SISG Bayes: Key 5

Ken Rice

2025-06-05

Q1

Using the script on the class site, install the rstan package and run the FTO example from class. Make sure you are using R version 4.3 or higher. Windows users: you will need to install the Rtools toolchain, which may take some time.

Do check with us if this doesn't work on your machine!

Q2

Using the model and code in Q1, explore the impact of the prior on σ^2 in the FTO example. What happens to the posterior intervals for the β_j parameters if the prior for σ^2 gives more support to smaller values? Or to larger values?

Hint: use the Monte Carlo method from Session 2 to check what prior on σ^2 is given by different values of ν_0, σ_0.

If you have trouble with the first part - getting rstan to run - please ask for help, as this package will be used again in some later sessions.

For the second part, we illustrate the posteriors changing the "shape" parameter of the Gamma prior on $1/\sigma^2$. We use shape=0.5, 5 and 50, and show the impact on the prior for σ^2 and $1/\sigma^2$ – obtained using simple Monte Carlo methods:

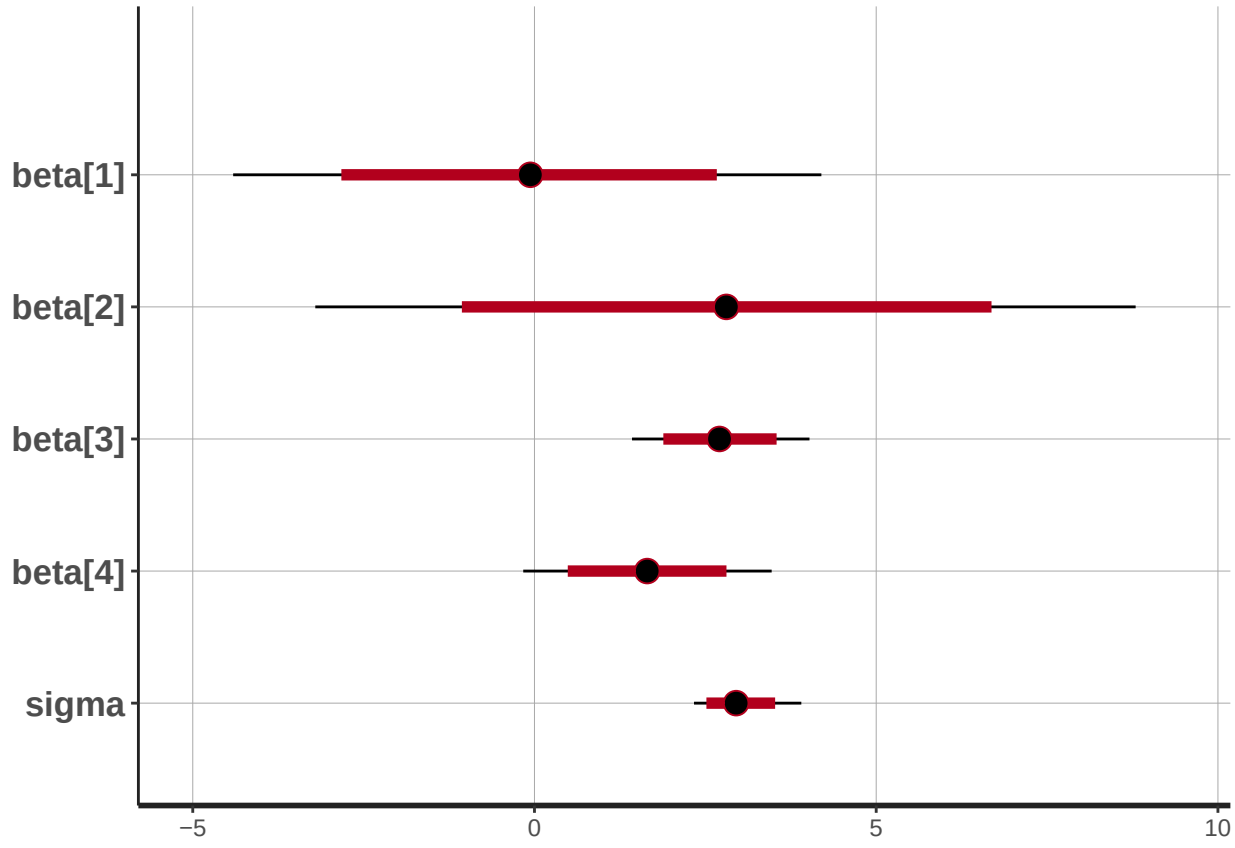
```
# Question 1 - most of the code cut-and-pasted from class example:

# install.packages("rstan")
#setwd("C:/Users/kenrice/Desktop/SISGBayes")
load("yX_FTO.Rdata")
y <- yX$y
X <- yX$X
n <- nrow(X)
p <- ncol(X)

cat(file="FTOexample.stan", "
data {
  int n; //the number of observations
  int p; //the number of columns in the model matrix
  real y[n]; //the response
  matrix[n,p] X; //the model matrix
  real g; // Zellner scale factor
```

```
  vector[p] mu; // Zellner prior mean (all zeros)
  matrix[p,p] XtXinv; // information matrix
}
parameters {
  vector[p] beta; //the regression parameters
  real invsigma2; //the precision, a.k.a. inverse-variance
}
transformed parameters {
  vector[n] linpred;
  cov_matrix[p] Sigma;
  real sigma;
  linpred = X*beta;
  sigma = 1/sqrt(invsigma2);
  for (j in 1:p){
    for (k in 1:p){
      Sigma[j,k] = g*sigma^2*XtXinv[j,k];
      } } }
model {
  beta ~ multi_normal(mu, Sigma);
  y ~ normal(linpred, sigma);
  invsigma2 ~ gamma(5, 1.839); // change the first argument here, can be 0.5/5/50
}
")
# do the MCMC, store the results
library("rstan")
stan2 <- stan(file = "FTOexample.stan",
data = list(n=n,p=p, y=y, X=X, g=n, mu=rep(0,p), XtXinv=solve(crossprod(X)) ),
iter = 100000, chains = 1, pars=c("beta","sigma"))
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000129 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.29 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: 11.853 seconds (Warm-up)
## Chain 1:                12.267 seconds (Sampling)
## Chain 1:                24.12 seconds (Total)
## Chain 1:
```

