Dynamic range model in Stan

Alexa Fredston and Malin Pinsky

begun November 2020

General notes

Useful sources for Bayesian population models and Bayesian implementation in R that we referenced while building this script:

- https://github.com/DanOvando/learn-stan
- https://arxiv.org/src/2002.02001v1/anc/Appendix_S1.pdf p. 72
- https://cchecastaldo.github.io/BayesianShortCourse/Syllabus.html
- https://mc-stan.org/docs/2_25/stan-users-guide/mark-recapture-models.html
- https://mc-stan.org/docs/2_25/functions-reference/nbalt.html (and entire manual)

Model notes

- Stan can't estimate discrete latent variables, so we've moved N from being discrete to continuous

To-do (short term)

- Revisit choices of initial values, especially when adding in dimensions over which variables are indexed, which will reduce the total number of individuals in each cell
- Add random Poisson noise around estimates of N every year (took this out to ensure Stan model was working correctly)
- Move away from uniform priors (done?)
- Figure out how to deal with true NAs in data, which are instances with no samples of a patch in a year (not true zeroes, although I've replaced them with zeroes temporarily)
- Add stage structure and transition rates
- Add in dispersal between patches
- Replace r (and/or dispersal rates) with temperature-dependent functions
- Choose data models for all species (currently just modeling fluke)

dan's changes

OK, there's a lot going on here.

- change estimation of N to estimation of process error terms
- basically get rid of population model. . . .

What would this ideally look like?

If you're observing total N across all age classes, you could just estimate N in each time step as a random walk / AR processm, where you can set an autocorrelation strength...

So then you can estimate a CV and an autocorrelation

stage-prep model

Tried this version, but doesn't really work with just one stage. Once we get multipple stages, can separate out recruitment and mortality.

here is a suggestion for an alternative model structure. Following the idea here, this is a simplified version with 1 "stage", but can easily be generalized to multiple stages with length transition matrices

for time t and patche p, estimate

$$n_{t=1,p} \sim negbinom(x, sigma_{obs})$$

for future time steps

$$n_{t,p} = n_{t-1,p}e^{-m} + r_{t,p}$$

where r is recruits is an AR1 process

raw are independent recruitment deviates

$$raw_t \sim lnorm(-\sigma_r^2/2, \sigma_r)$$

$$r_{t=1,p} = raw_{t=1}$$

and

$$r_{t,p} = \alpha \times r_{t-1,p} + \sqrt{1-\alpha^2} \times raw_t)$$

AR1 Model

To keep is simpler for now, shifting this to just an autoregressive (AR) 1 model, where process error in time step t is potentially correlated with process error in time t - 1.

This works, and I think lays the foundation for

for time t and patche p, estimate

$$n_{t=1,p} \sim negbinom(x, sigma_{obs})$$

for future time steps

$$n_{t,p} = n_{t-1,p}e^{r_{t,p}}$$

where r is is an AR1 process error term

raw are independent recruitment deviates

$$raw_t \sim norm(-\sigma_r^2/2, \sigma_r)$$

$$r_{t=1,p} = raw_{t=1,p}$$

and

$$r_{t,p} = -\sigma_r^2/2 + \alpha \times (r_{t-1,p} - -\sigma_r^2/2) + raw_{t=1,p}$$

