SFS 2024 Short Course – Bayesian Applications in Environmental and Ecological Studies with R and Stan

Song S. Qian and Mark R. DuFour

6/1/2024

Why Bayesian?

With Stan, Bayesian inference can be readily applied to more complicated problems that are impossible for classical MLE. More importantly, Bayesian inference using Stan can provide useful information on model formulation to identify potential problems. Bayesian inference is also the best approach for information accumulation. These benefits facilitate an iterative modeling process allowing feedback between the problems of model formulation, estimation, and evaluation.

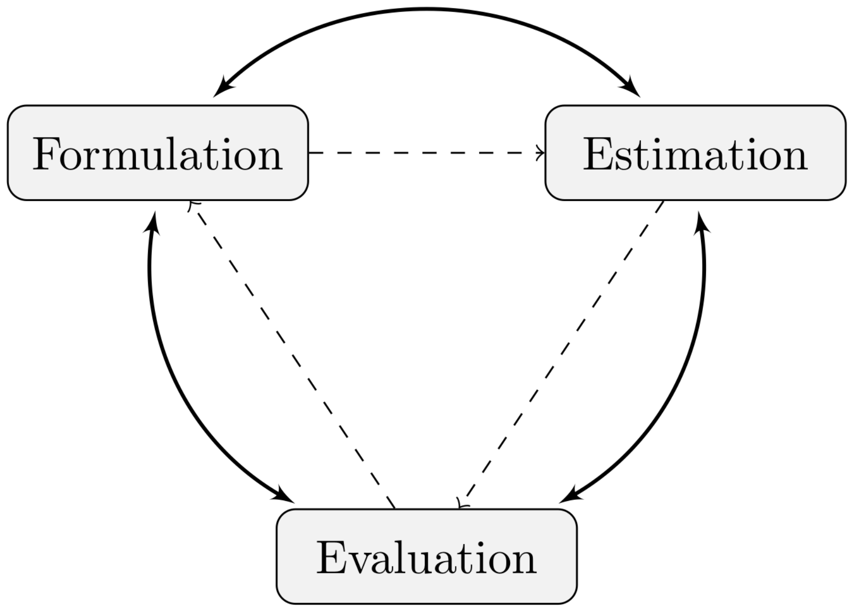


Figure 1: Figure 1.5 of Qian, DuFour, Alammedine, 2022)

Information accumulation

Suppose that we conducted a new survey of the snake fungal disease a year after the initial study to update the prevalence. Or, we may want to evaluate the temporal trend of the disease prevalence in the subsequent years.

All the methods we used resulted in a consistent estimate of the prevalence of about 22% with a wide range of between 0.07-0.4. The uncertainty is a result of the small sample size. Suppose that we now have a new sample of $n_2 = 12$ and $y_2 = 4$. Using classical statistics, we either estimate the prevalence using only the second sample, assuming that the prevalence has changed or combine the data (i.e., $n = 32$ and $y = 9$), assuming the population stays the same.

With a Bayesian approach, we can summarize the posterior of θ from the first sample to form a prior of θ and update the prior using the new data. We typically use a beta distribution to summarize the distribution of a probability. Given the mean and standard deviation of 0.23 and 0.11 (from the Stan model), we propose a beta distribution with parameters $\alpha = 3.52$ and $\beta = 11.56$ (based on the method of moments). The prior can also be from similar studies elsewhere. All we need is an estimate of the mean and a likely range. We can treat the likely range as the usual 95% confidence interval (roughly 4 times standard deviation), from which derive a rough estimate of the standard deviation.

```r
ybar <- mean(stan_out$theta)
s2 <- var(stan_out$theta)
alpha0 <- ybar * (ybar * (1 - ybar)/s2 - 1)
beta0 <- alpha0 * (1 - ybar)/ybar

alpha0 + beta0
```

```
## [1] 15.44927
```

Once the initial estimates of α_0 and β_0 are available, we can further revise the estimate based on our assessment of the relevancy of the prior to our data at hand. For example, the relevancy of the prior may be measured by $\alpha + \beta$ if we interpret the sum as the sample size from which the prior was derived. For example, $\alpha_0 + \beta_0 = 15$, somewhat smaller than the sample size we used (20), perhaps, because of the imperfect detection of the qPCR method. If we want to weight the prior less than 15 data points, we can rescale the prior parameter by setting $\alpha_0 + \beta_0 = n_0$ while keeping $\alpha_0/(\alpha_0 + \beta_0)$ the same (equals to $\bar{y}$).

