Bayesian models - lab 1

< your name here >

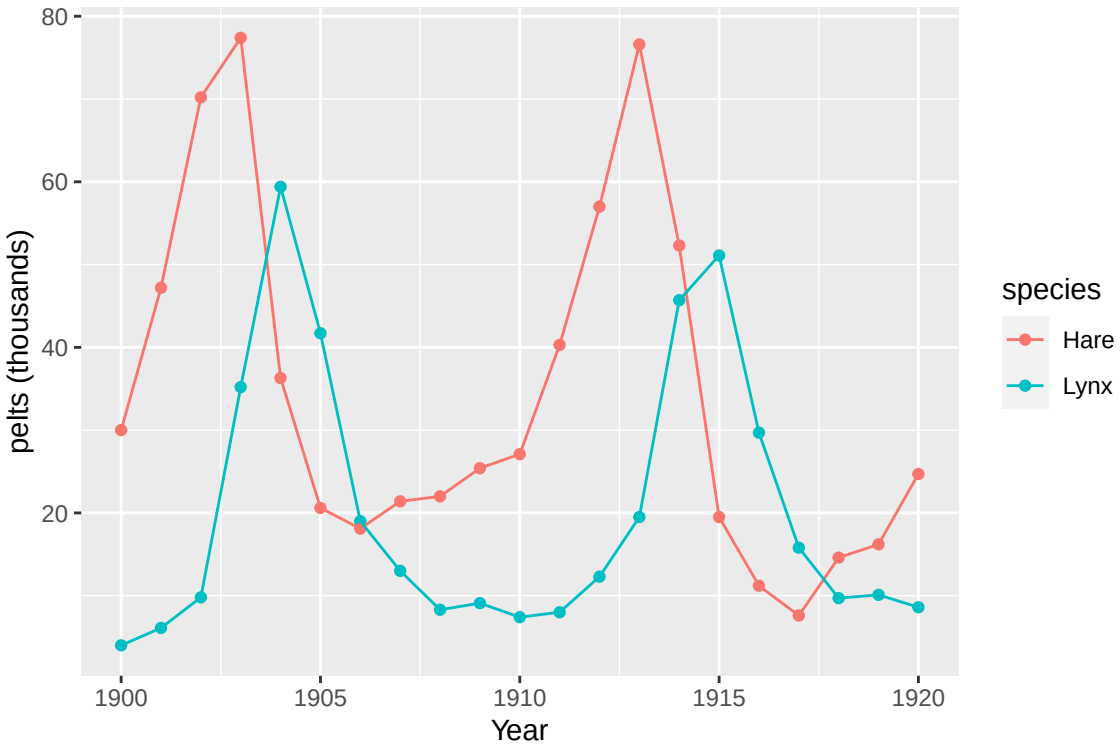
2/15/2021

Load and plot data

These data are 20 years of canadian lynx (*Lynx canadensis*) and snowshoe hare (*Lepus americanus*) pelts traded by the Hudson Bay company in Canada.

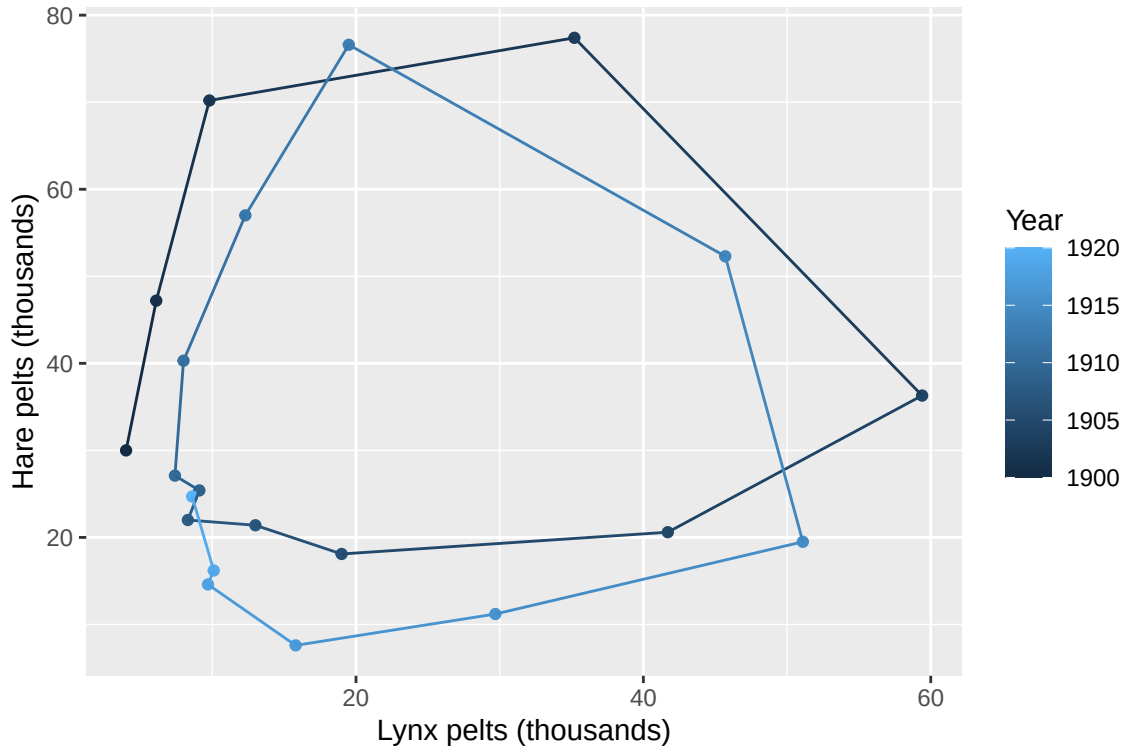
Time Series Plot

```
### Read in Hudson bay lynx and hare pelt data
lynx_hare <- read.csv("hudson-bay-lynx-hare.csv")

### Plot lynx and hare pelts over time
lynx_hare %>%
  pivot_longer(cols = c(Lynx, Hare), names_to = "species", values_to = "pelts") %>%
  ggplot(aes(Year, pelts, color = species)) +
  geom_point() +
  geom_line() +
  ylab("pelts (thousands)")
```



