Math 463 HW8

Ian McConachie

14.7: Model building and comparison

First we can write a little snippet to get the data from the authors' GitHub repository.

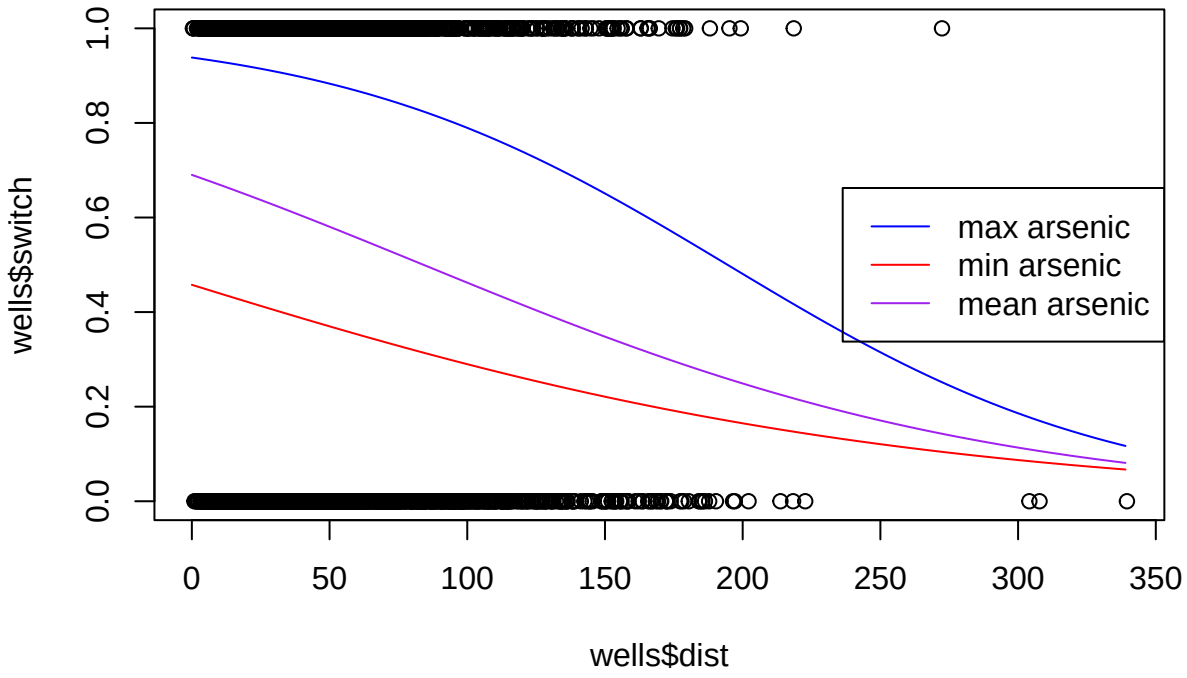
```
myfile <- "https://raw.githubusercontent.com/avehtari/ROS-Examples/master/Arsenic/data/wells.csv"
wells <- read.csv(myfile, header=T)
```

Notice that the switch variable indicating whether the household switched wells after the study is a binary variable (always taking on a value of either 0 or 1). We can use this binary variable as a response and the numeric variables of distance, log(arsenic), and their interaction as predictors in a logistic regression model fit to this data.

```
wfit <- stan_glm(switch~dist*log(arsenic), data=wells, family=binomial(link = "logit"), refresh=0)
```

Now that we have a model fit to this data, we can display the logistic regression model graphically by holding one predictor to different constant values and seeing how the expected predicted probability of switching wells changes as we vary the other predictor. In the graph below, we hold the arsenic level constant (at its max, min, and mean) and show how the probability of switching changes as we go from 0 distance to the maximum distance. Note that the y-axis is on a binary scale for the switch variable, but this also works as an effective scale for probability which always sits between 0 and 1.

```
d <- seq(0,max(wells$dist),1)
wf = coef(wfit)
plot(wells$dist, wells$switch, xlim=c(0,max(wells$dist)))
min_AR = min(log(wells$arsenic)); max_AR = max(log(wells$arsenic)); mean_AR = mean(log(wells$arsenic))
fl_1 <- invlogit(wf[1]+(wf[2]*d) +(wf[3] * min_AR) + (wf[4] * d *min_AR))
lines(d,fl_1,col="red",lty=1)
fl_2 <- invlogit(wf[1]+(wf[2]*d) +(wf[3] * max_AR) + (wf[4] * d *max_AR))
lines(d,fl_2,col="blue",lty=1)
fl_3 <- invlogit(wf[1]+(wf[2]*d) +(wf[3] * mean_AR) + (wf[4] * d *mean_AR))
lines(d,fl_3,col="purple",lty=1)
legend("right", c("max arsenic", "min arsenic", "mean arsenic"),lty = c(1,1), col = c("blue", "red", "pu
```



