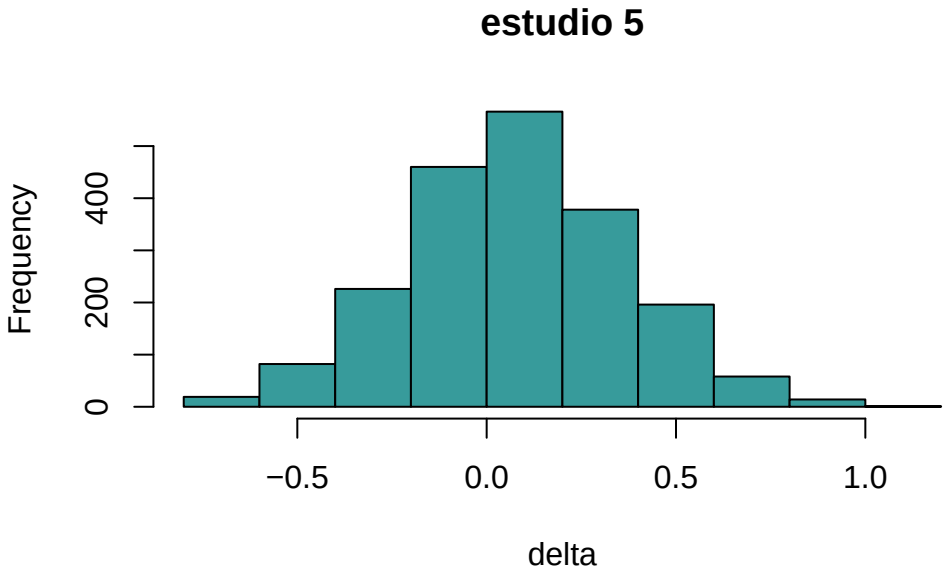
```r
# par(mfrow = c(5, 5))

for (i in 1:22) {

  # Gráfico de densidad kernel de delta para el estudio i
  hist(delta_samples[, i], main = paste("estudio", i), xlab = "delta", col = "#379b9b")

}
```

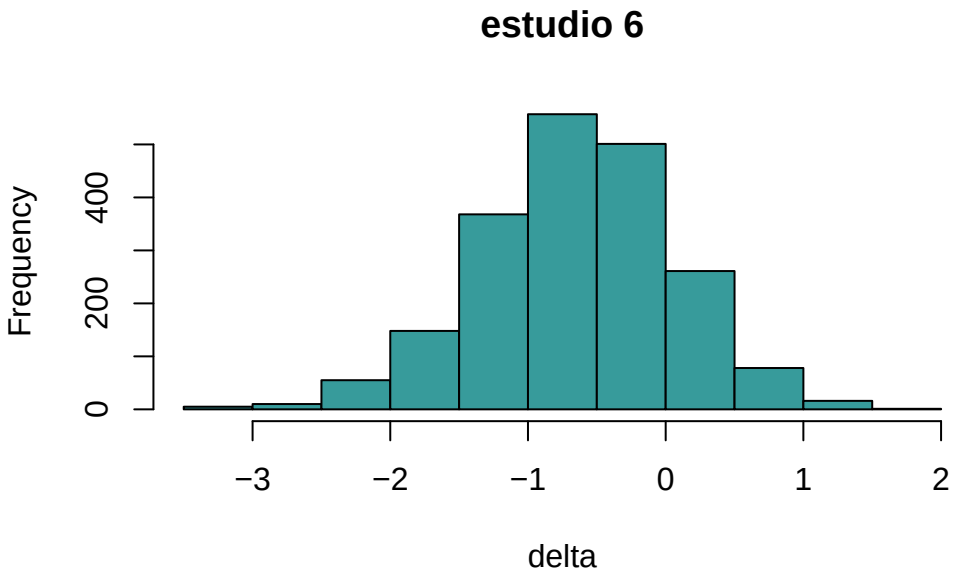


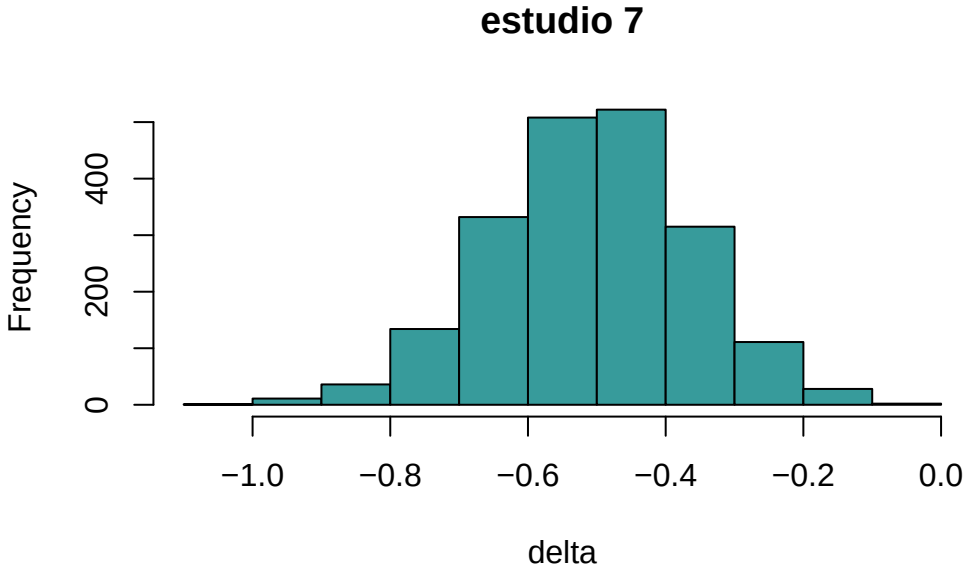

estudio 2
Frequency
0
200
600
−2
−1
0
1
delta

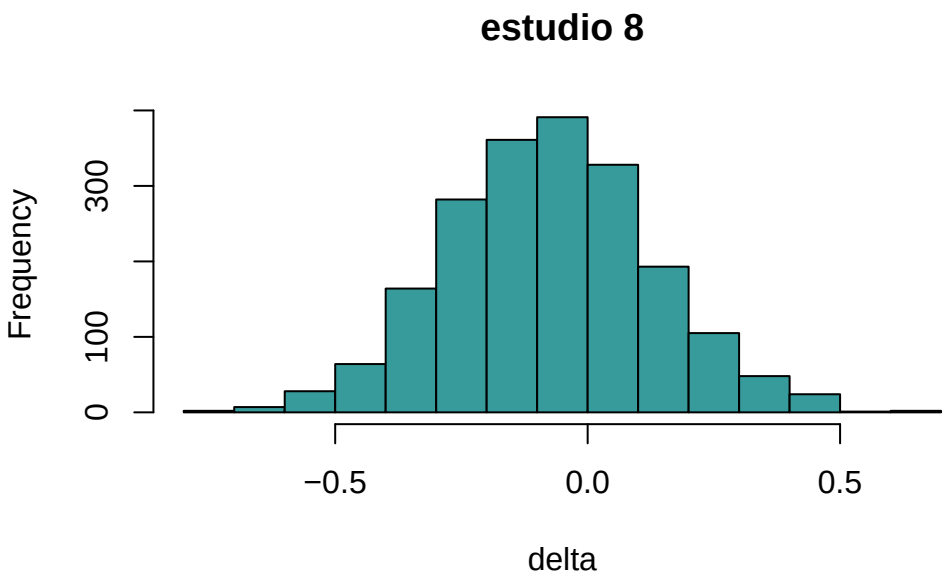

estudio 3
Frequency
0
200
400
600
−3
−2
−1
0
1
delta

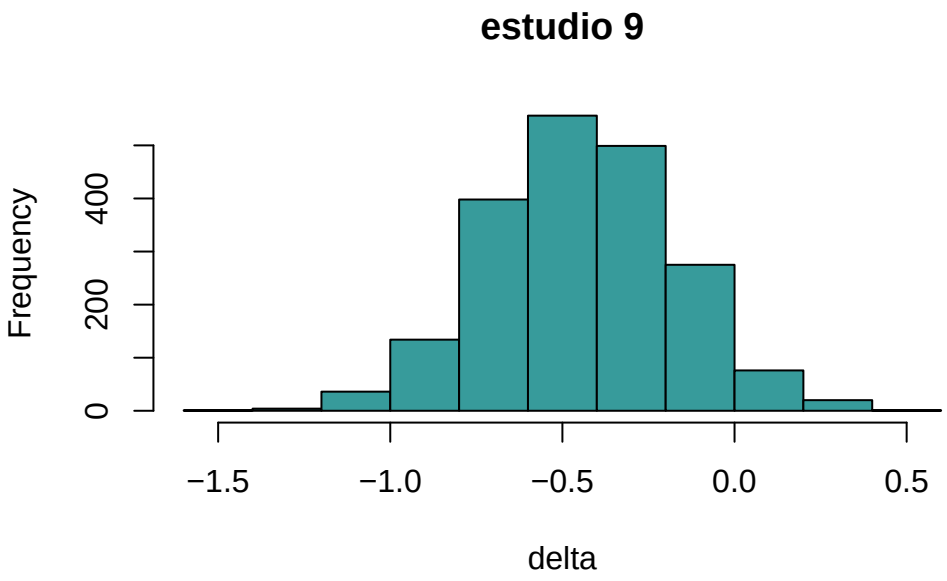

estudio 4
Frequency
0
200
400
−0.8
−0.6
−0.4
−0.2
0.0
0.2
delta

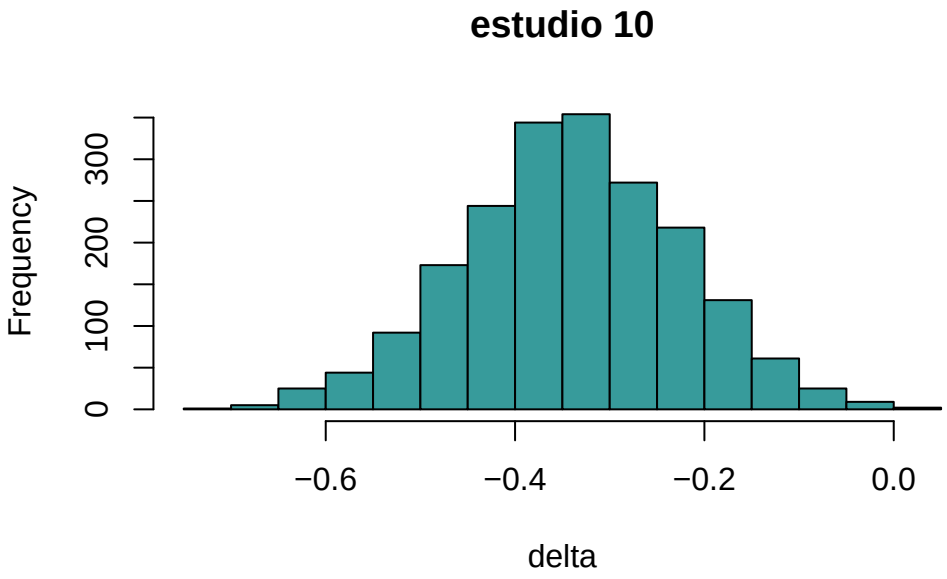

estudio 5
Frequency
0
200
400
−0.5
0.0
0.5
1.0
delta

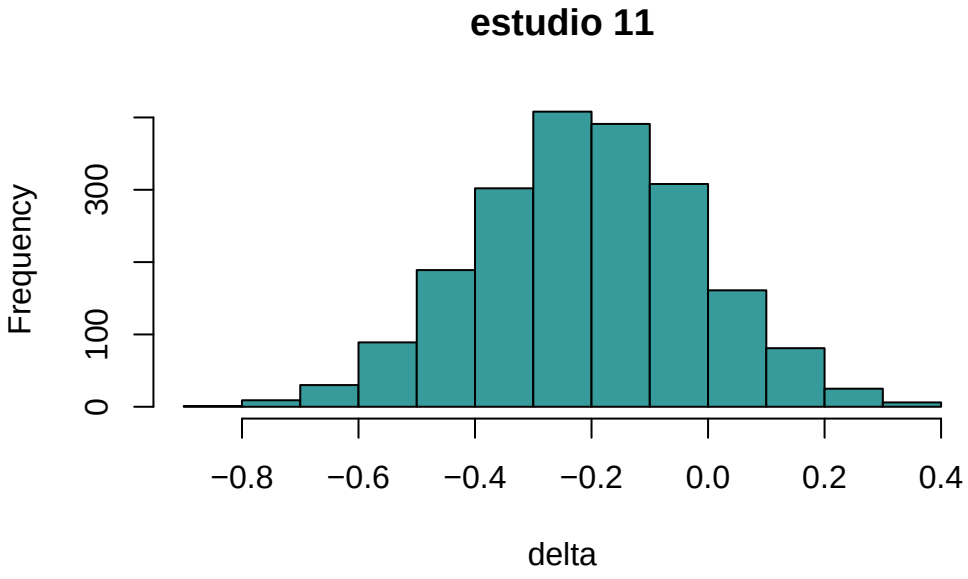

estudio 6
Frequency
0
200
400
−3
−2
−1
0
1
2
delta

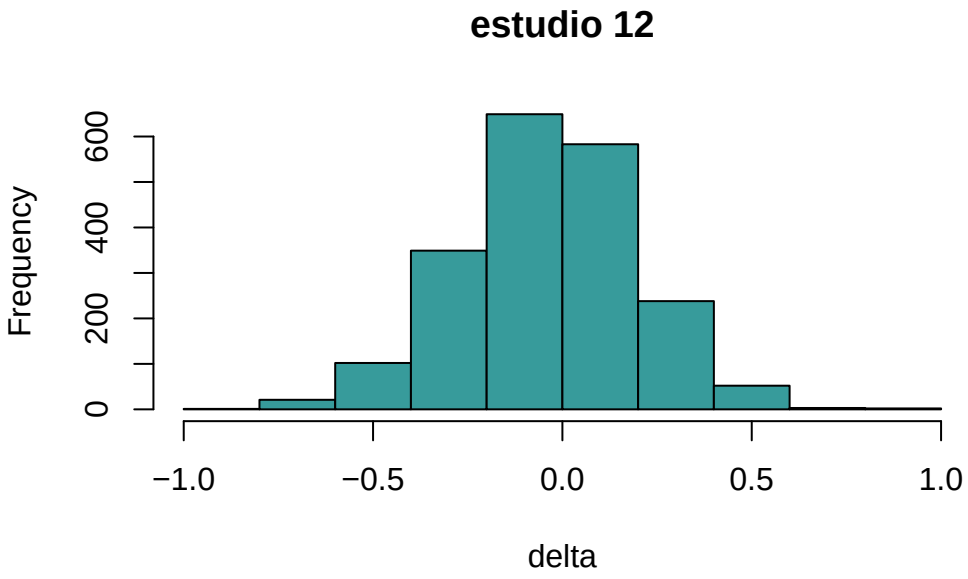

estudio 7
Frequency
0
200
400
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
delta