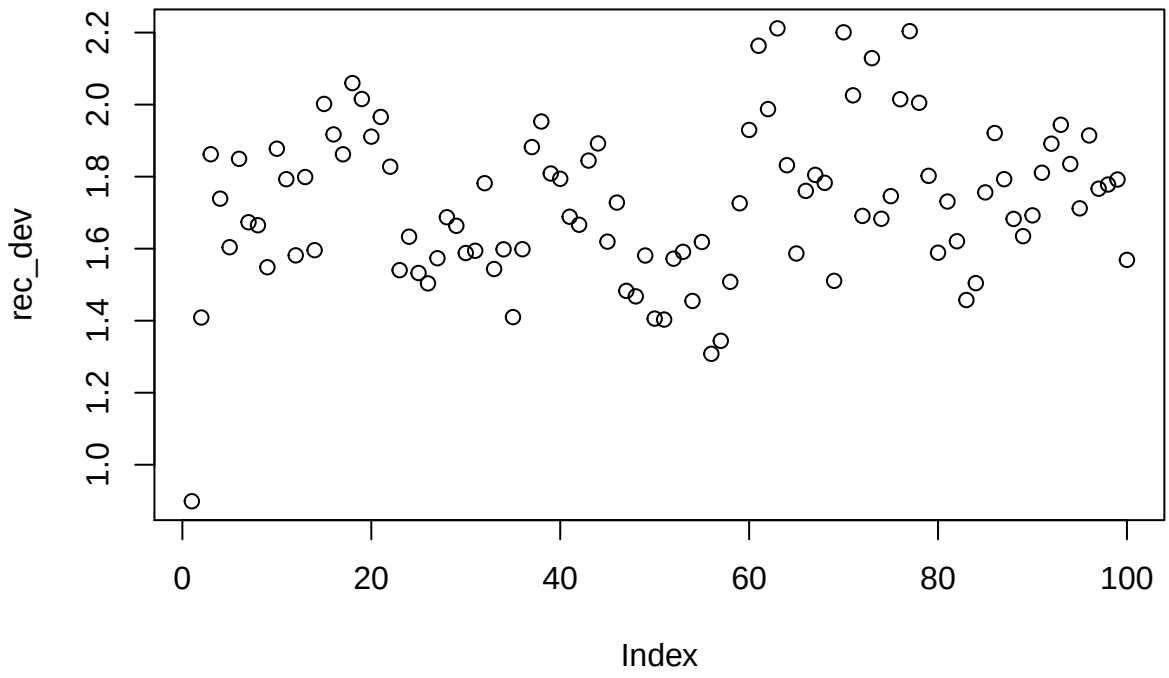
and

$$\sigma_{obs} \sim normal(0.75, 0.25)$$

$$\sigma_r \sim normal(.5, .25)$$
$$\alpha \sim normal(0, .25)$$

From there, we generate posterior predictive to the fits, and then use the posterior predictive to generate the predictions for the future, where now we generate both process error and observation error for the projected time steps

```
sigma_r <- 0.2

p <- .5

y <- 100

rec_dev <- rep(0, y)

rec_dev[1] <- rlnorm(1, -sigma_r ^ 2 / 2, sigma_r)

for (t in 2:y) {
  rec_dev[t] <-
    rec_dev[t - 1] * p + rlnorm(1, -sigma_r ^ 2 / 2, sigma_r) * sqrt(1 - p ^ 2)
}

plot(rec_dev)
```