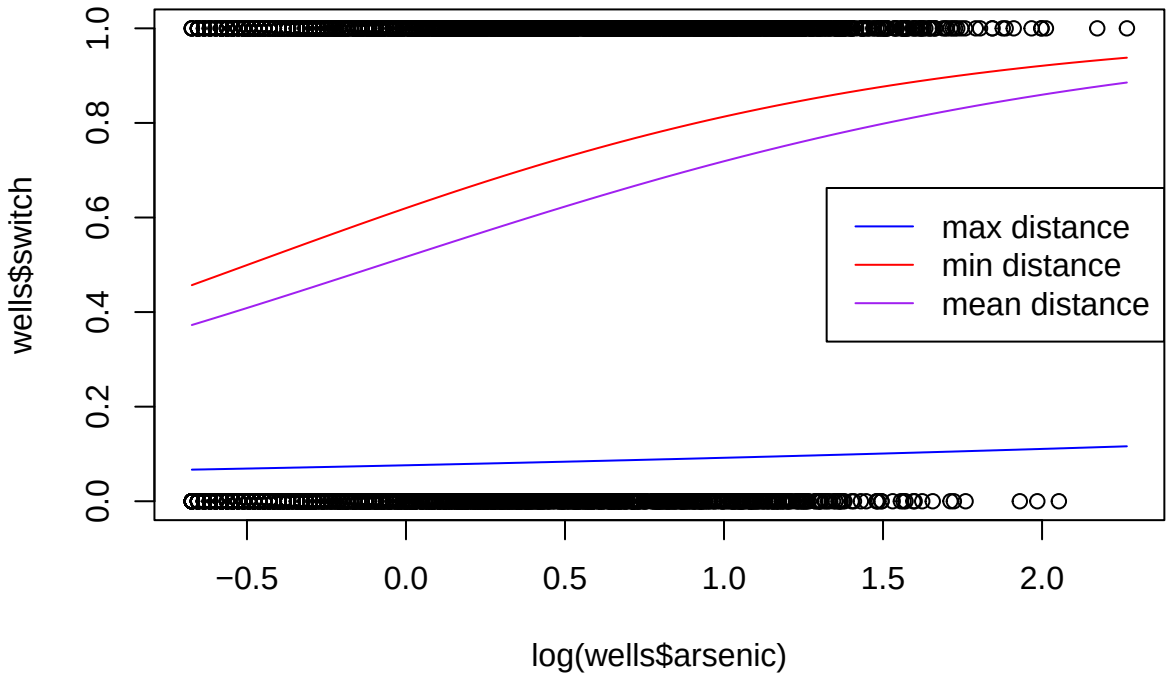
The above graph does not show any real surprises in terms of the trends we see in distance affecting the probability of switching wells: at longer distances, the households are increasingly less likely to switch their well. In terms of the different values of arsenic, we see that at shorter distances, the difference in probability of switching wells is dramatically affected by the amount of arsenic present. This can be explained by the textbooks explanation that "risks estimated to be proportional to exposure"—in small amounts, arsenic may not be that dangerous, so households may not switch even if the nearest well is very close, while very high levels of arsenic provide strong motivation to switch wells.

Now we can generate a similar chart, but this time, we can hold distance constant and have the log of arsenic levels vary across its observed range in the data. In the graph below, we have three lines for the three constant values of distance that we use (its max, its min, and its mean). Note that this display is on the same scale as the previous one with the exception of the x-axis, which now represents the log of arsenic levels in the household's well rather than distance to the next closest well.

```
a <- seq(min(log(wells$arsenic)),max(log(wells$arsenic)),0.01)
wf = coef(wfit)
plot(log(wells$arsenic), wells$switch, xlim=c(min(log(wells$arsenic)),max(log(wells$arsenic))))
min_D = min(wells$dist); max_D = max(wells$dist); mean_D = mean(wells$dist)
fl_4 <- invlogit(wf[1]+(wf[2]*min_D) +(wf[3] * a) + (wf[4] * min_D *a))
lines(a,fl_4,col="red",lty=1)

fl_5 <- invlogit(wf[1]+(wf[2]*max_D) +(wf[3] * a) + (wf[4] * max_D *a))
lines(a,fl_5,col="blue",lty=1)

fl_6 <- invlogit(wf[1]+(wf[2]*mean_D) +(wf[3] * a) + (wf[4] * mean_D *a))
lines(a,fl_6,col="purple",lty=1)
legend("right", c("max distance", "min distance", "mean distance"),lty = c(1,1), col = c("blue", "red",
```



The most apparent observation in the graph above is the stark contrast in probability of switching wells between the maximum distance and the min and mean distance. We see that the minimum distance and the mean distance have generally the same trend for increasing in the probability of switching as arsenic levels increase. However, the maximum distance sees a much smaller increase in likelihood of switching wells even when the arsenic levels are very high. This makes intuitive sense as the nearest well being very far away may make switching very difficult or even impossible. If a household does not have the means or ability to switch wells, the difference in arsenic levels is unlikely to make much of a difference.

The above graphs are fairly effective in giving us an idea of the effect of an isolated predictor on the probability that the binary response is 1, but if we wanted a more numeric way to approach this question, we could examine average predictive differences in high and low values for our two predictors. The R code below does this for different values of the distance variable–printing out the predictive difference between different values of distance. The arsenic level in this calculation is held constant at its mean.

```
D0_prob <- invlogit(wf[1]+(wf[2]*0) +(wf[3] * mean_AR) + (wf[4] * 0 *mean_AR))
D100_prob <- invlogit(wf[1]+(wf[2]*100) +(wf[3] * mean_AR) + (wf[4] * 100 *mean_AR))
D200_prob <- invlogit(wf[1]+(wf[2]*200) +(wf[3] * mean_AR) + (wf[4] * 200 *mean_AR))

avg_D_diff1 <- (D100_prob - D0_prob) / 100
avg_D_diff2 <- (D200_prob - D100_prob) / 100

print("Comparison of dist = 0 to dist = 100, with arsenic held constant:")
```

```
## [1] "Comparison of dist = 0 to dist = 100, with arsenic held constant:"
```

```
print(avg_D_diff1)
```

```
##  (Intercept)
## -0.002279584
```

```
print("Comparison of dist = 100 to dist = 200, with arsenic held constant:")
```

```
## [1] "Comparison of dist = 100 to dist = 200, with arsenic held constant:"
```

```
print(avg_D_diff2)
```

```
##  (Intercept)
## -0.002131849
```

The above values give us an estimate for the effect of increasing the distance by 1 meter on the probability of a household switching wells. The effect is calculated based on the difference of two 100 meter ranges in the data: the first being from 0m to 100m and the second being from 100m to 200m. These values will always be something different due to the pseudo-randomness involved in Bayesian regression, but when I ran this code, I have found that the effects are always negative and very close to each other with the 0 to 100m meter effect being slightly larger than the 100 to 200m effect.

This tells us that in the shorter distance range, a change in 1 meter has a larger effect on the probability of switching wells than a change in 1 meter in the longer distance range. This makes sense on some level because the difference in 230 and 231 meters is not as big of a change proportionally than the difference between 10 and 11 meters. In both ranges, longer distances reduce the probability of switching on average–which again, has an intuitive explanation.

