Workshop

Exercise # 3: Analysis of catch-per-unit effort data

Goal

The goal of this exercise is to gain experience fitting a Bayesian hierarchical model using the `rstanarm` package.

The dataset

For this exercise we will model night-time boat electrofishing survey data for *adult* brown trout from the West Branch Delaware River. We want to make inferences about mean catch-per-unit effort (CPE) across time and whether or not the estimated CPE exceeds a management threshold of **100 fish/hr** in any given year (note: I made up this management threshold). Because we are fitting a model using Bayesian estimation, we can ask the question: "What is the posterior probability that brown trout CPE exceeds 100 fish/hr in a given year?". This is a useful way of examining this question since we could imagine a scenario where a specific management action might be implemented if the probability of exceeding 100 fish/hr was less than, say, 0.70 in two consecutive years (again, I am making this up...but you get the point.). I'd argue that a probabilistic statement like this is more intuitive and easier to communicate to stakeholders than a P-value.

Getting started

Open **R** from the `.proj` file associated with this workshop (`R_Workshop.Rproj`) and navigate to the `07_Exercises` –> `Ex_3` folder and open the `exercise_3.R` script. This script will have necessary code and areas where you can fill in code.

Analysis steps

1. Read in the data set `Delaware_tailwaters/NBE WB 2017 to 2021 TNP ver 121622 djp 122022.csv` using the `read_csv` function. The data are located in the directory `02_Data/Delaware_tailwater`. Name the data frame "adult_dat".
2. This data set is 6504 rows and has 92 columns. Surveys were conducted across 4 sites from 2017 – 2021. Surveys also occurred during one or more months within a year at each site, so the data are fairly unbalanced. Each row in the data set is an individual fish that was caught during a survey, so we will have to do some summarizing (aggregate individuals to total catch) to calculate total catch.
3. Count total catch across site, year, month, riverside (whether the survey occurred on the left or right side of the river). We also want to retain effort for each survey and river discharge. We will want to include discharge as a predictor in our model given it's potential to affect CPE.
4. Create data summary table and plot raw data.
5. Fit a hierarchical regression model (details below).
6. Summarize posteriors, plot, make inferences.

Data manipulation

In order to get the data into a usable format we will need perform a few data type conversions, filter, and clean column headings. The **R** chunk below shows the necessary steps we will take to prepare the data. Specifically, we are:

1. Using `mutate()`, convert `SampleDate` to a date data type
2. Using `mutate()`, convert some character to factors
3. Using `filter()` Select only adult fish, wild fish, and brown trout
4. Convert all column headings to lowercase

```r
adult_dat <- adult_dat %>%
  mutate(SampleDate = mdy(SampleDate), # sample date
         Site = factor(Site), # sample site
         Wild = factor(Wild), # Wild fish designation
         Stage = factor(Stage)) %>%  # Life-stage designation
  filter(Stage != "Yearling" & Stage != "Young of year") %>% # select adult fish
  filter(Wild != "Stocked") %>% # Select wild fish
  filter(SpeciesCd == 328) %>% # select brown trout (species code == 328)
  rename_with(tolower) # make all column names lower case
```

Calculate total catch

Task

Write code to create a new data frame called `catch_dat` using the data in `adult_dat`. `catch_dat` will contain the total catch (the count of individual fish caught in `adult_dat`). To do this, you will need to group the data (using `group_by()`) by `site`, `year`, and `month`, We will also group by `riverside` (whether the survey was on the left or right side of the river) for visualizing the data in table form. Next, use the `summarize` function to create new variables called `total_catch`, `effort_hrs`, `discharge`, and `cpe` (**Be sure to name your new variables these names so downstream code works!).

