APSTA-GE 2123 Assignment 2 Answer Key

Due by 1:45 PM on May 4, 2022

1 On data science

Read this article. What is the relationship between data science and Bayesian analysis? What aspects of Bayesian analysis are or are not data science? What parts of data science are outside of Bayesian analysis? How does Bayesian analysis as it is commonly understood in the data science realm differ from the approach to Bayesian analysis that we have put forward in this course?

If data science were defined as the intersection of three fields, as in this one by Drew Conway on the next page, then Bayesian analysis would be the epitome of data science if we associate

- "Math and statistics knowledge" with "using probability"
- "Substantive expertise" with the "ability to specify a generative model, including priors"
- "Hacking skills" with "capability with MCMC"

However, many self-identified data scientists have not studied probability formally or in enough detail to conduct a Bayesian analysis. Nor are they trained in MCMC algorithms, such as Stan. So, whatever substantive expertise they might have in an area is not manifesting itself in the choice of prior distributions over the parameters. In practice, data science does not seem to be well described by a *symmetric* Venn diagram like that above but rather one the prioritizes hacking skills over math and statistics knowledge, which in turn is prioritized over substantive expertise.

Self-identified data scientists often claim that a substantial part of their job is collecting data (often from websites), cleaning it efficiently, and putting it into a (possibly tidy) format for analysis. Self-identified data scientists also tend to expend a great deal of effort creating and maintaining an analysis "pipeline", which automates as much of the process as possible as new data becomes available. Neither of these are part of the definition of Bayesian analysis, but nor are they inconsistent with Bayesian analysis.

However, if data science were defined as the intersection of three fields in the Venn diagram above, then one would be hard-pressed to explain why Bayesian analysis (especially using MCMC) is so rare among self-identified data scientists. If a Bayesian approach is used within data science, it tends to either be as a classification method that utilizes point priors or, rather than using MCMC, approximates a posterior distribution using some tractable parametric family.

Data science is often associated with "big data" and priority is given to quantitative methods that scale favorably as the number of observations becomes large, not for any Frequentist consideration but for practicality. Although the log-kernel can be often be evaluated quickly in compiled code like C++, Stan has to evaluate the log-kernel millions of times in order to obtain a few thousand draws from the posterior distribution. Thus, many self-identified data scientists see MCMC as a non-starter and see Bayesian analysis as unnecessary if the large number of observations would make the priors irrelevant and the posterior distribution concentrated near its mode.

Hierarchical models are the hallmark of Bayesian analysis and have essentially no counterpart in data science. Self-identified data scientists typically follow some version of what Hadley Wickham calls the "split-apply-combine" paradigm of quantitative analysis, where a (training) dataset is split according to the levels of some (perhaps arbitrary) factor(s), a function is applied to the observations in that stratum (often in parallel), and then the point estimates or predictions are combined. For example, when using Census data, the data scientist might split by state, apply apply an estimator, and combine the state-level estimates. From a Bayesian

