KU Leuven Summer School
Segment 1
Jumping into Bayes

Paul Gustafson

September 15, 2022

Example story

Going to be getting some patient-level data:

- take standard ($X = 0$) or new ($X = 1$) drug;
- symptoms improve ($Y = 1$) or not ($Y = 0$).

Define $p_x = Pr(Y = 1|X = x)$, $x = 0, 1$.

In advance of receiving data, experts feels that $p_x < 0.05$ is unlikely, as is $p_x > 0.33$, for both $x = 0$ and $x = 1$.

This expert knowledge is encoded in a **prior distribution**:

$p_0, p_1 \overset{\text{iid}}{\sim} \text{beta}(a, b)$.

```r
### declare our hyperparameter specification
hyp <- list(a=3, b=15)

### and confirm this specification is fit-for-purpose
qbeta(c(.05,.95),hyp$a, hyp$b)
```

```
## [1] 0.0499 0.3262
```

Arrival of the data

```
dim(dat)
```

```
## [1] 37  2
```

```
head(dat)
```

```
##   x y
## 1 1 0
## 2 0 0
## 3 0 0
## 4 0 0
## 5 0 0
## 6 0 0
```

```
table(dat)
```

```
##    y
## x    0  1
##   0 22  1
##   1  9  5
```

37

Statistical model for data given parameters

Sufficient statistics $S_j = \sum_{i:x_i=j} y_i$, $j = 0, 1$.

$S_j \sim Bin(n_j, p_j)$, independently for $j = 0, 1$.

```
n <- as.vector(table(dat$x))
s.dat <- c(sum(dat$y[dat$x==0]), sum(dat$y[dat$x==1]))
```

```
n
```

```
## [1] 23 14
```

```
s.dat
```

```
## [1] 1 5
```

Scalar parameter of most interest:

$\psi = \text{logit}(p_1) - \text{logit}(p_0)$ $= \log\left(\frac{p_1}{1-p_1}\right) - \log\left(\frac{p_0}{1-p_0}\right)$

$= \log OR(X, Y)$

Generative model

The amalgamation of the prior distribution and the statistical model can be referred to as the *generative model*.

- A completely specified *joint* dist. for data and parameters.
- May feel weird, since from the investigator's perspective parameters are **fixed but unknown**, not **random**. Need to remember: prior distribution is being used to describe **investigator knowledge** about these fixed but unknown quantities.
- Perhaps helpful heuristic: 'Mother Nature' uses the generative model to generate both the state of the world and the ensuing data. But she keeps the former private, only shares the data with the investigator.

true param. values

Posterior distribution

p

Describes investigator's knowledge about the state of the world after Nature has shared the data.

s

Bayes theorem

$$f(p|s) = \frac{f(s|p)f(p)}{f(s)}$$

$$\propto \underbrace{f(s|p)}_{\text{likelihood}}\underbrace{f(p)}_{\text{prior}}$$

proportional as a function of p, for fixed s

sometimes see more explicitly with

$f(s) = \int f(s|\tilde{p}) f(\tilde{p})\, d\tilde{p}$

The Holy Grail of Bayesian Statistics

generative model

- prior
- statistical model

data

posterior dist.

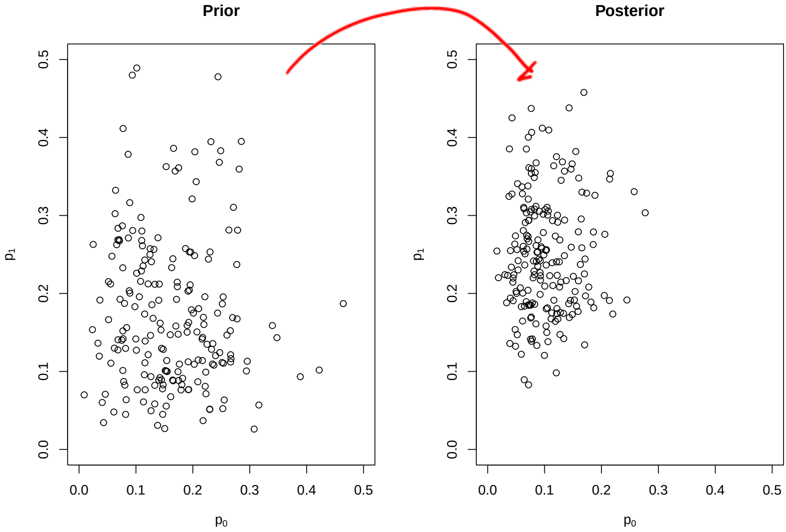
Determining the posterior distribution - in a literal, brute-force manner