We can then do the same kind of calculation again, but this time the average predictive differences between high and low values of arsenic levels in the wells studied with distance held constant at its mean.

```
A05_prob <- invlogit(wf[1]+(wf[2]*mean_D) +(wf[3] * log(0.5)) + (wf[4] * mean_D *log(0.5)))
A10_prob <- invlogit(wf[1]+(wf[2]*mean_D) +(wf[3] * log(1.0)) + (wf[4] * mean_D * log(1.0)))
A20_prob <- invlogit(wf[1]+(wf[2]*mean_D) +(wf[3] * log(2.0)) + (wf[4] * mean_D * log(2.0)))

avg_A_diff1 <- (A10_prob - A05_prob) / 0.5
avg_A_diff2 <- (A20_prob - A10_prob) / 1.0

print("A comparison of arsenic = 0.5 to arsenic = 1.0, with dist held constant.")
```

```
## [1] "A comparison of arsenic = 0.5 to arsenic = 1.0, with dist held constant."
```

```
print(avg_A_diff1)
```

```
## (Intercept)
##   0.2961247
```

```
print("A comparison of arsenic = 1.0 to arsenic = 2.0, with dist held constant.")
```

```
## [1] "A comparison of arsenic = 1.0 to arsenic = 2.0, with dist held constant."
```

```
print(avg_A_diff2)
```

```
## (Intercept)
##   0.1451905
```

Just like the values calculated for distance, the numbers above will be different every time this document is knit. However in my observations, I have generally seen that the predictive difference in switch probability between the arsenic values of 0.5 and 1.0 is greater than the predictive difference in the switch probability between 1.0 and 2.0. Even though arsenic levels are on a very different scale than distance, these values are still telling us the same kind of statistic as above: an estimate for the effect of increasing the arsenic levels by 1.0 on the probability of a household switching wells (with all other predictors held constant at their mean).

These statistics seem to suggest that increases in arsenic are more likely to encourage households to switch when they are in the lower range for arsenic than the higher range. This seems somewhat counter intuitive because of the proportional risk associated with arsenic exposure (it seems like differences in higher ranges would lead to more noticeable differences in the poison's effect). However, this could be attributed to the lower variability of response probability at higher values of arsenic—that is to say that if a household was not deterred by a high arsenic level, then they are unlikely to be deterred by a slightly higher one. This could be because at high arsenic levels, the households who have the means to switch have already switched and the ones that do not have the means are unable to switch, so the switch probability is less elastic.

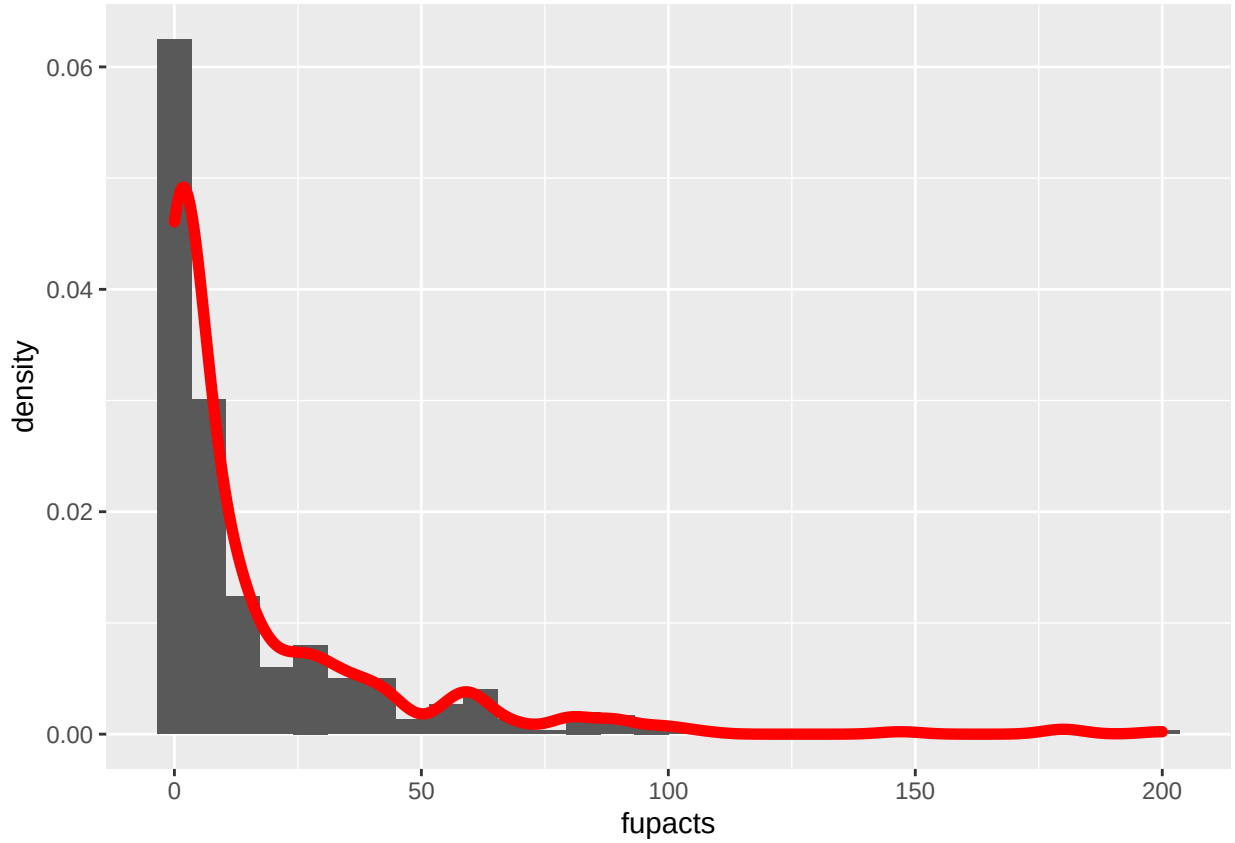
15.1: Poisson and negative binomial regression

