SISG Bayes: Key 7

Ken Rice

2026-05-28

Q1

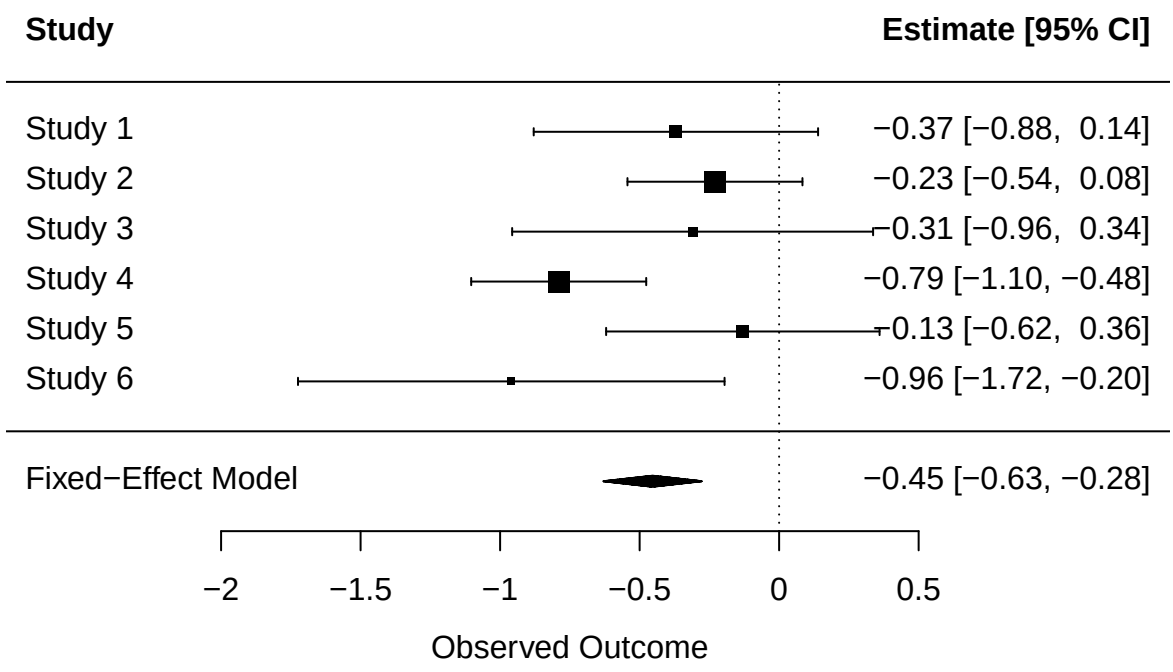
The site also contains an R script fitting a widely-used Bayesian model, in Stan or JAGS, using data from a blood pressure GWAS meta-analysis, combining linear regression estimates across 6 studies. It assumes the sampling model for the observed $\hat{\beta}_i$ is $N(\beta_i, \sigma_i^2)$ with known σ_i^2. The prior for the β_i is specified hierarchically, and (as the data have been suitably scaled) can be written as

$$\begin{aligned} \beta_i &\sim N(\mu, \tau^2) \text{ for each } i, \\ \mu &\sim N(0, 1), \\ \tau &\sim U(0, T), \end{aligned}$$

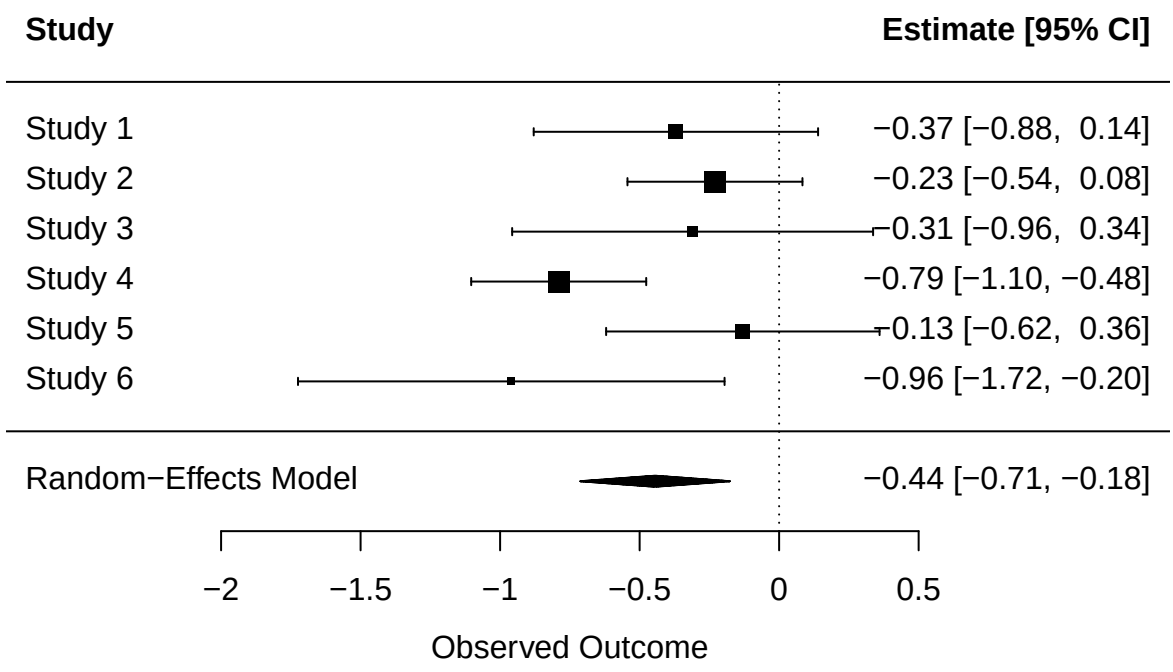
i.e. a flat distribution between 0 and a known, fixed value T. For T=1/5,1 and 5, what is the posterior for $\beta_F = (\sum_{i=1}^6 \beta_i/\sigma_i^2)/(\sum_{i=1}^6 1/\sigma_i^2)$? For the same values of T, what is the posterior for μ, the mean of the random effects distribution?

