Math 459 Lecture 21

Todd Kuffner

rstan

`Stan` is software which uses the No-U-Turn sampler (NUTS), which is a variant of Hamiltonian Monte Carlo. `rstan` provides an interface to `Stan`.

Hamiltonian Monte Carlo is an approach to MCMC designed to avoid the problem of correlated samples due to the random walk nature of many MCMC algorithms.

- uses first-order gradient information to help choose which steps to take
- requires a lot of tuning, but properly-tuned is much more efficient even than Gibbs

NUTS tries to make it easier for users, less tuning.

rstanarm

`rStan` requires first compiling a model (see chunk of code in last lecture). `rstanarm` uses some pre-compiled models to illustrate `rstan` in applied regression modeling.

- Bayesian estimation via MCMC
- GLMs with binary, binomial, continuous, ordinal and count responses

Bayesian analysis steps

1. Specify a joint distribution for the outcome(s) and all the unknowns, which typically takes the form of a marginal prior distribution for the unknowns multiplied by a likelihood for the outcome(s) conditional on the unknowns. This joint distribution is proportional to a posterior distribution of the unknowns conditional on the observed data
2. Draw from posterior distribution using Markov Chain Monte Carlo (MCMC).
3. Evaluate how well the model fits the data and possibly revise the model.
4. Draw from the posterior predictive distribution of the outcome(s) given interesting values of the predictors in order to visualize how a manipulation of a predictor affects (a function of) the outcome(s).

First we focus on steps 1 and 2 when the likelihood is the product of conditionally independent binomial distributions.

Binomial GLM Likelihood

The likelihood for one observation y can be written as the conditionally binomial mass function

$$\binom{n}{y}\pi^y(1-\pi)^{n-y}$$

with n the number of trials.

- $\pi = g^{-1}(\eta)$ is probability of success
- $\eta = \alpha + x^T\beta$ is linear predictor
- for sample of size N, likelihood of sample is product of N individual likelihoods (for each observation)

Since π is a probability, the linnk function g maps from $(0,1) \mapsto \mathbb{R}$ (so $g^{-1} : \mathbb{R} \mapsto (0,1)$). Two common link functions for binomial GLM:

logit the logit (log-odds) link function is $g(x) = \log(x/1-x)$, so the likelihood for a single observation is

$$\binom{n}{y}(\text{logit}^{-1}(\eta))^y(1-\text{logit}^{-1}(\eta))^{n-y} = \binom{n}{y}\left(\frac{e^\eta}{1+e^\eta}\right)^y\left(\frac{1}{1+e^\eta}\right)^{n-}$$

probit the probit link function $g(x) = \Phi^{-1}(x)$ gives the likelihood

$$\binom{n}{y}(\Phi(\eta))^y(1-\Phi(\eta))^{n-y}$$

with Φ the CDF of the standard normal.

- by default `rstanarm` scales the probit so the slope at 0 is the same as the logit; then the two link functions typically give similar results
- the logit link tends to be faster and more numerically stable, so recommended to use this unless probit has strong theoretical advantage

Priors and posterior

Unfortunately, slapping a normal prior on β doesn't help analytically since it's not conjugate to the logistic model. But since we are using MCMC, we don't really care.

- example, suppose there are K predictors and we think they are just as likely to be positive as negative, but unlikely to be far from zero
- then use normal, mean-zero priors with small standard deviation
- specify this using `prior_intercept=normal(0,1)` and `prior=normal(0,1)`

With independent priors, the joint posterior is simply

$$f(\alpha, \beta | y, X) \propto f(\alpha) \times \prod_{k=1}^{K} f(\beta_k) \times \prod_{i=1}^{N} g^{-1}(\eta_i)^{y_i} (1 - g^{-1}(\eta_i))^{n_i - y_i}$$

this n_i means each response could be thought of as a binomial experiment with varying numbers of trials n_i.

Example (Gelman & Hill (2007))

Suvery of 3200 residents in small region of Bangladesh suffering from arsenic contamination of the ground water.

- respondents with elevated arsenic levels in their wells were encouraged to switch to safer public or private wells
- the survey was conducted several years later to see which of the affected residents had switched wells
- Goal: learn about what factors are associated with switching wells

