# SFS 2023 Short Course – Bayesian Applications in Environmental and Ecological Studies with R and Stan

Song S. Qian

6/3/2023

## Introduction

Statistical Inference Mode: This mode, proposed by Fisher in 1921, involves three fundamental "problems" in statistical inference:

1. Problem of Formulation: This problem pertains to determining the distribution of the response variable.
2. Problem of Estimation: Here, the focus is on estimating the parameters of the model derived from the problem of formulation once data are obtained.
3. Problem of Distribution: This problem involves evaluating the model's fit and performance.

A Mathematical Summary of Statistical Inference:

- Probability Distribution Density Function: The density function of the response variable, denoted as $\pi(y, \theta)$.
- Joint Distribution: Given observed data $y_1, y_2, \ldots, y_n$, the aim is to determine the joint distribution $\pi(y_1, y_2, \ldots, y_n, \theta)$. This holds true for both classical and Bayesian inference.
- Classical Statistics: In classical statistics, the joint distribution is expressed as $\pi(y_1, \ldots, y_n, \theta) = \pi(y_1, \ldots, y_n \mid \theta)\pi(\theta)$. The likelihood function, denoted as $L(\theta; y)$, corresponds to $\pi(y_1, \ldots, y_n \mid \theta)$ and is used for maximum likelihood estimation (MLE). The sampling distribution provides information about uncertainty.
- Bayesian Statistics: In Bayesian statistics, the joint distribution is written as $\pi(y_1, \ldots, y_n, \theta) = \pi(y_1, \ldots, y_n \mid \theta)\pi(\theta) = \pi(\theta \mid y_1, \ldots, y_n)\pi(y_1, \ldots, y_n)$. The posterior distribution, denoted as $\pi(\theta \mid y_1, \ldots, y_n)$, can be derived using Bayes' theorem: $\pi(\theta \mid y_1, \ldots, y_n) = \frac{\pi(y_1, \ldots, y_n|\theta)\pi(\theta)}{\pi(y_1, \ldots, y_n)} \propto \pi(y_1, \ldots, y_n \mid \theta)\pi(\theta)$. Bayesian inference combines estimation and distribution considerations by considering the posterior distribution. Relationship Between Bayesian Inference and MLE: Bayesian inference can be seen as a generalization of MLE-based classical inference. MLE corresponds to the mode of the posterior distribution when a uniform prior is used.

### From Mathematics to Computation

- Derivative (classical) versus integration (Bayesian)
  - Maximum Likelihood Estimation (MLE): Finding parameter values that maximize the likelihood function, denoted as $\hat{\theta} = arg\ max_{\theta \in \Theta} L(\theta; y)$
  - Bayesian: Calculating the posterior distribution $\pi(\theta \mid y_1, \cdots, y_n)$ using integration: $\pi(\theta \mid y_1, \cdots, y_n) = \frac{\pi(y_1, \cdots, y_n|\theta)\pi(\theta)}{\int \pi(y_1, \cdots, y_n|\theta)\pi(\theta)d\theta}$
- Classical statistics: A collection of efficient numerical algorithms for quantifying MLEs across various classes of models.
  - Historically, results were tabulated.
  - Nowadays, these algorithms are implemented in software packages such as SAS, SPSS, and R.

- Bayesian statistics: Using Monte Carlo simulation to avoid integration
  - Markov chain Monte Carlo simulation (MCMC) algorithm – drawing random numbers of $\theta$ from its density function (or a function proportional to its density function)
  - Evaluation of Posterior Distributions: Posterior distributions of model parameters are evaluated using random samples generated from their posterior distributions.
  - The Bayesian posterior is proportional to the product of the likelihood function and the prior: $L(\theta; y_1, \cdots, y_n)\pi(\theta)$
  - If $y_1, \cdots, y_n$ i.i.d., $L(\theta; y_1, \cdots, y_n) = \prod_{i=1}^{n} L(\theta; y_i)$.
  - If the observations $y_1, \cdots, y_n$ are independent and identically distributed (i.i.d.), the likelihood function becomes the product of individual likelihoods: $L(\theta; y_1, \cdots, y_n) = \prod_{i=1}^{n} L(\theta; y_i)$.
  - The logarithm of the posterior is given by: $\log(\pi(\theta \mid y_1, \cdots, y_n)) = \log(\pi(\theta)) + \sum_{i=1}^{n} \log(L(\theta; y_i))$
- `Stan` via `rstan` in R:
  - The `Stan` program implements the MCMC algorithm when the logarithm of the prior $\log(\pi(\theta))$ and the logarithm of the likelihood function $\log(L(\theta; y_i))$ are provided.

**Example – Snake Fungal Disease**

- Estimating the prevalence of fungal infection in a population of protected snake species in Michigan
  - qPCR test with false positive rate (probability) $f_p$ and false negative rate $f_n$.
  - In a sample of $n$ snakes, $y$ tested positive
  - What is the prevalence of the disease in the population ($\theta$)
- Response variable $y$ modeled by the binomial distribution $y \sim Bin(p, n)$ where $p$ is the probability of testing positive, and $p = \theta(1 - f_n) + (1 - \theta)f_p$
- The likelihood function $L = p^y(1 - p)^{n-y}$
- Log likelihood: $\log(L) = y\log(p) + (n - y)\log(1 - p)$
- Using a beta distribution prior for $\theta$: $\pi(\theta) \propto \theta^{\alpha}(1 - \theta)^{\beta}$
- The log posterior density $\pi(\theta \mid y, n) \propto \alpha\log(\theta) + \beta\log(1 - \theta) + y\log(p) + (n - y)\log(1 - p)$
- Computational options:
  - Classical statistics: $\hat{p} = y/n$, $\hat{\theta} = \frac{\hat{p} - f_p}{1 - (f_p + f_n)}$ (because MLE is transformation invariant). Uncertainty: variance of $\hat{p} \approx (1 - \hat{p})\hat{p}/n$, variance of $\hat{\theta}$ is $(1/(1 - f_p - f_n))^2(1 - \hat{p})\hat{p}/n$