"Bayesian socks" — video

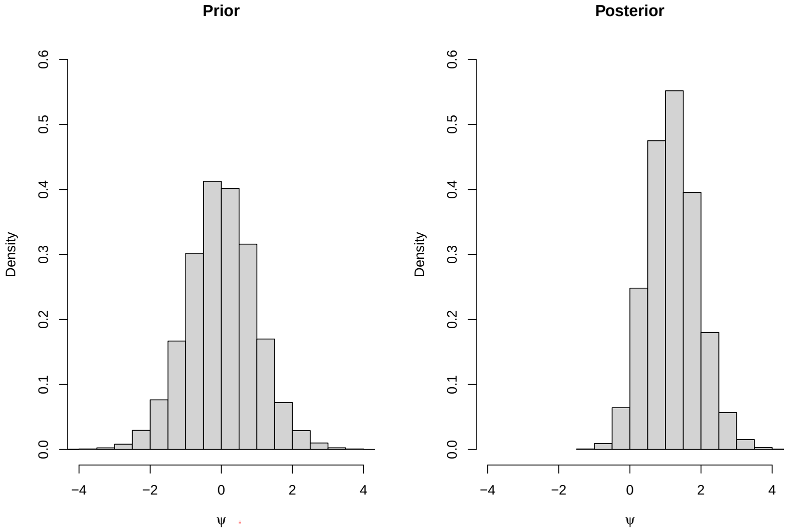
```
### represent the posterior via m MC realizations
m <- 20000; opt.brute <- matrix(NA, m, 2)
colnames(opt.brute) <- c("p0","p1")

have <- 0
while (have < m) {

  ### simulate from generative model
  p.0.sim <- rbeta(1, hyp$a, hyp$b)
  p.1.sim <- rbeta(1, hyp$a, hyp$b)
  s.sim <- c(rbinom(1, size=n[1], prob=p.0.sim),
             rbinom(1, size=n[2], prob=p.1.sim))

  ### only keep if the simulated data matches observed
  if (all(s.sim==s.dat)) {
    have <- have + 1
    opt.brute[have,] <- c(p.0.sim, p.1.sim)
  }
}
```

Prior and posterior distributions of (p_0, p_1)

impact of data



Estimate the target



Do some inference

A point estimate, $E(\psi|\text{Data})$: posterior mean

```
mean(opt.brute[,"psi"])
```

```
## [1] 1.2
```

A "standard error" $SD(\psi|\text{Data})$: posterior SD

```
sqrt(var(opt.brute[,"psi"]))
```

```
## [1] 0.705
```

A 95% interval estimate:

```
quantile(opt.brute[,"psi"], c(0.025, 0.975))
```

```
##   2.5%  97.5%
## -0.111  2.644
```

Nuance: Statistical error versus numerical / Monte Carlo error

$SD(\psi|\text{Data})$ is a 'standard-error-like' quantification of how well the sample quantity $E(\psi|\text{Data})$ estimates the population parameter ψ.

That said, carefully distinguish the roles of:

```r
sqrt(var(opt.brute[,"psi"]))
```

```
## [1] 0.705
```

governed by n

- numerical approx of $SD(\psi|\text{Data})$
- becomes exact as $m \to \infty$

last slide $\hat{\psi} = \frac{1}{m}\sum_{i=1}^{m} \psi^{(i)} = 1.20$

posterior mean

```r
sqrt(var(opt.brute[,"psi"])/m)
```

```
## [1] 0.00498
```

so numerical approx good to about $\pm 2(.005) = \pm .01$

Monte Carlo SE

Answer by package - JAGS

JAGS syntax, not R code

```
genmod.JAGS <- "model{

  ### prior distribution
  p0 ~ dbeta(a,b)
  p1 ~ dbeta(a,b)

  ### statistical model
  s0 ~ dbinom(p0, n0)
  s1 ~ dbinom(p1, n1)

  ### and for conveience, store the target param also
  psi <- logit(p1)-logit(p0)
}"
```

encoding gen. model

Answer by JAGS, continued

```
require(rjags)
```

```
### generative model, data go in
mod <- jags.model(textConnection(genmod.JAGS),
                  data=list(s0=s.dat[1], n0=n[1],
                            s1=s.dat[2], n1=n[2],
                            a=hyp$a, b=hyp$b),
                  n.chains=4)
```

hyperparams

```
### bit of burn-in
update(mod,2000)

###  MC output comes out
opt.JAGS <- coda.samples(mod,
                 variable.names=c("psi"),
                 n.iter=10000)
```