- Hints

1. When using `summarize()`, we can count the number of rows (fish in our case) by using the syntax: `total_catch = n()`. The `n()` function is counting the number of rows (fish) in each site, year, and month (or whatever we choose to group by).
2. Because the effort and discharge values are repeated for every site, year, month, and riverside, we can simply take the `mean()` of these values when creating our new variables. For example, `effort_hrs = mean(timefish)` will give us the effort for a given `site`, `year`, and `month` and `riverside` that we grouped by.
3. `cpe` will be calculated as `total_catch`/`effort_hrs`. Note: we are calculating `cpe` in this manner so we can do some initial exploratory summaries and plots; however, we will model the actual catch data as the response variable in our model and *not* `cpe` (see the Statistical model description below).
4. Be sure to `ungroup()` at the end of calculations.

Table 1 shows the first 10 rows of what your data summary should look like.

Table 1: Adult summary catch by site, month, and year

Site	Year	Month	River side	Total catch	Total effort (hrs)	Discharge	CPE
AR	2017	7	right	55	0.16	435	343.75
AR	2017	8	right	35	0.13	526	269.23
AR	2017	10	right	27	0.21	419	128.57
AR	2018	5	left	13	0.10	1630	130.00
AR	2018	5	right	16	0.12	1630	133.33
AR	2018	6	left	33	0.19	525	173.68
AR	2018	6	right	51	0.17	525	300.00
AR	2018	7	left	51	0.17	423	300.00
AR	2018	7	right	54	0.19	423	284.21
AR	2018	9	left	12	0.10	1280	120.00

Plot the data

We can use the following code to plot the CPE by site, year, and month as shown in Figure 1:

```r
# Plot data
ggplot(data = catch_dat, mapping = aes(x=year, y=cpe)) +
  facet_wrap(~site) +
  geom_point(aes(color=factor(month))) +
  theme_bw() +
  theme(axis.text = element_text(size = 13),
        axis.title = element_text(size = 13),
        strip.text.x = element_text(size=13)) +
  theme(plot.margin = margin(0, .5, .5, .5, "cm")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  theme(legend.text=element_text(size=11)) +
  labs(title="", y="CPE (fish/hr)", x="Year", color="Month")
```

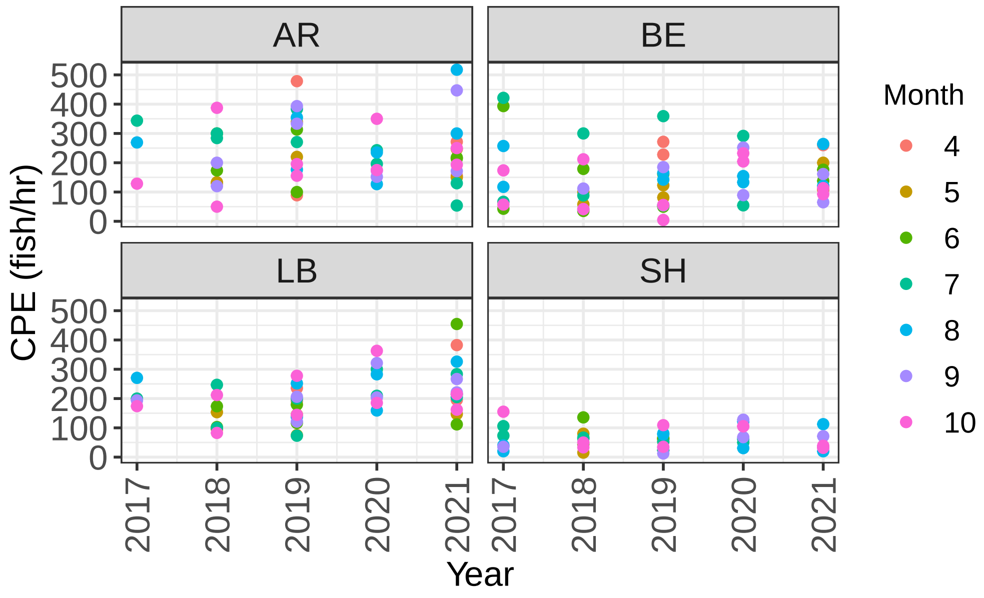

Figure 1: CPE of adult brown trout across sites, years, and months.

