Lab 3: The Poisson model and posterior predictive checks

Due: 11:59 pm on Friday, February 13

Turning in solutions

This lab is part of Homework 3. Solutions to the exercises, as well as the non-lab homework exercises are to be written up and uploaded to Gradescope as a PDF.

Getting started

You will need the following R packages. If you do not already have them installed, please do so first using the `install.packages` function.

```r
require(tidyverse)
require(rstanarm)
require(magrittr)
require(rstan)
require(bayesplot)
require(loo)
require(readxl)
```

For this lab, you will need two stan files `lab-03-poisson-simple.stan` and `lab-03-poisson-simple.stan`, which you can download here:

- https://omelikechi.github.io/sta402spring26/labs/lab03/lab-03-poisson-simple.stan; and
- https://omelikechi.github.io/sta402spring26/labs/lab03/lab-03-poisson-hurdle.stan.

Download both and make sure to save them in the same folder as the R script or R markdown file you are working from.

The Data

The data we have are counts of deaths per episode of Game of Thrones. Download the data (here: https://omelikechi.github.io/sta402spring26/labs/lab03/GoT_Deaths.xlsx) and save it locally to the same directory as your R markdown file. Once you have downloaded the data file into the SAME folder as your R markdown file, load the data by using the following R code.

```r
GoT <- read_xlsx("GoT_Deaths.xlsx", col_names = T)
head(GoT)
```

```
## # A tibble: 6 x 3
##   Season Episode Count
##    <dbl>   <dbl> <dbl>
## 1      1       1     4
## 2      1       2     3
## 3      1       3     0
## 4      1       4     1
## 5      1       5     5
## 6      1       6     4
```

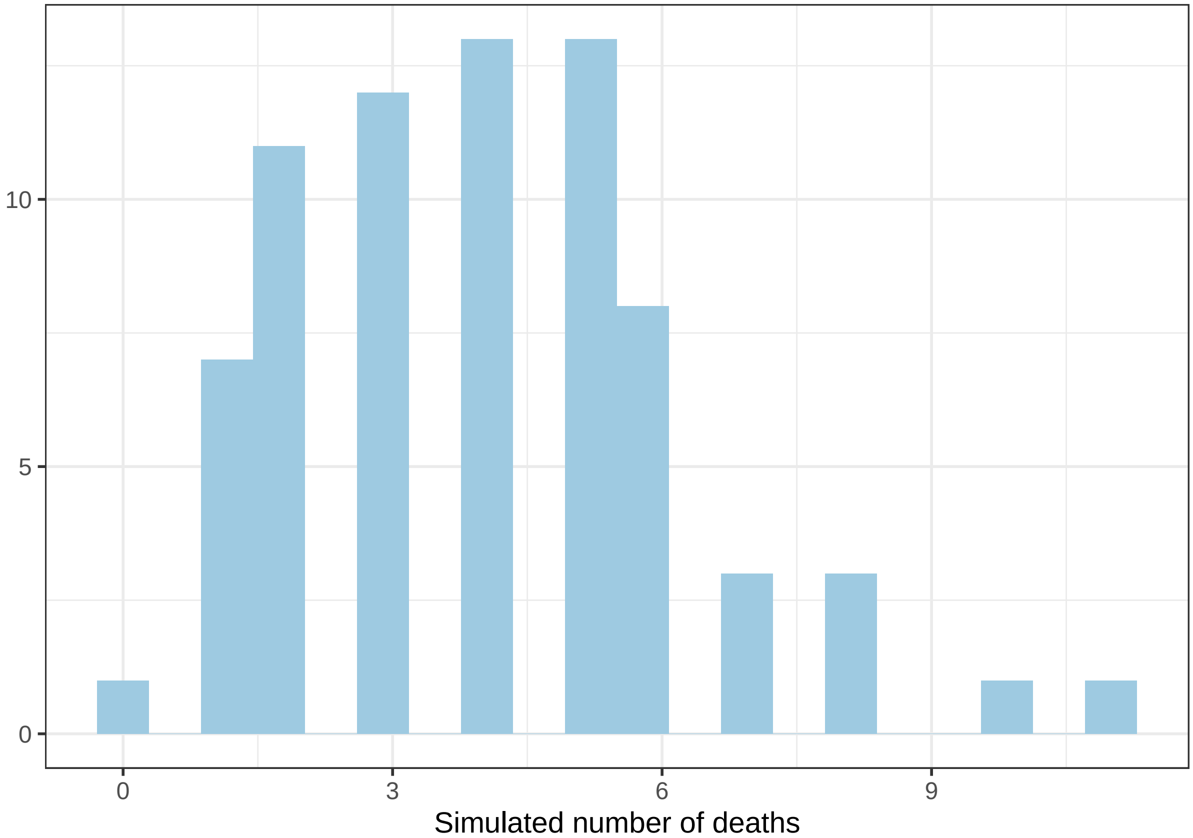
1. Plot a histogram of the death counts.

