Math 651 Final Project

Mary Peng, Hillary Dunn, and Max Kearns

December 17, 2018

1 Abstract

2 Introduction

Every four summers, the Olympic Games become the center of the world's attention, as elite athletes seek honor for both themselves and for their countries. Many countries associate tremendous national pride with their medal counts, since a nation's athletic competence also projects its soft power.

Given the fierce competition and high profile nature of medal counts, one may wonder what factors influence the number of medals that a country wins at the summer Olympics. Certainly countries with the largest economies and populations, such as the United States and China, commonly dominate the top of the billboard. However, Azerbaijan, which ranks 91st in population and 72nd in total GDP, also ranked in the top 20 countries by total medal count in the 2016 Summer Olympics.

Our team will develop multiple regression models using predictors like GDP per Capita, Population, whether a country is a host country, and whether a country is a former soviet or communist state. We will use these factors to predict medal count, and examine how these each of these factors influences medal count. We will build the model on countries' total medal count for the 1996, 2004, 2008, and 2012 Olympics, and then project the model onto the 2016 Olympics to understand the accuracy of the model's predictions.

3 Methods and Materials

3.1 Data

3.1.1 Dependent Variable

We model on the total number of medal counts, by participating country and year during which a Summer Olympics occurred. The dependent variable data come from a website called *www.medalspercapita.com*. For any given Summer Olympics, this website only lists the 80 - 90 countries that have won at least one medal, out of the 200 or so nations that participate in each summer Olympics.

Table 1 and Figure 1 both illustrate that the medal counts exhibit right-skewed distribution. Among countries that have won at least one medal, the median medal count is 5, while the mean medal count is 12. The skewed distribution suggests the need for a log transformation (in the case of a linear model), which can also ensure that the predictions are positive numbers. Moreover, given we are modeling on count data, where large counts rarely occur, a Poisson model may be more appropriate than linear regression.

3.1.2 Predictor Variables

Based on existing literature (Bernard and Busse 2006, Goldman Sachs 2016, Bian 2005), potentially significant predictors of medal count include size of population, GDP per capita, whether or not a country has hosted or will host a summer Olympics, and whether or not a country is or was a command economy.

X_1 : *Population*: This variable indicates population of people in the country in the associated year. Figure 2 shows a histogram of the population data. The left chart illustrates the right-skewed distribution of the

original data (with China and India representing the far right). The right chart shows that applying a natural log transformation reduces the skewness and the influence of extreme X variable values on the model.

$X_2 : GDP/Capita$: This variable indicates GDP per capita in the associated year. These data show a similar distribution to the population variable, so we also applied the natural log transformation.

$X_3 : Host(1/0)$: This is a binary predictor variable with value equal to 1 if a country has hosted a summer Olympics in the 8 years prior to the Olympics in consideration, or will host in the next 8 years. For example, Greece, which hosted the 2004 Olympics, will have value = 1 for the 1996, 2000, 2004, 2008, and 2012 Olympics. 25 observations are classified as a host by this measure.

$X_4 : CommandEconomy(1/0)$: This binary predictor variable indicates whether the country was once a member of the Soviet Union or Yugoslavia, or ruled by a Communist party. We assembled a list of formerly or currently Soviet, Yugoslavian, or Communist countries using *www.worldatlast.com*, *Wikepedia*'s list of communist parties, and *www.sporcle.com*. 111 data points are classified as former Soviet Union or Communist.

We generate a correlation matrix (Table 2) and paired scatterplot (Figure 3) to illustrate the relationships between the variables, especially any multicollinearity.

The scatter plot matrix in Figure 3 shows positive correlation between count of medals and population size. We do not observe strong linear relationship between medal count and GDP per capita, whether or not a country has hosted or will host, and whether a country used to be part of a Command Economy. The correlation matrix corroborates the finding that population size has the strongest linear correlation with medal count (correlation = 0.48), and Command Economy (Y/N) has the weakest linear correlation (correlation = 0.09).

This correlation matrix in Table 3 also suggests that collinearity between variables will not be a significant problem. The most significant correlation between predictor variables is between the log of the GDP per capita and the indicator of Communism or Soviet Union inclusion variable. However, the correlation coefficient is only -0.2989. This suggests that multicollinearity should not be much of an issue.

3.2 Model Building and Selection

We will consider two types of models: a linear model and a Poisson model, given the count nature of the data.

3.2.1 Linear Model

According to the selected model diagnostics shown in Table 3, the model with the smallest C_p statistic, largest adjusted R^2 value, smallest AIC and smallest PRESS value is the model that includes all four predictor variables. We constructed a linear model (shown below), using the natural log to transform the count data. We decided to transform the count data to get the response variable to look more normally distributed so that it would follow the linear regression assumption that the response variable needs to be approximately normally distributed.

$$X_1 = \ln(\text{population})$$

$$X_2 = \ln(\text{gdp/capita})$$

$$X_3 = \text{host}$$

$$X_4 = \text{Soviet Country or Communist}$$

$$Y = \ln(\text{Count})$$

$$\hat{Y} = -9.63058 + 0.44275X_1 + 0.40830X_2 + 0.86726X_3 + 1.03281X_4$$

According to the Normal Q-Q plot (Figure X), it is reasonable to assume the residuals follow a normal distribution. However, the Residuals vs Fitted plot indicates possible heteroskedasticity or non-constant variance of the errors. The Breusch-Pagan test will help us determine if heteroskedasticity is affecting the model.

