Bayesian exposure-response modeling for binary data

Outline

- Brief introduction to Bayesian analysis
- Fitting models
- Model checking

Approach to Bayesian modeling in this course

- For this series of classes we are going to use Stan to do Bayesian modeling
  - Stan is a probabilistic programming language for fitting Bayesian models.
  - By default it uses Hamiltonian Monte Carlo (HMC), specifically a variation called the no U-turn sampler (NUTS).
  - We will go into more details about HMC/NUTS later in the course
- Start with using `brms` as our gateway to Stan
  - brms is a package that enables simple fitting of many types models that you can fit using `glm`, `survreg`, `lme`, `nlme`, etc.
  - Allows quick access to Bayesian inference
- Later we will program our own Stan models and run them with `rstan`
  - Learn more about the Stan language
  - Fit models that are not supported in `brms`

A brief review of Bayesian inference

Bayes Rule is the basis for inference about model parameters θ given data y and prior knowledge about model parameters $p(\theta)$:

$$\begin{aligned} p(\theta \mid y) &= \frac{p(\theta)p(y \mid \theta)}{p(y)} = \frac{p(\theta)p(y \mid \theta)}{\int p(\theta)p(y \mid \theta)d\theta} \\ &\propto p(\theta)\, p(y \mid \theta) \end{aligned}$$

Goals:

- Inference about θ or a function of θ
- Predictions of future observations

- The posterior summarizes what we know about θ, but typically we can't express $p(\theta \mid y)$ in closed-form.
  - We'll use Markov Chain Monte Carlo to obtain samples from $p(\theta \mid y)$.

Bayesian modeling/inference process using MCMC

1. Construct a model for the data, conditional on parameters θ, $p(y \mid \theta)$
2. Construct a prior distribution for θ, $p(\theta)$
   - Ideally based on all available evidence/knowledge (or belief)
   - Or deliberately select a non-informative (or weakly informative) prior
3. Sample from the posterior distribution for θ, $p(\theta \mid y)$.
   - Look at convergence and sampler diagnostics
   - Use for inferences regarding parameter values
4. Sample from the posterior predictive distribution for y_{new}: $p(y_{\text{new}} \mid y) = \int p(y_{new} \mid \theta)\, p(\theta \mid y)\, d\theta$.
   - Use for inferences regarding future observations
   - Sample from $p(y \mid \theta)$ for values of θ from step 3.

Bayesian ingredients for MCMC sampling from a posterior

- Data
- Model for the outcome(s) – the likelihood
- Models for the parameters – the prior distribution
- MCMC tool – Stan (via `brms` or ‘rstan“)

Ingredients for HMC/NUTS

- A starting point in the parameter space (initial value, one per chain)
- Number of MCMC samples used to tune the HMC/NUTS algorithm (`warmup`)
  - This is not exactly the same as the burn-in in other MCMC algorithms
  - NUTS uses these samples to adaptively tune the sampler
- Total number of samples to take, including the warm-up (`iter`)
- Parameters which inform how the sampler should adapt
  - Defaults are usually good for 'simple' models
  - Often need to modify for more complex, hierarchical models
  - Return to these later in the course

Let's re-fit our AE model using brms

```r
mod01_glm <- glm(formula = AE01 ~ CAVGSS + BWT + PTTYPE + SEXTXT,
                  data = aedat,
                  family = binomial(link = "logit")
                 )
```

```r
mod01_stan <- brm(formula = AE01 ~ CAVGSS + BWT + PTTYPE + SEXTXT,
                  data = aedat,
                  family = bernoulli(link = "logit"),
                  warmup = 500,
                  iter = 2000,
                  chains = 4,
                  init = "random",
                  cores = 2,
                  seed = 123)
```

What about the prior distributions?