In the code below, we perform non-Bayesian meta-analysis, and set up the Stan version of the meta-analysis code supplied;

```
study.names <- c("AGES", "ARIC", "CHS", "FHS",
    "RS", "RES")
betahat <- c(-0.37, -0.23, -0.31, -0.79,
    -0.13, -0.96)
stderr <- c(0.26, 0.16, 0.33, 0.16, 0.25,
    0.39)
n <- c(3219, 8047, 3277, 8096, 4737, 1760)

# install.packages('metafor')
library("metafor")
# Non-Bayesian meta-analysis, default
# Fixed Effects and Random Effects
# versions:
rma1 <- rma.uni(yi = betahat, vi = stderr^2,
    method = "FE")
rma2 <- rma.uni(yi = betahat, vi = stderr^2,
    method = "DL")
forest(rma1)
```


Study
Estimate [95% CI]
Study 1
−0.37 [−0.88, 0.14]
Study 2
−0.23 [−0.54, 0.08]
Study 3
−0.31 [−0.96, 0.34]
Study 4
−0.79 [−1.10, −0.48]
Study 5
−0.13 [−0.62, 0.36]
Study 6
−0.96 [−1.72, −0.20]
Fixed−Effect Model
−0.45 [−0.63, −0.28]
−2
−1.5
−1
−0.5
0
0.5
Observed Outcome

```
forest(rma2)
```




```
# Setting up Bayes meta in Stan

cat(file = "metaanalysis.stan", "
data {
  int p; //the number of studies
  vector[p] betahat; // study-specific estimates
  vector[p] sigma; // study-specific standard errors
  real T; // upper bound of uniform prior on tau
}
parameters {
  vector[p] beta; // study-specific parameters
  real mu; // mean of the mixing distribution for beta[i]'s
  real tau; // SD of the mixing distribution
}
transformed parameters {
  vector[p] invsigma2;
  vector[p] betafpart;
  real betaf;
  for(i in 1:p){ invsigma2[i] = pow(sigma[i],-2);}
  for(i in 1:p){ betafpart[i] = beta[i]*invsigma2[i];}
  betaf = sum(betafpart[1:p])/sum(invsigma2[1:p]); // precision-weighted mean of study-specific effects
}
model {
  for(i in 1:p){
\tbetahat[i] ~ normal(beta[i], sigma[i]);
```

```
\tbeta[i] ~ normal(mu, tau);
   }
   mu ~ normal(0,1);
   tau ~ uniform(0,T);
}
")
```

