Bayesian inference

Introduction

The purpose of this crash course is to develop a working knowledge of Bayesian statistics. We will focus on 1) the philosophy of Bayesian statistics, 2) how to use probabilistic programming languages to estimate Bayesian models, 3) how to ensure MCMC estimation of Bayesian models has converged, 4) how to interpret results of model fitting, and 5) how to assess and compare fitted models. This is by no means comprehensive - look forward to Bayes with Becky in the Fall!

Distribution	**Parameters**	**Probability function**	**Mean**	**Variance**	**MGF**
Bernoulli Bern(p)	$p \in [0,1]$	$f(x) = p^x(1-p)^{1-x}$; $x \in \{0,1\}$	p	$p(1-p)$	$pe^t + (1-p)$
Binomial Bin(p)	$p \in [0,1]$	$f(x) = \binom{n}{x} p^x (1-p)^{n-x}$; $x \in \{0,1,...,n\}$	np	$np(1-p)$	$\left[pe^t + (1-p)\right]^n$
Geometric Geom(p)	$p \in [0,1]$	$f(x) = p(1-p)^{x-1}$; $x \in \{1,2,\dots\}$	$\frac{1}{p}$	$\frac{1-p}{p^2}$	$\frac{pe^t}{1-(1-p)e^t}$
Hypergeometric	$N \in \{0,1,\dots\}$ $r \in \{0,1,\dots,N\}$ $n \in \{0,1,\dots,N\}$	$f(x) = \frac{\binom{r}{x}\binom{N-r}{n-x}}{\binom{N}{n}}$; $x \in \{0,1,\dots,n\}$ if $n \leq r$, $x \in \{0,1,\dots,r\}$ if $n > r$	$\frac{nr}{N}$	$n\left(\frac{n}{r}\right)\left(\frac{N-r}{N}\right)\left(\frac{N-n}{N-1}\right)$	Don't bother
Poisson Pois(λ)	$\lambda > 0$	$f(x) = \frac{e^{-\lambda}\lambda^x}{x!}$; $x \in \{0,1,\dots\}$	λ	λ	$\exp\left[\lambda(e^t-1)\right]$
Negative binomial NegBin(r,p)	$r \in \{0,1,\dots\}$ $p \in [0,1]$	$f(x) = \binom{x+r+1}{x} p^r (1-p)^x$; $x \in \{0,1,\dots\}$	$\frac{r(1-p)}{p}$	$\frac{r(1-p)}{p^2}$	$\left(\frac{p}{1-(1-p)e^t}\right)^r$
Beta beta(a, b)	$a, b > 0$	$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}$; $x \in (0,1)$	$\frac{a}{a+b}$	$\frac{ab}{(a+b)^2(a+b+1)}$	DNE
Chi-square χ^2_ν	$\nu \in \{1,2,\dots\}$	$f(x) = \frac{1}{2^{\nu/2}\Gamma(\frac{\nu}{2})} x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}$; $x > 0$	ν	2ν	$(1-2t)^{-v/2}$
Exponential Exp(λ)	$\lambda > 0$	$f(x) = \lambda e^{-\lambda x}$; $x > 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$	$\frac{\lambda}{\lambda - t}$
Gamma Ga(α, β)	$\alpha, \beta > 0$	$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$; $x > 0$	$\frac{\alpha}{\beta}$	$\frac{\alpha}{\beta^2}$	$\left(1 - \frac{t}{\beta}\right)^{-\alpha}$
Normal $N(\mu, \sigma^2)$	$\mu \in (-\infty, \infty)$	$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$; $x \in (-\infty, \infty)$	μ	σ^2	$\exp\left\{\mu t + \frac{t^2\sigma^2}{2}\right\}$
Uniform Unif(θ_1, θ_2)	$\{\theta_1, \theta_2 : \theta_1 < \theta_2\}$	$f(x) = \frac{1}{\theta_2 - \theta_1}$; $\theta_1 \leq x \leq \theta_2$	$\frac{1}{2}(\theta_1 + \theta_2)$	$\frac{1}{12}(\theta_2 - \theta_1)^2$	$\frac{e^{t\theta_2} - e^{t\theta_1}}{t(\theta_2 - \theta_1)}$

