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)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
```

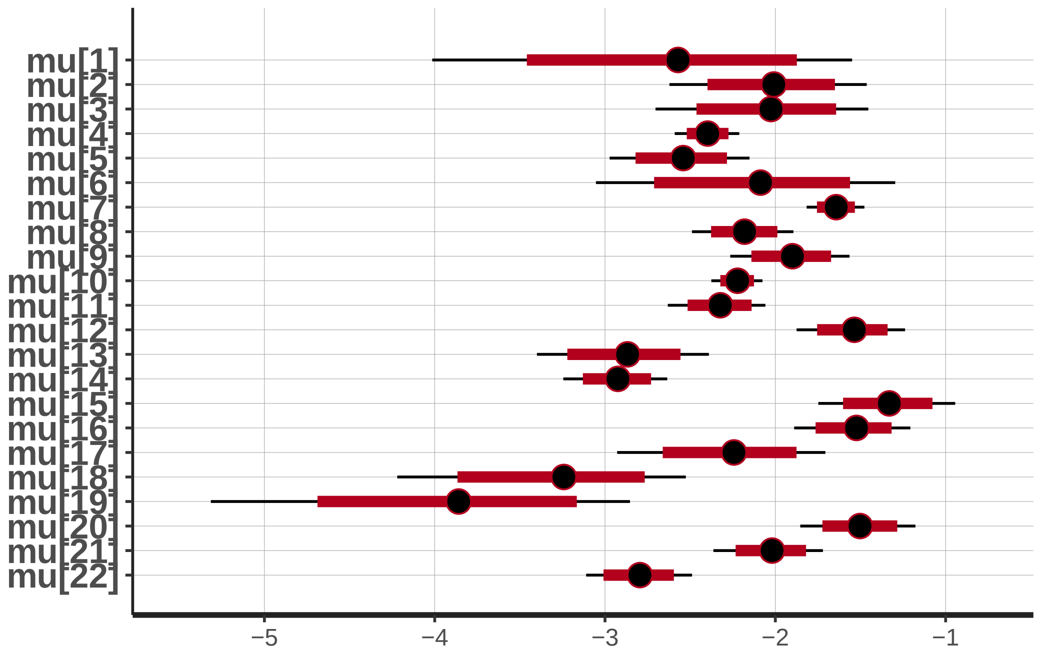
```
Chain 1:
Chain 1: Gradient evaluation took 1.9e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%]  (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: 0.181 seconds (Warm-up)
Chain 1:                0.116 seconds (Sampling)
Chain 1:                0.297 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 / 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: 0.186 seconds (Warm-up)
Chain 2:                0.195 seconds (Sampling)
Chain 2:                0.381 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 / 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: 0.181 seconds (Warm-up)
Chain 3:                0.147 seconds (Sampling)
Chain 3:                0.328 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: 0.183 seconds (Warm-up)
Chain 4:                0.157 seconds (Sampling)
Chain 4:                0.34 seconds (Total)
Chain 4:
```

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

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

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

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

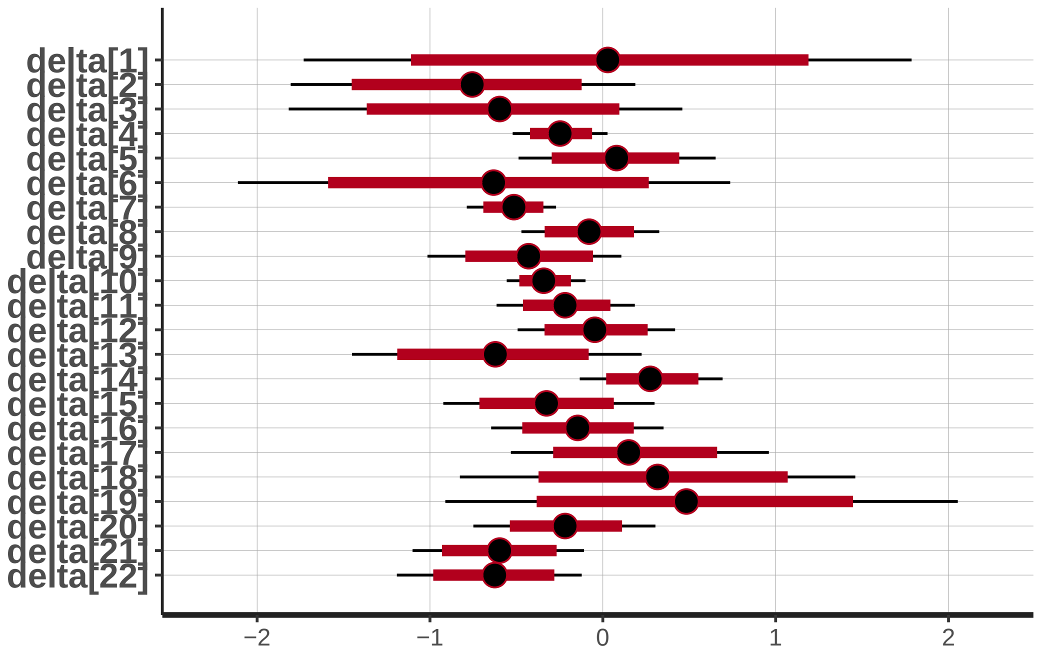


A continuación presentamos una gráfica para observar el rango de valores para cada δ_i

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

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

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

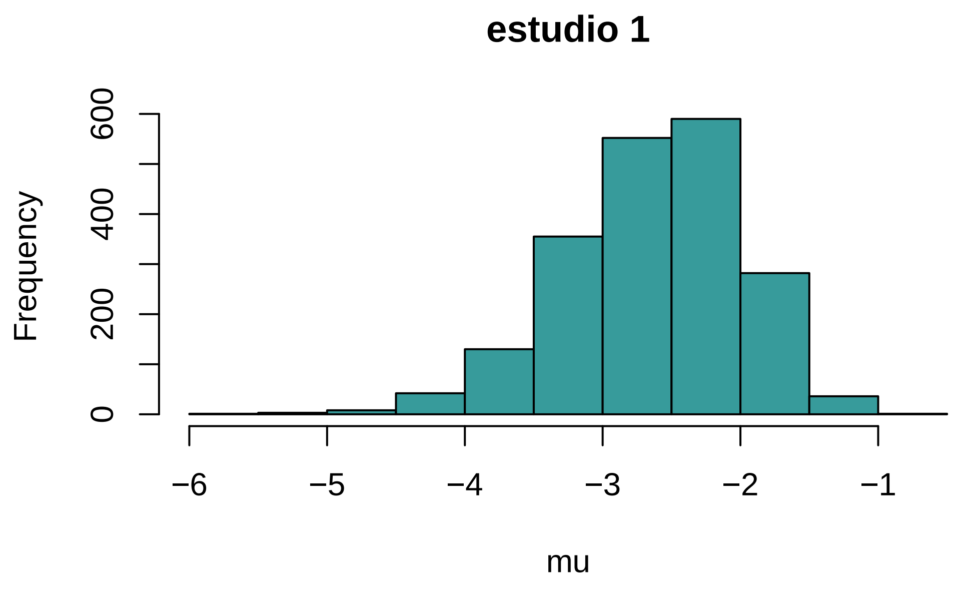


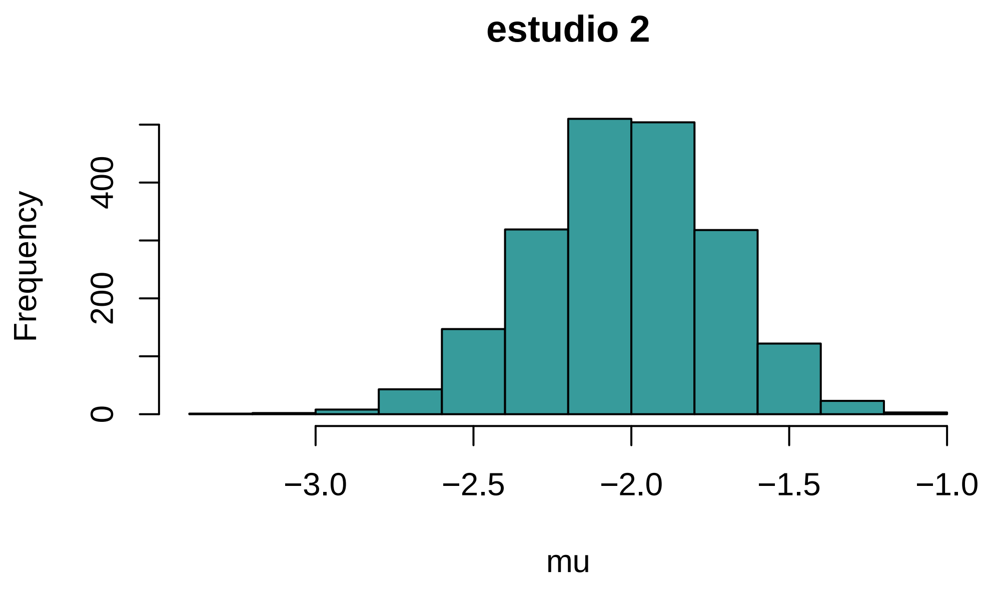
FInalmente, presentamos los histogramas para las distribuciones posteriores para cada μ_i y δ_i

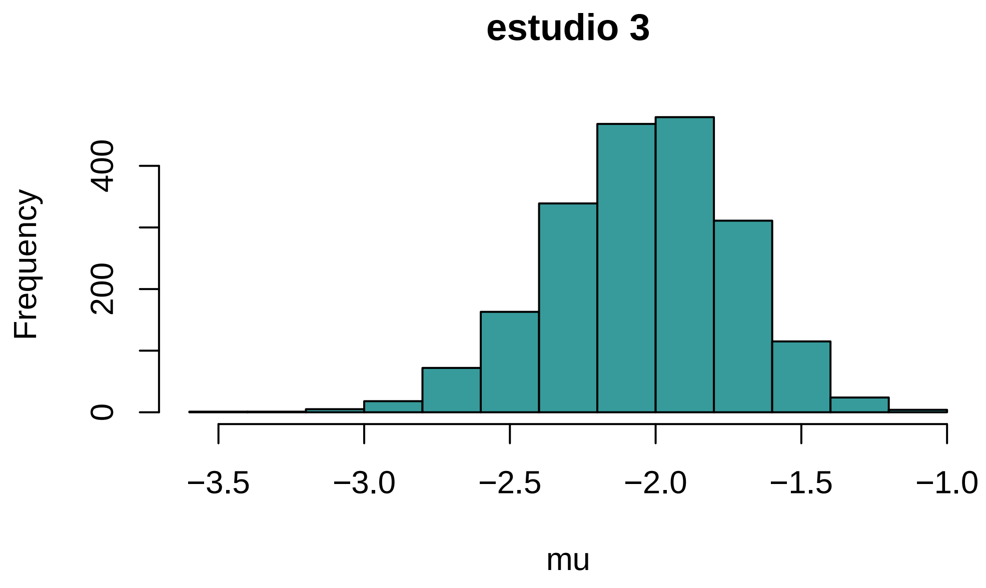
```r
# Extraer muestras de la posterior para mu y delta
posterior_samples <- rstan::extract(stan_samples)

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

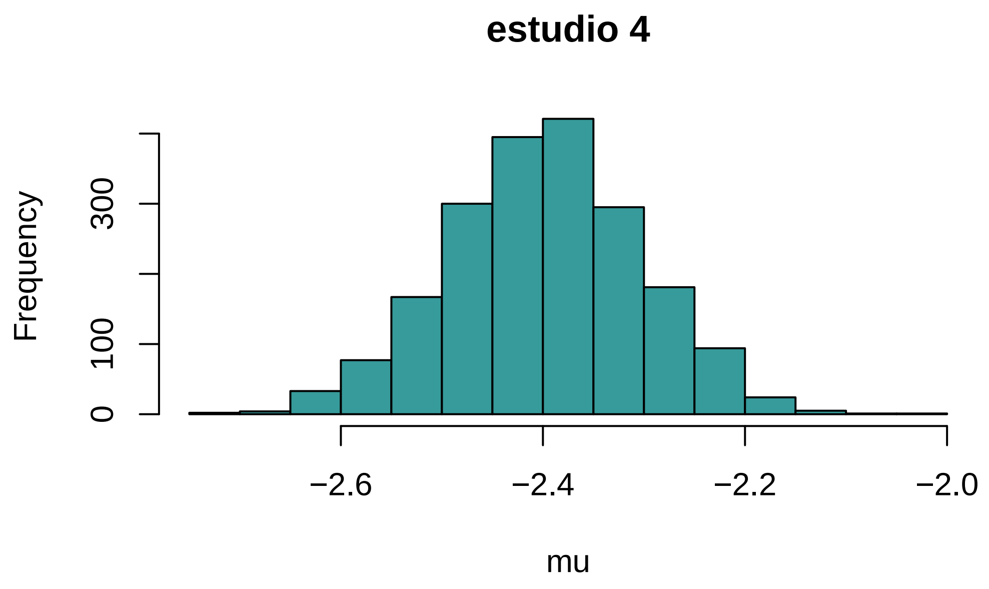
Histogramas de las posteriores de μ_i

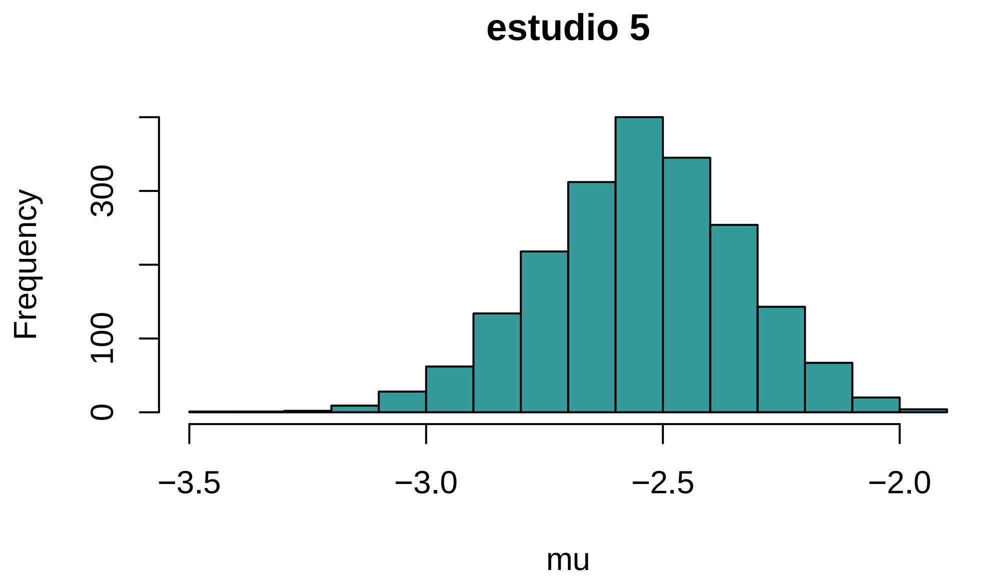
```r
# Graficar las distribuciones posteriores de mu y delta para cada estudio

# 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")

}
```

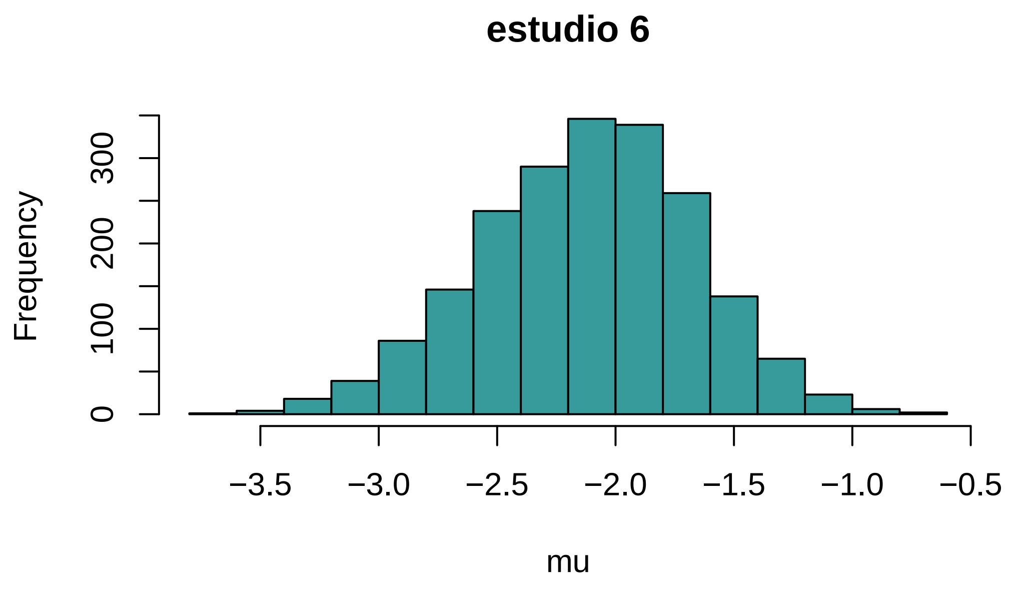


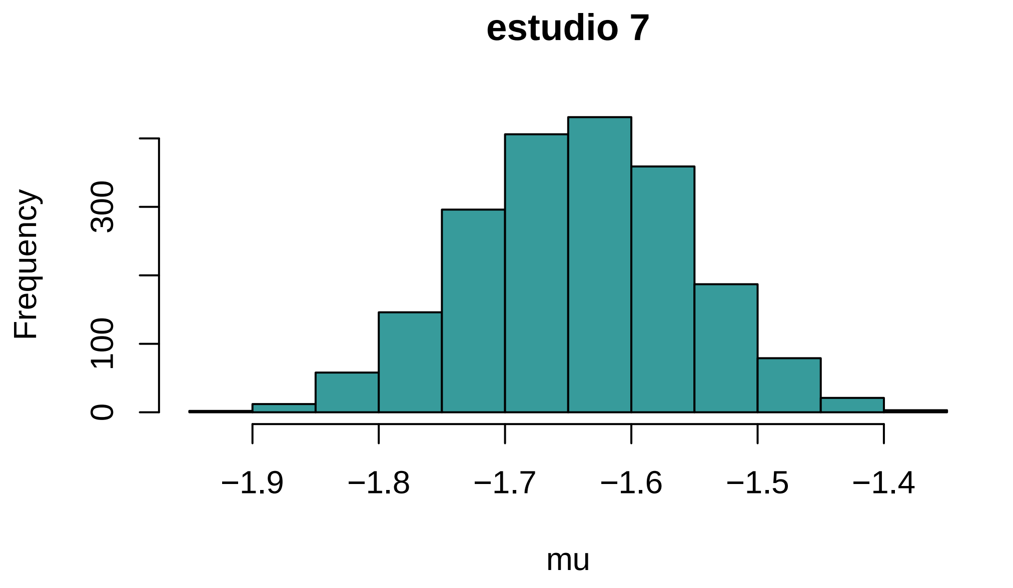

estudio 2
Frequency
0
200
400
−3.0
−2.5
−2.0
−1.5
−1.0
mu

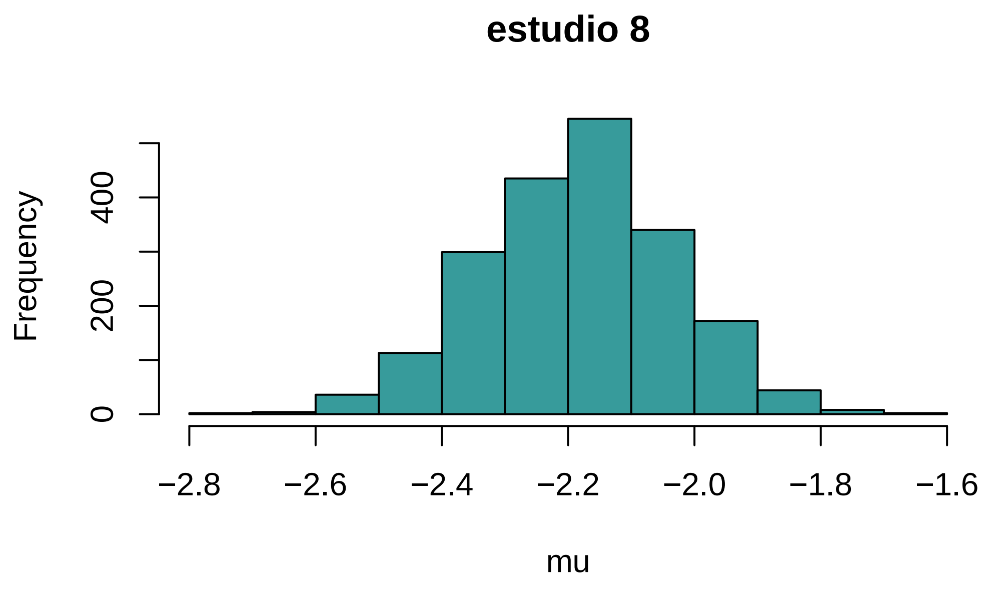

estudio 3
Frequency
0
200
400
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
mu

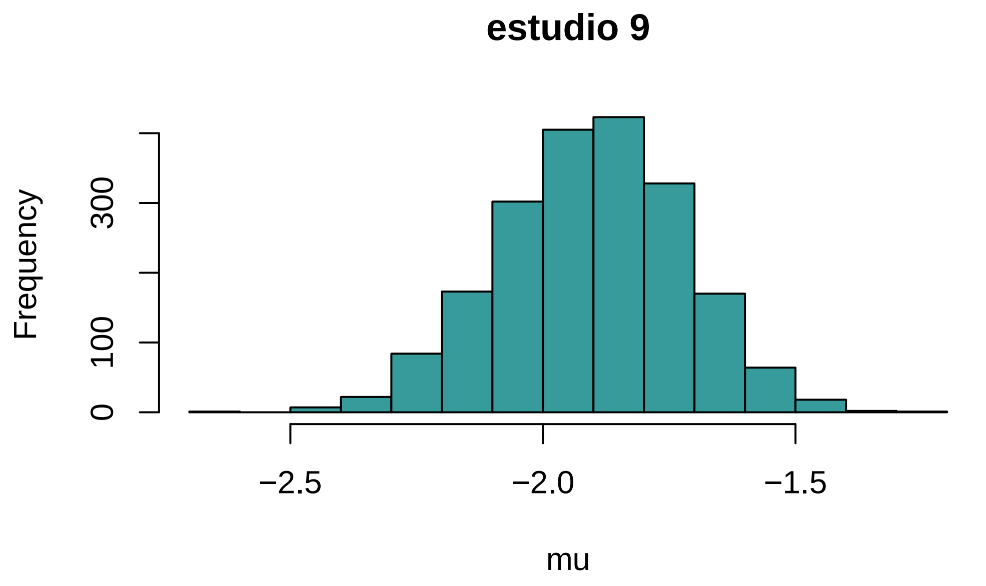

estudio 4
Frequency
0
100
300
−2.6
−2.4
−2.2
−2.0
mu

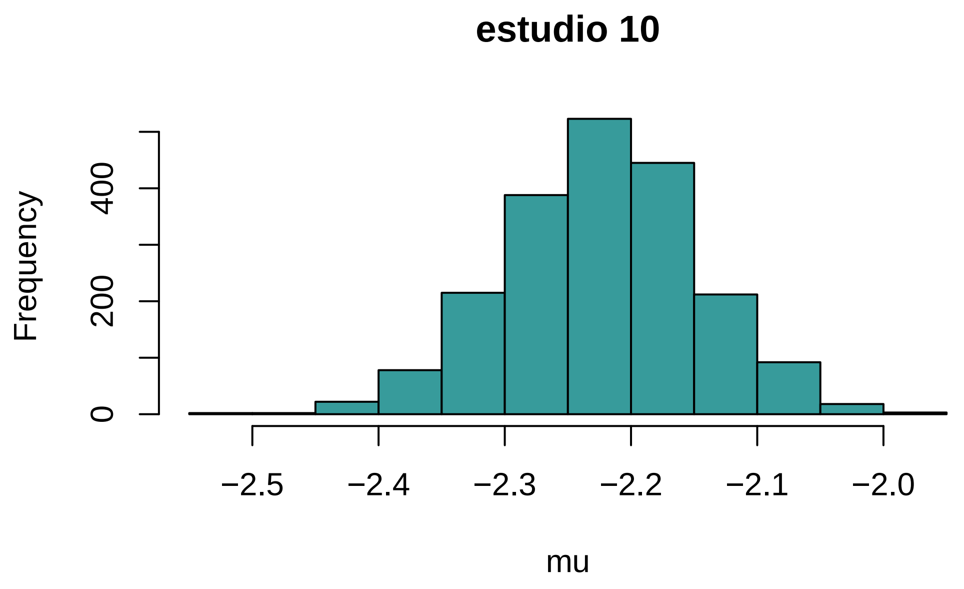

estudio 5
Frequency
0
100
300
−3.5
−3.0
−2.5
−2.0
mu

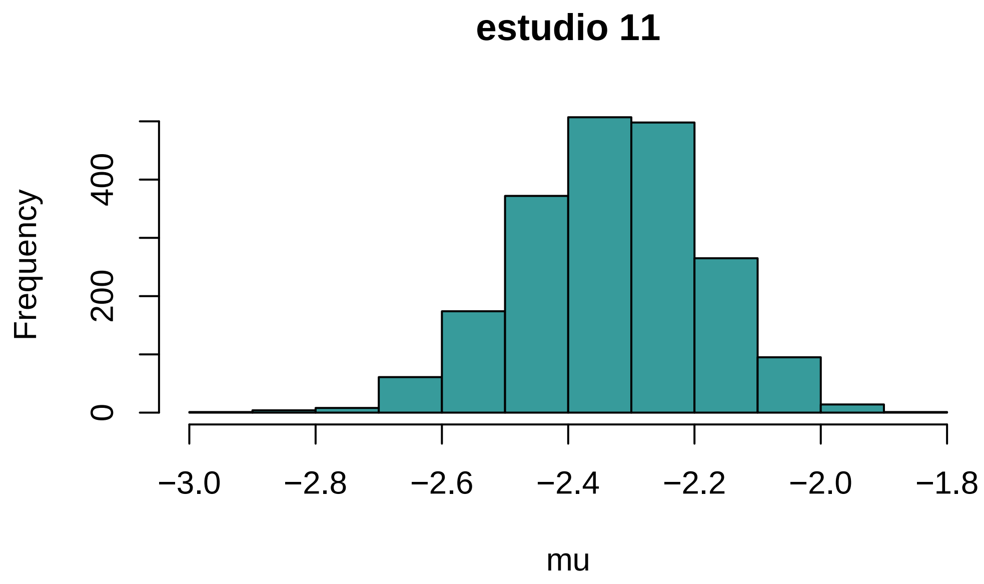

estudio 6
Frequency
0
100
200
300
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
mu