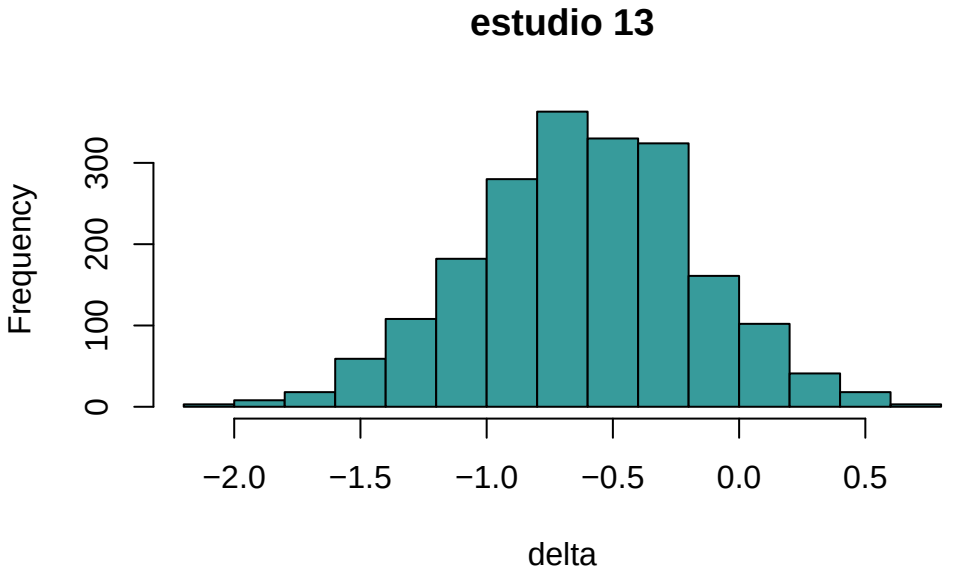

estudio 8
Frequency
0
100
300
−0.5
0.0
0.5
delta

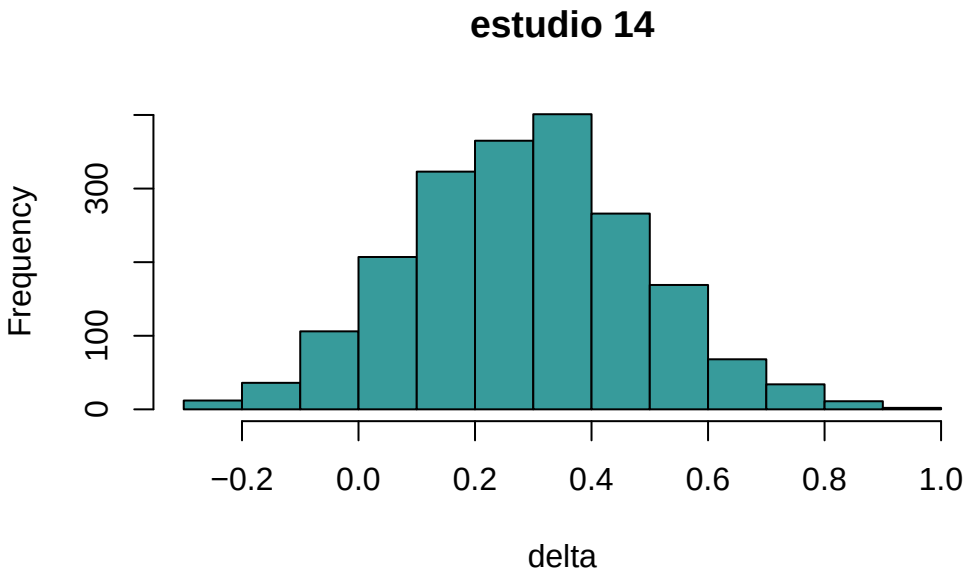

estudio 9
Frequency
0
200
400
−1.5
−1.0
−0.5
0.0
0.5
delta

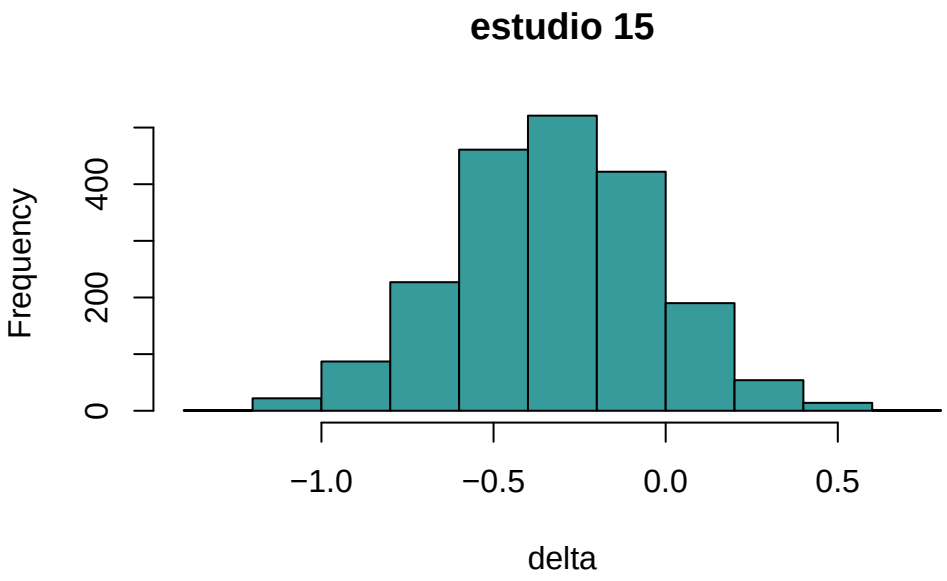

estudio 10
Frequency
0
100
200
300
−0.6
−0.4
−0.2
0.0
delta

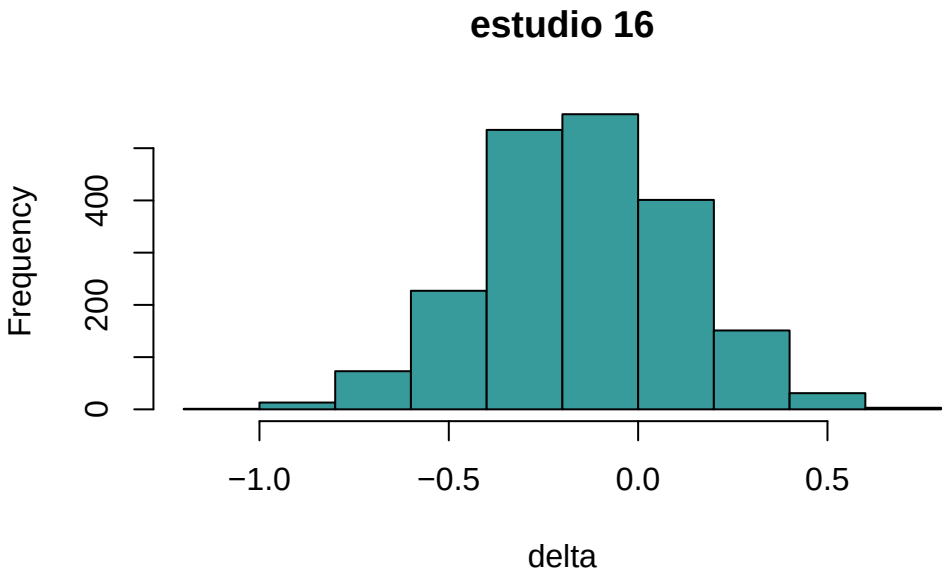

estudio 11
Frequency
0
100
300
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
delta

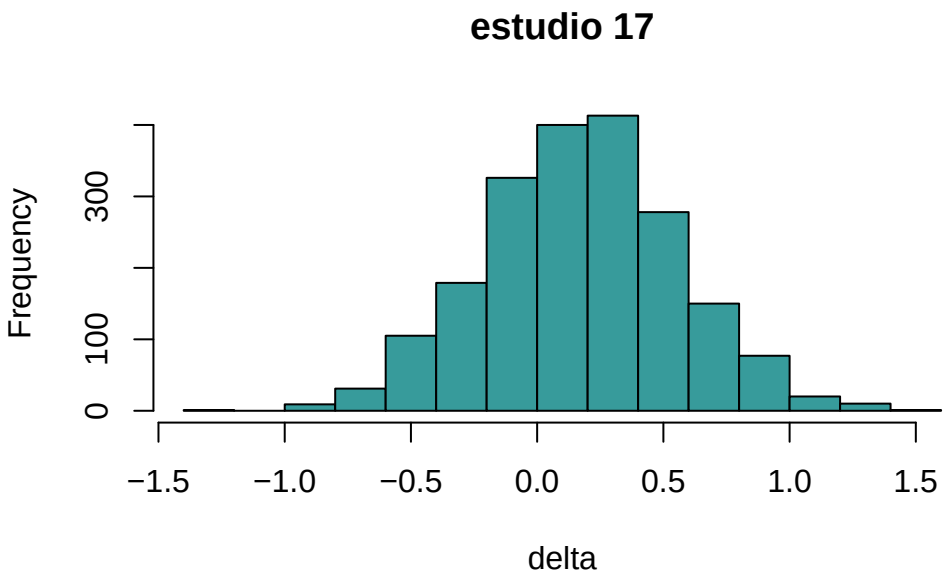

estudio 12
Frequency
0
200
400
600
−1.0
−0.5
0.0
0.5
1.0
delta

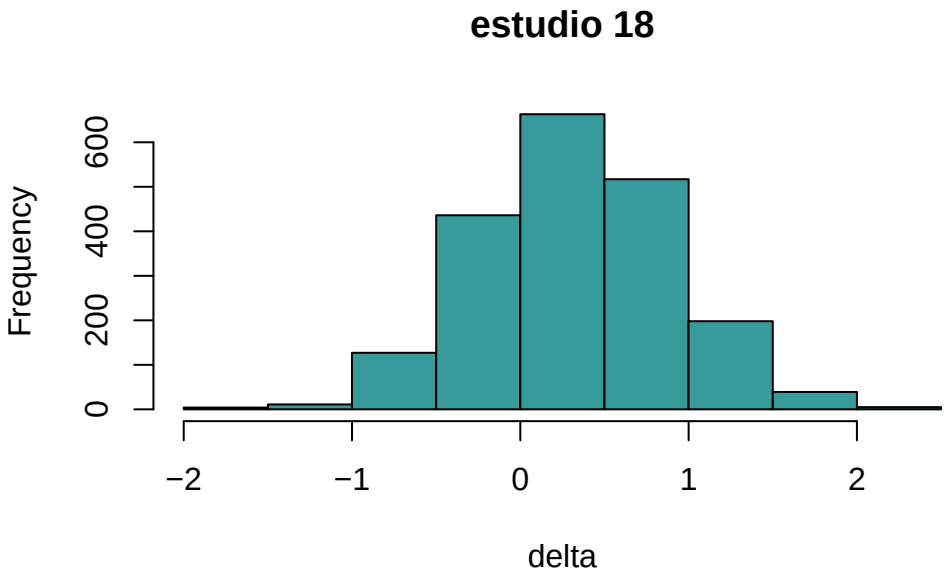

estudio 13
Frequency
0
100
200
300
−2.0
−1.5
−1.0
−0.5
0.0
0.5
delta

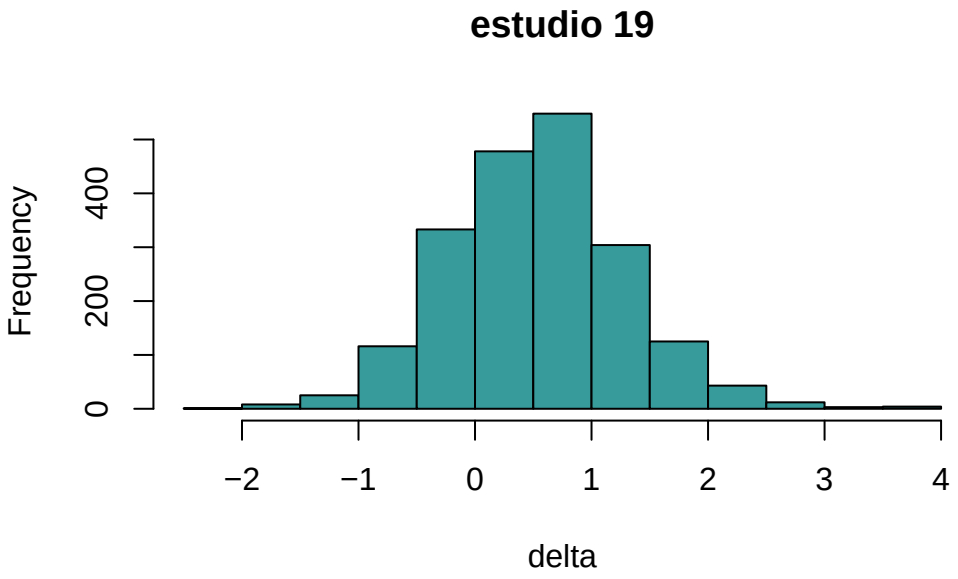

estudio 14
Frequency
0
100
300
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
delta

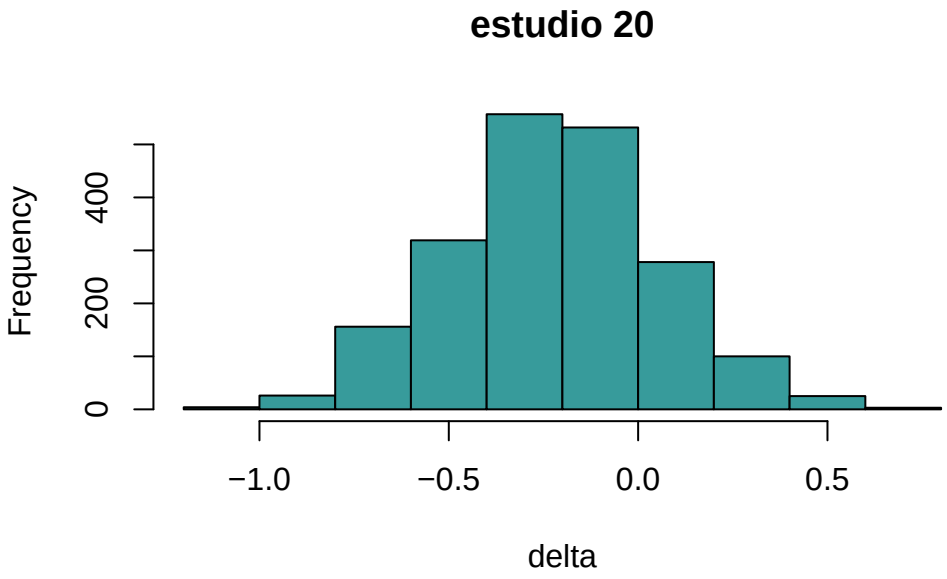

estudio 15
Frequency
0
200
400
−1.0
−0.5
0.0
0.5
delta

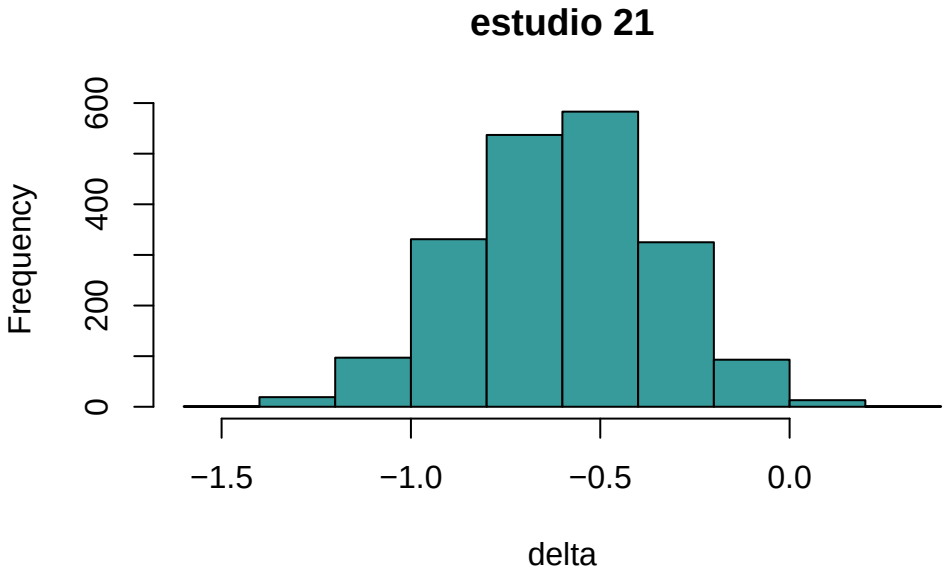

estudio 16
Frequency
0
200
400
−1.0
−0.5
0.0
0.5
delta

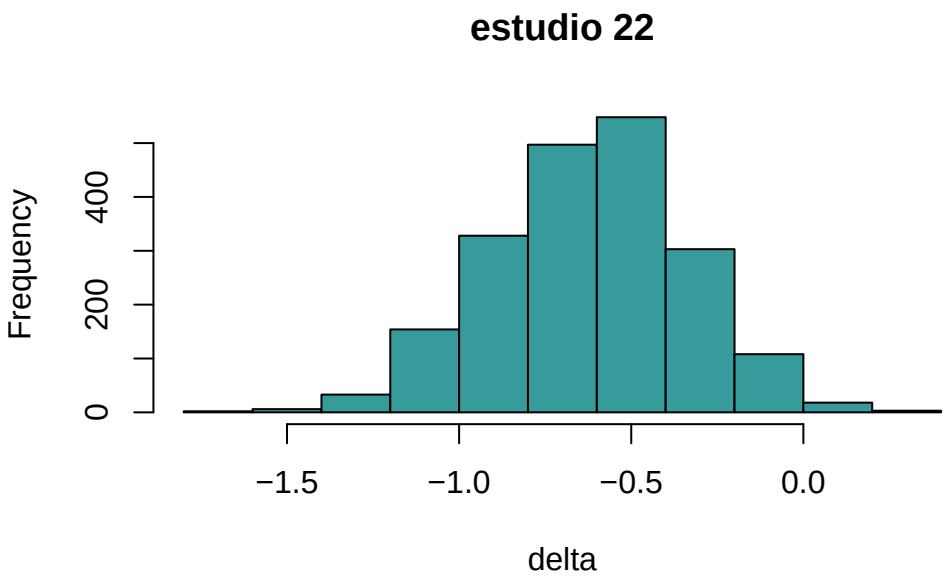

estudio 17
Frequency
0
100
300
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
delta


estudio 18
Frequency
0
200
400
600
−2
−1
0
1
2
delta


estudio 19
Frequency
0
200
400
−2
−1
0
1
2
3
4
delta


estudio 20
Frequency
0
200
400
−1.0
−0.5
0.0
0.5
delta


estudio 21
Frequency
0
200
400
600
−1.5
−1.0
−0.5
0.0
delta



Conclusiones:

- Variabilidad entre estudios: Como podemos observar, hay una amplia variabilidad en las distribuciones posteriores de δ_i, esto podría indicar que el efecto de los beta-bloqueadores varía según el contexto o las características de cada estudio.
- Efecto del tratamiento: La media de las δ_i se encuentran en el rango entre -0.77 y 0.53, sin embargo existe una clara tendencia a ser negativas, lo que indica que los beta-bloqueadores tienen un efecto beneficioso significativo en la reducción del riesgo de infarto en la mayoría de los estudios.

b. modelo jerárquico

Un marco alternativo es un modelo jerárquico donde se supone que hay una distribución común para todos los ensayos tal que $\delta_i \sim N(d, \sigma^2)$. Suponiendo las siguientes distribuciones iniciales de estos parámetros estimar este modelo: $d \sim N(0, 10)$, $\sigma^2 \sim Cauchy(0, 2.5)$

Ajustaremos el modelo para obtener un modelo jerárquico que supone que todos los efectos del tratamiento $ _i$ se distribuyen normalmente con una media común d y una desviación estándar común $ $

```
# Modelo en Stan
stan_code <- '
data {
  int<lower=0> J;           // Número de estudios
  int rt[J];                // Número de muertes en tratamiento
  int nt[J];                // Tamaño de la muestra en tratamiento
  int rc[J];                // Número de muertes en control
  int nc[J];                // Tamaño de la muestra en control
}

parameters {
  real delta[J];            // Efecto del tratamiento
  real mu[J];                  // Media común de los efectos del tratamiento
  real<lower=0> sigma_sq;      // Desviación estándar común de los efectos del tratamiento
  real d;
}

model {
  d ~ normal(0, 10);           // Priori para la media común
  sigma_sq ~ cauchy(0, 2.5);   // Priori para la sd común de los efectos del tratamiento
  delta ~ normal(d, sigma_sq);  // Modelo jerárquico para los efectos del tratamiento

  for (j in 1:J) {
    mu[j] ~ normal(0, 10);       // Priori para la media común de los  efectos del tratami
    rc[j] ~ binomial_logit(nc[j], mu[j]);                          // Verosimilitud del c
    rt[j] ~ binomial_logit(nt[j], mu[j] + delta[j]);               // Verosimilitud del t
  }
}
'


# Compilar el modelo
stan_model <- stan_model(model_code = stan_code)
```

```
Trying to compile a simple C file

Running /usr/lib/R/bin/R CMD SHLIB foo.c
using C compiler: ‘gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0’
gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-
In file included from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
```

```
                 from /home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-pc-linux-gnu/StanHea
                 from <command-line>:
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
  628 | namespace Eigen {
      | ^~~~~~~~~
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
  628 | namespace Eigen {
      |                 ^
In file included from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-pc-linux-gnu/StanHea
                 from <command-line>:
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
   96 | #include <complex>
      |          ^~~~~~~~~
compilation terminated.
make: *** [/usr/lib/R/etc/Makeconf:191: foo.o] Error 1
```

```r
# Datos para Stan
stan_data <- list(
  J = nrow(data), # Número de estudios
  rt = data$rt,
  nt = data$nt,
  rc = data$rc,
  nc = data$nc,
  N = data$N
)


# Muestreo de la distribución posterior
stan_samples <- sampling(stan_model, data = stan_data, iter = 10000, chains = 4)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
```

```
Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 1.494 seconds (Warm-up)
Chain 1:                1.544 seconds (Sampling)
Chain 1:                3.038 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.3e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 1.368 seconds (Warm-up)
Chain 2:                1.131 seconds (Sampling)
Chain 2:                2.499 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
```