We now need additional data for the model:

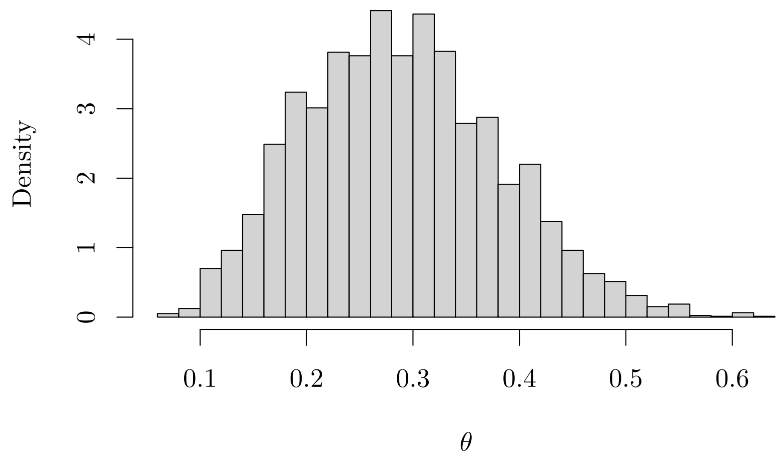
```r
### Stan Code ###
snake_code2 <- "
  data{
    int<lower=1> n;
    int<lower=0> y;
    real<lower=0,upper=1> fn;
    real<lower=0,upper=1> fp;
    real<lower=0> alpha0;
    real<lower=0> beta0;
  }
  parameters{
    real<lower=0,upper=1> theta;
  }
  transformed parameters{
   real<lower=0,upper=1> p_pos;
   p_pos = theta*(1-fn)+(1-theta)*fp;
}
model{
   theta ~ beta(alpha0,beta0);
   target += binomial_lpmf(y | n, p_pos);
}
"

fit2 <- stan_model(model_code = snake_code2)
## save(fit2, file='fit2.RData') load('fit2.RData')
```

Running and processing output:

```r
input_snake2 <- function(n = 12, y = 5, alpha, beta, nchns = nchains) {
    data <- list(y = y, n = n, fp = 0.07, fn = 0.05, alpha0 = alpha, beta0 = beta)
    inits <- list()
    for (i in 1:nchns) inits[[i]] <- list(theta = runif(1))
    pars = c("theta")
    return(list(data = data, inits = inits, pars = pars))
}
input.to.stan <- input_snake2(alpha = alpha0, beta = beta0)
fit2keep <- sampling(fit2, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$pars, iter = niters, thin = nthin, chains = nchains)
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## theta   0.29    0.00 0.09   0.13   0.22   0.28   0.35   0.48  2371    1
## lp__  -10.71    0.01 0.68 -12.71 -10.91 -10.43 -10.25 -10.20  2447    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 13:10:38 2024.
## 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
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta, xlab = "$\\theta$", main = "")
```



```r
quantile(stan_out$theta, prob = c(0.05, 0.95))
```

```
##        5%       95%
## 0.1506828 0.4458456
```

From here, we can evaluate whether the prevalence is increased by calculating the probability that the posterior is larger than the prior:

```r
prior<-rvbeta(1, alpha0, beta0)
Pr(theta>prior)
```