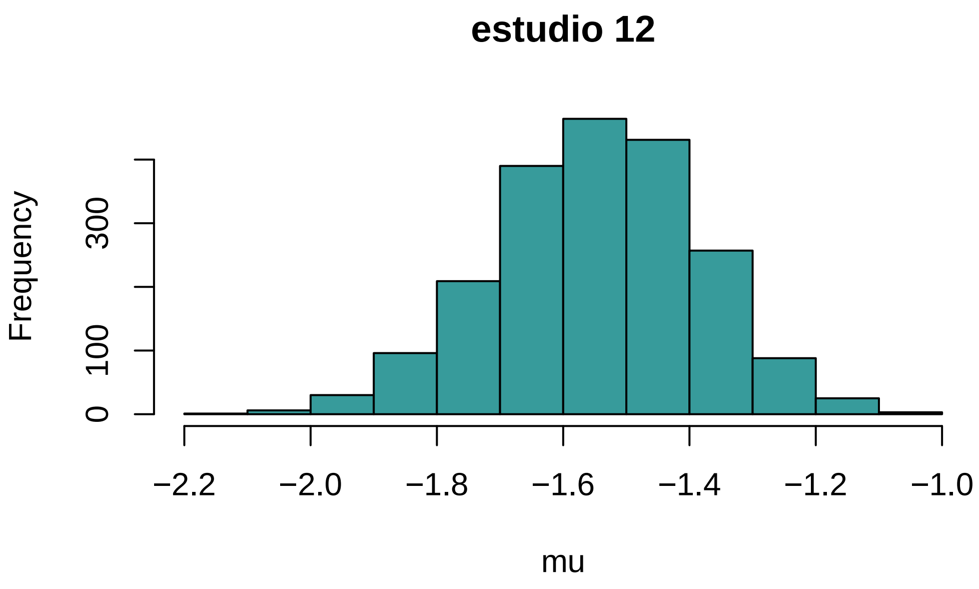

estudio 7
Frequency
0
100
300
−1.9
−1.8
−1.7
−1.6
−1.5
−1.4
mu

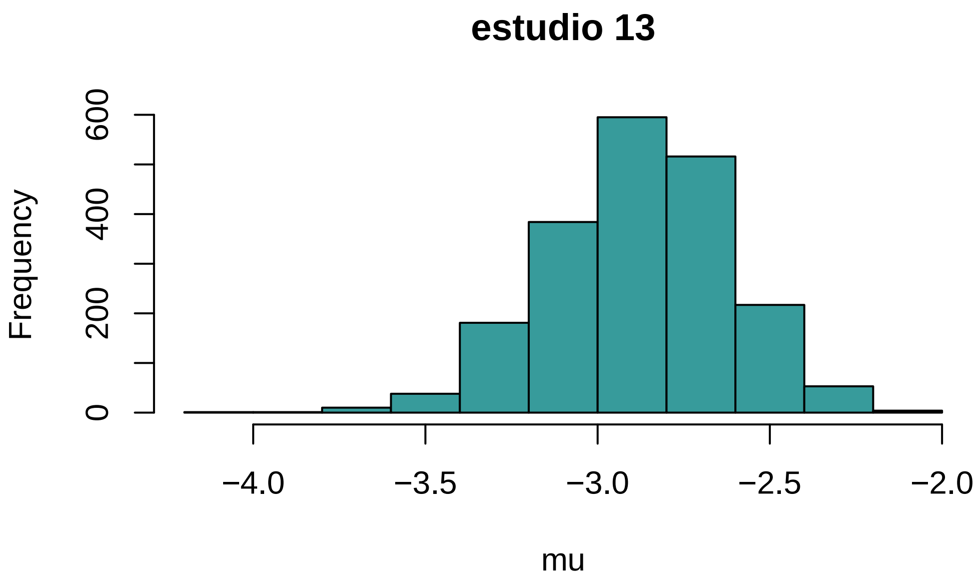

estudio 8
Frequency
0
200
400
−2.8
−2.6
−2.4
−2.2
−2.0
−1.8
−1.6
mu

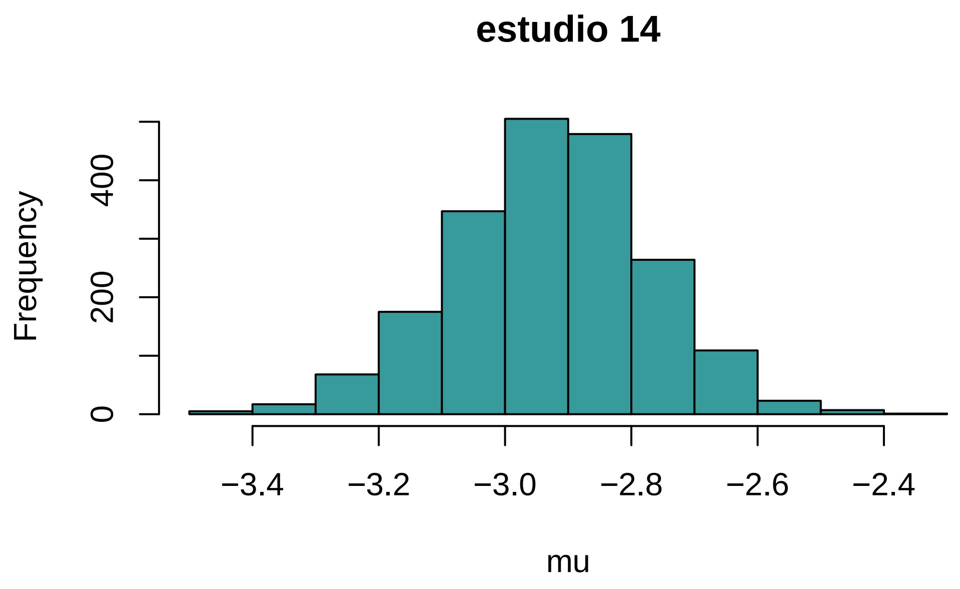

estudio 9
Frequency
0
100
300
−2.5
−2.0
−1.5
mu

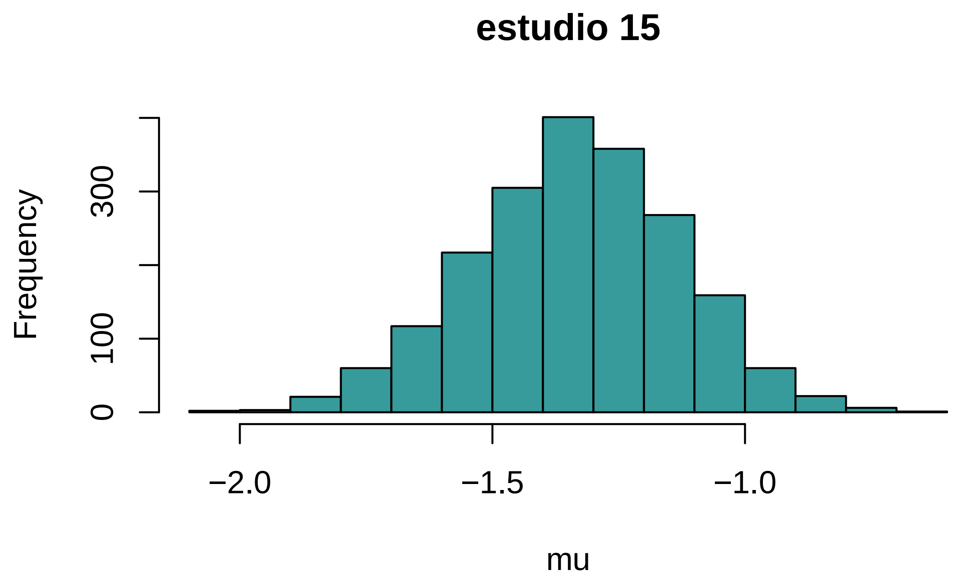

estudio 10
Frequency
0
200
400
−2.5
−2.4
−2.3
−2.2
−2.1
−2.0
mu

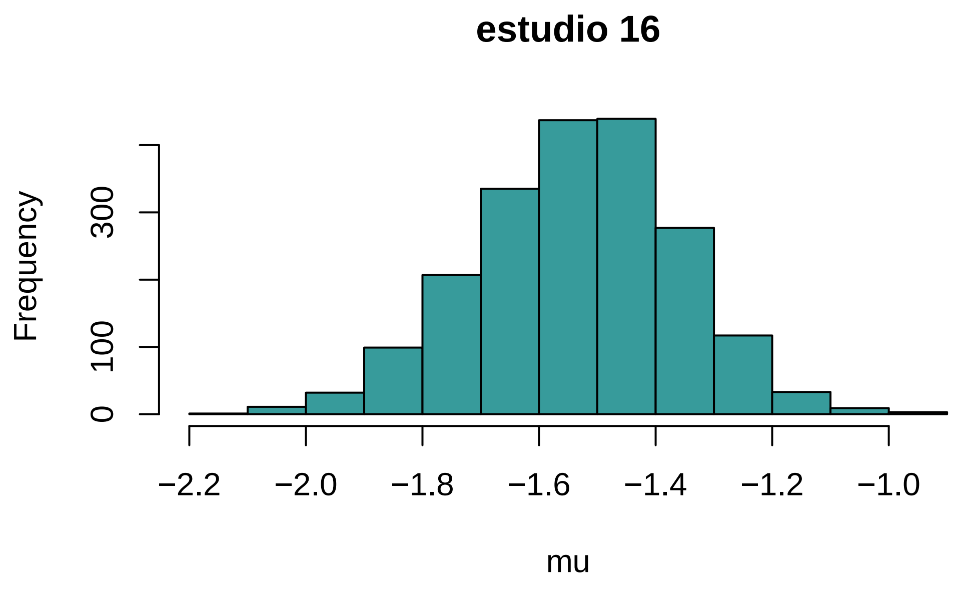

estudio 11
Frequency
0
200
400
−3.0
−2.8
−2.6
−2.4
−2.2
−2.0
−1.8
mu

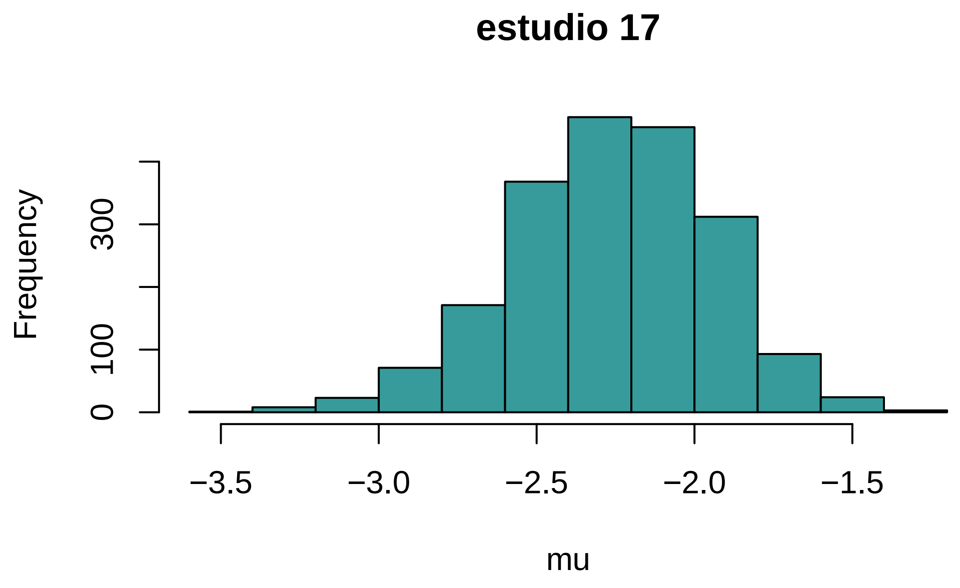

estudio 12
Frequency
0
100
300
−2.2
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

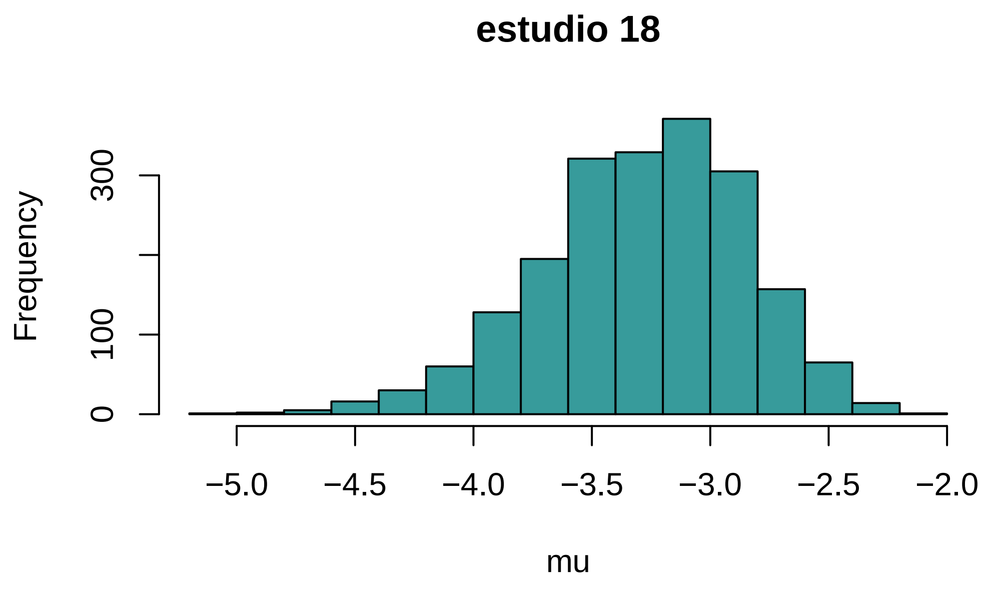

estudio 13
Frequency
0
200
400
600
−4.0
−3.5
−3.0
−2.5
−2.0
mu

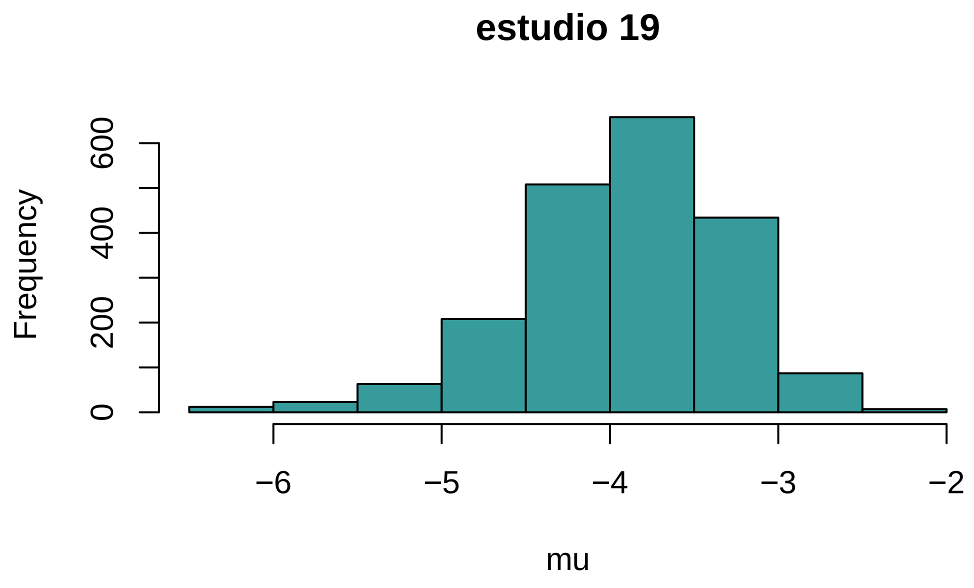

estudio 14
Frequency
0
200
400
−3.4
−3.2
−3.0
−2.8
−2.6
−2.4
mu

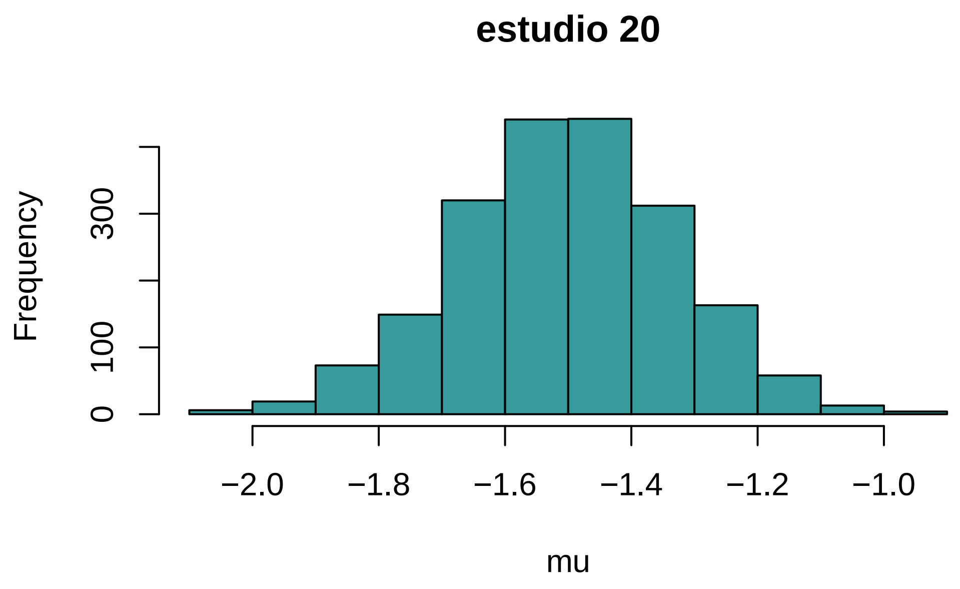

estudio 15
Frequency
0
100
300
−2.0
−1.5
−1.0
mu

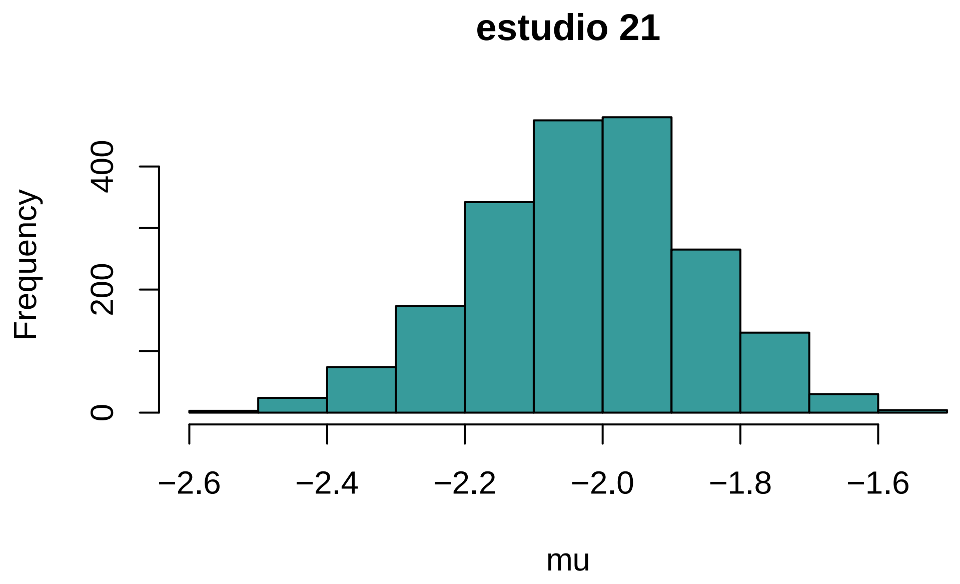

estudio 16
Frequency
0
100
300
−2.2
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

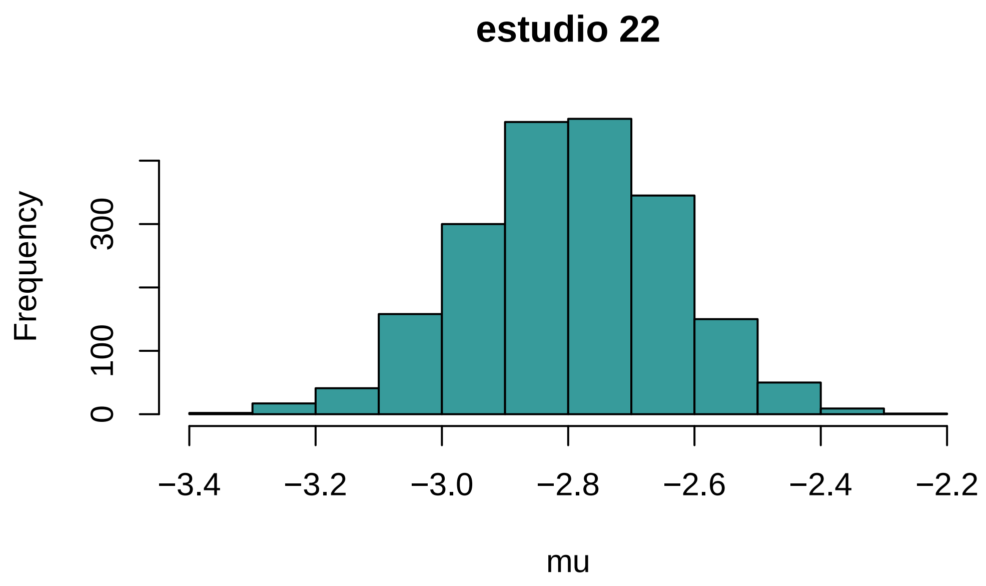

estudio 17
Frequency
0
100
300
−3.5
−3.0
−2.5
−2.0
−1.5
mu


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
−6
−5
−4
−3
−2
mu

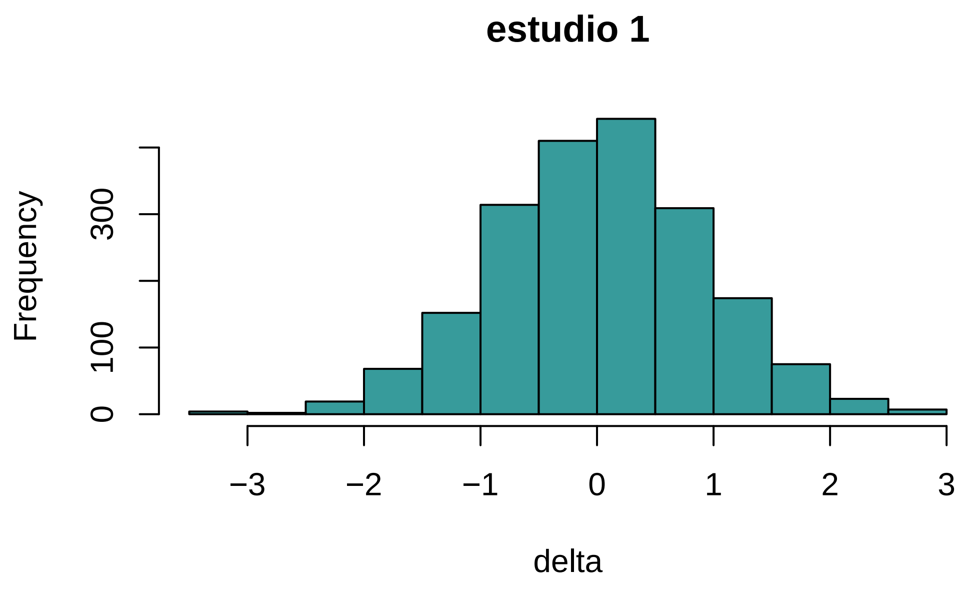

estudio 20
Frequency
0
100
300
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