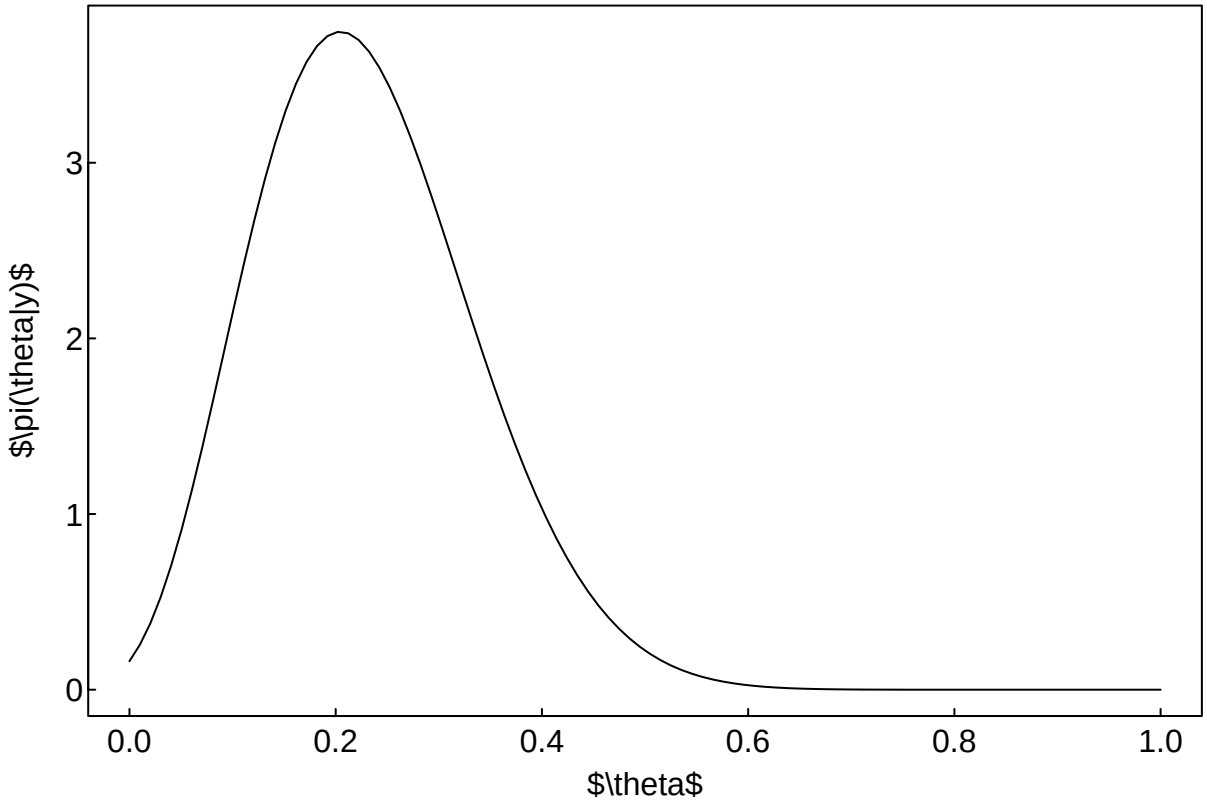
```
n <- 20
y <- 5
fn <- 0.05
fp <- 0.07
p_hat <- y/n
p_hat_sd <- sqrt(p_hat*(1-p_hat)/n)
theta_hat <- (p_hat - fp)/(1-fp-fn)
theta_hat_sd <- p_hat_sd/(1-fp-fn)
```

- Bayesian 1: brute force numerical integration Calculating the posterior density over a grid between 0 and 1 and normalize the results.

```
## prior beta(1,1)
  post_impft <- function(x=5, n=20, fp=0.07, fn=0.05, k=100){
    theta <- seq(0, 1,, k)
    fpst <- theta*(1-fn) + (1-theta)*fp
    post <- x*log(fpst) + (n-x)*log(1-fpst)
    return(list(pdf=exp(post)/(theta[2]*sum(exp(post))), cdf=cumsum(exp(post))/sum(exp(post))))
}

k <- 100
post_theta <- post_impft(k=k)
par(mar=c(3, 3, 1, 0.5), mgp=c(1.25, 0.125, 0), las=1, tck=0.01)
plot(seq(0, 1,, 100), post_theta$pdf, type="l", xlab="$\\theta$",
```

```
     ylab="$\\pi(\\theta|y)$")
```




```
## the mean: expected value
theta_seq <- seq(0,1,,100)
theta_mean <- theta_seq[2]*sum(post_theta$pdf * seq(0,1,,100))
theta_sd <- sqrt(sum(theta_seq[2]*post_theta$pdf*(seq(0,1,,100)-theta_mean)^2))

## mode = MLE
theta_mod <- theta_seq[post_theta$pdf==max(post_theta$pdf)]

## median
theta_med <- theta_seq[post_theta$cdf-0.5==min(abs(post_theta$cdf-0.5))]

## 5%- and 95% -tiles
theta_CI <- c(theta_seq[post_theta$cdf-0.05==min(abs(post_theta$cdf-0.05))],
              theta_seq[post_theta$cdf-0.95==min(abs(post_theta$cdf-0.95))])
```

