Exercises on Prior and posterior predictive distributions

Shravan Vasishth

September 02, 2019

Background: Generating prior predictive distributions

The prior predictive distribution is the distribution of observations we expect *before* we observe any data. It is useful for assessing whether the choice of prior distribution captures our prior beliefs.

Beta-Binomial case

Consider asking 100 questions from aphasic participants in an experiment.

Say that we have prior beliefs about the probability of correct responses: $\theta \sim Beta(a, b)$.

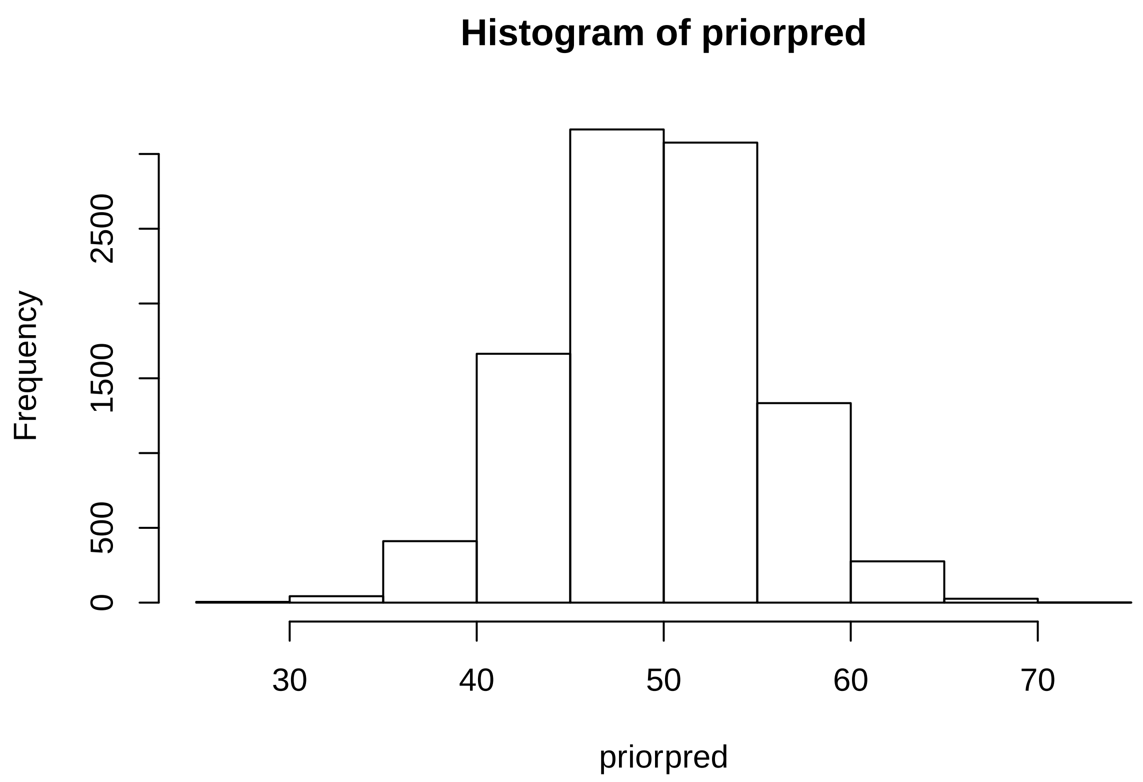
We can simulate the **prior predictive distribution** in one of two ways:

Using R

```
rbinom(1,size=10,prob=0.5)
```

```
## [1] 3
```

```
nsim<-10000
n<-100
priorpred<-rep(NA,nsim)
for(i in 1:nsim){
theta<-rbeta(1,shape1=200,shape2=200)
priorpred[i]<-rbinom(1,size=n,prob=theta)
}
hist(priorpred)
```



Exercise 1

Plot the prior predictive distribution under different priors:

- Beta(2,2)
- Beta(20,20)
- Beta(200,200)
- Beta(10,90)
- Beta(90,10)

Using RStan

We can also generate the prior predictive distribution using RStan. First, load rstan:

```
## load rstan
library(rstan)
```

```
## Loading required package: StanHeaders

## rstan (Version 2.19.2, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
options(mc.cores = parallel::detectCores())
```