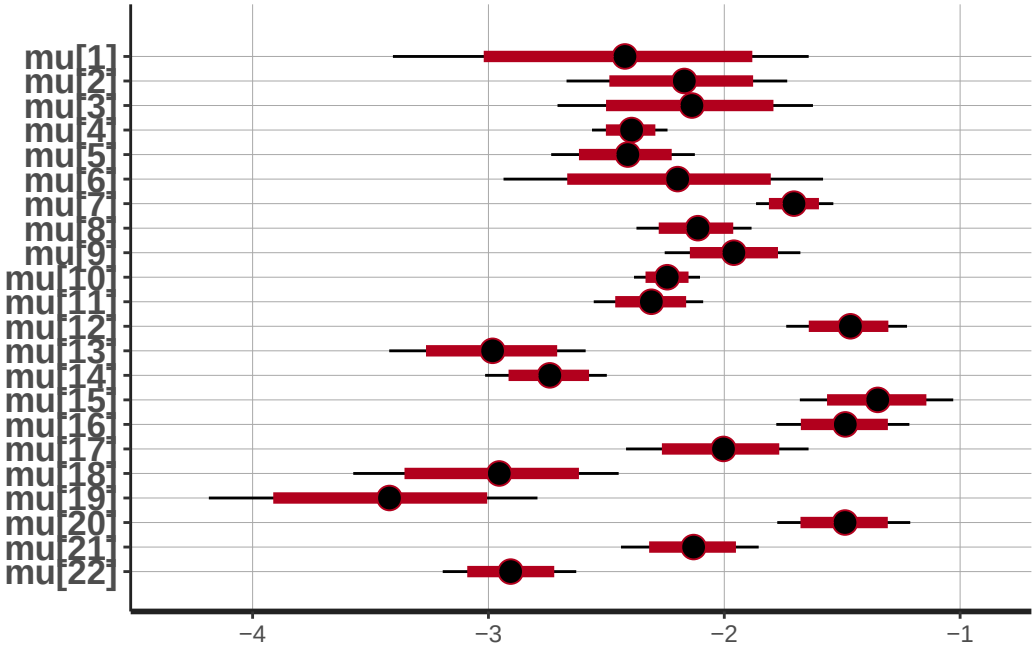
```
Chain 3: Gradient evaluation took 1.3e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 3: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 3: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 3: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 3: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 1.393 seconds (Warm-up)
Chain 3:                2.077 seconds (Sampling)
Chain 3:                3.47 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.3e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 4:
```

```
Chain 4:  Elapsed Time: 1.441 seconds (Warm-up)
Chain 4:                1.093 seconds (Sampling)
Chain 4:                2.534 seconds (Total)
Chain 4:

Warning: There were 488 divergent transitions after warmup. See
https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
to find out why this is a problem and how to eliminate them.

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

Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and ta
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
```

A continuación presentamos una gráfica para observar el rango de valores para μ

```r
plot(stan_samples, pars = c("mu"))
```

```
ci_level: 0.8 (80% intervals)

outer_level: 0.95 (95% intervals)
```

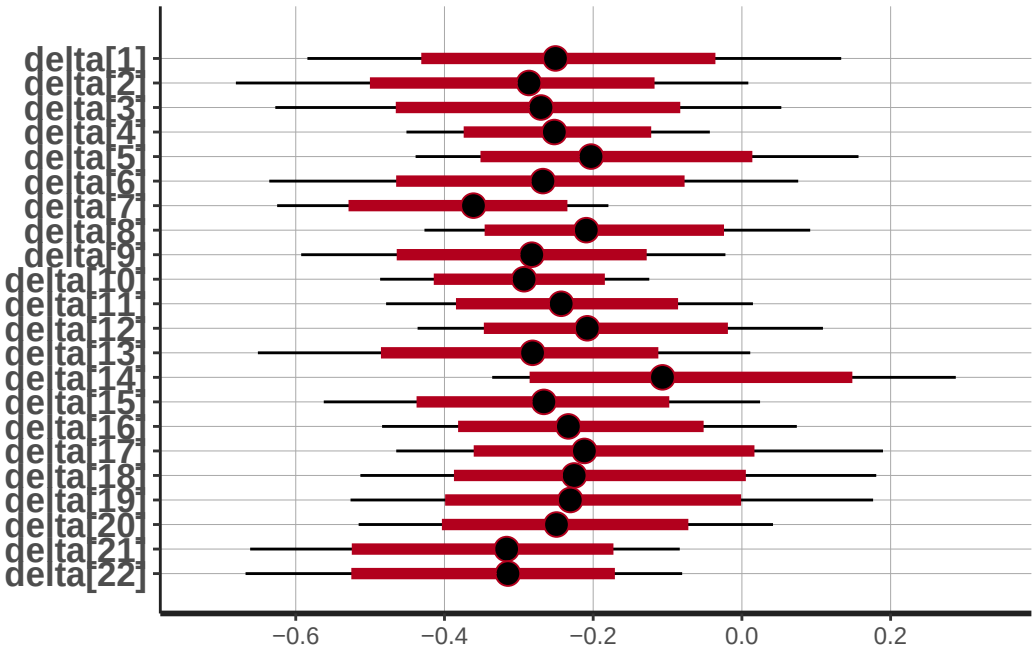


A continuación presentamos una gráfica para observar el rango de valores para cada δ

```
plot(stan_samples, pars = c("delta"))
```

```
ci_level: 0.8 (80% intervals)
```

```
outer_level: 0.95 (95% intervals)
```

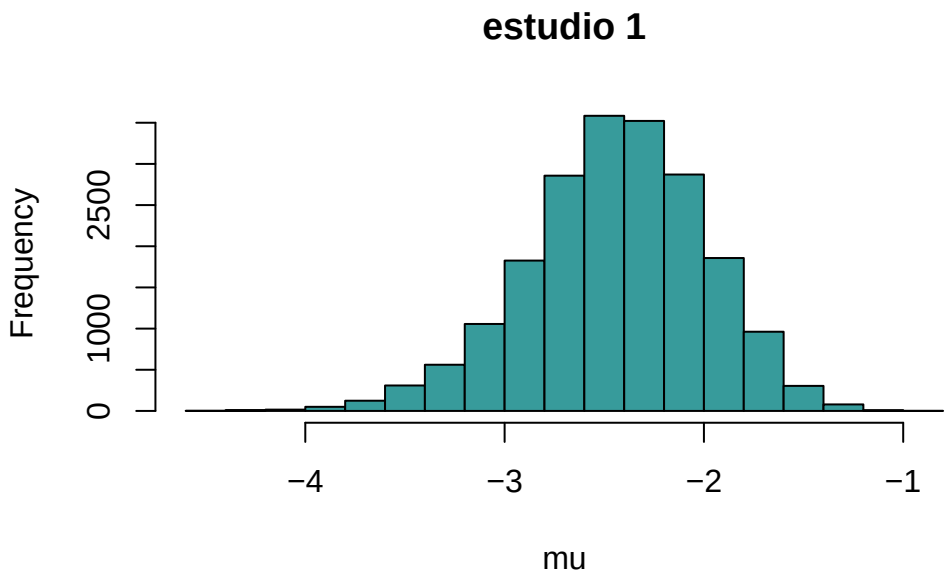


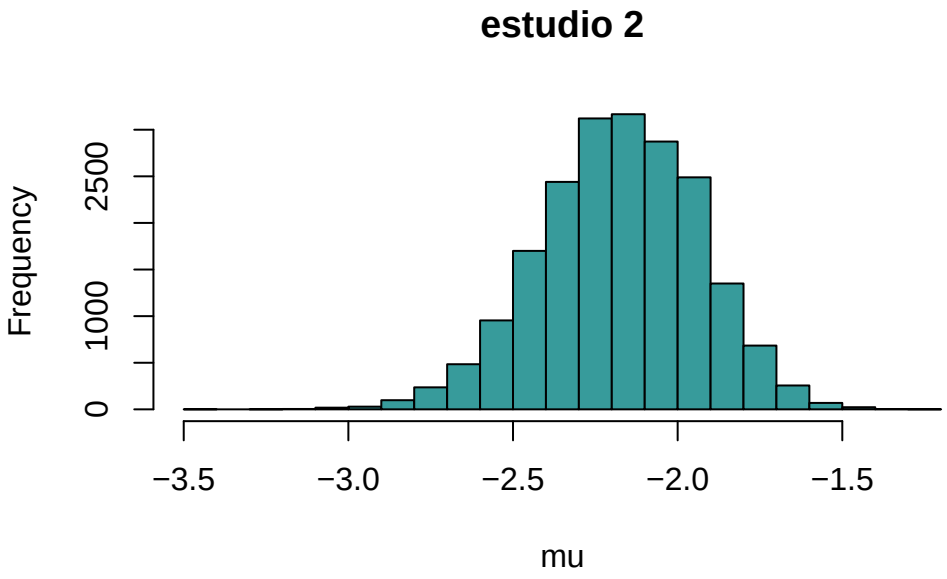
FInalmente, presentamos los histogramas para las distribuciones posteriores para cada μ_i y δ_i

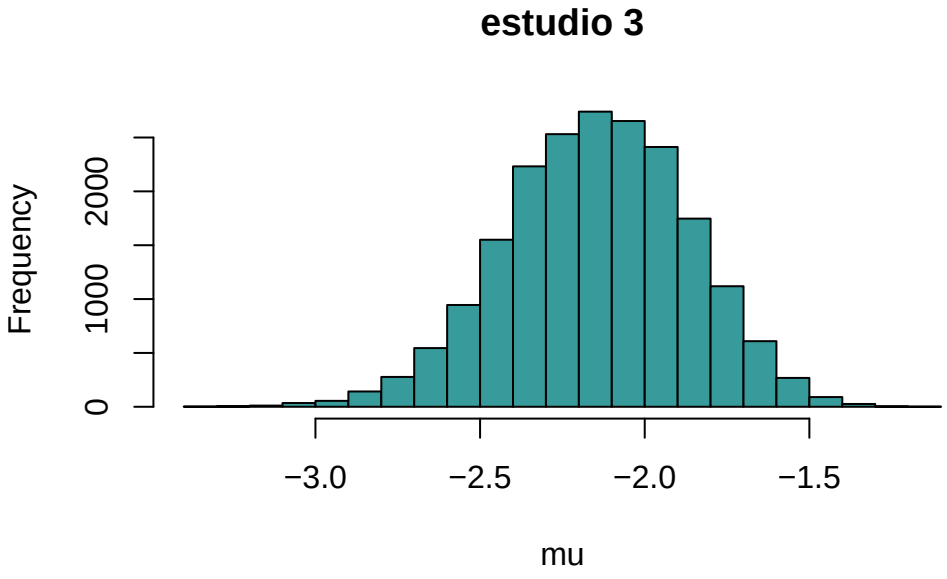
```
# Extraer muestras de la posterior para mu y delta
posterior_samples <- rstan::extract(stan_samples)

# Obtener muestras de la posterior para mu y delta
mu_samples <- posterior_samples$mu
delta_samples <- posterior_samples$delta
```

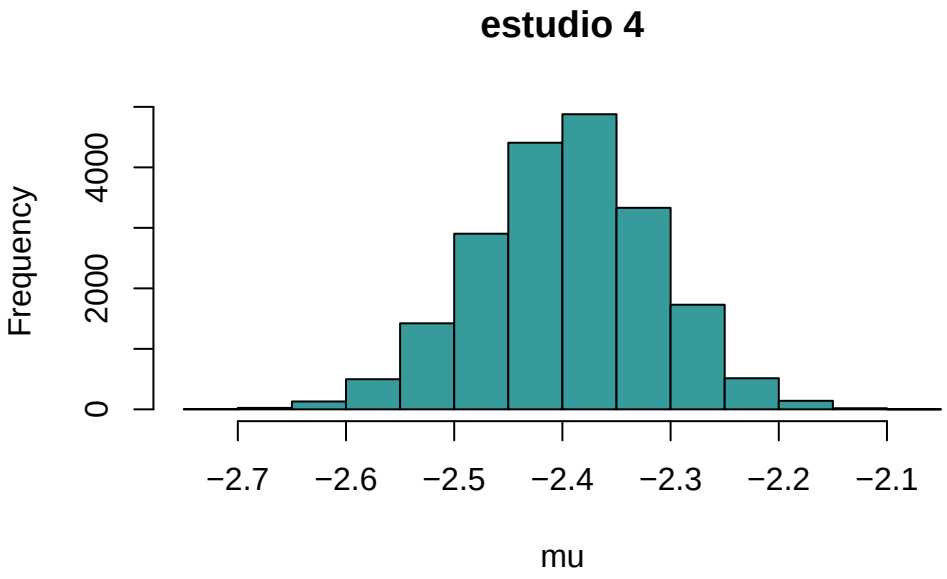
Histogramas de las posteriores de μ_i

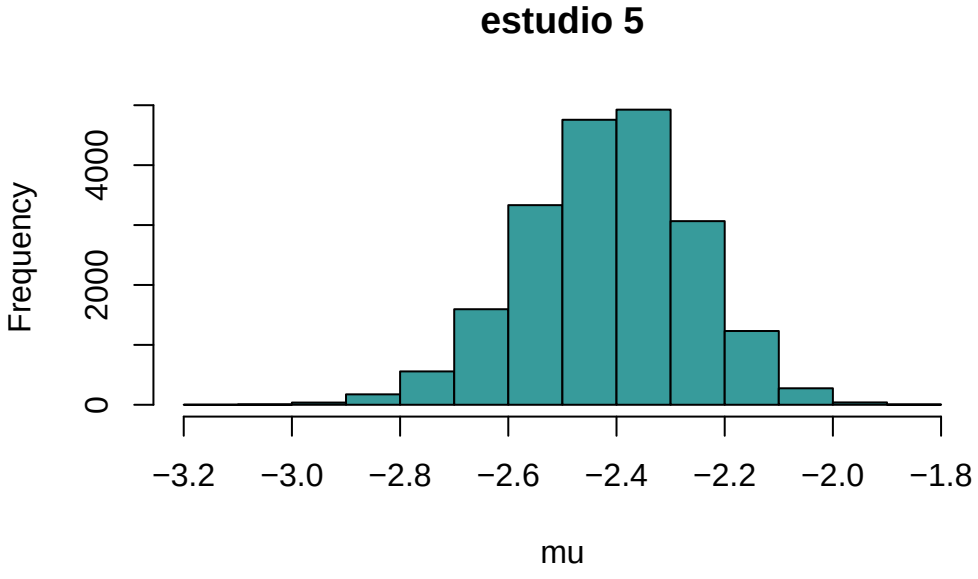
```r
# Graficar las distribuciones posteriores de mu y delta para cada estudio

# par(mfrow = c(5, 5))

for (i in 1:22) {

  # Histograma de mu para el estudio i
  hist(mu_samples[, i], main = paste("estudio", i), xlab = "mu", col = "#379b9b")

}
```