Because we have count data, it is natural to model the data with a Poisson distribution with parameter λ. If we take the empirical mean of the data as our estimate for λ, what does the sampling distribution look like?

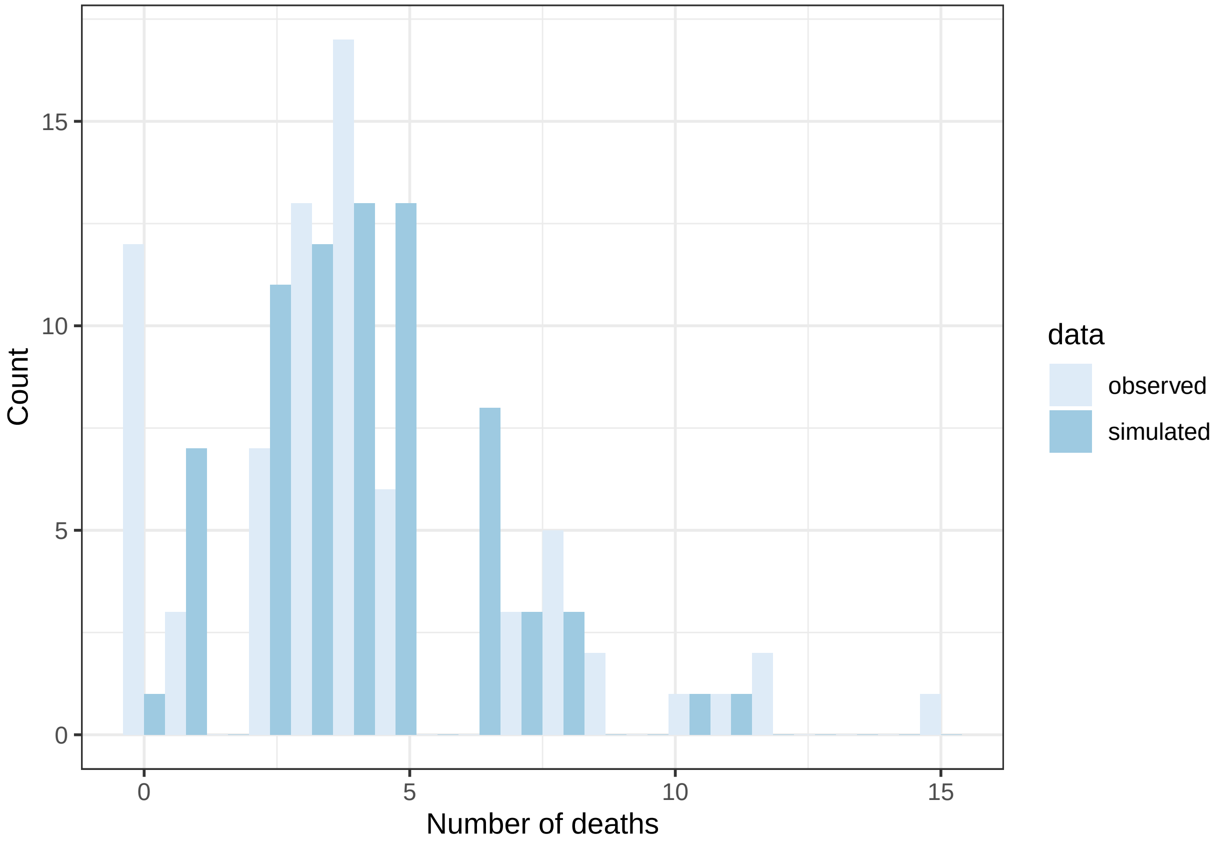
```
y <- GoT$Count
n <- length(y)

# Finish: obtain empirical mean
ybar <- mean(y)
sim_dat <- rpois(n, ybar)
qplot(sim_dat, bins = 20, xlab = "Simulated number of deaths", fill = I("#9ecae1"))
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once per session.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