Then, define the model:

```
priorpred<-"data {
  int n; // number of trials
  int p; // number of simulated patients
}
parameters {
real<lower=0,upper=1> theta;
}
model {
  // prior
    theta ~ beta(200,200);
}
generated quantities {
  vector[p] y_sim;
  // prior predictive distributions for p patients:
  for(i in 1:p) {
    y_sim[i] = binomial_rng(n,theta);
  }
}
"
```

Then we generate the data:

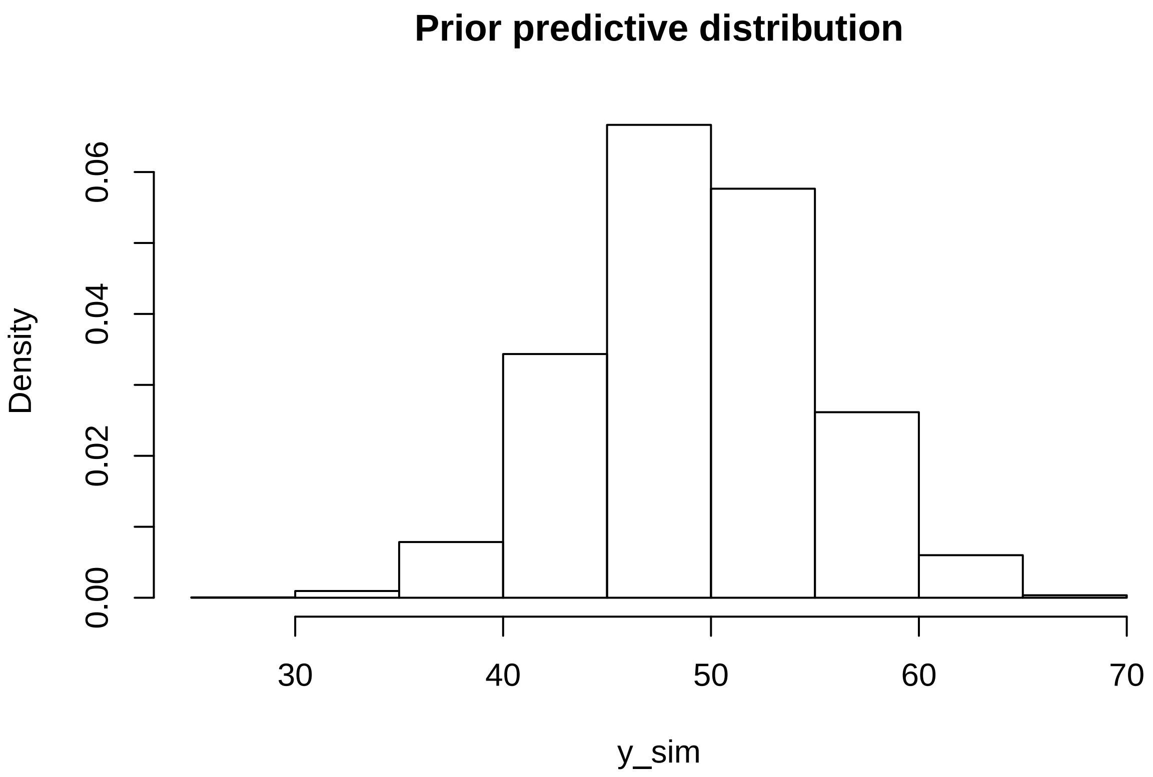
```
## generate 100 data points each for 50 patients
dat<-list(n=100,p=50)

## fit model:
m1priorpred<-stan(model_code=priorpred,
                  data=dat,
                  chains = 4,
                iter = 2000)

summary(m1priorpred)$summary

## extract and plot one of the data-sets:
y_sim<-extract(m1priorpred,pars="y_sim")
```

Here is example prior predictive data from a hypothetical subject:

```
str(y_sim)
```

```
## List of 1
##  $ y_sim: num [1:4000, 1:50] 51 42 49 42 47 58 51 45 48 43 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ iterations: NULL
##   .. ..$           : NULL
```

```
## data from hypothetical subject:
hist(y_sim$y_sim[,1],
     main="Prior predictive distribution",
     xlab="y_sim",freq=FALSE)
```



Exercise 2

Generate different prior predictive distributions using

- Beta(40,60)
- Beta(60,40)

Normal-Normal case

Say we are modeling reaction times in milliseconds.