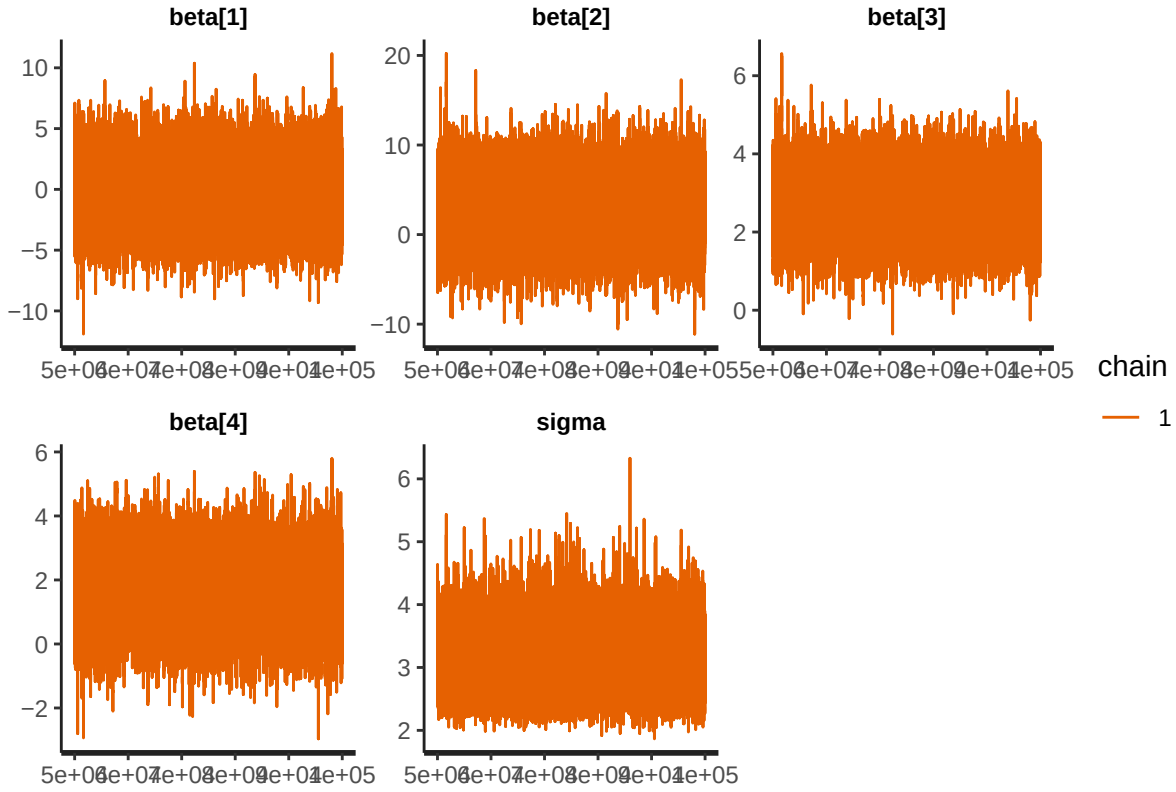
```r
print(stan2)
```

```
## 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
## beta[1]  -0.08    0.02 2.18  -4.41  -1.50  -0.06   1.35   4.20 13443    1
## beta[2]   2.81    0.03 3.06  -3.21   0.79   2.81   4.80   8.80 13457    1
## beta[3]   2.71    0.01 0.66   1.43   2.28   2.71   3.14   4.02 13741    1
## beta[4]   1.65    0.01 0.92  -0.16   1.04   1.65   2.25   3.47 13575    1
## sigma     2.99    0.00 0.40   2.33   2.71   2.95   3.23   3.90 21442    1
## lp__    -52.49    0.01 1.63 -56.53 -53.32 -52.15 -51.30 -50.35 16560    1
##
## Samples were drawn using NUTS(diag_e) at Thu Jun  5 14:39:19 2025.
## 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).
```

```r
plot(stan2)
```




```r
traceplot(stan2)
```