First, let's write a chunk to get the risky behavior data imported and formatted in a way we can use.

```
myfile <- "https://pages.uoregon.edu/dlevin/DATA/ROS-Examples-master/RiskyBehavior/data/risky.csv"
risky <- na.omit(read.csv(myfile, header=T))
# Modifying the variables in the dataframe a bit to help us with our regression analysis below
risky$fupacts <- round(risky$fupacts)
risky$trtmt <- factor(risky$women_alone):factor(risky$couples)
```

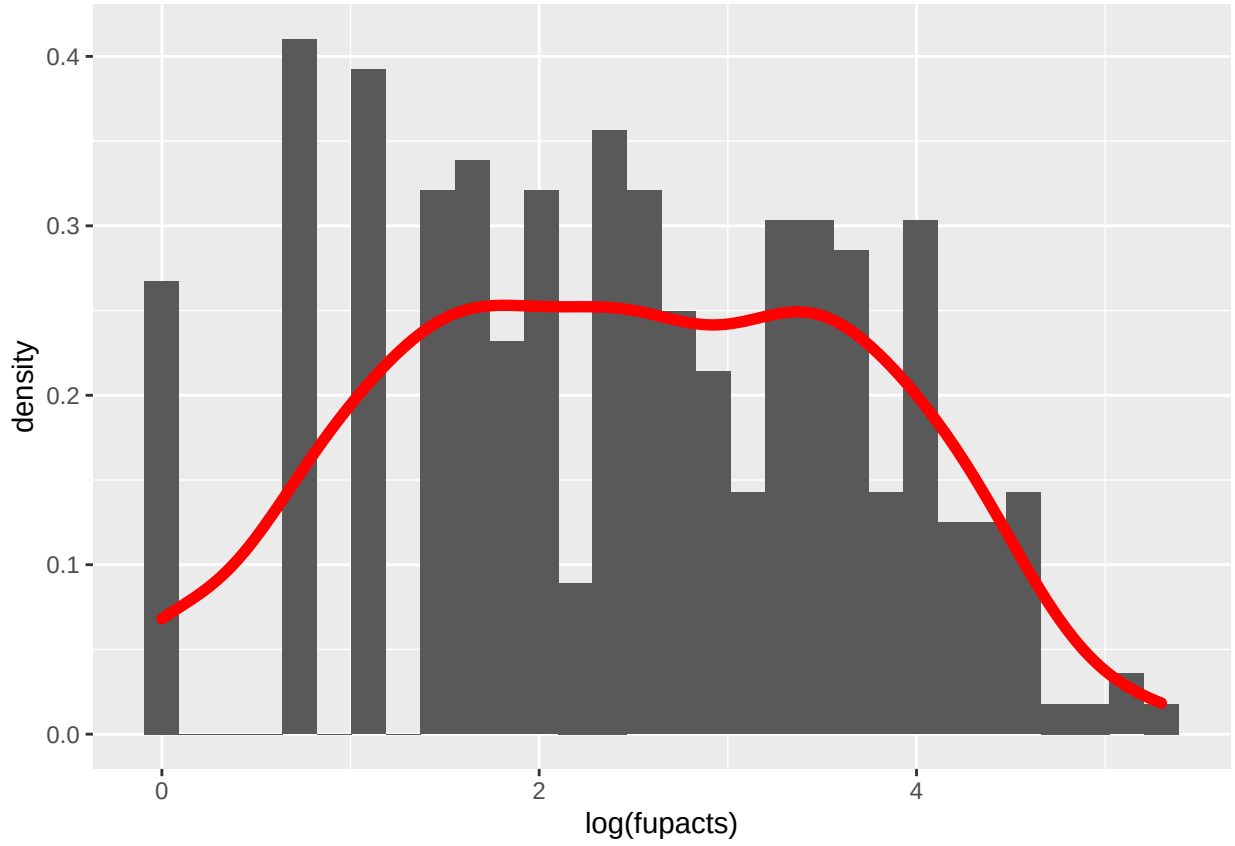
Before we fit any regression models to our data, let's take a look at the response variable we're interested in—which here is a count of the number of unprotected sex acts in a 3 month period after the study's intervention. We want to start by just looking at a histogram of the values that this variable takes on to see if any transformations could be useful in generating a better fit for the data.

```
ggplot(risky, aes(x=fupacts)) +  geom_histogram(aes(y = ..density..)) + geom_density(color = "red", siz
```



Note that the distribution of future risky sex acts has a large mass at the lower values and has a long tail stretching out into values of higher magnitudes. This suggests to us that a log transformation may improve any regression analysis we attempt to do. We can see if this transformation results in a more normal distribution of the response by making a histogram of the transformed variable below.

```
ggplot(risky, aes(x=log(fupacts))) +  geom_histogram(aes(y = ..density..)) + geom_density(color = "red"
```