```
# create data frame for side-by-side histograms
df <- rbind(data.frame(y, "observed") %>% rename(count = 1, data = 2),
            data.frame(sim_dat, "simulated") %>% rename(count = 1, data = 2))
ggplot(df, aes(x = count, fill = data))+
  geom_histogram(position = "dodge", bins = 20) +
  scale_fill_brewer()  +
  labs(x = "Number of deaths", y = "Count")
```



Plotting the histogram of the data next to the histogram of the sampled data using $\hat{\lambda} = \bar{y}$, we see that there are many more 0-valued observations in the observed data than there are in the simulated data. We might suspect that the Poisson model will not be a good fit for the data. For now, let's proceed using the Poisson model and see how it behaves.

Poisson model

We have learned that the Gamma distribution is conjugate for Poisson data. Here, we have chosen the prior $\lambda \sim Gamma(10, 2)$ based on prior knowledge that Game of Thrones is a show filled with death. We run the model in Stan. We save the posterior sampled values of λ in the variable `lambda_draws`.

```r
stan_dat <- list(y = y, N = n)
fit <- stan("lab-03-poisson-simple.stan", data = stan_dat, refresh = 0, chains = 2)
lambda_draws <- as.matrix(fit, pars = "lambda")
```

---

2. Use the function `mcmc_areas()` to plot the smoothed posterior distribution for λ with a 90% Highest Posterior Density region. Changing the `prob` parameter in `mcmc_areas()` will change the amount of probability mass that is highlighted. The highlighted region will begin in regions of highest density, and will move towards regions of lower density as `prob` gets larger.