Formally, we want to know the density $f(\cdot)$ of data points $y_1, \ldots, n$, given a vector of priors Θ. In the Normal example, $\Theta = \langle \mu, \sigma \rangle$. The prior predictive density is:

$$f(y_1, \ldots, y_n) = \int f(y_1) \cdot f(y_2) \cdots f(y_n) f(\Theta) \, d\Theta \tag{1}$$

In essence, we integrate out the parameters.

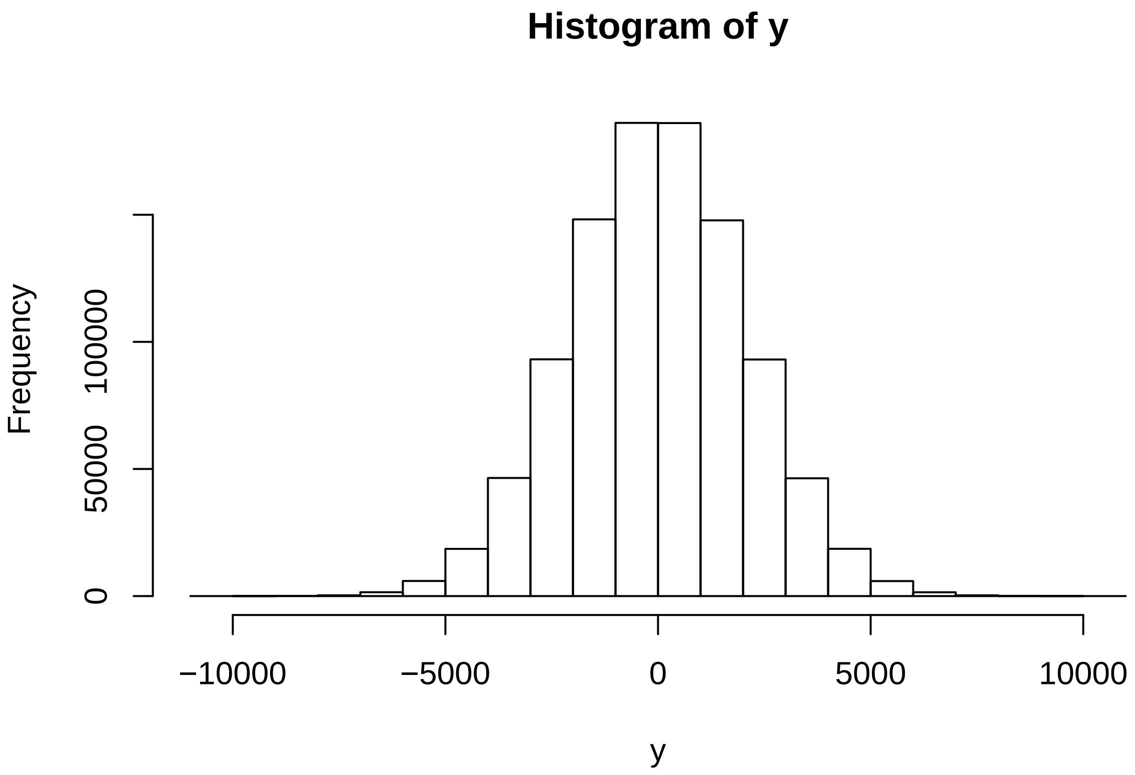
Using R

Here is one way to do it in R:

- Take one sample from each of the priors
- Generate data using those samples
- Repeat until you have a substantial amount of data
- Plot the prior predictive density

```r
nsim<-1000000
y<-rep(NA,nsim)
## uninformative prior
mu<-rnorm(nsim,mean=0,sd=2000)
## half-normal prior
sigma<-abs(rnorm(nsim,mean=0,sd=500))

for(i in 1:nsim){
y[i]<-rnorm(1,mean=mu[i],sd=sigma[i])
}
```

```r
hist(y)
```



This prior predictive distribution shows that the prior for μ is not realistic: how can reaction times have negative values?

Exercise 3

Try to redefine the prior for μ to have only positive values, and then check the prior predictive distribution again. We should still get some negative values, but that is because we are assuming that

$y \sim Normal(\mu, \sigma)$

which will have negative values for small μ and large σ.