Now to actually do the MCMC, storing the results, for each value of T:

```
library("rstan")
stan.meta1 <- stan(file = "metaanalysis.stan",
    data = list(betahat = betahat, sigma = stderr,
        p = 6, T = 1), iter = 1e+05, chains = 1,
    pars = c("betaf", "mu"))
## Running "C:/PROGRA~1/R/R-46~1.0/bin/x64/Rcmd.exe" SHLIB foo.c
## using C compiler: 'gcc.exe (GCC) 14.3.0'
## gcc  -I"C:/PROGRA~1/R/R-46~1.0/include" -DNDEBUG   -I"C:/Users/kenrice/AppData/Local/R/win-library/4
## cc1.exe: warning: command-line option '-std=c++14' is valid for C++/ObjC++ but not for C
## In file included from C:/Users/kenrice/AppData/Local/R/win-library/4.6/RcppEigen/include/Eigen/Core:
##                  from C:/Users/kenrice/AppData/Local/R/win-library/4.6/RcppEigen/include/Eigen/Dense
##                  from C:/Users/kenrice/AppData/Local/R/win-library/4.6/StanHeaders/include/stan/math
##                  from <command-line>:
## C:/Users/kenrice/AppData/Local/R/win-library/4.6/RcppEigen/include/Eigen/src/Core/util/Macros.h:679:
##   679 | #include <cmath>
##       |          ^~~~~~
## compilation terminated.
## make: *** [C:/PROGRA~1/R/R-46~1.0/etc/x64/Makeconf:297: foo.o] Error 1
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.61 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 1: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 1: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 1: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 1: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 1: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 1: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 1: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 1: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 1: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 1: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 1: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 11.912 seconds (Warm-up)
## Chain 1:                16.237 seconds (Sampling)
## Chain 1:                28.149 seconds (Total)
## Chain 1:
stan.meta0.2 <- stan(file = "metaanalysis.stan",
```

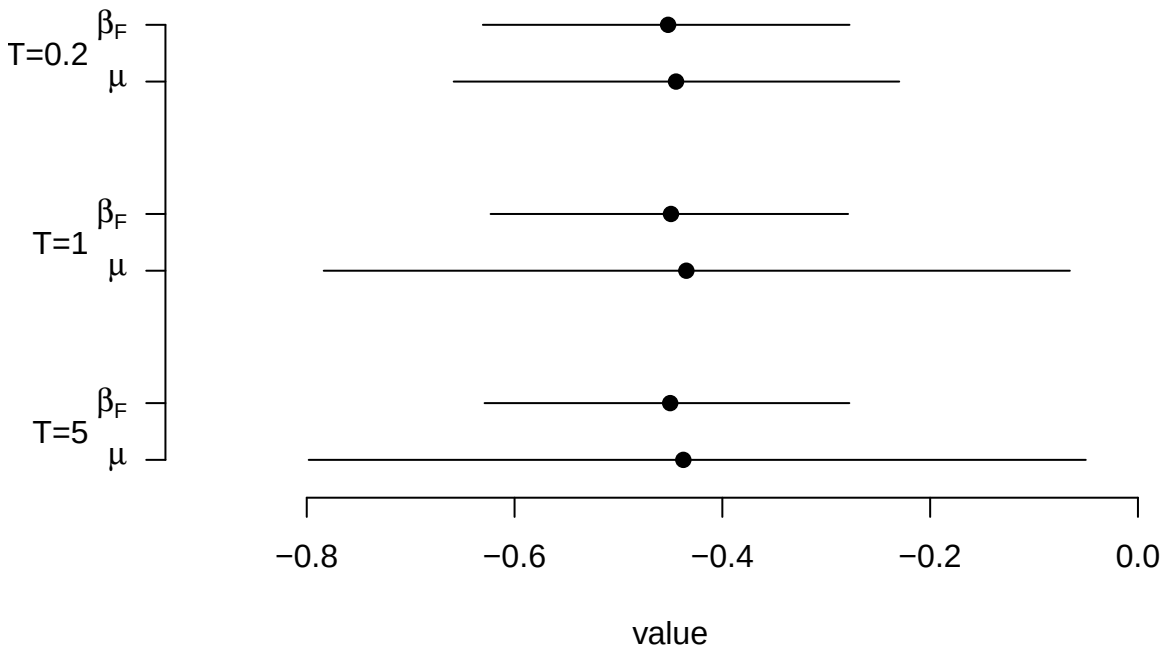
```
    data = list(betahat = betahat, sigma = stderr,
        p = 6, T = 0.2), iter = 1e+05, chains = 1,
    pars = c("betaf", "mu"))
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -1.84353, but must be positive! (in 'string', li
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -1.3974, but must be positive! (in 'string', lin
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.719362, but must be positive! (in 'string', l
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -1.17709, but must be positive! (in 'string', li
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.943987, but must be positive! (in 'string', l
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.3163, but must be positive! (in 'string', lin
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.665836, but must be positive! (in 'string', l
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.955647, but must be positive! (in 'string', l
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.125449, but must be positive! (in 'string', l
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.53059, but must be positive! (in 'string', li
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.797409, but must be positive! (in 'string', l
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.178746, but must be positive! (in 'string', l
```

```
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1: Rejecting initial value:
## Chain 1:   Log probability evaluates to log(0), i.e. negative infinity.
## Chain 1:   Stan can't start sampling from this initial value.
## Chain 1:
## Chain 1: Gradient evaluation took 9e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 1: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 1: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 1: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 1: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 1: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 1: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 1: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 1: Iteration: 70000 / 100000 [ 70%]  (Sampling)
## Chain 1: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 1: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 1: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 4.556 seconds (Warm-up)
## Chain 1:                3.814 seconds (Sampling)
## Chain 1:                8.37 seconds (Total)
## Chain 1:
stan.meta5 <- stan(file = "metaanalysis.stan",
    data = list(betahat = betahat, sigma = stderr,
        p = 6, T = 5), iter = 1e+05, chains = 1,
    pars = c("betaf", "mu"))
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: normal_lpdf: Scale parameter is -0.334952, but must be positive! (in 'string', l
## Chain 1:
## Chain 1: Gradient evaluation took 1.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 100000 [  0%]  (Warmup)
## Chain 1: Iteration: 10000 / 100000 [ 10%]  (Warmup)
## Chain 1: Iteration: 20000 / 100000 [ 20%]  (Warmup)
## Chain 1: Iteration: 30000 / 100000 [ 30%]  (Warmup)
## Chain 1: Iteration: 40000 / 100000 [ 40%]  (Warmup)
## Chain 1: Iteration: 50000 / 100000 [ 50%]  (Warmup)
## Chain 1: Iteration: 50001 / 100000 [ 50%]  (Sampling)
## Chain 1: Iteration: 60000 / 100000 [ 60%]  (Sampling)
## Chain 1: Iteration: 70000 / 100000 [ 70%]  (Sampling)
```

```
## Chain 1: Iteration: 80000 / 100000 [ 80%]  (Sampling)
## Chain 1: Iteration: 90000 / 100000 [ 90%]  (Sampling)
## Chain 1: Iteration: 100000 / 100000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 12.159 seconds (Warm-up)
## Chain 1:                17.205 seconds (Sampling)
## Chain 1:                29.364 seconds (Total)
## Chain 1:

print(stan.meta1)
## Inference for Stan model: anon_model.
## 1 chains, each with iter=1e+05; warmup=50000; thin=1;
## post-warmup draws per chain=50000, total post-warmup draws=50000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## betaf -0.45    0.00 0.09 -0.62 -0.51 -0.45 -0.39 -0.28 12381    1
## mu    -0.43    0.00 0.18 -0.78 -0.54 -0.43 -0.34 -0.07 17569    1
## lp__   2.50    0.17 3.45 -4.02  0.30  2.42  4.49 10.22   428    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 28 13:12:40 2026.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
print(stan.meta0.2)
## Inference for Stan model: anon_model.
## 1 chains, each with iter=1e+05; warmup=50000; thin=1;
## post-warmup draws per chain=50000, total post-warmup draws=50000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## betaf -0.45    0.00 0.09 -0.63 -0.51 -0.45 -0.39 -0.28  4466    1
## mu    -0.44    0.00 0.11 -0.66 -0.52 -0.44 -0.37 -0.23  4699    1
## lp__   6.08    0.11 3.38  0.37  4.06  5.66  7.53 14.69  1013    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 28 13:12:49 2026.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
print(stan.meta5)
## Inference for Stan model: anon_model.
## 1 chains, each with iter=1e+05; warmup=50000; thin=1;
## post-warmup draws per chain=50000, total post-warmup draws=50000.
##
##        mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## betaf -0.45    0.00 0.09 -0.63 -0.51 -0.45 -0.39 -0.28  8509    1
## mu    -0.43    0.00 0.19 -0.80 -0.54 -0.44 -0.33 -0.05 16755    1
## lp__   2.36    0.09 3.57 -4.64  0.16  2.35  4.43 10.22  1590    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 28 13:13:19 2026.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

And writing some code to plot the credible intervals:

```
do.one <- function(mmm) {
    mypos <- 6:1 + c(-1, 1, -1, 1, -1, 1) *
        0.2
    plot(0, 0, xlim = c(-0.9, 0), ylim = c(1,
        6), axes = FALSE, xlab = "value",
        ylab = "")
    segments(mmm[, 2], mypos, mmm[, 3], mypos)
    points(y = mypos, x = mmm[, 1], pch = 19)
    axis(side = 1)
    axis(side = 2, las = 1, at = mypos, c(expression(beta[F]),
        expression(mu), expression(beta[F]),
        expression(mu), expression(beta[F]),
        expression(mu)))
    mtext(side = 2, las = 1, line = 2, at = c(5.5,
        3.5, 1.5), c("T=0.2", "T=1", "T=5"))
}

# par(mar=c(4,6,0,0)+0.2)
do.one(rbind(summary(stan.meta0.2)$summary[1:2,
    c(6, 4, 8)], summary(stan.meta1)$summary[1:2,
    c(6, 4, 8)], summary(stan.meta5)$summary[1:2,
    c(6, 4, 8)]))
```



Compare these to the default non-Bayesian meta-analysis' estimates of β_F (fixed effects) and μ (random effects)

```
round(unlist(rma1[2:7]), 2)
##  beta    se  zval  pval ci.lb ci.ub
## -0.45  0.09 -5.07  0.00 -0.63 -0.28
round(unlist(rma2[2:7]), 2)
##  beta    se  zval  pval ci.lb ci.ub
## -0.44  0.14 -3.26  0.00 -0.71 -0.18
```

We see that, with respect to the prior on τ, inference on β_F is *much* more stable than inference on μ.

Q2

Using each of the same priors as in Q1, what is the marginal prior for any individual β_i? For the same values of T what is the marginal prior for any $\beta_i - \beta_j$, i.e. the difference between two study parameters? Do any differences surprise you?