Can we find our friendly neighbourhood estimates in the package output?

Yes, at least to some level of numerical precision

```
summary(opt.JAGS)
```

```
##
## Iterations = 3001:13000
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean             SD       Naive SE Time-series SE
##       1.19930        0.69515        0.00348        0.00450
##
## 2. Quantiles for each variable:
##
##    2.5%     25%     50%     75%   97.5%
## -0.105   0.724   1.179   1.649   2.625
```

" reflecting numerical (MC) precision

FYI, lots of output options

different quantification of numerical precision

as good as if we had this many independent MC draws

```
require(MCMCvis)
MCMCsummary(opt.JAGS)
```

```
##      mean    sd   2.5%  50% 97.5% Rhat n.eff
## psi   1.2 0.695 -0.105 1.18  2.63    1 23849
```

we asked for 4×10600, but they aren't independent

ASIDE: off-the-shelf **non-Bayesian** analysis of these data?

```
### ML estimate of psi
logit(s.dat[2]/n[2]) - logit(s.dat[1]/n[1])
```

```
## [1] 2.5
```

```
### standard error
sqrt(sum(1/c(s.dat[1], n[1]-s.dat[1], s.dat[2], n[2]-s.dat[2])))
```

```
## [1] 1.16
```

not close to
Bayes answers

```
### or if you prefer, get these from...
glm(y~x, family=binomial)
```

Can speculate on **two** reasons why our present Bayesian answer is not so close to this

- no prior
- sparse data

Black-box package actually overkill for problem we just did

Our problem has **conjugate** structure, ergo a clean math description of the posterior dist. for $(p_0, p_1 | S_0, S_1)$:

$(p_j | S = s) \sim \text{beta}(a + s_j, b + (n_j - s_j))$, independently for $j = 0, 1$.

So yet another route to our inference

```
opt.MC <-cbind(
  "p0"=rbeta(m, hyp$a+s.dat[1], hyp$b+n[1]-s.dat[1]),
  "p1"=rbeta(m, hyp$a+s.dat[2], hyp$b+n[2]-s.dat[2]))

opt.MC <- cbind(opt.MC,
  "psi"=logit(opt.MC[,"p1"])-logit(opt.MC[,"p0"]))
```

```
## posterior mean and SD of target
c(mean(opt.MC[,"psi"]), sqrt(var(opt.MC[,"psi"])))
```

```
## [1] 1.196 0.697
```

✓ same answer

```
### quality of numerical approximation
sqrt(var(opt.MC[,"psi"])/m)
```

```
## [1] 0.00493
```

In fact, for problem at hand, can skip MC approximation, compute exactly

```
### posterior mean of psi
(digamma(hyp$a+s.dat[2])-digamma(hyp$b+n[2]-s.dat[2])) -
(digamma(hyp$a+s.dat[1]) - digamma(hyp$b+n[1]-s.dat[1]))
```

```
## [1] 1.2
```

same

```
### posterior SD of psi
sqrt(trigamma(hyp$a+s.dat[2]) + trigamma(hyp$b+n[2]-s.dat[2]) +
     trigamma(hyp$a+s.dat[1]) + trigamma(hyp$b+n[1]-s.dat[1]))
```

```
## [1] 0.698
```

Back to the holy grail

What *might* a platinum grail spit out?

exact values of any posterior quantity

What *might* a golden grail spit out?

MC realizations $\theta^{(1)}, \ldots, \theta^{(m)}$ which are iid from the posterior $f(\theta|data)$

What *might* a silver grail spit out?

$\theta^{(1)}, \ldots, \theta^{(m)}$ are dependent

What **does** a bronze grail spit out?