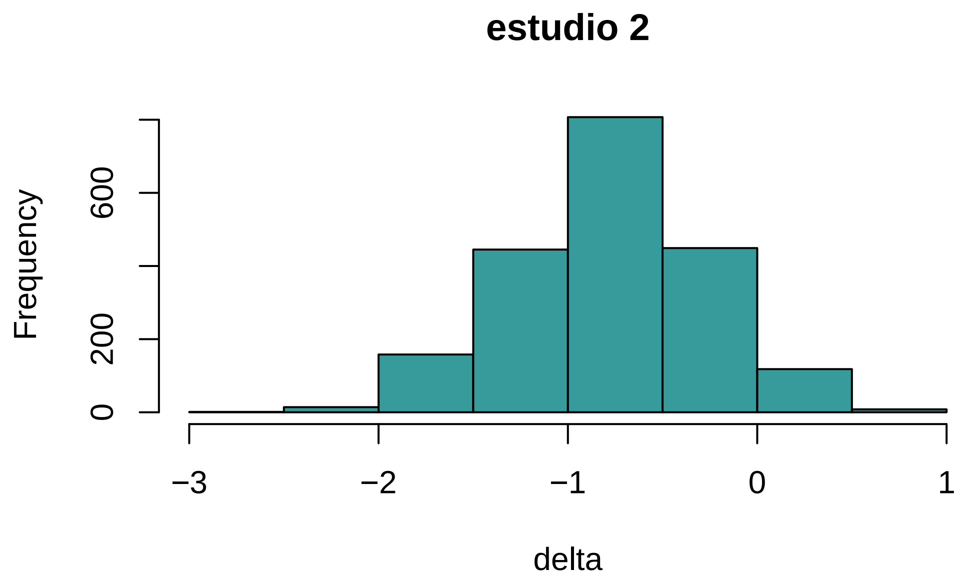

estudio 21
Frequency
0
200
400
−2.6
−2.4
−2.2
−2.0
−1.8
−1.6
mu

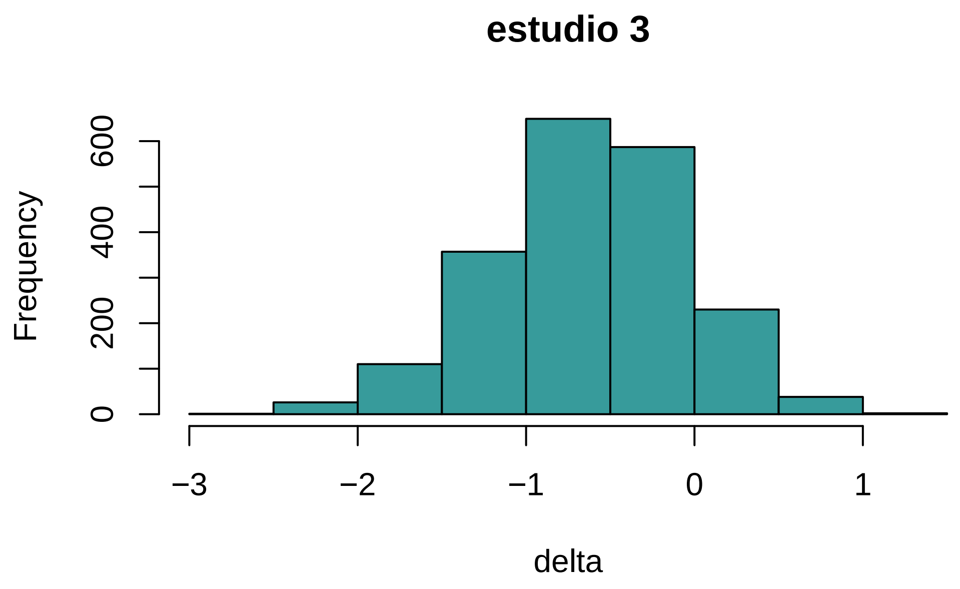


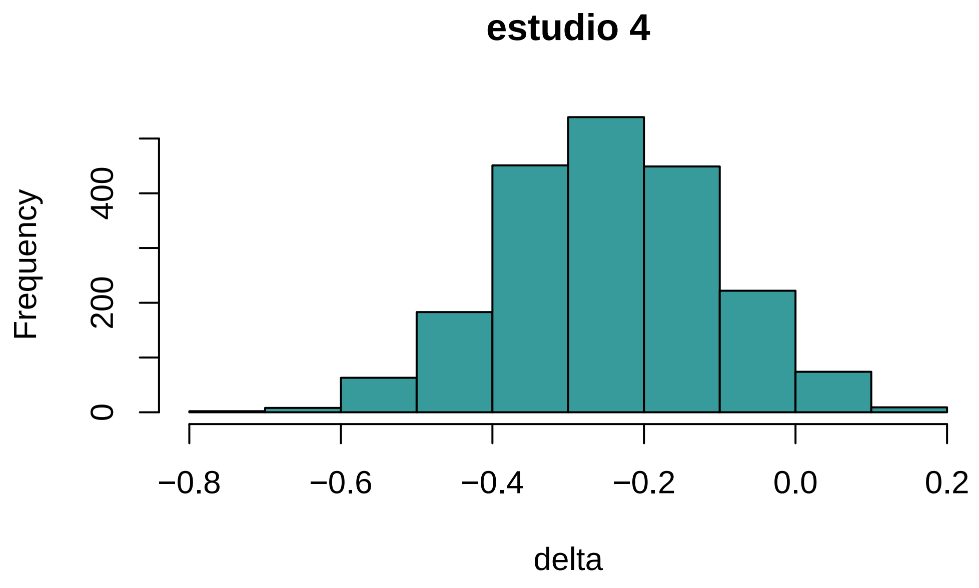
Histogramas de las posteriores de δ_i

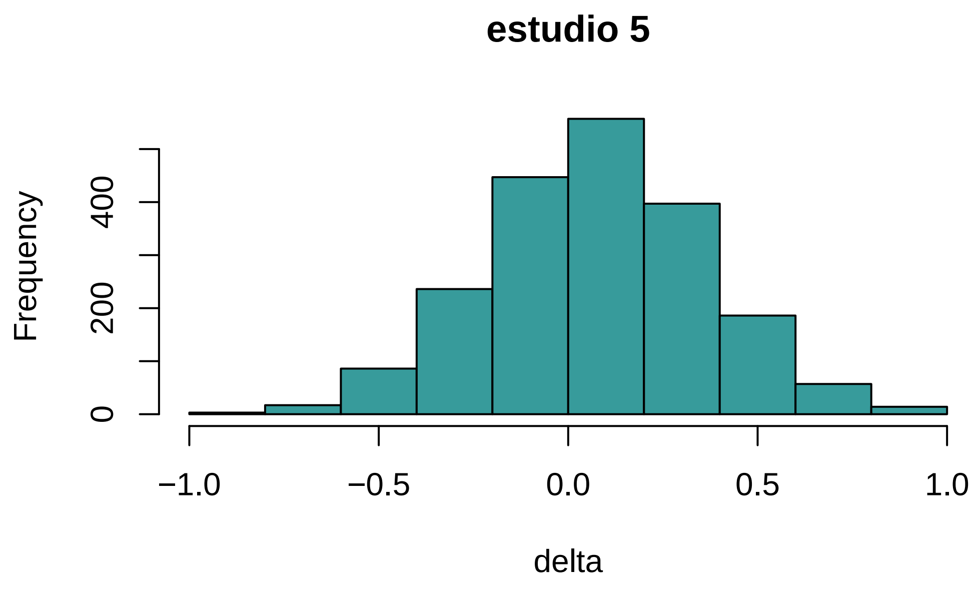
```
# par(mfrow = c(5, 5))

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")

}
```

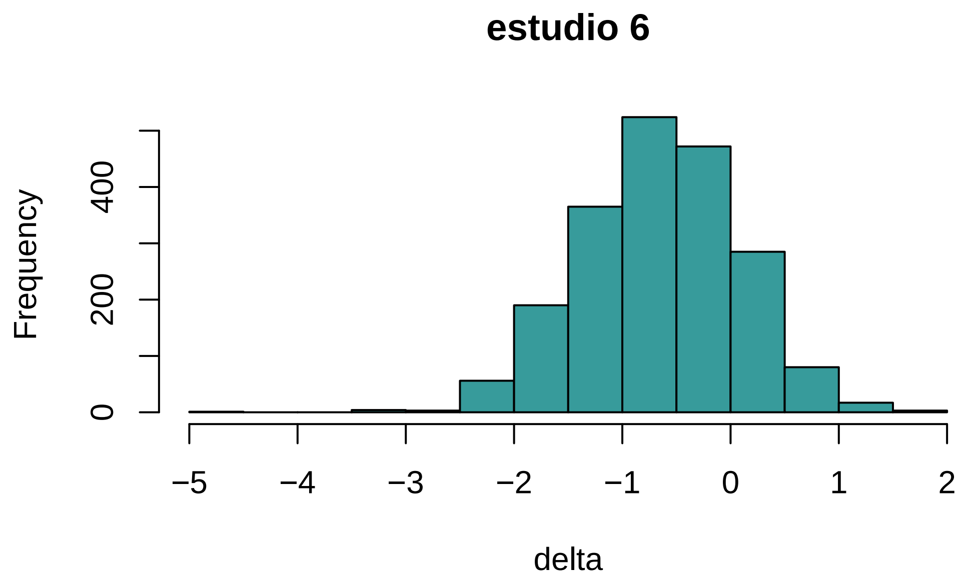

estudio 1
Frequency
0
100
300
−3
−2
−1
0
1
2
3
delta

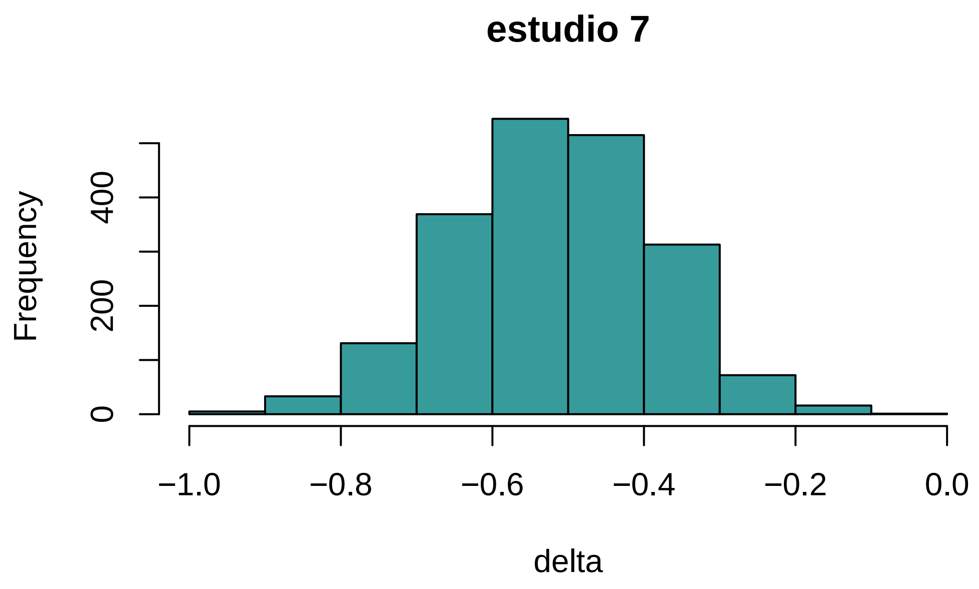

estudio 2
Frequency
0
200
600
−3
−2
−1
0
1
delta


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

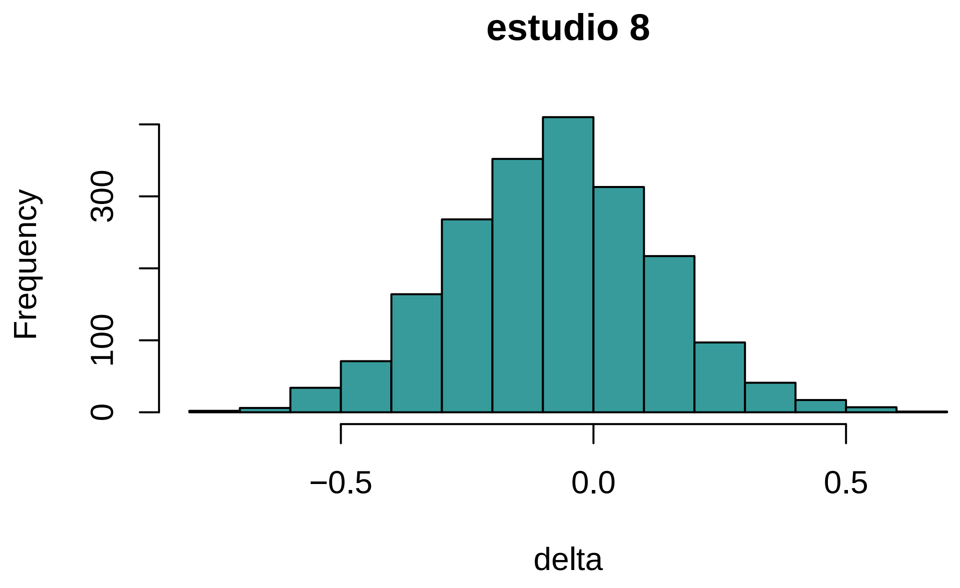

estudio 4
Frequency
0
200
400
−0.8
−0.6
−0.4
−0.2
0.0
0.2
delta


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

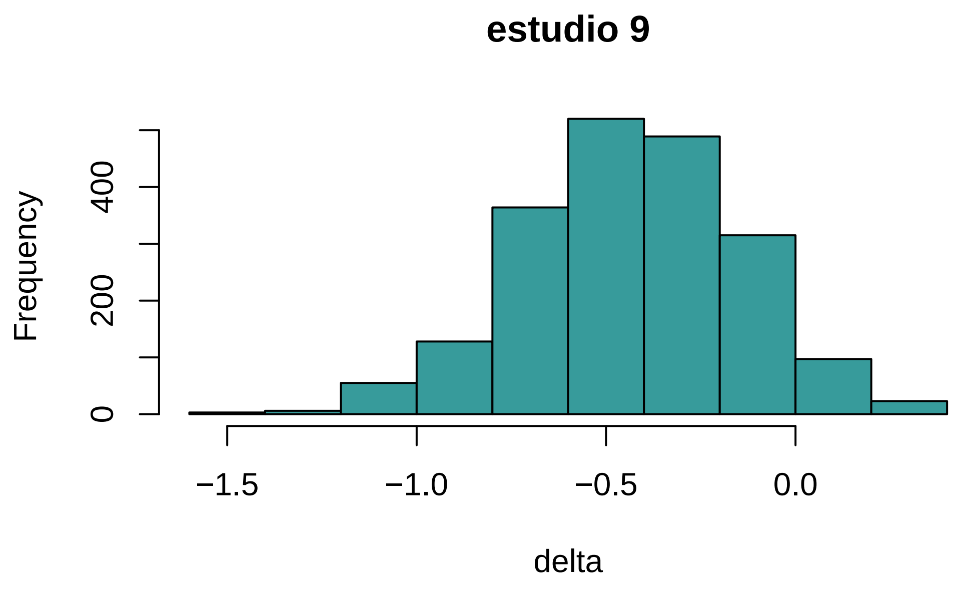

estudio 6
Frequency
0
200
400
−5
−4
−3
−2
−1
0
1
2
delta

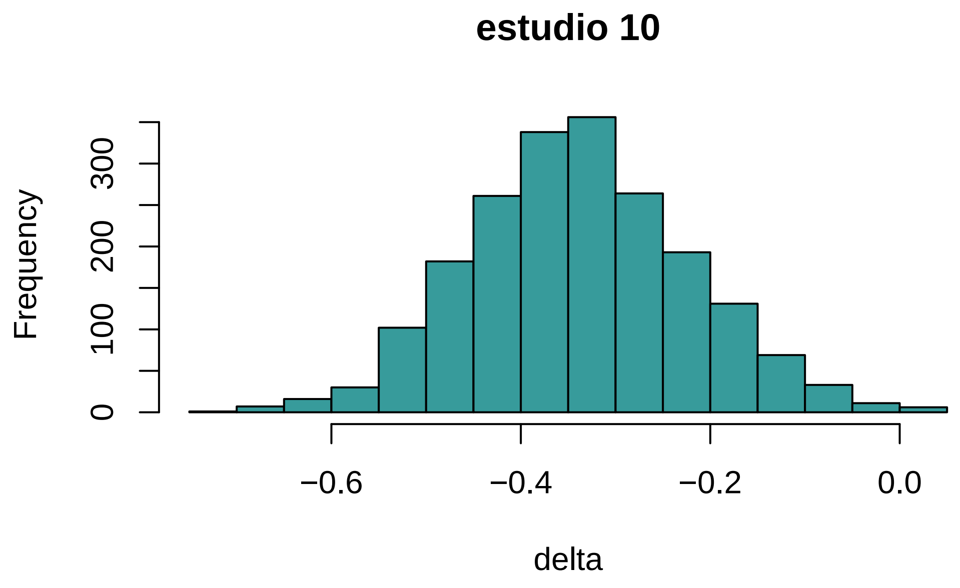


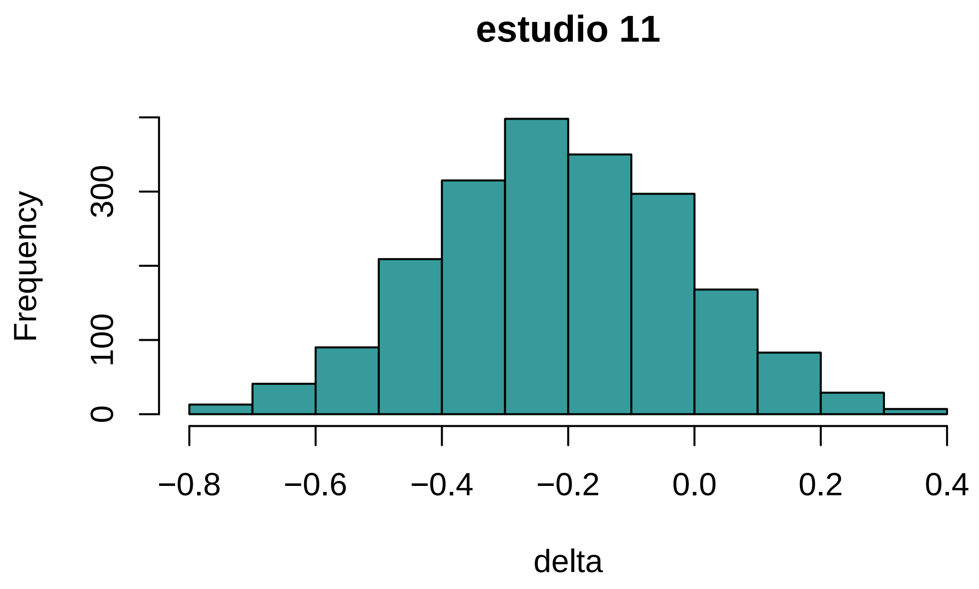
estudio 7
Frequency
0
200
400
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
delta

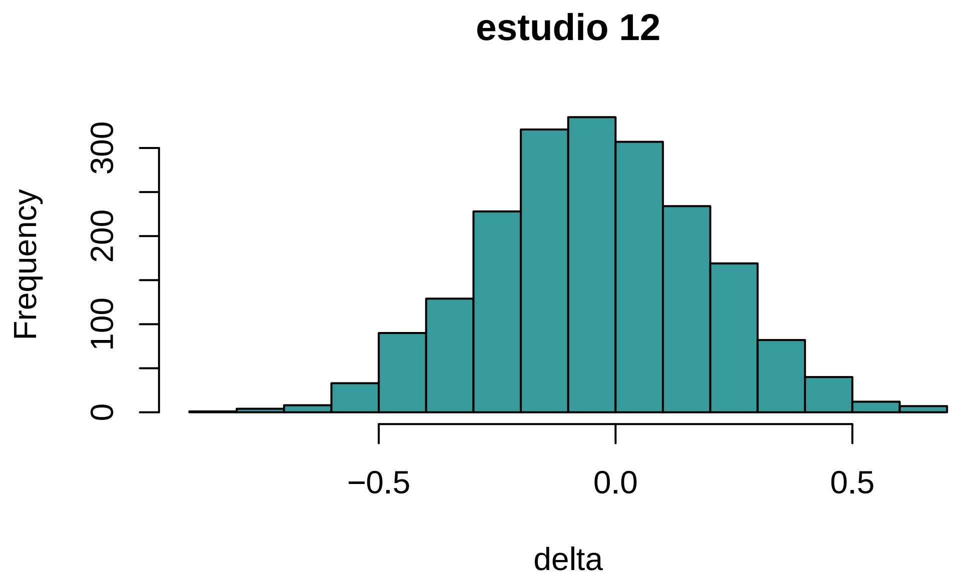


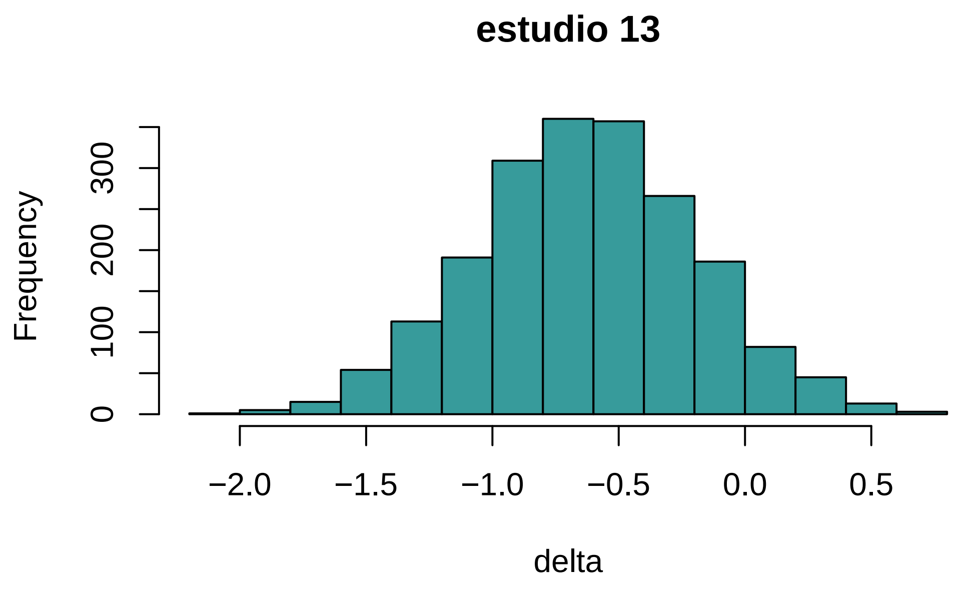
estudio 8
Frequency
0
100
300
−0.5
0.0
0.5
delta

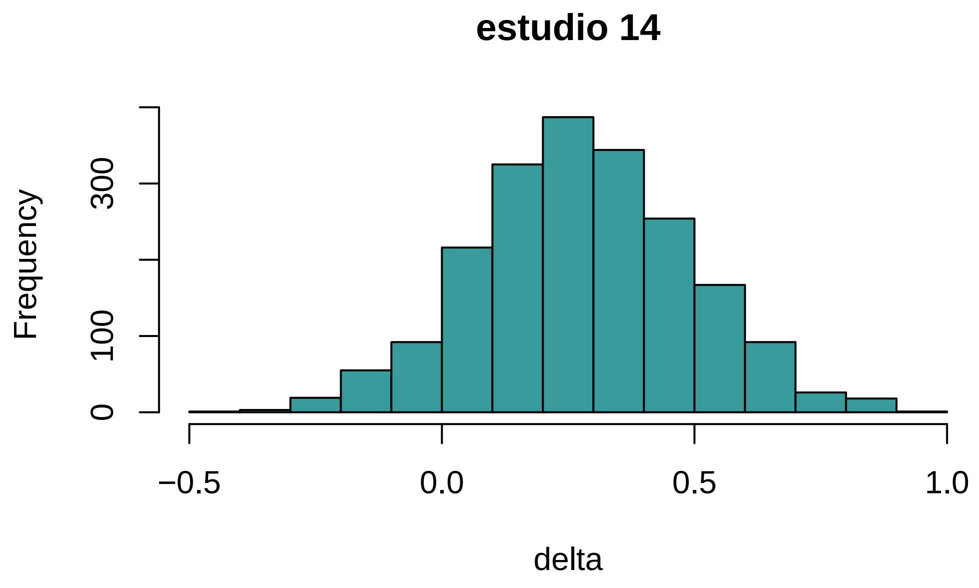

estudio 9
Frequency
0
200
400
−1.5
−1.0
−0.5
0.0
delta

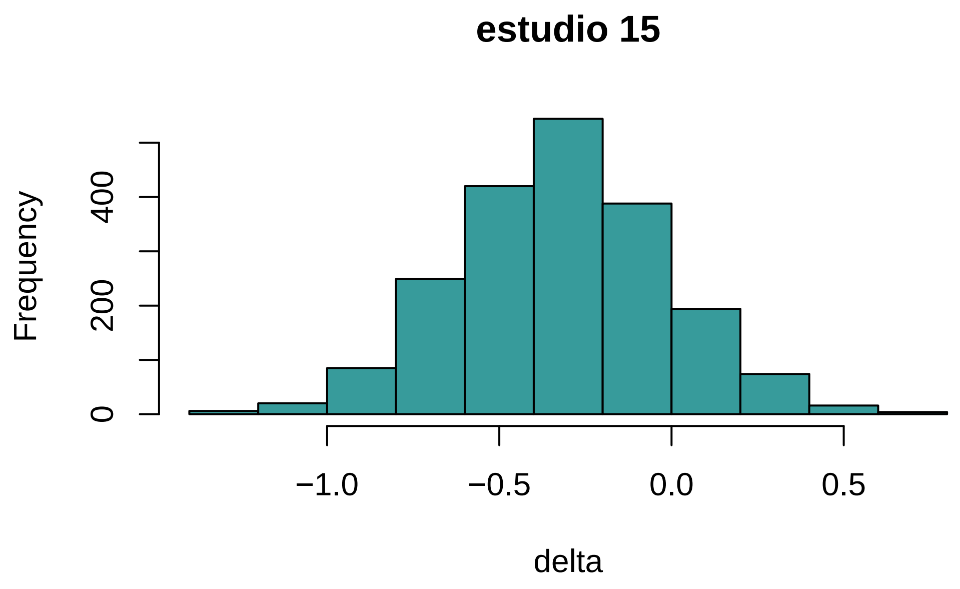

estudio 10
Frequency
0
100
200
300
−0.6
−0.4
−0.2
0.0
delta

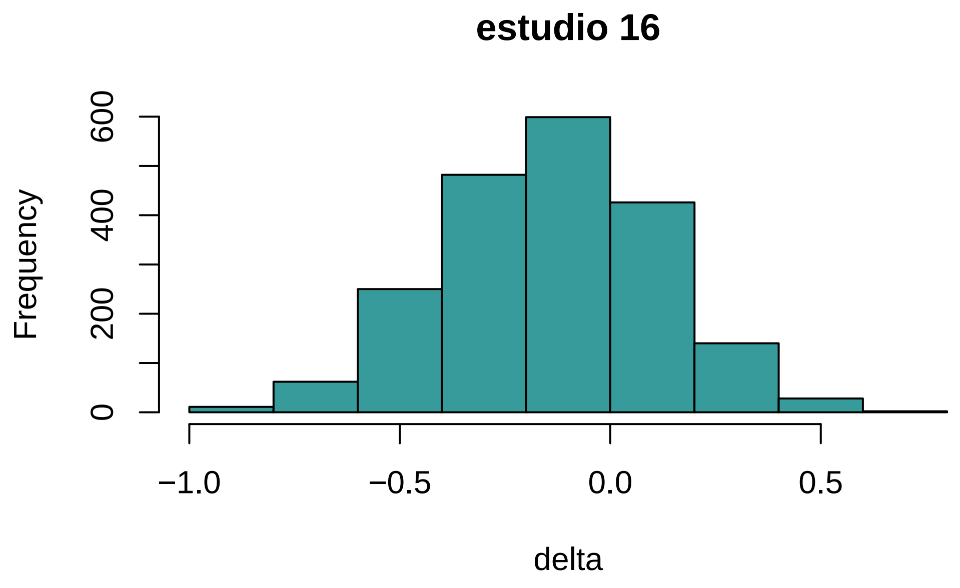

estudio 11
Frequency
0
100
300
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
delta