Plot: Lynx vs. Hare Pelts

```
lynx_hare %>%
  ggplot(aes(Lynx, Hare, color = Year)) +
  geom_point() +
  geom_path() +
  labs(x = "Lynx pelts (thousands)", y = "Hare pelts (thousands)")
```



Lotka-Volterra Predator-Prey Model

The Lotka-Volterra predator-prey model, which is the simplest model we can use to describe the coupled dynamics of a population of a prey species and the population of it's predator, is:

$$\frac{\mathrm{d}\mu_{1,t}}{\mathrm{d}t} = \alpha \times \mu_{1,t} - \beta \times \mu_{1,t} \times \mu_{2,t}$$
$$\frac{\mathrm{d}\mu_{2,t}}{\mathrm{d}t} = \delta \times \mu_{1,t} \times \mu_{2,t} - \gamma \times \mu_{2,t}$$

Where:

- The population size of the prey is μ_1 and the population size of the predator is μ_2.
- The parameter α is the intrinsic growth rate of the prey population. In the absence of predation, the prey population grows exponentially (i.e., without density dependence).
- The parameter β is the predation efficiency - the rate at which predators consume prey.
- The parameter δ is the rate of growth of the predator population, dependent on the rate of interaction between the predator and the prey (and thus, the number of prey). It can be written as the product of the "conversion efficiency" (i.e., the rate at which prey biomass is converted to predator biomass) and the predation rate β, but here we'll write it as its own parameter.

- The parameter γ is the loss rate of the predator population. In the absence of prey, the predator population declines exponentially to extinction.

What are some ways that we could make this model more realistic? List at least two. In particular, think about the ways in which the prey and predator populations are growing, and the relationship between prey density and predation here.

Bayesian Model Structure

Our Bayesian model needs three components:

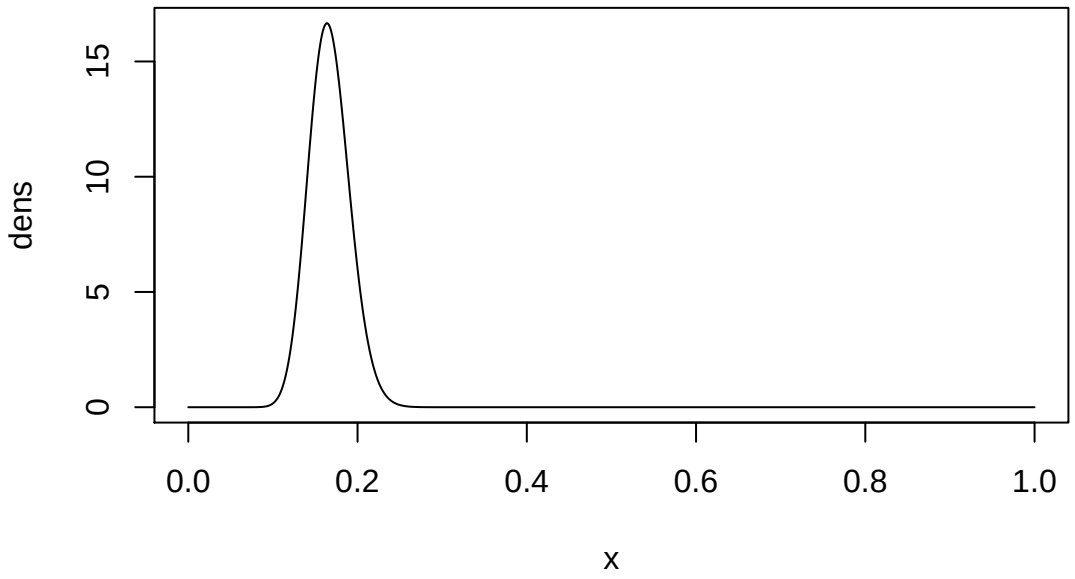
- A **deterministic model** describing the relationship between our variables (the number of hare pelts at time t - $N_{t,1}$, the number of lynx pelts at time t - $N_{t,2}$, and the time t) as a function of the parameters. The four main parameters are α, β, γ, and δ as described above, but we also need the **initial population sizes** (i.e., those at time $t = 0$, corresponding to the year 1900) - we'll denote these $\mu_{0,1}$ and $\mu_{0,2}$ for the hare and lynx populations, respectively.
- A **likelihood component** - i.e., the probabilistic model connecting the data to the deterministic model. For this, component, we'll assume that the observed number of pelts is lognormally distributed around log of the true population size $\mu_{t,i}$ multiplied by the trap rate p_i. The **trap rate** is the proportion of the population trapped in each time step. This is important because it connects our observed data - the number of pelts - to the true but unobserved (termed "latent") states of the lynx and hare populations. We'll denote this p_1 for the hares and p_2 for the lynx. Each species will also have it's own standard deviation for the lognormal distribution, σ_i.
- The **prior distributions** - we need these for all of the parameters above. We'll use relatively wide and uninformative priors for α, β, γ, δ, and μ_0, but we'll place a strong prior on the trap rate p_i - we'll give it a Beta distribution with a mean of about 0.15, indicating a 15% probability that any individual is trapped, and we won't put much variance around it.