Starting point

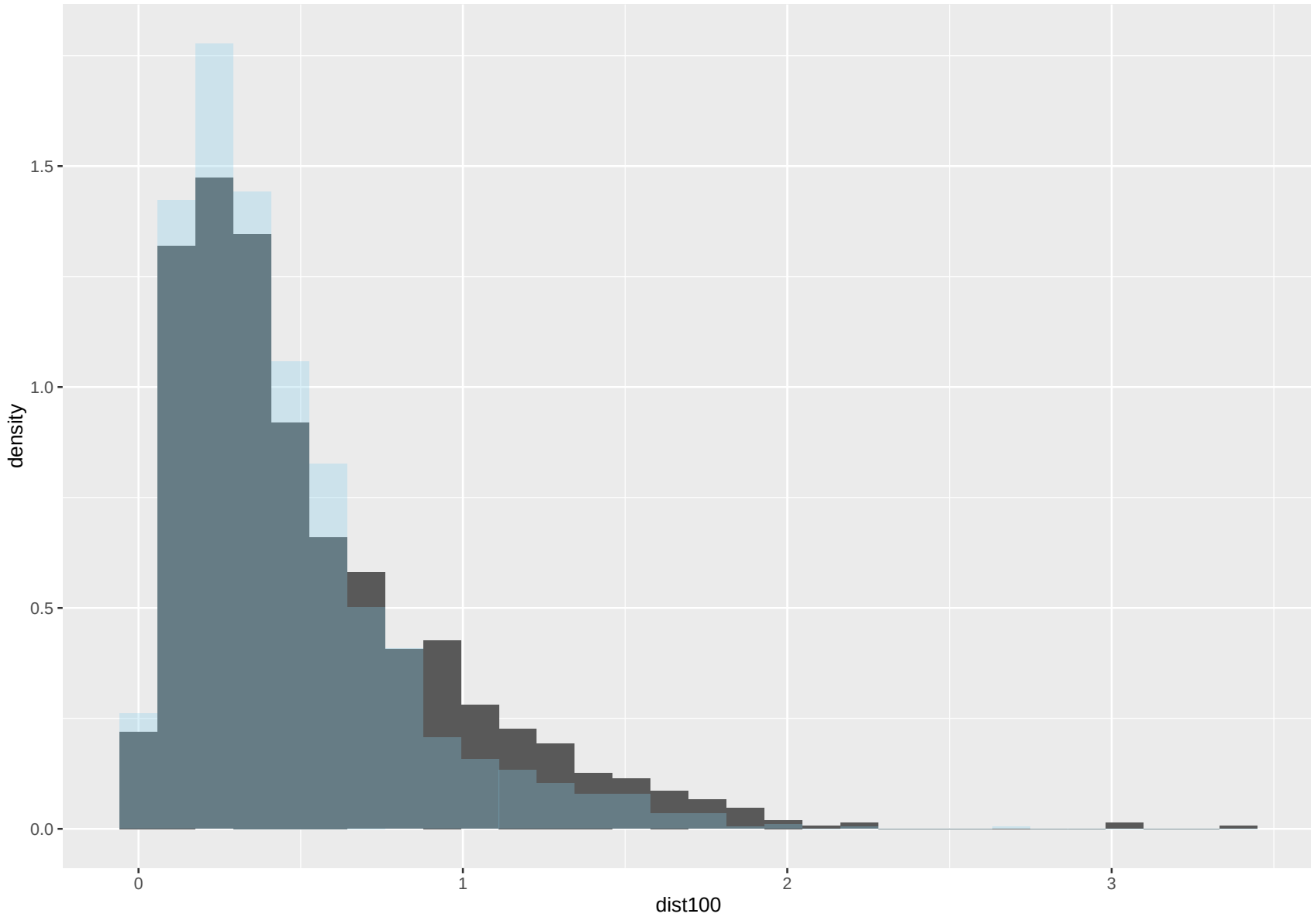
Use distance in meters between respondent's house and nearest safe water well as the only predictor of `switch` (0 or 1). We rescale the distance variable to be in units of 100 meters, since we think the effect of a marginal meter is not useful for interpretation.

```
library(rstanarm)
data(wells)
wells$dist100 <- wells$dist / 100
```

Visualizing distance data

```
library(ggplot2)
ggplot(wells, aes(x = dist100, y = ..density..)) +
  geom_histogram(data = subset(wells, switch == 0)) +
  geom_histogram(data = subset(wells, switch == 1),
                 fill = "skyblue", alpha = 0.3)
```

Blue bars correspond to the 1737 residents who said they switched wells and the bars with the darker outline show the distribution of dist100 for the 1283 residents who didn't switch.


1.5
1.0
0.5
0.0
density
0
1
2
3
dist100

Bayesian fit

The t-prior here is reasonable if coefficients are expected to be close to zero have have some non-negligible probability of being large.

```
t_prior <- student_t(df = 7, location = 0, scale = 2.5)
fit1 <- stan_glm(switch ~ dist100, data = wells,
         family = binomial(link = "logit"),
         prior = t_prior, prior_intercept = t_prior)
```

```
SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).

Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1, Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1, Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1, Iteration: 1001 / 2000 [ 50%]  (Sampling)
```