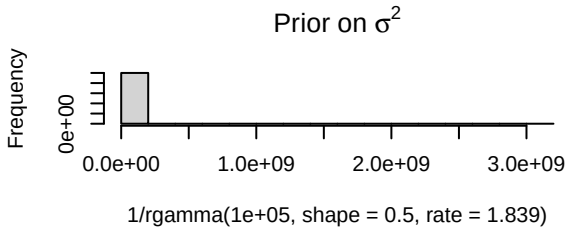


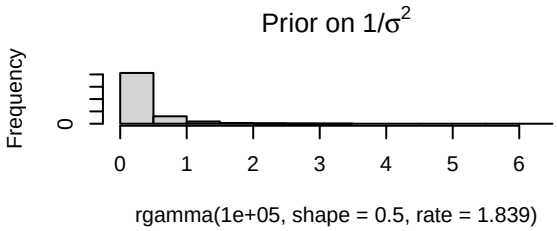

```
stan3 <- stan(file = "FTOexample.stan",
data = list(n=n,p=p, y=y, X=X, g=n, mu=rep(0,p), XtXinv=solve(crossprod(X)) ),
iter = 100000, chains = 1, pars=c("beta","sigma"))
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.37 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: 11.415 seconds (Warm-up)
```

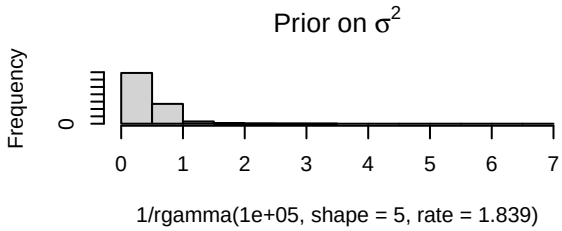
```
## Chain 1:                 13.253 seconds (Sampling)
## Chain 1:                 24.668 seconds (Total)
## Chain 1:
```