Assuming $Var(\epsilon_i) = \sigma_i^2$ such that $\log_e \sigma_i^2 = \gamma_0 + \gamma_1 X_i$:

$$H_0 : \gamma_1, \text{vs. } H_a : \gamma_1 \neq 0$$

At significance level $\alpha = 0.05$, if the p-value of the Breusch-Pagan test is less than α then we reject H_0 and conclude H_a, otherwise we fail to reject H_0. Rejecting H_0 in favor of H_a means that we conclude that the variance is non-constant.

The p-value = 0.001587 $< \alpha$, therefore we conclude that our variance is not constant.

$$H_0 : \text{Sample comes from a N}(\mu, \sigma^2) \text{ distribution}$$

$$H_a : \text{Sample does not come from a N}(\mu, \sigma^2) \text{ distribution}$$

At significance level $\alpha = 0.05$, if the p-value is less than α then we accept H_a and reject H_0, otherwise we fail to reject H_0.

The p-value= 0.2332 $> \alpha$, thus we fail to reject H_0. Therefore, it is reasonable to assume that the error terms are distributed normally.

We attempt to correct for non-constant variance and outliers by using robust linear regression, with Huber and Bisquare weights. As shown in table 6, the coefficients generated through robust linear regression, using Bisquare weights, fall within +/- 5% of the corresponding OLS regressions. The robust regression's standard errors are slightly larger than those of the OLS regression. We find similar results when using Huber weights. Re-running the Breusch-Pagan test results in a p-value of 0.002, which means we would still reject the null hypothesis of homoskedasticity at $\alpha = 0.05$ level, and conclude that the robust regression still exhibits non-constant variance.

3.2.1.1 Influential Cases

Appendix B summarizes the influential cases in our model. We can see from the list that highly populous countries, such as the United States, China, and India are influential on our model. Additionally, many of the influential cases are identified as being host nations. We wanted to identify these influential cases for the sake of full analysis, but we are choosing to not delete them from the model because they are important data points. It would not be feasible to delete 37 points and high populous countries are important to the analysis.

3.2.2 Generalized Linear Model

The first choice for a glm is typically a Poisson model, which may provide a reasonable model for these data. This generalized model was created with the same variables as the linear model; population, GDP per capita, the host dummy variable, and the communist/soviet dummy variable.

$$\ln(E(Y_i|X_i)) = \ln(\lambda_i) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4$$

$$\ln(E(Y_i|X_i)) = \ln(\lambda_i) = -11.28882 + 0.50479 X_1 + 0.52117 X_2 + 0.31070 X_3 + 1.02332 X_4$$

Like the linear model, the population and GDP per capita were log-transformed, but in this model it is unnecessary to transform the medal count. The full poisson model showed highly significant parameters, but the dispersion of this model was much greater than 1 (disp. = 6.621). This is a violation of the assumptions of a Poisson regression model, so a negative binomial model would likely provide a better fit for these data.

A negative binomial regression model assumes that dispersion is greater than 1, which is consistent with these data. Therefore, it allows for more accurate tests of the parameters. Among the negative binomial models, the best model again proved to be the full model, after a comparison of AIC between all subsets of variables (Table X).

$$\ln(E(Y_i|X_i)) = \ln(\lambda_i) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4$$
$$\ln(E(Y_i|X_i)) = \ln(\lambda_i) = -10.16577 + 0.49848 X_1 + 0.40239 X_2 + 0.69267 X_3 + 1.03376 X_4$$

Independence among observations constitutes a key assumption for the Poisson and negative binomial regressions. Since our model uses panel data, observations over time for the same country will exhibit serial autocorrelation, thereby violating the independence assumption. Moreover, the medal counts among countries within a given year are not completely independent, because of the fixed total number of medals awarded in a single Olympic year.

To address the above non-independence, we tried adding year and country fixed effects to the negative binomial regression. The models take the form:

$$\ln(Y_i t) = \beta_0 + \beta_1 X_1 it + \beta_2 X_2 it + \beta_3 X_3 it + \beta_4 X_4 it + d_t$$

$$\ln(Y_i t) = \beta_0 + \beta_1 X_1 it + \beta_2 X_2 it + \beta_3 X_3 it + \beta_4 X_4 it + v_i$$

where d_i and v_i represent dummies for year t and country i, respectively. The year fixed effect accounts for changing number of sports and number of country participants. The country fixed effect accounts for unobserved factors that vary little over time, such as a country's investment in national sports teams, cultural attitude toward the Olympics, etc.

4 Results

4.1 Comparison of Models

Table 6 compares the coefficients across different model specifications. We observe the following:

- Compared to the OLS linear model, the robust linear model produces slightly larger standard errors, but similar coefficient magnitudes. This finding suggests that outliers do not substantially influence the OLS linear model fit.
- As expected from the dependent variable's over-dispersion, the Poisson regression underestimates the standard error, compared to the other 4 model specifications.
- Negative binomial produces similar coefficient and standard error estimates as OLS and robust linear regressions.

Figure 5 compares the Actual medal count values vs. the values fit by the OLS linear regression and negative binomial regressions. The blue dots plot the predicted vs. actual values for all observations. The red line indicates perfect fit. We observe that the negative binomial regression appears to have better fit of the data.

Interestingly, table 6 shows that the OLS linear regression has better model fit, because it has lower AIC and MSE.

4.2 2016 Out-of-sample Projection:

As part of the data collection process, information for the year 2016 was also researched, but left out of the original analysis. Therefore, because the data was not part of the training set, we can use it as a testing set. The medal count data for 2016 is similarly distributed to the overall training data, so testing on it is a feasible plan.