Estimates and Intervals

```
(coef_fit1 <- round(coef(fit1), 3))
```

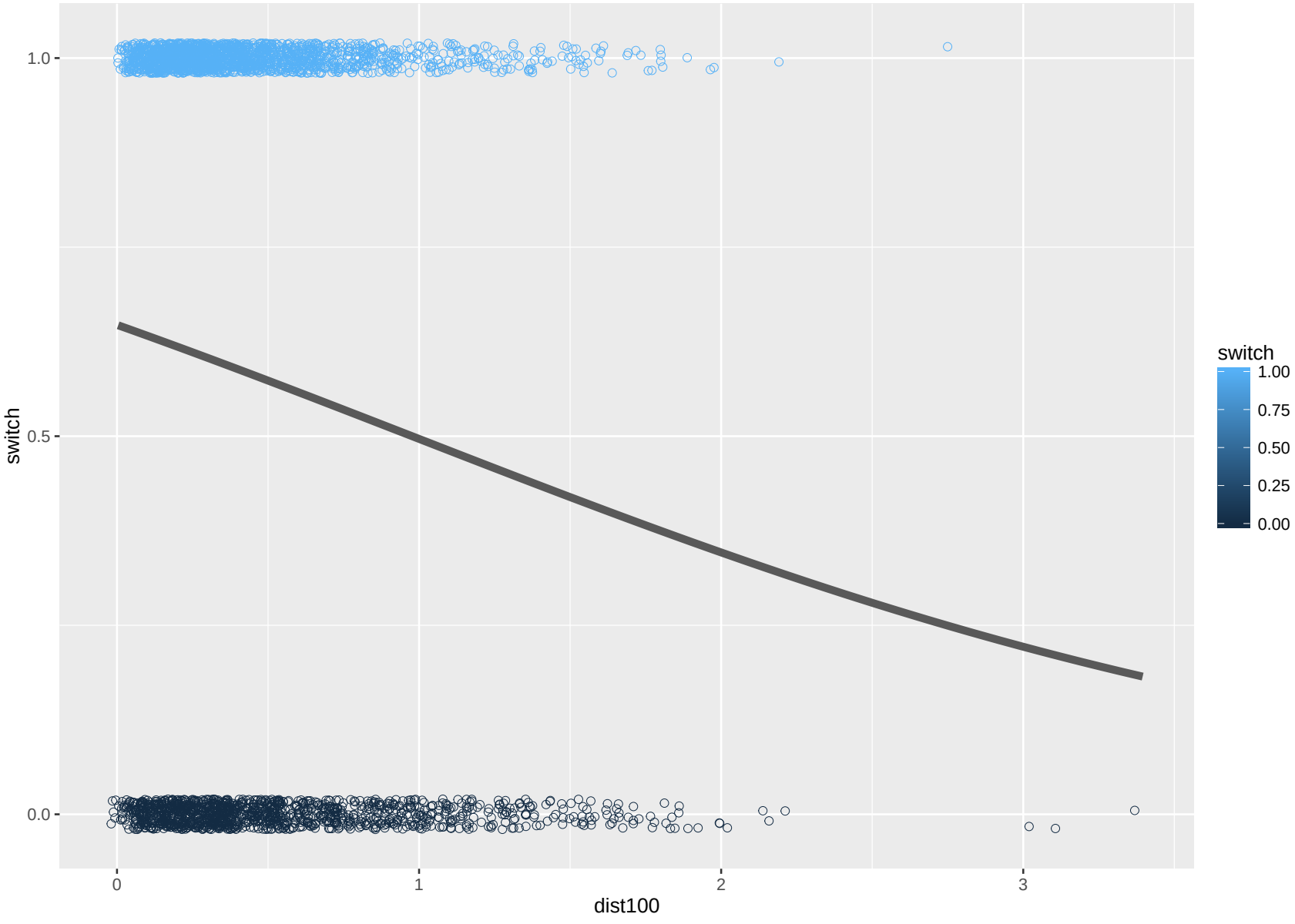
```
(Intercept)     dist100
      0.607      -0.621
```

```
round(posterior_interval(fit1, prob = 0.95), 2)
```

```
             2.5% 97.5%
(Intercept)  0.49  0.72
dist100     -0.82 -0.44
```

Predicted probability of switching vs. actual outcome

```
pr_switch <- function(x, ests) plogis(ests[1] + ests[2] * x)
# A function to slightly jitter the binary data
jitt <- function(...) {
  geom_point(aes_string(...),
      position = position_jitter(height = 0.05, width = 0.1),
             size = 2, shape = 21, stroke = 0.2)
}
ggplot(wells, aes(x = dist100, y = switch, color = switch)) +
  scale_y_continuous(breaks = c(0, 0.5, 1)) +
  jitt(x="dist100") +
  stat_function(fun = pr_switch, args = list(ests = coef(fit1)),
                size = 2, color = "gray35")
```


switch
1.0
0.5
0.0
0
1
2
3
dist100
switch
1.00
0.75
0.50
0.25
0.00

Comment

The plot shows that with this model, predicted probability of switching is above 50% for residents living close to safe water wells. For 300 meter distance, probability is about 25%.

Add another predictor: the arsenic level in the respondent's well

```
fit2 <- update(fit1, formula = switch ~ dist100 + arsenic)
```

```
SAMPLING FOR MODEL 'bernoulli' NOW (CHAIN 1).

Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1, Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1, Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1, Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1, Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```r
(coef_fit2 <- round(coef(fit2), 3))
```

```
(Intercept)     dist100     arsenic
      0.002      -0.898       0.460
```

```r
round(posterior_interval(fit2, prob = 0.95), 2)
```

```
             2.5% 97.5%
(Intercept) -0.16  0.17
dist100     -1.10 -0.69
arsenic      0.38  0.55
```

```r
(coef_fit1 <- round(coef(fit1), 3))
```

```
(Intercept)     dist100
      0.607      -0.621
```

```r
round(posterior_interval(fit1, prob = 0.95), 2)
```