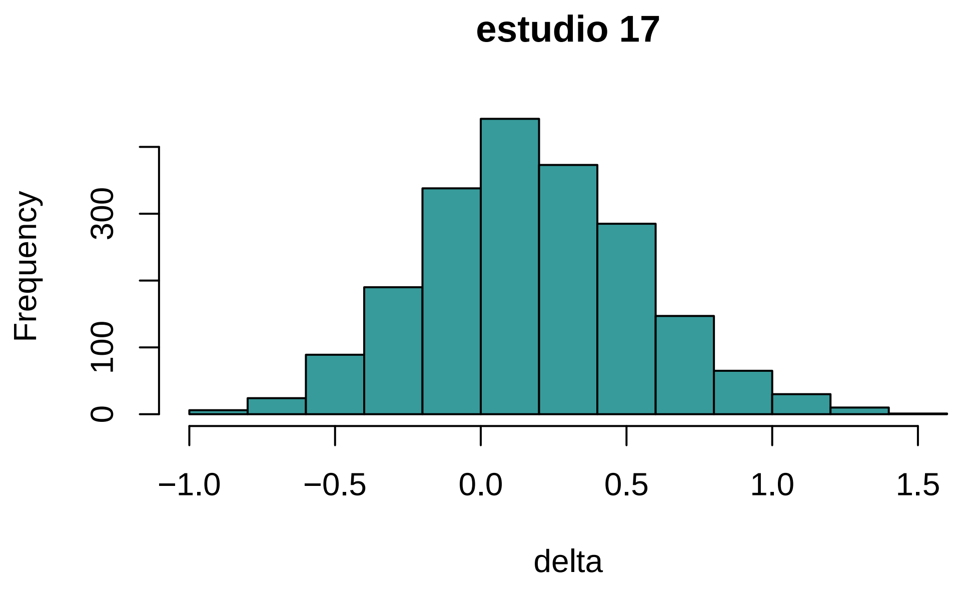

estudio 12
Frequency
0
100
200
300
−0.5
0.0
0.5
delta

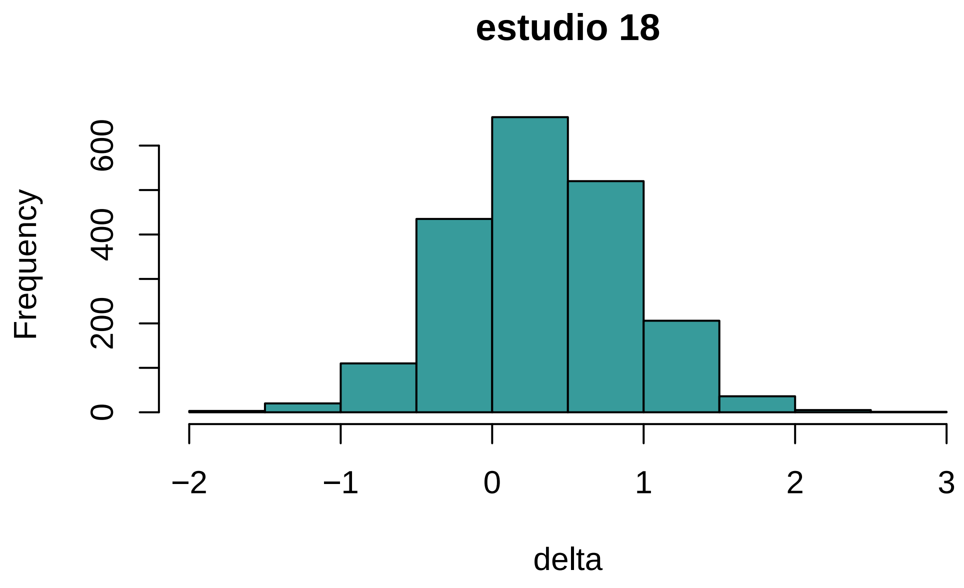

estudio 13
Frequency
0
100
200
300
−2.0
−1.5
−1.0
−0.5
0.0
0.5
delta

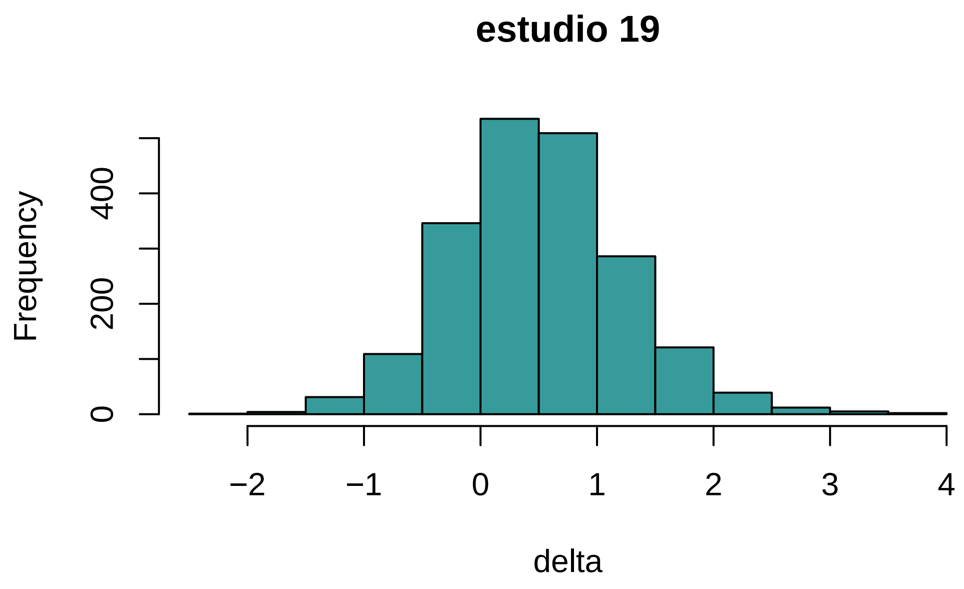

estudio 14
Frequency
0
100
300
−0.5
0.0
0.5
1.0
delta

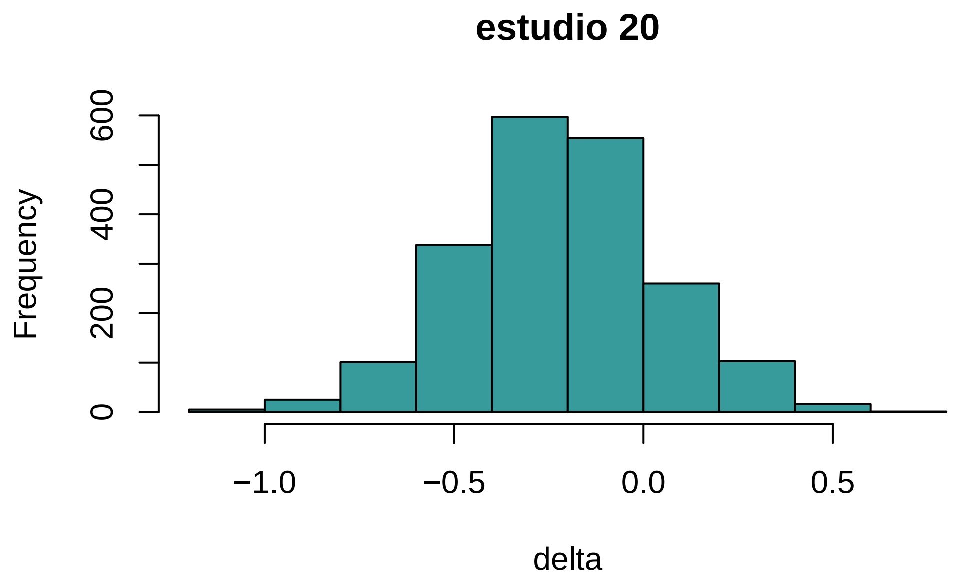

estudio 15
Frequency
0
200
400
−1.0
−0.5
0.0
0.5
delta

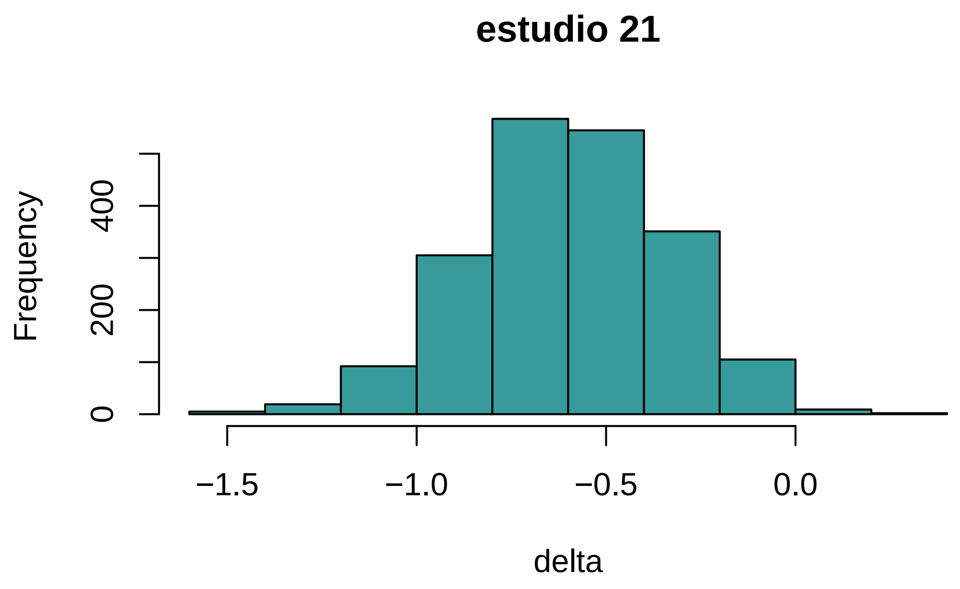

estudio 16
Frequency
0
200
400
600
−1.0
−0.5
0.0
0.5
delta

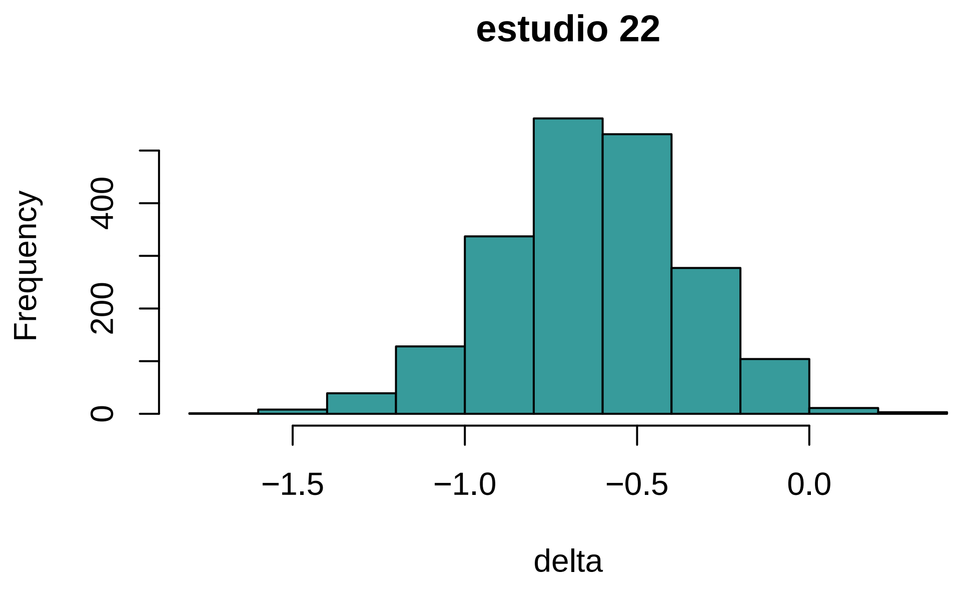

estudio 17
Frequency
0
100
300
−1.0
−0.5
0.0
0.5
1.0
1.5
delta


estudio 18
Frequency
0
200
400
600
−2
−1
0
1
2
3
delta


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


estudio 20
Frequency
0
200
400
600
−1.0
−0.5
0.0
0.5
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)
```

```
# 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 1.3e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.13 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: 1.02 seconds (Warm-up)
Chain 1:                0.95 seconds (Sampling)
Chain 1:                1.97 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 / 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: 1.693 seconds (Warm-up)
Chain 2:                1.473 seconds (Sampling)
Chain 2:                3.166 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 / 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: 1.841 seconds (Warm-up)
Chain 3:                1.832 seconds (Sampling)
Chain 3:                3.673 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: 1.352 seconds (Warm-up)
Chain 4:                1.773 seconds (Sampling)
Chain 4:                3.125 seconds (Total)
Chain 4:

Warning: There were 333 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: There were 1 chains where the estimated Bayesian Fraction of Missing Information was
https://mc-stan.org/misc/warnings.html#bfmi-low
```

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

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

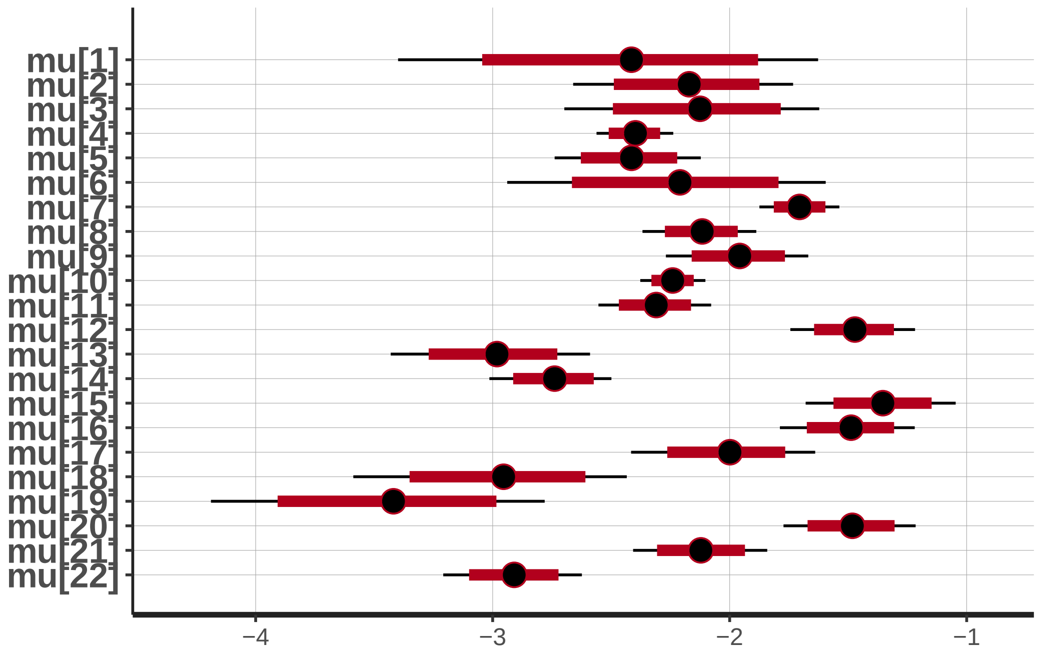
```
Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and ta
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#tail-ess
```

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

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

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

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

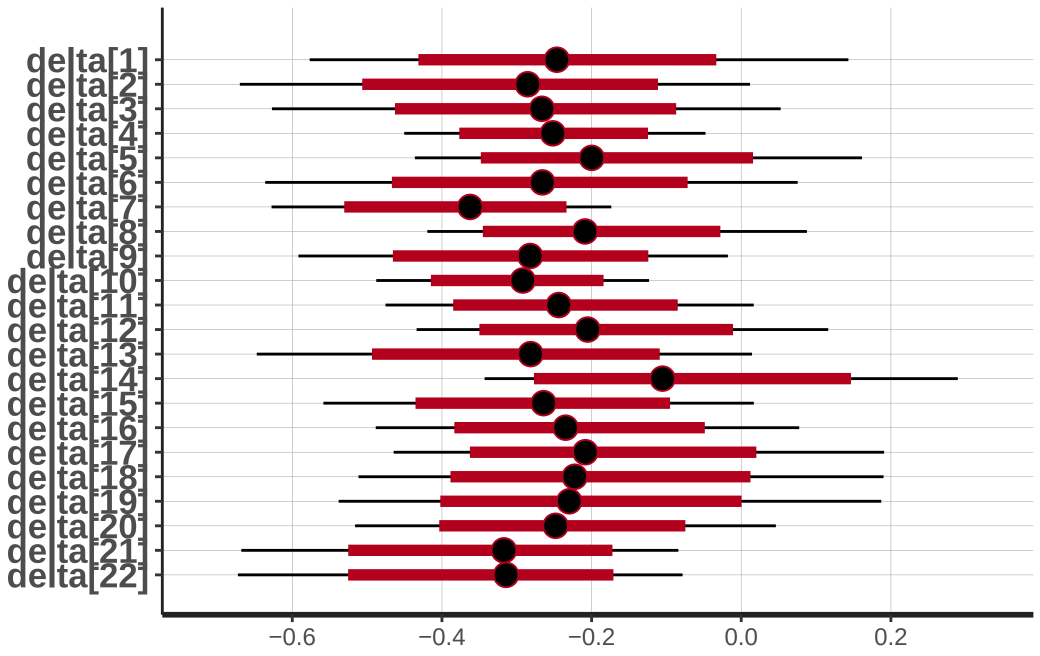


A continuación presentamos una gráfica para observar el rango de valores para cada δ

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

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

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

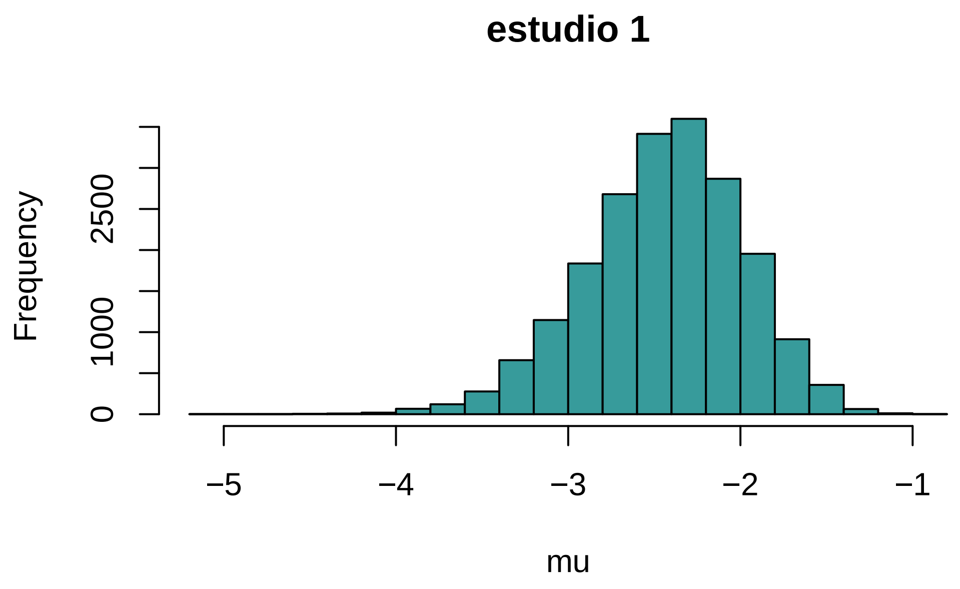


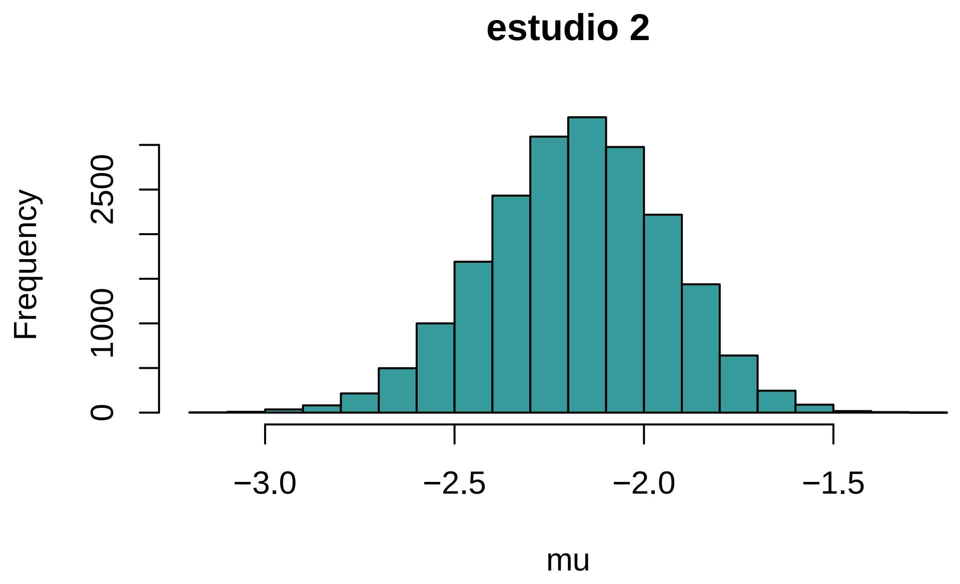
FInalmente, presentamos los histogramas para las distribuciones posteriores para cada μ_i y δ_i

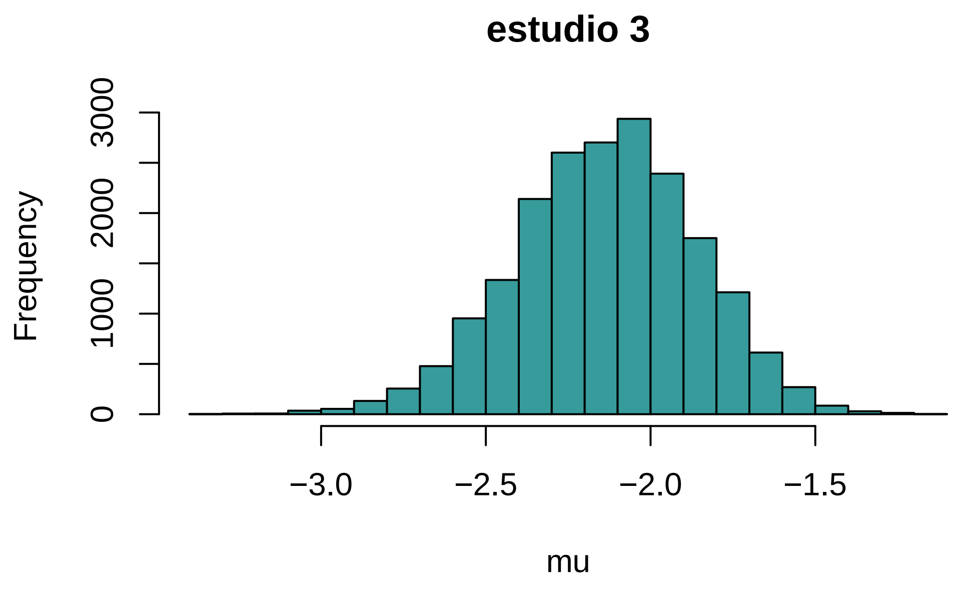
```r
# Extraer muestras de la posterior para mu y delta
posterior_samples <- rstan::extract(stan_samples)

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

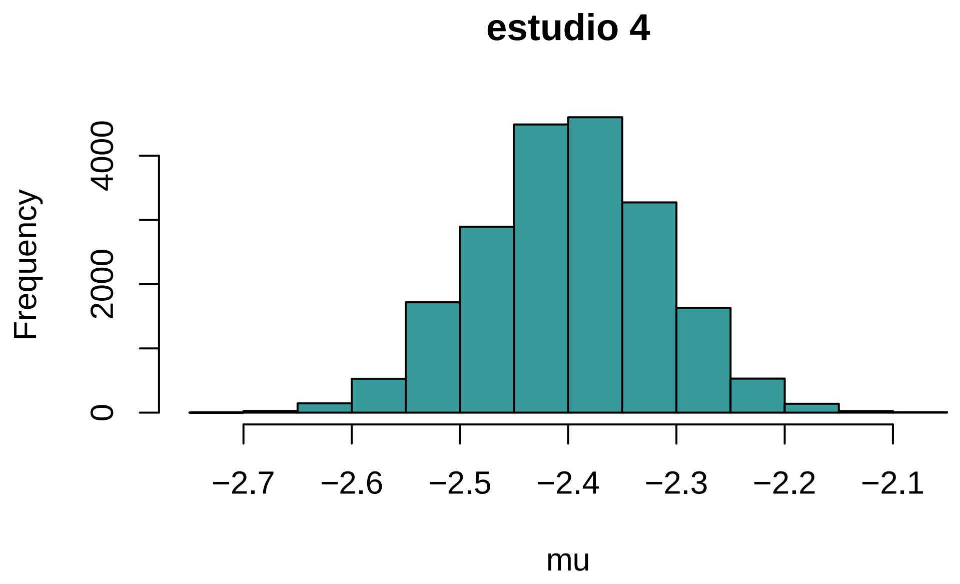
Histogramas de las posteriores de μ_i

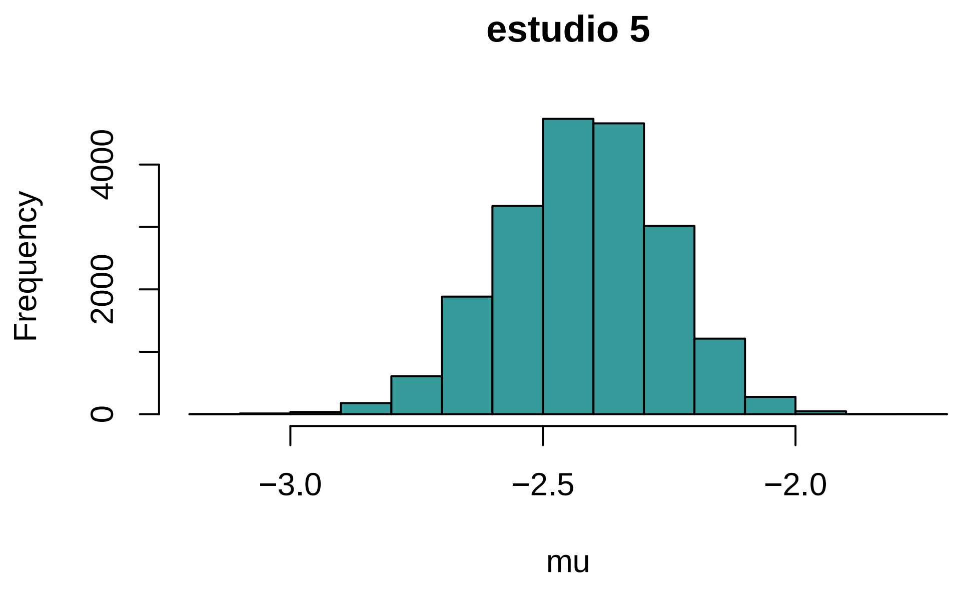
```r
# Graficar las distribuciones posteriores de mu y delta para cada estudio

# 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")

}
```