The Bayesian statistical paradigm

Bayesian statistics

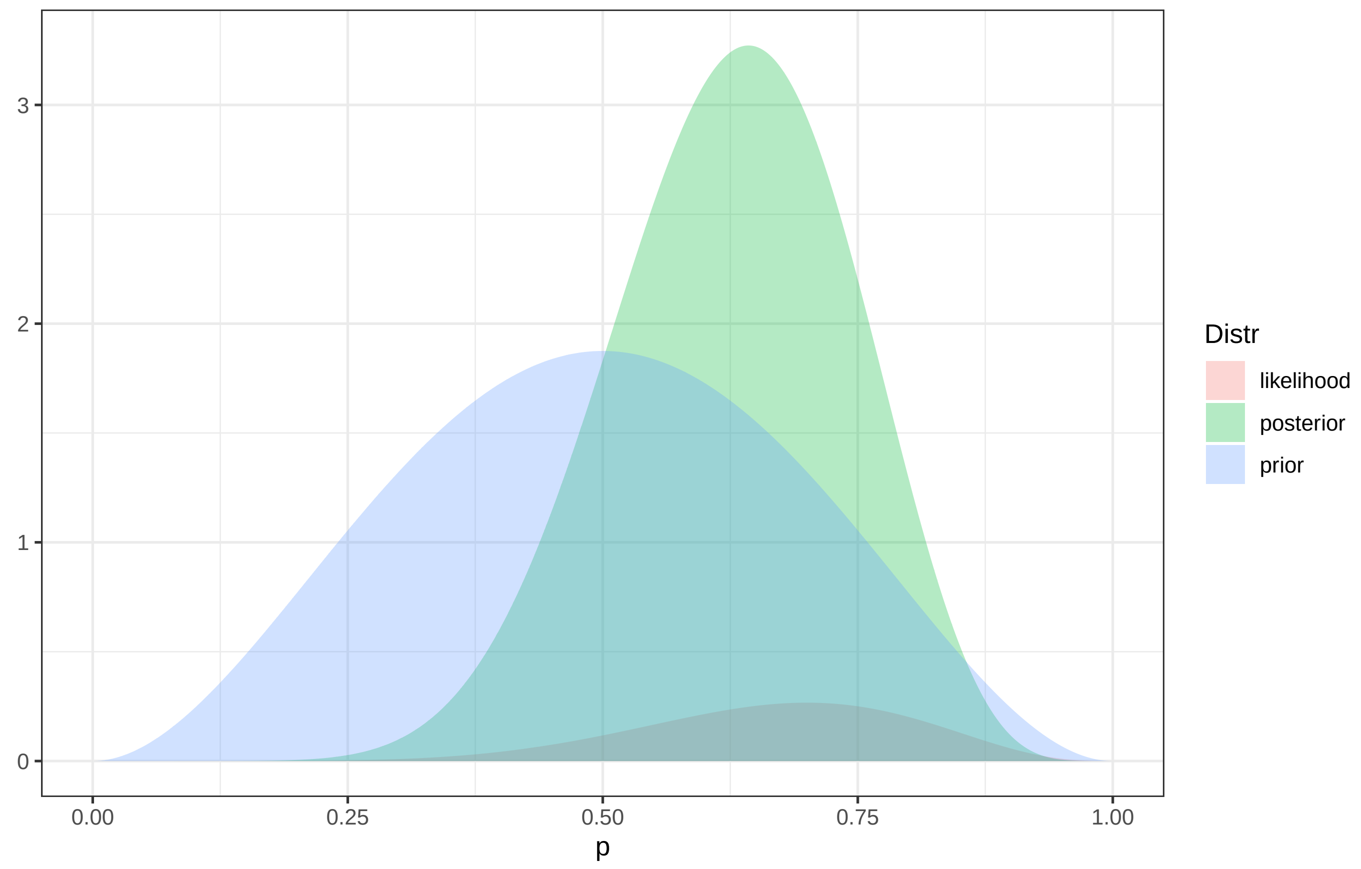
1. A model for the data generating mechanism is specified in terms of probability distributions with unknown parameters. The model should be specified in a way that the questions you have of the data may be asked of the model parameters.

2. Your prior belief about the unknown parameters is expressed in a probabilistic way via the **prior distribution**, $p(\theta)$.