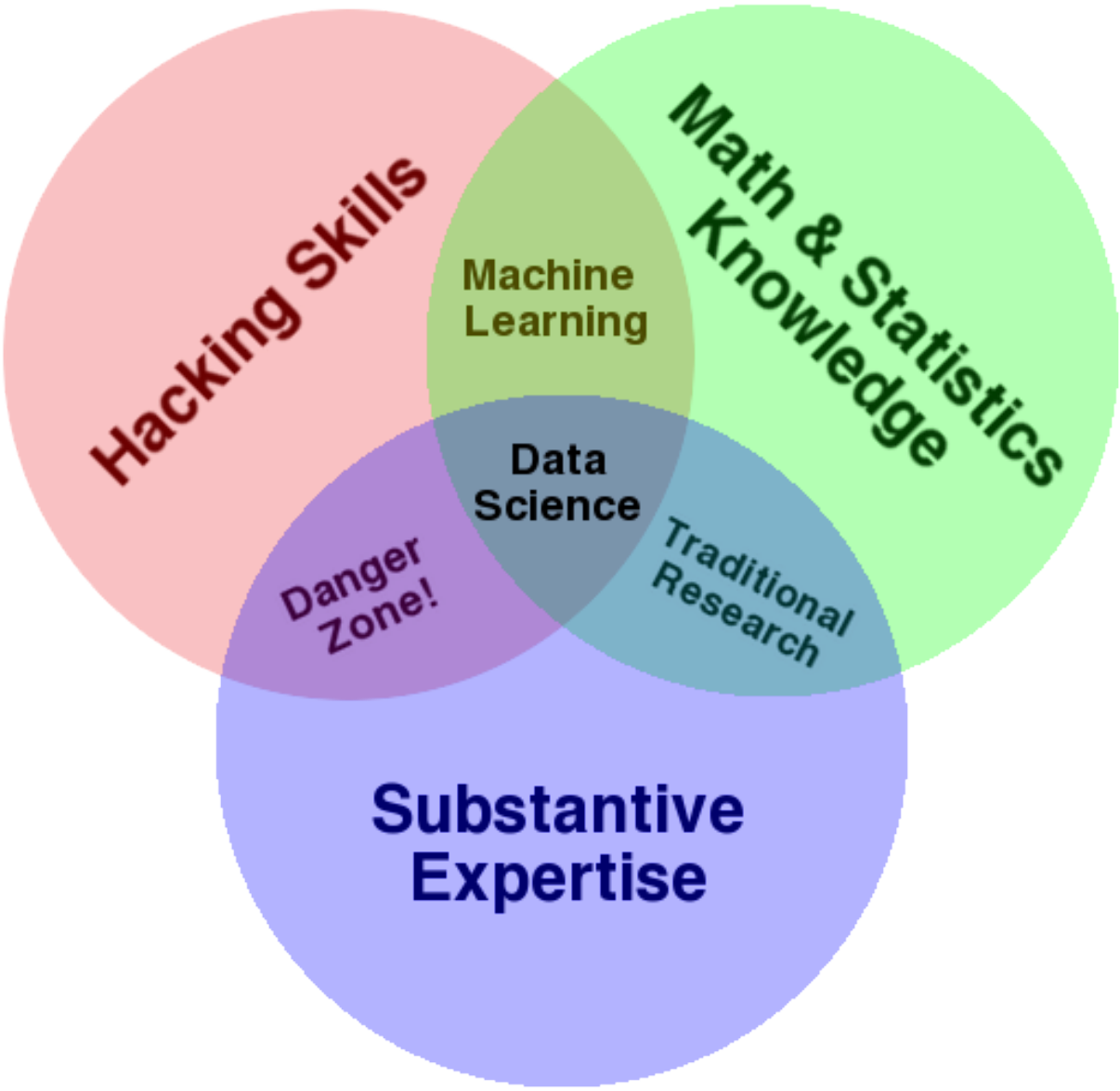


Figure 1: Venn Diagram

perspective, this corresponds to some "no pooling" model, which typically is worse than a "partial pooling" model that allows the data-generating process to vary by state to an unknown degree that is estimated along with all of the other unknowns. In essence, the posterior distribution of the parameters in one state is influenced by the observations in other states and the amount of influence is determined by the data (and the priors).

2 YouTube Views

```
youtube <- readr::read_csv("https://osf.io/25sz9/download")
```

2.1 Stan Program

```
#include quantile_functions.stan
data {
  int<lower = 0> N;     // number of observations
  vector[N] offset;     // a predictor with a coefficient of 1
  int<lower = 0> K;     // number of other predictors
  matrix[N, K] X;       // matrix of other predictors
  int<lower = 0> y[N];  // outcomes
  int<lower = 0, upper = 1> prior_only;   // ignore data?
  vector[K + 2] m;                        // prior medians
  vector<lower = 0>[K + 2] r;             // prior IQRs
  vector<lower = -1, upper = 1>[K + 2] a; // prior asymmetry
  vector<lower =  0, upper = 1>[K + 2] s; // prior steepness
}
transformed data {
  int poisson_max = 1073741824;
}
parameters {
  vector<lower = 0, upper = 1>[K + 2] p;  // CDF values
}
transformed parameters {
  real alpha = gld_qf(p[K + 1], m[K + 1], r[K + 1], a[K + 1], s[K + 1]);
  real phi   = gld_qf(p[K + 2], m[K + 2], r[K + 2], a[K + 2], s[K + 2]);
  vector[K] beta; // as yet undefined
  for (k in 1:K) beta[k] = gld_qf(p[k], m[k], r[k], a[k], s[k]); // now defined
}
model {
  if (!prior_only)
    target += neg_binomial_2_log_glm_lpmf(y | X, alpha + offset, beta, phi);
} // implicit: p ~ uniform(0, 1)
generated quantities {
  vector[N] log_lik;
  int y_rep[N];
  {
    vector[N] eta = alpha + offset + X * beta;
    for (n in 1:N) {
      real inv_mu = exp(-eta[n]);
      real lambda = gamma_rng(phi, phi * inv_mu);
      if (lambda > poisson_max) {
        y_rep[n] = 2 * (poisson_max - 1);
      } else {
        y_rep[n] = poisson_rng(lambda);
```

```
      }
      log_lik[n] = neg_binomial_2_log_lpmf(y[n] | eta[n], phi);
    }
  }
}
```