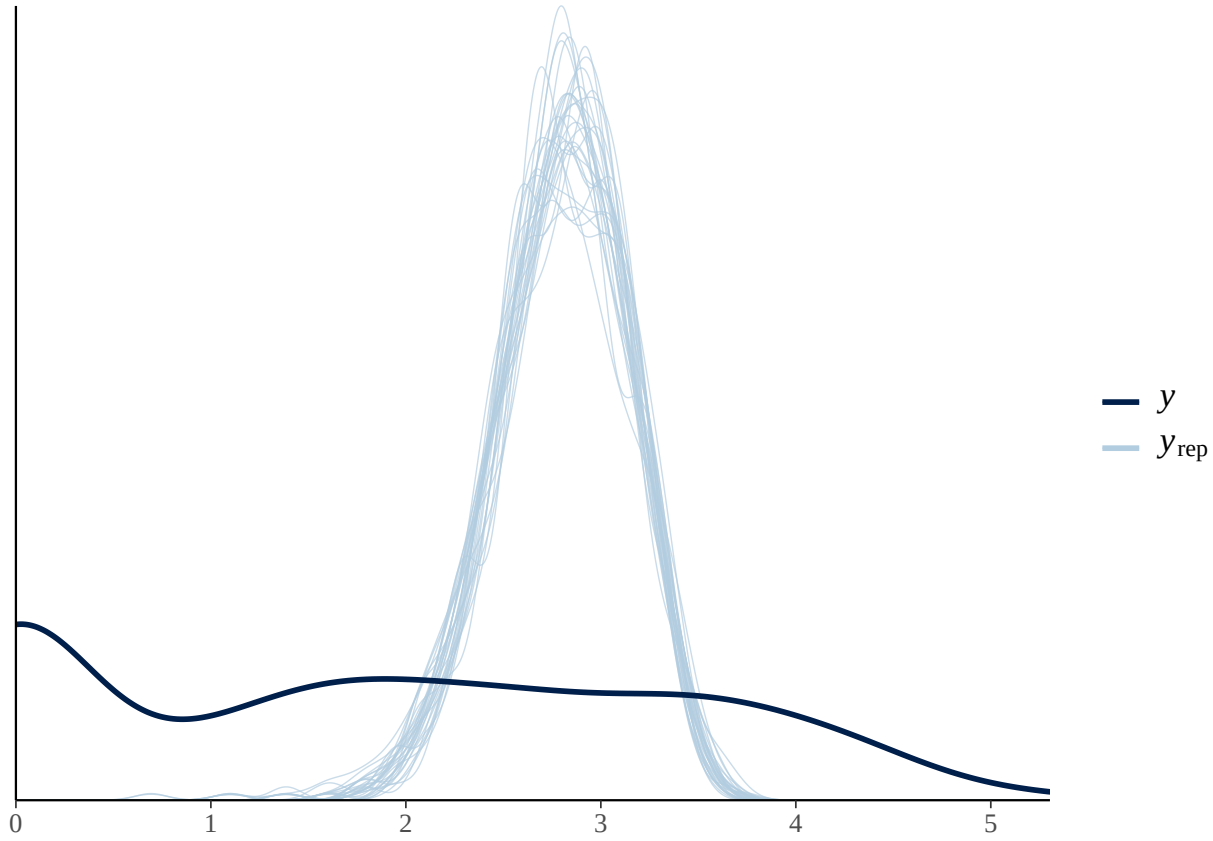
The histogram above displays a much more normal distribution of values and tells us that this is a reasonable transformation to make if we want to better fit Gaussian regression models to the trends seen in this data.

Now the first model we want to fit to this data is a Poisson regression that has log(fupacts) as a response and treatment received as the only predictor. The R code below generates such a model using a Bayesian approach built into stan_glm.

```
fitr_1 <- stan_glm(fupacts~trtmt, data=risky, refresh=0, family=poisson())
```

To get a an idea of the fit of this model, we can take a look at a handful (30) of the simulated draws from the posterior distribution of the coefficients and use these simulations to generate predictions for response values based on predictor values seen in the data frame. We can then make a smoothed histogram for each of these sets of predicted responses and overlay the histograms on a smoothed histogram of the observed responses.

```
pp <- posterior_predict(fitr_1)
ppc_dens_overlay(log(risky$fupacts + 1), log(pp[1:30,]+1))
```



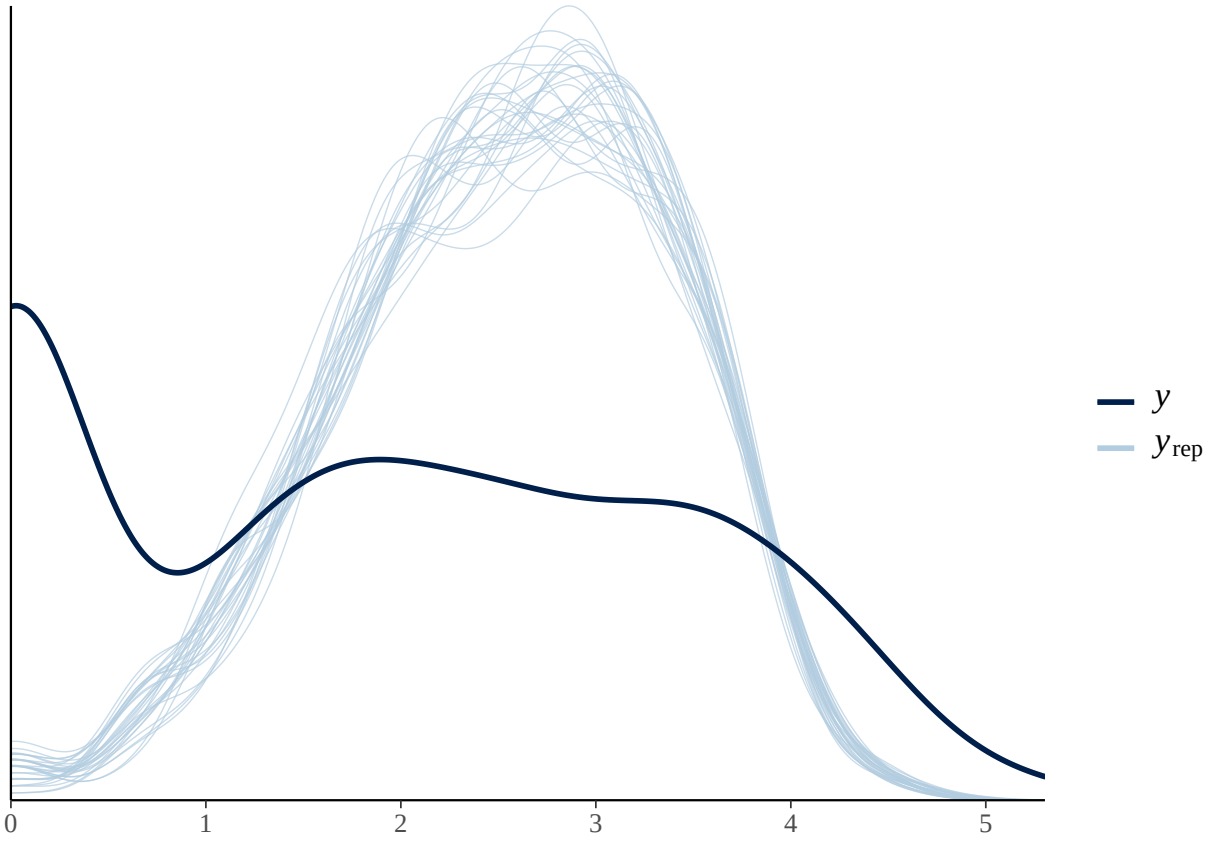
The above graph is a clear example of over-dispersion—we can see this by noticing that the predicted responses have much more mass around the average values of log(fupacts) while the observed responses are much more dispersed. This tells us that there are some flaws with our regression and we should try another fit.