3. Data are collected and are modeled via a probability distribution, $p(\boldsymbol{y}|\theta)$ (i.e. the likelihood or sampling model)

4. Your updated belief in θ is expressed via the **posterior distribution**, according to Bayes' rule:
$$p(\theta|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\theta)p(\theta)}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\theta)p(\theta)}{\int_\theta p(\boldsymbol{y}|\theta)p(\theta)d\theta}$$

5. All subsequent inference is based on the posterior distribution.



Example research questions

Suppose you are interested in modeling the count of invasive weeds obtained from various locations throughout a National Park as a function of precipitation.

Suppose 10 students each swab their mouths five times and you are interested in modeling the presence of a particular bacteria in each students' mouth.

Bayesian estimation

Once you have specified your prior distribution ($p(\theta)$) and likelihood ($p(\boldsymbol{y}|\theta)$), the posterior may be obtained according to Bayes' rule:

$$p(\theta|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\theta)p(\theta)}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{y}|\theta)p(\theta)}{\int_\theta p(\boldsymbol{y}|\theta)p(\theta)d\theta}$$

For most problems, this mathematical expression is not tractable because the integral in the denominator has no elementary anti-derivative. There exist combinations of priors and likelihoods for which there are analytic solutions for the posterior distribution, called **conjugate priors** - we will not focus on those this week. Instead, we will consider a flexible approach to finding the posterior distribution of a parameter, called **Markov Chain Monte Carlo** (MCMC).

Markov-Chain Monte Carlo

When we cannot find a closed form expression for the posterior distribution, we instead try to *sample from the unknown density function* by generating a Markov Chain whose stationary distribution is the joint posterior distribution of the unknown parameters. Thus, samples drawn from this Markov chain may be used to represent the unknown posterior density.

There exist many strategies for developing this Markov chain, including **Gibbs sampling**, which I expect you will see in the Fall. Today, we will focus on a very general algorithm for generating this Markov chain, called the **Metropolis-Hastings** algorithm, which proceeds as follows:

1. Initialize the vector of unknown parameters $\theta^{(s)}$.
2. Choose a "transition kernel" or "proposal distribution" $Q(\theta^*|\theta^{(s)})$, a way of proposing a new set of parameter values θ^* given $\theta^{(s)}$. This kernel should be symmetric, and most often is chosen to be a random walk. For instance,

$$\theta^*|\theta^{(s)} \sim \mathcal{N}(\theta^{(s)}, \tau_0^2\mathcal{I})$$

3. For s in `2:iter`:
   - sample θ^* from $Q(\theta^*|\theta^{(s)})$
   - compute the acceptance probability

$$q = \min\left(1, \frac{p(\theta^*|\boldsymbol{y})Q(\theta^{(s)}|\theta^*)}{p(\theta^{(s)}|\boldsymbol{y})Q(\theta^*|\theta^{(s)})}\right)$$

   - with probability q, set $\theta^{(s+1)} = \theta^*$. Otherwise, set $\theta^{(s+1)} = \theta^{(s)}$.

Probabilistic programming languages

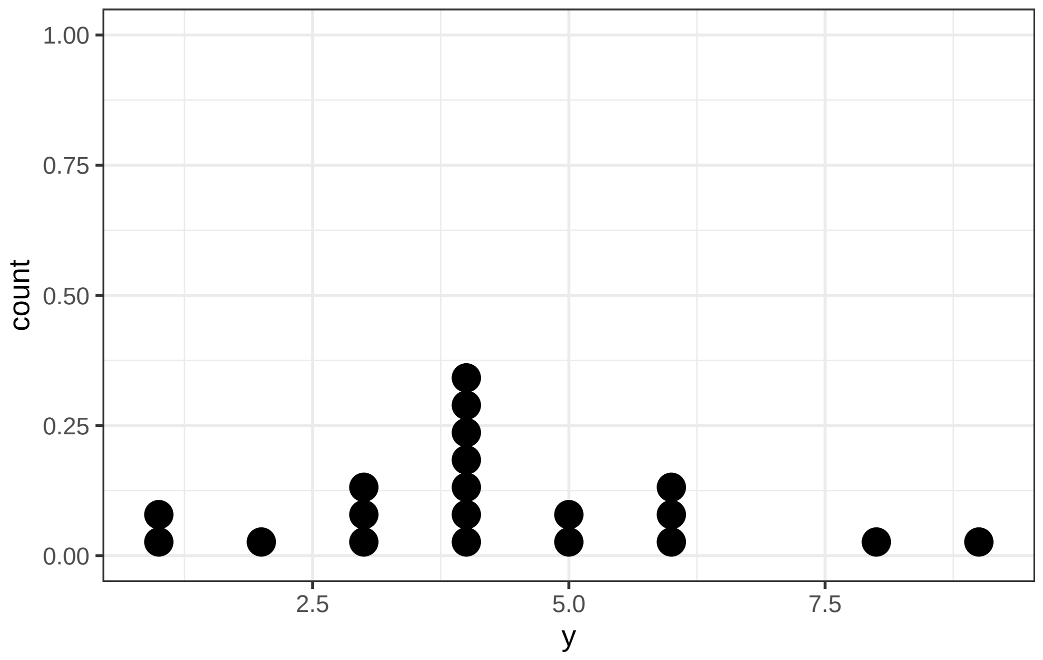
Implementing Metropolis-Hastings samplers by hand in R can be very fun, but also time consuming. Fortunately, there exist multiple flexible tools to implement Metropolis-Hastings, and usually with slightly more efficient proposal distributions, called **probabilistic programming languages** (PPLs). There exist many, but we will focus on `stan` and `nimble`.