$$
\begin{aligned}
N_{t,i} &\sim \text{Lognormal}(\ln(p_i \times \mu_{t,i}), \sigma_i) \; ; \; i \in \{1, 2\} \\
\mu_{t,i} &= \int_{j=0}^{t} \frac{\mathrm{d}\mu_{j,i}}{\mathrm{d}t} \; ; \; i \in \{1, 2\} \\
\frac{\mathrm{d}\mu_{1,t}}{\mathrm{d}t} &= \alpha \times \mu_{1,t} - \beta \times \mu_{1,t} \times \mu_{2,t} \\
\frac{\mathrm{d}\mu_{2,t}}{\mathrm{d}t} &= \delta \times \mu_{1,t} \times \mu_{2,t} - \gamma \times \mu_{2,t} \\
\mu_{0,i} &\sim \text{Lognormal}(\ln(10), 1) \; ; \; i \in \{1, 2\} \\
p_i &\sim \text{Beta}(40, 200) \\
\{\alpha, \gamma\} &\sim \text{Normal}(1, 0.5) \\
\{\beta, \delta\} &\sim \text{Normal}(0.05, 0.05) \\
\sigma_i &\sim \text{Exponential}(1) \; ; \; i \in \{1, 2\}
\end{aligned}
$$

Let's say that fur-trapping stopped in the year 1920, and you then went out for several years in a row and conducted a mark-recapture study and found that the actual capture rate was about 5%. How might you modify this model to accomodate that information? Play around with the `shape1` and `shape2` parameters of the Beta distribution (using the `dbeta` function) below to see if you can get a distribution with a peak around 5%.

```
shape1 <- 40
shape2 <- 200
x <- seq(0, 1, by = 0.001)
```

```r
dens <- dbeta(x, shape1, shape2)
plot(dens ~ x, type = "l")
```



Now let's say that you suspect lynx are easier to capture when there's fewer hares around, since you've put a little food in the traps and food is scarce. How might you modify the model above so that it accommodates this information? Which parameter would you change, and what additional parameters would you need it to depend on?

One glaring issue with this model is that it simultaneously assumes a capture rate of about 15%, but also assumes that the lynx and hare dynamics only depend on the number of lynx and hares. Let's assume that trapping occurs annually in the winter. How might you alter this model to accomodate that? No rigorous math required, just write down your thoughts.

Running the Model

Let's compile and run this model. Once we've written the model in Stan, it's just a matter of reading tha file in, compiling it with `stan_model`, storing all the data the model needs in a list, and then sampling from the model with the `sampling` function.

```r
### Compile Stan model
lokta_volterra <- stan_model("lotka-volterra.stan", model_name = "lotka_volterra")

### Put together data list
lv_data <- list(
  N = nrow(lynx_hare),
  pelts = lynx_hare[,c("Lynx", "Hare")]
```

```
)

### number of samples and warmup
n_warmup <- 1000
n_samples <- 5000

### Sample from posterior
lv_posterior <- sampling(
  lokta_volterra, lv_data, chains = 1,
  iter = n_samples + n_warmup, warmup = n_warmup
)
```

```
##
## SAMPLING FOR MODEL 'lotka_volterra' NOW (CHAIN 1).
## Chain 1: Rejecting initial value:
## Chain 1:   Error evaluating the log probability at the initial value.
## Chain 1: Exception: lognormal_lpdf: Location parameter is nan, but must be finite!  (in 'model63244e
##
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 6000 [  0%]  (Warmup)
## Chain 1: Iteration:  600 / 6000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1001 / 6000 [ 16%]  (Sampling)
## Chain 1: Iteration: 1600 / 6000 [ 26%]  (Sampling)
## Chain 1: Iteration: 2200 / 6000 [ 36%]  (Sampling)
## Chain 1: Iteration: 2800 / 6000 [ 46%]  (Sampling)
## Chain 1: Iteration: 3400 / 6000 [ 56%]  (Sampling)
## Chain 1: Iteration: 4000 / 6000 [ 66%]  (Sampling)
## Chain 1: Iteration: 4600 / 6000 [ 76%]  (Sampling)
## Chain 1: Iteration: 5200 / 6000 [ 86%]  (Sampling)
## Chain 1: Iteration: 5800 / 6000 [ 96%]  (Sampling)
## Chain 1: Iteration: 6000 / 6000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 16.818 seconds (Warm-up)
## Chain 1:                80.2 seconds (Sampling)
## Chain 1:                97.018 seconds (Total)
## Chain 1:
```

Checking model fit

Markov Chain Monte Carlo (and in particular, the version of MCMC that Stan uses) provides a *lot* of ways for telling if things went wrong in the model, but if you got this far without a warning from the sampler, than chances are you're good. Nonetheless, let's look at the most common method for diagnosing convergence - traceplots.