We can see if the regression fit is improved by adding more predictors; specifically, we can generate a model that adds past risky sexual acts (before intervention) and the HIV status of the participants as predictors for our response variable. The R code below uses a Bayesian approach to generate such a regression model.

```
fitr_2 <- stan_glm(fupacts~log(bupacts+1)+trtmt+bs_hiv, data=risky, refresh=0, family=poisson())
```

Now that we have another regression model, we can do the same sort of visual analysis that we did above with the model that had only treatment as a predictor. The R code below generates such a graphic.

```
pp2 <- posterior_predict(fitr_2)
ppc_dens_overlay(log(risky$fupacts + 1), log(pp2[1:30,]+1))
```



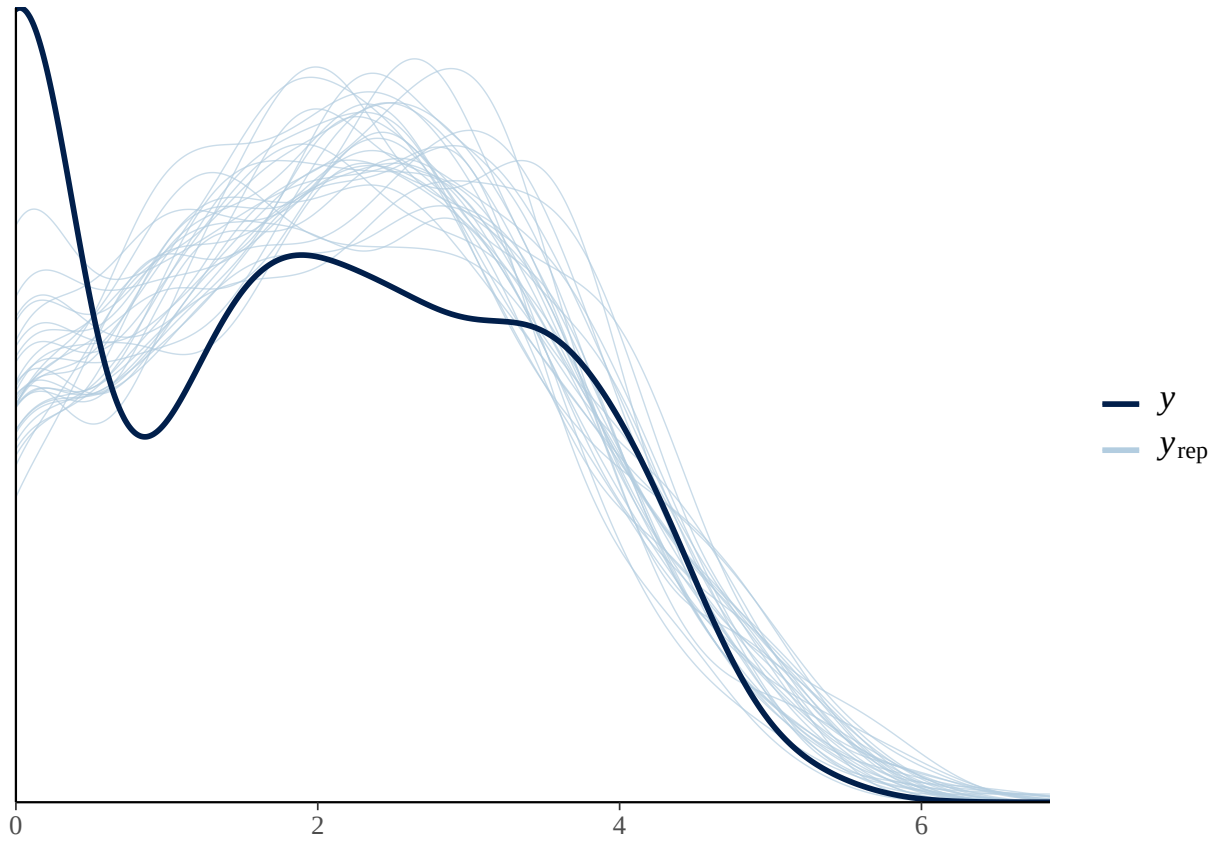
The above graph once again shows signs of over-dispersion. Specifically, we can note that, like before, there is significantly more predicted mass near the average of the response values and the observed values are much more dispersed. This difference is not as pronounced as it was in the previous chart, but it is still enough to conclude that this Poisson regression fit is far from optimal.

The two cases of over-dispersion in the smaller and the larger Poisson regression models suggests that this problem may be due to the type of regression rather than the number of predictors or different variable transformations. To correct this issue, we can use negative binomial regression which is a larger model with the addition of a "precision" coefficient that can account for over-dispersion. The R code below generates a Bayesian regression model using this new framework.

```
fitr_3 <- stan_glm(fupacts~log(bupacts+1)+trtmt+bs_hiv, data=risky, refresh=0, family=neg_binomial_2())
```

We can do the same kind of visual analysis we did for the two previous regression models in order to get an idea if the over-dispersion problem has been corrected.