$\theta^{(1)}, \ldots, \theta^{(m)}$

dist $\theta^{(m)} \longrightarrow f(\theta|data)$ as $m \to \infty$

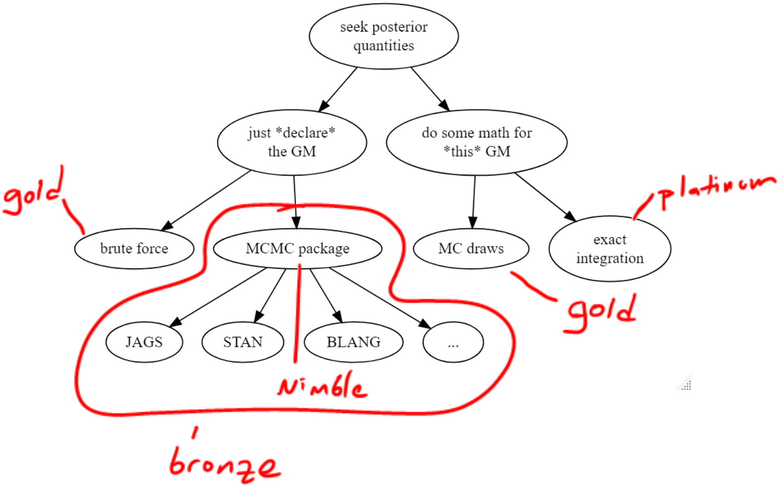
converges quickly

$\theta^{(1)} \to \theta^{(2)} \to \theta^{(3)} \ldots$ Markov chain

arbitrary

clever specification of the $\theta^{(i)} \to \theta^{(i+1)}$ recipe

Taxonomy of the computing options we've seen today



Since we are typically stuck with a bronze grail (i.e., MCMC) ...

choice of m
- how many chains
- how long is each

- Users can/should exert control over how much computing to do.
- Users can/should monitor how successful this computing is.

  - Monte Carlo SE
  - m.eff
  - R.hat
  - traceplots
  - ...

APPENDIX: Could you stand to see one more computational route to the same end?

```
genmod.STAN <- "
data {
  int<lower=0> n[2];
  int<lower=0> s[2];
  real<lower=0> a[2];
  real<lower=0> b[2];
}
parameters {
  real<lower=0,upper=1> p[2];
}
transformed parameters {
  real psi;
  psi =logit(p[2])-logit(p[1]);
}
model {
  p ~ beta(a,b);
  s ~ binomial(n,p);
}
```

STAN, continued

```r
require("rstan"); rstan_options(auto_write = TRUE)
```

```r
opt.STAN <- stan(model_code=genmod.STAN,
  data=list(s=s.dat, n=n, a=rep(hyp$a,2), b=rep(hyp$b,2)),
  iter=12000)
```

STAN, continued

```
summary(opt.STAN)$summary
```

```
##           mean  se_mean     sd     2.5%      25%      50%     75%
## p[1]    0.0965 0.000304 0.0447   0.0286   0.0631   0.0902   0.123
## p[2]    0.2496 0.000529 0.0746   0.1200   0.1961   0.2442   0.298
## psi     1.2074 0.004873 0.6817  -0.0863   0.7423   1.1863   1.654
## lp__  -32.1083 0.009706 1.0006 -34.7985 -32.4934 -31.8057 -31.400
##         97.5% n_eff Rhat
## p[1]    0.199 21618    1
## p[2]    0.405 19883    1
## psi     2.601 19572    1
## lp__  -31.127 10627    1
```

Due diligence: Two different black-boxes give same answer?

```r
MCMCsummary(opt.STAN)
```

```
##           mean     sd     2.5%      50%   97.5% Rhat n.eff
## p[1]    0.0965 0.0447   0.0286   0.0902   0.199    1 21618
## p[2]    0.2496 0.0746   0.1200   0.2442   0.405    1 19883
## psi     1.2074 0.6817  -0.0863   1.1863   2.601    1 19572
## lp__  -32.1083 1.0006 -34.7985 -31.8057 -31.127    1 10627
```

```r
MCMCsummary(opt.JAGS)
```

```
##      mean    sd   2.5%  50% 97.5% Rhat n.eff
## psi   1.2 0.695 -0.105 1.18  2.63    1 23849
```