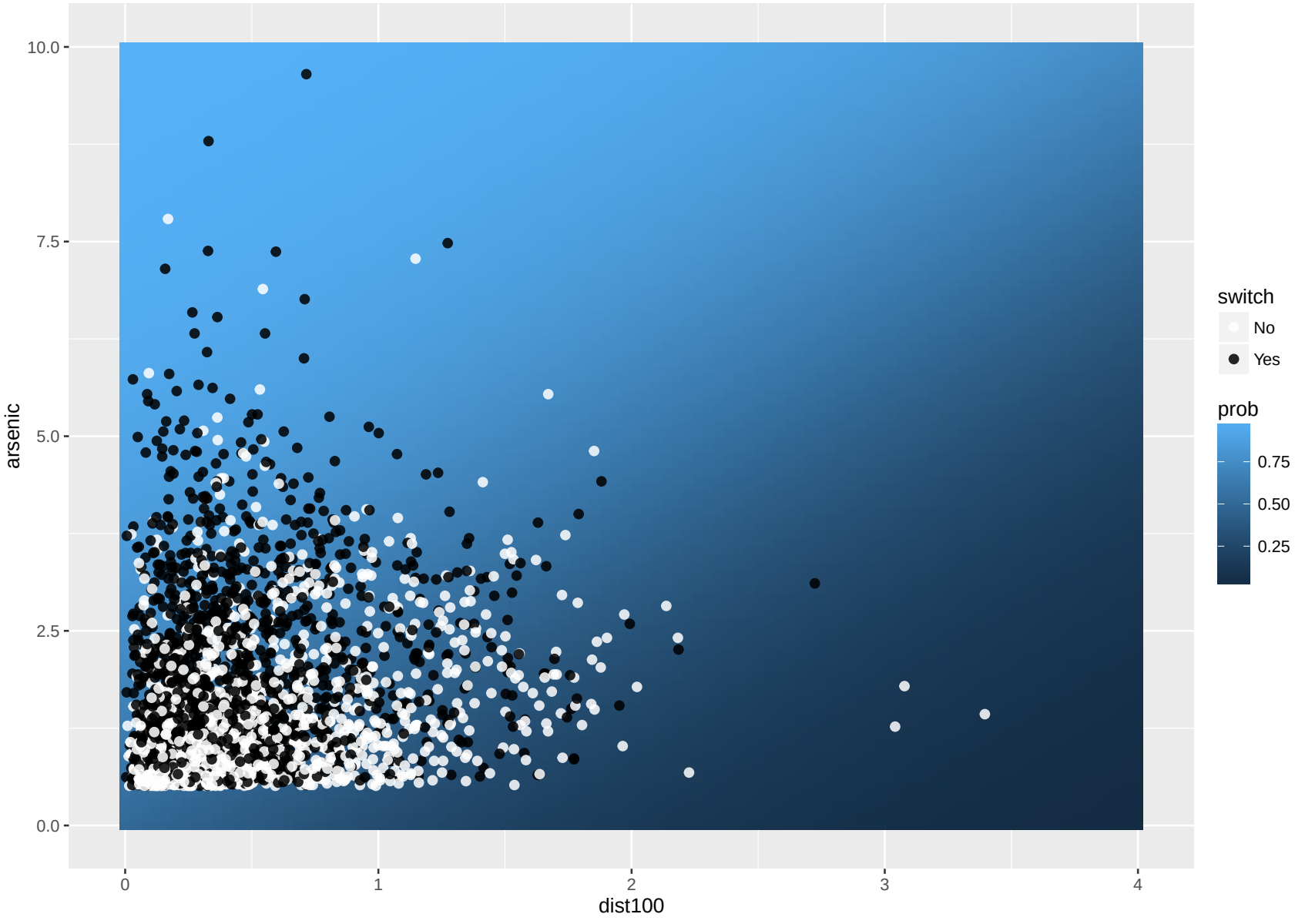
```
             2.5% 97.5%
(Intercept)  0.49  0.72
dist100     -0.82 -0.44
```

A new plot

The plot below shows distance on the x-axis and arsenic level on the y-axis with the predicted probability of switching mapped to the color of the background tiles.

- the lighter the color the higher the probability
- observed value of switch is indicated by the color of the points