To determine the accuracy of each of our models we determined 95% prediction intervals for each of the 81 points in the 2016 medal count data. Then we calculated the ratio of data points that had their actual counts within their respective prediction intervals to total number of data points. For our estimated linear regression function,

$$\hat{Y} = -9.63058 + 0.44275X_1 + 0.40830X_2 + 0.86726X_3 + 1.03281X_4$$

74 out of 81 points had medal counts within their respective 95% prediction intervals.

The same analysis on the negative binomial model showed that a similar number of the 95% predictions intervals for 2016 fell around their observed intervals.

$$\ln(E(Y_i|X_i)) = \ln(\lambda_i) = -10.16577 + 0.49848X_1 + 0.40239X_2 + 0.69267X_3 + 1.03376X_4$$

5 Discussion and Conclusions

6 Bibliography

7 Appendix A

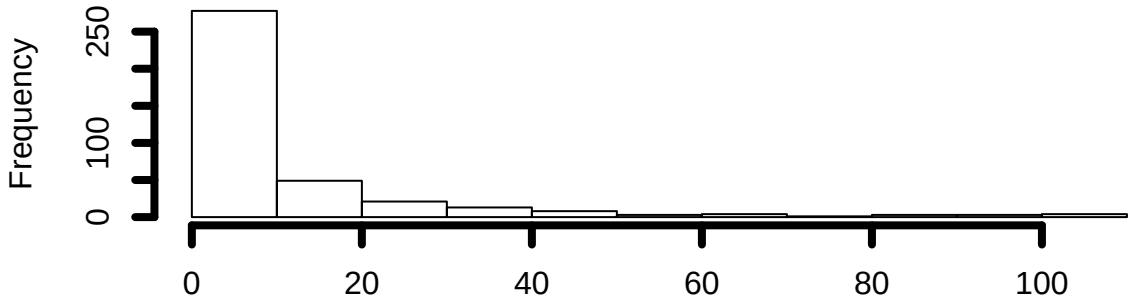

Figure 1: Distribution of Count

Table 1: Summary of Numerical Variables

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Count	1.0000	2.000	5.000	11.78811	12.5	110.0
Population	96369.0000	5150391.000	14447562.000	65765693.91731	48164957.0	1350695000.0
GDP/Cap	123.8762	2159.839	7029.231	15349.91682	24171.9	101668.2

Table 2: Summary of Categorical Variables

	Off	On
Host	362	25
Communist/Soviet	276	111

Table 3: Correlation Matrix

	count	log_pop	log_gdp_per_cap	host	comm_soviet
count	1.0000000	0.4839466	0.2291600	0.4594891	0.0871524
log_pop	0.4839466	1.0000000	-0.1901839	0.2548345	-0.1480479
log_gdp_per_cap	0.2291600	-0.1901839	1.0000000	0.1636597	-0.2989122
host	0.4594891	0.2548345	0.1636597	1.0000000	-0.0736893
comm_soviet	0.0871524	-0.1480479	-0.2989122	-0.0736893	1.0000000