```r
mcmc_areas(lambda_draws, prob = 0.9)
print(fit, pars = "lambda")
```

The line in the plot represents the posterior mean, and the shaded area represents 90% of the posterior density. How does the posterior mean for λ compare to the sample mean?

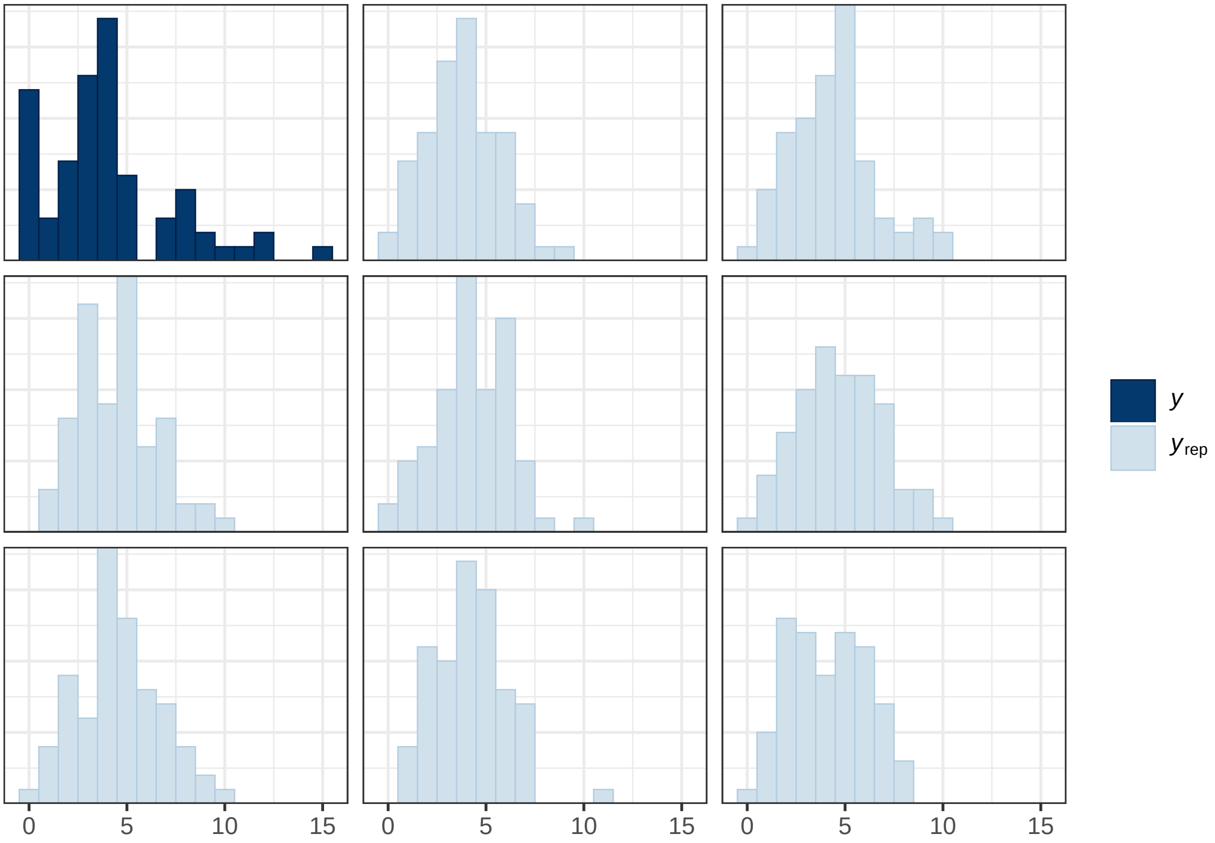
Graphical Posterior Predictive Checks (PPC)

---

3. Generate posterior predictive samples using the posterior values of λ and store the result in a `length(lambda_draws)` by `n` matrix called `y_rep`.

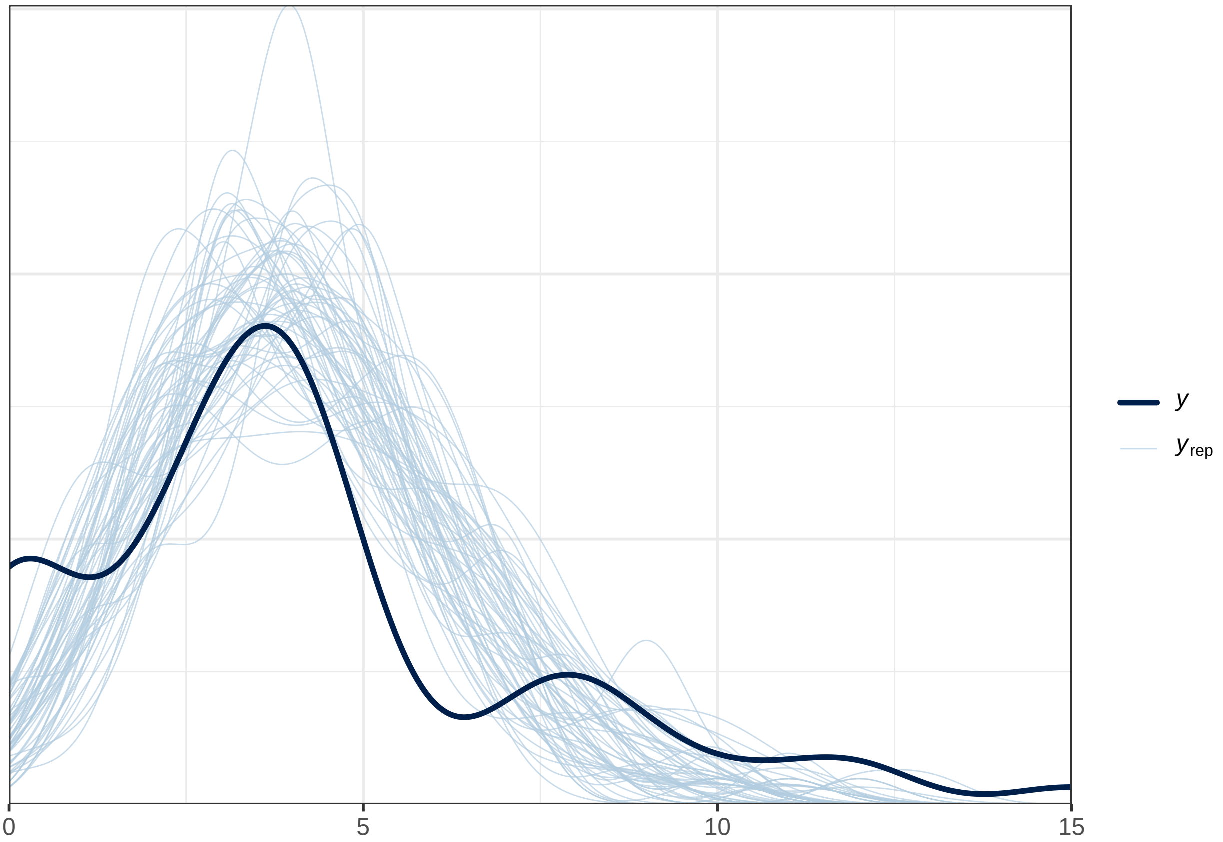
We now step through some graphical posterior predictive checks. Here we plot the histogram of the observed counts against several of the posterior predictions. What do you notice?

```
ppc_hist(y, y_rep[1:8, ], binwidth = 1)
```



We can compare the density estimate of y to the density estimates for several (60) of the y_reps. What do you notice here?