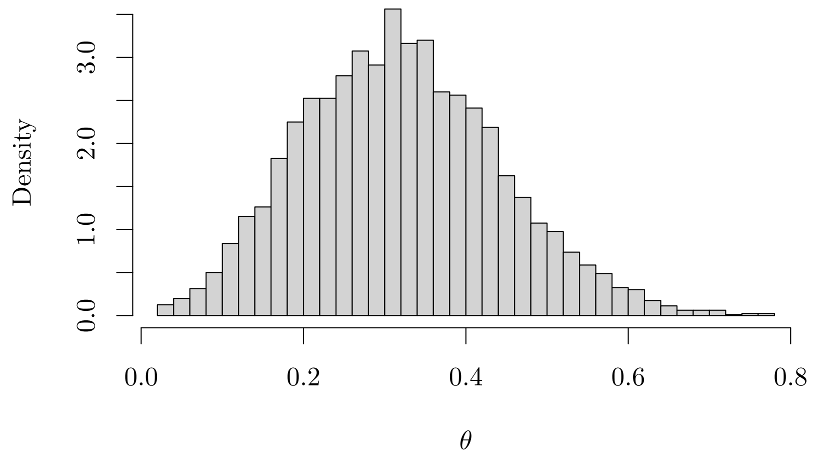
```
## [1] 0.66475
```

If we down weight the prior by setting $\alpha_0 + \beta_0 = 6$, half of the data from the current year, the new prior are based on $n = 12, y = 5$, we simply rerun the model by changing only the input data:

```r
alpha0 <- 6 * ybar
beta0 <- 6 - alpha0

input.to.stan <- input_snake2(n = 12, y = 5, alpha = alpha0, beta = beta0)
fit2keep <- sampling(fit2, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$pars, iter = niters, thin = nthin, chains = nchains)
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##         mean se_mean   sd  2.5%   25%   50%  75% 97.5% n_eff Rhat
## theta  0.32    0.00 0.12  0.10  0.23  0.32  0.4  0.58  2667    1
## lp__  -5.51    0.02 0.82 -7.73 -5.67 -5.19 -5.0 -4.95  2549    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 13:11:08 2024.
## 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
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta, xlab = "$\\theta$", main = "")
```



```r
quantile(stan_out$theta, prob = c(0.05, 0.95))
```

```
##        5%       95%
## 0.1330015 0.5338328
```

```
Pr(theta > prior)
```

```
## [1] 0.7095
```

Realistic modeling

Most likely, we don't know exactly the values of f_p and f_n. But we may have prior knowledge on their likely values, from which we can derive prior distributions of f_p and f_n. For example, the values $f_p = 0.07$ and $f_n = 0.05$ are estimated from similar tests elsewhere based on over 50 tests. We can use the beta distribution to quantify the prior knowledge:

```
a_p <- 0.07 * 50
b_p <- 50 - a_p
a_n <- 0.05 * 50
b_n = 50 - a_n
```

Now the only change needed is to include f_p and f_n as parameters and their the priors:

```
### Stan Code ###
snake_code3 <- "
  data{
    int<lower=1> n;
    int<lower=0> y;
    real<lower=0> alpha0;
    real<lower=0> beta0;
    real<lower=0> a_p;
    real<lower=0> b_p;
    real<lower=0> a_n;
    real<lower=0> b_n;
  }
  parameters{
    real<lower=0,upper=1> theta;
    real<lower=0,upper=1> fn;
    real<lower=0,upper=1> fp;
  }
  transformed parameters{
   real<lower=0,upper=1> p_pos;
   p_pos = theta*(1-fn)+(1-theta)*fp;
}
model{
   theta ~ beta(alpha0,beta0);
   fp ~ beta(a_p, b_p);
   fn ~ beta(a_n, b_n);
   target += binomial_lpmf(y | n, p_pos);
}
"

fit3 <- stan_model(model_code = snake_code3)
## save(fit3, file='fit3.RData') load('fit3.RData')
```