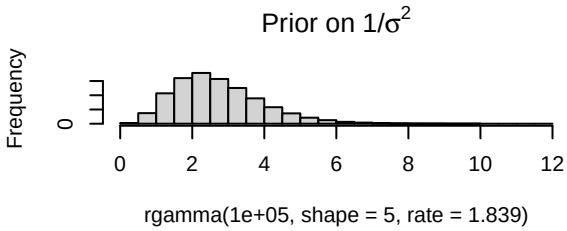
```
stan4 <- stan(file = "FTOexample.stan",
data = list(n=n,p=p, y=y, X=X, g=n, mu=rep(0,p), XtXinv=solve(crossprod(X)) ),
iter = 100000, chains = 1, pars=c("beta","sigma"))
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: model3770773f4bd7__namespace::log_prob: Sigma is not symmetric. Sigma[1,2] = nan
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: model3770773f4bd7__namespace::log_prob: Sigma is not symmetric. Sigma[1,2] = nan
## Chain 1:
## Chain 1: Gradient evaluation took 1.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.17 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: 6.446 seconds (Warm-up)
## Chain 1:                3.733 seconds (Sampling)
## Chain 1:                10.179 seconds (Total)
## Chain 1:
```

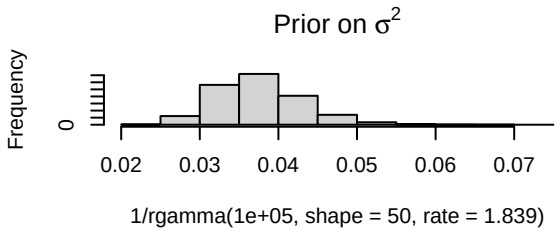
```
par(mfrow=c(3,2))
hist(1/rgamma(1E5, shape=0.5, rate=1.839), main=expression("Prior on "*sigma^2))
hist(rgamma(1E5, shape=0.5, rate=1.839), main=expression("Prior on 1/"*sigma^2))

hist(1/rgamma(1E5, shape=5, rate=1.839), main=expression("Prior on "*sigma^2))
hist(rgamma(1E5, shape=5, rate=1.839), main=expression("Prior on 1/"*sigma^2))

hist(1/rgamma(1E5, shape=50, rate=1.839), main=expression("Prior on "*sigma^2))
hist(rgamma(1E5, shape=50, rate=1.839), main=expression("Prior on 1/"*sigma^2))
```