```
ppc_dens_overlay(y, y_rep[1:60, ])
```



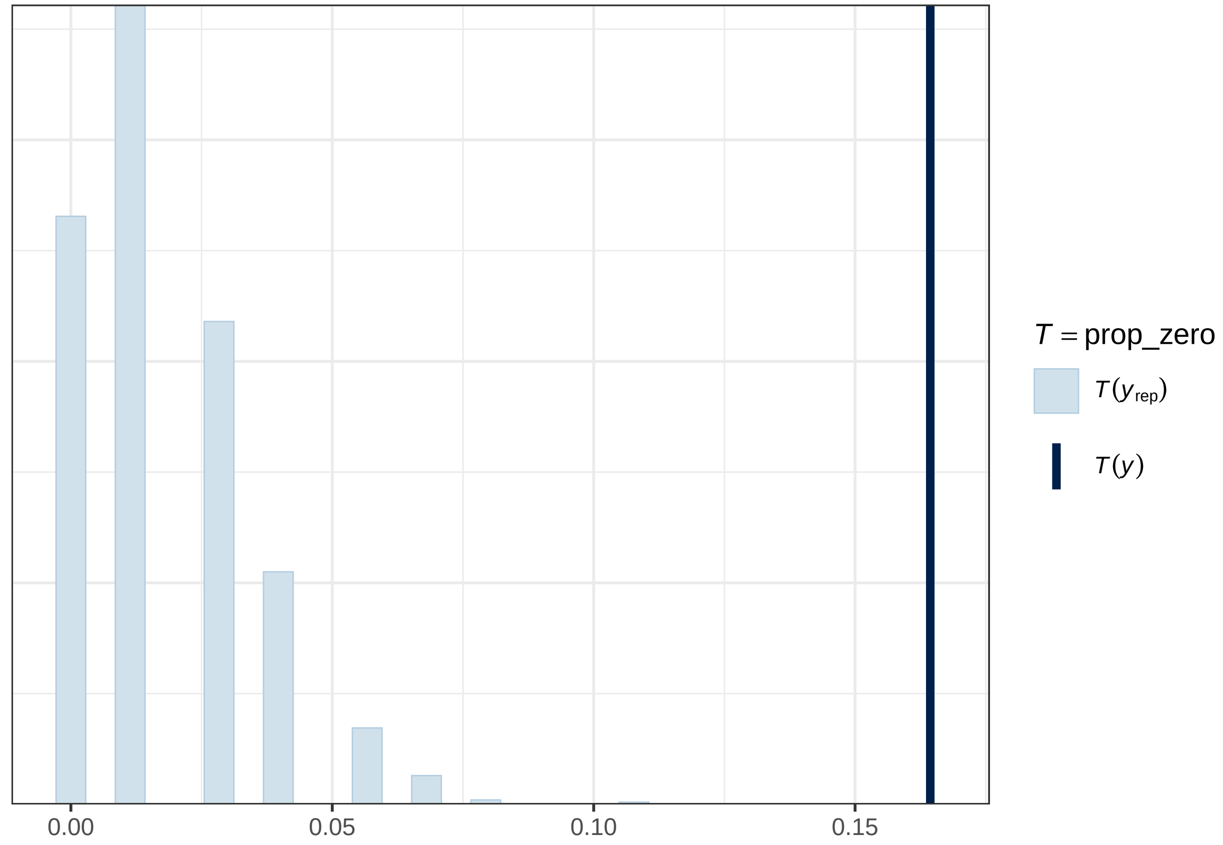
In our first histogram of the data, we noticed the high number of 0 counts. Let us calculate the proportion of zeros in y, and compare that to the proportion of zeros in all of the posterior predictive samples.