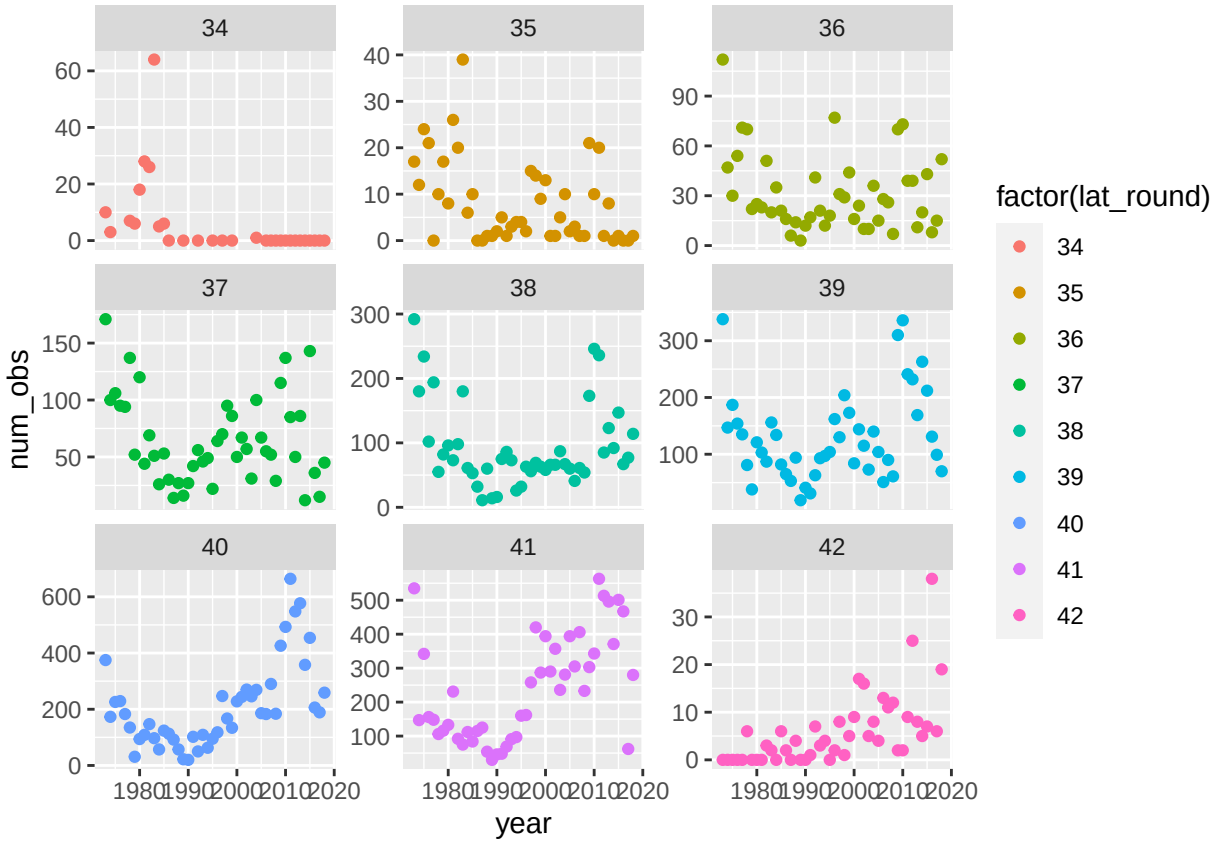
So just estimate a vector of recruitemtn deviates with the prior defined as above

```
#library(rjags)

#library(MCMCvis)

# dogfish <- read_csv(here("processed-data","dogfish_prepped_data.csv"))

# dogfishTest <- dogfish %>%
#   filter(lengthclass=="adult") %>%
#   group_by(year) %>%
#   mutate(num_obs = sum(numlengthclass),
#          sd_obs = sd(numlengthclass)) %>% # this is standard deviation of different counts in a year,
#   select(year, num_obs, sd_obs) %>%
#   distinct()
# moving away from spiny dogfish because it is massively overdispersed and no single distribution will

fluke <- read_csv(here("processed-data","fluke_prepped_data.csv"))

fluke_lat_time <- fluke %>%
  mutate(lat_round = round(lat)) %>%
  filter(lengthclass=="adult",
         year>=1973,
         lat_round >= 34,
         lat_round < 43 # because we're just using nefsc survey for now, which rarely goes below 34N
         ) %>%
  group_by(year, lat_round) %>%
```