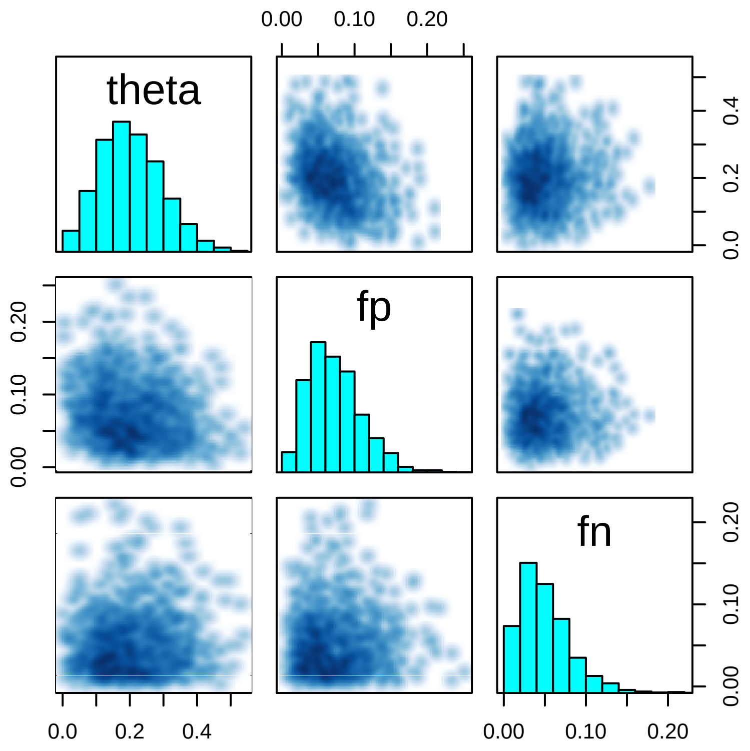
Running and processing output

```
input_snake3 <- function(n = 20, y = 5, alpha, beta, ap, bp, an, bn, nchns = nchains) {
    data <- list(y = y, n = n, fp = 0.07, fn = 0.05, alpha0 = alpha, beta0 = beta,
        a_p = ap, b_p = bp, a_n = an, b_n = bn)
    inits <- list()
    for (i in 1:nchns) inits[[i]] <- list(theta = runif(1))
    pars = c("theta", "fp", "fn")
    return(list(data = data, inits = inits, pars = pars))
}
input.to.stan <- input_snake3(alpha = alpha0, beta = beta0, ap = a_p, an = a_n, bp = b_p,
    bn = b_n)
fit2keep <- sampling(fit3, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$pars, iter = niters, thin = nthin, chains = nchains)
print(fit2keep)
```

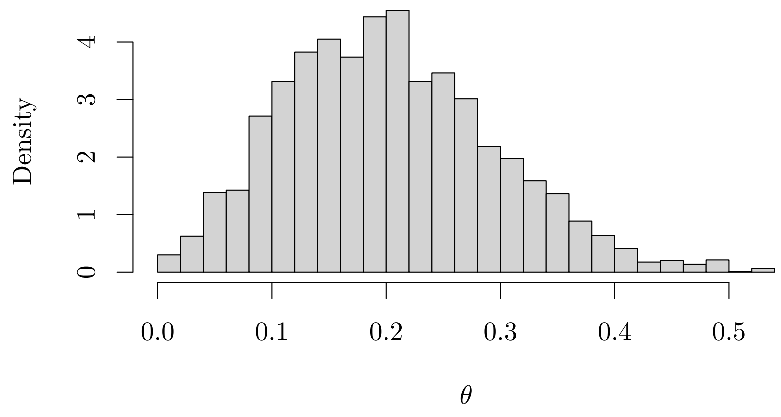
```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## theta   0.20    0.00 0.09   0.04   0.14   0.20   0.26   0.40  2406    1
## fp      0.07    0.00 0.04   0.02   0.05   0.07   0.09   0.16  2420    1
## fn      0.05    0.00 0.03   0.01   0.03   0.04   0.07   0.13  2124    1
## lp__  -29.14    0.03 1.37 -32.64 -29.70 -28.80 -28.15 -27.59  2477    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 13:13:40 2024.
## 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).
```

```
pairs(fit2keep, pars = c("theta", "fp", "fn"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



```
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta, xlab = "$\\theta$", main = "")
```