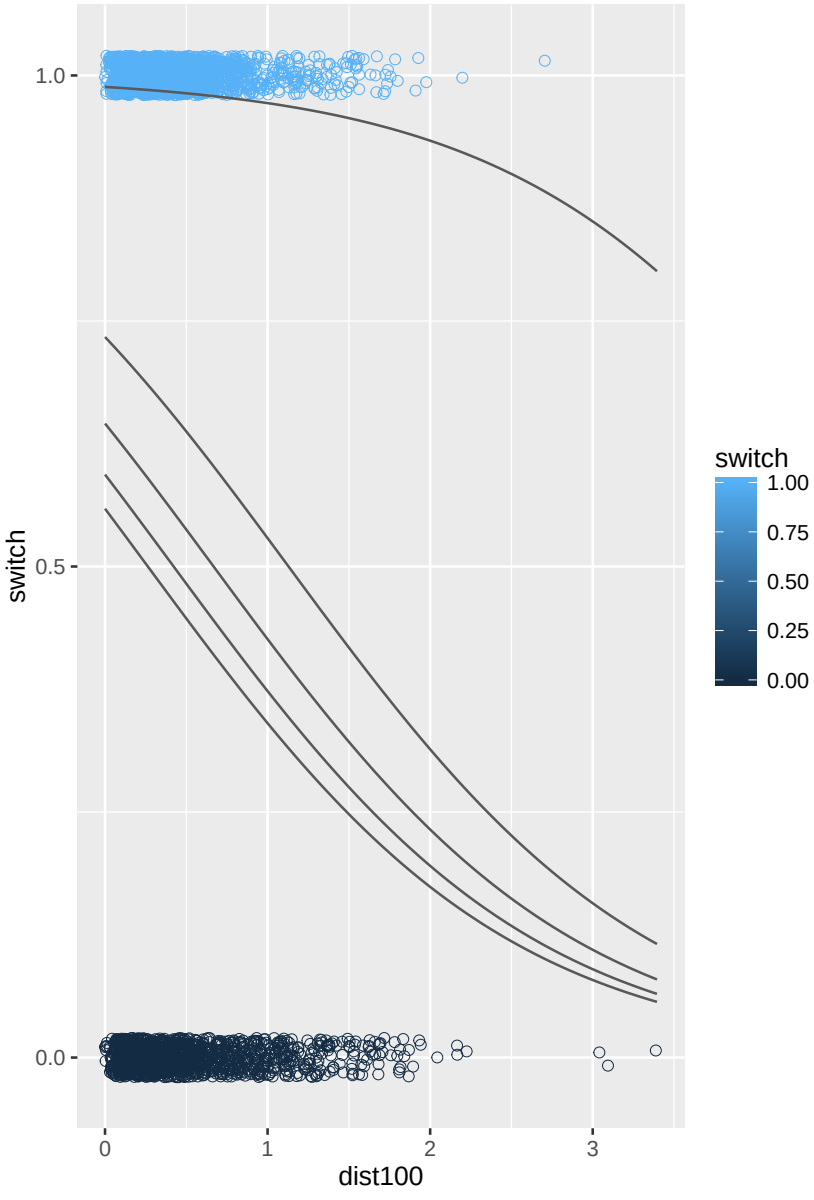
```
pr_switch2 <- function(x, y, ests) plogis(ests[1] +
                            ests[2] * x + ests[3] * y)
grid <- expand.grid(dist100 = seq(0, 4, length.out = 100),
                    arsenic = seq(0, 10, length.out = 100))
grid$prob <- with(grid, pr_switch2(dist100, arsenic,
                                   coef(fit2)))
ggplot(grid, aes(x = dist100, y = arsenic)) +
  geom_tile(aes(fill = prob)) +
  geom_point(data = wells, aes(color = factor(switch)),
             size = 2, alpha = 0.85) +
  scale_fill_gradient() +
  scale_color_manual("switch", values = c("white", "black"),
                     labels = c("No", "Yes"))
```