Estimate CPE over time

The statistical model

Two common statistical distributions used to model count data (e.g., fish catch data) are the Poisson and negative binomial distributions (NB). Without going into too many details, the Poisson distribution has an assumption (that the expected value [the mean; λ] and the variance are equal) that is rarely met with fisheries data. The NB relaxes this assumption and includes an additional parameter – an overdispersion parameter (ϕ). As such, we will model catch using a NB distribution. We will account for effort through the use of an "offset" term in the model which will effectively allow us to model catch-per-unit effort and not just the counts of total catch. The model we will fit for catch (C) in site i, month j, and year t is as follows:

$$C_{ijt} \sim NB(\lambda ijt, \phi)$$

$$log(\lambda_{ijt}) = \alpha_0 + X_{it}\beta + \delta_j + \gamma_i + log(effort_{ijt})$$

where λ is the relative abundance, α_0 is the intercept, X are our predictor variables (i.e., fixed effects of discharge and year), δ_j is a random month effect, and γ_i is a random site effect where $\delta_j \sim N(0, \sigma^2_{month})$ and $\gamma_i \sim N(0, \sigma^2_{site})$.

Fitting the model using `stan_glmer`

The two fixed effects in our model are discharge and year. We need to prepare these predictors for model fitting. First, it is almost always advisable to standardize continuous predictor variables prior to fitting a model. To standardize, we subtract the mean from each observation and divide by the standard deviation ($(x_i - mean(x))/sd(x)$). This is a simple calculation, but we can also use the `scale()` function to accomplish this. Second, we will convert year to a factor. Both can be accomplished using the following code:

```
# Standardize discharge and make year a factor
catch_dat <- catch_dat %>%
  mutate(z_discharge = as.numeric(scale(discharge)),
         year = factor(year))
```

Notice how the `scale()` function is nested within the `as.numeric()` function. This is done to ensure that the resulting scaled discharge variable (called `z_discharge`) is returned as a numeric vector.

Our final data set for model fitting looks like:

```
str(catch_dat)
```

```
tibble [191 x 9] (S3: tbl_df/tbl/data.frame)
 $ site       : Factor w/ 4 levels "AR","BE","LB",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ year       : Factor w/ 5 levels "2017","2018",..: 1 1 1 2 2 2 2 2 2 2 ...
 $ month      : num [1:191] 7 8 10 5 5 6 6 7 7 9 ...
 $ riverside  : chr [1:191] "right" "right" "right" "left" ...
 $ total_catch: int [1:191] 55 35 27 13 16 33 51 51 54 12 ...
 $ effort_hrs : num [1:191] 0.16 0.13 0.21 0.1 0.12 0.19 0.17 0.17 0.19 0.1 ...
 $ discharge  : num [1:191] 435 526 419 1630 1630 525 525 423 423 1280 ...
 $ cpe        : num [1:191] 344 269 129 130 133 ...
 $ z_discharge: num [1:191] -0.866 -0.701 -0.895 1.299 1.299 ...
```

The `stan_glmer()` syntax for fitting this model is below:

```
m1 <- stan_glmer(formula = total_catch ~ 1 + z_discharge + year +
                   (1 | month) + (1|site) , family = neg_binomial_2(link='log'),
                 offset = log(effort_hrs),
                 data = catch_dat,
                 iter = 1500, chains = 3)
```

Where `total_catch` is our response variable, the `1` indicates we are estimating in intercept term (this is also done by default), the `+` adds our fixed and random effects. The fixed effects are `z_discharge` and `year`, while our random effects are `month` and `site`. The syntax `(1|month)` and `(1|site)` are indicating that we are including a random intercept term for `month` and `site`, respectively. The `family` argument states what statistical distribution we are using, in this case a negative binomial distribution with a log link function. Because we want to model CPE and not just catch, we include the natural log of effort as an offset term in the model (`offset = log(effort_hrs)`). The `data` argument states what data set we are using, `iter` states how many MCMC iterations we are running, and `chains` indicates how many MCMC chains we will use. We are using default priors on all parameters, however, priors could be explicitly stated in additional arguments (see the `stan_glmer()` help file if you want more details). We can now run this code and fit the model – it will take a few seconds or minutes depending on your computer CPU.