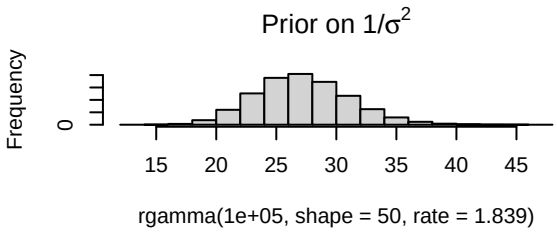










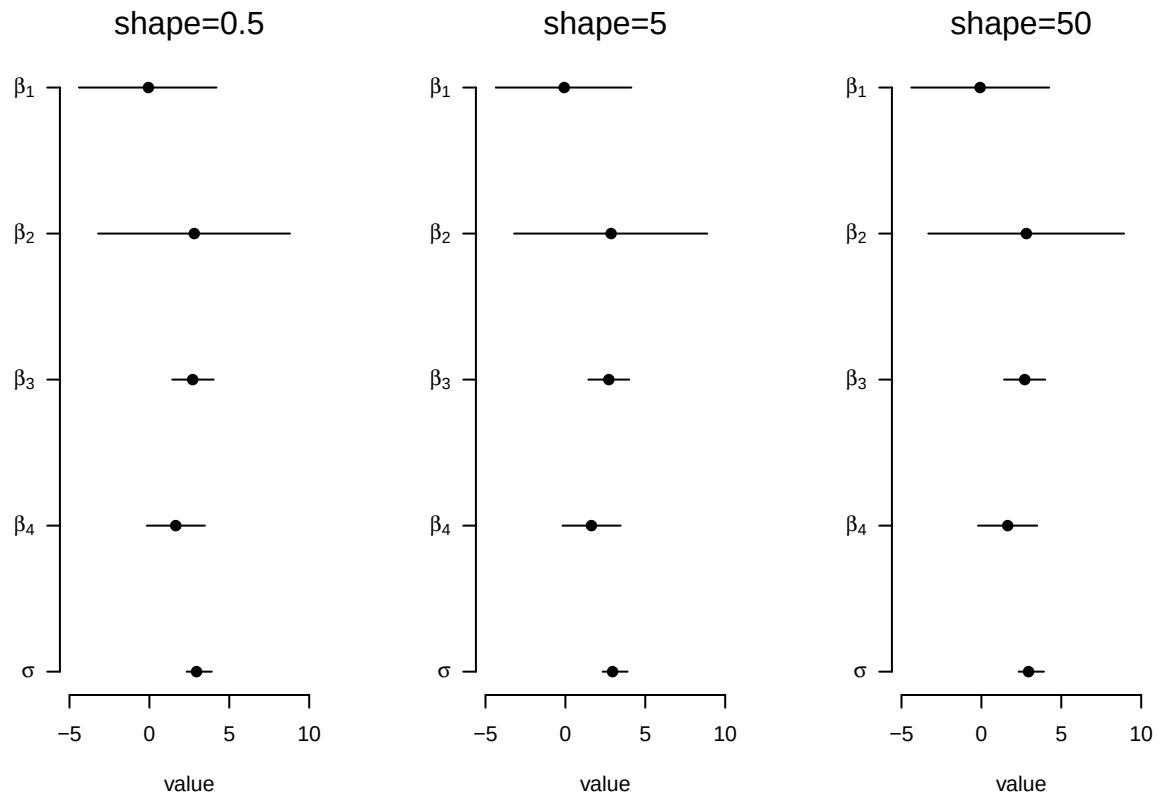



```
#tables of what we'll plot
signif(summary(stan2)$summary[1:5,c(6,4,8)],2)
##           50%  2.5% 97.5%
## beta[1] -0.06 -4.40   4.2
## beta[2]  2.80 -3.20   8.8
## beta[3]  2.70  1.40   4.0
## beta[4]  1.60 -0.16   3.5
## sigma    3.00  2.30   3.9
```

```
signif(summary(stan3)$summary[1:5,c(6,4,8)],2)
##            50%  2.5% 97.5%
## beta[1] -0.074 -4.40   4.1
## beta[2]  2.900 -3.20   8.9
## beta[3]  2.700  1.40   4.0
## beta[4]  1.600 -0.18   3.5
## sigma    3.000  2.30   3.9
```

```
signif(summary(stan4)$summary[1:5,c(6,4,8)],2)
##            50%  2.5% 97.5%
## beta[1] -0.081 -4.40   4.2
## beta[2]  2.800 -3.30   8.9
## beta[3]  2.700  1.40   4.0
## beta[4]  1.600 -0.21   3.5
## sigma    3.000  2.30   3.9
```

```r
# plotting function
do.one <- function(mmm){
    plot(0,0,xlim=c(-5,10), ylim=c(1, 5), axes=FALSE, xlab="value", ylab="")
    segments(mmm[,2], 5:1, mmm[,3], 5:1)
    points(y=5:1, x=mmm[,1], pch=19)
    axis(side=1)
    axis(side=2, las=1, at=5:1, c(
        expression(beta[1]),
        expression(beta[2]),
        expression(beta[3]),
        expression(beta[4]),
        expression(sigma))
    )
    }

#make the plots
par(mfrow=c(1,3))
do.one(summary(stan2)$summary[1:5,c(6,4,8)])
mtext(side=3, line=1, "shape=0.5")
do.one(summary(stan3)$summary[1:5,c(6,4,8)])
mtext(side=3, line=1, "shape=5")
do.one(summary(stan4)$summary[1:5,c(6,4,8)])
mtext(side=3, line=1, "shape=50")
```



The impact on the posteriors for the regression parameters can be seen from the following plots of their 95%

credible intervals – the posterior medians stay at similar values but the intervals shrink as our prior is more confident that each data point is highly informative.