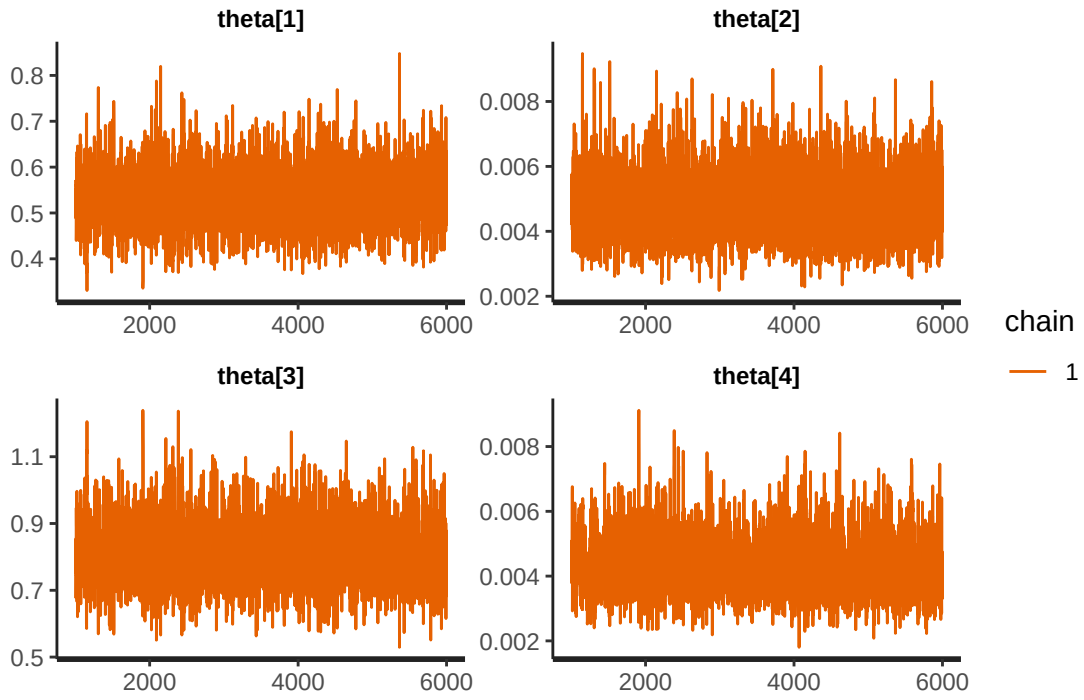
Traceplots

Traceplots show the position of the sampler in the posterior distribution / parameter space on the y-axis, and the iteration number on the x-axis. What we want to see is something akin to a "fuzzy caterpillar," which

indicates that the sampler is jumping around the posterior distribution more-or-less randomly. If the chains stay in the same place for a long time, or wander around but never settle around one area, than the model hasn't converged.

Let's plot the traceplots for α, β, γ, and δ, which in this model are stored as theta 1-4. These look pretty darn good:

```
### Plot chains for parameters
traceplot(lv_posterior, pars = paste0("theta[", 1:4, "]"))
```



Parameter summary

We can get summaries of the posterior distributions of these parameters using the `summary` function. These include the posterior means, standard errors, and quantiles, but also a couple of statistics to summarize model fit - the effective sample size `n_eff`, which is a metric of the number of samples from the posterior adjusted for autocorrelation in the chains - and `Rhat`, which is a metric of within and between chain convergence - ideally, this is *very* close to 1.

```
### Check parameter values and convergence
lv_summary <- summary(lv_posterior)$summary
lv_summary[c(paste0("theta[", 1:4, "]")),]
```

```
##                mean      se_mean           sd        2.5%         25%
## theta[1] 0.530936730 1.293257e-03 0.0629581060 0.414210289 0.487214693
## theta[2] 0.004764183 1.927058e-05 0.0009866317 0.003097876 0.004075263
## theta[3] 0.810257589 2.020929e-03 0.0943043984 0.642835140 0.745540310
## theta[4] 0.004312211 1.849320e-05 0.0008766390 0.002832260 0.003698025
##                50%         75%       97.5%    n_eff      Rhat
## theta[1] 0.528217859 0.570454087 0.665189201 2369.920 1.0000252
## theta[2] 0.004669607 0.005363387 0.006921309 2621.322 0.9998551
## theta[3] 0.803954673 0.869947920 1.015660995 2177.518 1.0000308
## theta[4] 0.004246644 0.004831436 0.006247543 2247.075 0.9998035
```

Posterior Distributions

Extract Posterior Distributions

Let's extract the posteriors for the paramters in our deterministic model - these are α, β, γ, and δ, but also the initial population sizes.