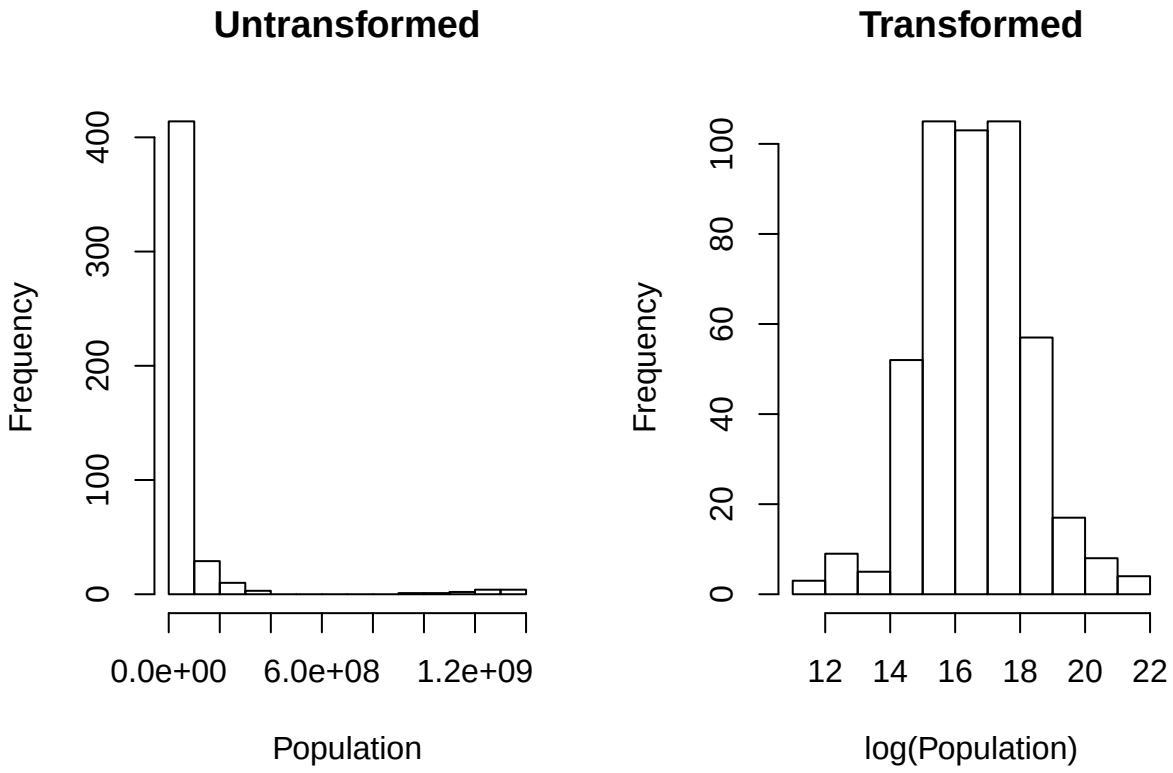

Figure 2: Histogram of the Population Variables

Table 4: Model Selection for Linear Model

	1	2	3	4	Cp	aR2	AIC	PRESS
1	1	0	0	0	250.23	0.22	2165.40	481.0928
1	0	0	1	0	319.74	0.13	2176.89	532.7428
1	0	1	0	0	357.45	0.09	2247.80	561.6892
1	0	0	0	1	418.54	0.01	2265.72	608.1812
2	1	1	0	0	123.00	0.38	2109.14	498.7497
2	1	0	1	0	197.75	0.29	2100.94	384.4047
2	1	0	0	1	220.38	0.26	2154.15	458.7142
2	0	1	1	0	273.23	0.19	2166.76	498.7497
2	0	0	1	1	302.56	0.16	2171.60	520.7314
2	0	1	0	1	317.78	0.14	2238.77	532.9378
3	1	1	0	1	23.77	0.50	2059.22	366.8114
3	1	1	1	0	100.33	0.41	2061.48	309.0094
3	1	0	1	1	164.64	0.33	2084.38	416.4538
3	0	1	1	1	230.32	0.25	2153.04	466.7762
4	1	1	1	1	5.00	0.53	2008.82	294.4012

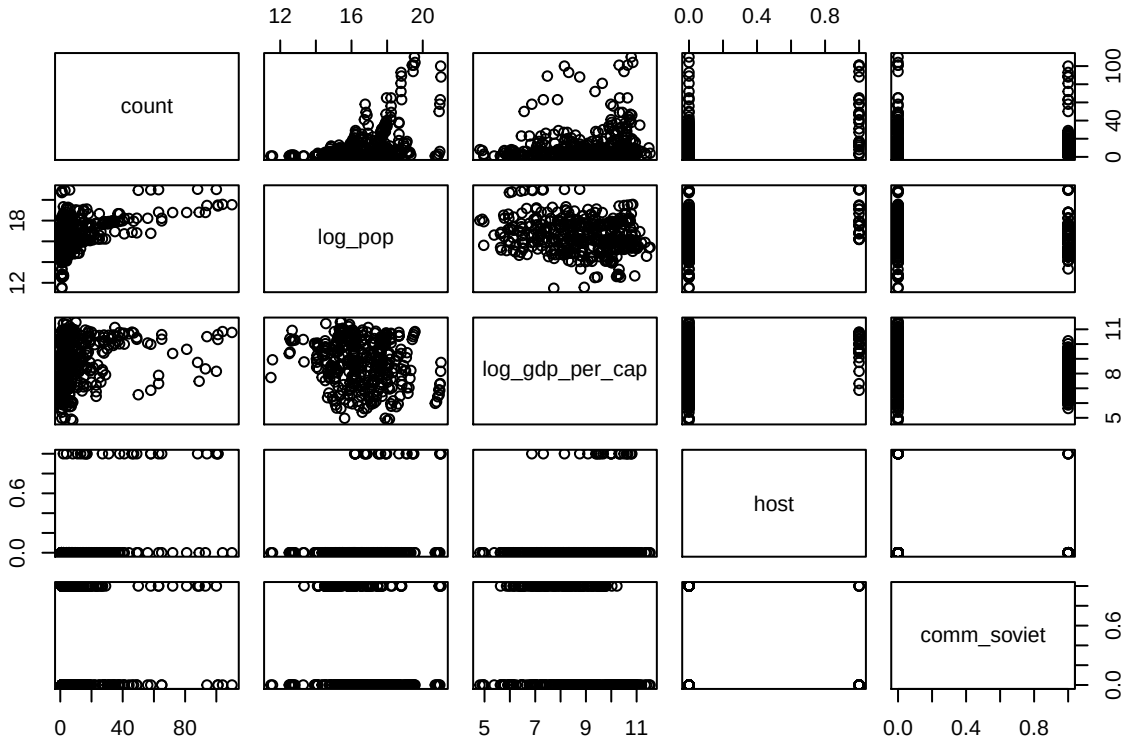

Figure 3: Scatter Matrix

Table 5: Model Selection Diagnostics for a Negative Binomial Model

Pop	GDP/C	Host	Soviet	Parameters	Cp.nb	AIC.nb
1	0	0	0	2	198.46	2497.82
0	0	1	0	2	215.98	2633.68
0	1	0	0	2	336.41	2658.69
0	0	0	1	2	370.52	2695.53
1	0	1	0	3	108.70	2474.29
1	1	0	0	3	119.18	2428.33
1	0	0	1	3	180.88	2489.15
0	1	1	0	3	199.48	2601.80
0	0	1	1	3	206.80	2626.84
0	1	0	1	3	318.21	2614.08
1	1	0	1	4	58.39	2338.79
1	1	1	0	4	60.95	2417.52
1	0	1	1	4	87.77	2457.19
0	1	1	1	4	178.38	2556.83
1	1	1	1	5	5.00	2319.23