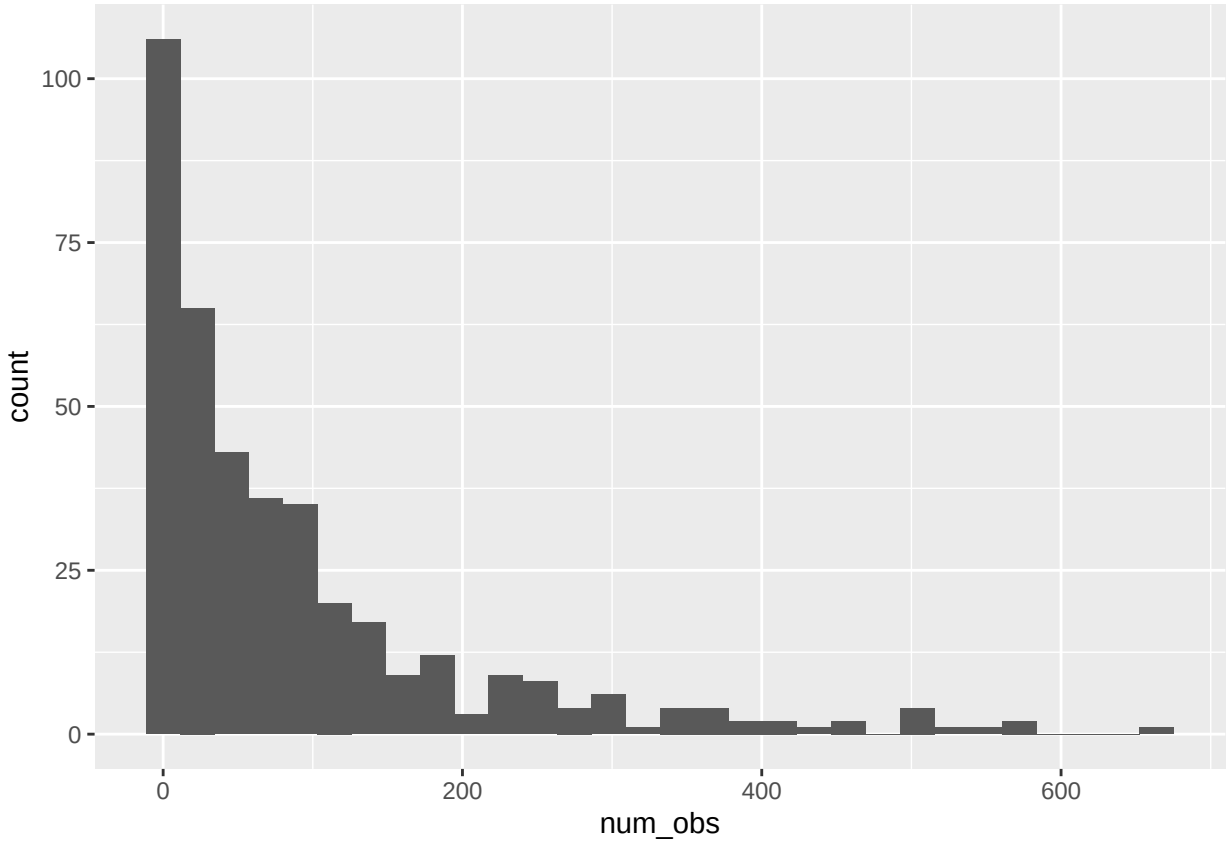
```
  summarise(num_obs = sum(numlengthclass)) %>%
  ungroup()

fluke_train <- fluke_lat_time %>%
  filter(year <= 2012)

fluke_test <- fluke_lat_time %>%
  filter(year > 2012)


head(fluke_lat_time)
```

```
## # A tibble: 6 x 3
##    year lat_round num_obs
##   <dbl>     <dbl>   <dbl>
## 1  1973        34      10
## 2  1973        35      17
## 3  1973        36     112
## 4  1973        37     171
## 5  1973        38     292
## 6  1973        39     338
```

```
fluke_lat_time %>%
ggplot(aes(year, num_obs, color = factor(lat_round))) +
  geom_point() +
  facet_wrap(~lat_round, scales = "free_y")
```




```
fluke_lat_time %>%
ggplot(aes(num_obs)) +
  geom_histogram()
```



Here's the stan script:

```
data{
int<lower=1> len_t; // the number of time points

int<lower=0> len_i; // number of patches

int<lower=0> y[len_i, len_t]; // defining y as an array of integers with patches as rows and years as c

// data inputs
vector<lower=0>[len_i] z0; // vector of starting pop. values, one per patch

}

//transformed data{
//}

parameters{
row_vector[len_i] log_mean_rec; // average number of recruits per year

real<lower=0> sigma_r; // recruitment deviates CV

real<lower=0> phi_obs; // observation error

matrix[len_i,len_t-1] raw; // array of raw recruitment deviates
```

```
real<lower = 0, upper = 1> alpha; // autocorrelation parameter

real log_m; // average mortality, this is really not right... eventually would need to fix M and estima

vector<lower=1e3, upper=1e7>[len_i] gamma_shape0; // gamma parameter for process model at time step 1 -
vector<lower=1, upper=1e3>[len_i] gamma_rate0; // gamma parameter for process model at time step 1 - ma

}

transformed parameters {

matrix[len_i, len_t] y_hat; // estimated numbers in patch i and timestep t

matrix[len_i,len_t-1] rec_dev; // array of realized recruitment deviates

real m;

row_vector[len_i] mean_rec;

mean_rec = exp(log_mean_rec);

m = exp(log_m);

y_hat[1:len_i,1] = z0;

for (t in 2:len_t){

  if (t == 2){
  rec_dev[1:len_i,t-1]  =  raw[1:len_i,1];
  } else {

  rec_dev[1:len_i,t-1] = alpha * rec_dev[1:len_i,t-2] + sqrt(1 - pow(alpha,2)) * raw[1:len_i,t-1]; // r

  }
  y_hat[1:len_i,t] = y_hat[1:len_i,t-1] * exp(-m) + mean_rec * exp(rec_dev[1:len_i,t-1]); // calculate

} // close time loop

}

model{

phi_obs ~ normal(0.75, 0.25); // from https://mc-stan.org/docs/2_20/functions-reference/nbalt.html  phi

log_m ~ normal(log(0.2),.05); // natural mortality prior

sigma_r ~ cauchy(0,.5); // process error prior

log_mean_rec ~ normal(log(50),2); // prior on mean number of recruits per patch

raw ~ normal(pow(-sigma_r,2)/2,sigma_r);

// observation model
for(t in 1:len_t) {
```