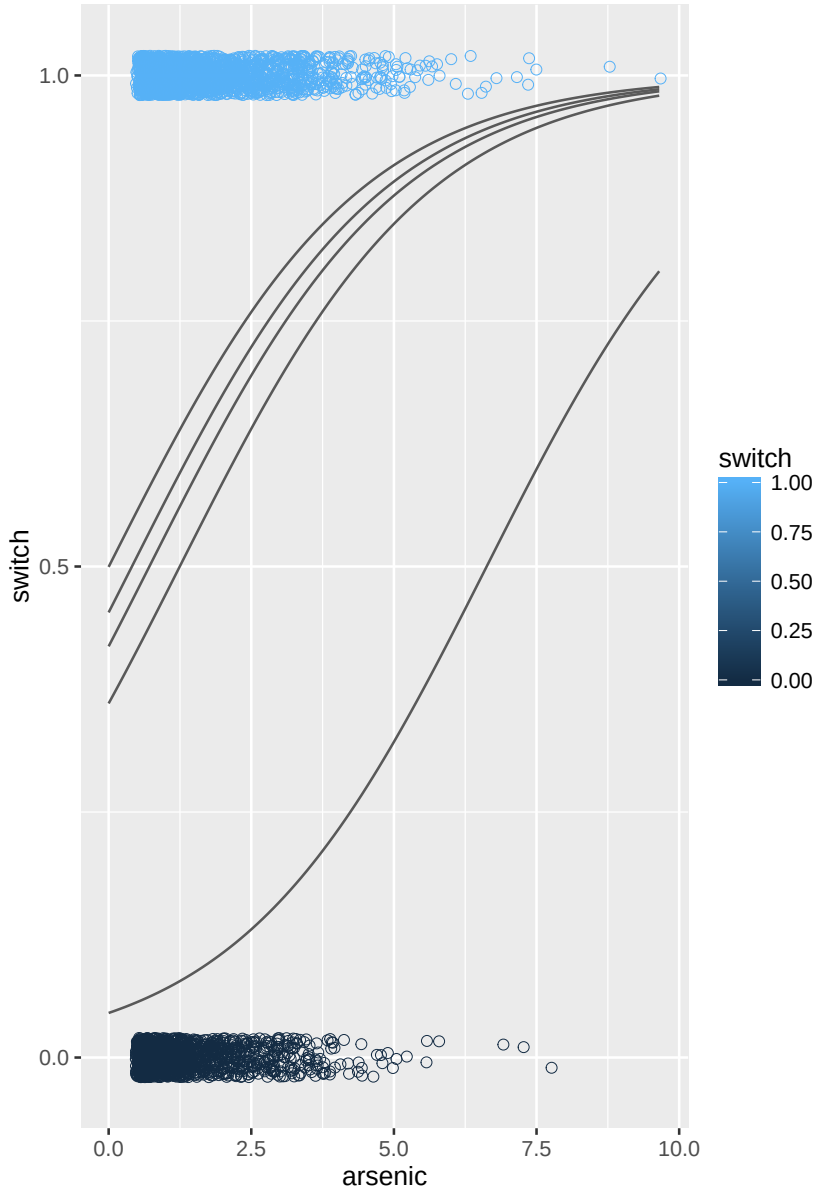

10.0
7.5
5.0
2.5
0.0
arsenic
0
1
2
3
4
dist100
switch
No
Yes
prob
0.75
0.50
0.25

Another visualization using gridExtra

Let's create separate plots for varying the arsenic level and distance. We plot curves representing predicted probability of switching for the min, max and quartile values of both variables.

```
library(gridExtra)
# Quantiles
q_ars <- quantile(wells$dist100, seq(0, 1, 0.25))
q_dist <- quantile(wells$arsenic, seq(0, 1, 0.25))
base <- ggplot(wells) + xlim(c(0, NA)) +
  scale_y_continuous(breaks = c(0, 0.5, 1))
vary_arsenic <- base + jitt(x="arsenic", y="switch", color="switch")
vary_dist <- base + jitt(x="dist100", y="switch", color="switch")
for (i in 1:5) {
  vary_dist <-
    vary_dist + stat_function(fun = pr_switch2, color = "gray35",
                  args = list(ests = coef(fit2), y = q_dist[i]))
  vary_arsenic <-
    vary_arsenic + stat_function(fun = pr_switch2, color = "gray35",
                args = list(ests = coef(fit2), x = q_ars[i]))
}
grid.arrange(vary_dist, vary_arsenic, ncol = 2)
```


switch
1.0
0.5
0.0
0
1
2
3
dist100
switch
1.00
0.75
0.50
0.25
0.00


switch
1.0
0.5
0.0
0.0
2.5
5.0
7.5
10.0
arsenic
switch
1.00
0.75
0.50
0.25
0.00

Count Data Response

Suppose $y|\lambda \sim \text{Poi}(\lambda)$ with mass function $\lambda^y e^{-\lambda}/(y!)$.

- $E(y|x) = \lambda = g^{-1}(\eta)$, $\eta = \alpha + x^T\beta$
- also $\text{var}(y|x) = \lambda$
- the rate parameter must be positive, so the link function g maps from the support of λ $\mathbb{R}^+ \mapsto \mathbb{R}$
- then inverse link $g^{-1}(\eta)$ maps from $\mathbb{R} \mapsto \mathbb{R}^+$

Canonical link for Poisson GLM is $g(x) = \log x$. Then inverse link is exponential and the likelihood for one observation is

$$\frac{g^{-1}(\eta)^y}{y!} e^{-g^{-1}(\eta)} = \frac{e^{\eta y}}{y!} e^{-e^\eta}$$

Priors and Posterior

Prior arguments are same as above for binomial regression, i.e. priors on α, β by default are usually mean-zero, small std. dev. distributions such as normal or t with moderate to large d.f.

Posterior with independent priors is

$$f(\alpha, \beta | y, X) \propto f(\alpha) \prod_{k=1}^{K} f(\beta_k) \times \prod_{i=1}^{N} \frac{g^{-1}(\eta_i)^{y_i}}{y_i!} e^{-g^{-1}(\eta_i)}.$$

Example from Gelman & Hill (2007)

Interested in efficacy of a pest control system at reducing roaches in urban apartments.

- treatment and control applied to 160 and 104 apartments, respectively
- outcome measurement y_i in each apartment i was number of roaches caught in a set of traps
- different apartments had traps for different numbers of days

Other than intercept, model predictors are pre-treatment number of roaches `roach1`, treatment indicator and an indicator for whether or not the aprtment is in a seniors-only building.