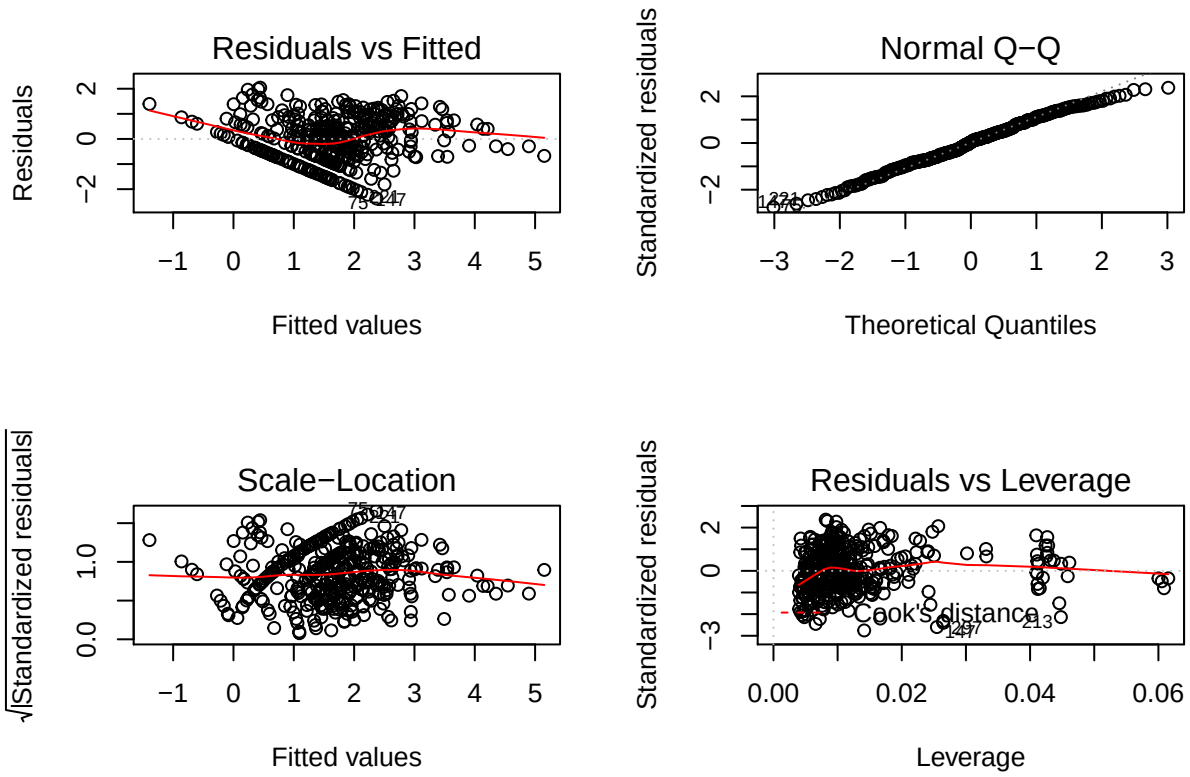

Figure 4: Exploratory Linear Model Plots

Table 6: Linear vs. GLM Model Comparison (Absolute Scale)

	AIC	MSE	MSE (out of sample)
Linear	2008.82	442.44	279.47
Negative Binomial	2319.23	480.74	675.54

