Reed Frogs

Stat 341 — Spring 2017

April, 2017

Models

```
# complete pooling
m12.0 <-
  map2stan(
    alist(
      surv ~ dbinom(density, p),
      logit(p) <- a,
      a ~ dnorm(0, 5)
    ), data = Frogs, refresh = 0)
```

```
# no pooling
m12.1 <-
  map2stan(
    alist(
      surv ~ dbinom(density, p),
      logit(p) <- a_tank[tank],
      a_tank[tank] ~ dnorm(0, 5)
    ), data = Frogs, refresh = 0)
```

```
# "Varying Intercepts model" (a.k.a., "partial pooling")
m12.2 <-
  map2stan(
    alist(
      surv ~ dbinom(density, p),
      logit(p) <- a_tank[tank],
      a_tank[tank] ~ dnorm(a, sigma),
      a ~ dnorm(0, 1),
      sigma ~ dcauchy(0, 1)
    ), data = Frogs, iter = 4000, chains = 4, refresh = 0 )
```

1. How many parameters does each model have? What does each paramter "mean"?

Comparing models

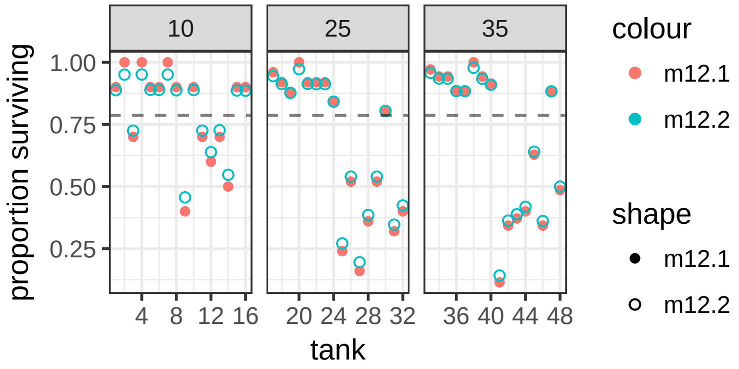
```
compare(m12.0, m12.1, m12.2)
```

```
##          WAIC pWAIC dWAIC weight    SE   dSE
## m12.2 1010.1  38.0   0.0      1 38.00    NA
## m12.1 1022.4  48.9  12.3      0 42.89  6.47
## m12.0 1372.0   1.0 361.9      0 26.01 35.12
```

2. How do you explain the effective number of parameters vs. the acutal number of paramters for each model?

A plot

The plot below shows the survival proportion in each tank (`m12.1`) and the model predicted survival proportion (`m12.2`). The horizontal line is the median survival proportion over all the tanks.



3. How do the two survival proportion compare in each tank? Why is there this pattern?
4. This phenomonon is sometimes called "shrinkage". Why does is it called that?
5. Which tanks exhibit the most shrinkage? Why do you think this is? (Note: there are at least two components to this answer.)
6. What good and/or bad features about the model does this plot reveal or suggest?

The distribution of survival across tanks

`m12.2` models a population of tanks.

7. What does the model say about the population of tanks?
8. How can we look at what the posterior distribution says about this?

A simulation to illustrate the effects of pooling

To distinguish our simulations from the real data, we refer to ponds rather than tanks.

Three levels of pooling:

- complete: All ponds are identical.
  - esimate 1 overall survival rate
- none: Each pond tells you only about itself and not about otehr ponds.
  - estimate separate survival rate for each pond
- partial: Each pond tell you something about itself and something about all ponds.
  - estimate survival rate for each pond "in context of population of ponds"
  - adaptive regularization (let degree of similarity among ponds drive amount of regularization)

Since Bayesian models are **generative**, for any choice of parameters, we should be able to generate data that matches the way the model thinks data arise.