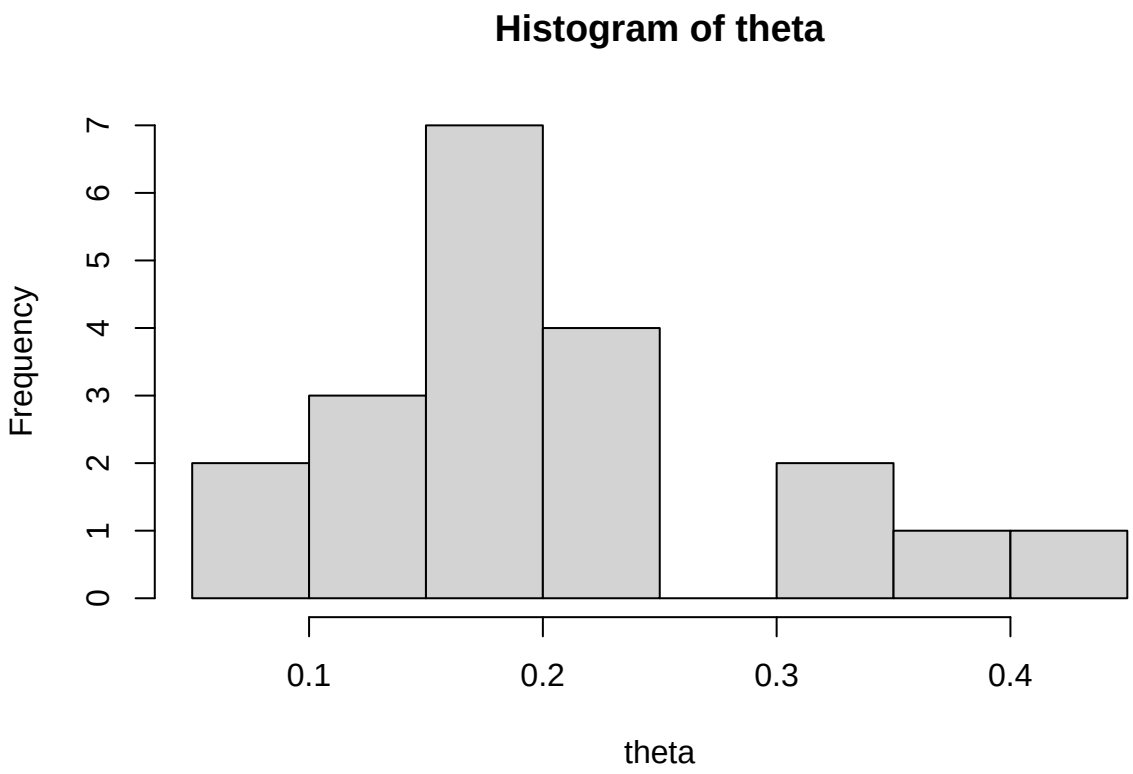
- Bayesian 2: Monte Carlo simulation Directly draw random numbers from the posterior distribution using the inverse-CDF method

```
## Using evenly spaced pdf from Chapter 1
post_impft_cdf <- function(x=5, n=20, fp=0.07, fn=0.05, k=100){
    theta <- seq(0, 1,, k)
    fpst <- theta*(1-fn) + (1-theta)*fp
    post <- x*log(fpst) + (n-x)*log(1-fpst)
    return(cumsum(exp(post)/sum(exp(post))))
}

post_cdf <- data.frame(theta=seq(0,1,,5000), cdf=post_impft_cdf(k=5000))
```

```
u <- runif(n)
tmp <- apply(post_cdf, 1, function(x, unf)
     return(x[2]-unf), unf=u)

theta <- apply(tmp, 1, function(x, theta)
     return(theta[abs(x)==min(abs(x))]),
               theta=post_cdf$theta)
hist(theta)
```




```
mean(theta)
```

```
## [1] 0.2064713
```

```
sd(theta)
```

```
## [1] 0.09078714
```

```
median(theta)
```

```
## [1] 0.1927385
```

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

```
##         5%        95%
## 0.08978796 0.36576315
```

- Bayesian 3: Using Metropolis-Hastings algorithm (See Section 2.4.1) An acceptance-rejection method (Section 2.3.2) with a candidate-generating distribution function and the acceptance probability probability is based on the ratio of posterior density function at two points. As a result, we only need to know the posterior density up to a proportional constant.

```
set.seed(10)
n_sims<-50000
```

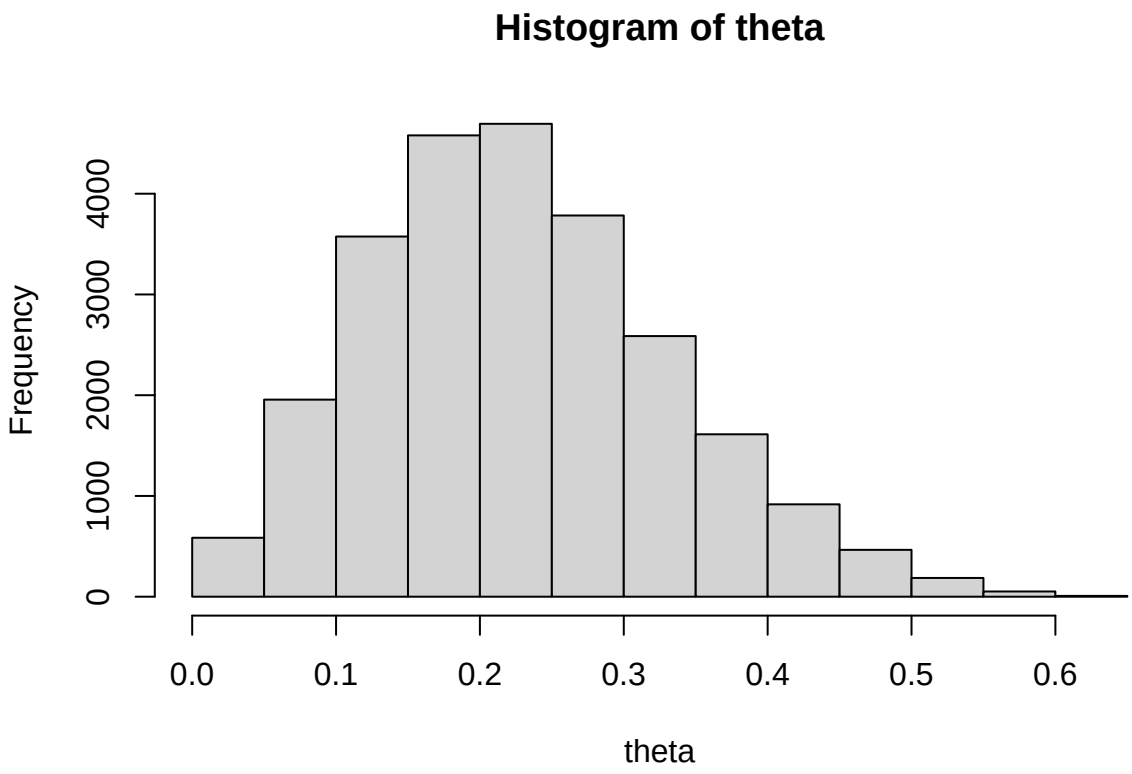
```r
theta <- numeric()
theta[1] <- runif(1) # initial value

n <- 20
x <- 5
log_lk <- function(theta, x=5, n=20, fp=0.07, fn=0.05){
    pp <- theta*(1-fn)+(1-theta)*fp
    llk <- x*log(pp)+(n-x)*log1p(-pp)
    return(llk)
}

for (j in 1:n_sims){
  y <- runif(1) ## unif(0,1) as the candidate-generating function
  alpha <- exp(log_lk(y)-log_lk(theta[j]))
  if (runif(1) < alpha) theta[j+1] <- y
  else theta[j+1] <- theta[j]
}

theta <- theta[round(1+n_sims/2):n_sims]

hist(theta)
```




```r
mean(theta)
```

```
## [1] 0.2276803
```

```r
sd(theta)
```

```
## [1] 0.104186
```

```
median(theta)
```

```
## [1] 0.2176312
```

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

```
##         5%        95%
## 0.07138723 0.41652067
```

## Stan using `rstan`

- Bayesian 4: Using Stan The basic task of Bayesian computation is to derive the posterior distribution