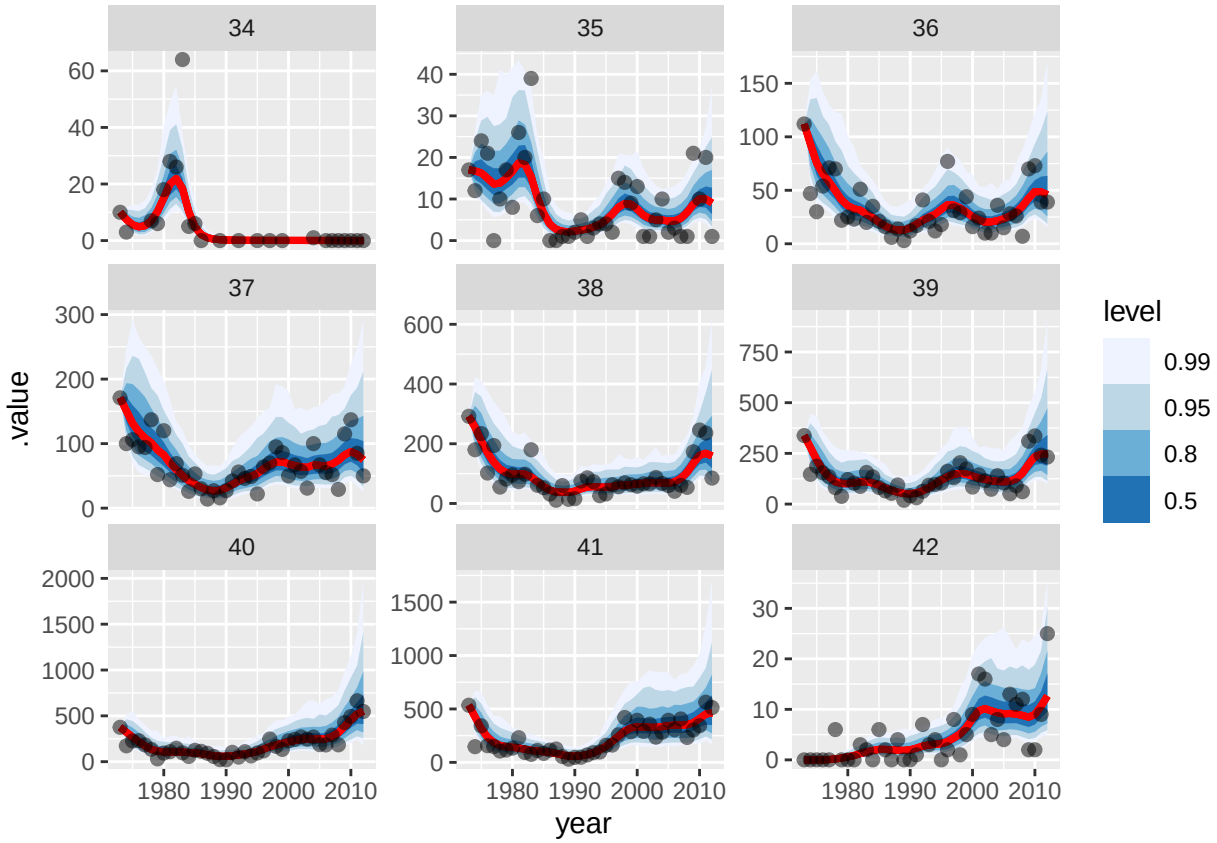
```
    for(i in 1:len_i){

      //raw[i,t] ~ normal(pow(-sigma_r,2)/2,sigma_r); // prior on raw process error

      y[i,t] ~ neg_binomial_2(y_hat[i,t], phi_obs); // this version of neg binom has a more familiar fo

    }


}

}
```