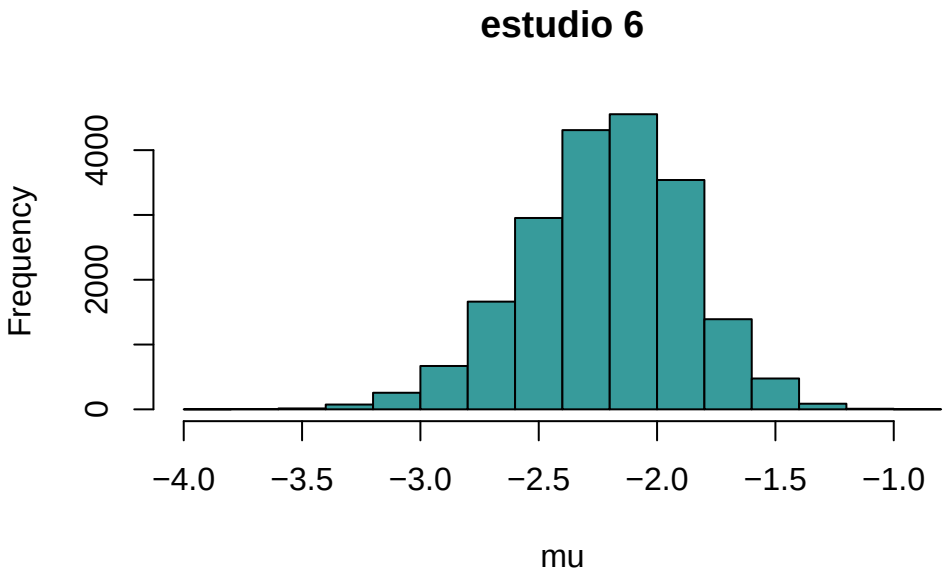


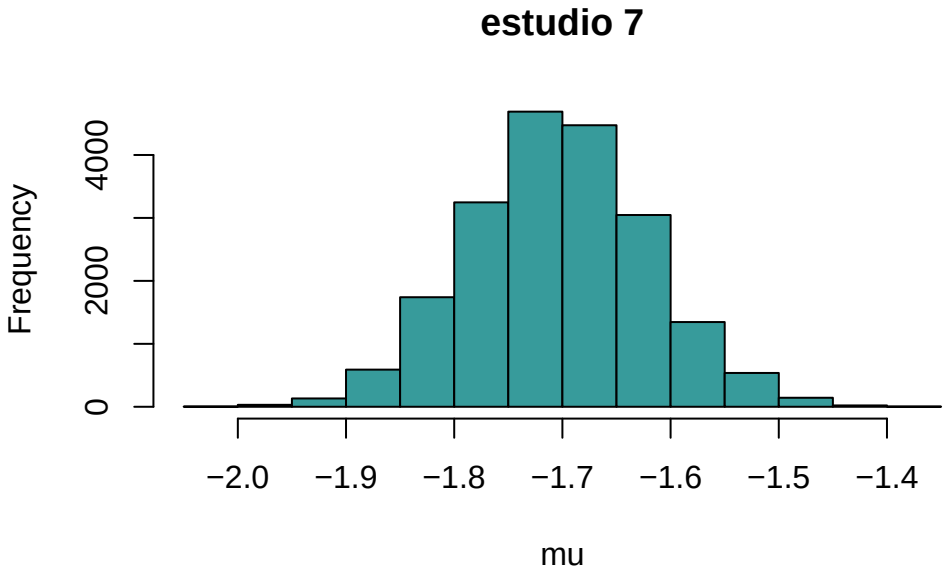

estudio 2
Frequency
0
1000
2500
−3.5
−3.0
−2.5
−2.0
−1.5
mu

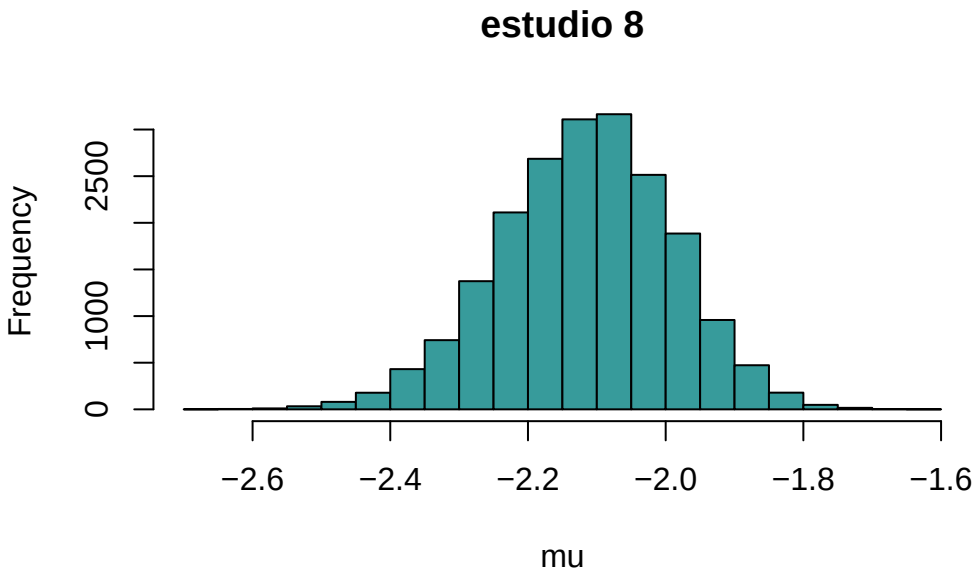

estudio 3
Frequency
0
1000
2000
−3.0
−2.5
−2.0
−1.5
mu

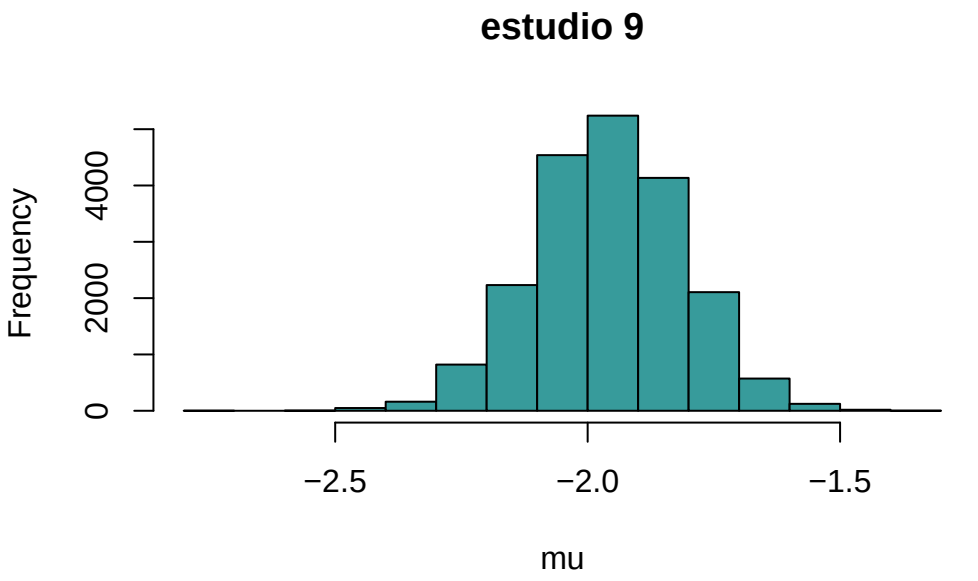

estudio 4
Frequency
0
2000
4000
−2.7
−2.6
−2.5
−2.4
−2.3
−2.2
−2.1
mu

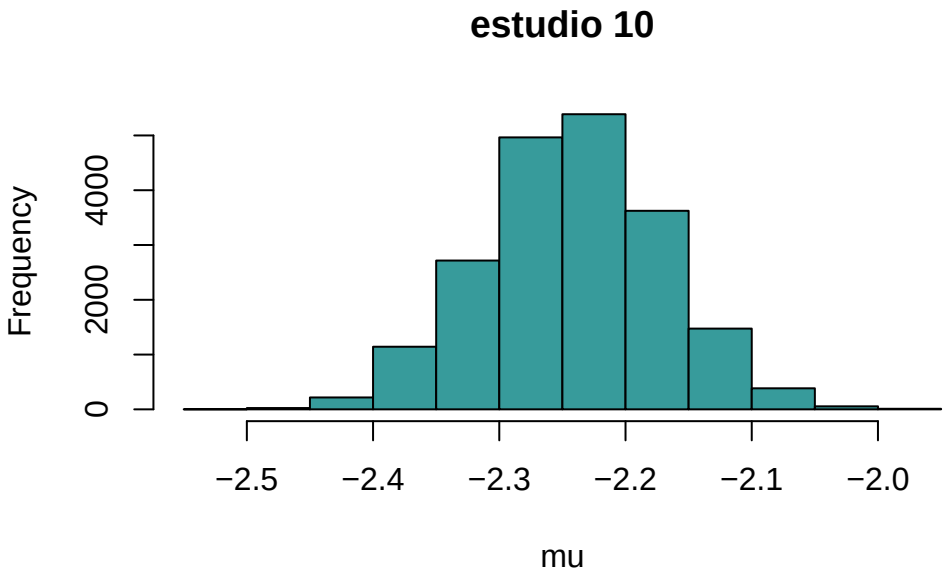

estudio 5
Frequency
0
2000
4000
−3.2
−3.0
−2.8
−2.6
−2.4
−2.2
−2.0
−1.8
mu

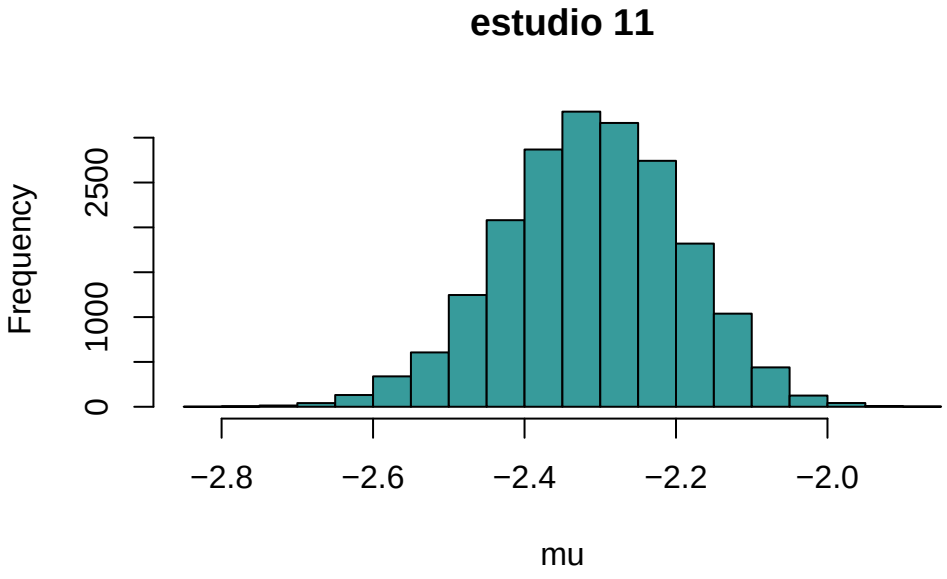

estudio 6
Frequency
0
2000
4000
−4.0
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
mu

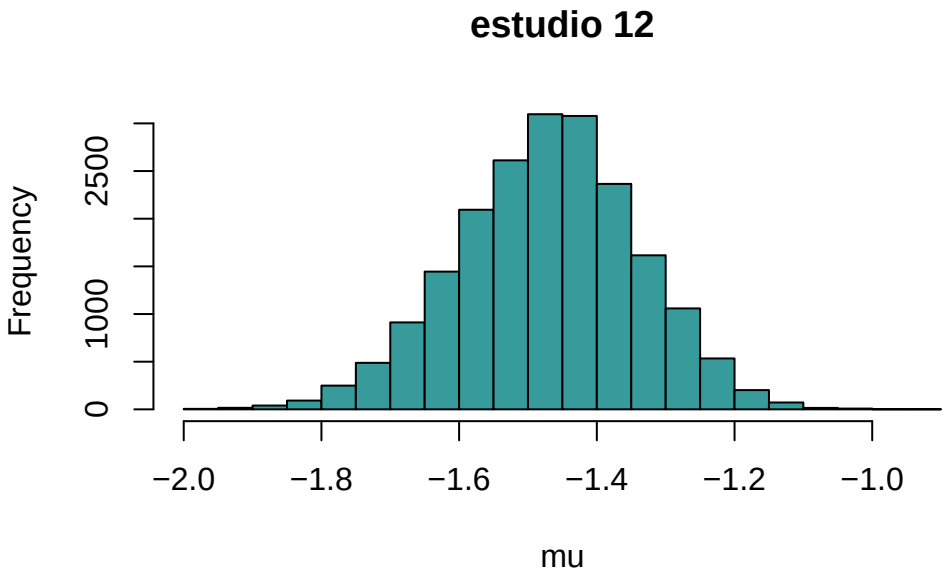

estudio 7
Frequency
0
2000
4000
−2.0
−1.9
−1.8
−1.7
−1.6
−1.5
−1.4
mu

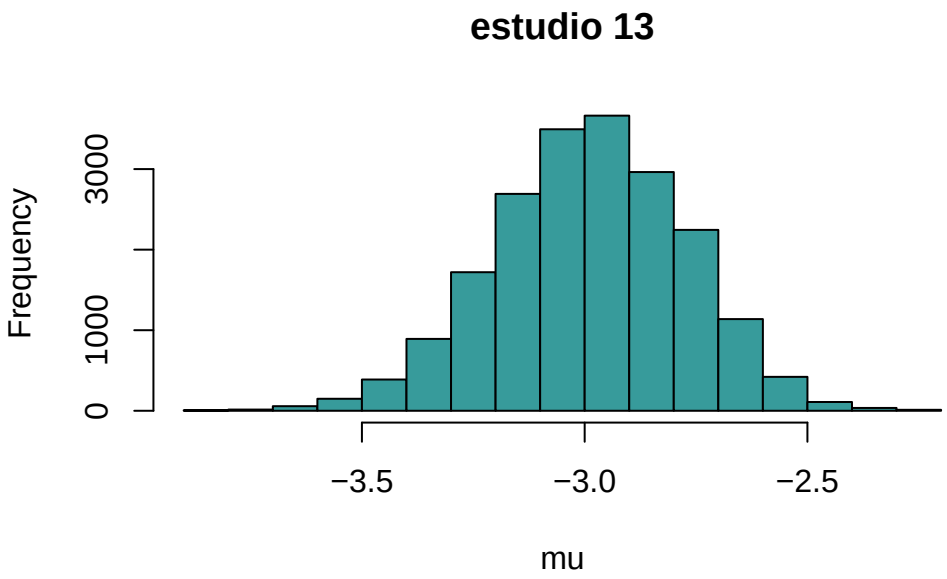

estudio 8
Frequency
0
1000
2500
−2.6
−2.4
−2.2
−2.0
−1.8
−1.6
mu

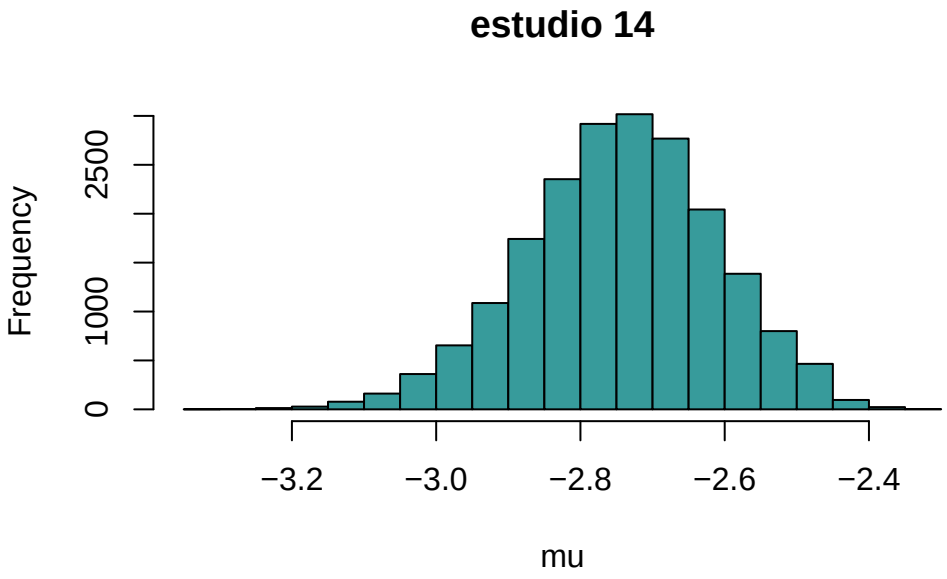

estudio 9
Frequency
0
2000
4000
−2.5
−2.0
−1.5
mu

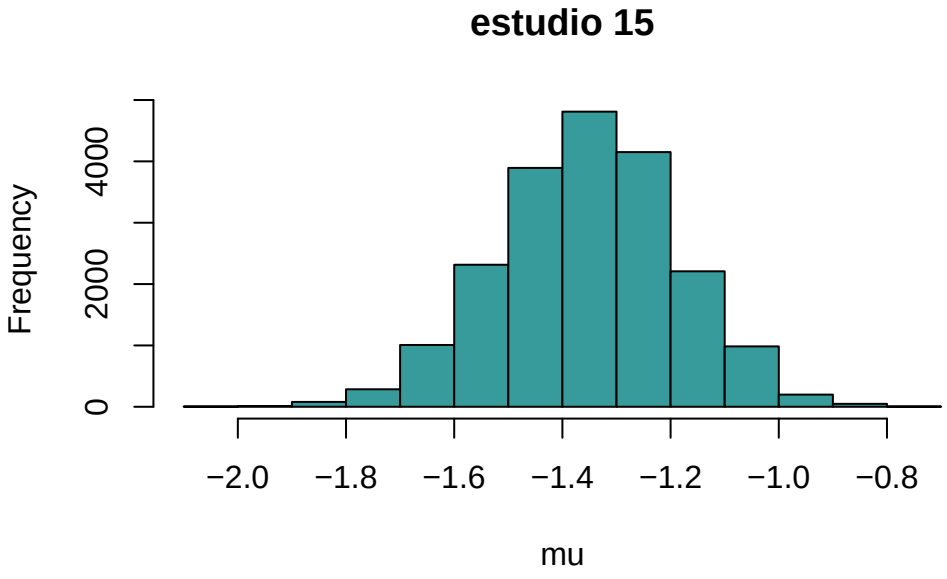

estudio 10
Frequency
0
2000
4000
−2.5
−2.4
−2.3
−2.2
−2.1
−2.0
mu

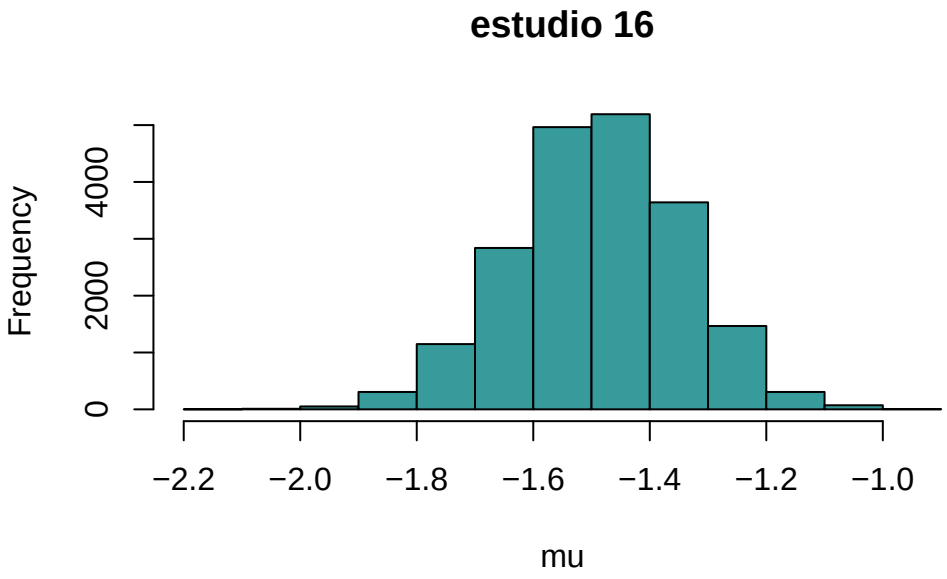

estudio 11
Frequency
0
1000
2500
−2.8
−2.6
−2.4
−2.2
−2.0
mu

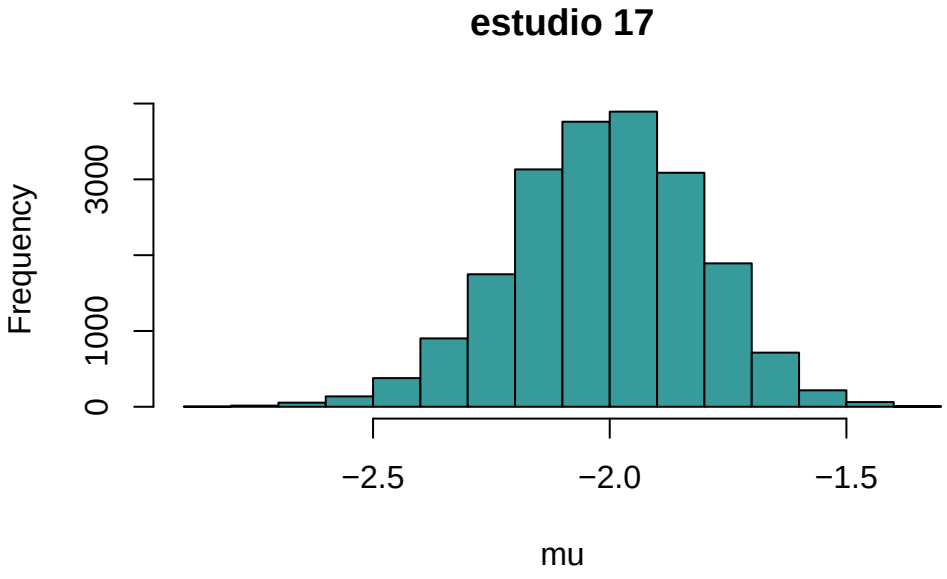

estudio 12
Frequency
0
1000
2500
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