Using RStan

We can generate a prior predictive distribution using Stan as follows.

First, we define a Stan model that defines the priors and defines how the data are to be generated.

```
priorpred<-"data {
  int N;
}
parameters {
real<lower=0> mu;
real<lower=0> sigma;
}
model {
  // priors
    mu ~ normal(0,2000);
    sigma ~ normal(0,500);
}
generated quantities {
  vector[N] y_sim;
  // prior predictive
  for(i in 1:N) {
    y_sim[i] = normal_rng(mu,sigma);
  }
}
"
```

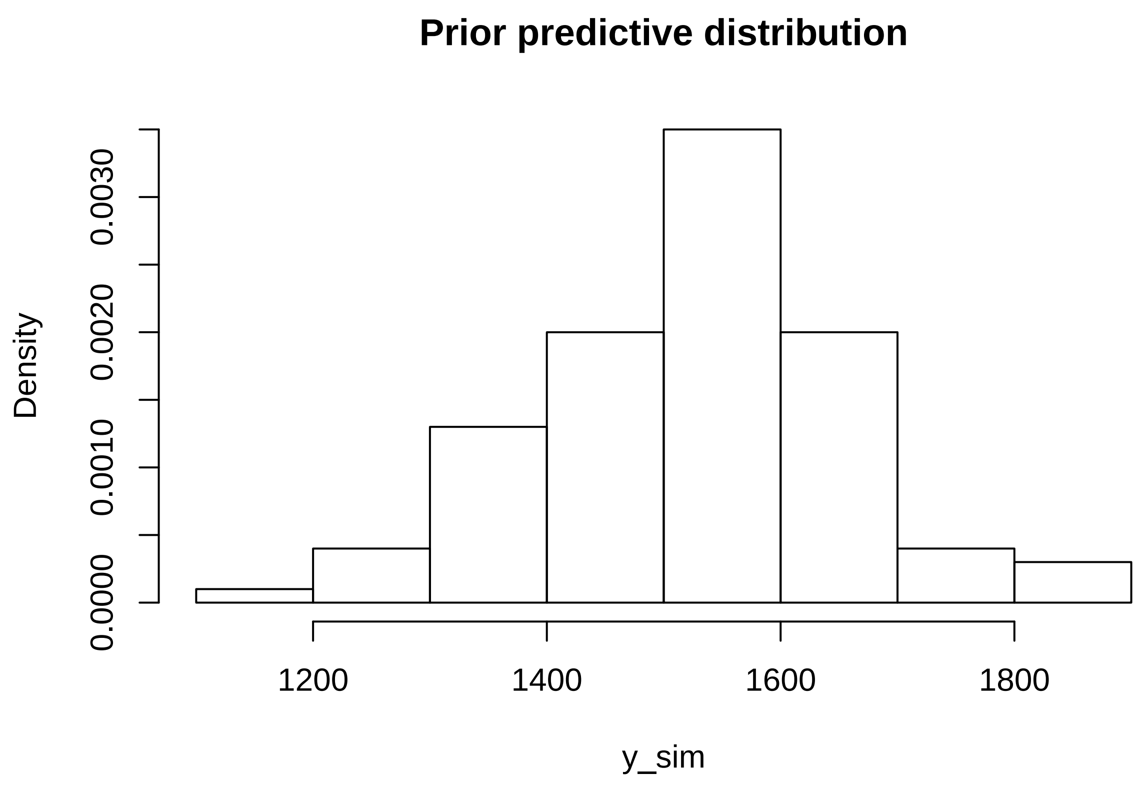
Then we generate the data:

```
## load rstan
library(rstan)
options(mc.cores = parallel::detectCores())

## generate 100 data points
dat<-list(N=100)

## fit model:
m1priorpred<-stan(model_code=priorpred,
                  data=dat,
                  chains = 4,
                iter = 2000)

## extract and plot one of the data-sets:
y_sim<-extract(m1priorpred,pars="y_sim")
```

```
str(y_sim)
```

```
## List of 1
##  $ y_sim: num [1:4000, 1:100] 1568 1234 1874 1442 1764 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ iterations: NULL
##   .. ..$           : NULL
```

```
hist(y_sim$y_sim[1,],
     main="Prior predictive distribution",
     xlab="y_sim",freq=FALSE)
```



Exercise 4

Change the Stan model such that the μ parameter has the prior: Normal(500,100). Display the prior predictive distribution.