And model implementation from R:

Evaluating the model:

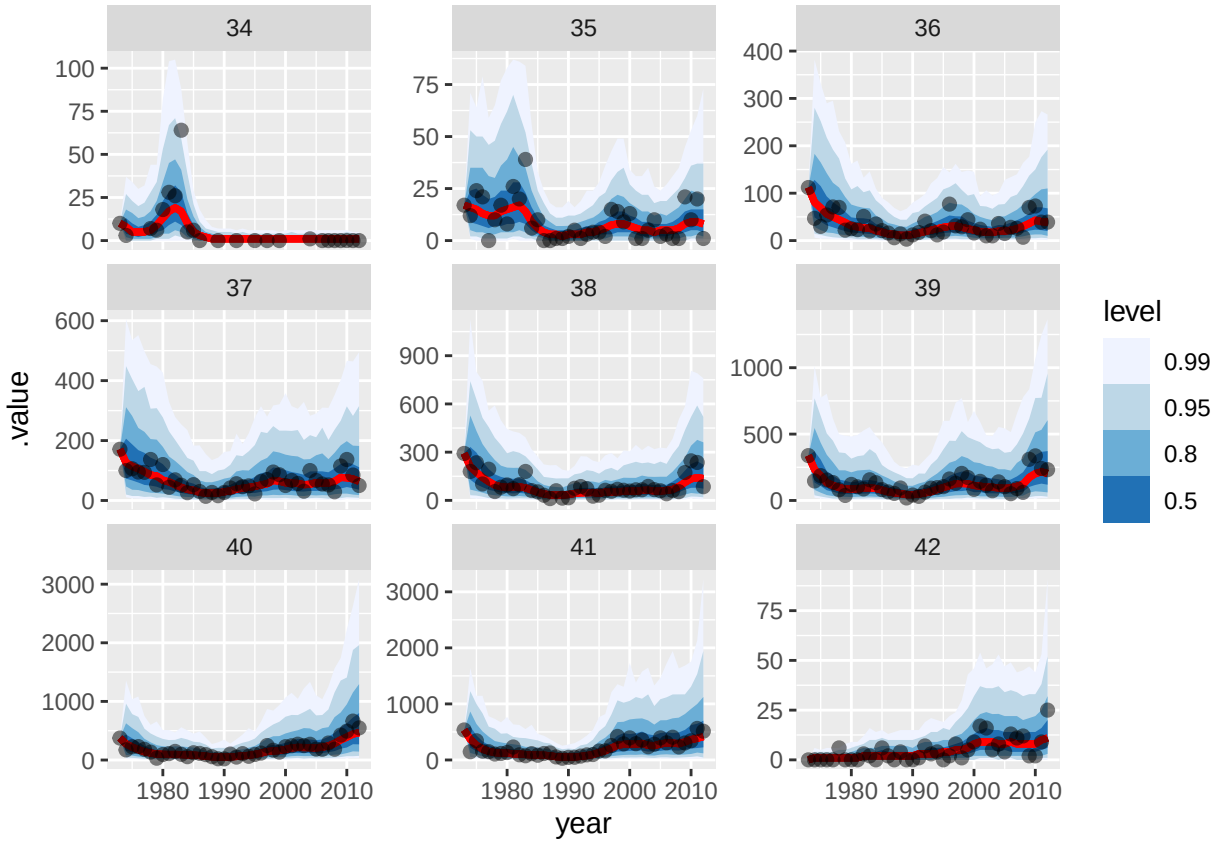
```
# summary(model1_fit)
# plot(model1_fit)
# check_divergences(model1_fit)
# rstanarm::launch_shinystan(model1_fit)

# convert to a df


predicted_fluke <- tidybayes::gather_draws(model1_fit,y_hat[patch,year], n = 1000) %>%
         mutate(patch = as.numeric(patch),
         lat = patch + min(fluke_lat_time$lat_round) - 1,
         year = year + min(fluke_lat_time$year) - 1)

pp_predicted_fluke <- tidybayes::gather_draws(model1_fit,pp_y_hat[patch,year], n = 1000) %>%
         mutate(patch = as.numeric(patch),
         lat = patch + min(fluke_lat_time$lat_round) - 1,
         year = year + min(fluke_lat_time$year) - 1)

pp_projected_fluke <- tidybayes::gather_draws(model1_fit,pp_proj_y_hat[patch,year], n = 1000) %>%
         mutate(patch = as.numeric(patch),
         lat = patch + min(fluke_lat_time$lat_round) - 1,
         year = year + min(fluke_test$year) - 1)
```

```
predicted_fluke %>%
  ggplot() +
  stat_lineribbon(aes(x = year, y = .value),.width = c(.99, .95, .8, .5), color = "red") +
  geom_point(data = fluke_train %>% rename(lat = lat_round), aes(year, num_obs), size = 2,alpha = 0.5) +
  facet_wrap(~lat, scales = "free_y") +
  scale_fill_brewer()
```