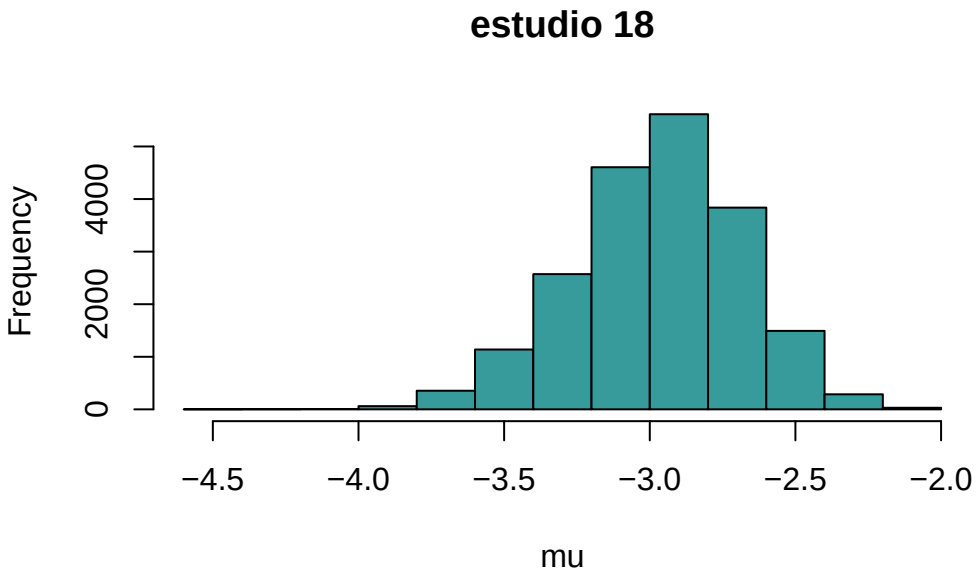

estudio 13
Frequency
0
1000
3000
−3.5
−3.0
−2.5
mu

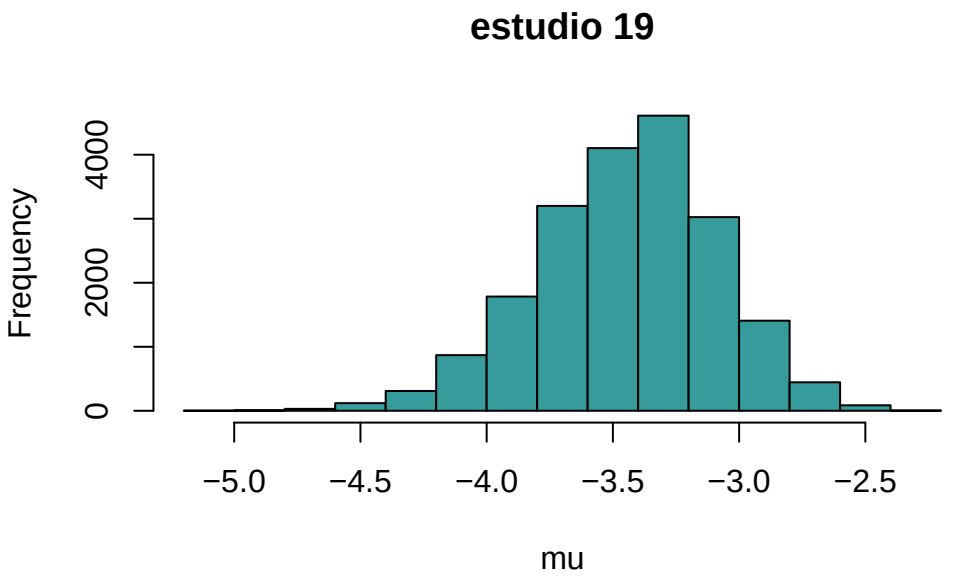

estudio 14
Frequency
0
1000
2500
−3.2
−3.0
−2.8
−2.6
−2.4
mu

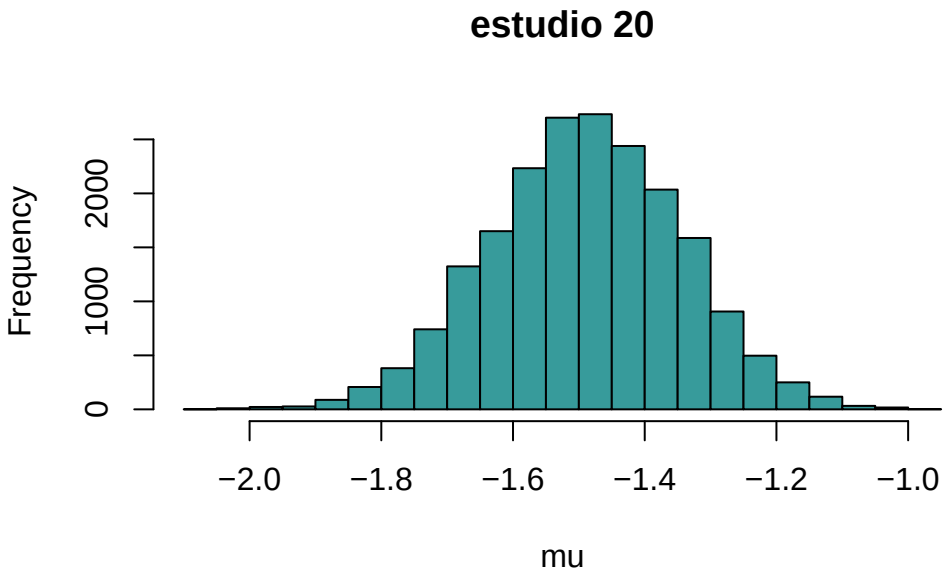

estudio 15
Frequency
0
2000
4000
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
mu

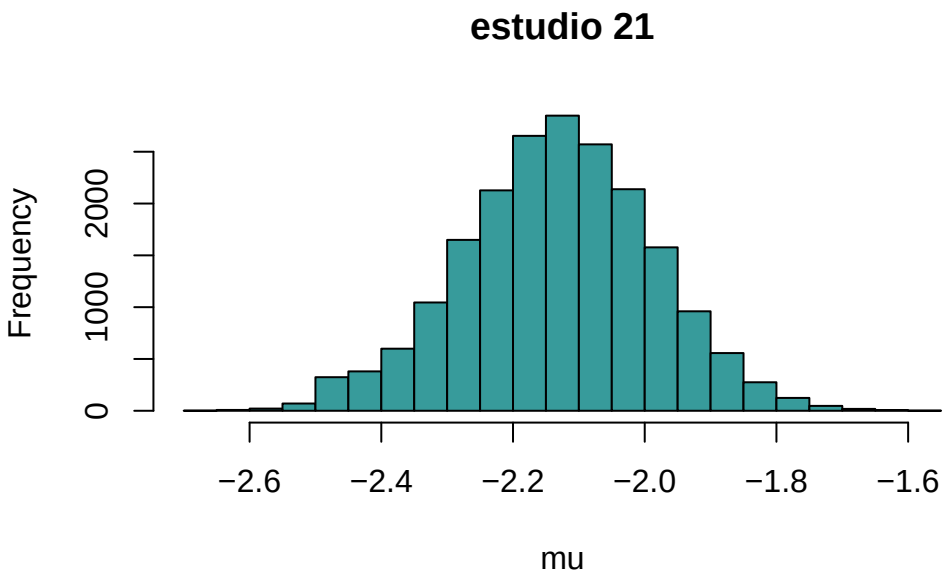

estudio 16
Frequency
0
2000
4000
−2.2
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

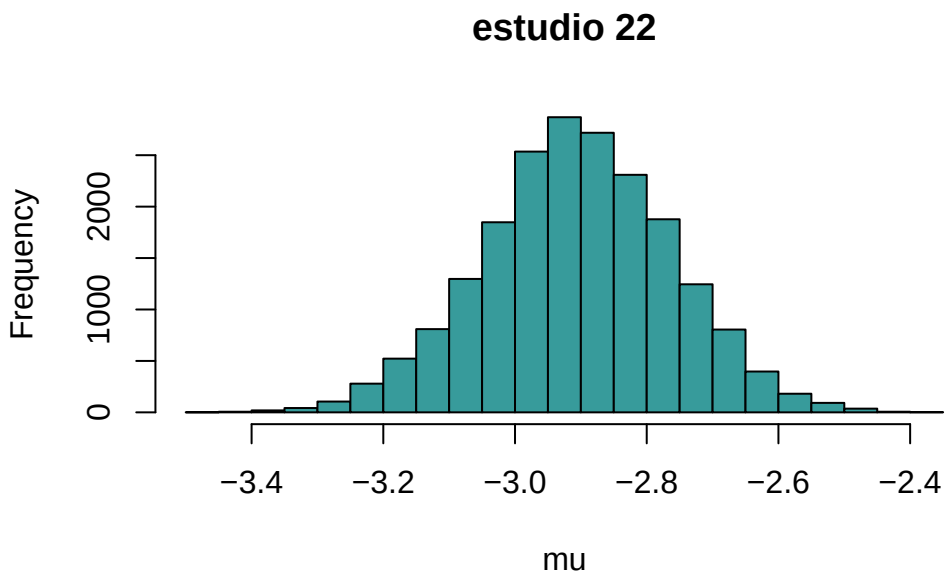

estudio 17
Frequency
0
1000
3000
−2.5
−2.0
−1.5
mu


estudio 18
Frequency
0
2000
4000
−4.5
−4.0
−3.5
−3.0
−2.5
−2.0
mu


estudio 19
Frequency
0
2000
4000
−5.0
−4.5
−4.0
−3.5
−3.0
−2.5
mu

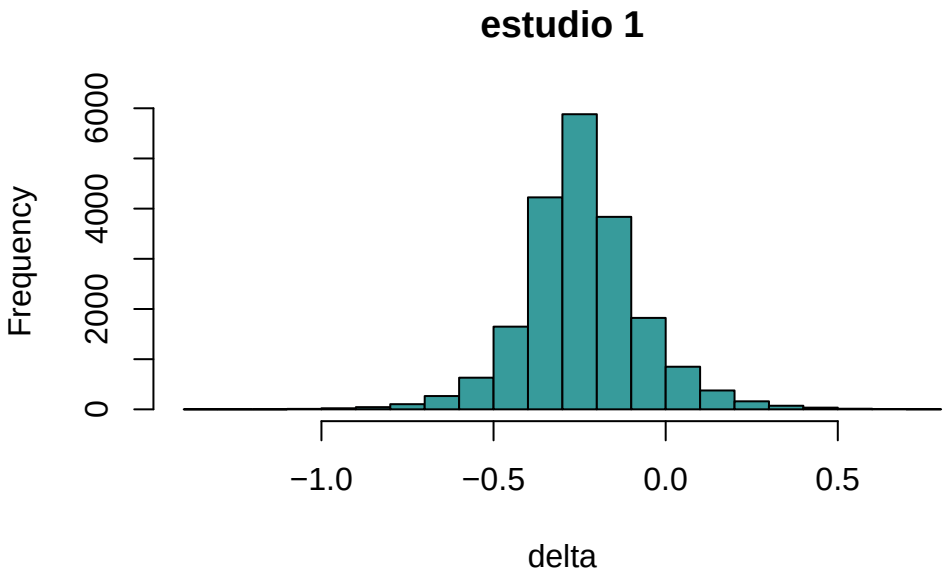

estudio 20
Frequency
0
1000
2000
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

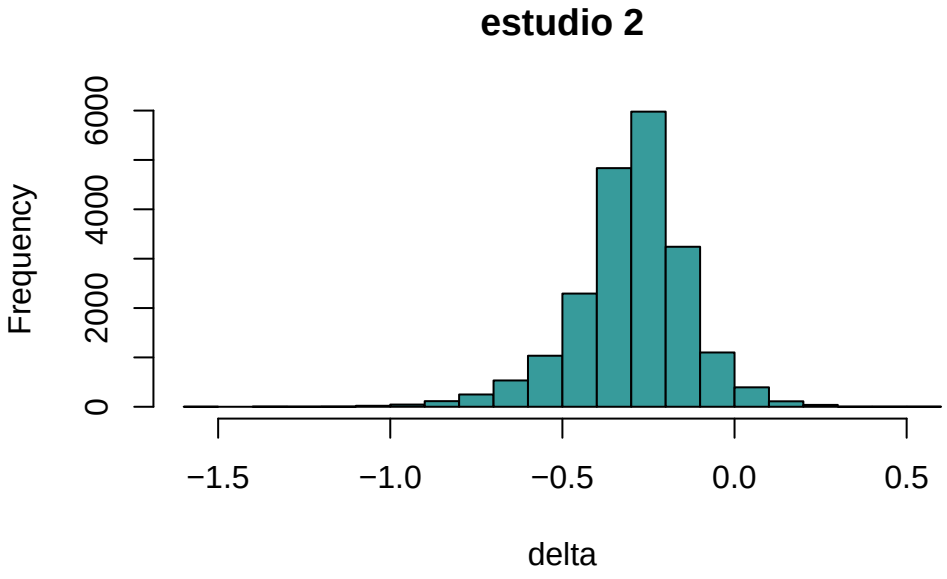

estudio 21
Frequency
0
1000
2000
−2.6
−2.4
−2.2
−2.0
−1.8
−1.6
mu

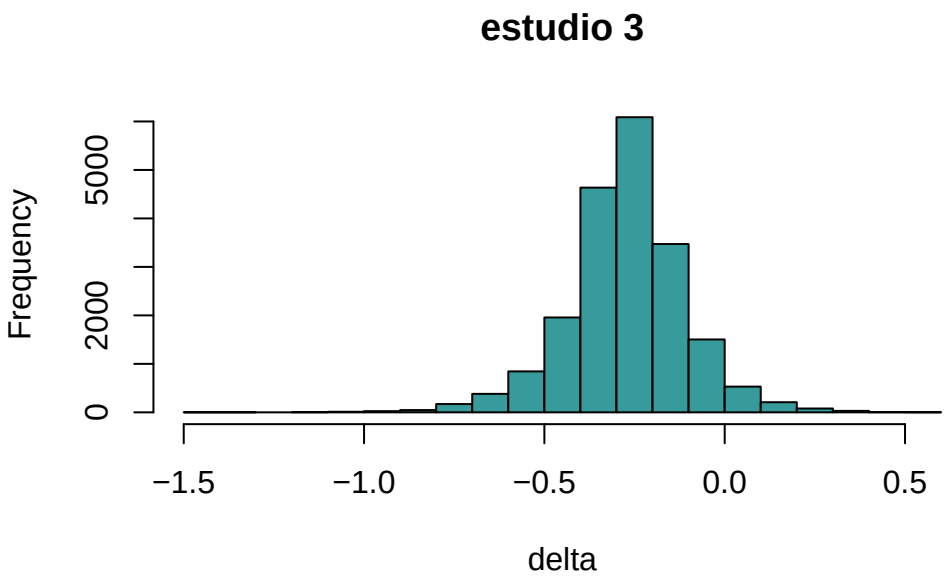


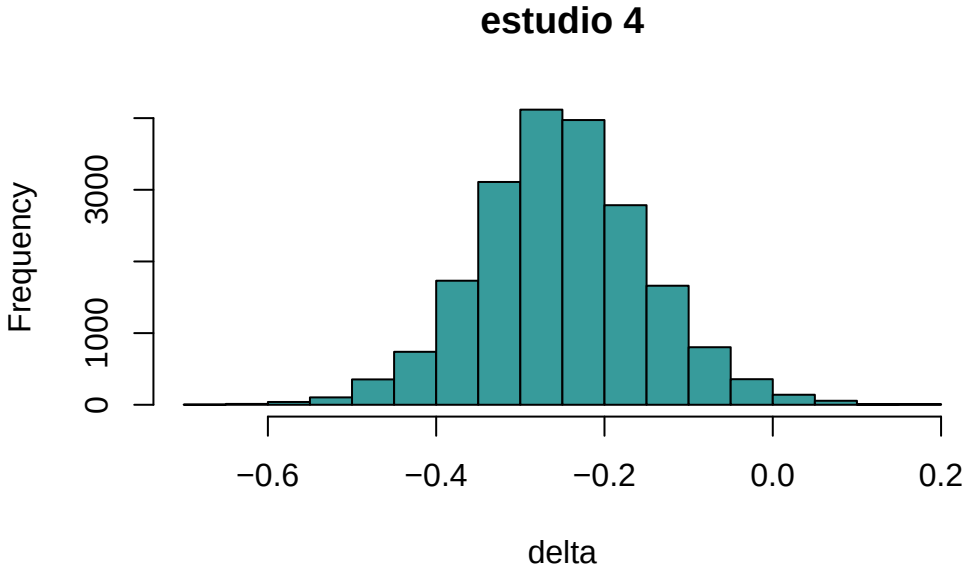
Histogramas de las posteriores de δ_i

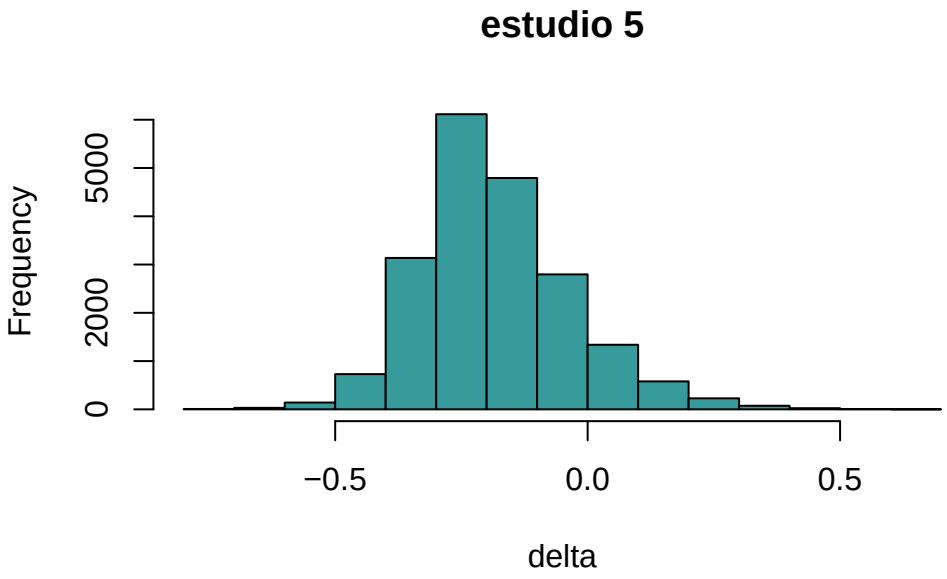
```
# par(mfrow = c(5, 5))

for (i in 1:22) {

  # Gráfico de densidad kernel de delta para el estudio i
  hist(delta_samples[, i], main = paste("estudio", i), xlab = "delta", col = "#379b9b")

}
```