- since the # days with traps not constant, we use it as a measure of exposure, which changes the likelihood function so that the rate parameter $\lambda_i = \exp(\eta_i)$ is multiplied by the exposure u_i giving $y_i \sim \text{Poi}(\mu_i \lambda_i)$
- equivalent to adding $\log(u_i)$ to the linear predictor η_i and is specified using the `offset` argument in the stan glm function

Fit with Stan

```
library(rstanarm)
data(roaches)
roaches$roach1 <- roaches$roach1 / 100
# Estimate original model
glm1 <- glm(y ~ roach1 + treatment + senior,
            offset = log(exposure2),
            data = roaches, family = poisson)
# Estimate Bayesian version with stan_glm
stan_glm1 <- stan_glm(y ~ roach1 + treatment + senior,
                      offset = log(exposure2),
                      data = roaches, family = poisson,
                      prior = normal(0,2.5),
                      prior_intercept = normal(0,5))
```

```
SAMPLING FOR MODEL 'count' NOW (CHAIN 1).

Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
```

Estimates and estimated standard errors

```
round(rbind(glm = coef(glm1), stan_glm = coef(stan_glm1)),
      digits = 2)
```

```
          (Intercept) roach1 treatment senior
glm              3.09    0.7     -0.52  -0.38
stan_glm         3.09    0.7     -0.52  -0.38
```

```
round(rbind(glm = summary(glm1)$coefficients[, "Std. Error"],
            stan_glm = se(stan_glm1)), digits = 3)
```

```
          (Intercept) roach1 treatment senior
glm             0.021  0.009     0.025  0.033
stan_glm        0.021  0.009     0.024  0.032
```

Checking model fit

One proposal: generate dataset from the model and compare with observed data to check quality of fit.

Think of the proportion of zeros like a test statistic; can compare observed value in sample to sampling distribution (under repeated sampling); not very Bayesian....

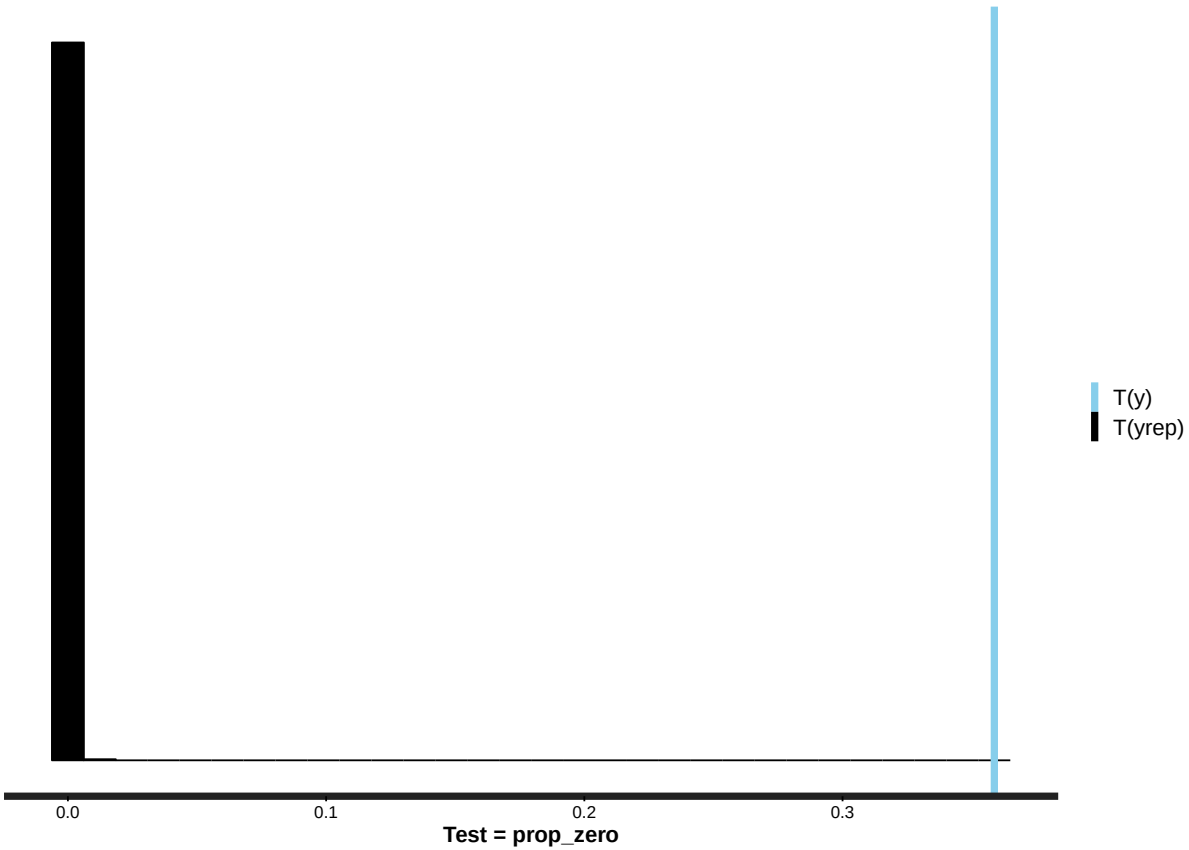
Can generate replicated datasets with a single line of code:

```
yrep <- posterior_predict(stan_glm1)
```

Creating a plot

Take the simulated datasets and compute the proportion of zeros; compare to the observed proportion in the original data:

```
prop_zero <- function(y) mean(y == 0)
(prop_zero_test1 <- pp_check(stan_glm1, check = "test",
                             test = "prop_zero"))
```



Criticism

Thinking of the proportion of zeros like a test statistic, this shows less than 1% of the replicated proportions are zero, but the observed proportion is 30%. Seems to be a bad model.