Evaluate the model

Before we make any inferences on the estimated parameters, we want to look at convergence. Here we will look at traceplots of the intercept and the effect of discharge on CPE. We would want to look at traceplots for all parameters, but for the sake of space we will just look at these two.

```
# Convert the model object mq to an array for use in the function
# bayesplot::mcmc_trace
posterior <- as.array(m1)
# Create traceplot
mcmc_trace(posterior, pars = c("(Intercept)", "z_discharge"))
```

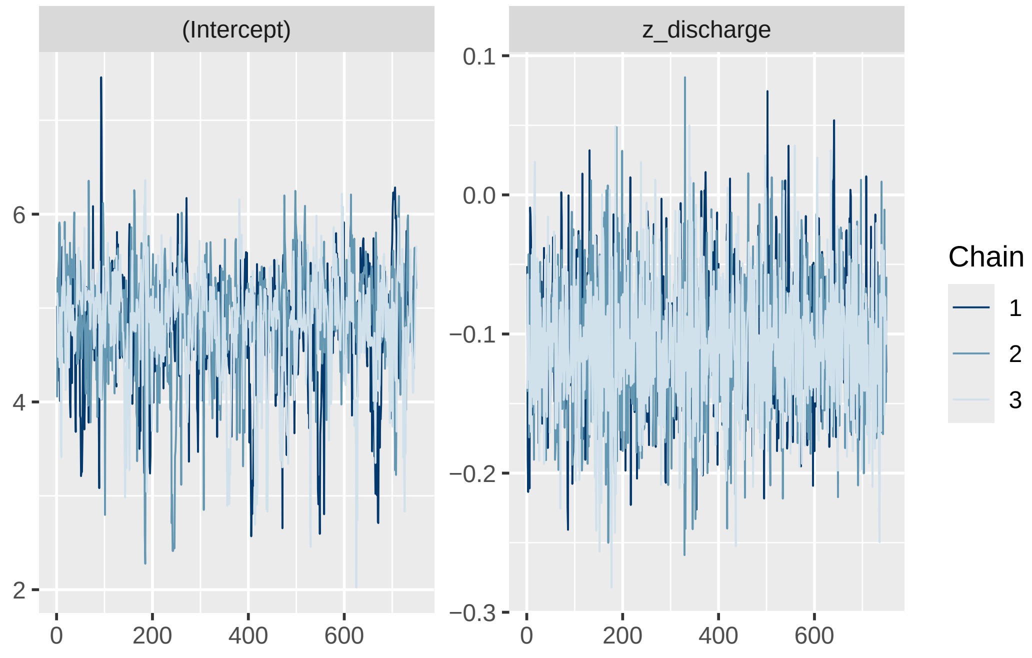

Figure 2: Traceplots for the intercept and slope parameter for discharge

Table 2: MCMC posterior summaries

Parameter	Posterior mean	SD	Lower 95% CI	Upper 95% CI	Effective sample size	Rhat
(Intercept)	4.83	0.60	3.27	5.79	386	1.01
z_discharge	-0.11	0.05	-0.20	-0.01	2619	1.00
year2018	-0.25	0.16	-0.56	0.05	1371	1.00
year2019	-0.15	0.14	-0.43	0.11	1292	1.00
year2020	0.00	0.15	-0.32	0.30	1463	1.00
year2021	0.01	0.15	-0.30	0.30	1244	1.00
b[(Intercept) month:4]	0.04	0.08	-0.07	0.26	1742	1.00
b[(Intercept) month:5]	0.00	0.07	-0.16	0.15	2584	1.00
b[(Intercept) month:6]	-0.01	0.07	-0.15	0.13	2312	1.00
b[(Intercept) month:7]	0.01	0.06	-0.10	0.16	1930	1.00
b[(Intercept) month:8]	0.00	0.06	-0.14	0.14	2452	1.00
b[(Intercept) month:9]	-0.03	0.07	-0.18	0.08	2241	1.00
b[(Intercept) month:10]	-0.02	0.06	-0.18	0.08	2074	1.00
b[(Intercept) site:AR]	0.75	0.59	-0.20	2.25	380	1.01
b[(Intercept) site:BE]	0.28	0.59	-0.67	1.79	384	1.01
b[(Intercept) site:LB]	0.53	0.59	-0.40	2.07	396	1.01
b[(Intercept) site:SH]	-0.59	0.59	-1.52	0.92	389	1.01
Sigma[month:(Intercept),(Intercept)]	0.01	0.02	0.00	0.05	1263	1.00
Sigma[site:(Intercept),(Intercept)]	1.73	3.31	0.13	10.94	477	1.01