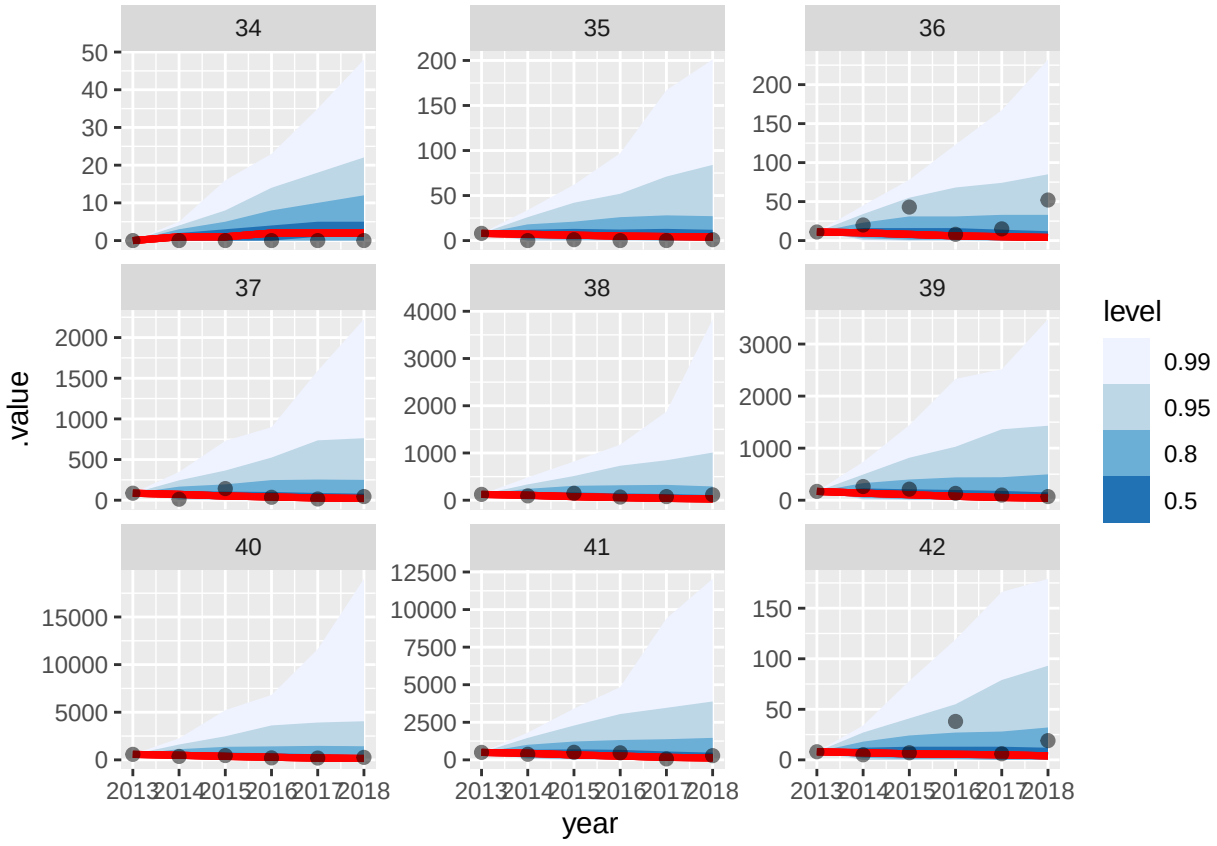
Posterior predictive fits

```
pp_predicted_fluke %>%
  ggplot() +
  stat_lineribbon(aes(x = year, y = .value),.width = c(.99, .95, .8, .5), color = "red") +
  geom_point(data = fluke_train %>% rename(lat = lat_round), aes(year, num_obs), size = 2,alpha = 0.5) +
  facet_wrap(~lat, scales = "free_y") +
  scale_fill_brewer()
```