Here we had to take care in the `generated quantities` block to avoid integer overflow when drawing from the prior predictive distribution. Unless your priors are chosen *very* carefully, it is all-too-easy to put some prior probability on counts that are greater than what typical hardware can represent.

2.2 Prior Predictive Distribution

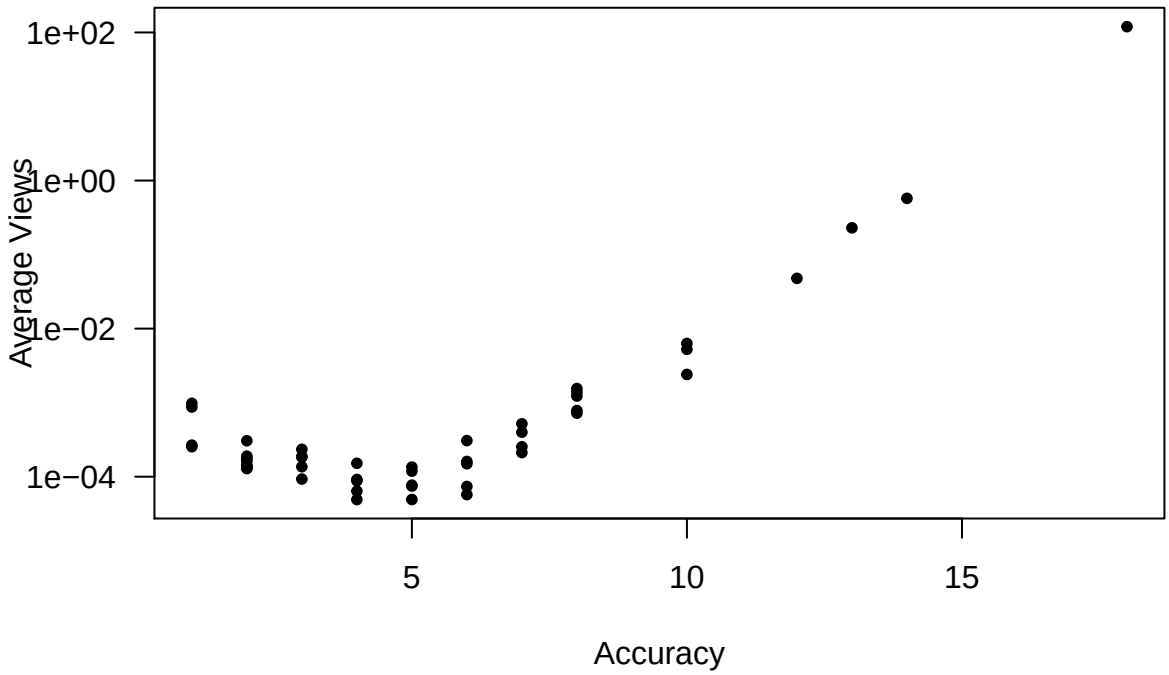
```
source(file.path("..", "..", "Week2", "GLD_helpers.R"))
```

```
a_s_alpha <- GLD_solver_LBFGS(lower_quartile = 0, median = 5, upper_quartile = 10,
                              other_quantile = 14, alpha = 0.95)
a_s_beta  <- GLD_solver(lower_quartile = -0.5, median = 0, upper_quartile = 0.5,
                        other_quantile = 1, alpha = 0.95)
a_s_phi <- GLD_solver_LBFGS(lower_quartile = 0.5, median = 2, upper_quartile = 5,
                            other_quantile = 0, alpha = 0)
```

```
library(rstan)
options(mc.cores = parallel::detectCores())
stan_data <- with(youtube, list(N = nrow(youtube), offset = log(age2), K = 1,
                                X = as.matrix(scol - mean(scol)), y = views,
                                prior_only = TRUE, m = c(0, 5, 2), r = c(1, 10, 1.5),
                                a = c(a_s_beta[1], a_s_alpha[1], a_s_phi[1]),
                                s = c(a_s_beta[2], a_s_alpha[2], a_s_phi[2])))
```

```
prior <- stan("negative_binomial.stan", data = stan_data,
              seed = 20220504, save_warmup = FALSE, init_r = 0.5)
```

```
y_rep <- rstan::extract(prior, pars = "y_rep")[[1]]
plot(youtube$scol, colMeans(y_rep) / youtube$age2, pch = 20, log = "y",
     xlab = "Accuracy", ylab = "Average Views", las = 1)
```



It seems odd that the predictive distribution would be J-shaped, so we might want to rethink the priors.