$\pi(\theta \mid y_1, \cdots, y_n) \propto \pi(\theta) L(\theta; y_1, \cdots, y_n)$

Using Stan, we draw random samples of the unknown parameters $\theta$ from their joint posterior distribution. As we know from the Metropolis-Hastings algorithm, we can draw random samples by knowing the posterior density upto a proportional constant. The rest can be automatic using a computer. In a Bayesian computing software, we need to provide three groups of information to formulate the MCMC algorithm. 1. Input data (e.g., $y, n, f_p, f_n$) 2. Parameters to be estimated 3. The likelihood function and prior distributions Stan puts the above three groups of information in three code blocks. For the snake fungal example, we have $x = 5, n = 20$, $f_p = 0.07$ and $f_n = 0.05$. The response variable $(y)$ is assumed to follow a binomial distribution $y \sim Bin(p, n)$, with the probability of a positive result being $p = \theta(1 - f_n) + (1 - \theta)f_p)$. Translating to Stan, we have

```
### Stan Code ###
snake_code1 <- "
  data{
    int<lower=1> n;
    int<lower=0> y;
    real<lower=0,upper=1> fn;
    real<lower=0,upper=1> fp;
  }
  parameters{
    real<lower=0,upper=1> theta;
  }
  model{
    theta ~ beta(1,1);
    y ~ binomial(n, theta*(1-fn) + (1-theta)*fp );
  }
"
```

We can also specify log-likelihood function directly by changing the `model` block to:

```
model{
   theta ~ beta(1,1);
   target += y*log(theta*(1-fn)_(1-theta)*fp)+
             (n-y)*log(1-theta*(1-fn)-(1-theta)*fp);
}
```

We can also add a `transformed parameters` block to clarify the code:

```
transformed parameters{
   real<lower=0mupper=1> p_pos;
   p_pos = theta*(1-fn)+(1-theta)*fp;
}
model{
   theta ~ beta(1,1);
```

```
   target += y*log(p_pos)+(n-y)*log(1-p_pos);
}
```

or directly use the log-probability function come with Stan:

```
transformed parameters{
   real<lower=0mupper=1> p_pos;
   p_pos = theta*(1-fn)+(1-theta)*fp;
}
model{
   theta ~ beta(0,1);
   target += binomial_lpmf(y | n, p_pos);
}
```

Here we used $beta(1,1)$ (the same as uniform between 0 and 1) as the non-informative prior of $\theta$. Because its density is a constant, we can omit the line `theta ~ beta(1,1);`.

Before we can run the Stan (code `snake_code1`), we need to load the package `rstan` and set some options

```
packages(rv)
require(rstan)
packages(car)
rstan_options(auto_write = TRUE)
options(mc.cores = min(c(parallel::detectCores(), 8)))

nchains <-  min(c(parallel::detectCores(), 8))
niters <- 5000
nkeep <- 2500
nthin <- ceiling((niters/2)*nchains/nkeep)
```

After that, we typically follow the following steps.

1. Compile the model

```
fit1 <- stan_model(model_code = snake_code1)
```

2. Organizing input data, initial values, and parameters to monitor using a function

```
input_snake <- function(y=5, n=20, nchns=nchains){
  data <- list(y=y,n=n, fp=0.07,fn=0.05)
  inits <- list()
  for (i in 1:nchns)
    inits[[i]] <- list(theta=runif(1))
  pars=c("theta")
  return(list(data=data, inits=inits, pars=pars))
}
```

3. Run the model

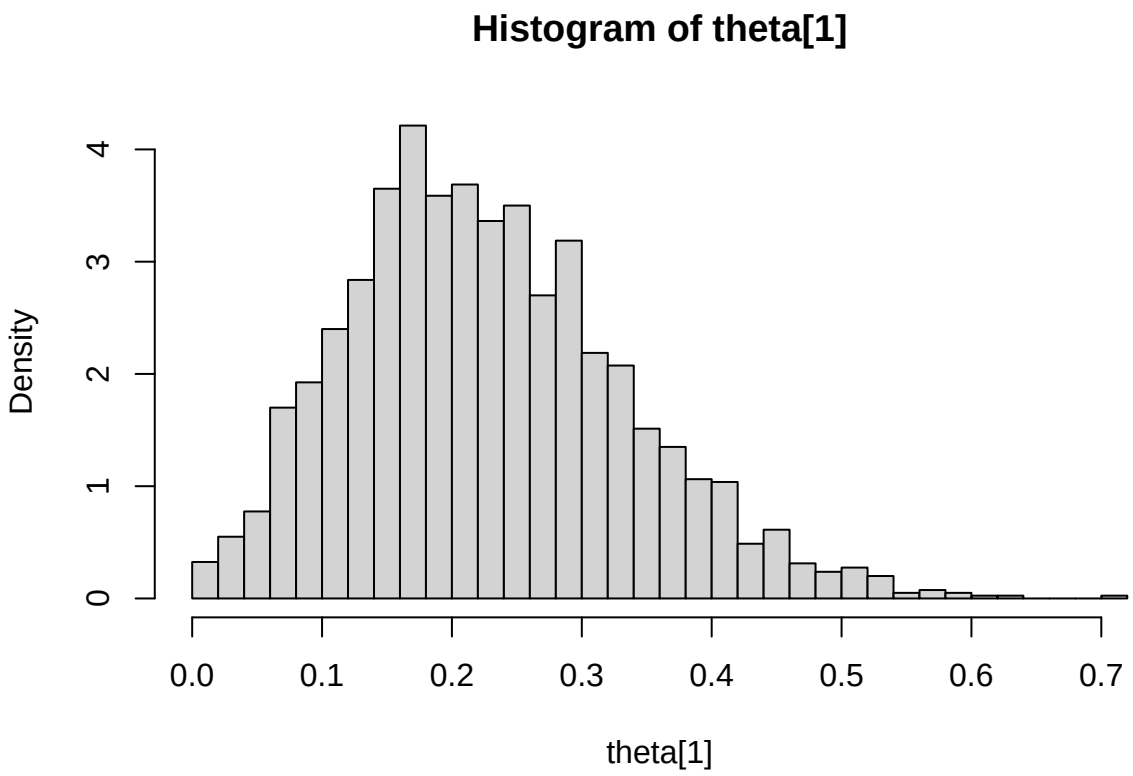
```
input.to.stan <- input_snake()
fit2keep <- sampling(fit1, data = input.to.stan$data,
                     init=input.to.stan$inits,
                     pars = input.to.stan$pars,
                     iter=niters, thin=nthin,
                     chains=nchains)
print(fit2keep)
```

```
## 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 0.11   0.05   0.15   0.22   0.29   0.46  2202    1
## lp__  -13.59    0.02 0.85 -16.03 -13.77 -13.26 -13.06 -13.00  2369    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 23 12:15:45 2023.
## 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).
```

4. Processing output

```
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta)
```




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

```
##         5%        95%
## 0.07238173 0.41610727
```

### Why Bayesian?

With Stan, Bayesian inference can be readily applied to more complex problems that are impossible for classical MLE. More importantly, Bayesian inference using Stan can provide useful information on model formulation and identify potential problems. Bayesian inference is also the best approach for information accumulation.