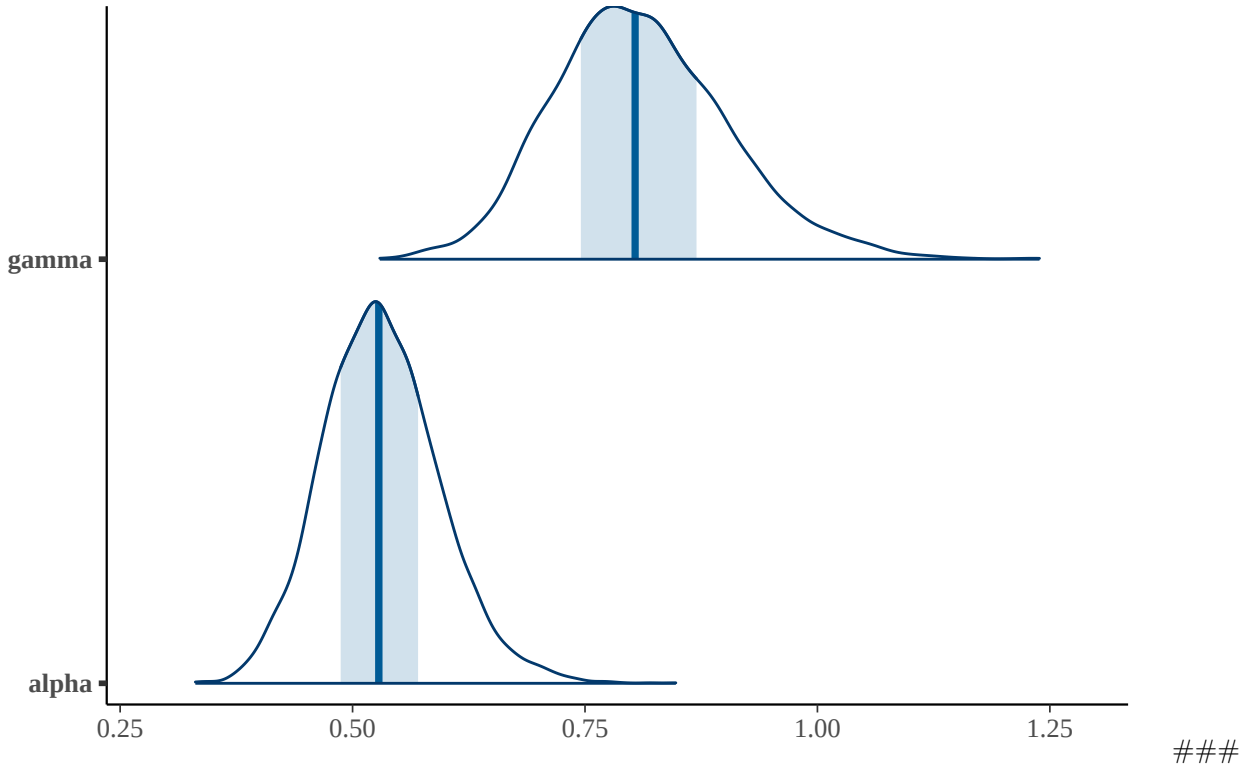
```
par_names <- c(
  paste0("theta[", 1:4, "]"), ## alpha, beta, gamma, delta
  c("pop_init[1]", "pop_init[2]") ## mu_0 for prey & predator
)

par_samples <- extract(lv_posterior, pars = par_names, inc_warmup = FALSE)
names(par_samples) <- c("alpha", "beta", "gamma", "delta", "mu2_0", "mu1_0")
```

Plot Posterior Distributions

alpha and gamma

Here are the posterior distributions for α, the intrinsic growth rate of the hare population, and γ, the loss rate for the lynx population. What do the differences between these distributions suggest?

```
mcmc_areas(lv_posterior, c("theta[1]", "theta[3]")) +
  scale_y_discrete(breaks = c("theta[3]", "theta[1]"), labels = c("gamma", "alpha"),
                   expand = c(0.05, 0))
```



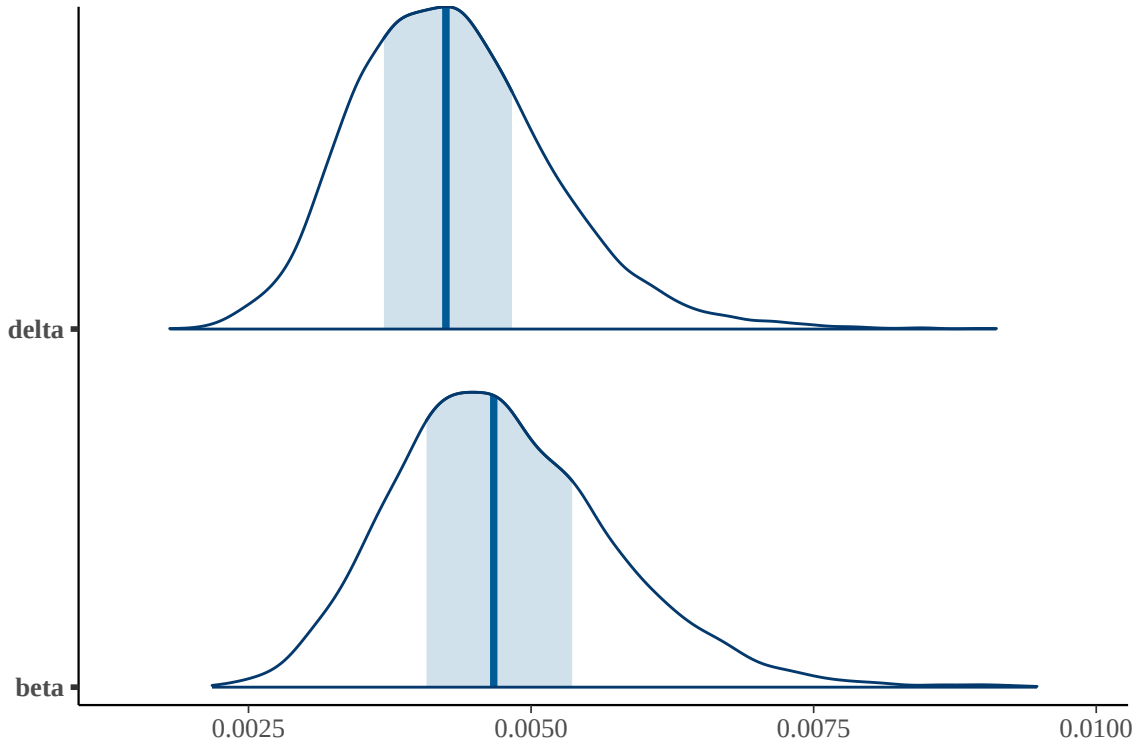
###

beta and delta

Here are the posterior distributions for β, the predation efficiency, and δ, the conversion efficiency from predators to prey. Does it make sense that the mean of the posterior distribution for δ is nearly as large as that of β? Why or why not?