Q2

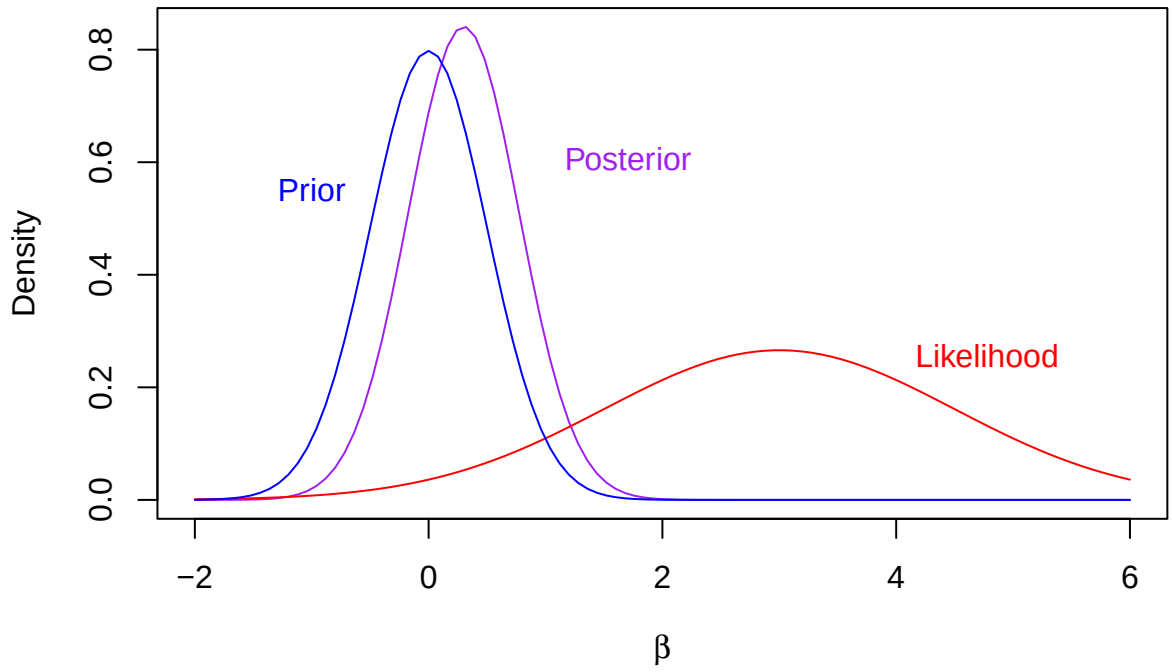
Suppose with a small and noisy study of a parameter β, you obtain an estimate $\hat{\beta} = 3$ with standard error 1.5. (This gives a Z-score of 3/1.5=2, and hence a p-value just below 0.05.) Suppose your scientific colleagues have a skeptical $N(0, 0.5^2)$ prior, with very little support for values of β as large as the observed estimate.

Assuming (reasonably) that the likelihood is well-approximated by $N(3, 1.5^2)$, what is the posterior? Do you think your p-value below 0.05 should be the start of a Nobel-worthy revolution of all that your colleagues foolishly believed?

We have prior $N(0, 0.5^2)$ on β, and the likelihood contribution is $N(3, 1.5^2)$. Hence the prior precision is $1/0.5^2 = 4$, and the precision from the likelihood is $1/1.5^2 = 4/9$. The posterior precision is therefore $4 + 4/9 = 40/9$, and the posterior variance is $9/40 = 0.225$. To calculate the posterior mean, we take a weighted average of the prior mean (zero) and the mean from the likelihood. The weight on this value is $4/9/(40/9) = 1/10$, so the posterior mean is $3/10$.

In a picture:

```
curve(dnorm(x, 3/10, sqrt(9/40)), -2,6, col="purple", ylab="Density", xlab=expression(beta))
curve(dnorm(x, 3, 1.5), -2,6, col="red", add=TRUE)
curve(dnorm(x, 0, 0.5), -2,6, col="blue", add=TRUE)

text(x=c(-1, 1, 4), y=c(0.6, 0.6, 0.25), c("Prior","Posterior","Likelihood"),
col=c("blue","purple","red"), pos=c(1,4,4))
```



While your colleagues may be wrong but this data should not convince them of that: the prior information is much stronger than what we get from the data/likelihood. For (rational) people with their knowledge, beliefs still focus on small effects.