### 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 reflects 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 $\theta$ from the first sample to form a prior of $\theta$ 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.

```
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] 14.78959
```

Once the initial estimates of $\alpha_0$ and $\beta_0$ 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:

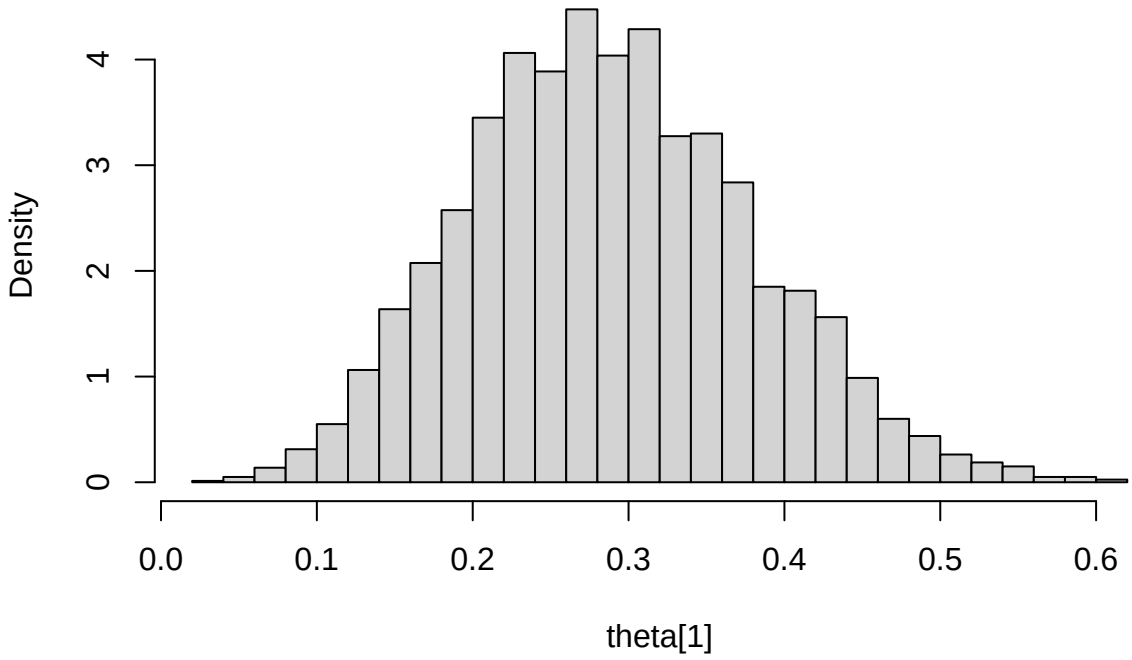
```
### Stan Code ###
snake_code2 <- "
  data{
    int<lower=1> n;
    int<lower=0> y;
    real<lower=0,upper=1> fn;
    real<lower=0,upper=1> fp;
    real<lower=0> alpha0;
    real<lower=0> beta0;
  }
  parameters{
    real<lower=0,upper=1> theta;
  }
  transformed parameters{
   real<lower=0,upper=1> 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)
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  0.47  2650    1
## lp__  -10.27    0.01 0.70 -12.25 -10.43 -10.01 -9.83 -9.78  2489    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 23 12:16:41 2023.
## 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).
```

```
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta)
```




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

```
##        5%       95%
## 0.1464969 0.4421897
```

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

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

```
## [1] 0.67475
```

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

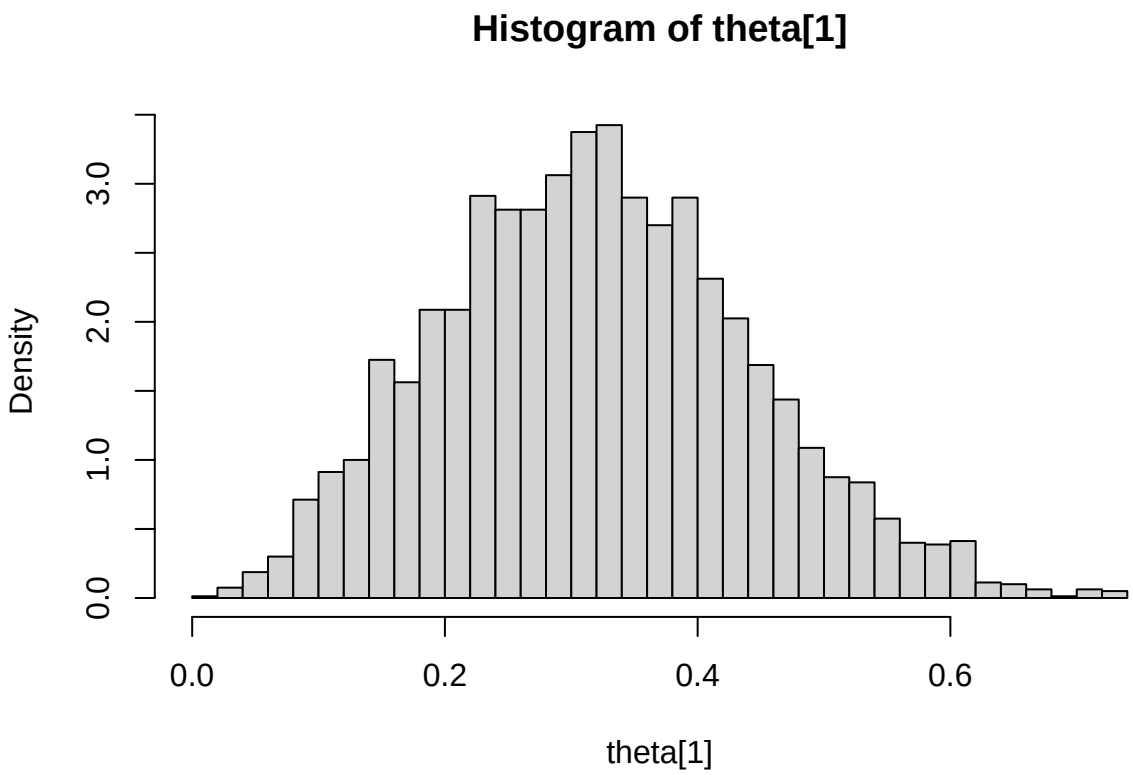
```
alpha0 <- 6*ybar
beta0 <- 6-alpha0

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  0.40  0.58  2484    1
## lp__  -5.48    0.02 0.79 -7.68 -5.63 -5.17 -4.98 -4.93  2225    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 23 12:16:44 2023.
## 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).
```

```
stan_out <- rstan::extract(fit2keep)
theta <- rvsims(stan_out$theta)
rvhist(theta)
```