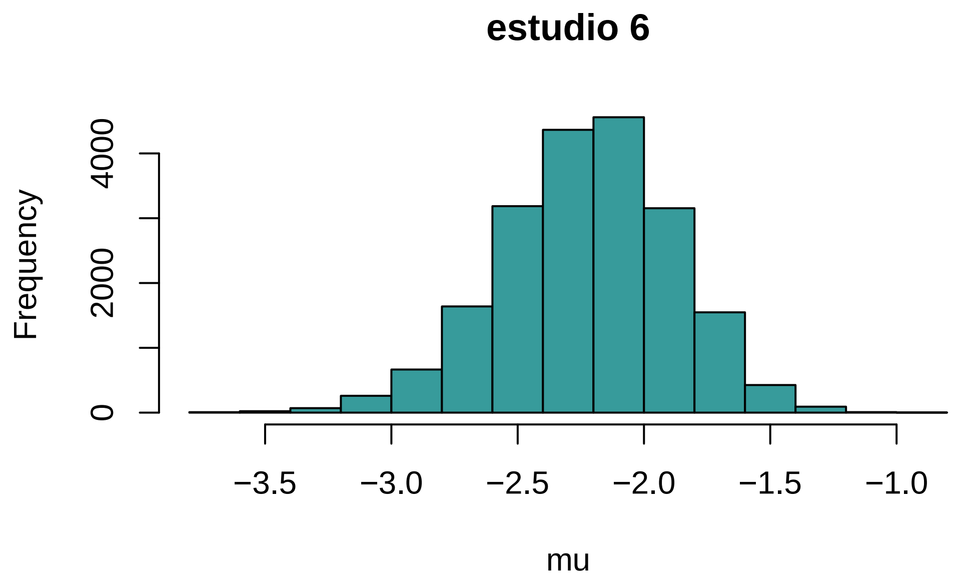

estudio 1
Frequency
0
1000
2500
−5
−4
−3
−2
−1
mu

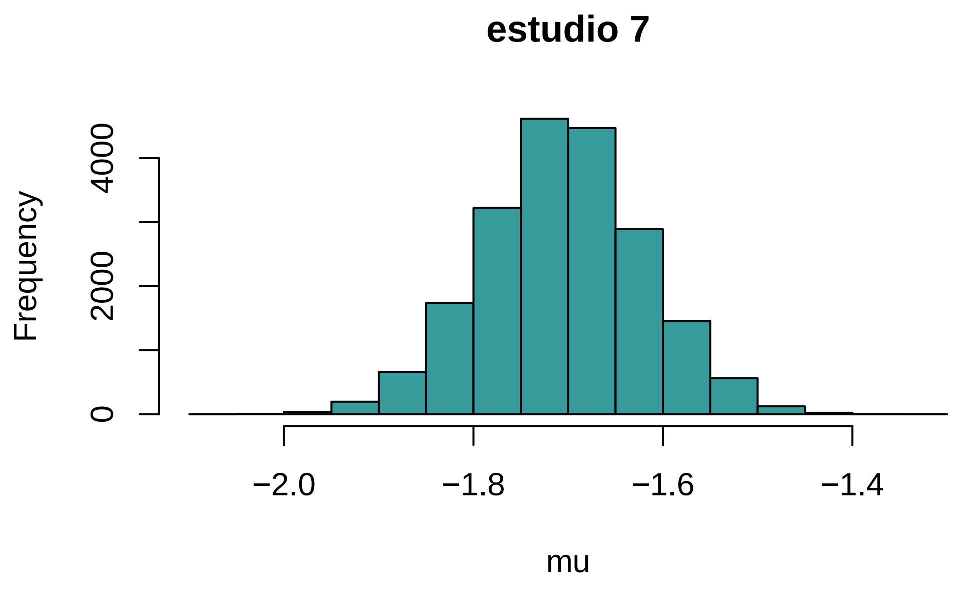

estudio 2
Frequency
0
1000
2500
−3.0
−2.5
−2.0
−1.5
mu

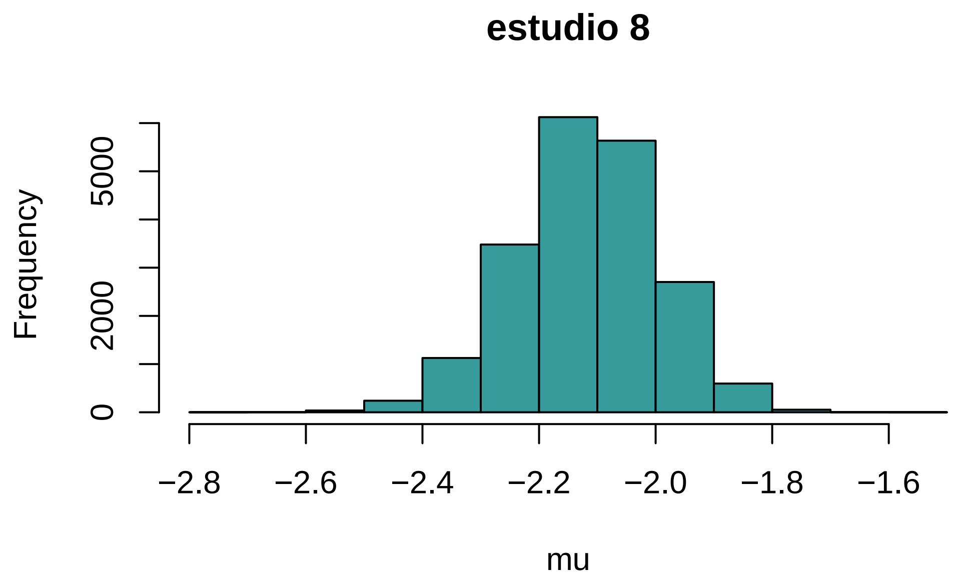

estudio 3
Frequency
0
1000
2000
3000
−3.0
−2.5
−2.0
−1.5
mu

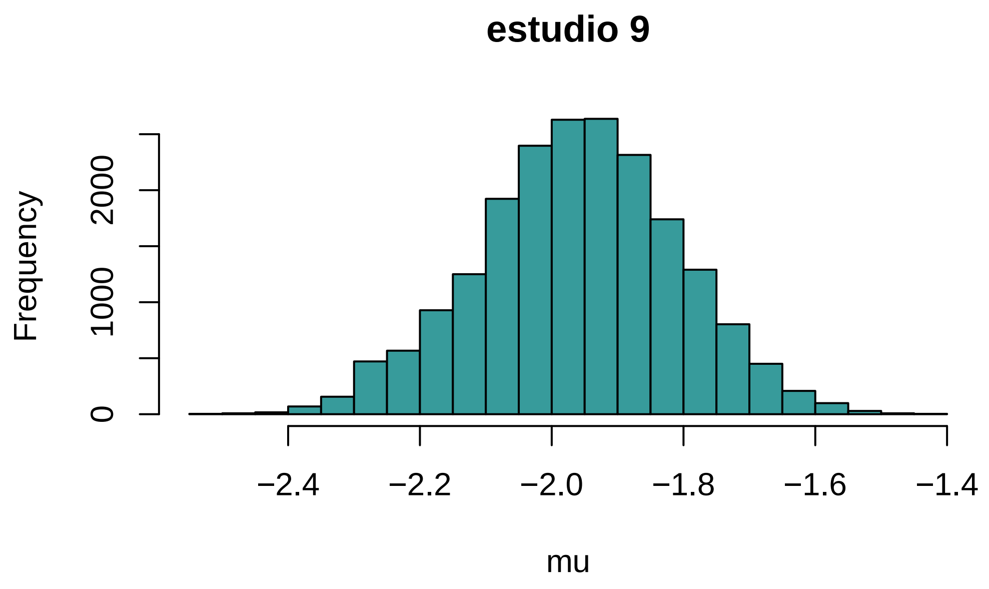

estudio 4
Frequency
0
2000
4000
−2.7
−2.6
−2.5
−2.4
−2.3
−2.2
−2.1
mu

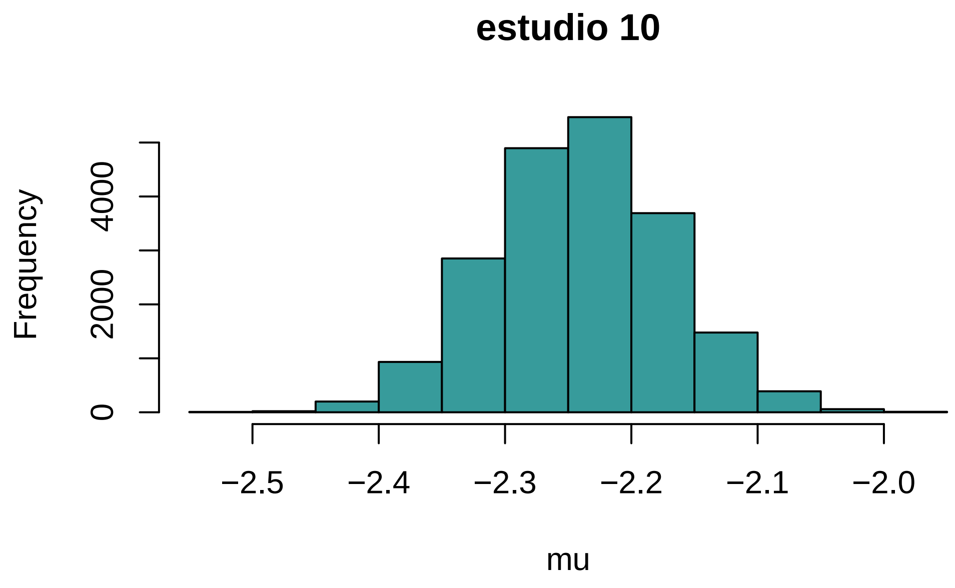

estudio 5
Frequency
0
2000
4000
−3.0
−2.5
−2.0
mu

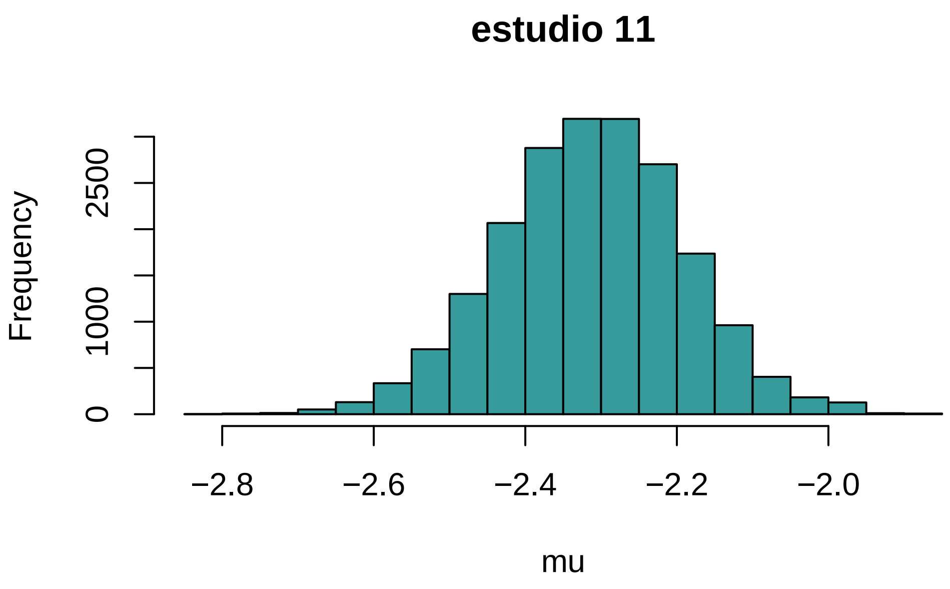

estudio 6
Frequency
0
2000
4000
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
mu

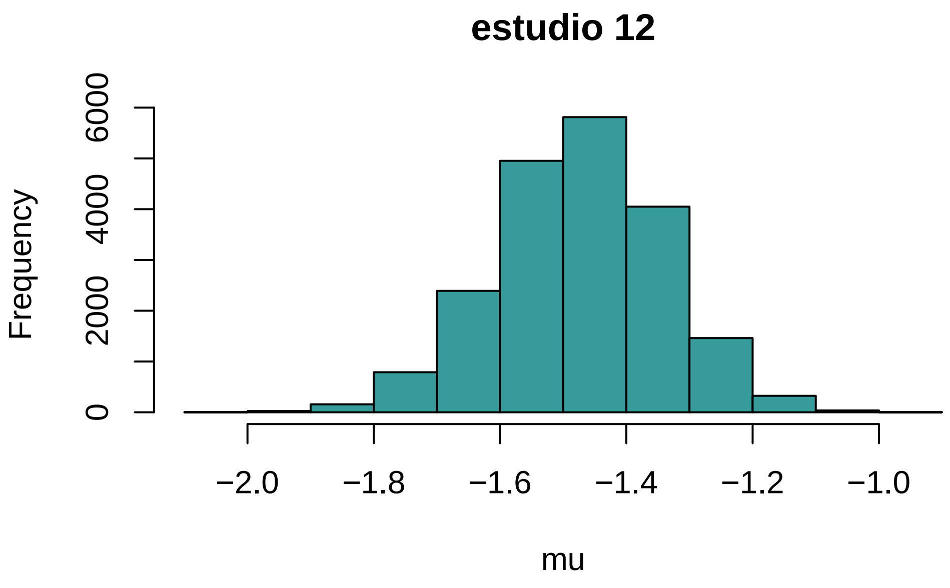

estudio 7
Frequency
0
2000
4000
−2.0
−1.8
−1.6
−1.4
mu

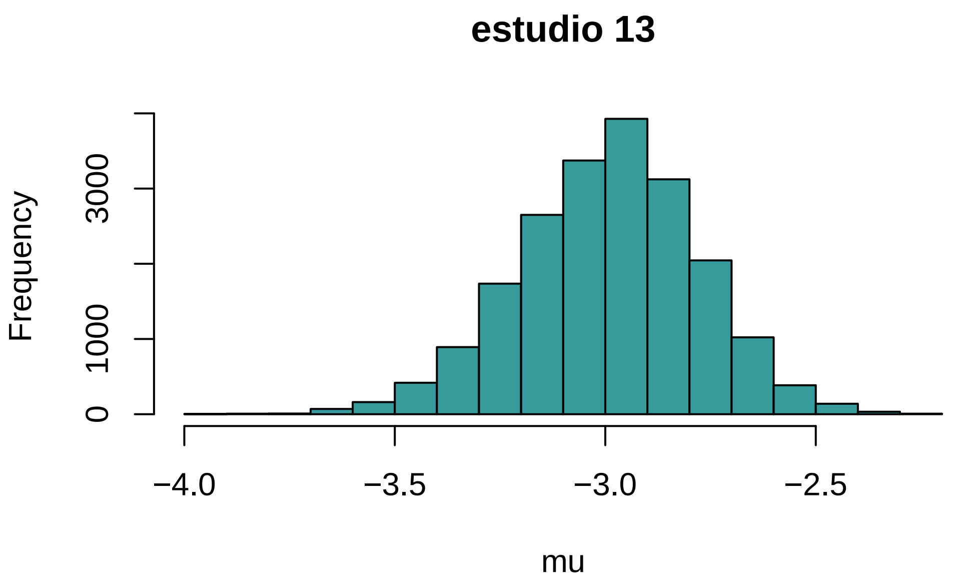

estudio 8
Frequency
0
2000
5000
−2.8
−2.6
−2.4
−2.2
−2.0
−1.8
−1.6
mu

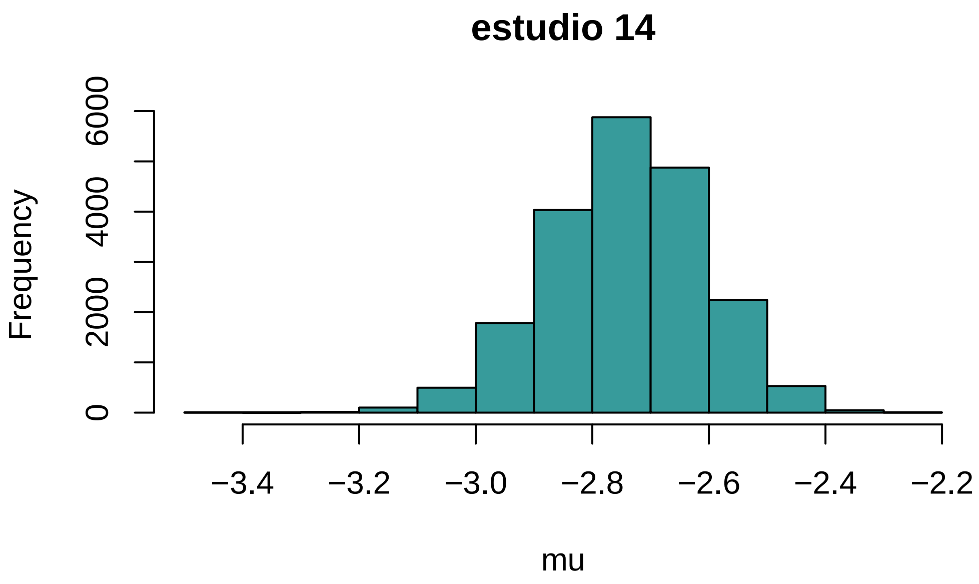

estudio 9
Frequency
0
1000
2000
−2.4
−2.2
−2.0
−1.8
−1.6
−1.4
mu

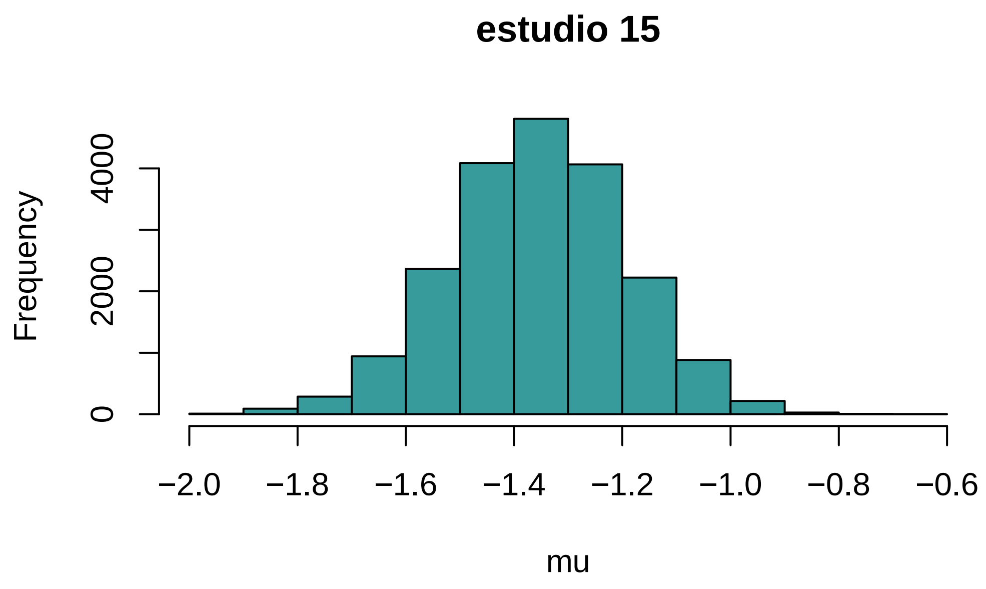

estudio 10
Frequency
0
2000
4000
−2.5
−2.4
−2.3
−2.2
−2.1
−2.0
mu

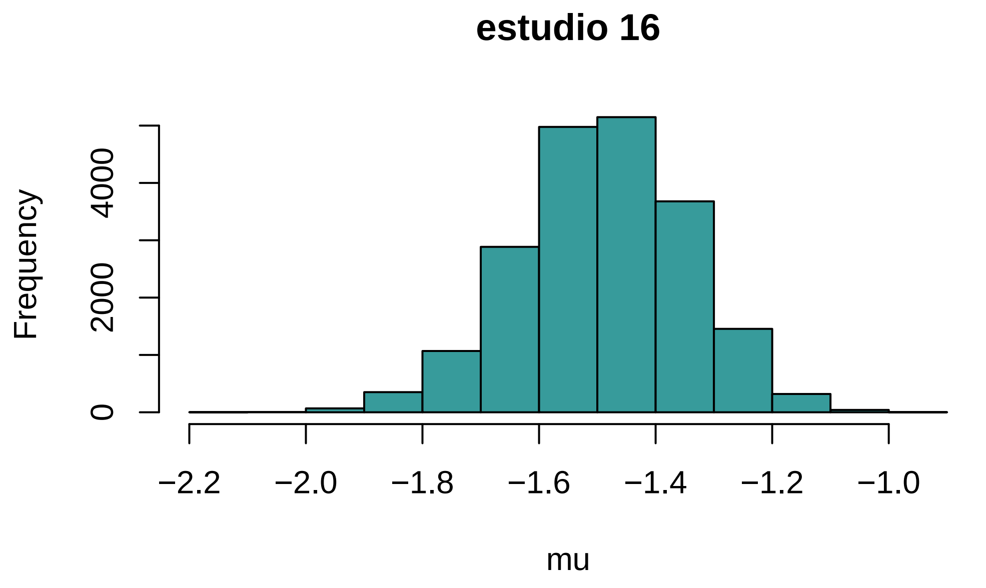

estudio 11
Frequency
0
1000
2500
−2.8
−2.6
−2.4
−2.2
−2.0
mu

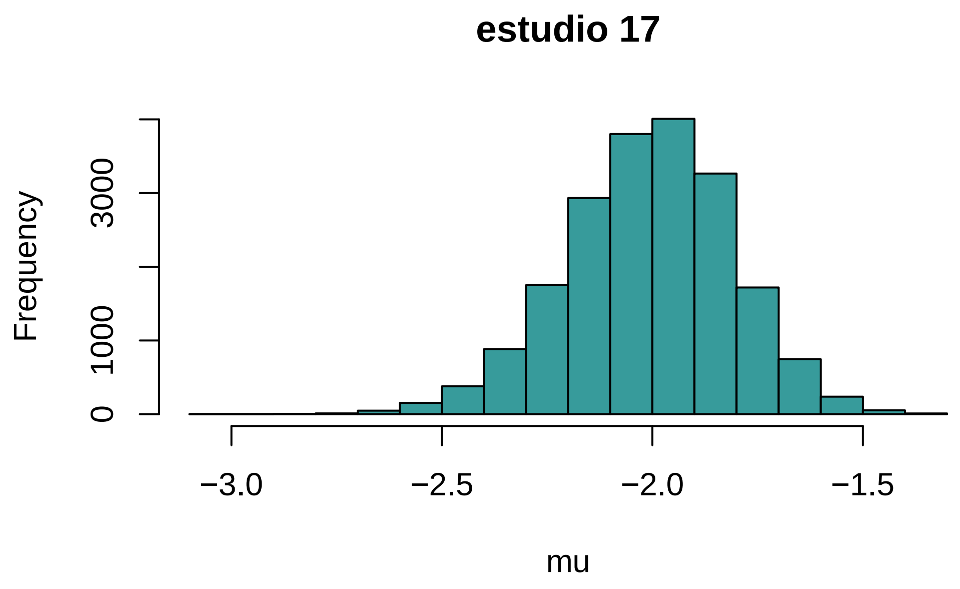

estudio 12
Frequency
0
2000
4000
6000
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