Suppose we observe the following data:

```r
y <- rpois(20, lambda = 5)
y
```

```
 [1] 4 4 5 4 6 6 6 3 4 3 5 9 8 4 1 4 3 4 2 1
```

```r
tibble(y = y) %>% ggplot() + geom_dotplot(aes(x = y)) + theme_bw()
```



Let's use nimble and stan to estimate the λ parameter of the Poisson distribution. See the Nimble documentation and Stan documentation for help! In each of the following subsections, we

- Use the PPL to sample from the posterior distribution
- Assess whether the algorithm converged using trace plots and the **Gelman-Rubin statistic** ($\hat{R}$)
- Obtain posterior summaries and **highest density intervals**
- Assess how well the model fit by examining the **posterior predictive distribution**

Nimble

Step 1:

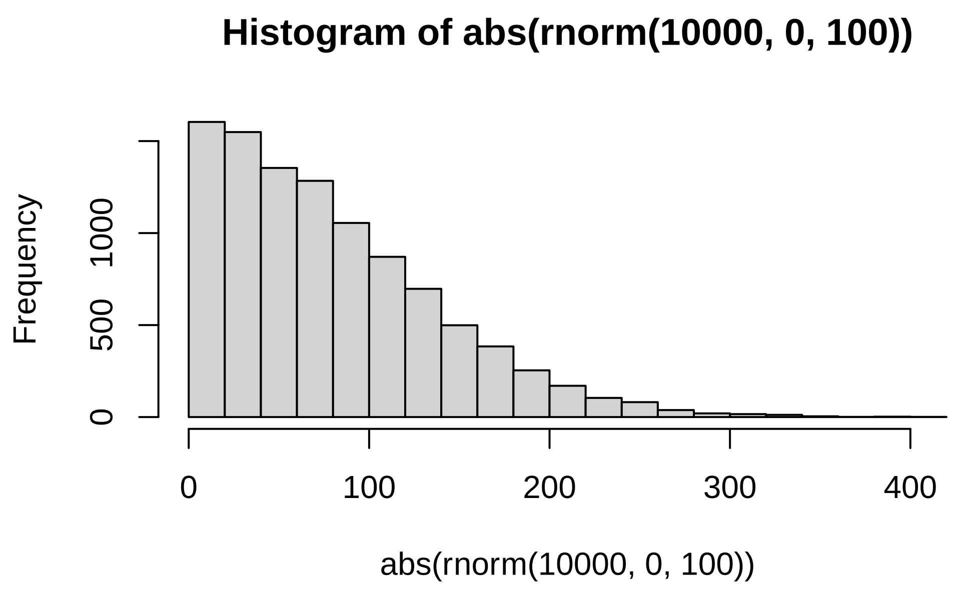
Likelihood/sampling model

$$y_i \sim \text{Poisson}(\lambda)$$

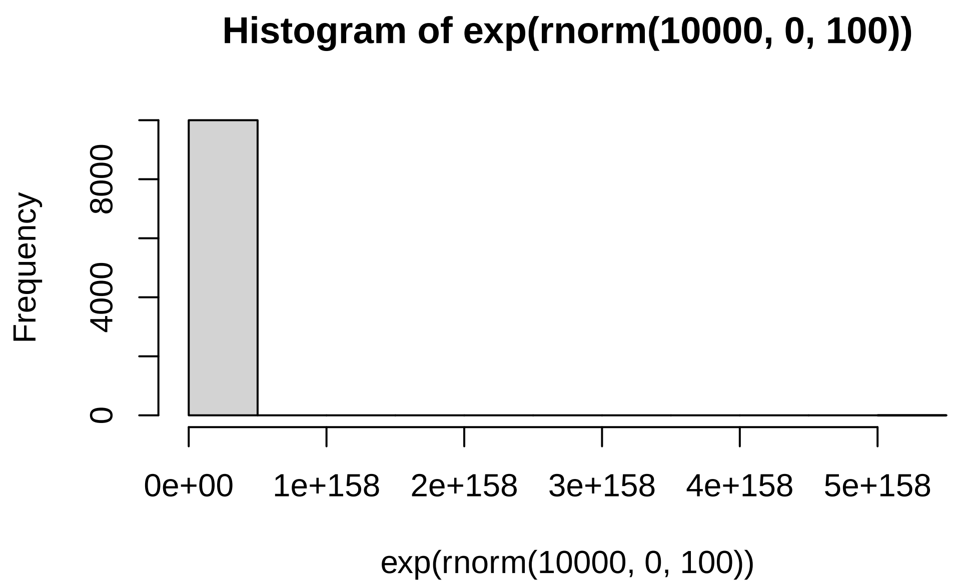
Priors (pick one of these)

$$\begin{aligned}\lambda &\sim \text{N}_{(0,\infty)}(0, 100)\\ \log \lambda &\sim N(0, 100)\\ \lambda &\sim \text{Uniform}(0, 10000)\\ \lambda &\sim \text{Gamma}(.001, .001)\end{aligned}$$

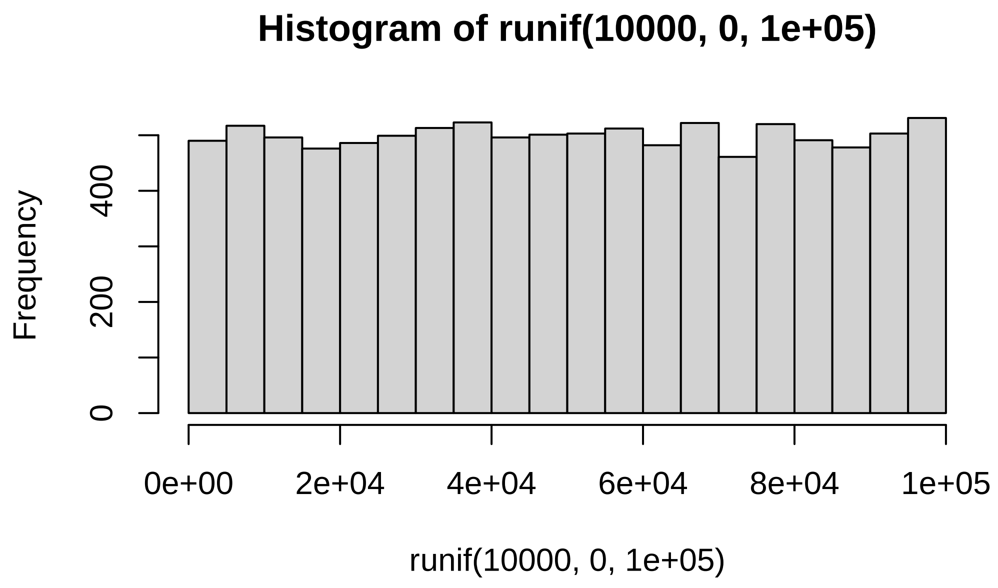
```r
# inspect priors
hist(abs(rnorm(10000, 0, 100))) # first prior
```