Hint: for Q2 use the Monte Carlo method in base R: take a large sample from the prior, and make a histogram with many bars.

The code below, for a specified value of T, samples a million observations from the prior for β_i and β_j and makes the histograms suggested; of the distributions β_i and also $\beta_i - \beta_j$.

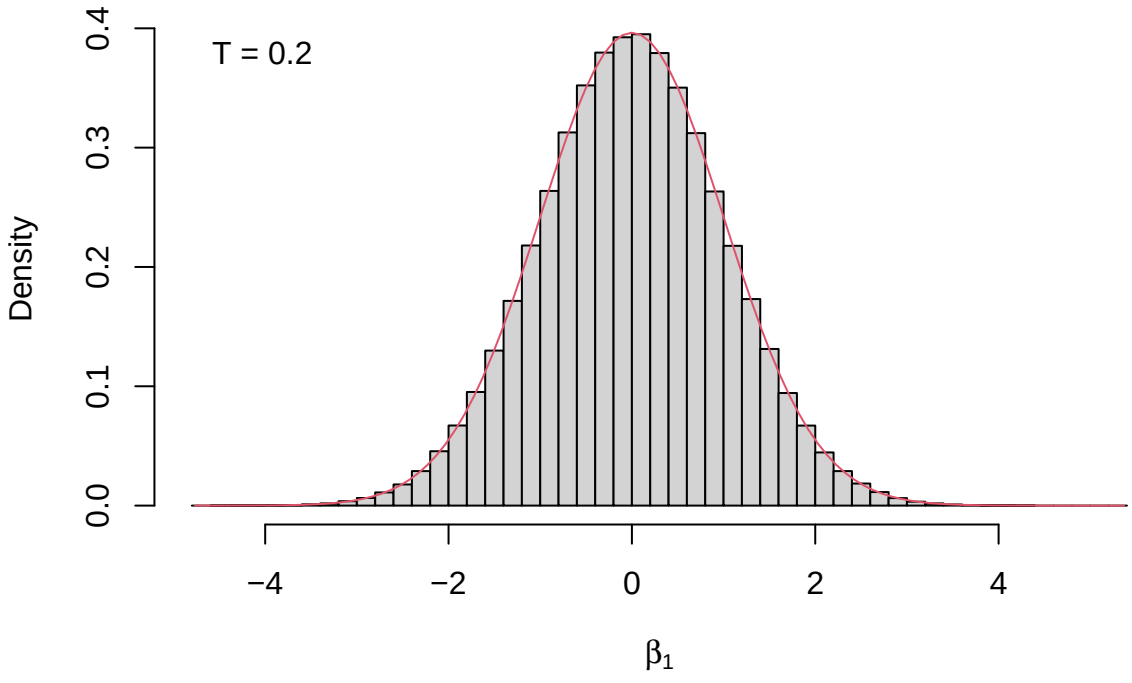
To show it can be done in this setting, we also give numeric integration code to do the same thing – working out the marginal distribution of β_i and $\beta_i - \beta_j$, i.e. averaging over the prior on μ, τ.

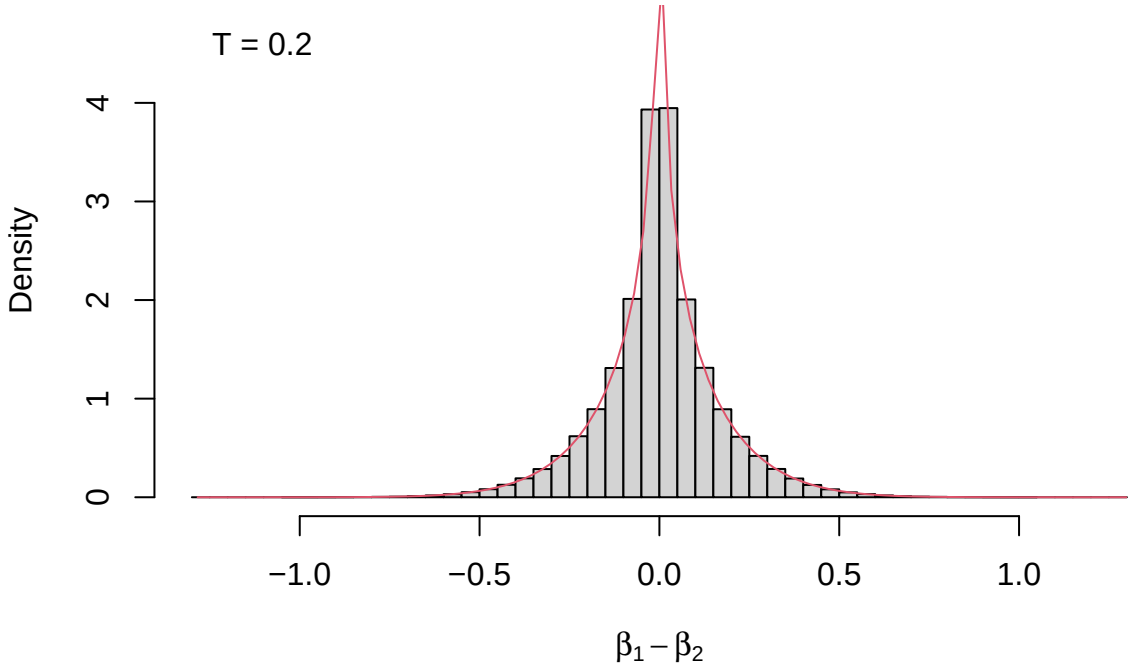
```
par(mfrow = c(3, 2))
par(mar = c(4, 4, 1, 1))
B <- 1e+06
set.seed(4)
do.one <- function(T) {
    mu <- rnorm(B, 0, 1)
    tau <- runif(B, 0, T)
    beta1 <- rnorm(B, mean = mu, sd = tau)
    beta2 <- rnorm(B, mean = mu, sd = tau)
    v1 <- function(x) {
        integrate(function(t) {
            dnorm(x, 0, sqrt(1 + t^2)) *
                dunif(t, 0, T)
        }, 0, T)$value
    }
    v1 <- Vectorize(v1)
    hist(beta1, xlab = expression(beta[1]),
        main = "", freq = FALSE, ylim = c(0,
            v1(0)), n = 60)
    # \tcurve(dnorm(x,0,sqrt(1+T^2)),
    # min(beta1), max(beta1), add=TRUE,
    # col=4)
    curve(v1, min(beta1), max(beta1), add = TRUE,
        col = 2)
    legend("topleft", bty = "n", lty = 0,
        paste("T =", T))