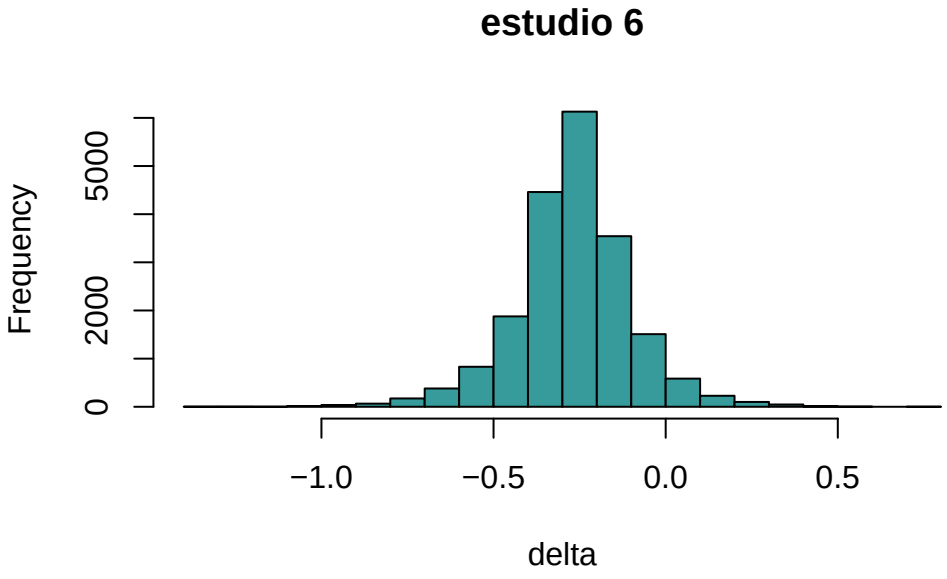

estudio 1
Frequency
0
2000
4000
6000
−1.0
−0.5
0.0
0.5
delta

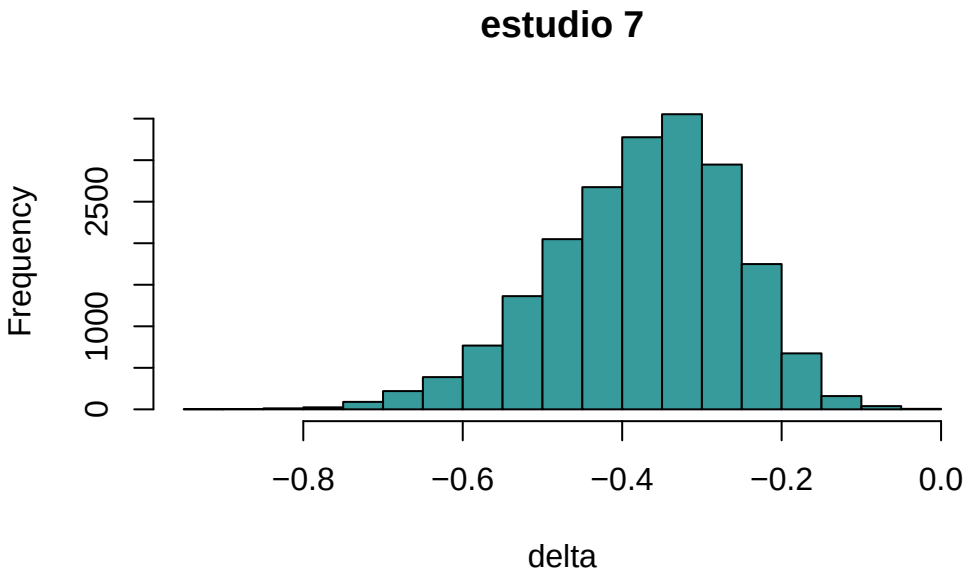

estudio 2
Frequency
0
2000
4000
6000
−1.5
−1.0
−0.5
0.0
0.5
delta

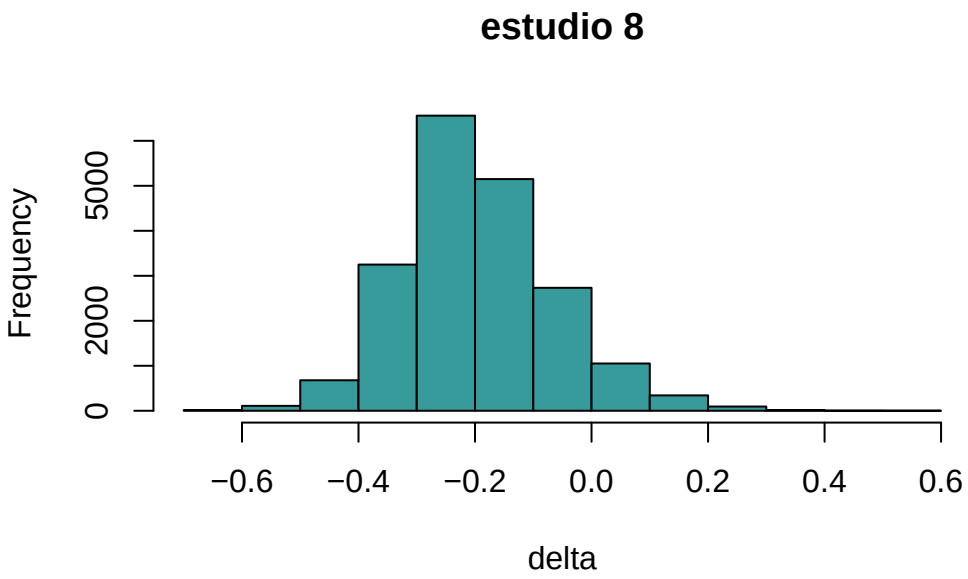

estudio 3
Frequency
0
2000
5000
−1.5
−1.0
−0.5
0.0
0.5
delta

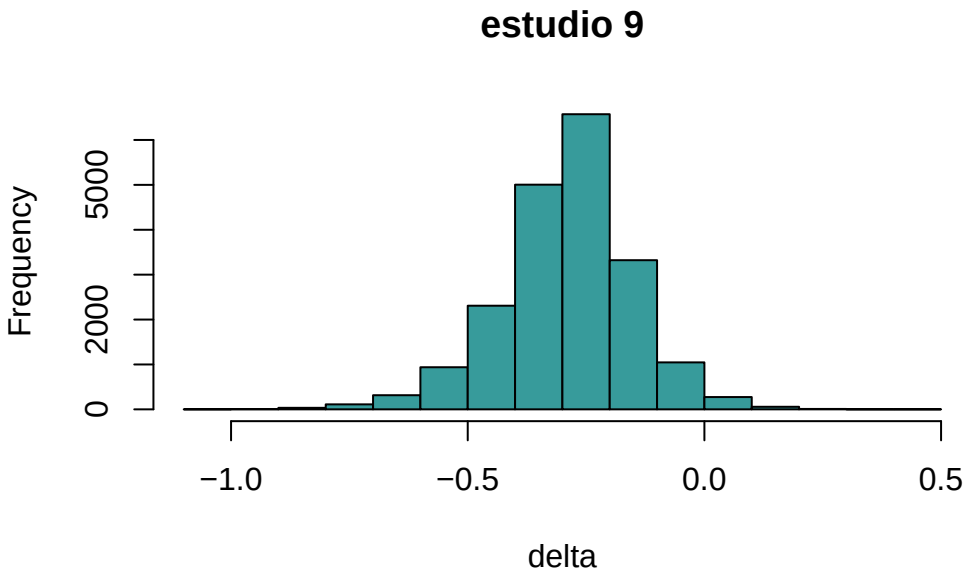

estudio 4
Frequency
0
1000
3000
−0.6
−0.4
−0.2
0.0
0.2
delta

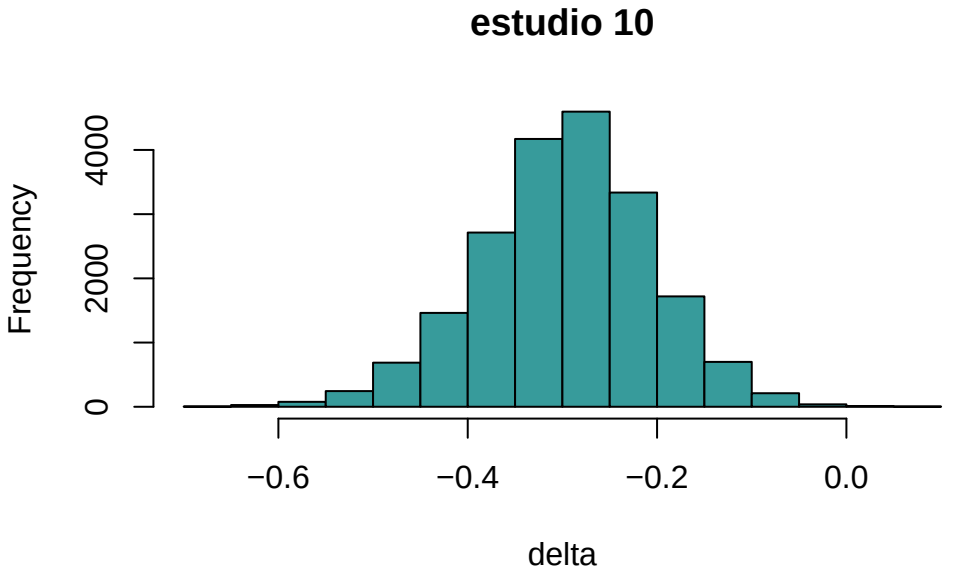

estudio 5
Frequency
0
2000
5000
−0.5
0.0
0.5
delta

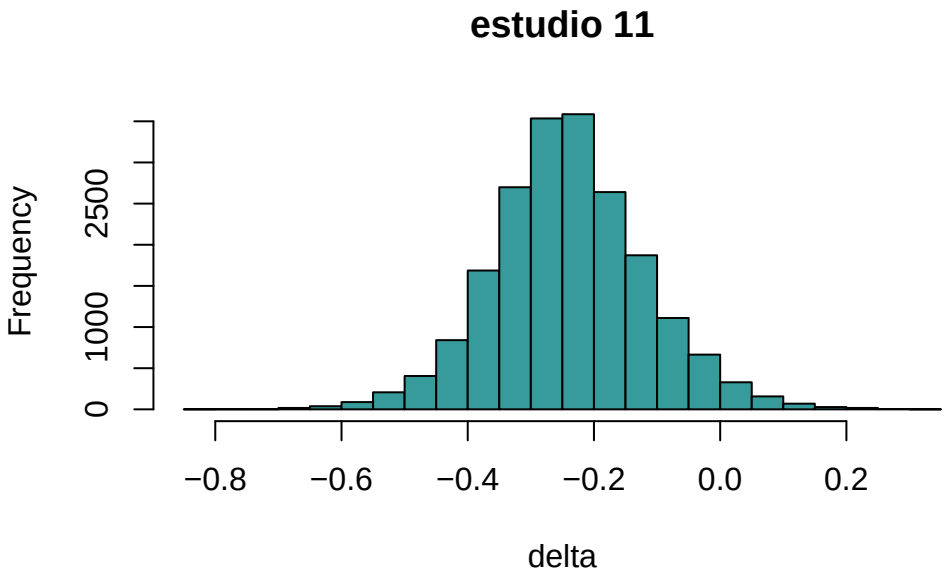

estudio 6
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

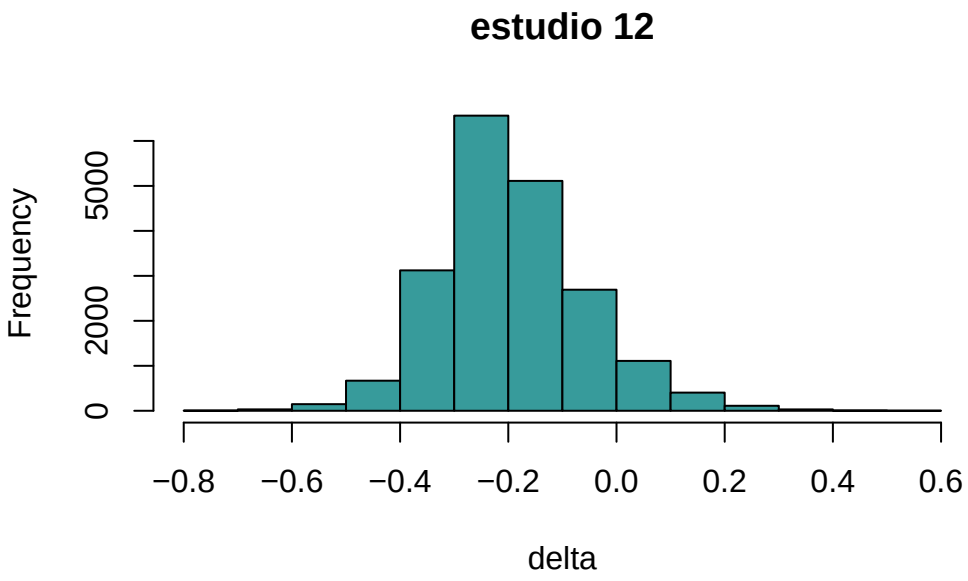

estudio 7
Frequency
0
1000
2500
−0.8
−0.6
−0.4
−0.2
0.0
delta

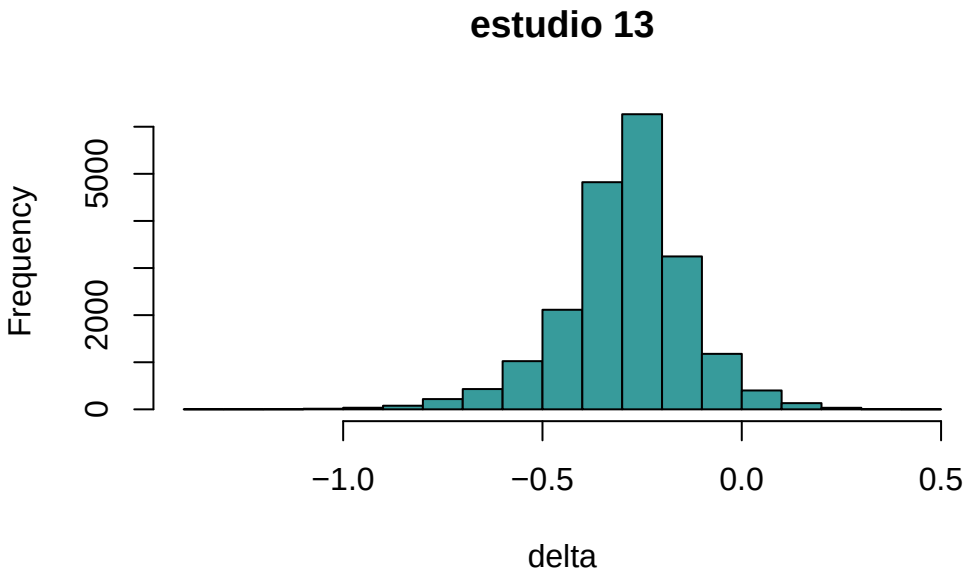

estudio 8
Frequency
0
2000
5000
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
delta