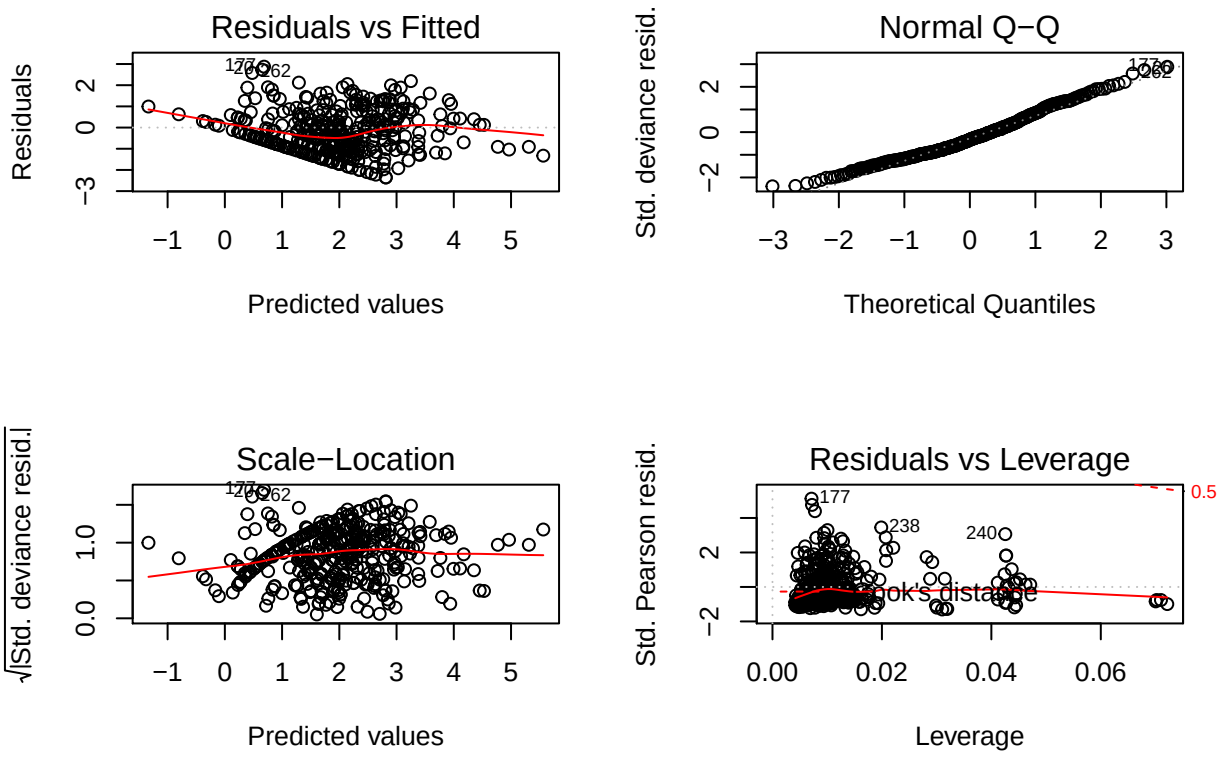

Figure 5: Exploratory Negative Binomial Model Plots

Table 7: Coefficient Comparison

	Linear	Robust (Bisquare)	Poisson	Neg Binom	Neg Binom (fixed effects
Intercept	-9.63 (0.638)	-10.04 (0.67)	-11.23 (0.254)	-10.17 (0.647)	-10.57 (0.641)
log(GDP/Cap)	0.408 (0.032)	0.412 (0.034)	0.521 (0.014)	0.402 (0.029)	0.452 (0.033)
log(Pop)	0.443 (0.029)	0.464 (0.031)	0.505 (0.010)	0.498 (0.032)	0.506 (0.028)
Host	0.867 (0.190)	0.829 (0.199)	0.311 (0.042)	0.693 (0.159)	0.608 (0.153)
Soviet/Comm	1.033 (0.105)	1.04 (0.110)	1.02 (0.037)	1.03 (0.098)	1.08 (0.097)

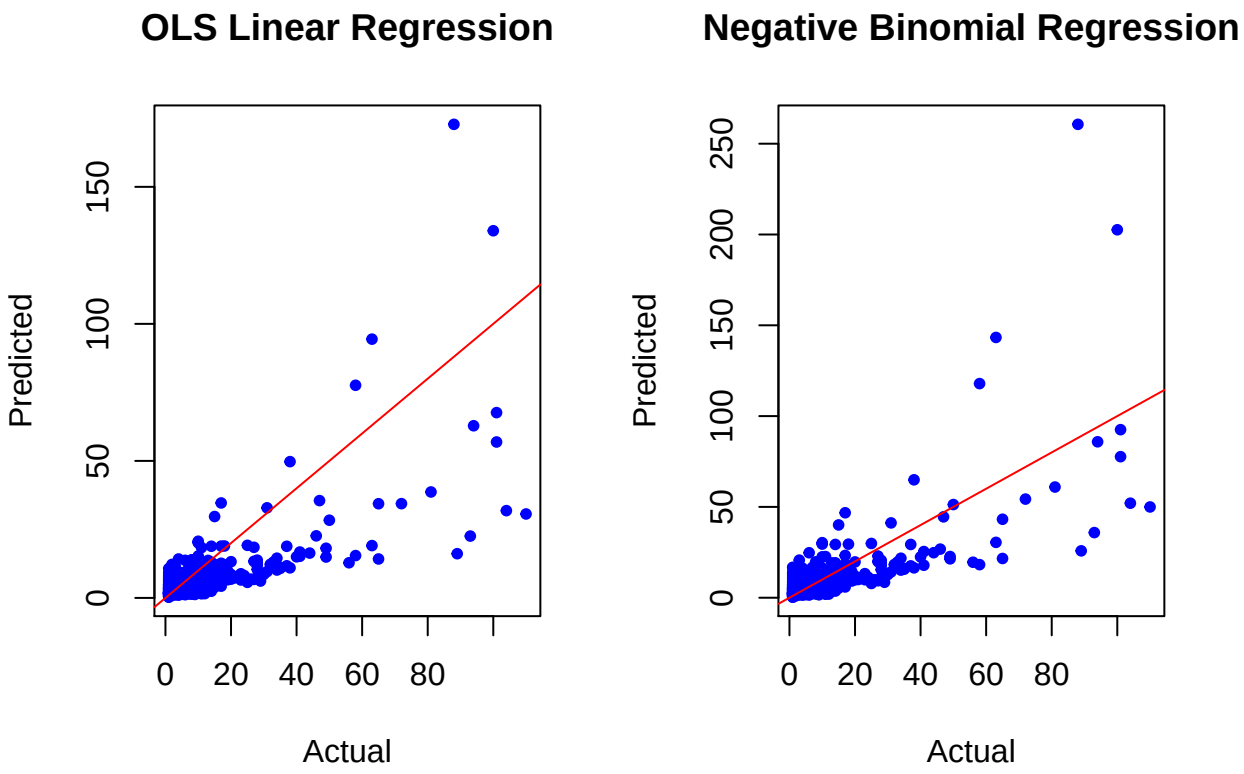

Figure 6: Comparison of Predicted vs. Actual

8 Appendix B: Influential Cases

	country	count	year	gdp	pop	host	comm_soviet
2	China	100	2008	4598206091384	1324655000	1	1
4	United Kingdom	47	2008	2890564338235	61806995	1	0
5	Australia	46	2008	1052584601611	21249200	1	0
17	Brazil	15	2008	1695824571927	192979029	1	0
20	Jamaica	11	2008	13678606692	2790122	0	0
48	Greece	4	2008	354460802549	11077841	1	0
75	Vietnam	1	2008	99130304099	86707801	0	1
86	United States	101	2004	12274928000000	292805298	1	0
88	China	63	2004	1955347004963	1296075000	1	1
90	Australia	49	2004	611904253806	20127400	1	0
94	United Kingdom	31	2004	2398555474185	59987905	1	0
102	Greece	16	2004	240521260988	10955141	1	0
147	India	1	2004	699688852930	1126135777	0	0
238	China	88	2012	8560547314679	1350695000	1	1
240	United Kingdom	65	2012	2662085168499	63700300	1	0
242	Japan	38	2012	6203213121334	127629000	1	0
252	Brazil	17	2012	2465188674415	200560983	1	0
258	Jamaica	12	2012	14800165407	2840992	0	0
294	Greece	2	2012	245670666639	11045011	1	0