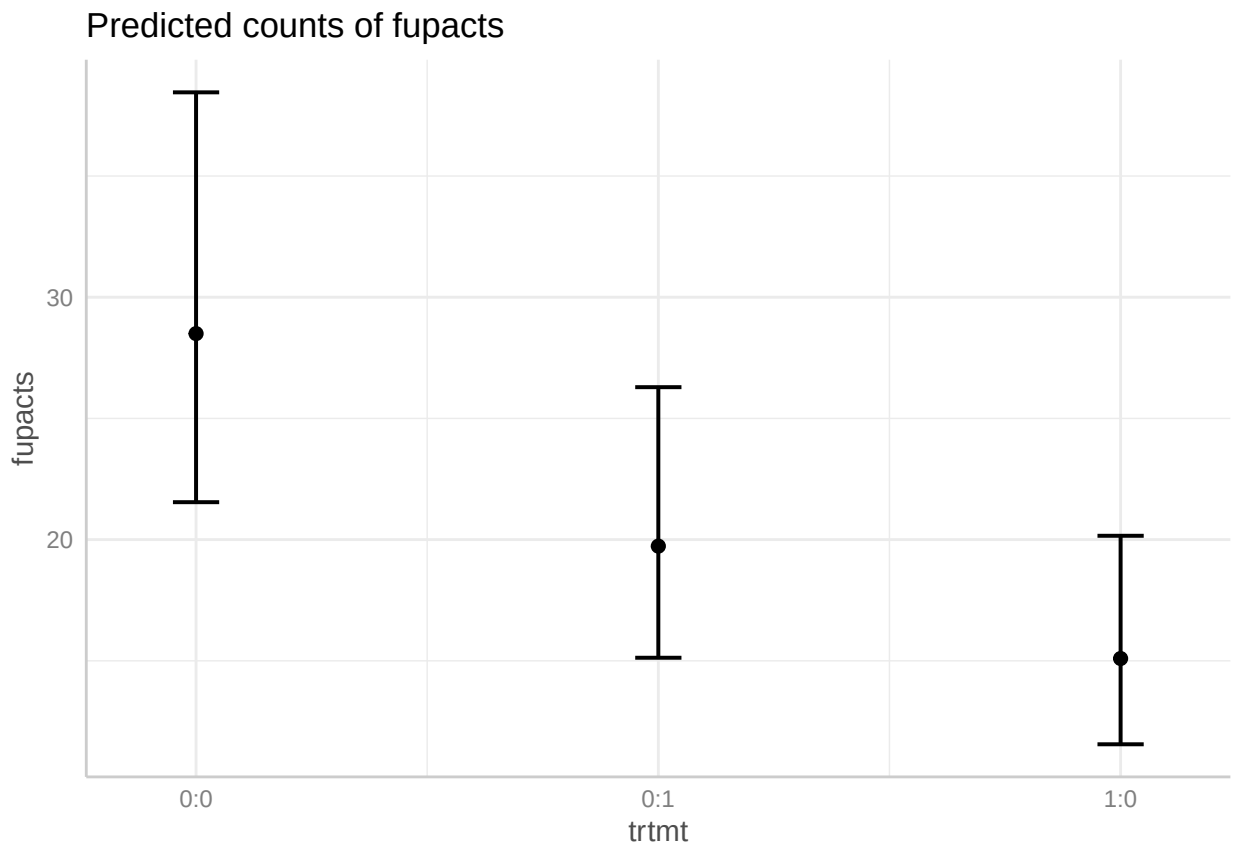
```
pp3 <- posterior_predict(fitr_3)
ppc_dens_overlay(log(risky$fupacts + 1), log(pp3[1:30,]))
```



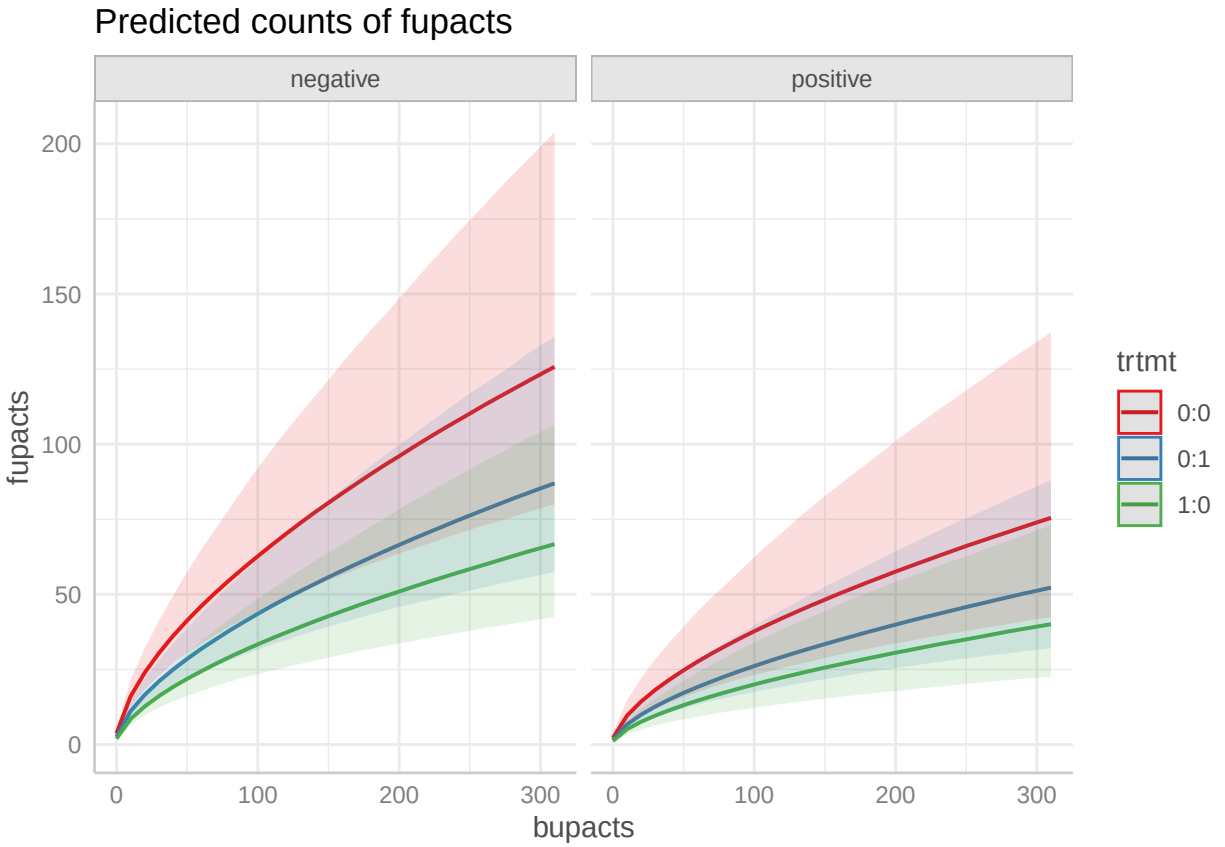
From the above graph, we can see that the over-dispersion of the observed responses in previous graphs is essentially accounted for in the predicted responses for this model. The smoothed histograms from the negative binomial regression model are not a perfect fit: they put extra mass at the average response values and not enough at the low values; however, the fit is definitely an improvement from what we saw in either Poisson regression model.