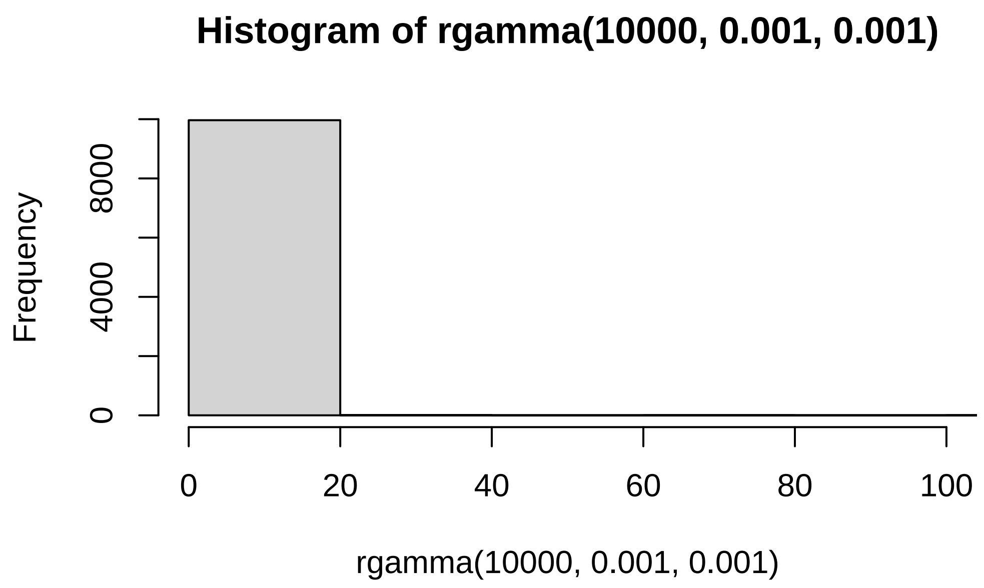
```r
hist(exp(rnorm(10000, 0, 100))) # second prior
```



```r
hist(runif(10000, 0, 100000)) # third prior
```



```r
hist(rgamma(10000, .001, .001), xlim = c(0, 100), breaks = 50) # four prior
```



```r
# nimble code for model
code <- nimbleCode({
  # priors
  lambda ~ T(dnorm(0, sd = 100), 0, Inf) # half-normal prior

  # likelihood as a loop
  for(i in 1:n){
    y[i] ~ dpois(lambda)
  }
})

# build model
rmodel <- nimbleModel(
  code = code, # model code
  data = list(y = y), # list of data (the random stuff)
  constants = list(n = length(y)),
  inits = list(lambda = 500)
) # model in R
cmodel <- compileNimble(rmodel) # model in C++

# build MCMC
mcmc_conf <- configureMCMC(cmodel)
rmcmc <- buildMCMC(mcmc_conf) # MCMC in R
cmcmc <- compileNimble(rmcmc, project = cmodel) # MCMC in C++

# run the MCMC
```

```
samples <- runMCMC(
  cmcmc,
  niter = 10000, # number of posterior samples
  nburnin = 5000, # how many posterior samples to discard
  thin = 1, # interval of samples to keep
  nchains = 4, # number of independent Markov chains to run
  samplesAsCodaMCMC = TRUE # nice formatting
)
```

```
# lots of options here - can use ggmcmc, bayesplot, coda, posterior, or by hand!
library(ggmcmc)
ggmcmc_samples <- ggs(samples)

# built in summary function
summary(samples)

# can also manipulate the samples manually!
do.call("rbind", samples) %>% colMeans()

# traceplots and convergence
## do they look like fuzzy catepillars?
### using ggmcmc
ggs_traceplot(ggmcmc_samples) + theme_bw()

### by hand
as_tibble(do.call("rbind", samples)) %>%
  pivot_longer(everything(), names_to = "param", values_to = "trace") %>%
  mutate(
    iter = rep(1:nrow(samples[[1]]), length(samples)),
    chain = rep(1:length(samples), each = nrow(samples[[1]])) %>% factor(),
  ) %>%
  ggplot() +
  geom_line(aes(x = iter, y = trace, col = chain)) +
  facet_wrap(~ param) +
  theme_bw()

# bayestestr? new to me!
library(bayestestR)
describe_posterior(samples)
hdi(samples) # this is nice

# posterior - new to me, but I like it
library(posterior)
samples_posterior <- as_draws_matrix(samples)
```

```r
posterior::summarise_draws(
  samples_posterior
)
```

```r
library(bayesplot)

# posterior predictive checks
samples_mat <- do.call("rbind", samples)
ppc <- matrix(NA, nrow(samples_mat), length(y))
for(i in 1:nrow(ppc)){
  ppc[i,] <- rpois(ncol(ppc), samples_mat[i,])
}

# check out a few
bayesplot::pp_check(
  y, ppc[sample(1:nrow(ppc), 100),], ppc_dens_overlay
)

# can do it by hand to make it look a little nicer
ppc_df <- t(rbind(y, ppc[sample(1:nrow(ppc), 100), ])) %>%
  as_tibble() %>%
  pivot_longer(everything(), names_to = "var", values_to = "val") %>%
  mutate(obs = ifelse(var == "y", "y", "yrep"))
ggplot() +
  geom_density(
    data = ppc_df %>% filter(obs != "y"),
    aes(x = val, group = var), col = "lightblue", alpha = .5
  ) +
  geom_density(
    data = ppc_df %>% filter(obs == "y"),
    aes(x = val), size = 1.1
  ) +
  theme_bw() +
  labs(x = "")
```