- By default, `brms` uses flat, non-informative prior distributions for regression coefficients
- We can specify priors directly through the `prior` argument.
  - More to come in a few slides

Model summary

```
summary(mod01_stan)
```

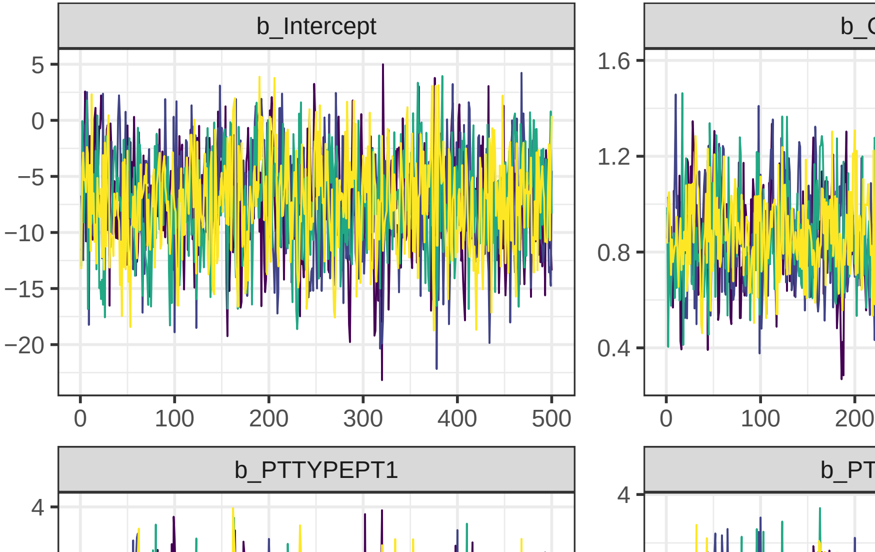
```
.  Family: bernoulli
.   Links: mu = logit
. Formula: AE01 ~ CAVGSS + BWT + PTTYPE + SEXTXT
.    Data: aedat (Number of observations: 180)
.   Draws: 4 chains, each with iter = 1000; warmup = 500; t
.          total post-warmup draws = 2000
.
. Population-Level Effects:
.            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk
. Intercept     -7.32      4.47   -16.50     1.00 1.00
. CAVGSS         0.86      0.19     0.52     1.26 1.00
. BWT            0.04      0.06    -0.08     0.16 1.01
. PTTYPEPT1      1.52      0.75     0.17     3.05 1.00
. PTTYPEPT2      0.88      0.97    -1.05     2.74 1.00
. SEXTXTMALE     0.07      0.89    -1.70     1.84 1.01
.
```

MCMC convergence diagnostics

- Traceplots
  - Plot of sampled values vs iteration
  - Look for stationarity and good mixing: fuzzy caterpillar
- $\hat{R}$
  - Heuristically: $\frac{\text{total variance of } \theta \text{ (between and within-chain)}}{\text{average within-chain variance of } \theta}$
  - Target: $\hat{R} < 1.01$ (sometimes, you'll see a rule of $\hat{R} < 1.05$)
  - Output: Summary and plot (`mcmc_plot(mod01_stan, type='rhat')`)
- Effective sample sizes
  - bulk ESS for assessing posterior means, medians, etc
  - tail ESS for assessing tail percentiles (5th, 95th)
  - Target: Depends on your goals
  - Output: Summary

Traceplots

```
bayesplot::mcmc_trace(mod01_stan)
```



Let's see the default priors in our model

```
prior_summary(mod01_stan)
```

```
.                 prior     class       coef group resp dpa
.                (flat)         b
.                (flat)         b        BWT
.                (flat)         b     CAVGSS
.                (flat)         b  PTTYPEPT1
.                (flat)         b  PTTYPEPT2
.                (flat)         b SEXTXTMALE
.  student_t(3, 0, 2.5) Intercept
.        source
.       default
.  (vectorized)
.  (vectorized)
.  (vectorized)
.  (vectorized)
.  (vectorized)
.       default
```

BRMS centers all predictors in the model

Mathematically, the RHS of the model `y ~ x1 + x2` is

$b\_Intercept + b\_x1 \cdot x1 + b\_x2 \cdot x2$

Or, equivalently,

$$Intercept + b\_x1 \cdot (x1 - \overline{x1}) + b\_x2 \cdot (x2 - \overline{x2})$$

where

$$b\_Intercept = Intercept - b_x 1 \cdot \overline{x1} - b\_x2 \cdot \overline{x2}$$

This is the parameterization that `brms` uses. (***Why do you think that is?***)

We need to specify priors for `Intercept`, `b_x1` and `b_x2`

To change them use the `set_priors` function

A Normal(mean=0, sd=5) prior on all covariate effects:

```
priors_mod01 <- set_prior('normal(0,5)', class='b')
```

A Normal(0,5) prior on CAVGSS and a DoubleExponential prior on the other effects:

```
priors_mod01_de <- set_prior('normal(0,5)', class='b', coe
  set_prior('double_exponential(0,1)', class='b')
```

See Stan functions reference for list of all available distributions.