Can you think of a possible reason why these distributions are so close?

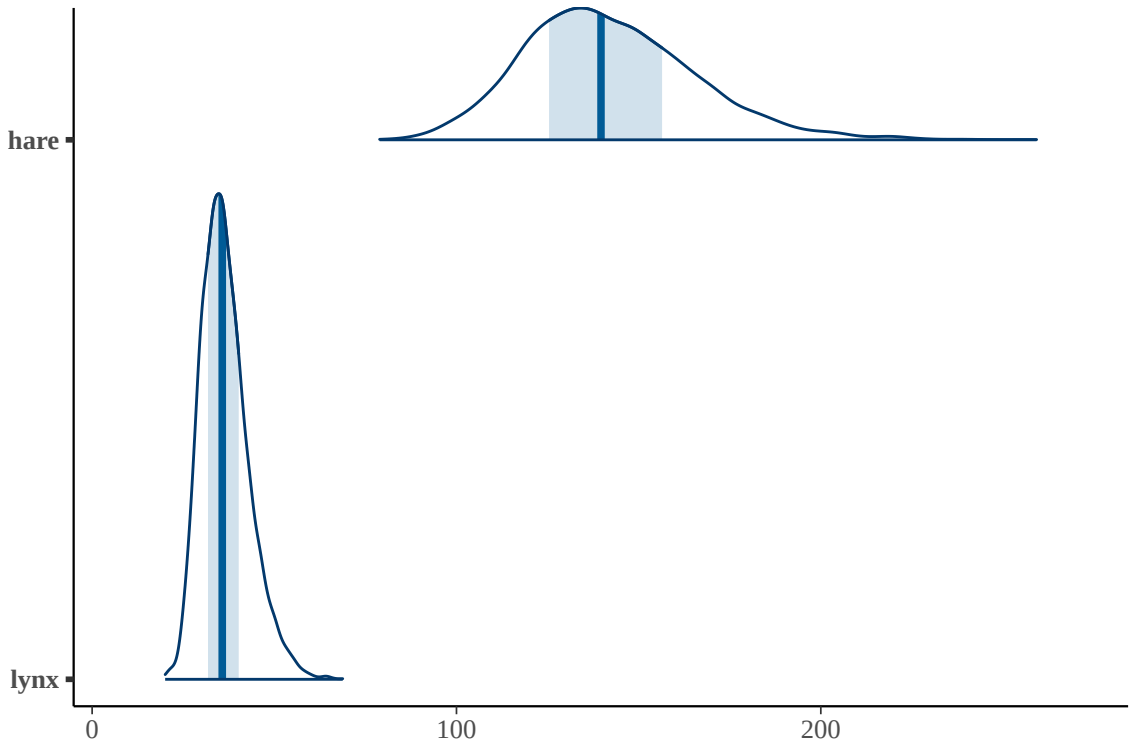
```
mcmc_areas(lv_posterior, c("theta[2]", "theta[4]")) +
  scale_y_discrete(breaks = c("theta[4]", "theta[2]"), labels = c("delta", "beta"),
                   expand = c(0.05, 0))
```



Initial Population Sizes

Here are the posterior distributions for the initial hare population size, $\mu 1$, and the initial lynx population size, μ_2. We can see that the initial population size for the hares is much larger than for the lynxes, as we'd expect.

```
mcmc_areas(lv_posterior, c("pop_init[1]", "pop_init[2]")) +
  scale_y_discrete(breaks = c("pop_init[2]", "pop_init[1]"), labels = c("hare", "lynx"),
                   expand = c(0.05, 0))
```



Posterior Predicted Population

Obtaining Posterior Predictions

We can obtain the posterior distribution for the population size at time t of both the hare and lynx by using the posterior distributions for each of the above parameters, which we've stored in `par_samples`. For each sample, we just integrate the lotka-volterra model over time, and store the simulated populations that result.