	country	count	year	gdp	pop	host	comm_soviet
302	Saudi Arabia	1	2012	735974843360	29086357	0	0
303	Venezuela	1	2012	381286237848	29893080	0	0
318	United States	94	2000	10284779000000	282162411	1	0
320	China	58	2000	1211346869605	1262645000	1	1
321	Australia	58	2000	415034227218	19153000	1	0
336	Greece	13	2000	130133845771	10805808	1	0
340	Spain	11	2000	595402616547	40567864	1	0
343	Jamaica	9	2000	8985352832	2656864	0	0
378	India	1	2000	462146799338	1053050912	0	0
395	United States	101	1996	8100201000000	269394000	1	0
398	China	50	1996	863746717504	1217550000	0	1
399	Australia	41	1996	400302731411	18311000	1	0
402	South Korea	27	1996	598099073901	45524681	1	0
409	Spain	17	1996	640998292395	39889852	1	0
420	Greece	8	1996	145861612826	10608800	1	0
452	India	1	1996	387656017799	978893217	0	0
453	Mexico	1	1996	410975595310	95687452	0	0
468	Tonga	1	1996	219583570	96369	0	0

	dfb.1_	dfb.lg_p	dfb.l____	dfb.host	dfb.cmm_	dffit	cov.r	cook.d	hat
2	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
4	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
5	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
17	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
20	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
48	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
75	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
86	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
88	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
90	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
94	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
102	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
147	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
157	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
159	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
161	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
171	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
177	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
213	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE
221	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
222	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
237	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
239	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
240	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
255	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
259	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
262	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
297	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
314	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
317	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
318	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
321	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
328	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
339	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE
371	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
372	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
387	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