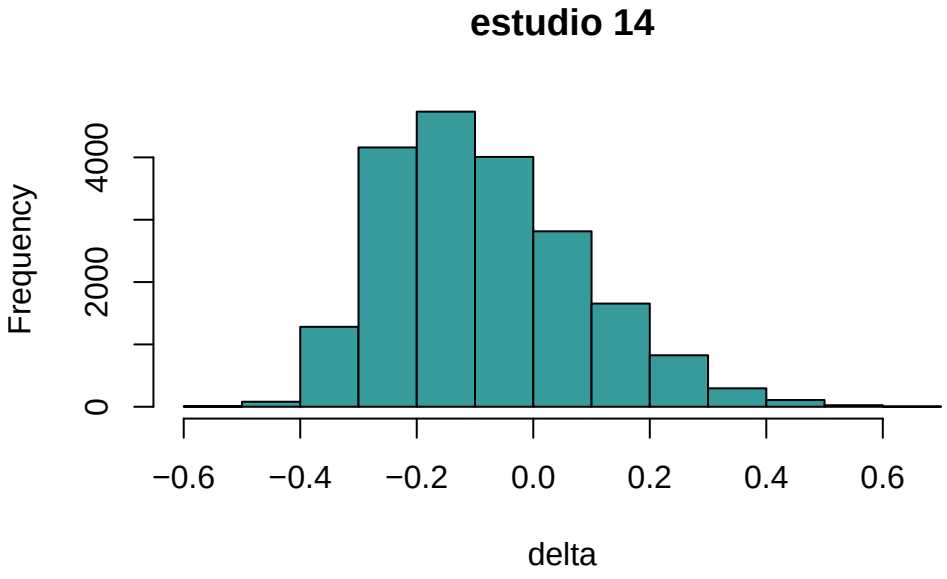

estudio 9
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

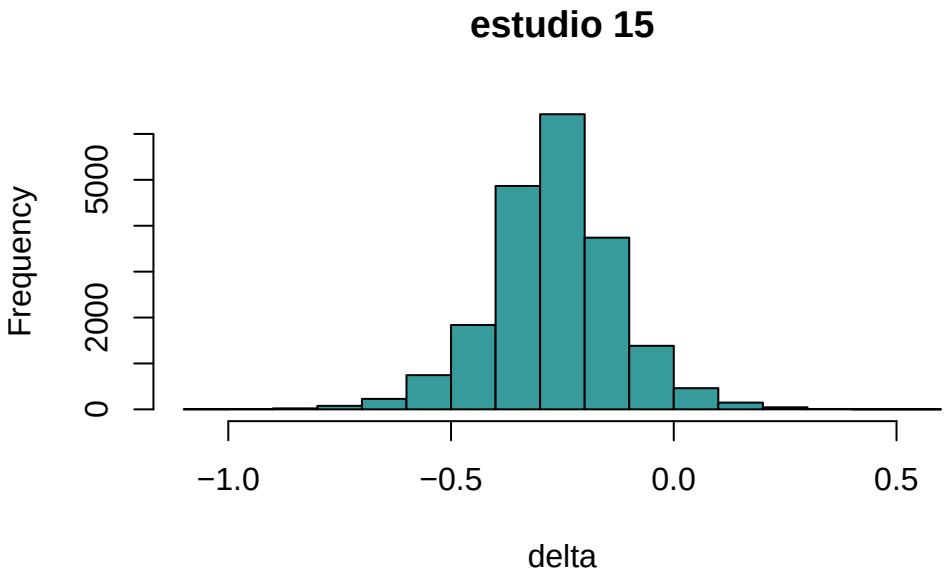

estudio 10
Frequency
0
2000
4000
−0.6
−0.4
−0.2
0.0
delta

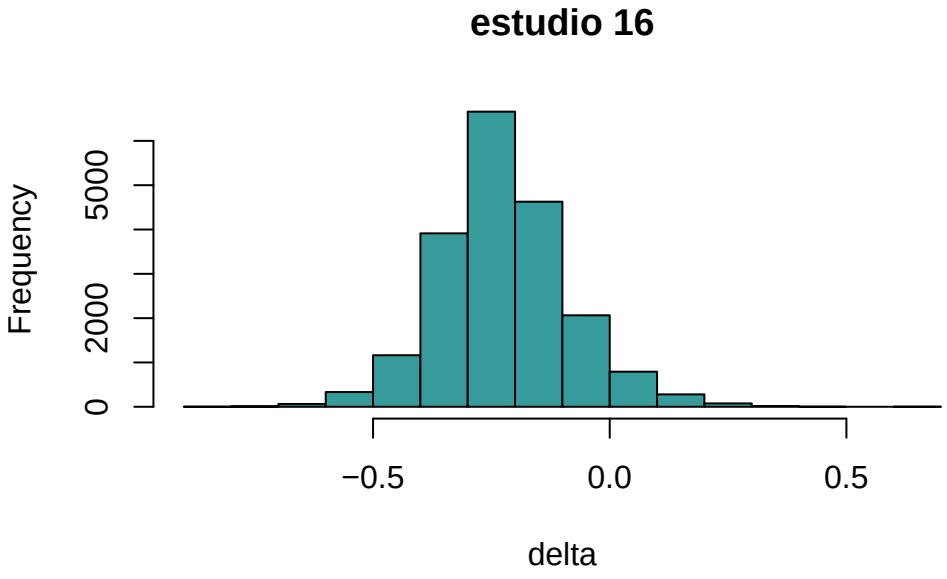

estudio 11
Frequency
0
1000
2500
−0.8
−0.6
−0.4
−0.2
0.0
0.2
delta

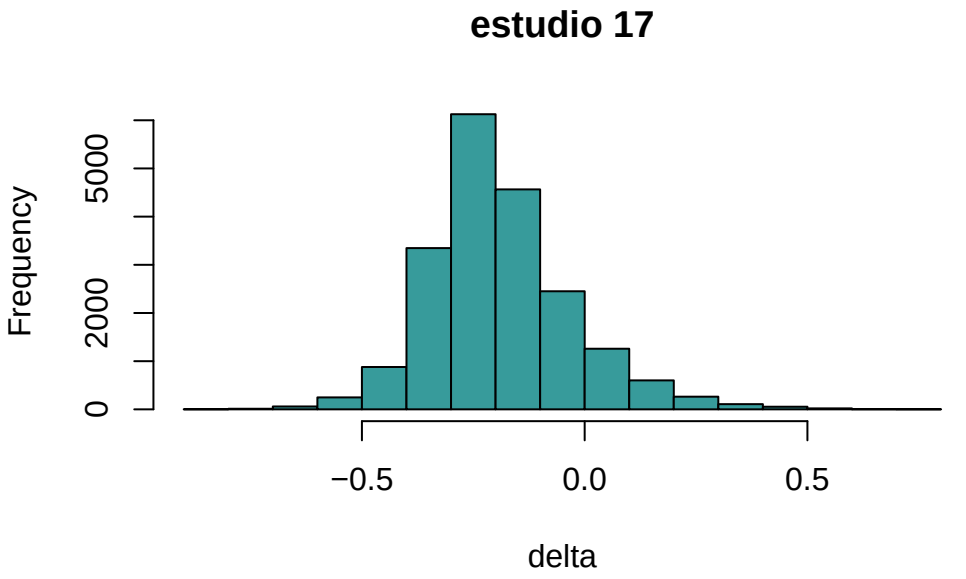

estudio 12
Frequency
0
2000
5000
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
delta

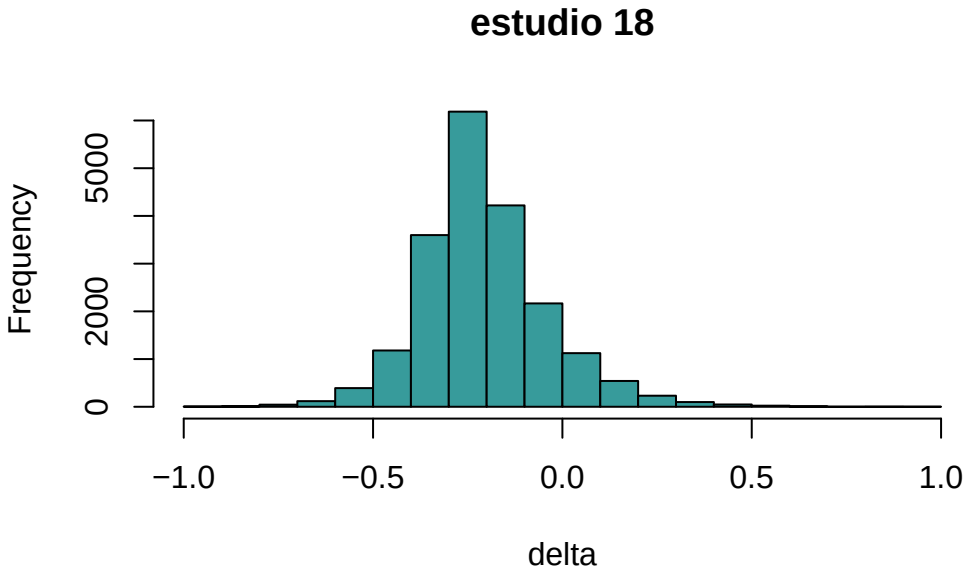

estudio 13
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

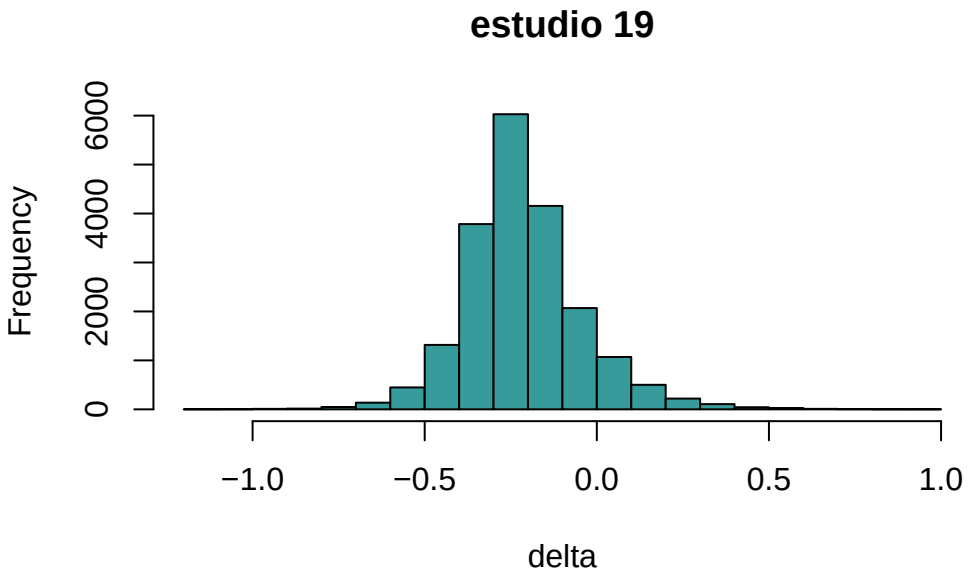

estudio 14
Frequency
0
2000
4000
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
delta

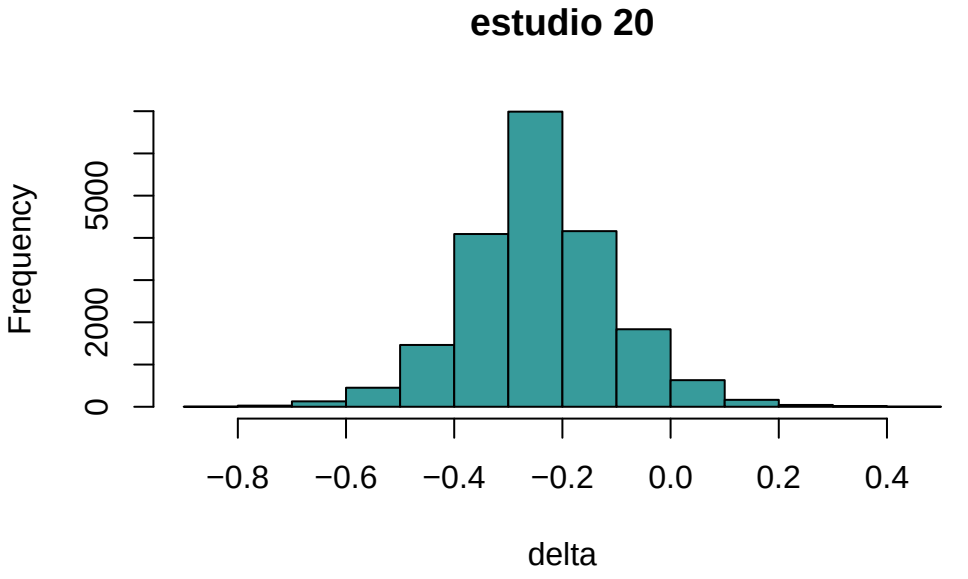

estudio 15
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

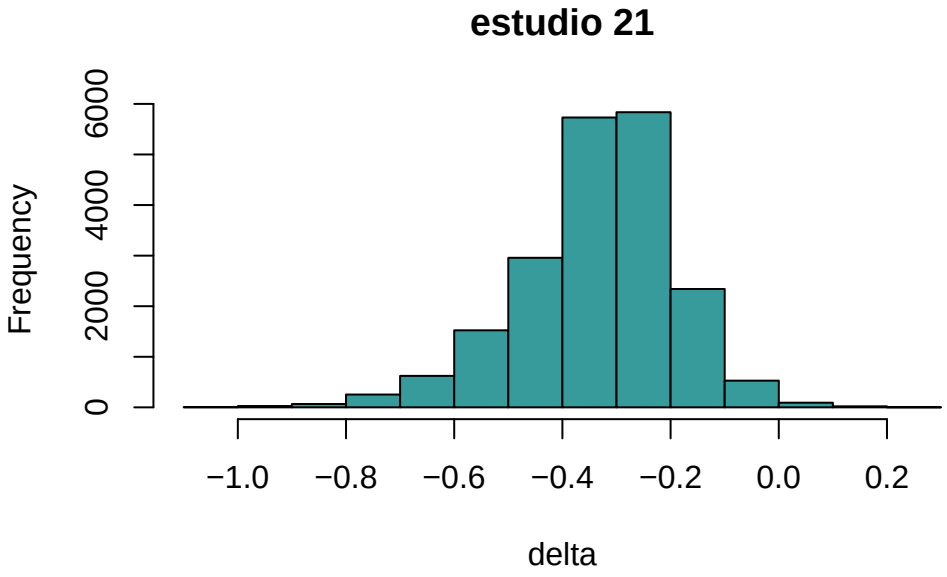

estudio 16
Frequency
0
2000
5000
−0.5
0.0
0.5
delta

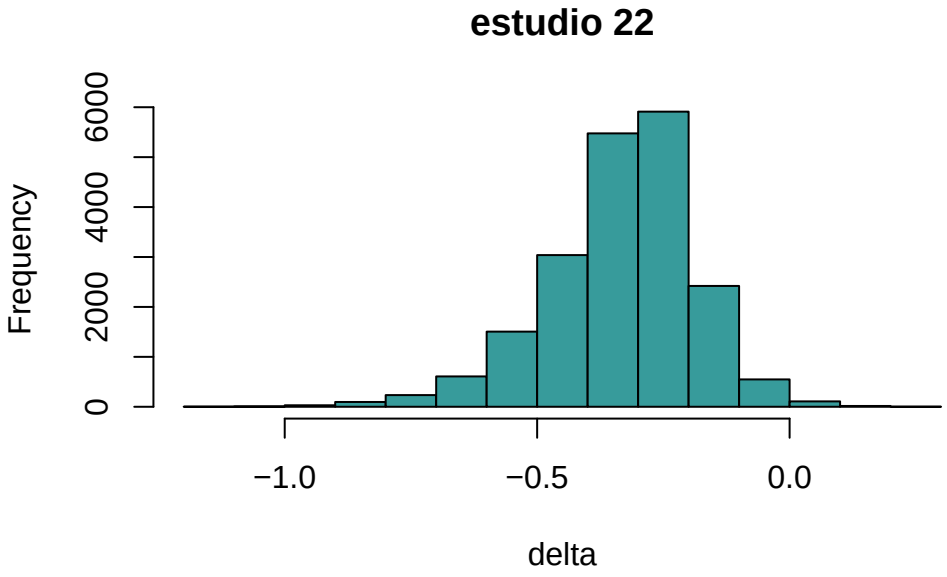

estudio 17
Frequency
0
2000
5000
−0.5
0.0
0.5
delta


estudio 18
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
1.0
delta


estudio 19
Frequency
0
2000
4000
6000
−1.0
−0.5
0.0
0.5
1.0
delta


estudio 20
Frequency
0
2000
5000
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
delta





conclusiones:

- Como vemos, la diferencia principal entre este modelo jerárquico y el modelo anterior radica en cómo se modela la variabilidad entre los efectos del tratamiento:
  - En el modelo original, se suponía que cada efecto del tratamiento δ_j tenía su propia distribución independiente, con su propia media y desviación estándar, por lo que no hay un mecanismo para compartir información entre los diferentes efectos del tratamiento.
  - En el modelo jerárquico, en cambio, se introduce una distribución común para todos los efectos del tratamiento. Esto se logra al modelar cada δ_j como una muestra de una distribución normal con una media común μ y una desviación estándar común σ. Esta estructura jerárquica permite que los efectos del tratamiento compartan información entre sí.
  - Variabilidad entre estudios: Como podemos observar, hay una variabilidad considerablemente menor en las distribuciones posteriores de δ_i en comparación con las del modelo anterior, esto indica que el efecto de los beta-bloqueadores varía poco según el contexto o las características de cada estudio pues están relacionados entre sí.
  - Efecto del tratamiento: La media de las δ_i se encuentran en el rango entre -0.37 y -0.09, presentando una clara tendencia a ser negativas, lo que indica que los beta-bloqueadores tienen un efecto consistente y considerablemente beneficioso en la reducción del riesgo de infarto en todos los estudios.

c. Estimación

Para un ensayo fuera de la muestra suponer que se sabe que $\mu_i = 2.5$. Usando la estimación de δ del estudio cruzado, estimar la reducción en probabilidad para un paciente que toma beta-bloqueadores.

Con $\mu_i = 2.5$ tenemos información sobre la probabilidad de mortalidad en el grupo de control del ensayo específico. Queremos estimar la reducción en la probabilidad de mortalidad para un paciente que toma beta-bloqueadores, basándonos en la estimación de δ del estudio cruzado.

- Para el grupo de control: $p_c = logit^{-1}(\mu_i) = logit^{-1}(2.5)$
- Para el grupo de tratamiento: $p_t = logit^{-1}(\mu_i \delta_{estudiocruzado}) = logit^{-1}(2.5 + \delta_{estudiocruzado})$
- La reducción en la probabilidad de mortalidad sería: Reducción $= p_c - p_t$

```r
reducciones_probabilidad_samples <- numeric(22)

# Para el grupo de control
mu_i <- 2.5
```

```r
p_c <- plogis(mu_i)

means_delta <- apply(delta_samples, 2, mean)
#mean_delta <- mean(means_delta) # media del estudio cruzado

# Para el grupo de tratamiento
#delta_estudio_cruzado <- mean_delta

p_t <- plogis(mu_i + means_delta)


# Reducción en la probabilidad de mortalidad
reduccion_probabilidad <- p_c - p_t
# Crear un data frame con los resultados
resultados <- data.frame(
  "p_c" = p_c,
  "p_t" = p_t,
  "Reduccion" = reduccion_probabilidad
)

# Imprimir el data frame
print(resultados)
```

```
        p_c       p_t   Reduccion
1  0.9241418 0.9054106 0.018731204
2  0.9241418 0.9003129 0.023828964
3  0.9241418 0.9026742 0.021467669
4  0.9241418 0.9046271 0.019514712
5  0.9241418 0.9102467 0.013895094
6  0.9241418 0.9030380 0.021103774
7  0.9241418 0.8933876 0.030754237
8  0.9241418 0.9091731 0.014968759
9  0.9241418 0.9011311 0.023010749
10 0.9241418 0.9005313 0.023610549
11 0.9241418 0.9055564 0.018585454
12 0.9241418 0.9093722 0.014769663
13 0.9241418 0.9010158 0.023126038
14 0.9241418 0.9180516 0.006090198
15 0.9241418 0.9032076 0.020934229
16 0.9241418 0.9068611 0.017280721
17 0.9241418 0.9097389 0.014402927
18 0.9241418 0.9082802 0.015861614
```

```
19 0.9241418 0.9076924 0.016449404
20 0.9241418 0.9052852 0.018856631
21 0.9241418 0.8971738 0.026968015
22 0.9241418 0.8971731 0.026968679
```

d.