2.3 Expected Log Predictive Density under the Prior

```
loo(prior)
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
##
## Computed from 4000 by 50 log-likelihood matrix
##
##            Estimate        SE
## elpd_loo -7984101.9 1333985.3
## p_loo     7983437.4 1333977.3
## looic    15968203.9 2667970.6
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)      0      0.0%  <NA>
##  (0.5, 0.7]   (ok)        0      0.0%  <NA>
##    (0.7, 1]   (bad)       0      0.0%  <NA>
##    (1, Inf)   (very bad) 50    100.0%  1
## See help('pareto-k-diagnostic') for details.
```

This is not estimated well enough to be meaningful because all of the Pareto k estimates are too high.

2.4 Posterior Distribution

```
stan_data$prior_only <- FALSE
post <- stan("negative_binomial.stan", data = stan_data,
             seed = 20220504, save_warmup = FALSE, init_r = 0.5)
```

```
print(post, pars = c("alpha", "beta", "phi"))
```

```
## Inference for Stan model: negative_binomial.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## alpha    3.23       0 0.13  3.00  3.14  3.23  3.31  3.50  2949    1
## beta[1] -0.13       0 0.04 -0.19 -0.15 -0.13 -0.10 -0.05  3152    1
## phi      1.34       0 0.06  1.27  1.29  1.32  1.36  1.50  3569    1
##
## Samples were drawn using NUTS(diag_e) at Wed May  4 01:02:13 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

The negative binomial model is clearly preferable to the Poisson because the posterior distribution of ϕ is concentrated on small numbers rather than large ones. The fact that almost all of the posterior draws of β are negative

```
mean(as.data.frame(post)$beta < 0)
```

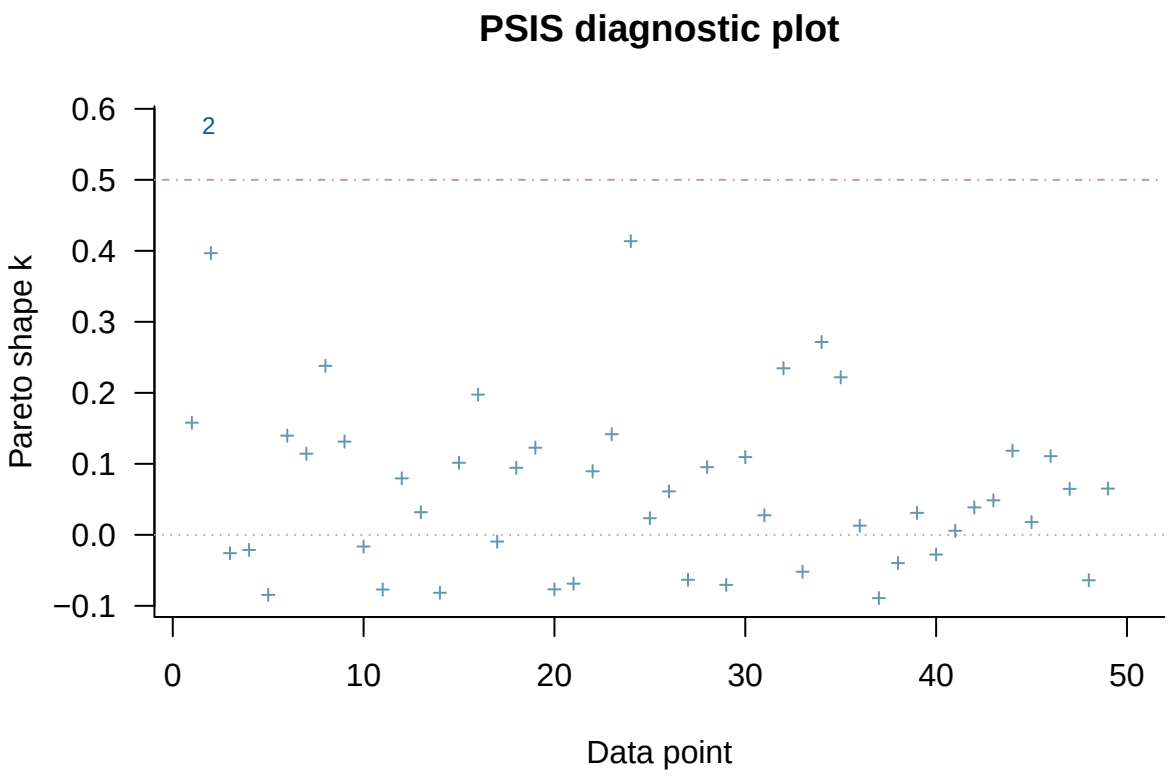
```
## [1] 0.99975
```

implies that more accurate videos are less watched.

2.5 Expected Log Predictive Density under the Posterior