```
prop_zero <- function(x){
  mean(x == 0)
}
prop_zero(y)
```

```
## [1] 0.16
```

```
ppc_stat(y, y_rep, stat = "prop_zero")
```

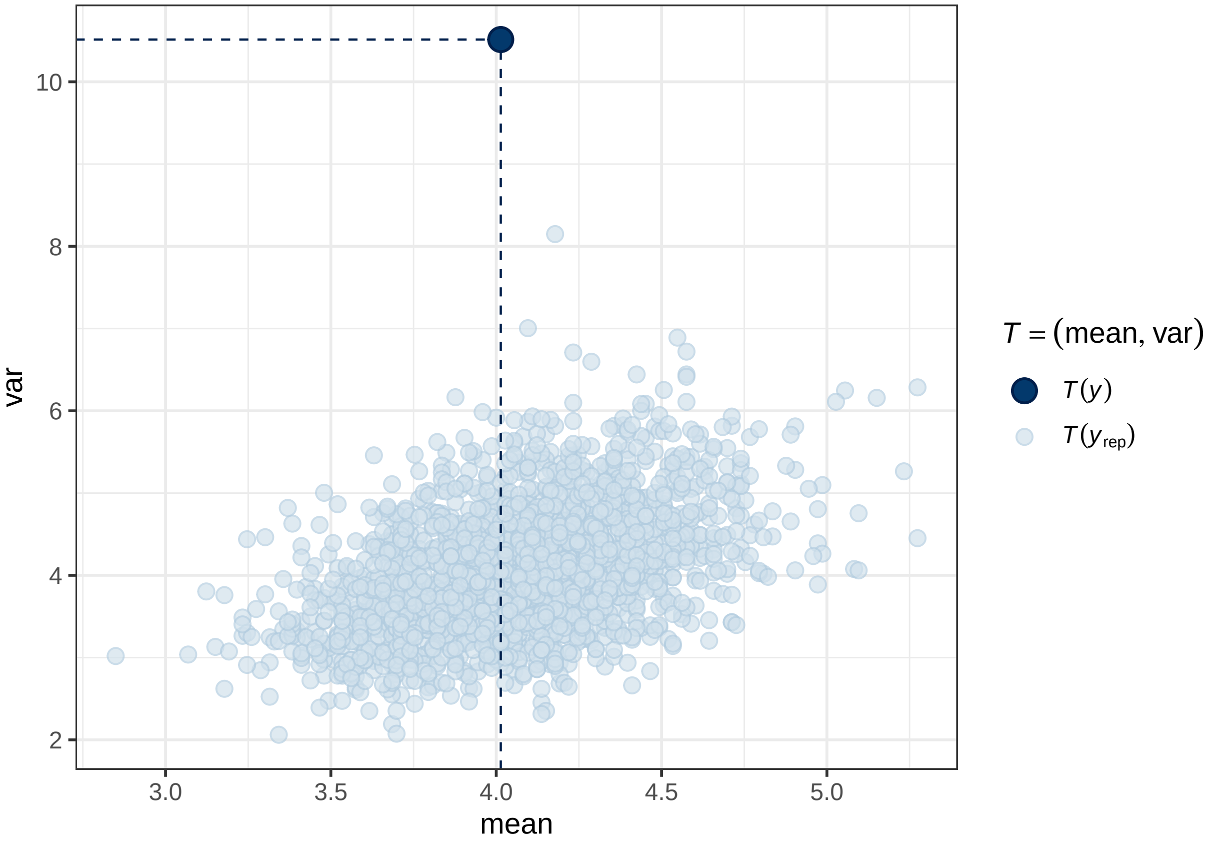
```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



We can plot the means and variances for all of the simulated datasets, along with the observed statistics.