Workbook Bayes01

- Model fitting in `brms`
- Convergence diagnostics

Model diagnostics

- We can use similar diagnostics as with likelihood based methods, but now using posterior predictive distributions
  - Quantile residual plots
  - Posterior predictive checks

Quantile residuals

1. Simulate from posterior predictive distribution
   - `brms` has a **`posterior_predict`** function to generate samples in a matrix
2. Use `createDHARMa` function in the `DHARMa` package to format output
   - Input is a matrix of posterior samples and the observed outcome data
3. Make plots as before
   - Residuals vs predicted
   - Residuals vs predictors

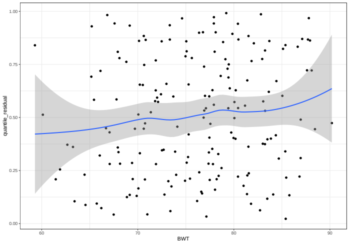
DHARMa examples

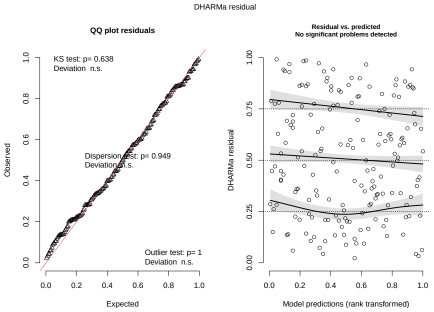
```
postpred_sample_mod01 = posterior

dharma_resids = createDHARMa(simu
                             obse
                             inte


plot(dharma_resids)
```

```
aedat %>% ungroup() %>%
  mutate(quantile_residual =
  ggplot(aes(x=BWT, y=quanti
  geom_point() +
  geom_smooth()
```





Posterior predictive checks

1. Simulate from posterior predictive distribution
   - `tidy_bayes` has an **`add_predicted_draws`** function to append the samples to a data frame
2. Compute some summary statistic on each posterior draw and on the observed data
   - Summary statistic depends on what you want to diagnose
   - For binary models, it is almost always the expected value (mean)
3. Plot distribution of summary statistics and overlay observed values
   - Type of plot depends on grouping factor and summary statistic

Simulate from posterior predictive distribution

```
aedat_pp = add_predicted_draws(newdata = aedat, mod01_stan

aedat_pp %>% ungroup() %>%
  select(USUBJID, PTTYPE, AE01, Quartile:.prediction) %>%
  slice_tail(n=4)
```

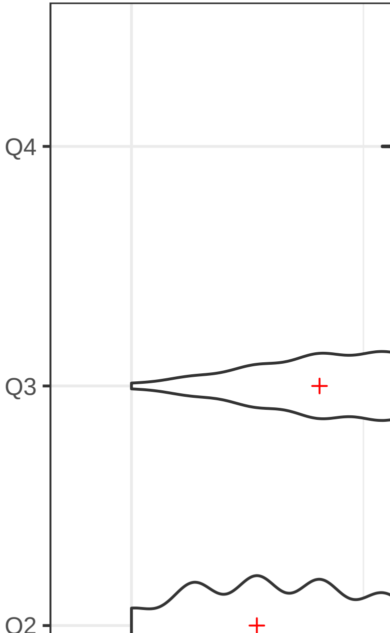
```
. # A tibble: 4 x 9
.   USUBJID PTTYPE  AE01 Quartile  .row .chain .iteration
.   <fct>   <fct>  <int> <chr>    <int>  <int>      <int>
. 1 UID-180 PT1        0 Q2         180     NA         NA
. 2 UID-180 PT1        0 Q2         180     NA         NA
. 3 UID-180 PT1        0 Q2         180     NA         NA
. 4 UID-180 PT1        0 Q2         180     NA         NA
```

Plot the VPC

```
# Observed data summary
obs_summary <- aedat %>%
  group_by(Quartile) %>%
  summarise(phat_obs = mean(AE

#Simulated data summary
sim_summary <- aedat_pp %>%
  group_by(.draw,Quartile) %>%
  summarise(phat_sim = mean(.p

# VPC
sim_summary %>%
  ggplot(aes(x=Quartile, y=pha
  geom_violin() +
  geom_point(data=obs_summary,
  labs(x='', y='Proportion wit
  coord_flip()
```