    v2 <- function(x) {
        integrate(function(t) {
```

```
            dnorm(x, 0, sqrt(2 * t^2)) *
                dunif(t, 0, T)
        }, 0, T)$value
    }
    v2 <- Vectorize(v2)
    hist(beta1 - beta2, xlab = expression(beta[1] -
        beta[2]), main = "", freq = FALSE,
        ylim = c(0, v2(0.01)), n = 60)
    # \tcurve(dnorm(x,0,sqrt(2*T^2)),
    # min(beta1-beta2),
    # max(beta1-beta2), add=TRUE,
    # col=4)
    curve(v2, min(beta1 - beta2), max(beta1 -
        beta2), add = TRUE, col = 2)
    legend("topleft", bty = "n", lty = 0,
        paste("T =", T))
}
```

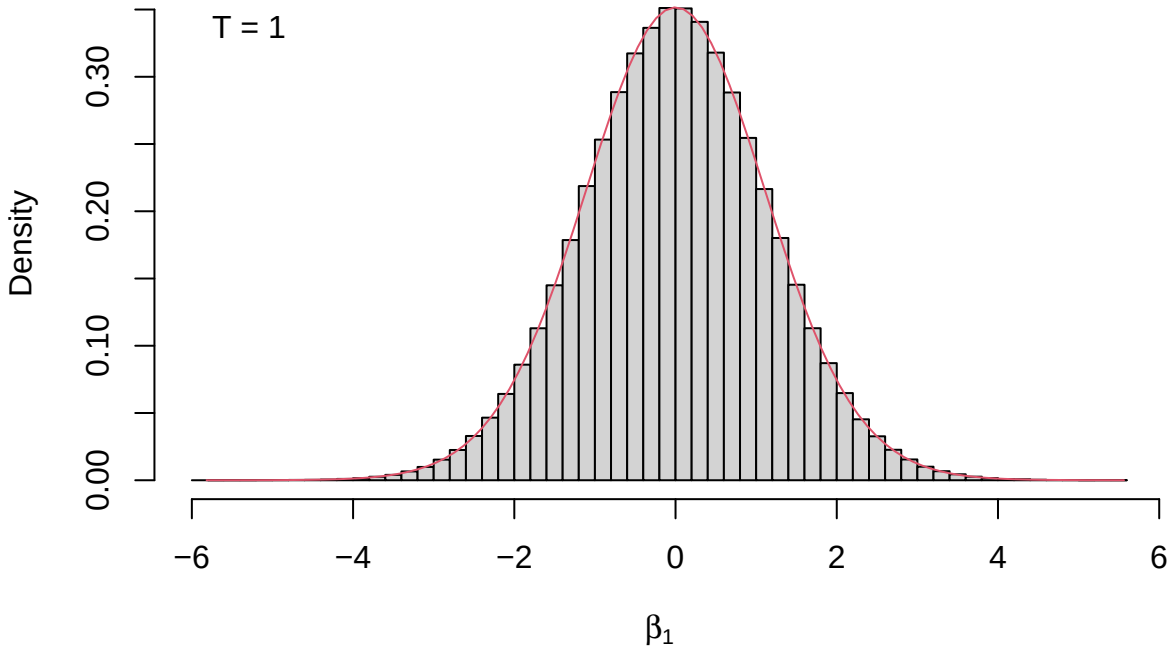
And running the code for the three values of T:

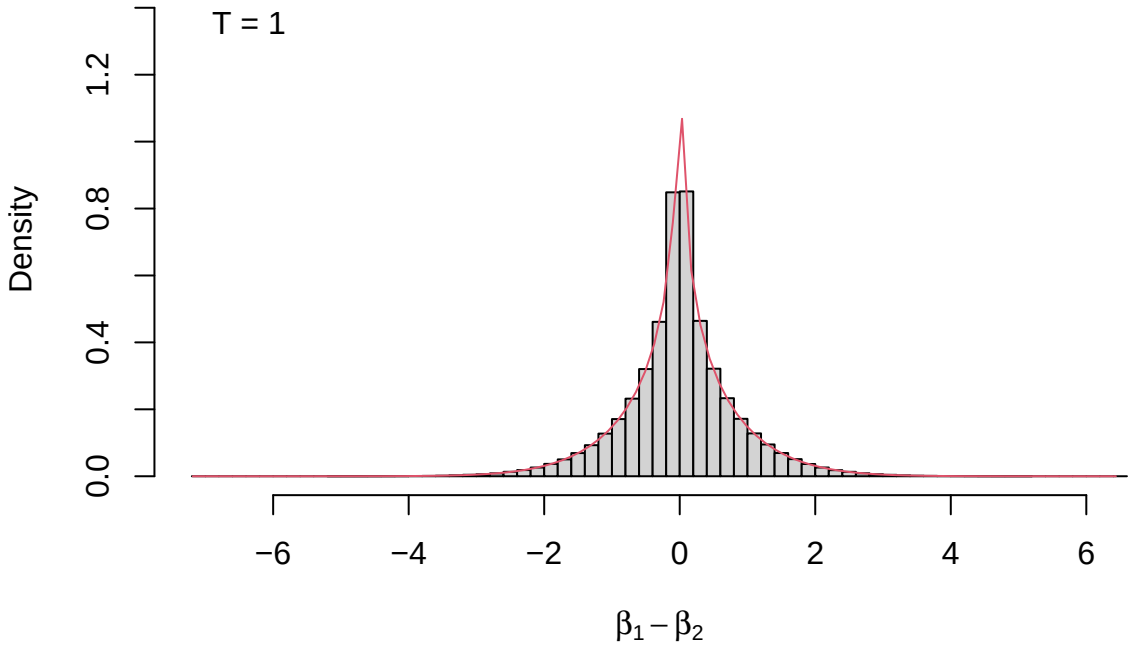
```
do.one(1/5)
```



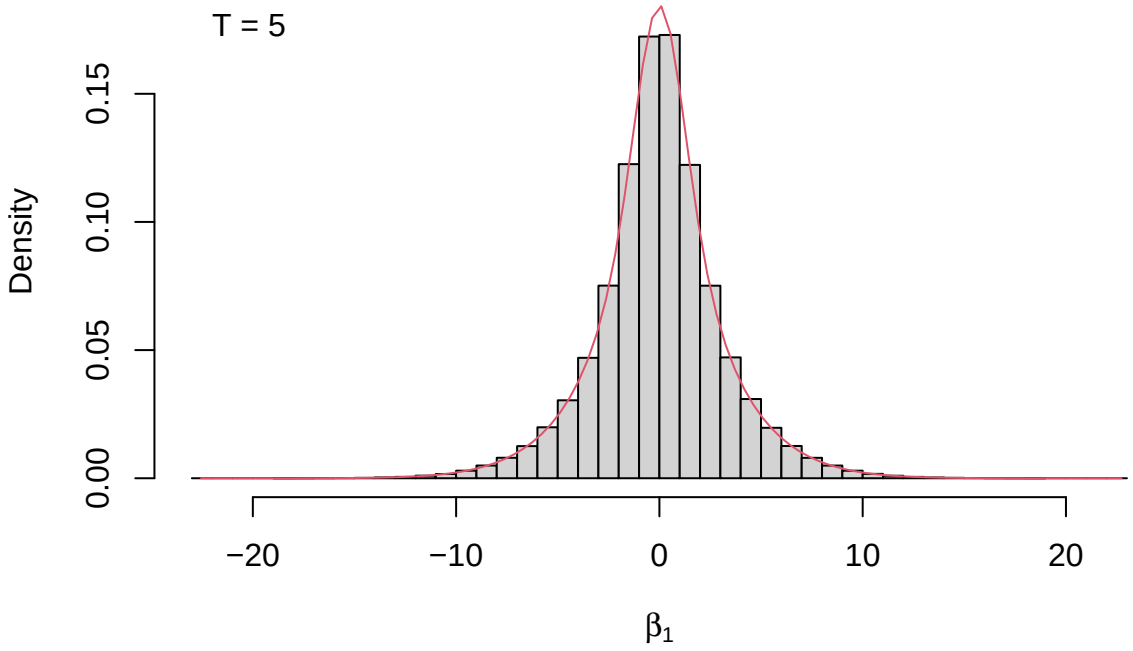