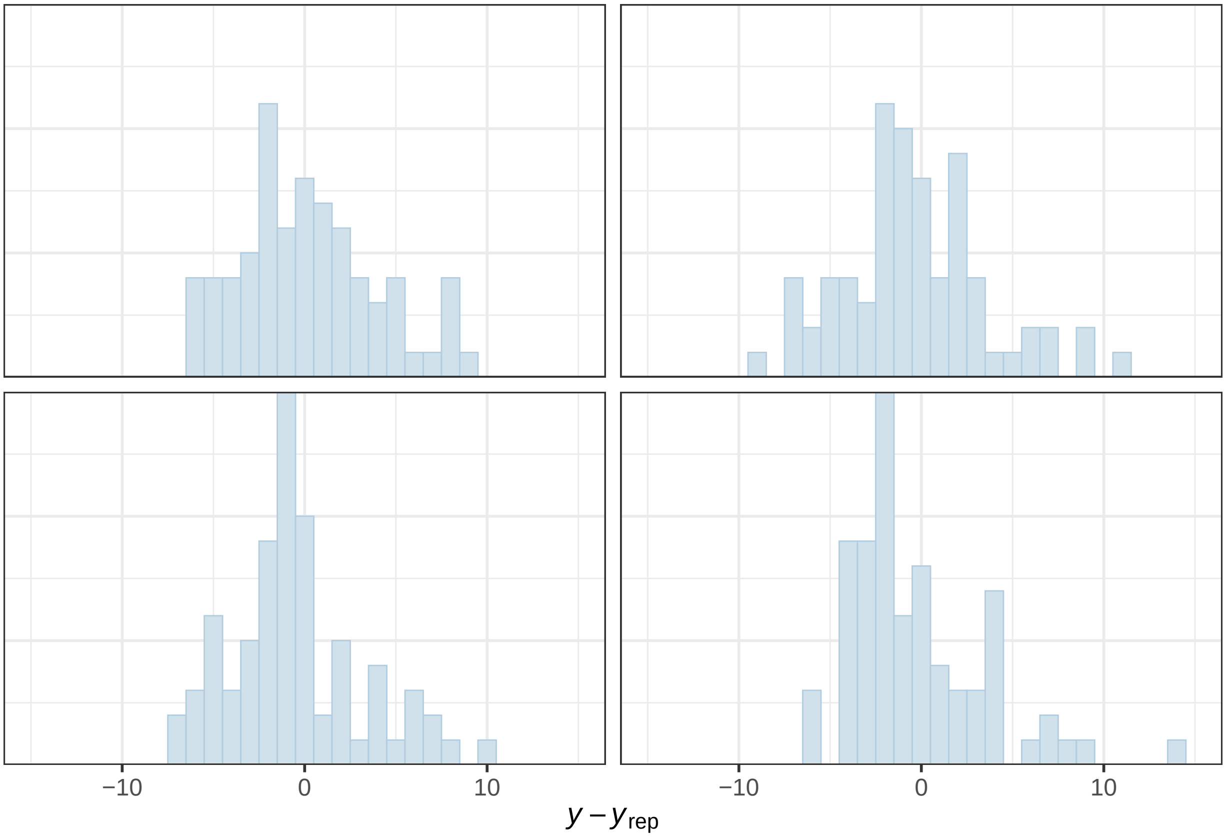
```
ppc_stat_2d(y, y_rep, stat = c("mean", "var"))
```

```
## Note: in most cases the default test statistic 'mean' is too weak to detect anything of interest.
```



```
ppc_error_hist(y, y_rep[1:4, ], binwidth = 1) + xlim(-15, 15)
```

```
## Warning: Removed 8 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```



---

4. Based on on these PPCs, does this model appear to be a good fit for the data?

Poisson Hurdle Model

How should we account for the high frequency of zeros? As it turns out, there is a lot of literature on how to work with this sort of data. We can fit a zero-inflated Poisson model, where the data is modeled as a mixture of a Poisson distribution with a point mass at zero. Here, we fit a hurdle model: $y_i = 0$ with probability θ, and $y_i > 0$ with probability $1 - \theta$. Conditional on having observed y_i nonzero, we model y_i with a Poisson distribution which is truncated from below with a lower bound of 1. For the truncated Poisson, we use the same $Gamma(10, 2)$ prior as in the simple Poisson model above.

---

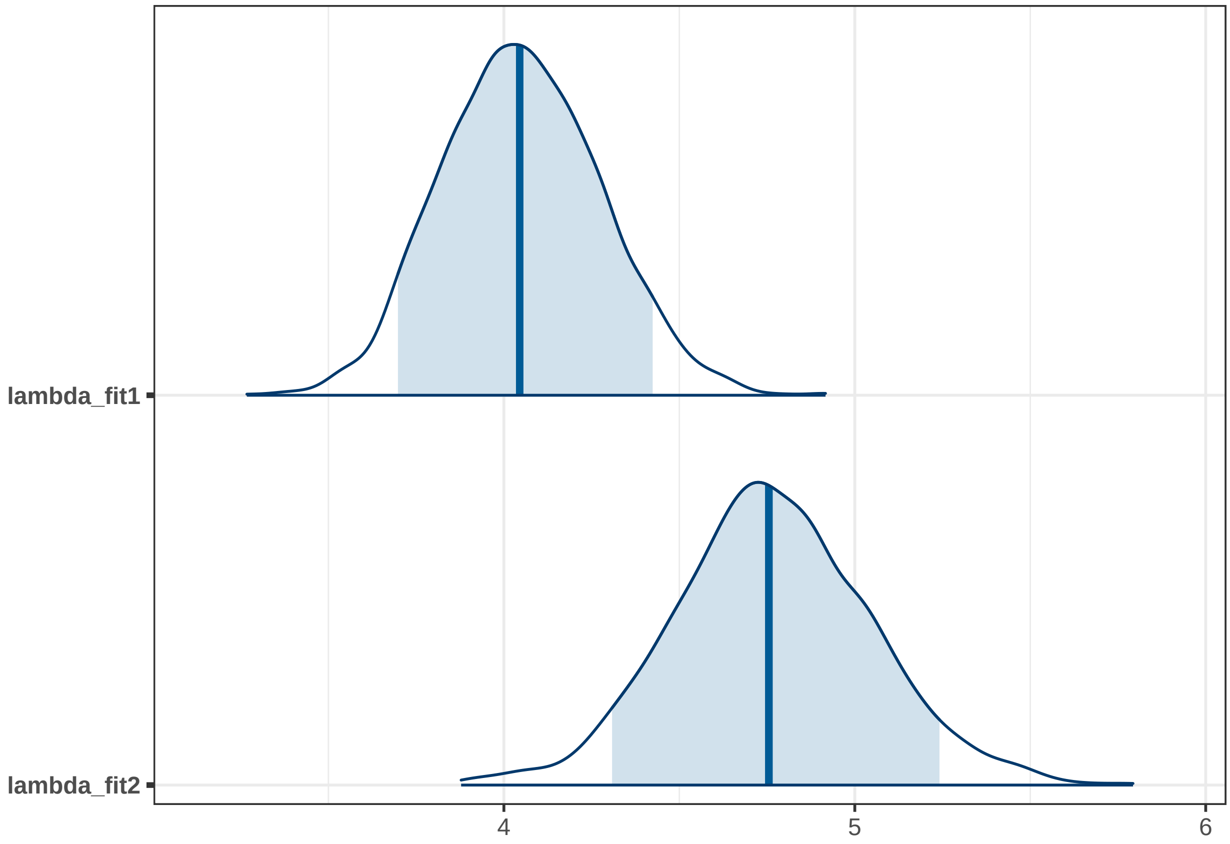
5. Using the code provided for the simple Poisson model, simulate draws from the posterior density of λ with the "Poisson Hurdle" model. The Stan file you'll need to use is called `poisson-hurdle.stan`. Store the resulting object in a variable called `fit2`.

Here we compare the posterior densities of λ from the simple Poisson model and the Poisson hurdle model.