```
quantile(stan_out$theta, prob = c(0.05, 0.95))
```

```
##          5%        95%
## 0.06101634 0.36929433
```

Identifiability

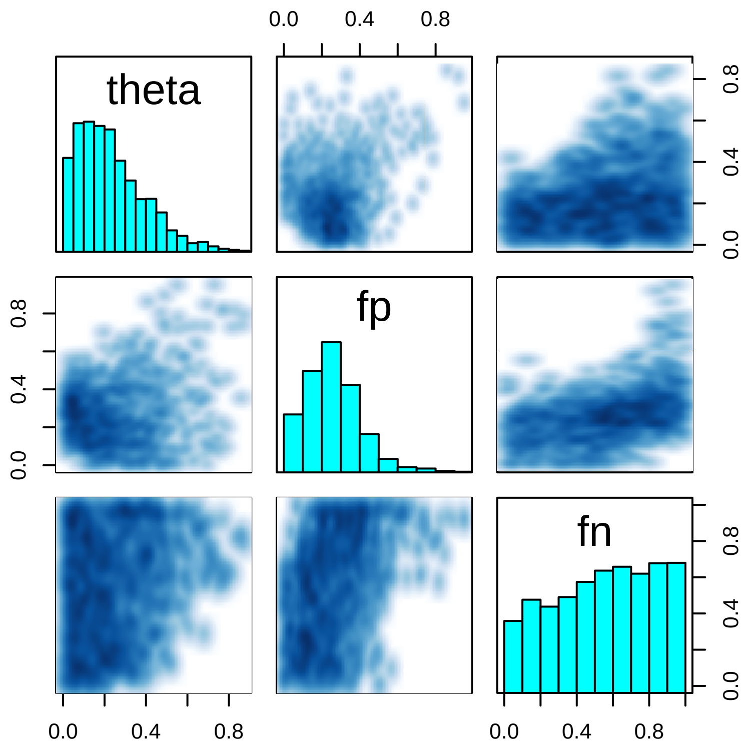
What happens if we have no prior information of f_p and f_n? We could try to use non-informative prior $beta(1,1)$ (uniform between 0 and 1). However, it is unlikely that we can simultaneously identify all three parameters. Mathematically, information in the data is represented in the likelihood function, consisting of products of θf_p and θf_n. In other words, the data have information of the two products. To definitely separate them, we need additional information. If we try because we can, Stan will let us know that something is wrong.

```
input.to.stan <- input_snake3(alpha = alpha0, beta = beta0, ap = 1, an = 1, bp = 1,
    bn = 1)
fit2keep <- sampling(fit3, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$pars, iter = niters, thin = nthin, chains = nchains)
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##        mean se_mean   sd   2.5%    25%   50%   75% 97.5% n_eff Rhat
## theta  0.23    0.00 0.16   0.02   0.11  0.20  0.33  0.63  2551    1
## fp     0.26    0.00 0.15   0.02   0.15  0.24  0.34  0.60  2664    1
## fn     0.55    0.01 0.28   0.05   0.32  0.57  0.79  0.98  2560    1
## lp__  -9.77    0.03 1.41 -13.50 -10.45 -9.43 -8.72 -8.12  2220    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 13:14:10 2024.
## 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).
```

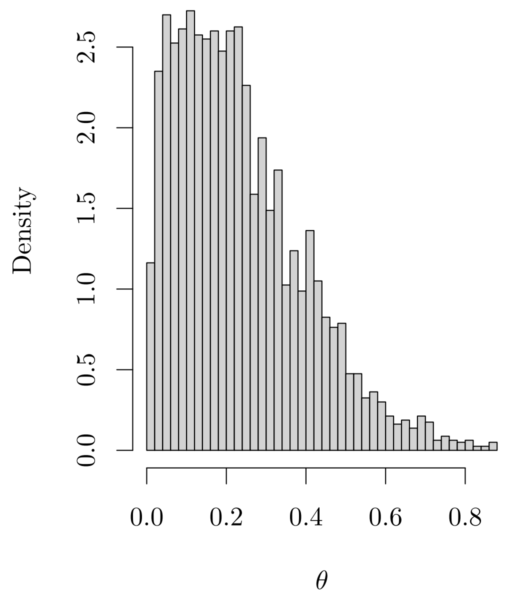
```
pairs(fit2keep, pars = c("theta", "fp", "fn"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



The estimated prevalence is highly uncertain:

```
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta, xlab="$\\theta$", main="")
```



```r
quantile(stan_out$theta, prob=c(0.05,0.95))
```

```
##         5%        95%
## 0.03118636 0.54069398
```

To show the non-identifiability clearly, we increase the sample size from $n = 20$ to $n = 2000$

```r
input.to.stan <- input_snake3(n = 2000, y = 500, alpha = alpha0, beta = beta0, ap = 1,
    an = 1, bp = 1, bn = 1)
fit2keep <- sampling(fit3, data = input.to.stan$data, init = input.to.stan$inits,
    pars = input.to.stan$pars, iter = niters, thin = nthin, chains = nchains)
```