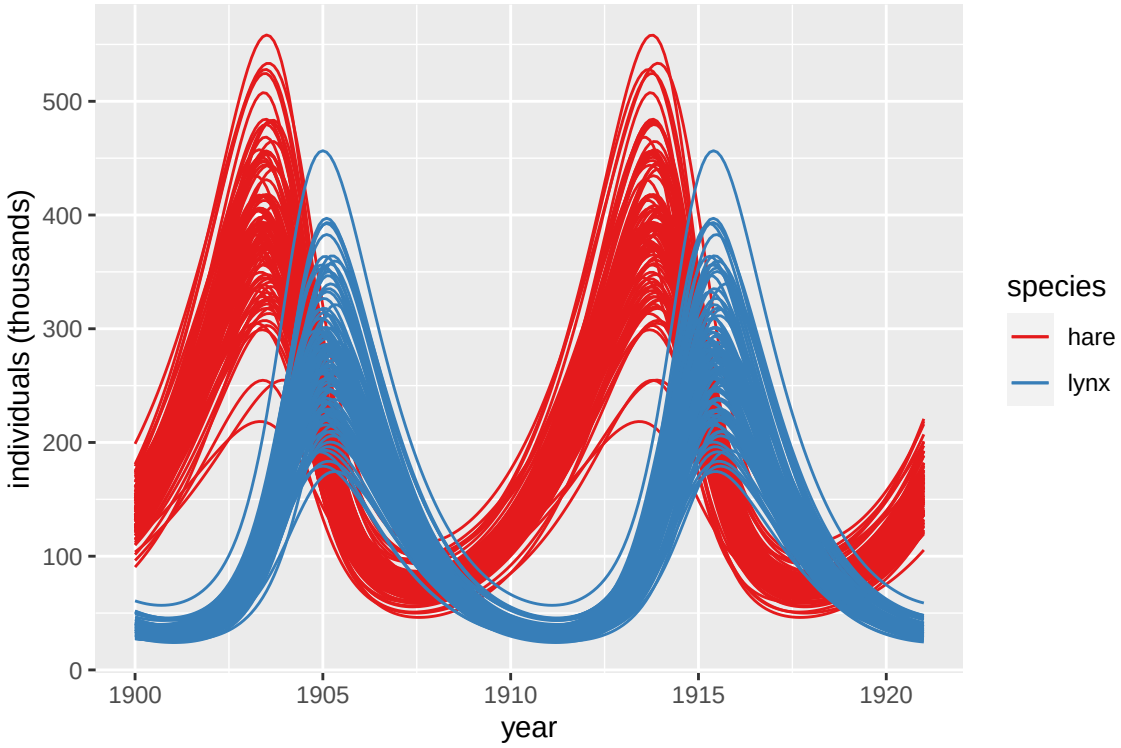
```
## Lotka-Volterra ODE function
lotka_volterra_ode  <- function(t, mu, par) {

  mu1 <- mu[[1]]  # population size for prey
  mu2 <- mu[[2]]  # population size for predator

  with(as.list(par), {
    dmu1_dt <- alpha * mu1 - beta * mu1 * mu2
    dmu2_dt <- delta * mu1 * mu2 - gamma * mu2
    return(list(c(dmu1_dt, dmu2_dt)))
  })
}

## time points to integrate over
time_step <- 0.1
time <- seq(0, nrow(lynx_hare), by = time_step)

## empty list to store simulated populations
mu_t_posterior <- vector("list", n_samples)

## For each sample, store the simulated population
```

```r
for (i in 1:n_samples) {

  pars <- with(par_samples, c(
    alpha = alpha[i], beta = beta[i], gamma = gamma[i], delta = delta[i]
  ))

  mu <- with(par_samples, c(mu1 = mu1_0[i], mu2 = mu2_0[i]))

  ## integrate population over time and store in list
  mu_t_posterior[[i]] <- as.data.frame(
    cbind(lsoda(mu, time, lotka_volterra_ode, pars), sample = i)
  )

}

## bind samples together
mu_t_posterior <- do.call("rbind", mu_t_posterior)
```

Time Series of Posterior Samples

One way to visualize the posterior for this model is to just plot the simulated population trajectories - here, I'll plot the first 100 samples.

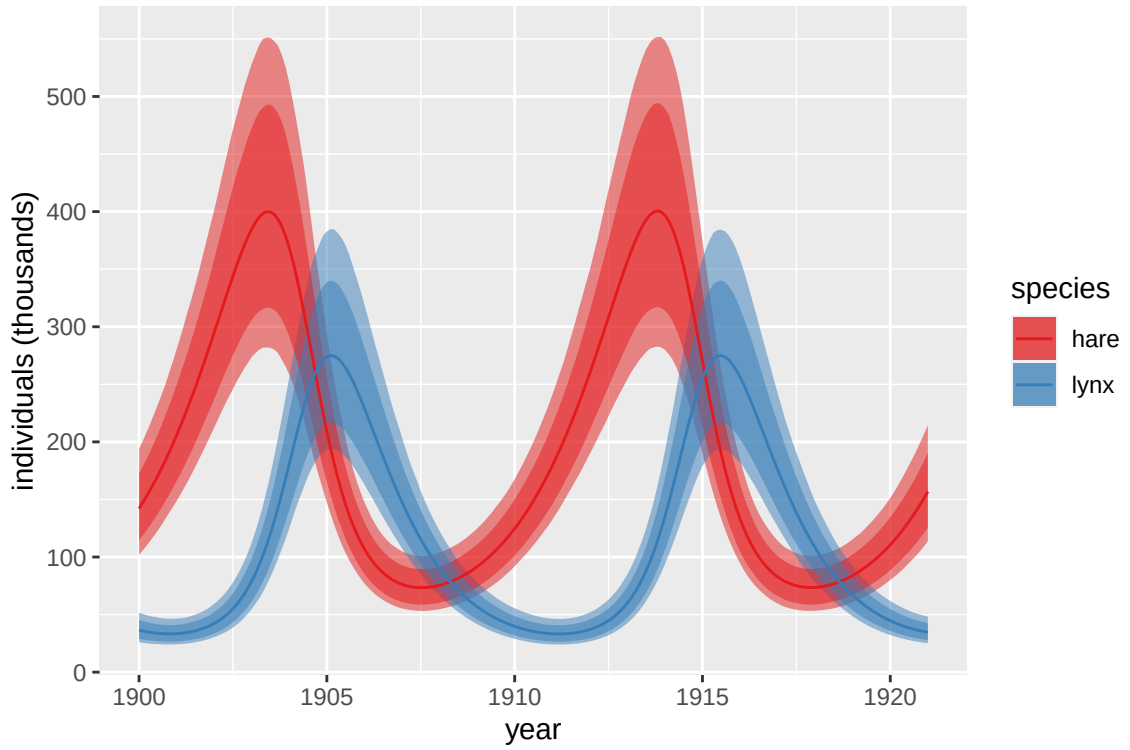
```r
mu_t_posterior %>%
  filter(sample <= 100) %>%
  mutate(time = time + min(lynx_hare$Year)) %>%
  ggplot(aes(time, mu1, group = sample)) +
  geom_line(aes(color = "hare")) +
  geom_line(aes(y = mu2, color = "lynx")) +
  scale_color_brewer(palette = "Set1") +
  labs(x = "year", y = "individuals (thousands)", color = "species")
```