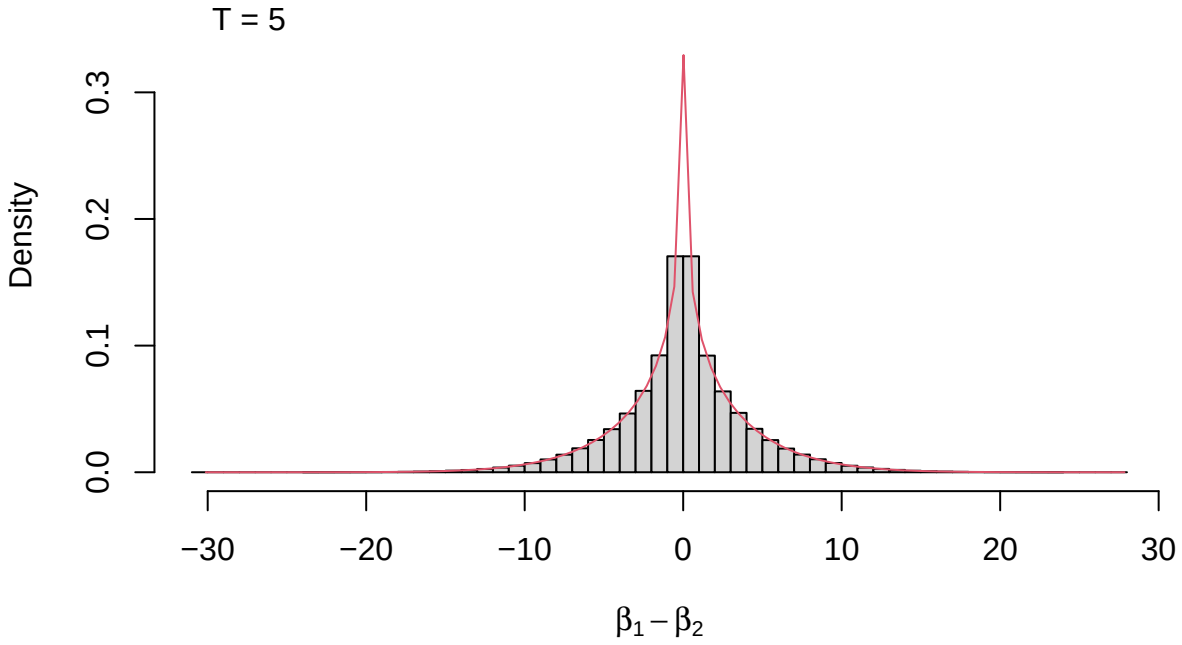

```
do.one(1)
```


T = 1
Density
0.00
0.10
0.20
0.30
−6
−4
−2
0
2
4
6
β1




```
do.one(5)
```


T = 5
Density
0.00
0.05
0.10
0.15
−20
−10
0
10
20
β1



The marginal prior on each β_i is very close to Normal for small T, and slightly heavier-tailed at larger T. The impact on the *difference* $\beta_i - \beta - j$ is more noticeable: the marginal prior is strongly peaked near zero, particularly for large T, with a non-smooth mode there.

This may seem non-intuitive, but note that the prior on τ indicates quite strong support for all the β_i being very close ($\tau \approx 0$) and also strong support for them all being far apart ($\tau \gg 0$) – so it follows that the prior on $\beta_i - \beta_j$ will have something like a spike near zero, and a heavy tail on both sides.