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

```
##        5%       95%
## 0.1293575 0.5315524
```

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

```
## [1] 0.72375
```

**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)
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   0.13   0.19   0.26   0.39  2622    1
## fp      0.07    0.00 0.04   0.02   0.05   0.07   0.10   0.16  2493    1
```

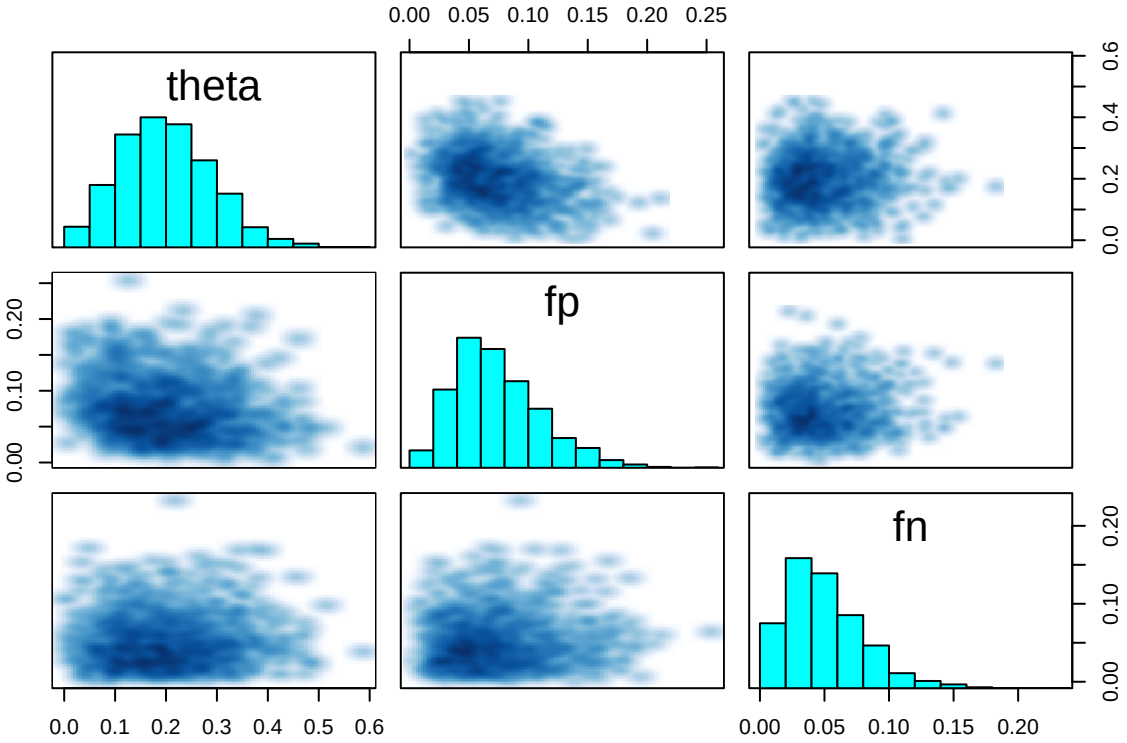
```
## fn      0.05    0.00 0.03   0.01   0.03   0.04   0.07   0.12  2581    1
## lp__  -29.02    0.03 1.32 -32.44 -29.61 -28.66 -28.07 -27.56  2601    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 23 12:17:34 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

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

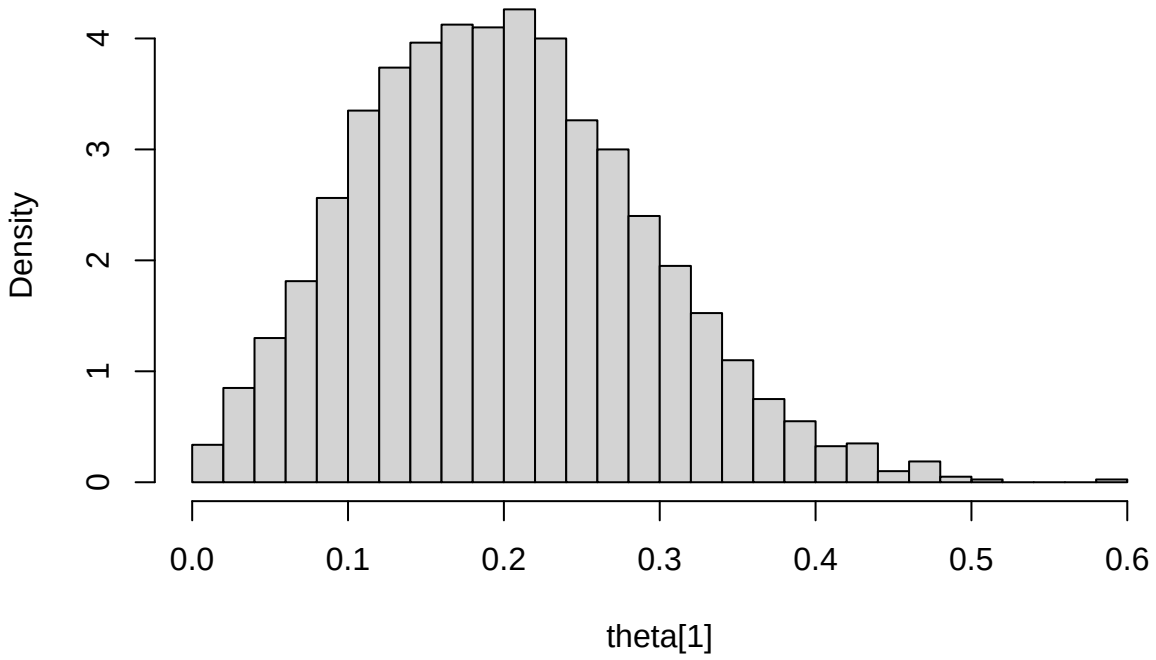
```
## Warning in par(usr): argument 1 does not name a graphical parameter

## 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)
```