```
plot(loo(post), label_points = TRUE)
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```



```r
loo(post)
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
##
## Computed from 4000 by 50 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -603.8  9.3
## p_loo         3.4  1.1
## looic      1207.7 18.6
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     49    98.0%   1329
##  (0.5, 0.7]   (ok)        1     2.0%   339
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

All the Pareto k estimates are now fine, although the second observation is borderline, and the estimated ELPD is much higher than the (useless) estimate of it under the prior.

3 Vaccination Rates

Revisit the data we used in Week3 on covid vaccinations at the county-level. You can load these data via

```r
Gabba <- readr::read_csv(file.path("..", "..", "Week3", "Gabba.csv"),
                         col_types = c("ccccddddddddddd"), skip = 1, col_names =
                           c("FIPS", "ST", "State", "County", "Trump#", "Votes#", "Trump", "Pop",
                             "Vaccinated#", "Vaccinated", "Death1", "Death2", "Death3", "Death4"))
Gabba <- dplyr::filter(Gabba, Vaccinated < 100) # some data points were messed up
Gabba$ST <- as.factor(Gabba$ST)
```

In Week3, we modeled the vaccinated percentage using a normal distribution. In this problem, we are going to utilize hierarchical GLMs.

3.1 Beta Likelihood

```
data {              // saved as "beta.stan"
  int<lower = 0> N; // number of observations
  int<lower = 0> K; // number of predictors
  matrix[N, K] X;   // matrix of predictors
  vector<lower = 0, upper = 1>[N] y;      // outcomes
  int<lower = 1> J;                       // number of groups
  int<lower = 1, upper = J> group[N];     // group membership
  int<lower = 0, upper = 1> prior_only;   // ignore data?
  vector[K + 1] m;                        // prior means
  vector<lower = 0>[K + 1] scale;         // prior scales
  vector<lower = 0>[2] rate;              // prior rates
}
```

```
parameters {
  vector[K] beta;
  real mu_alpha;
  real<lower = 0> sigma;
  vector[J] alpha;
  real<lower = 0> kappa;
}
model {
  if (!prior_only) {
    vector[N] mu = inv_logit(alpha[group] + X * beta);
    target += beta_proportion_lpdf(y | mu, kappa);
  }
  target += normal_lpdf(beta  | m[1:K],   scale[1:K]);
  target += normal_lpdf(mu_alpha | m[K + 1], scale[K + 1]);
  target += normal_lpdf(alpha | mu_alpha, sigma);
  target += exponential_lpdf(sigma | rate[1]);
  target += exponential_lpdf(kappa | rate[2]);
}
generated quantities {
  vector[N] y_rep;
  {
    vector[N] mu = inv_logit(alpha[group] + X * beta);
    for (n in 1:N) y_rep[n] = beta_proportion_rng(mu[n], kappa);
  }
}
```

3.2 Beta Posterior

```
stan_data <- with(Gabba, list(N = nrow(Gabba), K = 1,
                              X = as.matrix(Trump - mean(Trump)) / 100,
                              y = Vaccinated / 100, J = nlevels(ST),
                              group = as.integer(ST), prior_only = FALSE,
                              m = c(0, 0), scale = c(0.25, 0.5), rate = c(0.5, 1)))
```

```
post_beta <- stan("beta.stan", data = stan_data, seed = 20220504, save_warmup = FALSE)
```