Estimar un modelo con sólo valores constantes δ y μ a través de los ensayos. Graficar la posterior de δ, y comparar con el estimador del modelo jerárquico del estudio.

```
# Modelo en Stan
stan_code <- '
data {
  int<lower=0> J;           // Número de estudios
  int rt[J];                // Número de muertes en tratamiento
  int nt[J];                // Tamaño de la muestra en tratamiento
  int rc[J];                // Número de muertes en control
  int nc[J];                // Tamaño de la muestra en control
}

parameters {
  real delta;               // Efecto constante del tratamiento
  real mu;                  // Efecto constante del control
}

model {
  mu ~ normal(0, 10);       // Priori para el efecto del control
  delta ~ normal(0, 10);    // Priori para el efecto del tratamiento

  for (j in 1:J) {
    rc[j] ~ binomial_logit(nc[j], mu);                          // Verosimilitud del cont
    rt[j] ~ binomial_logit(nt[j], mu + delta);                  // Verosimilitud del trat
  }
}
'

# Compilar el modelo
stan_model <- stan_model(model_code = stan_code)
```

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

```
Running /usr/lib/R/bin/R CMD SHLIB foo.c
using C compiler: ‘gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0’
gcc -I"/usr/share/R/include" -DNDEBUG   -I"/home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-
In file included from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-pc-linux-gnu/StanHead
                 from <command-line>:
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
  628 | namespace Eigen {
      | ^~~~~~~~~
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
  628 | namespace Eigen {
      |                 ^
In file included from /home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-li
                 from /home/saraluz/.cache/R/renv/cache/v5/R-4.3/x86_64-pc-linux-gnu/StanHead
                 from <command-line>:
/home/saraluz/Documents/2Sem/MLG/Tareas/renv/library/R-4.3/x86_64-pc-linux-gnu/RcppEigen/incl
   96 | #include <complex>
      |          ^~~~~~~~~
compilation terminated.
make: *** [/usr/lib/R/etc/Makeconf:191: foo.o] Error 1
```

```r
# Datos para Stan
stan_data <- list(
  J = nrow(data), # Número de estudios
  rt = data$rt,
  nt = data$nt,
  rc = data$rc,
  nc = data$nc,
  N = data$N
)

# Muestreo de la distribución posterior
stan_samples <- sampling(stan_model, data = stan_data, iter = 10000, chains = 4)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
Chain 1: Adjust your expectations accordingly!
```

```
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.174 seconds (Warm-up)
Chain 1:                0.19 seconds (Sampling)
Chain 1:                0.364 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 2: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 2: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 2: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 2: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 2: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 2: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 2: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 2: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 2: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 2: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 2: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.173 seconds (Warm-up)
Chain 2:                0.204 seconds (Sampling)
Chain 2:                0.377 seconds (Total)
```

```
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 7e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 3: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 3: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 3: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 3: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 3: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 3: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 3: Iteration: 7000 / 10000 [ 70%]  (Sampling)
Chain 3: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 3: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 3: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.173 seconds (Warm-up)
Chain 3:                0.188 seconds (Sampling)
Chain 3:                0.361 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 7e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 4: Iteration: 1000 / 10000 [ 10%]  (Warmup)
Chain 4: Iteration: 2000 / 10000 [ 20%]  (Warmup)
Chain 4: Iteration: 3000 / 10000 [ 30%]  (Warmup)
Chain 4: Iteration: 4000 / 10000 [ 40%]  (Warmup)
Chain 4: Iteration: 5000 / 10000 [ 50%]  (Warmup)
Chain 4: Iteration: 5001 / 10000 [ 50%]  (Sampling)
Chain 4: Iteration: 6000 / 10000 [ 60%]  (Sampling)
Chain 4: Iteration: 7000 / 10000 [ 70%]  (Sampling)
```

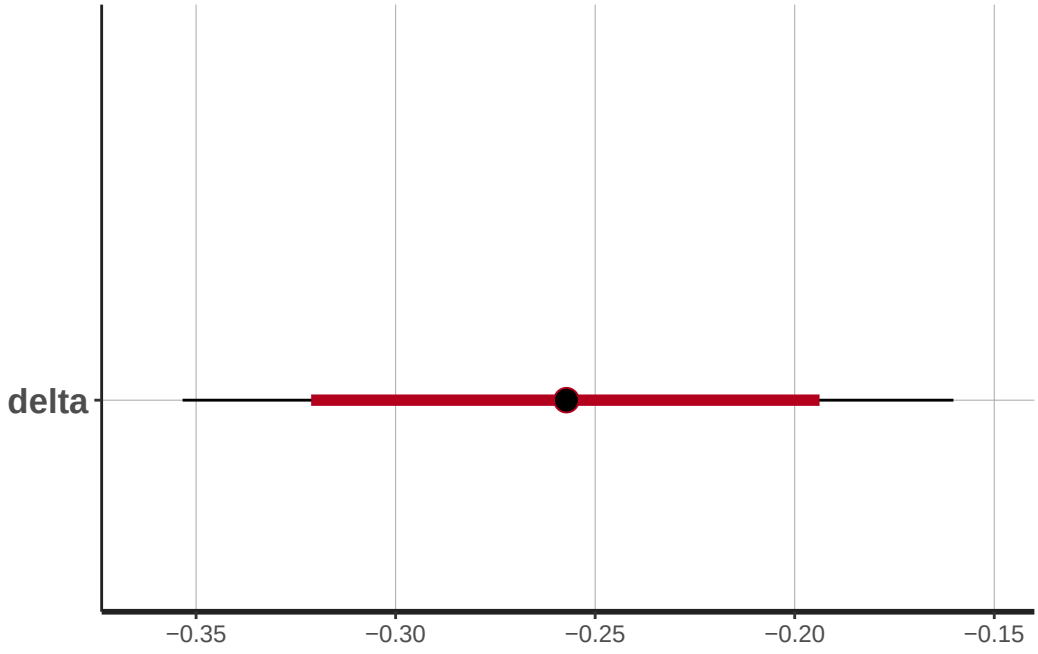
```
Chain 4: Iteration: 8000 / 10000 [ 80%]  (Sampling)
Chain 4: Iteration: 9000 / 10000 [ 90%]  (Sampling)
Chain 4: Iteration: 10000 / 10000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.172 seconds (Warm-up)
Chain 4:                0.188 seconds (Sampling)
Chain 4:                0.36 seconds (Total)
Chain 4:
```

Graficamos la posterior de δ y su histograma

```r
# Graficar la posterior de delta
plot(stan_samples, pars = "delta")
```

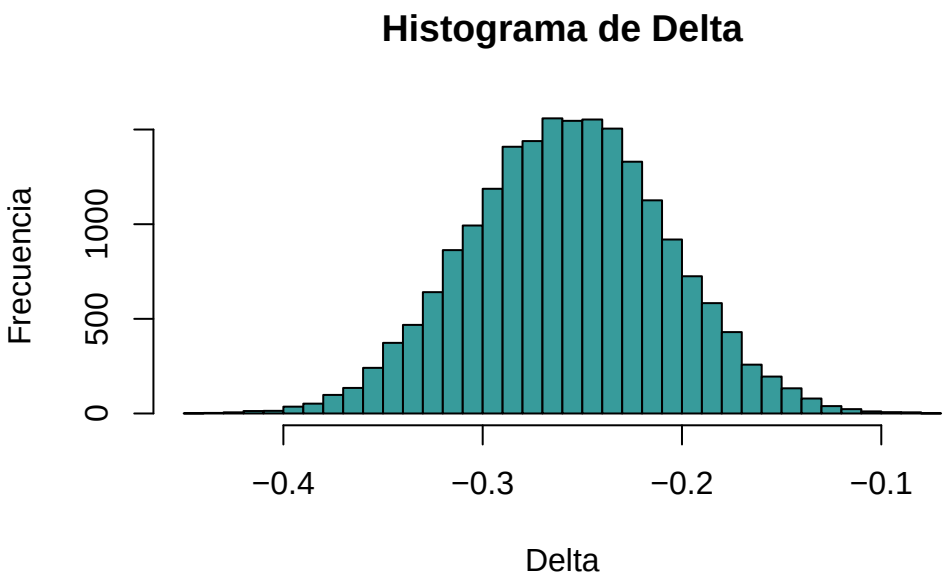
```
ci_level: 0.8 (80% intervals)
```

```
outer_level: 0.95 (95% intervals)
```




```r
posterior_samples <- rstan::extract(stan_samples)
```

```r
par(mfrow = c(1, 1))

# Histograma de las muestras posteriores de delta
hist(posterior_samples$delta, breaks = 30, col = "#379b9b", xlab = "Delta", ylab = "Frecue
```



Conclusiones:

- Complejidad del modelo: El modelo con solo valores constantes es menos complejo que el modelo jerárquico, ya que no tiene parámetros aleatorios ni estructura jerárquica.
- Flexibilidad: El modelo jerárquico permite que los efectos del tratamiento δ y del control μ varíen entre los ensayos, lo que captura mejor la heterogeneidad entre los estudios.
- Observamos que la media de delta en este modelo es de -0.26 lo que refuerza el efecto beneficioso que observamos en el modelo jerárquico.

2. Los siguientes datos son de un estudio (Belenky, et. al. 2003) que mide el efecto de la privación del sueño en el desempeño cognitivo. Hubo 18 sujetos elegidos de una población de internet (conductores de camiones) a los que se les restringió 3 horas de sueño durante el ensayo. En cada día del experimento se midió el tiempo de reacción visual a un estímulo. Los datos para este ejemplo están en el archivo evaluation_sleepstudy.csv, consiste de tres variables: Reaction, Days y SubjetID, que mide el tiempo de reacción de un sujeto dado en un día particular. Un modelo

simple que explica la variación en tiempos de reacción es un modelo de regresión lineal de la forma: $R(t) \sim \mathcal{N}(\alpha + \beta t, \sigma^2)$, donde $R(t)$ es el tiempo de reacción en el día t del experimento a través de todas las observaciones.

```
sleep <- read.csv("evaluation_sleepstudy.csv")
print(unique(sleep$Subject))
```

```
 [1] 308 309 310 330 331 332 333 334 335 337 349 350 351 352 369 370 371 372
```

```
sleep <- sleep %>%
  mutate(Subject = Subject-307) %>%  # Arregle estos indices porque sino el modelo se quej
  mutate(Subject = ifelse(Subject >= 23, Subject-19, Subject)) %>%
  mutate(Subject = ifelse(Subject >= 11, Subject-1, Subject)) %>%
  mutate(Subject = ifelse(Subject >= 22, Subject-11, Subject)) %>%
  mutate(Subject = ifelse(Subject >= 31, Subject-16, Subject))
glimpse(sleep)
```

```
Rows: 180
Columns: 4
$ X        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18~
$ Reaction <dbl> 249.5600, 258.7047, 250.8006, 321.4398, 356.8519, 414.6901, 3~
$ Days     <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0~
$ Subject  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3~
```

```
mean(sleep$Reaction)
```

```
[1] 298.5079
```

```
sd(sleep$Reaction)
```

```
[1] 56.32876
```

a. Suponiendo iniciales $\mathcal{N}(0, 250)$ para ambos α y β, ajustar el modelo anterior, usando 1000 muestras por cadena, para cinco cadenas. £Converge el algoritmo?

```
data_list = list(
  N = nrow(sleep),
```

```r
    rt_obs = sleep$Reaction,
    t = sleep$Days,
    subject = sleep$Subject
  )

  modelo_2a <- cmdstan_model("./Tarea5_2a.stan")
  print(modelo_2a)
```

```
data {
  int<lower=0> N;      // Numero de datos
  vector<lower=0>[N] rt_obs;    // R(t), tiempos de reacción observados
  vector<lower=0>[N] t;         // días sin dormir
  array[N] int subject;     // i-ésima persona
}

parameters {
  vector[N] alpha;
 /* vector[N] beta; */
  real<lower=0> sigma;
}

transformed parameters {
  vector[N] media;
  for (i in 1:N){
    media[i] = alpha[subject[i]] + alpha[subject[i]] * t[i]; //modelo dado
  }
}


model {
  for (i in 1:N){
    rt_obs[i] ~ normal(media[i], sigma); //modelo dado
  }
  alpha ~ normal(0, 250);  // distribucion dada
 // beta ~ normal(0, 250);   // distribucion dada
}
```

```r
  fit2a <- modelo_2a$sample(
    data = data_list,
    seed = 123,
    chains = 5,
```

```
    iter_sampling = 1000,
    iter_warmup = 1000,
    # show_messages = FALSE,
    # show_exceptions = FALSE
    )
```

```
Running MCMC with 5 sequential chains...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 1.8 seconds.
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2 finished in 1.9 seconds.
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3 finished in 1.7 seconds.
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
```

```
Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4 finished in 1.6 seconds.
Chain 5 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 5 Iteration:  100 / 2000 [  5%]  (Warmup)
Chain 5 Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 5 Iteration:  300 / 2000 [ 15%]  (Warmup)
Chain 5 Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 5 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 5 Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 5 Iteration:  700 / 2000 [ 35%]  (Warmup)
Chain 5 Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 5 Iteration:  900 / 2000 [ 45%]  (Warmup)
Chain 5 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 5 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 5 Iteration: 1100 / 2000 [ 55%]  (Sampling)
Chain 5 Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 5 Iteration: 1300 / 2000 [ 65%]  (Sampling)
Chain 5 Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 5 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 5 Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 5 Iteration: 1700 / 2000 [ 85%]  (Sampling)
Chain 5 Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 5 Iteration: 1900 / 2000 [ 95%]  (Sampling)
Chain 5 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 5 finished in 1.7 seconds.

All 5 chains finished successfully.
Mean chain execution time: 1.7 seconds.
Total execution time: 9.0 seconds.
```

b. Graficar las muestras de la distribución posterior tanto de α como de β, £Cuál es la relación entre las dos variables y por qué?

```
fit2a
```

```
variable     mean   median   sd  mad       q5      q95 rhat ess_bulk ess_tail
lp__     -1030.32 -1029.85 9.45 9.52 -1046.52 -1015.44 1.00     1698     2726
alpha[1]    53.57    53.42 6.20 6.30    43.50    63.88 1.00     9615     3466
alpha[2]    31.15    31.14 6.16 6.18    21.10    41.18 1.00     9521     3218
alpha[3]    34.27    34.28 6.42 6.50    23.67    44.80 1.00    10172     3551
alpha[4]    44.06    44.12 6.00 6.02    34.16    53.94 1.00    11239     3404
alpha[5]    45.24    45.22 6.14 5.98    35.12    55.28 1.00     9868     3677
alpha[6]    45.90    45.94 6.28 6.40    35.50    56.26 1.00     8170     3693
alpha[7]    47.13    47.07 6.15 6.14    37.08    57.16 1.00    11322     3699
alpha[8]    44.73    44.72 6.20 6.06    34.27    54.84 1.00     9686     3068
alpha[9]    35.13    35.05 6.32 6.14    24.73    45.79 1.00    10701     3489

# showing 10 of 362 rows (change via 'max_rows' argument or 'cmdstanr_max_rows' option)
```

```
# mcmc_hist(fit1a$draws("alpha"), binwidth = 40)
```

c. Generar muestras de la distribución posterior predictiva. Superponiendo la serie de tiempo real para cada individuo sobre la gráfica de la distribución posterior predictiva, comentar sobre el ajuste del modelo a los datos.

d. Ajustar un modelo separado (α, β) para cada individuo en el conjunto de datos. Usar independientes iniciales normales separadas $\mathcal{N}(0, 250)$ para cada parámetro. De nuevo, usar 1000 muestras por cadena para cinco cadenas.

```
data_list <- list(
  N = length(sleep),
  rt_obs = sleep$Reaction,
  t = sleep$Days,
  subject = sleep$Subject
)

modelo_2d <- cmdstan_model("./Tarea5_2d.stan")
```

```
Warning in readLines(stan_file): incomplete final line found on
'./Tarea5_2d.stan'
```

```
print(modelo_2a)
```

```
data {
  int<lower=0> N;      // Numero de datos
  vector<lower=0>[N] rt_obs;    // R(t), tiempos de reacción observados
  vector<lower=0>[N] t;         // días sin dormir
  array[N] int subject;     // i-ésima persona
}

parameters {
  vector[N] alpha;
 /* vector[N] beta; */
  real<lower=0> sigma;
}

transformed parameters {
  vector[N] media;
  for (i in 1:N){
    media[i] = alpha[subject[i]] + alpha[subject[i]] * t[i]; //modelo dado
  }
}


model {
  for (i in 1:N){
    rt_obs[i] ~ normal(media[i], sigma); //modelo dado
  }
  alpha ~ normal(0, 250);  // distribucion dada
 // beta ~ normal(0, 250);   // distribucion dada
}
```

e. Calcular los estimados de las medias posteriores de los parámetros *beta* para el modelo de parámetros heterogéneos. £Cómo se compara esto al estimador *beta* obtenido del modelo homogéneo?

f. Generar muestras de la distribución predictiva posterior. Comparando los datos individuales de cada sujeto las muestras predictivas, comentar sobre el ajuste del nuevo modelo.

g. Particionar los datos en dos subconjuntos: un conjunto de entrenamiento (sujetos 1-17) y un conjunto de prueba (sujeto 18). Ajustando ambos modelos heterogéneo y homogéneo con los datos de entrenamiento, calcular el desempeño de cada modelo para predecir el conjunto de prueba.

h. Alternativamente, se puede ajustar un modelo jerárquico a los datos que (esperamos) capture algunos de los mejores elementos de cada uno de los modelos previos. Ajustar esta tal modelo usando normales iniciales para α_i y β_i y distribuciones iniciales para los hiperparámetros de estas distribuciones.