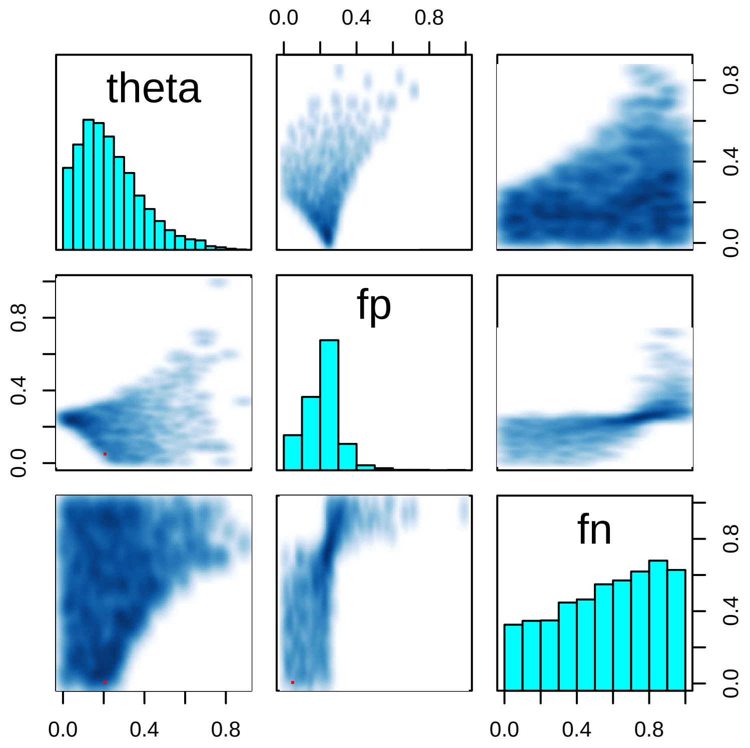
```
## Warning: There were 1 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

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

```r
print(fit2keep)
```

```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## theta   0.23    0.00 0.16   0.02   0.12   0.21   0.32   0.62  1582    1
## fp      0.21    0.00 0.10   0.03   0.16   0.22   0.26   0.42  1778    1
## fn      0.56    0.01 0.28   0.04   0.34   0.60   0.81   0.98  2152    1
## lp__  -12.08    0.03 1.36 -15.52 -12.76 -11.74 -11.08 -10.50  2171    1
##
## Samples were drawn using NUTS(diag_e) at Thu May 30 13:14:49 2024.
## 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).
```

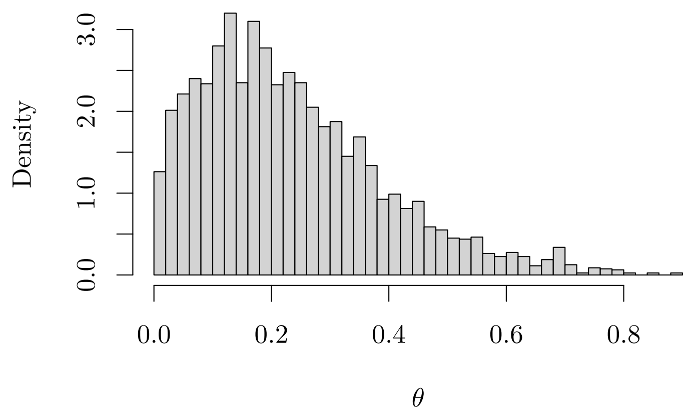
```r
pairs(fit2keep, pars = c("theta", "fp", "fn"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



The estimated prevalence is still highly uncertain:

```r
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta, xlab="$\\theta$", main="")
```



```
quantile(stan_out$theta, prob=c(0.05,0.95))
```

```
##         5%        95%
## 0.03410267 0.53228205
```

Examining the marginal distributions is not enough.

Summary

Using Stan in R is a natural way for implementing Bayesian modeling. Writing the Stan model requires us to explicitly formulate a model, identify parameters of interest, and provide any prior information we have on the parameters. The strong diagnostic feature of Stan provides more information for model evaluation. A typical Bayesian inference workflow starts with writing down the response variable model (what is the likely probability distribution model and how the mean parameter relates to predictors). From the likelihood function, we identify parameters of interest and relevant prior information. We then organize all available information into three basic blocks (data, parameters, and model) to construct the Stan model code. Because the MCMC algorithm in Stan is constructed through a function proportional to the log-posterior density, the log-likelihood function in a Stan model does not have to be based on a known distribution function (e.g., the normal distribution).

Computational Notes – Using Package `rv`

See cran rv vignette