```
## Trying to compile a simple C file
```

```
print(post_beta, pars = c("alpha", "log_lik", "y_rep"), include = FALSE)
```

```
## Inference for Stan model: beta.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## beta[1]    -1.91    0.00 0.05   -2.00   -1.94   -1.90   -1.87   -1.81  3571
## mu_alpha    0.11    0.00 0.03    0.06    0.09    0.11    0.13    0.17  4351
## sigma       0.19    0.00 0.02    0.15    0.17    0.19    0.20    0.24  3208
## kappa      34.17    0.01 0.85   32.51   33.61   34.18   34.74   35.84  5311
## lp__     3342.70    0.13 5.32 3330.89 3339.43 3343.12 3346.37 3351.93  1783
##          Rhat
## beta[1]     1
## mu_alpha    1
## sigma       1
## kappa       1
```

```
## lp__        1
##
## Samples were drawn using NUTS(diag_e) at Wed May  4 03:35:39 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Although we could plot the non-linear relationship, the fact that the posterior distribution of the coefficient is so concentrated on negative values implies that counties with more support for Trump in 2020 tend to have much lower vaccination rates in 2022.

3.3 Binomial Likelihood

```
data {              // saved as "binomial.stan"
  int<lower = 0> N; // number of observations
  int<lower = 0> K; // number of predictors
  matrix[N, K] X;   // matrix of predictors
  int<lower = 0> y[N];                    // outcomes
  int<lower = 0> pop[N];                  // population
  int<lower = 1> J;                       // number of groups
  int<lower = 1, upper = J> group[N];     // group membership
  int<lower = 0, upper = 1> prior_only;   // ignore data?
  vector[K + 1] m;                        // prior means
  vector<lower = 0>[K + 1] scale;         // prior scales
  real<lower = 0> rate;                   // prior rates
}
parameters {
  vector[K] beta;
  real mu_alpha;
  real<lower = 0> sigma;
  vector[J] alpha;
}
model {
  if (!prior_only) target += binomial_logit_lpmf(y | pop, alpha[group] + X * beta);
  target += normal_lpdf(beta  | m[1:K],   scale[1:K]);
  target += normal_lpdf(mu_alpha | m[K + 1], scale[K + 1]);
  target += normal_lpdf(alpha | mu_alpha, sigma);
  target += exponential_lpdf(sigma | rate);
}
generated quantities {
  vector[N] y_rep;
  {
    vector[N] mu = inv_logit(alpha[group] + X * beta);
    for (n in 1:N) {
      y_rep[n] = binomial_rng(pop[n], mu[n]);
      y_rep[n] /= pop[n];
    }
  }
}
```

3.4 Binomial Posterior

```
stan_data <- with(Gabba, list(N = nrow(Gabba), K = 1,
                              X = as.matrix(Trump - mean(Trump)) / 100,
```

```
                                y = `Vaccinated#`, pop = Pop, J = nlevels(ST),
                                group = as.integer(ST), prior_only = FALSE,
                                m = c(0, 0), scale = c(0.25, 0.5), rate = 0.5))
```

```
post_binomial <- stan("binomial.stan", data = stan_data, seed = 20220504, save_warmup = FALSE)
```

```
## Trying to compile a simple C file
```

```
y_rep_beta <- rstan::extract(post_beta, "y_rep")[[1]]
y_rep_binomial <- rstan::extract(post_binomial, "y_rep")[[1]]

beta_low  <- apply(y_rep_beta, MARGIN = 2, FUN = quantile, probs = 1 / 3)
beta_high <- apply(y_rep_beta, MARGIN = 2, FUN = quantile, probs = 2 / 3)

binomial_low  <- apply(y_rep_binomial, MARGIN = 2, FUN = quantile, probs = 1 / 3)
binomial_high <- apply(y_rep_binomial, MARGIN = 2, FUN = quantile, probs = 2 / 3)

y <- stan_data$y / stan_data$pop

rbind(beta = c(too_low = mean(y < beta_low),
               just_right = mean(y > beta_low & y < beta_high),
               too_high = mean(y > beta_high)),
      binomial = c(mean(y < binomial_low),
                   mean(y > binomial_low & y < binomial_high),
                   mean(y > binomial_high)))
```

```
##          too_low just_right too_high
## beta     0.27392    0.45696  0.26912
## binomial 0.54976    0.01632  0.43264
```

The model with the Beta likelihood predicts the proportion of people vaccinated in a county much better, although it overfits somewhat by including too many counties in the middle third of the predictive distribution. If we were interested in the gross number of vaccinated people, the binomial model might well be more useful.