The trace plots look pretty good. We also want to look at the Rhat statistics to see if they are close to 1 (Table 2). Rhat close to 1 is an indication of convergence. The effective sample sizes are also ok, but we might want to run this model for more iterations. We will go ahead and use this model, however.

Table 2. has posterior summaries for most of the estimated parameters. There are several things to notice here:

1. The effect of discharge (`z_discharge`) is negative and the 95% credible intervals (CIs) do not overlap zero - indicating that at higher discharges we have lower CPE. This is expected since the discharge can negatively effect sampling efficiency.
2. The `(Intercept)` is the estimated log-scale CPE for 2017, while `year2018 - year2021` are the deviations in log CPE for years 2018 – 2021 from the intercept, 2017. We will use these estimates to derive CPE in each year and make inferences about whether or not the CPE in any given year was above the management threshold of 100 fish/hr.
3. The `b[(Intercept) month:4] - b[(Intercept) month:9]` are the month random effects. April has, on average, higher CPE than other months and September has the lowest CPE, on average.
4. The `b[(Intercept) site:AR] - b[(Intercept) site:SH]` are the site random effects. AR has, on average, highest CPE; whereas, SH has the lowest.
5. There are two standard deviation estimates. These are `Sigma[month:(Intercept),(Intercept)]` and `Sigma[site:(Intercept),(Intercept)]`. These correspond to σ_{month} and σ_{site} in the statistical model equation above and quantify among-month and among-site variability, respectively.

Annual estimates of CPE in relation to a management threshold

Now that our model is fit, we want to derive annual estimates of CPE and calculate the posterior probability of annual CPE exceeding the 100 fish/hr management threshold. You can refer to the **R** code for this exercise to see the code used to derive these estimates – which are shown in Figure 3.