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

```
##         5%        95%
## 0.06108758 0.35330701
```

**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 $\theta f_p$ and $\theta f_n$. In other words, the data have information of the two products. To definitely separate them, we need additional information. If we can 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 0.15   0.02   0.11  0.20  0.32  0.59  2467    1
## fp     0.25    0.00 0.15   0.02   0.15  0.24  0.33  0.58  2545    1
## fn     0.55    0.01 0.28   0.04   0.32  0.57  0.80  0.98  2780    1
## lp__  -9.71    0.03 1.34 -13.13 -10.41 -9.36 -8.69 -8.11  2274    1
```

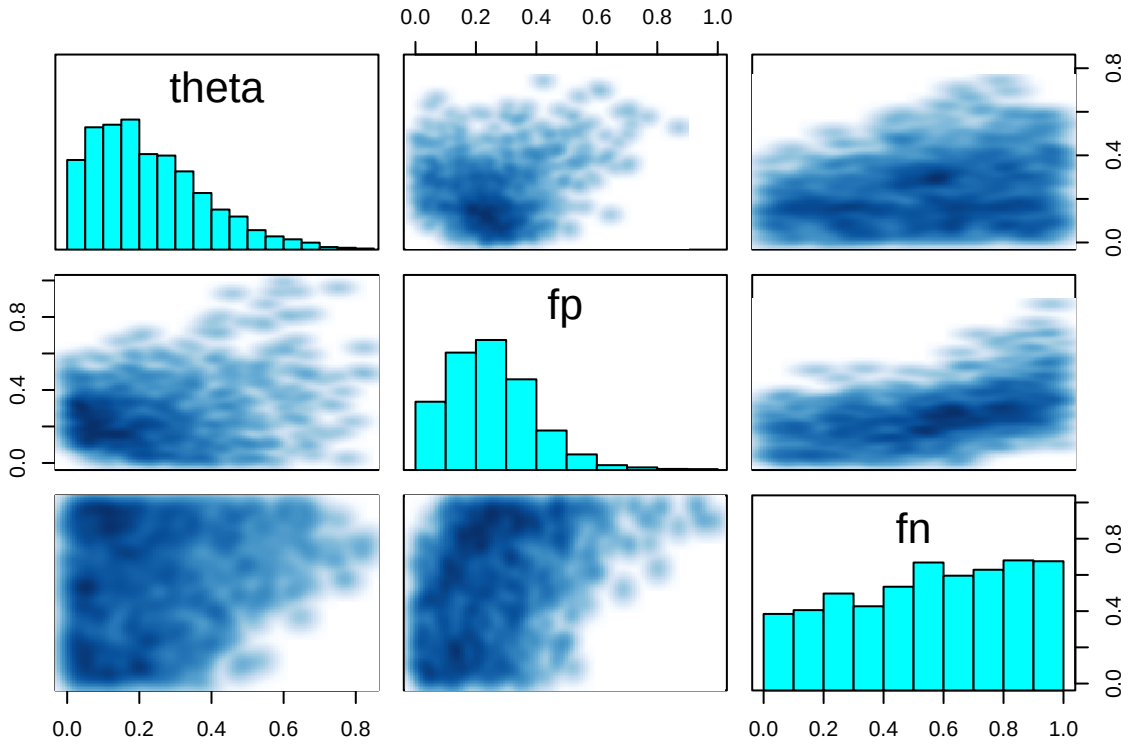
```
##
## Samples were drawn using NUTS(diag_e) at Tue May 23 12:17:38 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

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

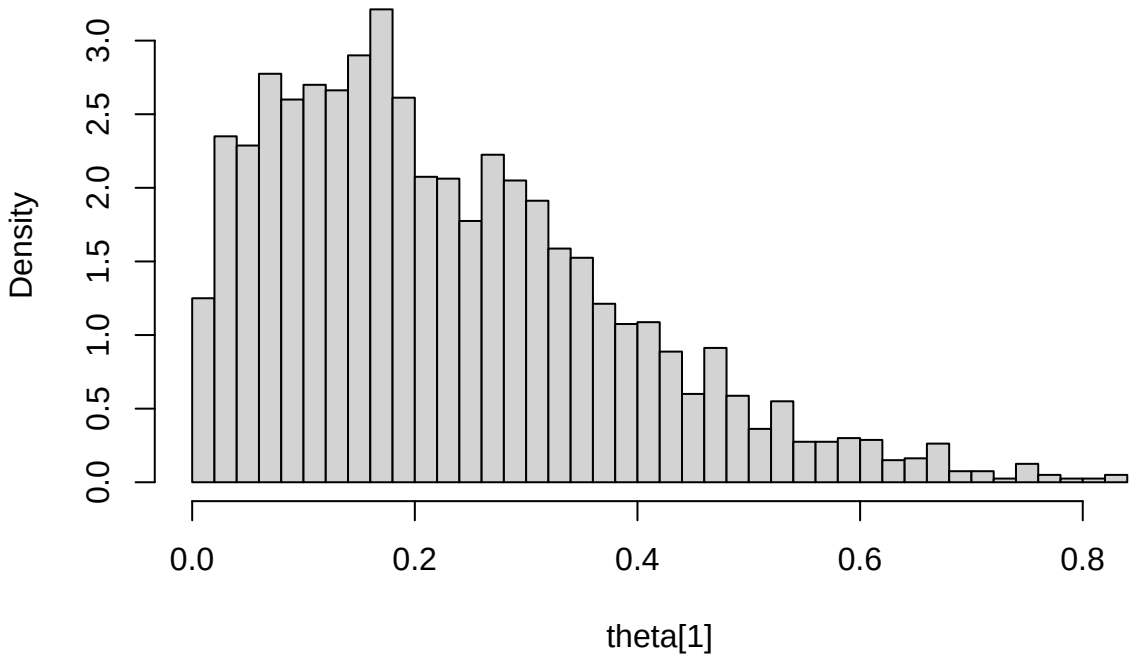
```
## Warning in par(usr): argument 1 does not name a graphical parameter

## 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)
```




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

```
##         5%        95%
## 0.02886154 0.52579035
```

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

```
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)
print(fit2keep)
```