Fake-data simulation for model validation

Having checked that the priors approximately make sense, we now ask: can our model recover unknown model parameters?

To check this, first, we re-write the above Stan model, adding a likelihood in the model block:

```
m1<-"data {
  int N;
  real y[N]; // data
}
parameters {
real<lower=0> mu;
real<lower=0> sigma;
}
model {
// priors
mu ~ normal(0,2000);
sigma ~ normal(0,500);
// likelihood
y ~ normal(mu,sigma);
}
```

```
generated quantities {
  vector[N] y_sim;
 // posterior predictive
  for(i in 1:N) {
    y_sim[i] = normal_rng(mu,sigma);
  }
}
"
```

Then generate fake data with known parameter values (we decide what these are):

```
set.seed(123)
N <- 500
true_mu <- 400
true_sigma <- 125
y <- rnorm(N, true_mu, true_sigma)

y <- round(y)
fake_data <- data.frame(y=y)
dat<-list(y=y,N=N)
```

Finally, we fit the model:

```
## fit model:
m1rstan<-stan(model_code=m1,
                  data=dat,
                  chains = 4,
                iter = 2000)

## extract posteriors:
posteriors<-extract(m1rstan,pars=c("mu","sigma"))
```

Figure 1 shows that the true parameters that we defined when generating the fake data fall within the posterior distributions. This shows that the model can in principle recover the parameters. (One should do several fake data simulations to check that the model consistently recovers the true parameters.)

Exercise 5

Change the above Stan model's parameter's priors to be Cauchy(0,2.5) and re-evaluate whether the model can recover the true mu and sigma parameters.

Exercise 6

Now change the above Stan model's parameter's priors to be Normal(0,1) and re-evaluate whether the model can recover the true mu and sigma parameters. What changes relative to the other example in Exercise 5 and why?

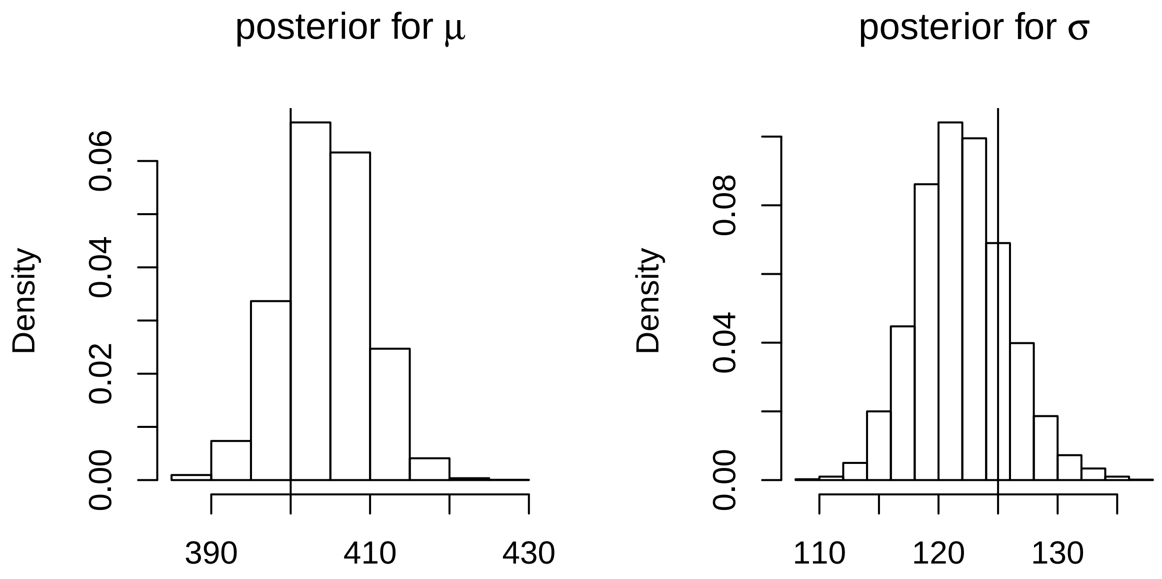


Figure 1: Posteriors from fake data, model m1. Vertical lines show the true values of the parameters.