Now that we have a regression model that we can say fits the trends of our data reasonably well, we can use the predicted expected responses generated by this model to make a conclusion about the effectiveness of the treatment provided. To make such a conclusion, we can examine an effects plot which will visually give us an idea of the different trends in the response variable depending on different categories of predictors. Two types of effects plots are generated below for our negative binomial regression.

```
# First we can generate an effects plot that looks only at treatment groups
plot(ggpredict(fitr_3,terms=c("trtmt")))
```




```
# Then we can generate one that takes into account all the different categories for the different predi
plot(ggpredict(fitr_3,terms=c("bupacts", "trtmt", "bs_hiv")))
```



The above plots tell us that the treatment does appear to have some sort of effect on the the number of future risky sexual acts—this is seen most clearly in the first effects plot which shows the two treatment groups having a mean predicted future risky acts outside of the error bars of the no treatment group. However, there is overlap between the error bars of the three groups (besides between the women alone treatment and the no treatment group), which may suggest that the evidence of an effective treatment is not as strong as some would like.

The second effects plot does not do as good of a job visually displaying what the model predicts about the effectiveness of the treatment, but it does give us some extra insight on how the other predictors affect the response. Specifically, we see that as past acts increase, we expect future acts to increase roughly logarithmically along with there being increasing error bars (probably explained by fewer data points in the upper values of number of past acts). Also, we see that people who are HIV positive generally have a lower number of future risky sexual acts across all of the treatment groups.

The conclusions we have made above appear pretty reasonable when we only look at the variables we have selected to be a part of our regression model; however, we run into some problems when we consider the sex variable as well. Notice that in the data, there are a few datapoints where the sex of the participant is male, but the treatment was received by the woman alone.

This could possibly lead to a violation of the validity assumption of regression analysis which says that data analyzed should map directly to the question we are trying to answer. In particular, if we are trying to answer the question: "Is the treatment effective at reducing future risky sexual acts?" It may stand to reason that we only want to look at data from participants who received the treatment personally (and not just their significant other). Something could be said about the ability of a partner's treatment to positively affect the other partner, but the presence of these data points still may lead to some doubt in the conclusions we reach.

Count Data Regression

First we can write a little bit of R code to import the data into our R workspace.

```
myfile <- "https://pages.uoregon.edu/dlevin/DATA/cntdata.csv"
cnt <- read.csv(myfile, header = T)
```

Then we can split the data into a training set and a validation set in order to do some rudimentary cross validation.

```
ind <- sample(1:200,66,replace=FALSE)
cnt_test <- cnt[-ind,]
cnt_train <- cnt[ind,]
```

Now that we have our training data partitioned, we can fit two models to this subset of the datapoints using the Bayesian regression approach built into stan_glm. In the first model, we will use Poisson regression to model the response n using x as our sole predictor and for the second we'll do the same, but with a negative binomial regression instead of a Poisson.

```
cnt_fit1 <- stan_glm(n~x, family=poisson(), data=cnt_train, refresh=0)
cnt_fit2 <- stan_glm(n~x, family=neg_binomial_2(), data=cnt_train, refresh=0)
```

We can use cross validation to see which of these models have a better estimated predictive performance by estimating the mean squared error. This can be done using our validation data set to find the error in the predictions and then squaring it and summing the results, which estimates predictive performance of the models.

```
preds1 <- posterior_linpred(cnt_fit1, newdata=cnt_test)
preds2 <- posterior_linpred(cnt_fit2, newdata=cnt_test)
rmsef <- function(x,y){
  sqrt(mean((x-y)^2))
}
pois_rmse <- apply(preds1,1,rmsef,y=cnt$n)
bino_rmse <- apply(preds2,1,rmsef,y=cnt$n)
print("Estimated MSE for the Poisson regression model"); print(mean(pois_rmse));
```

```
## [1] "Estimated MSE for the Poisson regression model"
```

```
## [1] 17.62799
```

```
print("Estimated MSE for the Negative Binomial regression model"); print(mean(bino_rmse))
```

```
## [1] "Estimated MSE for the Negative Binomial regression model"
```

```
## [1] 17.62533
```

The model with the smaller MSE value is the one we conclude has better predictive performance according to this test. The values above will be different every time due to the stochastic processes involved in the sample function and Bayesian regression, but in the handful of times that I have ran this code, I have found that the estimated MSE for the two models is essentially the same across all of my trials.

This seems to indicate that the models have similar predictive performance, which normally would tell us we could favor either regression in explaining the trends in this data. However, in this case one of our models is a smaller model that is contained within the larger model (i.e. the Negative Binomial model is essentially just a Poisson model with an extra parameter for "precision"). Because of this, we want to favor the smaller model as that is less likely to lead to over-fitting to the data provided. Therefore, we conclude that the Poisson regression model is best suited to be fit to this data.