VPC for continuous variable: define summary statistic

- Fit generalized additive model (smoother) to each simulated dataset
- Predict at a fixed grid of values (0 to 95th percentile)

```
summary_function <- function(.data, .x_name, .y_name='value
  .data <- .data %>% ungroup() %>% rename('xvar' = .x_name,
  x_grid <- with(.data, seq(from = min(xvar),
                            to = quantile(xvar,probs = 0.95)
                            length = 100))
  fit <- gam(yvar ~ s(xvar), family=binomial(link='logit'),
  predictions <- predict(fit, newdata = data.frame(xvar=x_g
  return( data.frame(xvar=x_grid, prediction = predictions)
}
```

Compute summary statistics for observed data

```r
obs_summary <- summary_function(aedat, .x_name = 'CAVGSS',
  mutate(type='Observed')

head(obs_summary)
```

```
.          xvar prediction     type
. 1 0.00000000 0.03526032 Observed
. 2 0.03718346 0.03644987 Observed
. 3 0.07436693 0.03767799 Observed
. 4 0.11155039 0.03894582 Observed
. 5 0.14873385 0.04025454 Observed
. 6 0.18591731 0.04160537 Observed
```

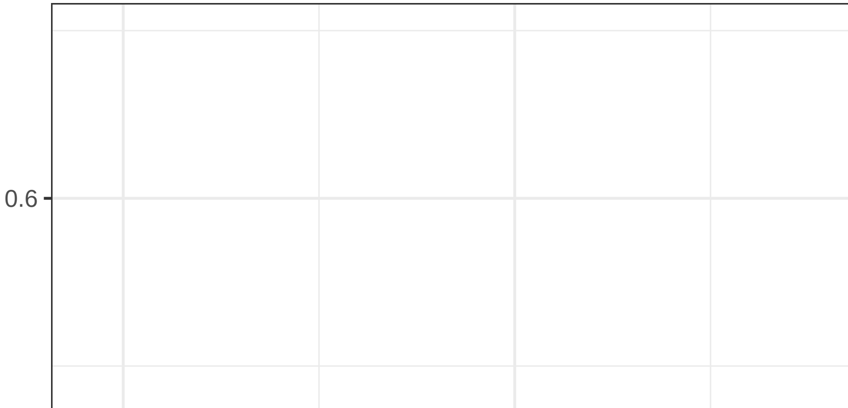
Compute summary on each simulated study

```
sim_summary <- aedat_pp %>%
  # Nest everying except the simulation name
  nest(cols=-.draw) %>%
  # Use 200 sims for demonstration
  sample_n(size=200) %>%
  # Compute summary stats for each simulated dataset
  mutate(predictions = map(cols, ~summary_function(.x,
                                                   .x_name=
                                                   .y_name=

  select(.draw,predictions) %>%
  unnest(cols=predictions) %>%
  # Summarise across simulated data sets
  group_by(xvar) %>%
  summarise(qlo = quantile(prediction, probs = 0.05),
            qhi = quantile(prediction, probs = 0.95),
            prediction=median(prediction)
            ) %>%
  mutate(type = 'Simulated')
```

Plot VPC

```
sim_summary %>% bind_rows(obs_summary) %>%
  ggplot(aes(x=xvar, y=prediction)) +
  geom_line(aes(col=type, group=type)) +
  geom_ribbon(aes(ymin=qlo, ymax=qhi, fill=type), alpha=0.2
  labs(x='Steady-state Cavg', y='Probability of severe AE')
```



Model Comparison

- Goal: maximize expected log predictive density (ELPD) for future data
  - This is a measure of out-of-sample prediction quality
- Leave-one-out cross-validation (LOO-CV) to approximate ELPD
  - Involves fitting N models
  - Can be approximated using pareto smoothed importance sampling of the posterior samples
  - `loo` package
- WAIC also approximates -2 ELPD
  - Proven to be asymptotocally equivalent to LOO-CV (modulo the -2)
  - Lower is better
  - LOO-CV generally preferable due to its ability to tell when the estimates are not trustworthy

Workbook Bayes02: Model evaluation and comparison

References