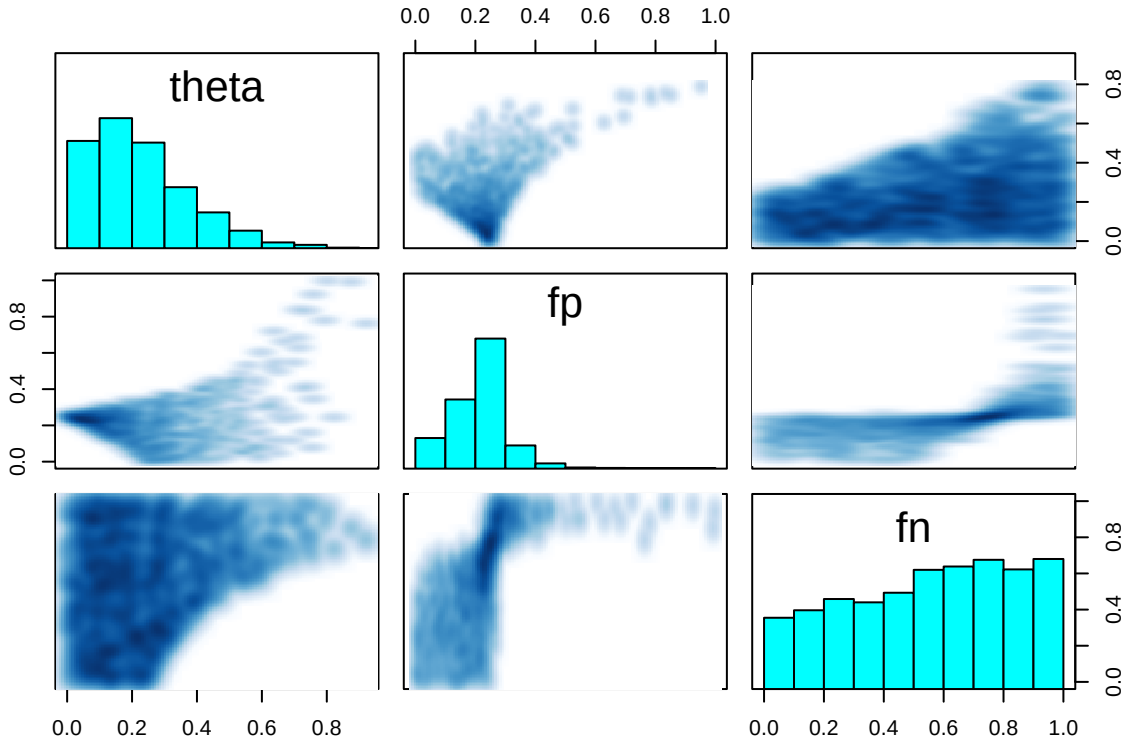
```
## Inference for Stan model: anon_model.
## 8 chains, each with iter=5000; warmup=2500; thin=8;
## post-warmup draws per chain=313, total post-warmup draws=2504.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## theta   0.23    0.00 0.15   0.02   0.11   0.20   0.31   0.59  2015    1
## fp      0.22    0.00 0.10   0.03   0.16   0.22   0.26   0.42  1933    1
## fn      0.56    0.01 0.28   0.03   0.33   0.58   0.79   0.98  2378    1
## lp__  -12.06    0.03 1.42 -15.84 -12.65 -11.70 -11.03 -10.45  1922    1
##
## Samples were drawn using NUTS(diag_e) at Tue May 23 12:17:44 2023.
## 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"))
```

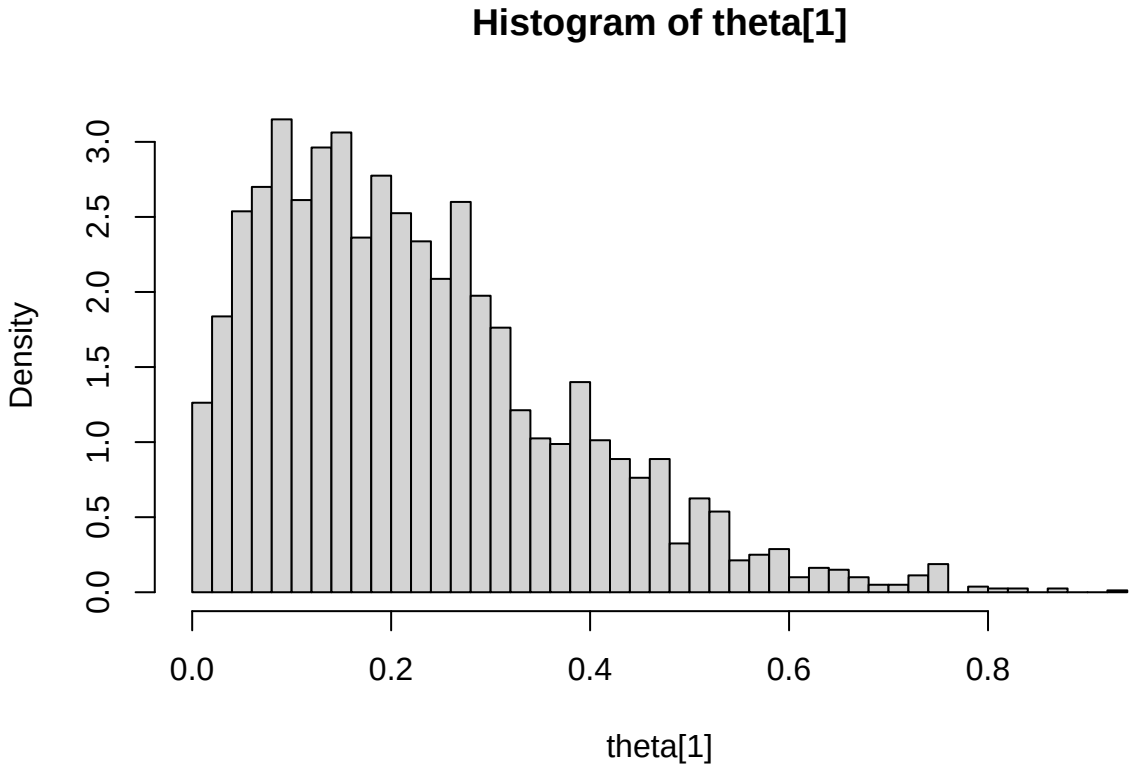
```
## Warning in par(usr): argument 1 does not name a graphical parameter

## 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)
```




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

```
##         5%        95%
## 0.03282353 0.51611333
```

Examine 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 explicitly formulate a model, identify parameters of interest, and provide any prior information we have on the parameters. The strong diagnostic feature of Stan makes model evaluation simple. 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