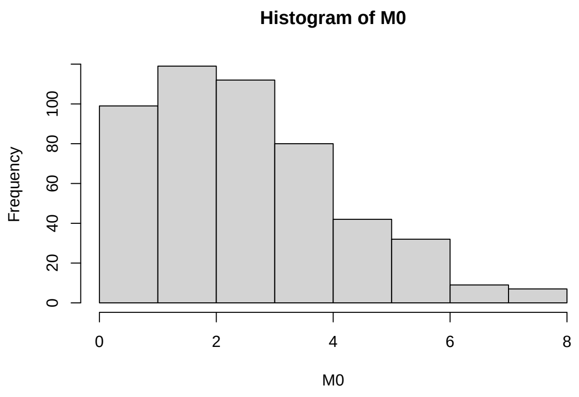
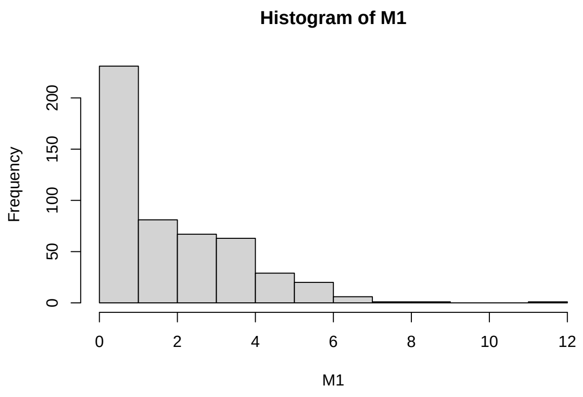
Math 445 HW 4

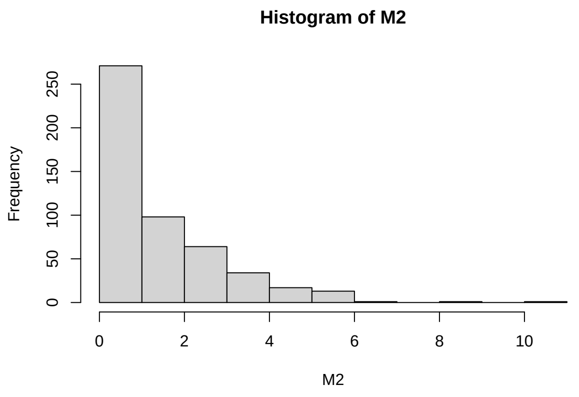
Erick Castillo

3/6/2021

Question 1:







From the above output it is apparent that the Poisson distribution has the least amount of zeros. It isn't immediately obvious unless the code sort(M0) is executed, but there's only 12 zeros in the Poisson random sample.

The following two graphs are slightly difficult to compare as there are a large collection of values on the left end of the histogram; however using sort(M1) and sort(M2) will show that the ZIP random sample (M1) has a larger collection of zeros than that of the negative binomial random sample (M2). This is expected.

Problem 2:

```
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
##
## Call:
## zeroinfl(formula = y ~ x1 + x2 | x1 + x2)
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.1871 -0.8188 -0.6468  1.0108  2.7682
##
## Count model coefficients (poisson with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.06769    0.03473   59.53   <2e-16 ***
## x1           2.93668    0.04015   73.14   <2e-16 ***
## x2           0.45473    0.02830   16.07   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.8199     0.2538   3.230 0.001238 **
## x1           -1.2068     0.3352  -3.600 0.000318 ***
## x2            0.3467     0.3200   1.083 0.278706
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 1
## Log-likelihood: -1037 on 6 Df
```

```
## [1] 2085.162
```

A. The above is the output of a ZIP model fit with x1 and x2 as links to both lambda and pi. It is clear that the coefficients estimated in this model are all significant at a 1% level, and that they are all very close to the true values set prior to running this model. The only value that is not significant is the x2 coefficient found in the pi link. This should be the case.

```
##
## Call:
## zeroinfl(formula = y ~ x1 + x2 | x1)
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.0953 -0.8156 -0.6413  1.0089  2.8843
##
## Count model coefficients (poisson with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.06769    0.03473   59.53   <2e-16 ***
## x1           2.93668    0.04015   73.14   <2e-16 ***
## x2           0.45473    0.02830   16.07   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.9849     0.2046   4.814 1.48e-06 ***
## x1           -1.1836     0.3341  -3.543 0.000396 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 1
## Log-likelihood: -1037 on 5 Df
```

```
## [1] 2084.338
```