```
# Extract the sampled vlaues for lambda, and store them in a variable called lambda_draws2:
lambda_draws2 <- as.matrix(fit2, pars = "lambda")

# Compare
lambdas <- cbind(lambda_fit1 = lambda_draws[, 1],
                   lambda_fit2 = lambda_draws2[, 1])

# Shade 90% interval
mcmc_areas(lambdas, prob = 0.9)
```



Obtaining posterior samples from the hurdle model is more complicated, so the Stan file includes code to draw from the posterior predictive distribution for you. We extract them here, and store the predictive samples in `y_rep2`.

```
y_rep2 <- as.matrix(fit2, pars = "y_rep")
```

---

6. Use the code given above to produce the same PPC visualizations as before for the new results. Comment on how this second model compares to both the observed data and to the simple Poisson model.

Leave-one-out cross-validation

Here, we assess the predictive performance of both models with leave-one-out cross-validation (LOOCV). LOOCV holds out a single observation form the sample, fits the model on the remaining observations, uses the held out data point as validation, and calculates the prediction error. This process is repeated n times, such that each observation has been held out exactly once. We make use of the hand loo() function, and compare the two models:

```
log_lik1 <- extract_log_lik(fit, merge_chains = FALSE)
r_eff1 <- relative_eff(exp(log_lik1))
(loo1 <- loo(log_lik1, r_eff = r_eff1))
```

```
##
## Computed from 2000 by 73 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo   -203.1 14.5
## p_loo         2.4  0.5
## looic       406.1 29.0
## ------
## MCSE of elpd_loo is 0.1.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.4, 0.4]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
log_lik2 <- extract_log_lik(fit2, merge_chains = FALSE)
r_eff2 <- relative_eff(exp(log_lik2))
(loo2 <- loo(log_lik2, r_eff = r_eff2))
```

```
##
## Computed from 2000 by 73 log-likelihood matrix.
##
##          Estimate   SE
## elpd_loo   -183.4 10.8
## p_loo         2.9  0.5
## looic       366.8 21.7
## ------
## MCSE of elpd_loo is 0.0.
## MCSE and ESS estimates assume MCMC draws (r_eff in [0.7, 0.9]).
##
## All Pareto k estimates are good (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
compare(loo1, loo2)
```

```
## Warning in compare(loo1, loo2): 'compare' is deprecated.
## Use 'loo_compare' instead.
## See help("Deprecated")
```

```
## elpd_diff        se
##      19.7       8.5
```

---

7. Which model performs better in terms of prediction?

Wrapping up

Some final questions to consider:

8. Why are PPCs important?
9. Was the second model a good fit for the data? Why or why not?
10. If someone reported a single LOOCV error to you, would that be useful? Why or why not?

Acknowledgement

This lab was adapted from this tutorial by Jordan Bryan and Becky Tang.