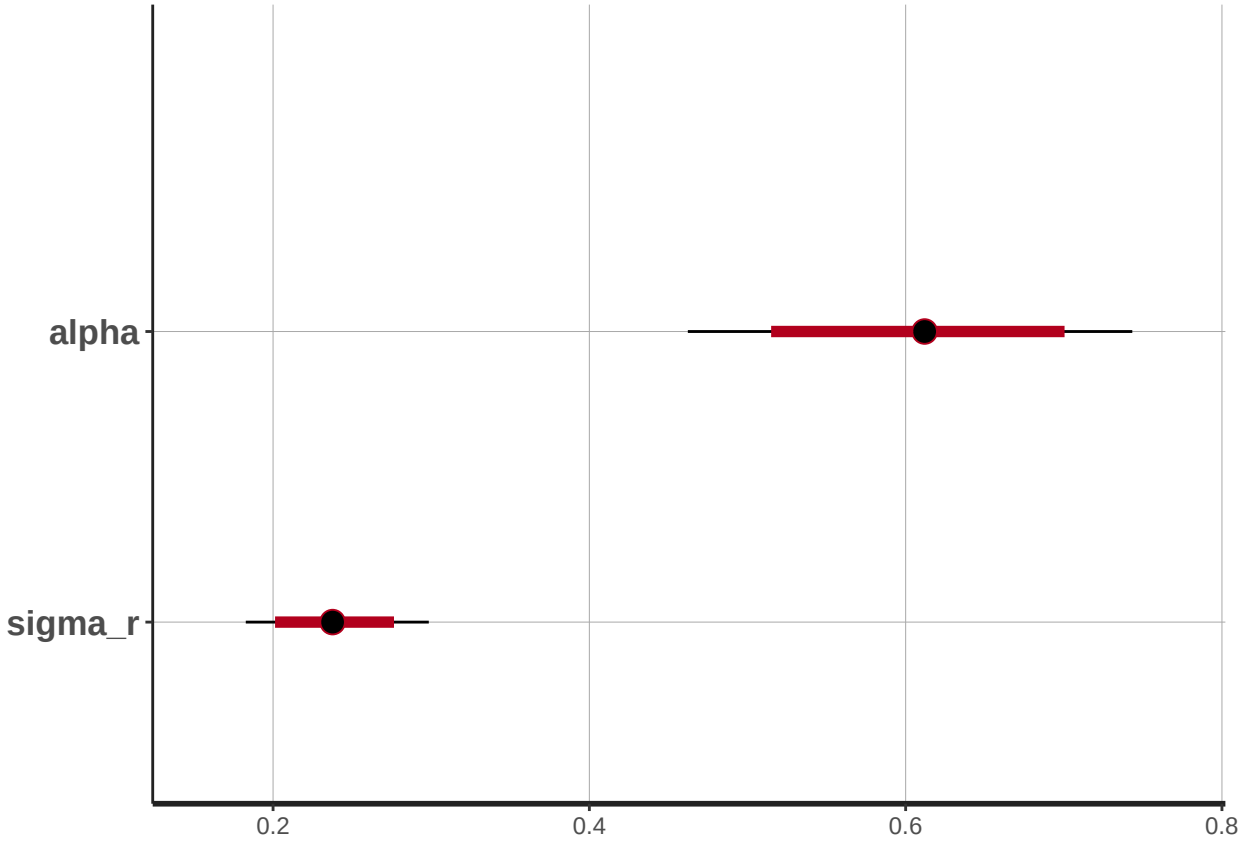


Posterior predictive projections: basically in the absence of more structure the AR1 process is useless after about 4 years, but it's a start

```
pp_projected_fluke %>%
  ggplot() +
  stat_lineribbon(aes(x = year, y = .value),.width = c(.99, .95, .8, .5), color = "red") +
  geom_point(data = fluke_test %>% rename(lat = lat_round), aes(year, num_obs), size = 2,alpha = 0.5) +
  facet_wrap(~lat, scales = "free_y") +
  scale_fill_brewer()
```




```
plot(model1_fit, pars = c("alpha","sigma_r"))
```


alpha
sigma_r
0.2
0.4
0.6
0.8