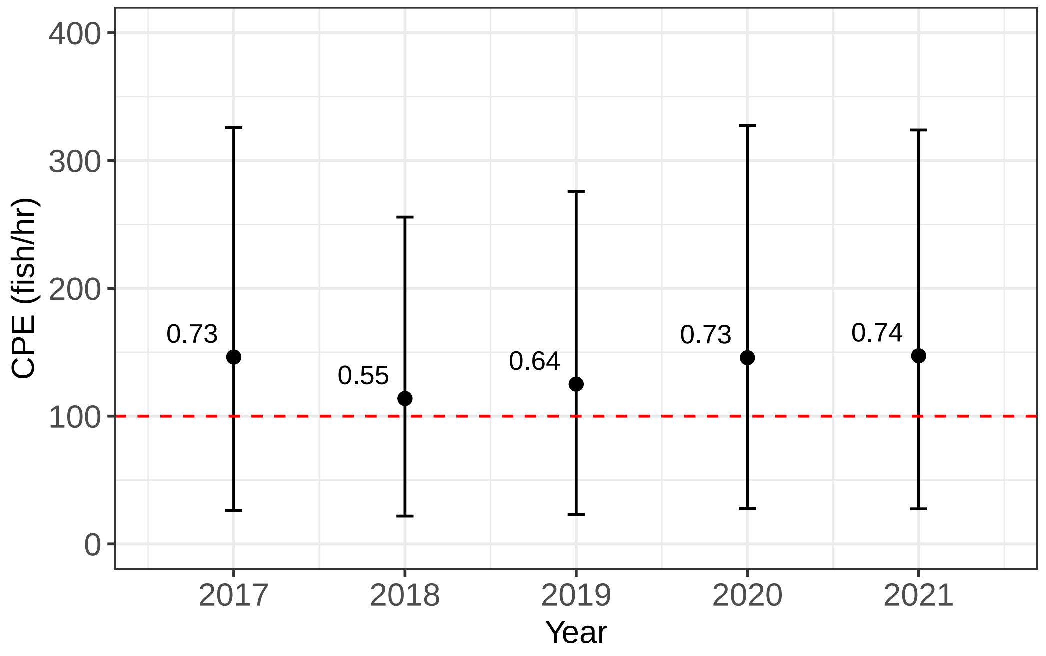

Figure 3: Estimated CPE of brown trout from the West Branch Delaware River from a hierarchical model. Points are posterior means and vertical bars are 95% credible intervals. Red dashed line is a management treshold of 100 fish/hr and values are the posterior probability of an annual CPE exceeding the management threshold.

Compare the hierarchical model to a negative binomial model without random effects

For demonstrative purposes, let's compare the hierarchical model with a negative binomial model without any random effects. The model is:

$$C_{ijt} \sim NB(\lambda ijt, \phi)$$
$$log(\lambda_{ijt}) = \alpha_0 + X_{it}\beta + log(effort_{ijt})$$

And all paramters are as described above. We can fit the model using `stan_glm` as follows:

Compare the estimated annual CPE and posterior probabilities from this model (Figure 4) to the hierarchical model estimates in Figure 3. How do they differ?

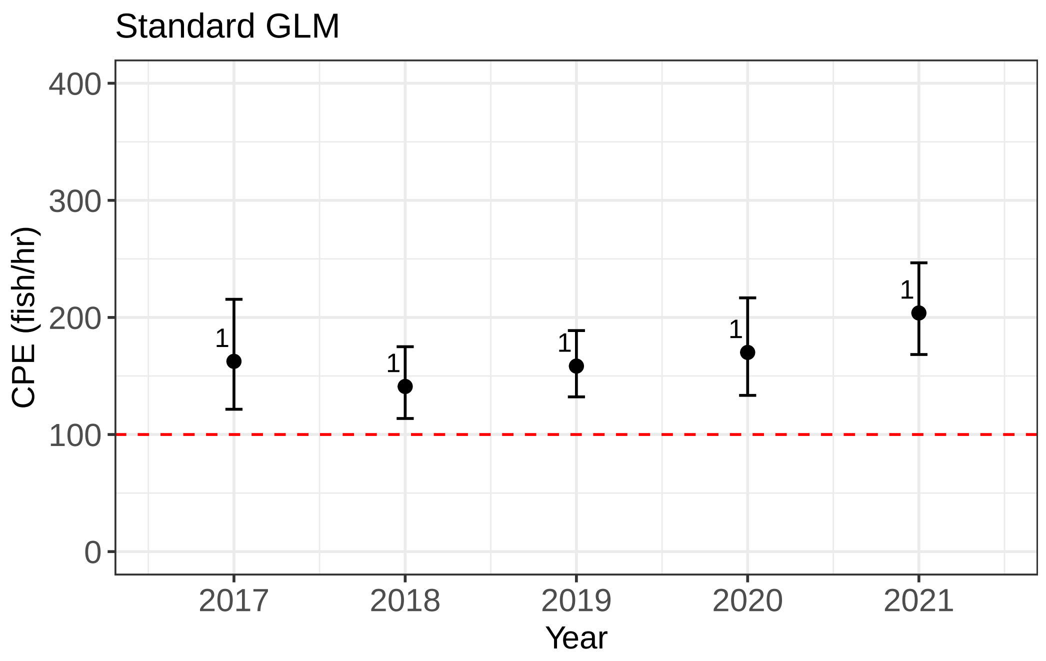

Figure 4: Estimated CPE of brown trout from the West Branch Delaware River from a non-hierarchical model. Points are posterior means and vertical bars are 95% credible intervals. Red dashed line is a management treshold of 100 fish/hr and values are the posterior probability of an annual CPE exceeding the management threshold.