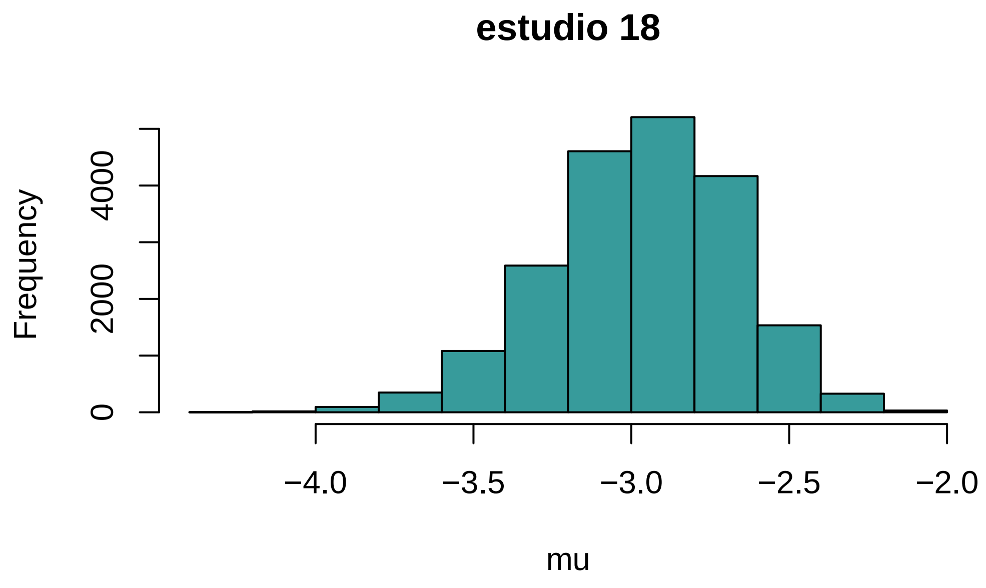

estudio 13
Frequency
0
1000
3000
−4.0
−3.5
−3.0
−2.5
mu

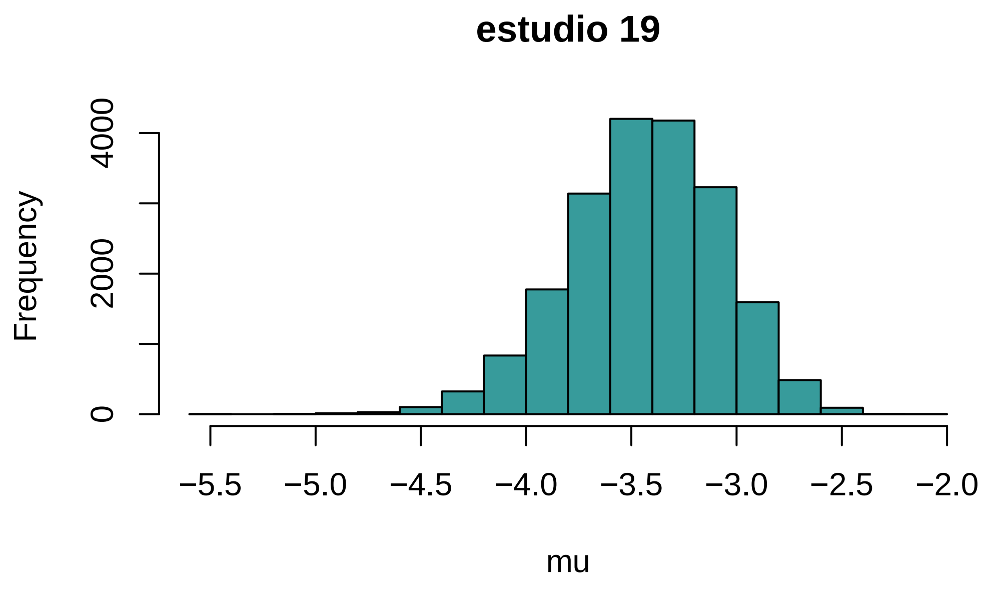

estudio 14
Frequency
0
2000
4000
6000
−3.4
−3.2
−3.0
−2.8
−2.6
−2.4
−2.2
mu

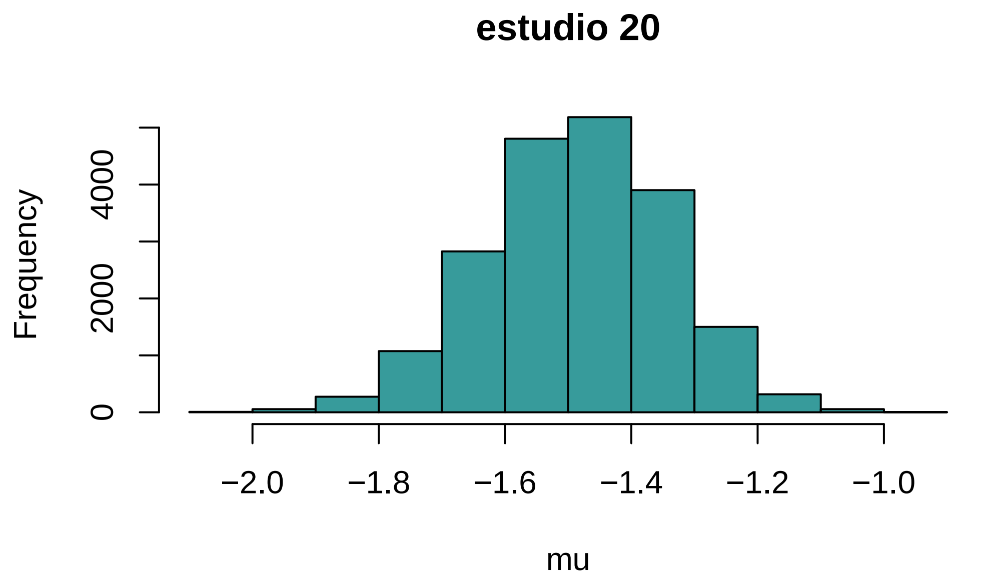

estudio 15
Frequency
0
2000
4000
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6
mu

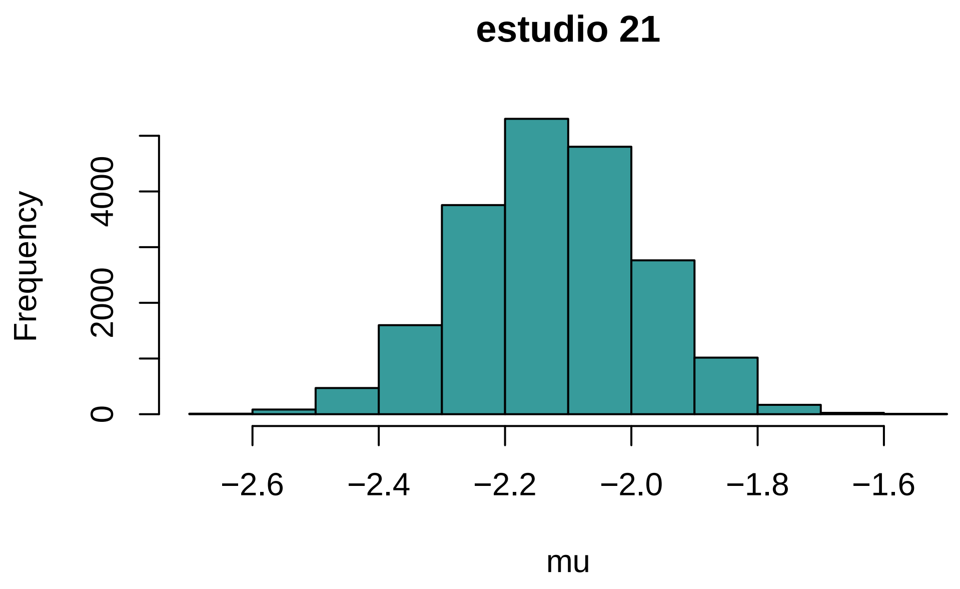

estudio 16
Frequency
0
2000
4000
−2.2
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

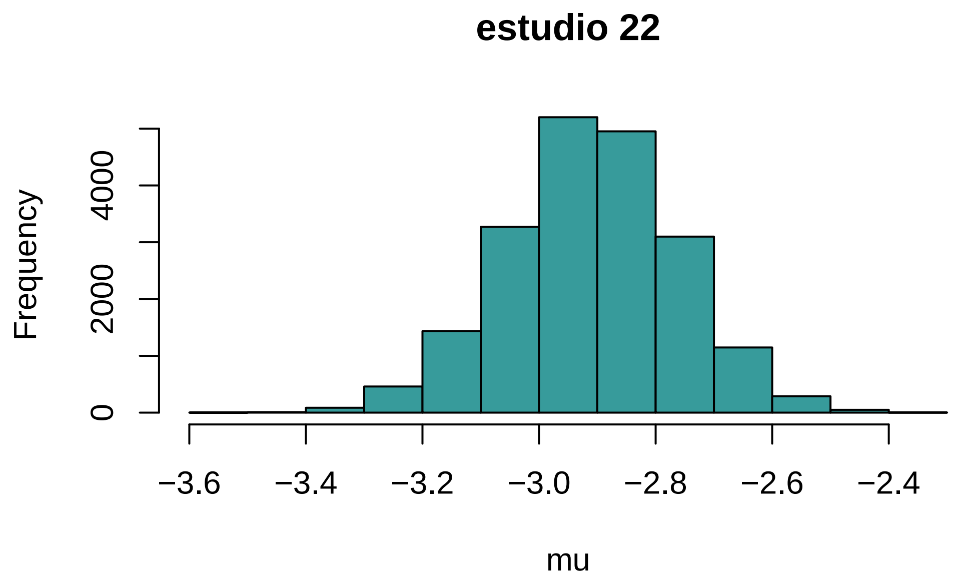

estudio 17
Frequency
0
1000
3000
−3.0
−2.5
−2.0
−1.5
mu


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


estudio 19
Frequency
0
2000
4000
−5.5
−5.0
−4.5
−4.0
−3.5
−3.0
−2.5
−2.0
mu

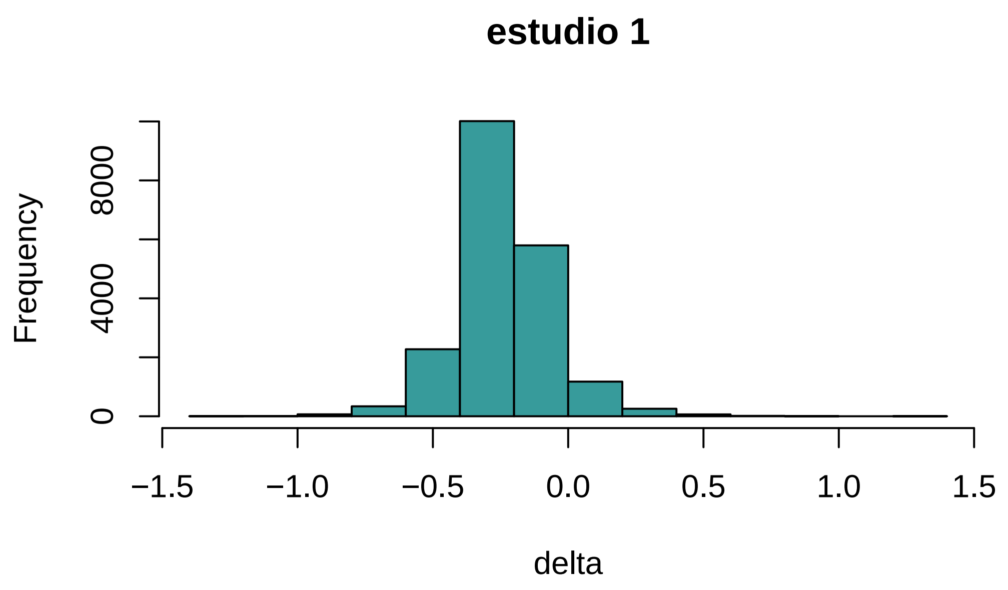

estudio 20
Frequency
0
2000
4000
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
mu

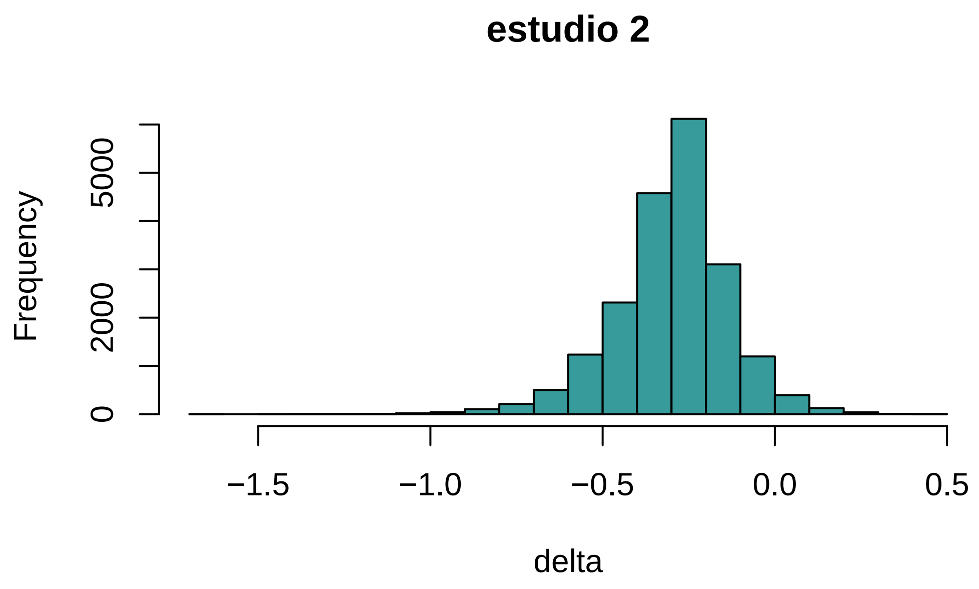


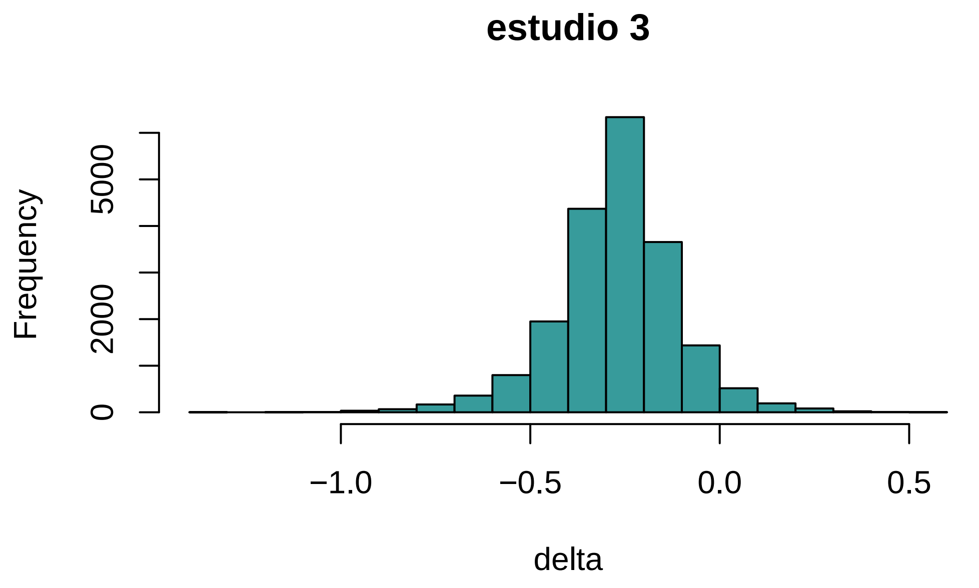


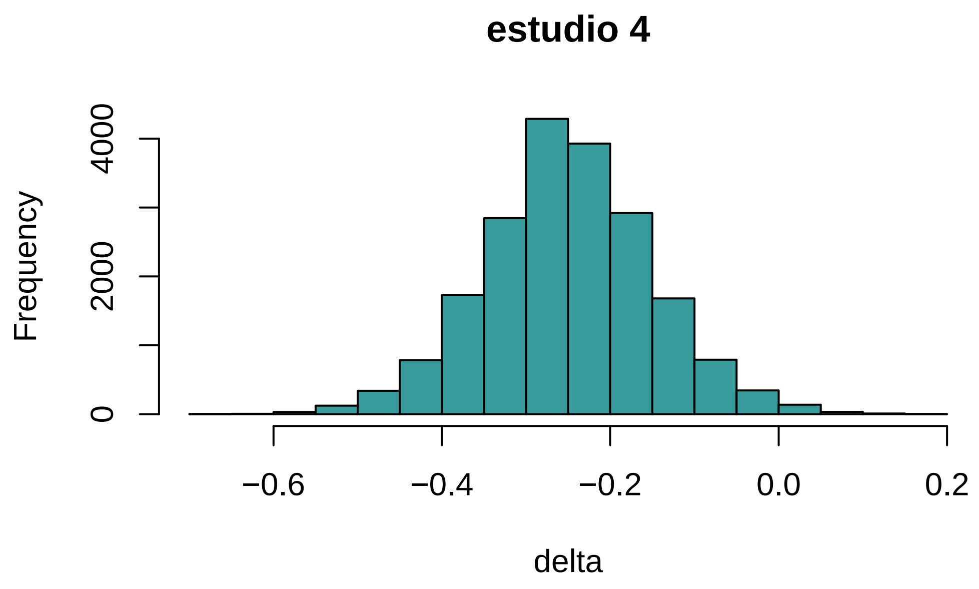
Histogramas de las posteriores de δ_i

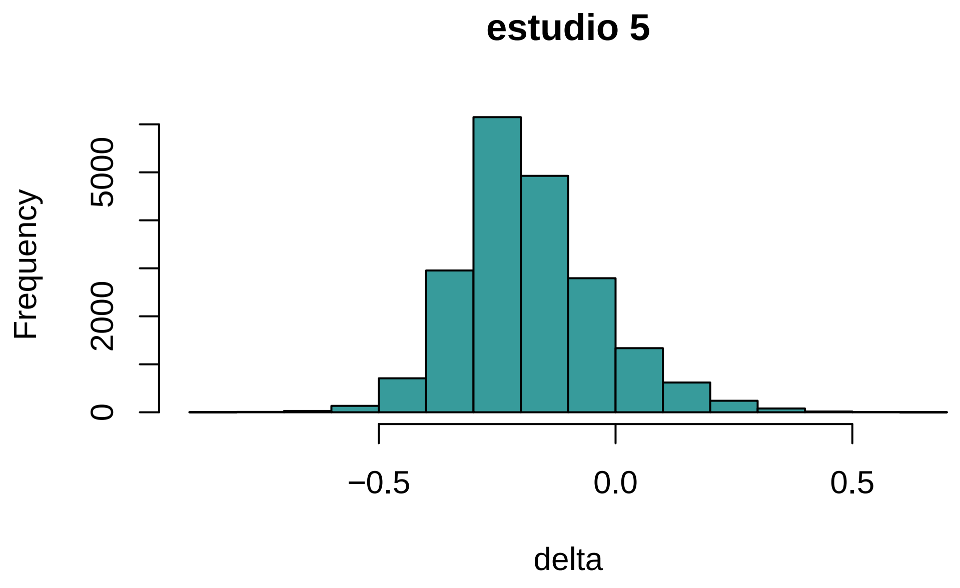
```
# par(mfrow = c(5, 5))

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")

}
```

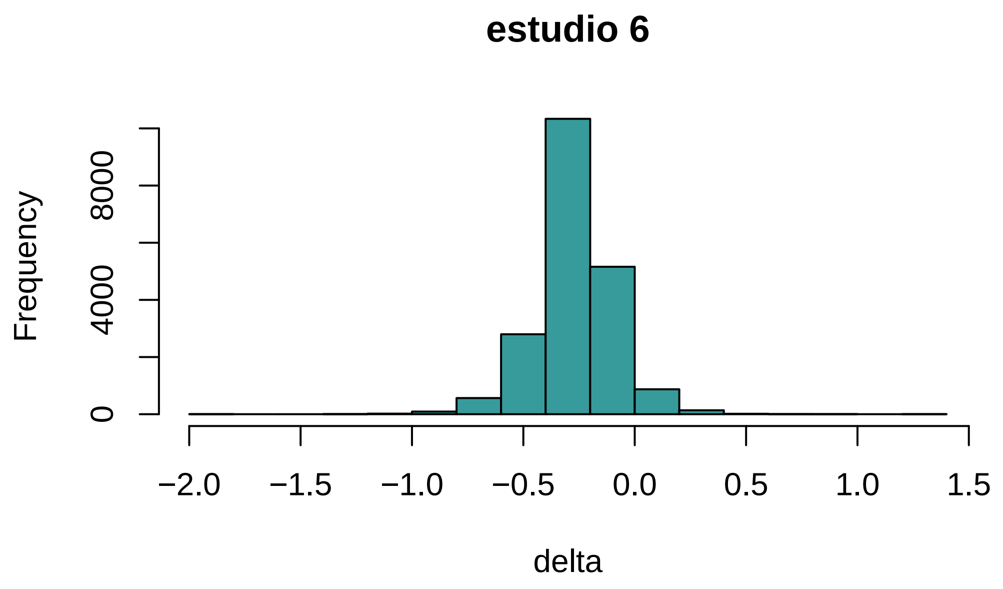


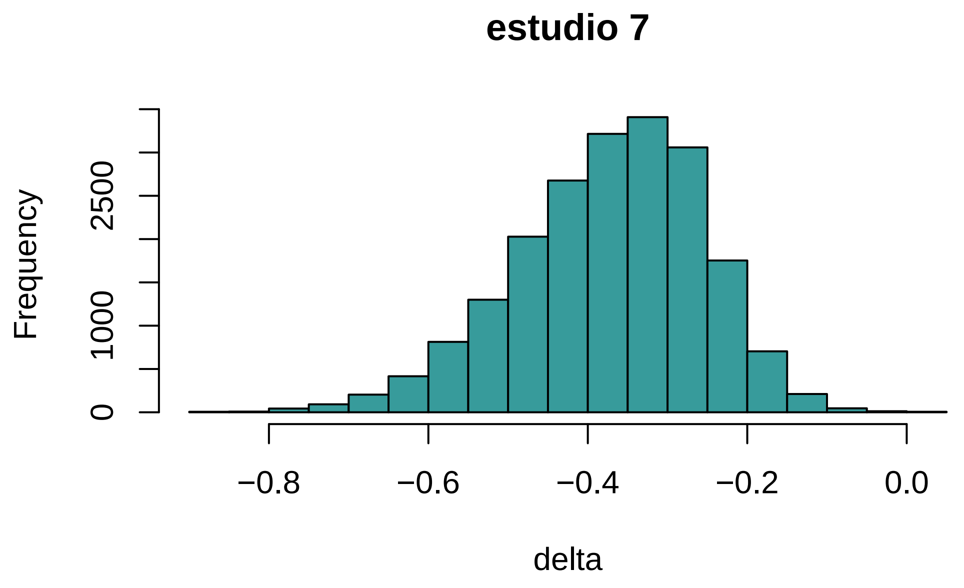

estudio 2
Frequency
0
2000
5000
−1.5
−1.0
−0.5
0.0
0.5
delta

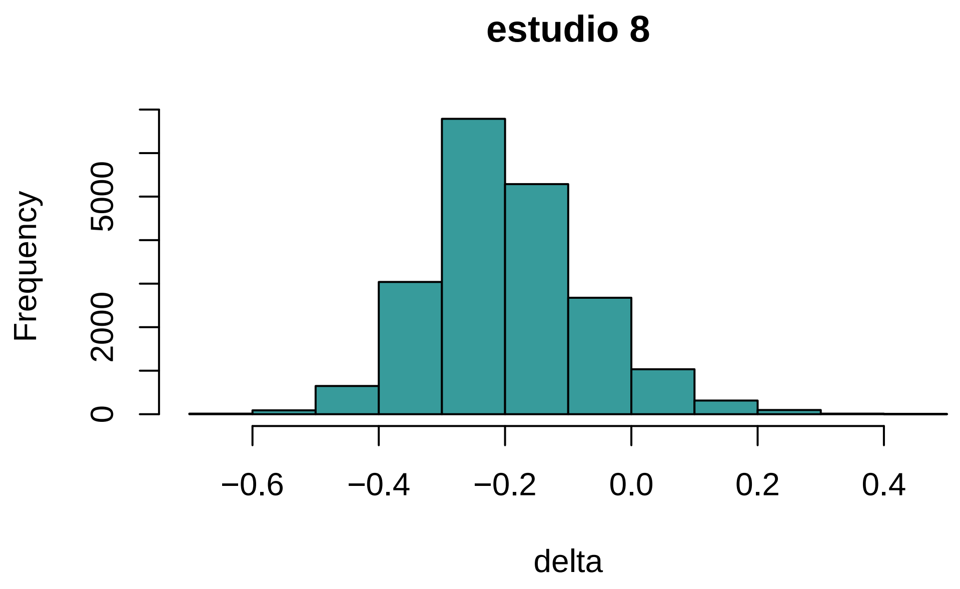

estudio 3
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

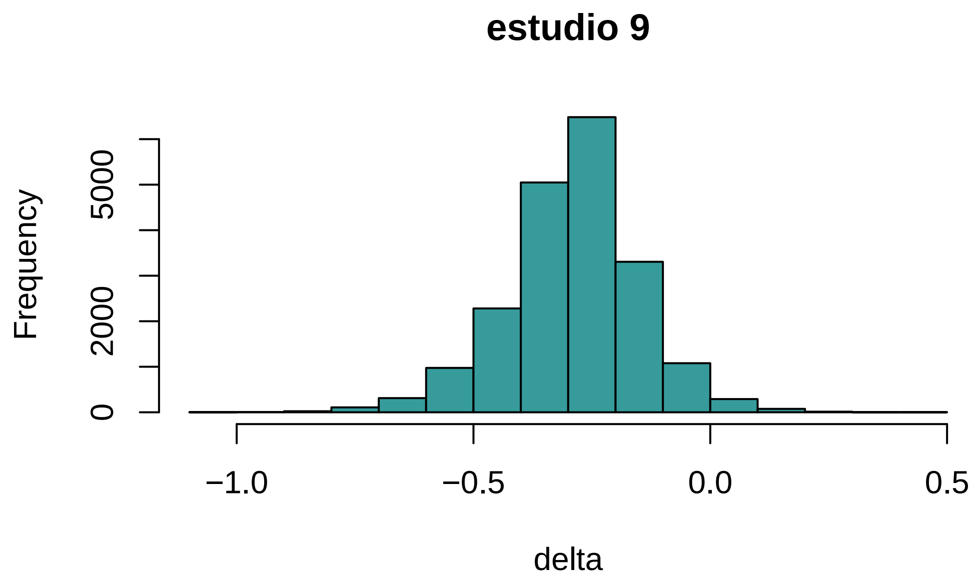

estudio 4
Frequency
0
2000
4000
−0.6
−0.4
−0.2
0.0
0.2
delta

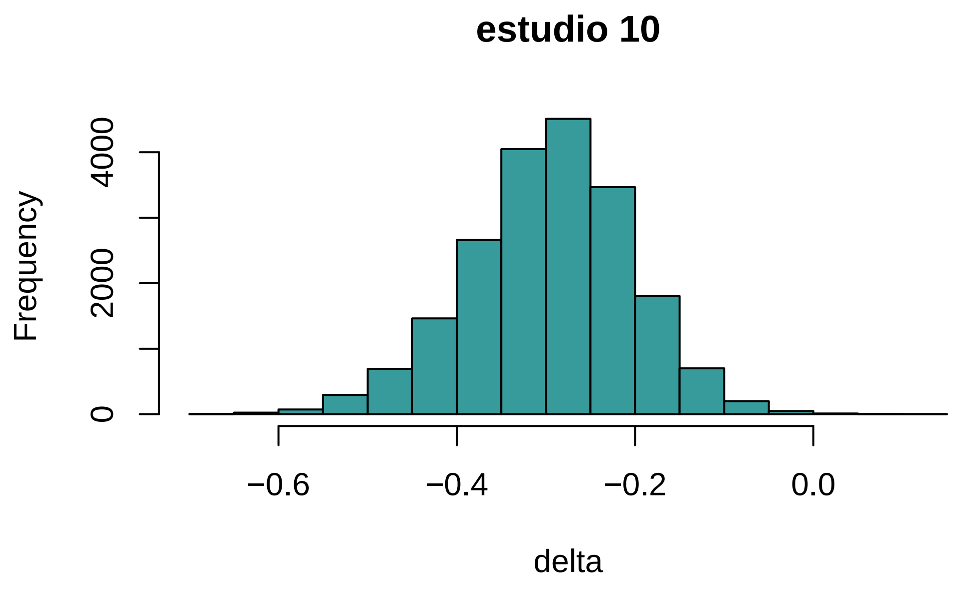

estudio 5
Frequency
0
2000
5000
−0.5
0.0
0.5
delta

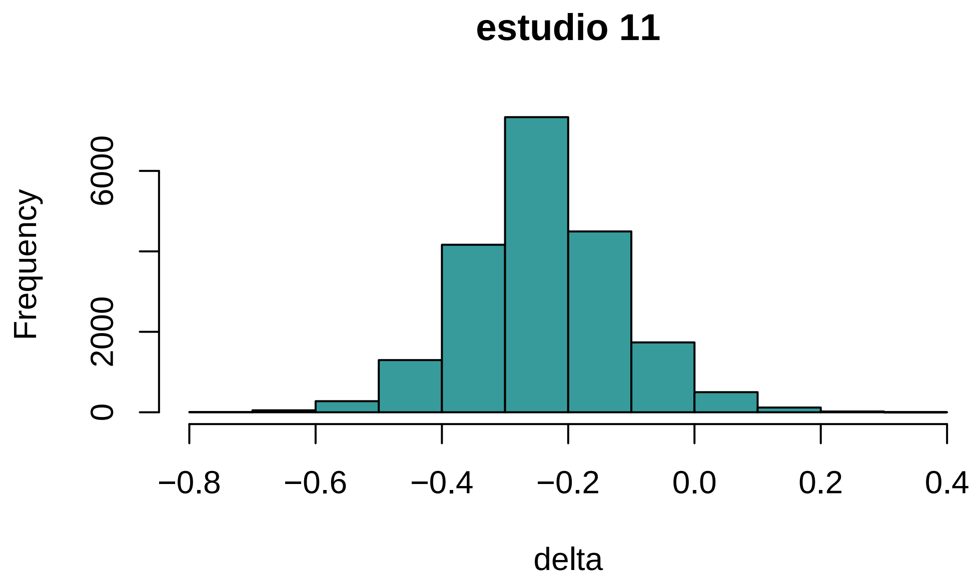

estudio 6
Frequency
0
4000
8000
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
delta