B. The above is the output of the ZIP model being performed with x1 and x2 present in the lambda link, and only x1 being present in the pi link. Notice that all the values are very significant in the model, and that the coefficients are very close to their true values.

```
##
## Call:
## zeroinfl(formula = y ~ x1 | x1)
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.0967 -0.8166 -0.6416  0.9965  2.6102
##
## Count model coefficients (poisson with log link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.28998    0.03145   72.81   <2e-16 ***
## x1           2.96382    0.04002   74.06   <2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.9849     0.2046   4.814 1.48e-06 ***
## x1           -1.1836     0.3341  -3.543 0.000396 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 1
## Log-likelihood: -1168 on 4 Df
```

```
## [1] 2343.315
```

C. This is the final ZIP model asked to be performed on the data. The values in the pi link of the model are very close to the values declared earlier. Notice that the intercept and x1 for the lambda link have coefficients that are very close to the true values; but because x2 is missing from the link, the values of both the coefficient and x1 have been inflated.

```
##
## Call:
## glm(formula = y ~ x1 + x2, family = poisson())
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -14.865   -6.227   -3.183    5.124   10.161
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.83972    0.03439   24.41   <2e-16 ***
## x1           3.63658    0.03883   93.66   <2e-16 ***
## x2           0.37472    0.02882   13.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 34772  on 499  degrees of freedom
## Residual deviance: 23009  on 497  degrees of freedom
## AIC: 24224
##
## Number of Fisher Scoring iterations: 6
```

D. In this case I am fitting a Poisson model onto the y data that was generated using the rzippois function defined earlier. I used x1 and x2 as predictors, and they are both extremely significant in the model; however, notice that the major issue is the gigantic AIC. I calculated the AIC of the prior ZIP model–which is output below their respective model output, and neither of the three models exceeded an AIC of 3000. This model's AIC is huge in comparison.

```
library(MASS)
library(foreign)
mod5 = glm.nb(y~x1+x2)
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
## Warning in sqrt(1/i): NaNs produced
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
## Warning in sqrt(1/i): NaNs produced
```

```
#summary(mod5)
AIC(mod5)
```

```
## [1] 24224.36
```

I was able to run a negative binomial regression, but the moment I utilize the summary() function, my RMD file refuses to knit because of a prettyNum error. However, I was able to extract an AIC score, and it was bigger than the Poisson model's AIC. Clearly the Negative Binomial model and the Poisson model do not do a great job at explaining a dataset that has ZIP origins.

Problem 3:

A. Below is the output for the kmeans() operation on the desired values from the titanic data frame.

```
library(ggplot2)

km = kmeans(pred.cols, centers = 2, nstart = 200)
km$cluster <- as.factor(km$cluster)

try1 = ifelse(km$cluster == 1, 1, 0)
try2 = ifelse(km$cluster == 1, 0, 1)

sum(abs(boat$Survived-try1))
```

```
## [1] 266
```

```
sum(abs(boat$Survived-try2))
```

```
## [1] 446
```

Notice that the kmeans() clustering accurately categorized 446 of the 712 values in the Survived column

B. Now I will attempt to use hclust() on the data to see if there is a comparative difference when compared to kmeans().

```
hclustpred = hclust(dist(pred.cols))
hclust = cutree(hclustpred, k = 2)

try1 = ifelse(hclust == 1, 1, 0)
try2 = ifelse(hclust == 1, 0, 1)

sum(abs(boat$Survived-try1))
```

```
## [1] 427
```

```
sum(abs(boat$Survived-try2))
```

```
## [1] 285
```

Notice that this comparison was slightly worse than the kmeans grouping. It only correctly categorized 427 of the values in the survived column of the data set.

C. Now I will use divisive hierarchical clustering.

```
#library(tidyverse)
library(cluster)
#library(factoextra)
#library(dendextend)

diana.clust = diana(pred.cols)
dclust <- cutree(diana.clust, k=2)
```

```r
try1 = ifelse(dclust == 1, 1, 0)
try2 = ifelse(dclust == 1, 0, 1)

sum(abs(boat$Survived-try1))
```

```
## [1] 427
```

```r
sum(abs(boat$Survived-try2))
```

```
## [1] 285
```

Comparing the divisive and agglomerative method shows that the results are the same, that is the divisive method was able to correctly categorize 427 of the values in the Survived column. In this case, kmeans is better at clustering data together than the other two methods as it got the most correct values.