One proposal: use an overdispersed Poisson distribution.

Another proposal: use a different response distribution, e.g. negative binomial which will allow the conditional mean and variance of y to be different (Poisson will not)

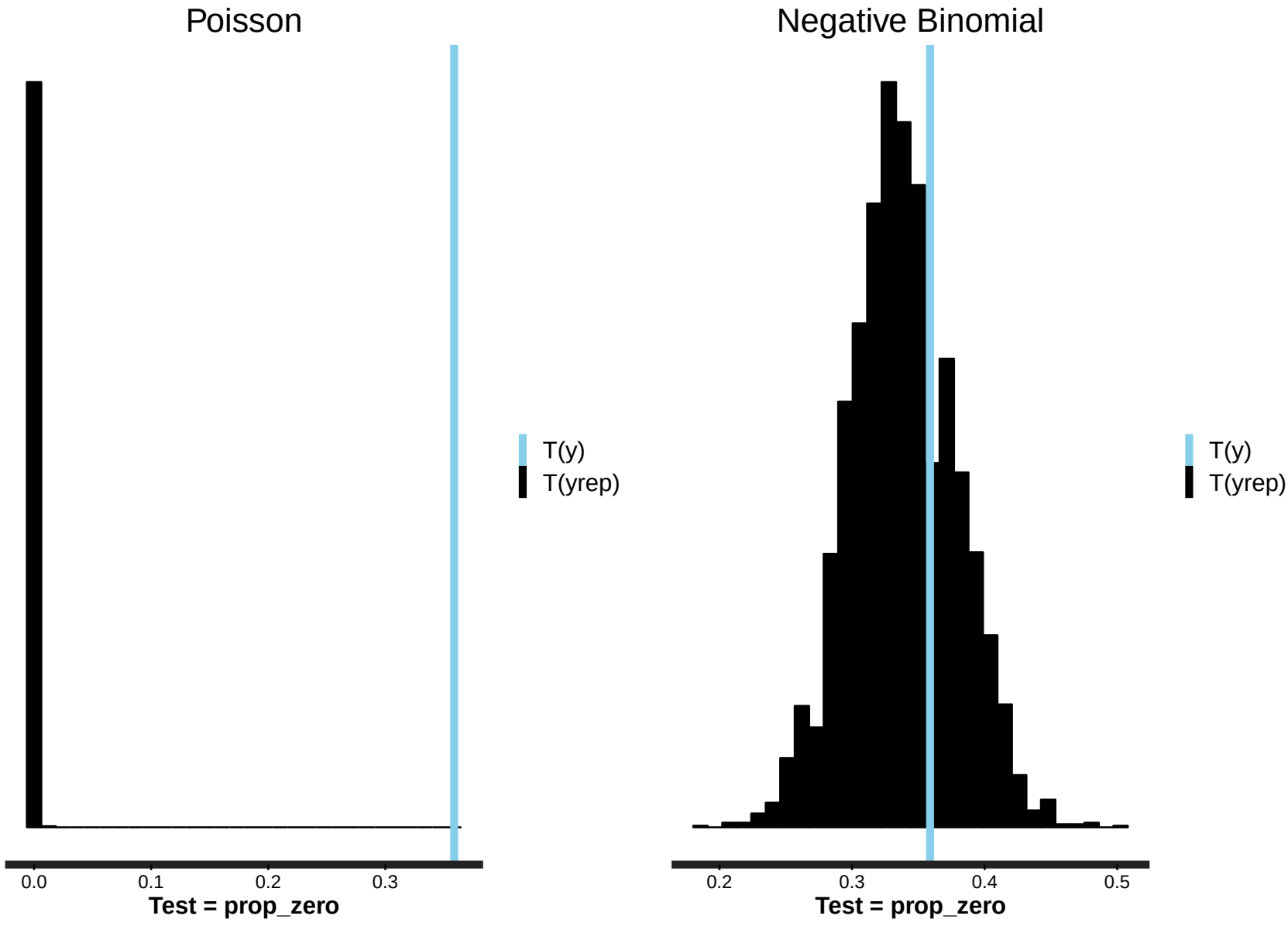
```
stan_glm2 <- update(stan_glm1, family = neg_binomial_2)
```

```
SAMPLING FOR MODEL 'count' NOW (CHAIN 1).

Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
```

Comparing Oberved vs. Replicated Proportion of Zeros in Each Model

```
library(gridExtra)
prop_zero_test2 <- pp_check(stan_glm2, check = "test",
                            test = "prop_zero")
# Show graphs for Poisson and negative binomial side by si
grid.arrange(prop_zero_test1 + ggtitle("Poisson"),
             prop_zero_test2 + ggtitle("Negative Binomial"
             ncol = 2)
```


Poisson
T(y)
T(yrep)
0.0
0.1
0.2
0.3
Test = prop_zero
Negative Binomial
T(y)
T(yrep)
0.2
0.3
0.4
0.5
Test = prop_zero