9. For model `m12.2`, what do we need to choose to simulate data?

One way to do the simulation

```
FrogSim <-
  function(
    reps = 15L,
    n = c(5L, 10L, 25L, 35L),
    a = 1.4,
    sigma = 1.5
    ) {
    nponds <- reps * length(n)
    expand.grid(rep = 1:reps, n = n) %>%
      mutate(
        pond = 1:nponds,
        a_true = rnorm(nponds, mean = a, sd = sigma),
        s = rbinom(nponds, prob = logistic(a_true), size = n)
      )
  }
```

10. Explain what each line of `FrogSim()` is doing.

Looking at our simulation

Since we know "truth" in our simulation, we can compare the model predictions to "truth". (We don't get to do this will real data, that's the joy of simulation.)

```
SFrogs <- FrogSim()
m12.3 <- map2stan(
  alist(
    s ~ dbinom(n, p),
    logit(p) <- a_pond[pond],
    a_pond[pond] ~ dnorm(a, sigma),
    a ~ dnorm(0, 1),
    sigma ~ dcauchy(0, 1)
  ), data = SFrogs, iter = 1e4, warmup = 1000, refresh = 0 )
```

Plotting with a function

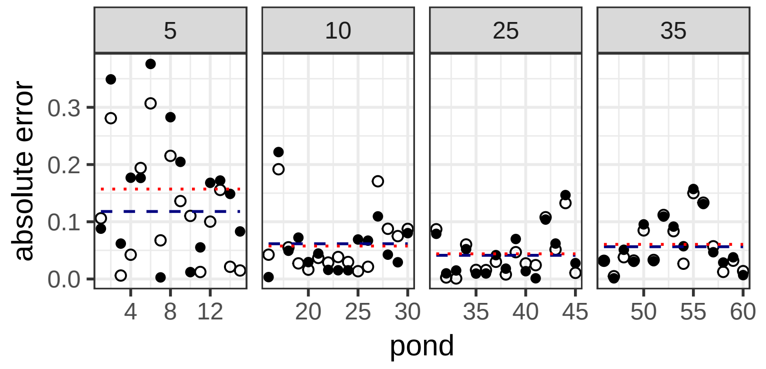
By combining this code into a function, we can apply it again later without retyping. This

- makes it clearer what is going on, and
- makes it easier to make systematic changes.

```
frog_plot <- function(model, data = model@data) {
  data <-
    data %>% as.data.frame() %>%
    mutate(
      a_pond_est = as.numeric(coef(model)[1:60]),
      p_est = logistic(a_pond_est),
      p_true = logistic(a_true),
      p_raw = s / n,
      nopool_error = abs(p_raw - p_true),
      partpool_error = abs(p_est - p_true)
    )
  gf_point(nopool_error ~ pond, data = data, shape = 16) %>%
    gf_point(partpool_error ~ pond, data = data, shape = 1) %>%
    gf_line(partpool_error ~ pond,
            data = data %>% group_by(n) %>%
              mutate(partpool_error = mean(partpool_error)),
            linetype = "dashed", color = "navy") %>%
    gf_line(nopool_error ~ pond,
            data = data %>% group_by(n) %>%
              mutate(nopool_error = mean(nopool_error)),
            linetype = "dotted", color = "red") %>%
    gf_facet_grid( ~ n, scale = "free") %>%
    gf_labs(y = "absolute error")
}
```

Generating a plot from a model is a one-liner now:

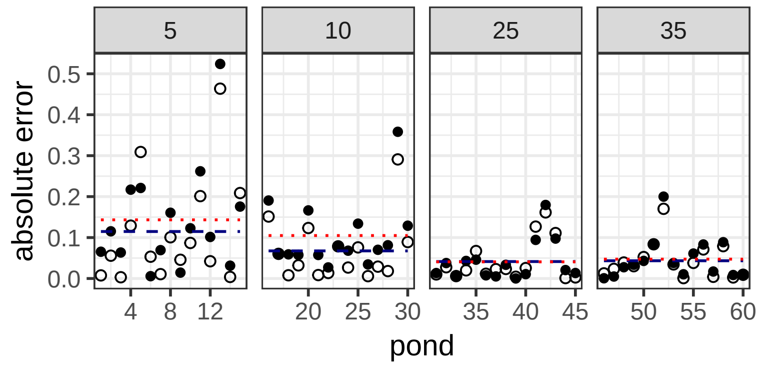
```
frog_plot(model = m12.3)
```



11. Explain what each line of `frog_plot()` is doing.
12. What does the resulting plot tell us about our three models.

A second simulation

```
NewData <- FrogSim()
m12.3new <-
  map2stan(
    m12.3,
    data = NewData,
    iter = 1e4, warmup = 1000, refresh = 0)
frog_plot(m12.3new)
```


absolute error
pond