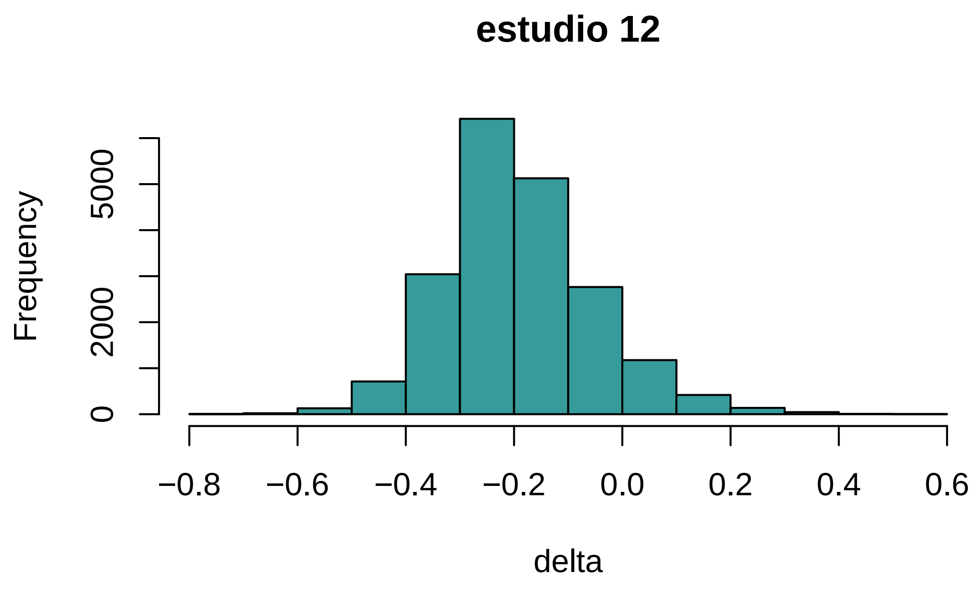

estudio 7
Frequency
0
1000
2500
−0.8
−0.6
−0.4
−0.2
0.0
delta

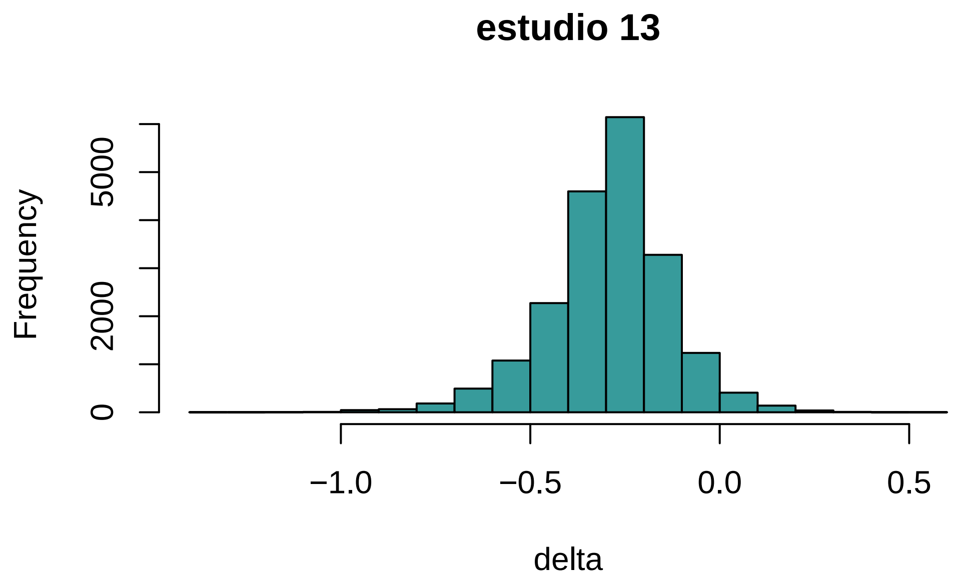

estudio 8
Frequency
0
2000
5000
−0.6
−0.4
−0.2
0.0
0.2
0.4
delta

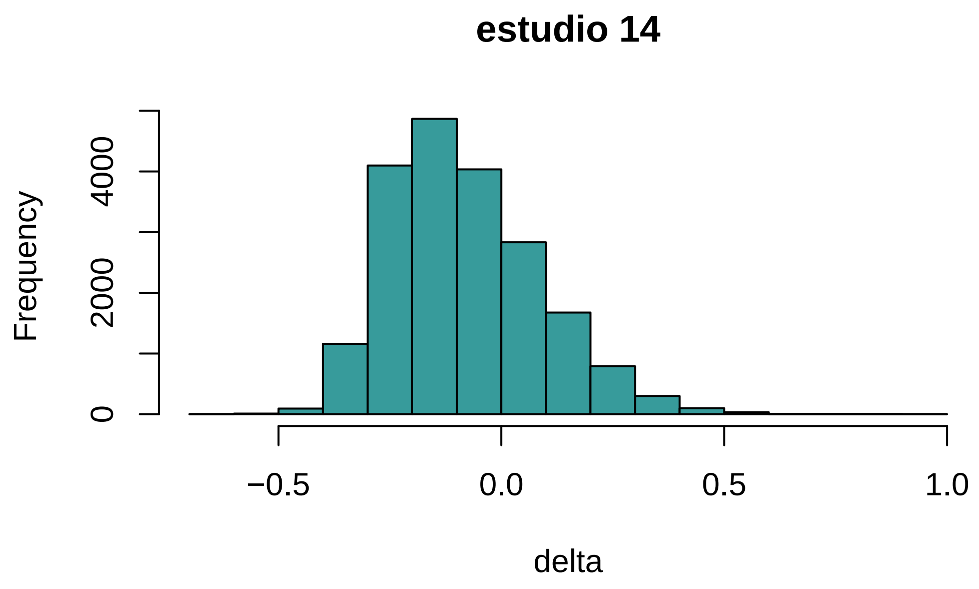

estudio 9
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

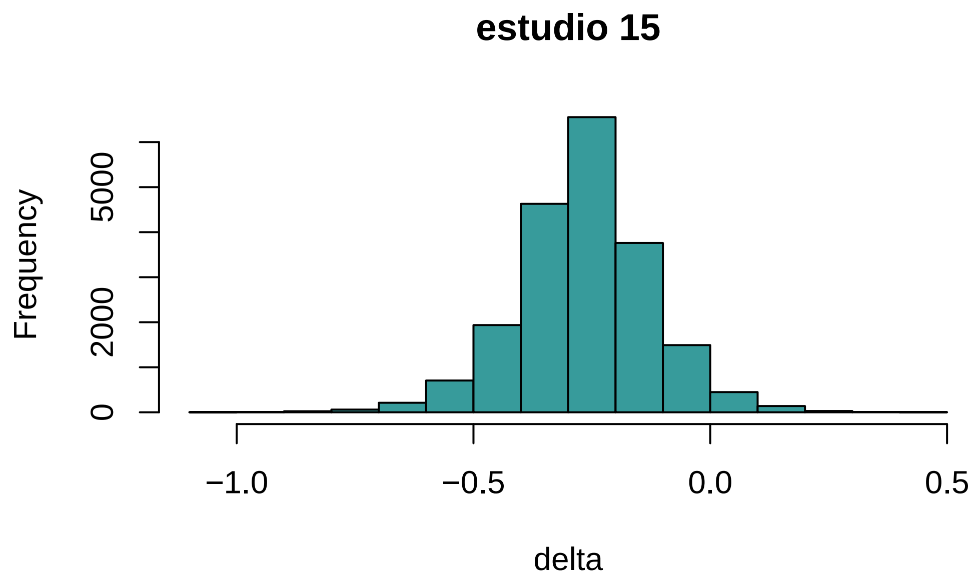

estudio 10
Frequency
0
2000
4000
−0.6
−0.4
−0.2
0.0
delta

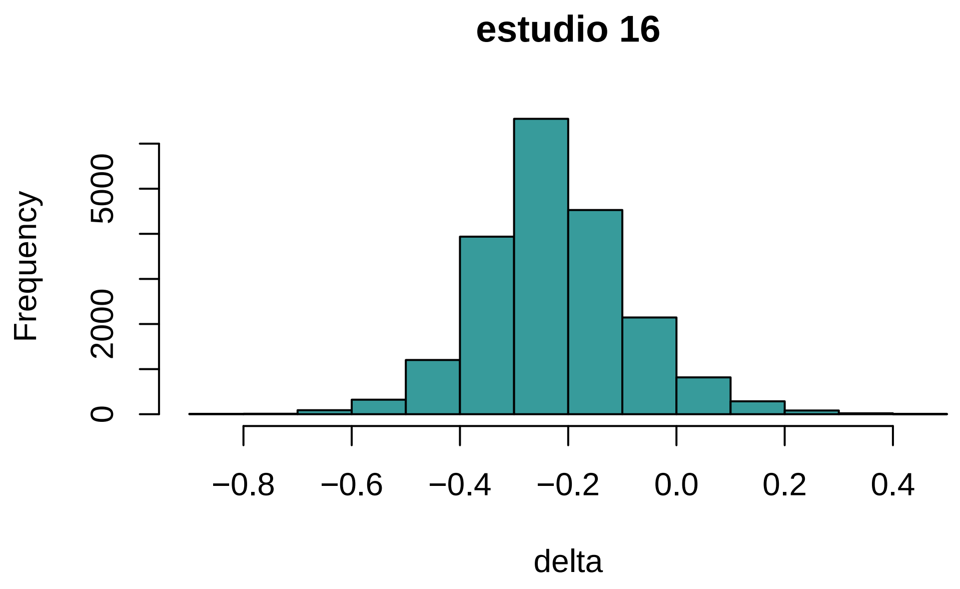

estudio 11
Frequency
0
2000
6000
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
delta

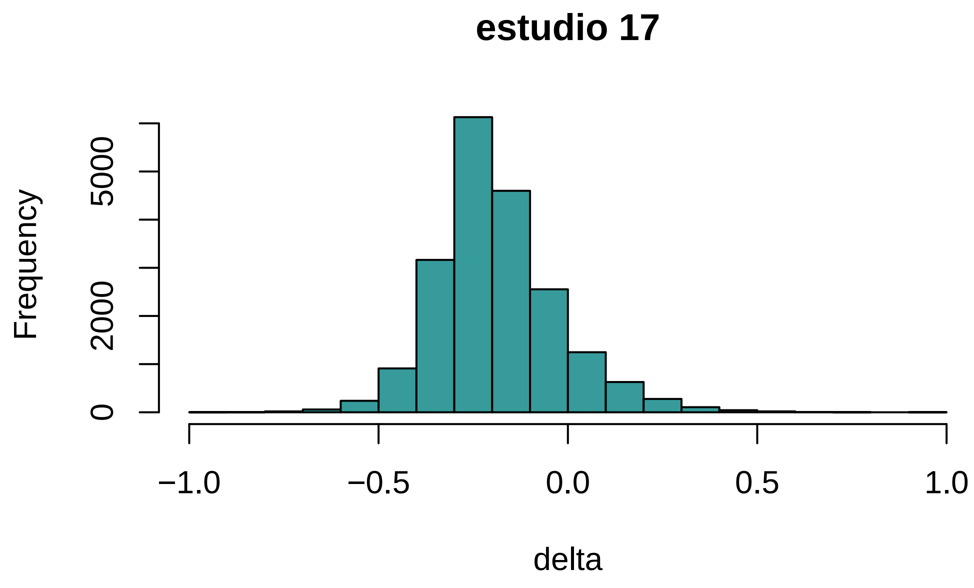

estudio 12
Frequency
0
2000
5000
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
delta

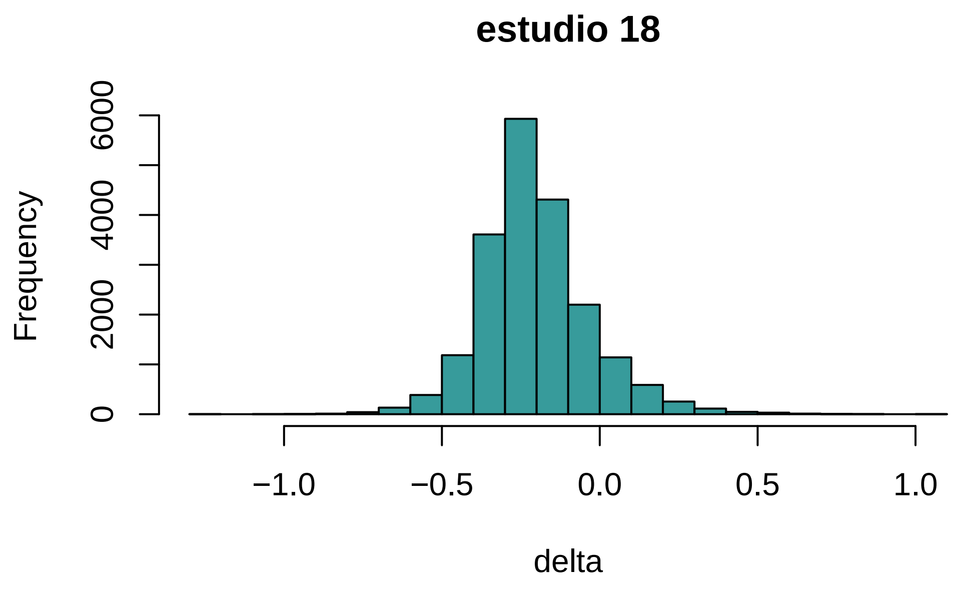

estudio 13
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

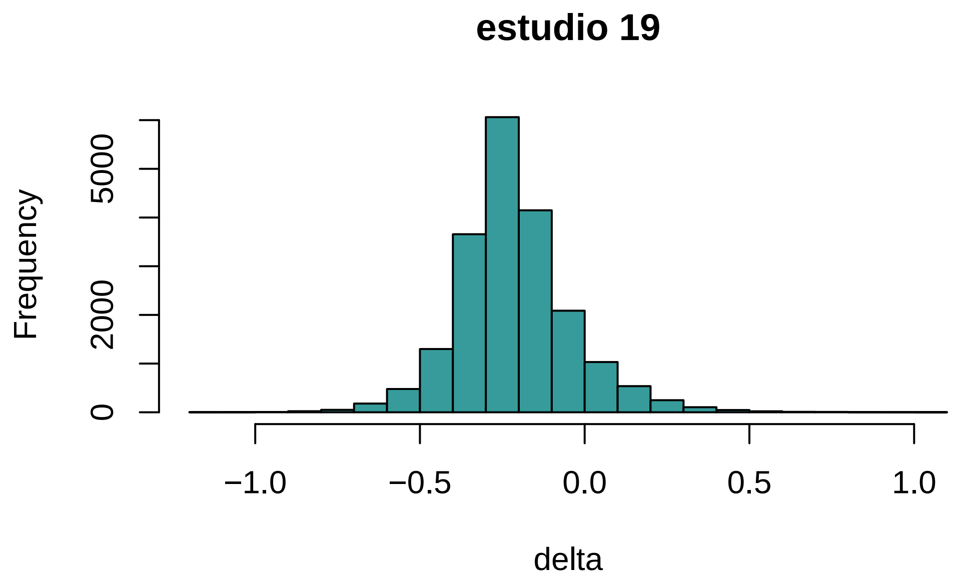

estudio 14
Frequency
0
2000
4000
−0.5
0.0
0.5
1.0
delta

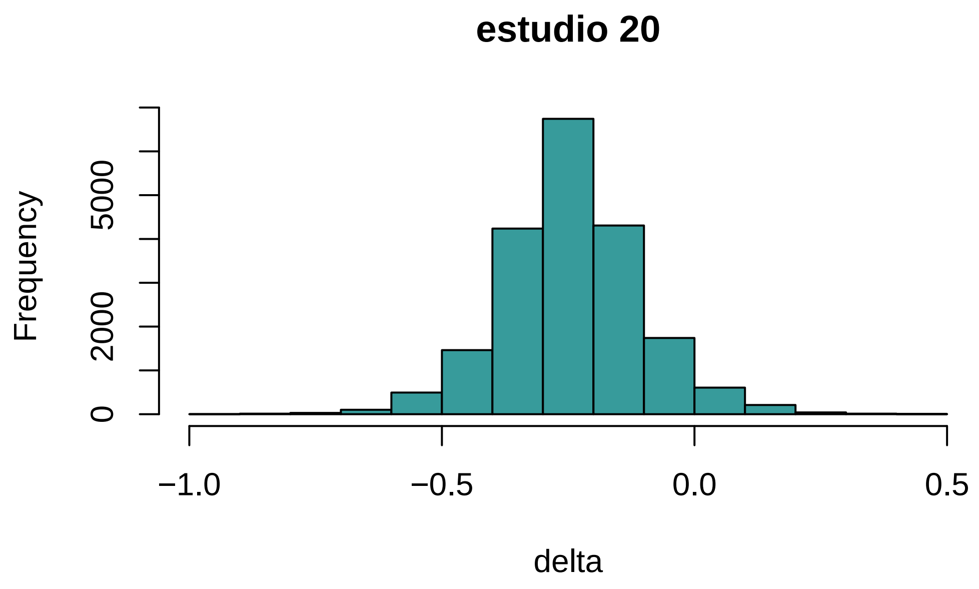

estudio 15
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta

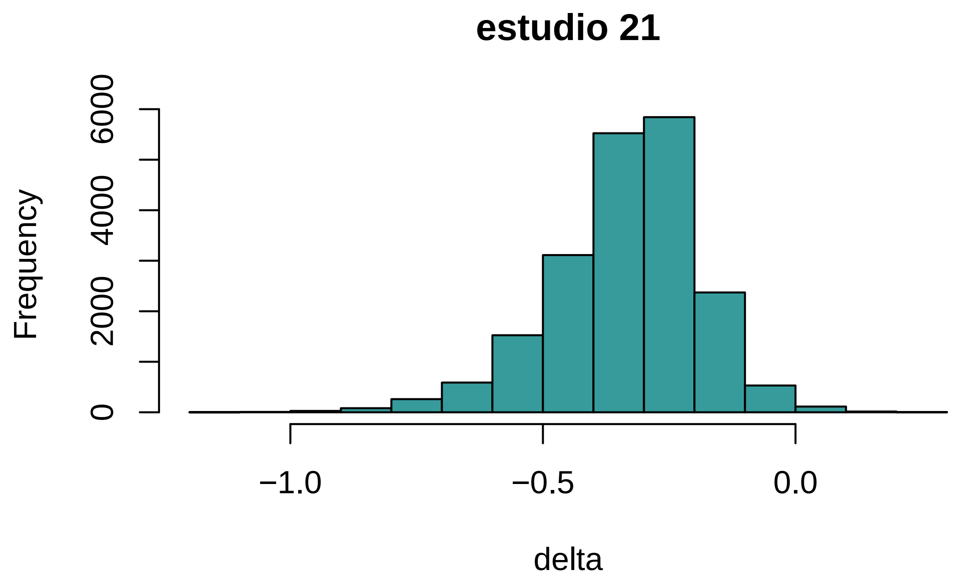

estudio 16
Frequency
0
2000
5000
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
delta

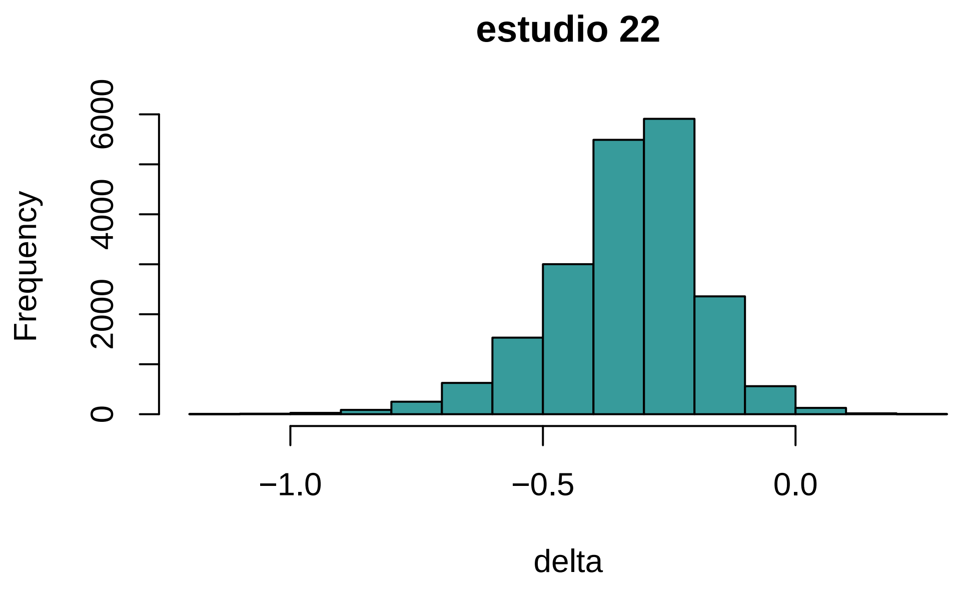

estudio 17
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
1.0
delta


estudio 18
Frequency
0
2000
4000
6000
−1.0
−0.5
0.0
0.5
1.0
delta


estudio 19
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
1.0
delta


estudio 20
Frequency
0
2000
5000
−1.0
−0.5
0.0
0.5
delta


estudio 21
Frequency
0
2000
4000
6000
−1.0
−0.5
0.0
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
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
)
```

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

```
         p_c       p_t   Reduccion
1  0.9241418 0.9056706 0.018471215
2  0.9241418 0.9003683 0.023773562
3  0.9241418 0.9028178 0.021324036
4  0.9241418 0.9046279 0.019513953
5  0.9241418 0.9104441 0.013697690
6  0.9241418 0.9030845 0.021057304
7  0.9241418 0.8934685 0.030673325
8  0.9241418 0.9092153 0.014926540
9  0.9241418 0.9011999 0.022941931
10 0.9241418 0.9006392 0.023502668
11 0.9241418 0.9056044 0.018537372
12 0.9241418 0.9096345 0.014507348
13 0.9241418 0.9009605 0.023181352
14 0.9241418 0.9181550 0.005986775
15 0.9241418 0.9032995 0.020842325
16 0.9241418 0.9068581 0.017283746
17 0.9241418 0.9098886 0.014253203
18 0.9241418 0.9085359 0.015605921
19 0.9241418 0.9076958 0.016446037
20 0.9241418 0.9052441 0.018897671
21 0.9241418 0.8970758 0.027065996
22 0.9241418 0.8971794 0.026962466
```

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.