Stan

```r
# lets use cmdstanr
rm(list = ls()[-which(ls() == "y")])
library(cmdstanr)
# doesn't need to be this nicely formatted
stan_mod <- cmdstan_model(
  stan_file = write_stan_file(
    "
    data {
      int<lower=0> N;
      array[N] int<lower=0> y;
    }

    parameters {
      real<lower=0> lambda;
    }

    model {
      // likelihood
      y ~ poisson(lambda);

      // priors
      lambda ~ normal(0, 100);
    }
    "
  )
)
dat <- list(
  N = length(y),
  y = y
)
fit <- stan_mod$sample(
  data = dat,
  chains = 4,
  parallel_chains = 4
)
# see ?CmdStanMCMC
```

```r
# built-in summary is nice
library(posterior)
library(coda)
library(cmdstanr)
library(ggmcmc)
```

```r
samples_array <- fit$draws() # new array format
samples <- mcmc.list(
  lapply(1:posterior::nchains(samples_array), function(i) {
    mcmc(as.matrix(samples_array[,i,,drop = T]))
  })
) # old coda format
fit$summary()

# convergence
ggs_traceplot(ggs(samples)) + theme_bw()
as_tibble(do.call("rbind", samples)) %>%
  mutate(
    iter = rep(1:nrow(samples[[1]]), length(samples)),
    chain = rep(1:length(samples), each = nrow(samples[[1]])) %>% factor(),
  ) %>%
  dplyr::select(iter, chain, everything()) %>%
  pivot_longer(-c(1:2), names_to = "param", values_to = "trace") %>%
  arrange(param, chain, iter) %>%
  ggplot() +
  geom_line(aes(x = iter, y = trace, col = chain)) +
  facet_wrap(~ param, scales = "free_y", nrow = 2) +
  theme_bw()

# hdi
bayestestR::hdi(samples_array)

# kind of fun
library(shinystan)
launch_shinystan(fit)
```

```r
library(bayesplot)

# posterior predictive checks
samples_mat <- do.call("rbind", samples)
ppc <- matrix(NA, nrow(samples_mat), length(y))
for(i in 1:nrow(ppc)){
  ppc[i,] <- rpois(ncol(ppc), samples_mat[i,2])
}

# check out a few
bayesplot::pp_check(
  y, ppc[1:50,], ppc_dens_overlay
)
```

```r
# can do it by hand to make it look a little nicer
ppc_df <- t(rbind(y, ppc[sample(1:nrow(ppc), 50), ])) %>%
  as_tibble() %>%
  pivot_longer(everything(), names_to = "var", values_to = "val") %>%
  mutate(obs = ifelse(var == "y", "y", "yrep"))
ggplot() +
  geom_density(
    data = ppc_df %>% filter(obs != "y"),
    aes(x = val, group = var), col = "lightblue", alpha = .5
  ) +
  geom_density(
    data = ppc_df %>% filter(obs == "y"),
    aes(x = val), size = 1.1
  ) +
  theme_bw() +
  labs(x = "")
```

Summarizing

Upon estimating a model using MCMC, you should:

- Assess convergence using $\hat{R}$ and trace-plots
- Summarize the posterior to obtain posterior means/medians (point estimates) and highest density intervals/credibility intervals (uncertainty)
- Create posterior predictive checks to assess how well your model is fitting the observed data