Compatability Intervals

Another way we can do this is to compute quantiles for the populations at each time step, and plot those. Here, I plot the means of the posterior distributions, as well as 80% and 95% "compatability intervals," also known as "credible intervals" - these are, essentially, the Bayesian equivalent of confidence intervals. Where confidence intervals (calculated for the maximum likelihood) represent a region of the distribution of the maximum likelihood estimate, these represent regions of the posterior distribution.

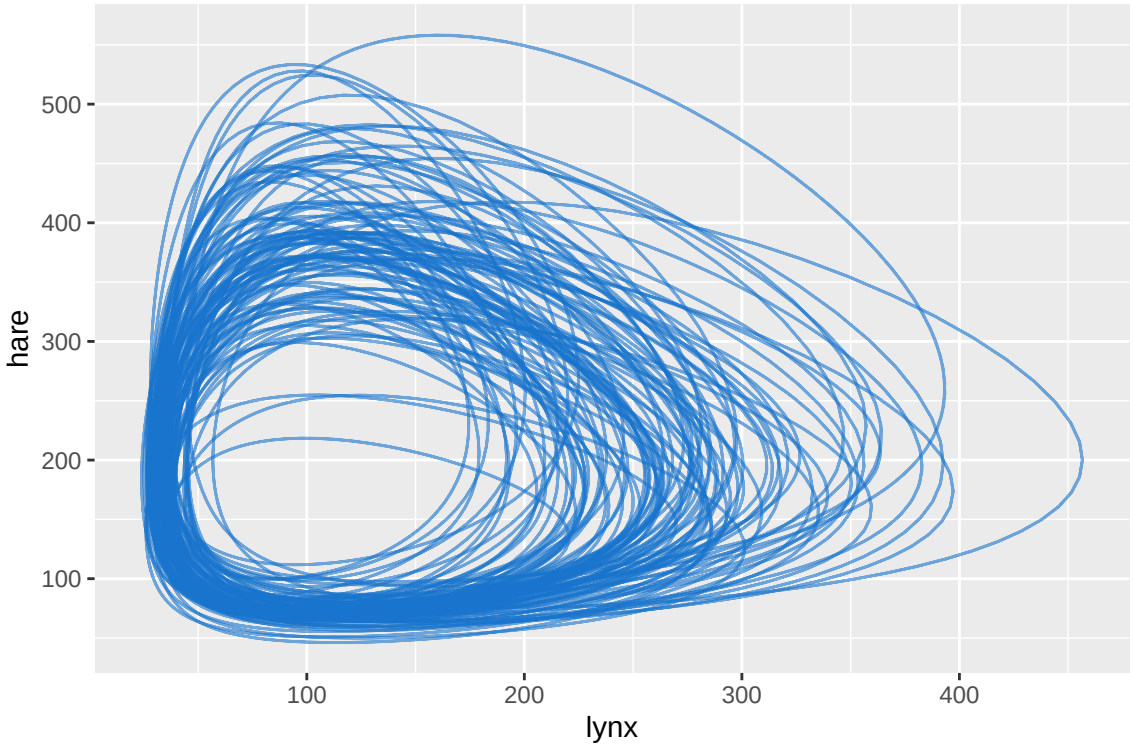
```
mu_t_summary <- mu_t_posterior %>%
  pivot_longer(cols = c(mu1, mu2), names_to = "species", values_to = "mu") %>%
  mutate(species = ifelse(species == "mu1", "hare", "lynx"),
         time = time + min(lynx_hare$Year)) %>%
  group_by(species, time) %>%
  summarize(mean = mean(mu),
            `2.5%` = quantile(mu, 0.025),
            `10%` = quantile(mu, 0.1),
            `90%` = quantile(mu, 0.9),
            `97.5%` = quantile(mu, 0.975))

mu_t_summary %>%
  ggplot(aes(time, mean, fill = species)) +
  geom_ribbon(aes(ymin = `2.5%`, ymax = `97.5%`), alpha = 0.5) +
  geom_ribbon(aes(ymin = `10%`, ymax = `90%`), alpha = 0.5) +
  geom_line(aes(color = species)) +
  labs(x = "year", y = "individuals (thousands)") +
  scale_color_brewer(palette = "Set1") +
  scale_fill_brewer(palette = "Set1")
```



Population Orbits

Plotting the lynx population size against the hare population size for each of the first 100 samples from the posterior distribution:

```
mu_t_posterior %>%
  filter(sample <= 100) %>%
  rename(hare = mu1, lynx = mu2) %>%
  ggplot(aes(lynx, hare, group = sample)) +
  geom_path(color = "dodgerblue3", alpha = 0.6)
```


500
400
300
200
100
hare
100
200
300
400
lynx