```r
# 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)
```

```
# 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 8e-06 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.08 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: 0.162 seconds (Warm-up)
Chain 1:                0.188 seconds (Sampling)
Chain 1:                0.35 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: 0.159 seconds (Warm-up)
Chain 2:                0.182 seconds (Sampling)
Chain 2:                0.341 seconds (Total)
Chain 2:

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

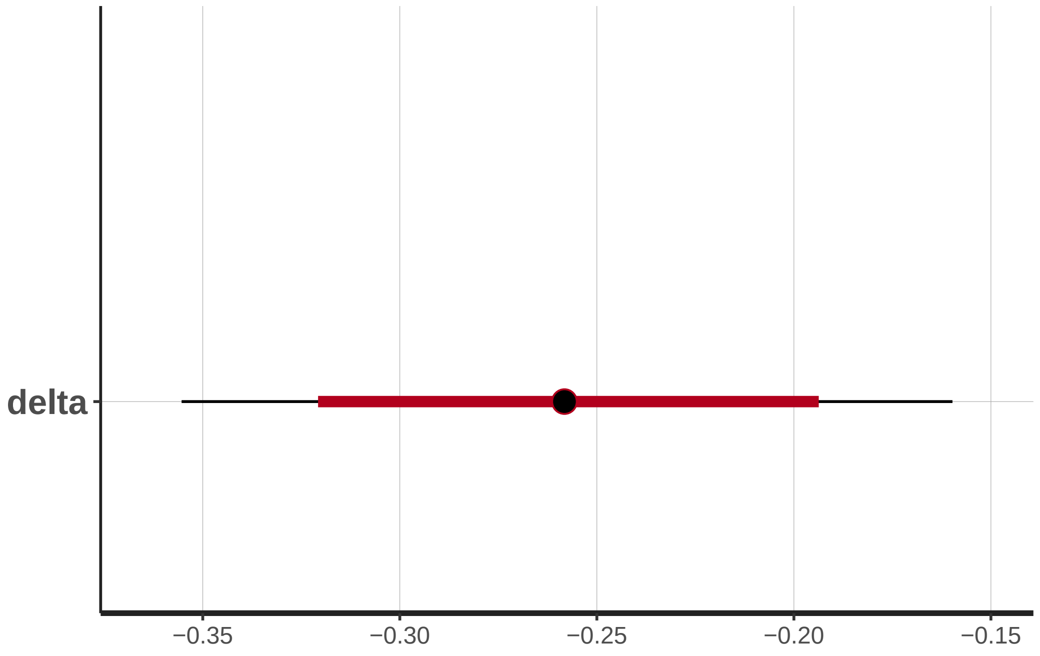
```
Chain 3:
Chain 3: Gradient evaluation took 6e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.06 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 10000 [  0%]  (Warmup)
Chain 3: Iteration: 1000 / 10000 [ 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: 0.157 seconds (Warm-up)
Chain 3:                0.173 seconds (Sampling)
Chain 3:                0.33 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: 0.158 seconds (Warm-up)
Chain 4:                0.182 seconds (Sampling)
Chain 4:                0.34 seconds (Total)
Chain 4:
```

Graficamos la posterior de δ y su histograma

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

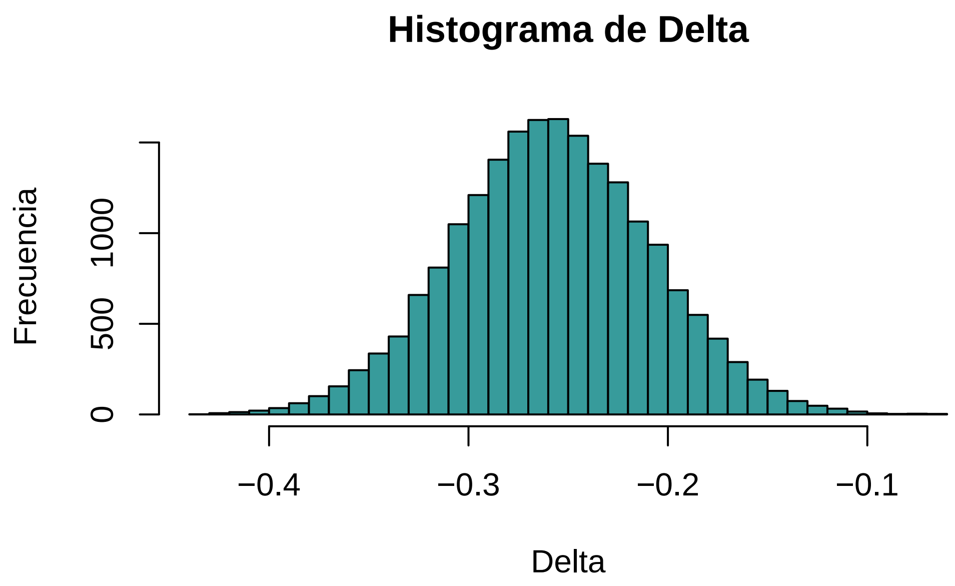
```
ci_level: 0.8 (80% intervals)
```

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



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

par(mfrow = c(1, 1))

# Histograma de las muestras posteriores de delta
```

```r
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. Privación de sueño

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.

```r
# Lectura de datos
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
```

```r
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~
```

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

```
[1] 298.5079
```

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

```
[1] 56.32876
```

Modelo agregado u homogéneo

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?

```r
data_list = list(
  N = nrow(sleep),
  rt_obs = sleep$Reaction,
  t = sleep$Days
)

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
}

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

transformed parameters {
  vector[N] media;
  for (i in 1:N){ // MODELO AGREGADO
    media[i] = alpha[i] + beta[i] * t[i]; //modelo dado OJO no se considera por sujeto
  }
}


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
  sigma ~ normal(57,25); // propuesta tomando en cuenta los datos pero tratando de que sea p
}
```

```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 5.8 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 5.9 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 6.8 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 8.6 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 7.1 seconds.

All 5 chains finished successfully.
Mean chain execution time: 6.9 seconds.
Total execution time: 34.7 seconds.
```

```
Warning: 5 of 5 chains had an E-BFMI less than 0.2.
See https://mc-stan.org/misc/warnings for details.
```

```r
fit2a$summary() %>% head()
```

```
# A tibble: 6 x 10
  variable    mean median    sd   mad     q5   q95  rhat ess_bulk ess_tail
  <chr>      <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>    <dbl>
1 lp__      -883.  -897.   86.3  83.4 -1002. -716.  1.24     17.5     51.5
2 alpha[1]  238.   242.   52.4  43.1   145.  318.  1.03   5432.     845.
3 alpha[2]  128.   128.  178.  178.   -162.  422.  1.00   6790.    3705.
4 alpha[3]   50.1   53.9 223.  223.   -320.  405.  1.00   5030.    3358.
5 alpha[4]   35.1   33.9 235.  233.   -350.  415.  1.00   6446.    3372.
6 alpha[5]   21.8   21.4 239.  233.   -369.  420.  1.00   6615.    3654.
```

b. Alpha y Beta homogéneas

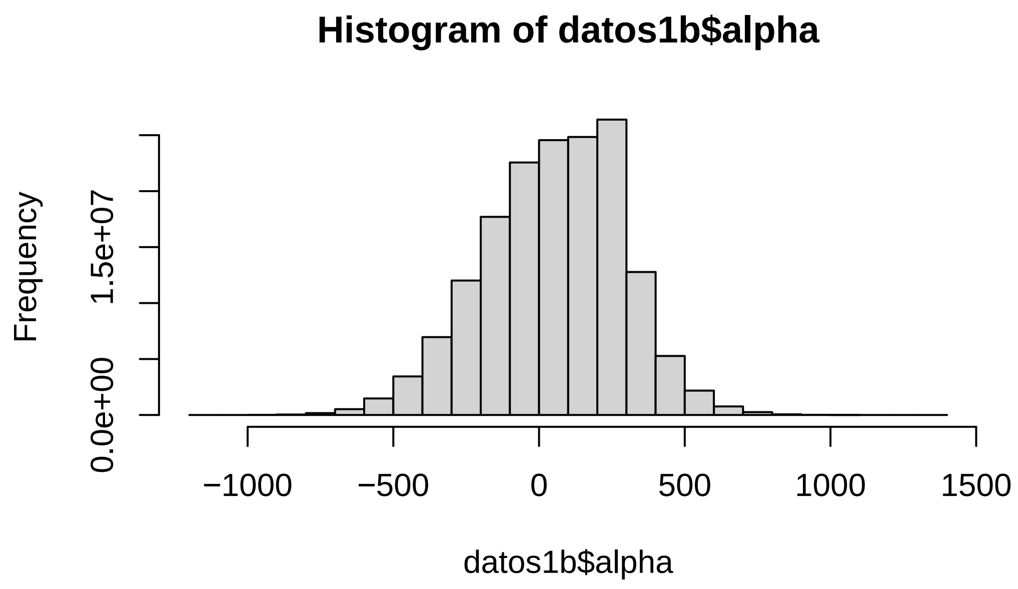
Graficar las muestras de la distribución posterior tanto de α como de β, £Cuál es la relación entre las dos variables y por qué?

```r
datos1b <- fit2a$draws(c("alpha","beta"), format = "df") %>%
  as_tibble() %>%
  pivot_longer(
    cols = starts_with("alpha"),
    names_to = "n",
    names_prefix = "alpha",
    values_to = "alpha") %>%
  pivot_longer(
    cols = starts_with("beta"),
    names_to = "n2",
    names_prefix = "beta",
    values_to = "beta") #%>% select(alpha, beta)
```

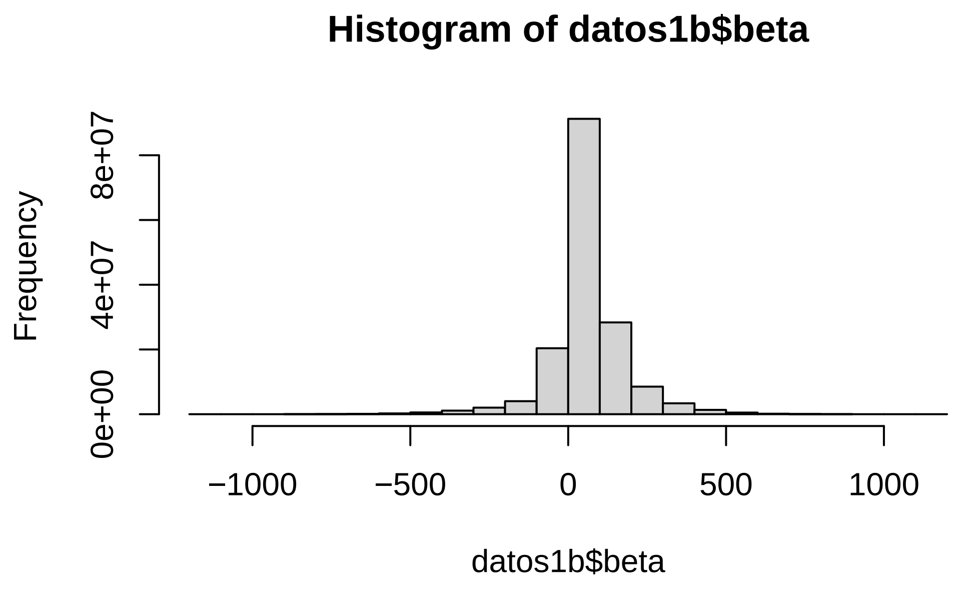
```r
glimpse(datos1b)
```

```
Rows: 162,000,000
Columns: 7
$ .chain     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ .iteration <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ .draw      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ n          <chr> "[1]", "[1]", "[1]", "[1]", "[1]", "[1]", "[1]", "[1]", "[1~
$ alpha      <dbl> 279.707, 279.707, 279.707, 279.707, 279.707, 279.707, 279.7~
$ n2         <chr> "[1]", "[2]", "[3]", "[4]", "[5]", "[6]", "[7]", "[8]", "[9~
$ beta       <dbl> 435.51100, 1.50489, 23.86000, -3.32989, 80.36850, 67.87190,~
```

```
hist(datos1b$alpha)
```



```
hist(datos1b$beta)
```



Para mantener el promedio de los tiempos de reaccion observados si alpha sube, beta tiene que bajar y viceversa, por lo que hay una fuerte relación negativa

c. Distribución posterior para cada individuo

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.

```r
modelo_2c <- cmdstan_model("./Tarea5_2c.stan")
print(modelo_2c)
```

```
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
}

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

model {
  for (i in 1:N){
    rt_obs[i] ~ normal(alpha[i] + beta[i] * t[i], sigma); //modelo dado
  }
  alpha ~ normal(0, 250);  // distribucion dada
  beta ~ normal(0, 250);   // distribucion dada
  sigma ~ normal(57,25); // propuesta tomando en cuenta los datos pero tratando de que sea po
}

generated quantities {
  vector[N] rt_sim;
  for (i in 1:N){
    rt_sim[i] = normal_rng(alpha[i] + beta[i] * t[i], sigma);
  }
}
```

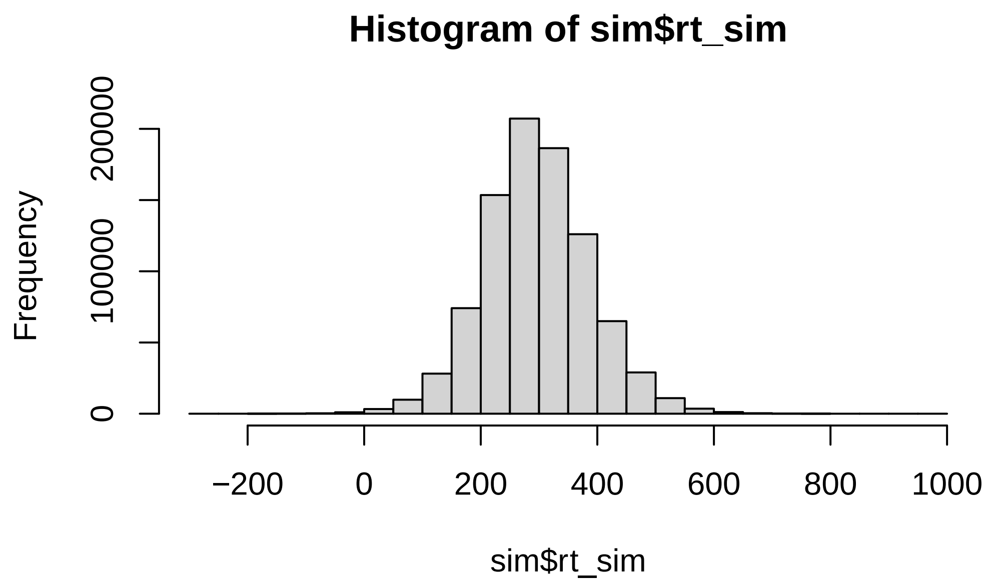
```
fit2c <- modelo_2c$sample(
  data = data_list,
  seed = 123,
  chains = 5,
  iter_sampling = 1000,
  iter_warmup = 1000,
  # show_messages = FALSE,
  # show_exceptions = FALSE
  )
```

```
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 5.5 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 5.5 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 6.4 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 8.2 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 6.9 seconds.

All 5 chains finished successfully.
Mean chain execution time: 6.5 seconds.
Total execution time: 33.8 seconds.

Warning: 5 of 5 chains had an E-BFMI less than 0.2.
See https://mc-stan.org/misc/warnings for details.
```

```
fit2c$summary()
```

```
# A tibble: 542 x 10
   variable    mean   median    sd   mad    q5   q95  rhat ess_bulk ess_tail
   <chr>      <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
 1 lp__     -872.    -882.    79.8  82.8 -984. -728.  1.25     15.6     40.0
 2 alpha[1]  239.     241.    45.7  38.5  162.  310.  1.02   4060.    1015.
 3 alpha[2]  128.     129.   183.  183.  -173.  426.  1.00   6398.    3751.
 4 alpha[3]   49.7     51.1  222.  217.  -323.  421.  1.00   5647.    3621.
 5 alpha[4]   30.0     32.0  238.  241.  -364.  424.  1.00   6230.    3186.
 6 alpha[5]   26.1     28.4  244.  238.  -386.  430.  1.00   6954.    3770.
 7 alpha[6]   17.2     21.4  249.  258.  -392.  419.  1.00   6975.    3519.
 8 alpha[7]    8.06    10.2  244.  241.  -398.  407.  1.00   6685.    3685.
 9 alpha[8]    4.86     8.57 245.  244.  -407.  405.  1.00   6121.    3208.
10 alpha[9]    0.777   -0.663 248.  242.  -402.  409.  1.00   6826.    3536.
# i 532 more rows
```

```
sim <- fit2c$draws(c("rt_sim"), format = "df") %>%
  as_tibble() %>%
  pivot_longer(
    cols = starts_with("rt_sim"),
    names_to = "n",
    names_prefix = "rt_sim",
    values_to = "rt_sim")  %>% select(rt_sim)
hist(sim$rt_sim)
```



Ojo los tiempos negativos no tienen sentido

d. Modelo heterogeneo

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 = nrow(sleep),
  rt_obs = sleep$Reaction,
  t = sleep$Days,
  subject = sleep$Subject
)

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

```
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[18] alpha;
  vector[18] beta;
  real<lower=0> sigma;
}

transformed parameters {
  vector[N] media;
  for (i in 1:N){
    media[i] = alpha[subject[i]] + beta[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
  sigma ~ normal(57,25); // propuesta tomando en cuenta los datos pero tratando de que sea p

}
```

```
fit2d <- modelo_2d$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 1.5 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 1.5 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 1.5 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 1.5 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 1.5 seconds.

All 5 chains finished successfully.
Mean chain execution time: 1.5 seconds.
Total execution time: 8.7 seconds.
```

e. Comparación entre Betas

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?

```
fit2d$summary()
```

```
# A tibble: 218 x 10
   variable   mean median    sd   mad    q5   q95  rhat ess_bulk ess_tail
   <chr>     <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
 1 lp__      -681.  -681.  4.86  4.94 -690. -674.  1.00    1954.    3024.
 2 alpha[1]   243.   243. 15.3  15.3   218.  268.  1.00    5338.    3674.
 3 alpha[2]   204.   204. 15.6  15.5   179.  229.  1.00    6633.    3862.
 4 alpha[3]   203.   203. 15.3  15.5   177.  227.  1.00    5670.    3838.
 5 alpha[4]   289.   289. 15.5  15.4   263.  314.  1.00    4719.    3562.
 6 alpha[5]   285.   285. 15.4  15.4   260.  310.  1.00    5809.    4010.
 7 alpha[6]   263.   263. 14.9  14.7   238.  288.  1.00    5483.    4148.
 8 alpha[7]   274.   274. 15.2  15.1   248.  299.  1.00    5468.    4034.
 9 alpha[8]   239.   239. 15.0  14.8   214.  264.  1.00    5675.    4037.
10 alpha[9]   262.   262. 15.3  15.3   237.  287.  1.00    5146.    3510.
# i 208 more rows
```

Los datos de beta varían de acuerdo con el idividuo, pero el media general debería ser parecida, probablemente no sean tan cercanas por que nuestras prior no son tan buenas

```
datos2b <- fit2a$draws(c("beta"), format = "df") %>%
  as_tibble() %>%
  pivot_longer(
    cols = starts_with("beta"),
    names_to = "n2",
    names_prefix = "beta",
    values_to = "beta") %>% select(beta)
datos2e <- fit2d$draws(c("beta"), format = "df") %>%
  as_tibble() %>%
  pivot_longer(
    cols = starts_with("beta"),
    names_to = "n2",
    names_prefix = "beta",
    values_to = "beta") %>% select(beta)
```

```r
mean(datos2b$beta)
```

```
[1] 61.93001
```

```r
mean(datos2e$beta)
```

```
[1] 10.62423
```

f. Distribución posterior para cada individuo

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

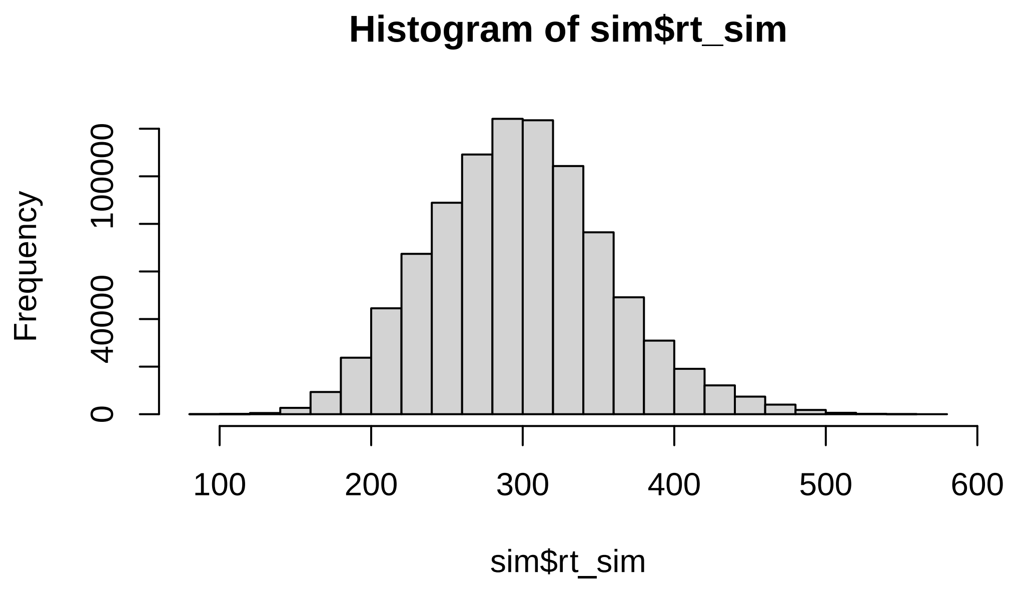
```r
modelo_2f <- cmdstan_model("./Tarea5_2f.stan")
print(modelo_2f)
```

```
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[18] alpha;
  vector[18] beta;
  real<lower=0> sigma;
}

transformed parameters {
  vector[N] media;
  for (i in 1:N){
    media[i] = alpha[subject[i]] + beta[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
  sigma ~ normal(57,25); // propuesta tomando en cuenta los datos pero tratando de que sea po
}

generated quantities {
vector[N] rt_sim; // store post-pred samples
  for (i in 1:N)
    rt_sim[i] = normal_rng(alpha[subject[i]] + beta[subject[i]] * t[i], sigma);
}
```

```
fit2f <- modelo_2f$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 2.3 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 1.7 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 2.2 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 1.7 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 1.9 seconds.

All 5 chains finished successfully.
Mean chain execution time: 2.0 seconds.
Total execution time: 11.0 seconds.
```

```r
sim <- fit2f$draws(c("rt_sim"), format = "df") %>%
  as_tibble() %>%
  pivot_longer(
    cols = starts_with("rt_sim"),
    names_to = "n",
    names_prefix = "rt_sim",
    values_to = "rt_sim")  %>% select(rt_sim)
hist(sim$rt_sim)
```



g. Conjunto de train y test

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.

```r
sleep_train <- sleep %>% filter(Subject<18) #entrenamiento
sleep_test <- sleep %>% filter(Subject<18)  #prueba
```

```r
data_list <- list(
  N = nrow(sleep_train),
  rt_obs = sleep_train$Reaction,
  t = sleep_train$Days,
  subject = sleep_train$Subject,
  N2 = nrow(sleep_test),
  t2 = sleep_test$Days
)

modelo_2g <- cmdstan_model("./Tarea5_2g.stan")
print(modelo_2g)
```

```
data {
  int<lower=0> N;      // Numero de datos (menos 1, training set)
  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
  int<lower=0> N2;          // observacion 18 o test
  vector<lower=0>[N] t2;    // datos de la persona 18
}

parameters {
  vector[18] alpha;
  vector[18] beta;
  real<lower=0> sigma;
}

transformed parameters {
  vector[N] media;
  for (i in 1:N){
    media[i] = alpha[subject[i]] + beta[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
  sigma ~ normal(57,25); // propuesta tomando en cuenta los datos pero tratando de que sea p
}

generated quantities {
  vector[N2] rt_sim; // store post-pred samples
  real alpha2;
  real beta2;
  alpha2 = normal_rng(0, 250);
  beta2 = normal_rng(0, 250);

  for (i in 1:N2)
    rt_sim[i] = normal_rng(alpha2 + beta2 * t2[i], sigma);
}
```

```r
fit2g <- modelo_2g$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 1.6 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 1.5 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 1.6 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 1.8 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: 1.6 seconds.
Total execution time: 9.1 seconds.
```

h. Modelo jerárquico

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.

```r
modelo_2h <- cmdstan_model("./Tarea5_2h.stan")
print(modelo_2h)
```

```
data {
  int<lower=0> N;      // Numero de datos (menos 1, training set)
  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
  int<lower=0> N2;          // observacion 18 o test
  vector<lower=0>[N] t2;    // datos de la persona 18
}

parameters {
  vector[18] alpha;
  vector[18] beta;
  real a;
  real<lower=0> b;
  real c;
  real<lower=0> d;
  real<lower=0> sigma;
}

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

```
model {
  for (i in 1:N){
    rt_obs[i] ~ normal(media[i], sigma); //modelo dado
  }
  alpha ~ normal(a, b);  // distribucion dada
  beta ~ normal(c, d);   // distribucion dada
  a ~ normal(100,100);
  b ~ cauchy(0,5);
  c ~ normal(10,5);
  d ~ cauchy(0,1);
  sigma ~ normal(57,25); // propuesta tomando en cuenta los datos pero tratando de que sea po

}

generated quantities {
  vector[N2] rt_sim; // store post-pred samples
  real alpha2;
  real beta2;
  alpha2 = normal_rng(a, b);
  beta2 = normal_rng(c, d);

  for (i in 1:N2)
    rt_sim[i] = normal_rng(alpha2 + beta2 * t2[i], sigma);
}
```

```r
fit2h <- modelo_2h$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 1.3 seconds.
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)

Chain 2 Informational Message: The current Metropolis proposal is about to be rejected becaus

Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rtmp7ou

Chain 2 If this warning occurs sporadically, such as for highly constrained variable types li

Chain 2 but if this warning occurs often then your model may be either severely ill-conditior

Chain 2

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 1.2 seconds.
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)

Chain 3 Informational Message: The current Metropolis proposal is about to be rejected becaus

Chain 3 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rtmp7ou

Chain 3 If this warning occurs sporadically, such as for highly constrained variable types li

Chain 3 but if this warning occurs often then your model may be either severely ill-conditior

Chain 3 

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 1.1 seconds.
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)

Chain 4 Informational Message: The current Metropolis proposal is about to be rejected becaus

Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rtmp7ou

Chain 4 If this warning occurs sporadically, such as for highly constrained variable types li

Chain 4 but if this warning occurs often then your model may be either severely ill-conditior

Chain 4

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 1.1 seconds.
Chain 5 Iteration:    1 / 2000 [  0%]  (Warmup)

Chain 5 Informational Message: The current Metropolis proposal is about to be rejected becaus
Chain 5 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rtmp7ou
Chain 5 If this warning occurs sporadically, such as for highly constrained variable types li
Chain 5 but if this warning occurs often then your model may be either severely ill-conditior
Chain 5
Chain 5 Informational Message: The current Metropolis proposal is about to be rejected becaus
Chain 5 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rtmp7ou
Chain 5 If this warning occurs sporadically, such as for highly constrained variable types li
Chain 5 but if this warning occurs often then your model may be either severely ill-conditior
Chain 5
Chain 5 Informational Message: The current Metropolis proposal is about to be rejected becaus
Chain 5 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rtmp7ou
Chain 5 If this warning occurs sporadically, such as for highly constrained variable types li
Chain 5 but if this warning occurs often then your model may be either severely ill-conditior
Chain 5
Chain 5 Informational Message: The current Metropolis proposal is about to be rejected becaus
Chain 5 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in '/tmp/Rtmp7ou
Chain 5 If this warning occurs sporadically, such as for highly constrained variable types li
Chain 5 but if this warning occurs often then your model may be either severely ill-conditior
Chain 5
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 1.1 seconds.

All 5 chains finished successfully.
Mean chain execution time: 1.2 seconds.
Total execution time: 6.9 seconds.
```