9 Appendix C

```
knitr::opts_chunk$set(echo = FALSE, comment = NA)
base <-read.csv('data/base_data.csv', stringsAsFactors = F)
P__disp <- function(x) {
  pr <- sum(residuals(x, type="pearson")^2)
  dispersion <- pr/x$df.residual
  c(pr, dispersion)
}
library(dplyr)
library(qpcR)
library(MuMIn)
```

```
base.total = base[which(base$year!=2016),]
Olympic = base.total[,c(2,3,4,5,6,7,8)]
#Clean the data (Mary to add)
#Make new dataframe with GDP / capita
Olympic_v2 <- data.frame(year = Olympic$year, country = Olympic$country, count = Olympic$count, log_pop
library(knitr)
summary(base$count)
hist(base$count)
hist(base[which(base$year == c("2008")),c("count")])
summary(base$gdp)
summary(base$pop)
sum(base$comm_soviet)
nrow(base)-sum(base$comm_soviet)
sum(base$host)
par(mfrow=c(1,2))
#Histogram Population
hist(base$pop,main = "Untransformed",xlab = "Population")
#Histogram log(Population)
hist(log(base$pop),main = "Transformed",xlab = "log(Population)")
#Log(GDP)
boxplot(log(base$gdp),ylab = "log(GDP)")
#log(Population)
boxplot(log(base$pop),ylab = "log(Population)")
pairs(Olympic[,c(2,4,5,6,7)])
kable(cor(Olympic[,c(2,4,5,6,7)]))
library(leaps)
#CP
olympic.leapCP <- leaps(y=log(Olympic_v2$count), x=Olympic_v2[,4:7])
#R2a
olympic.leapR2a <- leaps(y=log(Olympic_v2$count), x=Olympic_v2[,4:7], method = 'adjr2')
  ### Code for AIC
xList <- names(Olympic_v2)[4:7]
  #### Remove the last row that has all False's
vec <- olympic.leapCP$which
  ### Name the columns in the grid
names(vec) <- paste("X", 1:4, sep="")
  #### Build matrix of formula for every row
allModelsList <- apply(vec, 1, function(x) as.formula(
  paste(c("count ~ 1", xList[x]), collapse = "+")))
  ### Calculate the coefficients for all 16 models
allModelsResults.lm <- lapply(allModelsList,
                              function(x) lm(x, data=Olympic_v2))
olympic.lmfinal <- lm(log_count ~ log_pop + log_gdp_per_cap + host + comm_soviet, data = Olympic_v2)
par(mfrow=c(2,2))
plot(olympic.lmfinal)
library(lmtest)
bptest(log_count ~ log_pop + log_gdp_per_cap + host + comm_soviet,data = Olympic_v2,studentize = FALSE)
library(nortest)
lillie.test(olympic.lmfinal$residuals)
Olympic.pois<-glm(count~log_pop + log_gdp_per_cap + host + comm_soviet, data = Olympic_v2, family = pois
P__disp(Olympic.pois)
library(MASS)
olympic.nb <- glm.nb(count~log_pop + log_gdp_per_cap + host + comm_soviet, data = Olympic_v2)
```

```r
library(leaps)
olympic.nb_leap <- leaps(y=Olympic_v2$count, x=Olympic_v2[,4:7])
Cp.nb<-round(olympic.nb_leap$Cp, 2)
which<-olympic.nb_leap$which
rownames(which) <-NULL
colnames(which)<-c('Pop', 'GDP/C', 'Host', 'Soviet')
xList <- names(Olympic_v2)[4:7]
vec <- olympic.nb_leap$which
#Name the columns in the grid
names(vec) <- paste("X", 1:4, sep="")
#Build matrix of formula for every row
allModelsList <- apply(vec, 1, function(x) as.formula(
  paste(c("count ~ 1", xList[x]), collapse = "+")))
#Calculate the coefficients for all 16 models
allModelsResults <- lapply(allModelsList,
                           function(x) glm.nb(x, data=Olympic_v2))
AIC.nb<-matrix(unlist(lapply(allModelsResults, function(x) round(extractAIC(x),2))), ncol = 2, byrow = T
library(knitr)
#2016 Data
base.2016 = base[which(base$year==2016),]
attach(base.2016)
base.2016 = data.frame(country,count,year,log_pop=log(pop),log_gdp_per_cap = log(gdp/pop),host,comm_sovi


#Linear Fitted
fitted.lm = -9.63058+0.40830*base.2016$log_gdp_per_cap+0.44275*base.2016$log_pop+0.86726*base.2016$host+
#olympic.lmfinal




#Prediction intervals: Linear Model
nrow(base.2016)-5
t=qt(1-0.05/2,76)
mse.lm = 0.751
X = as.matrix(cbind(rep(1,nrow(Olympic_v2)),Olympic_v2$log_gdp_per_cap,Olympic_v2$log_pop,Olympic_v2$hos

spred = c()
for (i in 1:nrow(base.2016)){
  xh = as.matrix(cbind(1,base.2016$log_gdp_per_cap[i],base.2016$log_pop[i],base.2016$host[i],base.2016$c
  s2 = mse.lm*(1+xh%*%solve(t(X)%*%X)%*%t(xh))
  spred = c(spred,sqrt(s2))
}

lower.pred = fitted.lm - qt(1-0.05/2,76)*spred
upper.pred = fitted.lm + qt(1-0.05/2,76)*spred

forecast.lm = cbind(log(base.2016$count),lower.pred,upper.pred)
ininterval.lm = cbind(forecast.lm[,2]<=forecast.lm[,1] & forecast.lm[,3]>=forecast.lm[,1])
length(which(ininterval.lm))

#NB Method 2 (works!)
library(tidyverse)
```

```r
library(ciTools)
library(MASS)

set.seed(20181215)

base.2016 = base[which(base$year==2016),]
attach(base.2016)
base.2016 = data.frame(country,count,year,log_pop=log(pop),log_gdp_per_cap = log(gdp/pop),host,comm_sovi

newData <- data.frame(base.2016[,c(2,4,5,6,7)])
train_data <- Olympic_v2[,3:7]

#Generate prediction intervals
olympic.nb = glm.nb(count ~ log_pop + log_gdp_per_cap + host + comm_soviet,data=train_data)

#add_pi comes from the library ciTools
olympic.nb_pint <- add_pi(tb=newData,fit=olympic.nb, names=c("lpb", "upb"), alpha=0.1, nSims=20000)


olympic.nb_pint %>%
  mutate(inside=(count >lpb & count<upb))->preds.nb

summary(preds.nb)


####finding MSEs
preds<-data.frame(matrix(cbind(base.2016$count, preds.nb$pred, exp(fitted.lm)), ncol = 3))

colnames(preds)<-c('Actual', 'Prediction.nb', 'Prediction.lm')

mse.nb<-mean((preds$Actual-preds$Prediction.nb)^2)
mse.lm<-mean((preds$Actual-preds$Prediction.lm)^2)
hist(Olympic_v2$count, lwd = 4, xlab = '', main = '')
par(mfrow=c(1,2))
#Histogram Population
hist(base$pop,main = "Untransformed",xlab = "Population")
#Histogram log(Population)
hist(log(base$pop),main = "Transformed",xlab = "log(Population)")
options(scipen = 999)

kable(rbind('Count'=summary(Olympic_v2$count), 'Population'=summary(exp(Olympic_v2$log_pop)), 'GDP/Cap'=
this<-rbind('Host' = unname(table(Olympic_v2$host)),
'Communist/Soviet' = unname(table(Olympic_v2$comm_soviet)))

colnames(this)<-c('Off', 'On')

kable(this, caption = 'Summary of Categorical Variables')
plot(Olympic_v2[,c(3,4,5,6,7)])
kable(cor(Olympic_v2[,c(3,4,5,6,7)]), caption = 'Correlation Matrix')
#PRESS (Non-Mac)
library(qpcR)
olympic.lm = PRESS(lm(log_count~log_pop+log_gdp_per_cap+host+comm_soviet, data = Olympic_v2))
olympic.lmX1 = PRESS(lm(log_count~log_pop, data = Olympic_v2))
```

```
olympic.lmX2 = PRESS(lm(log_count~log_gdp_per_cap, data = Olympic_v2))
olympic.lmX3 = PRESS(lm(log_count~host, data = Olympic_v2))
olympic.lmX4 = PRESS(lm(log_count~comm_soviet, data = Olympic_v2))
olympic.lmX1X2 = PRESS(lm(log_count~log_pop+log_gdp_per_cap, data = Olympic_v2))
olympic.lmX1X3 = PRESS(lm(log_count~log_gdp_per_cap+host, data = Olympic_v2))
olympic.lmX1X4 = PRESS(lm(log_count~log_pop+comm_soviet, data = Olympic_v2))
olympic.lmX2X3 = PRESS(lm(log_count~log_gdp_per_cap+host, data = Olympic_v2))
olympic.lmX2X4 = PRESS(lm(log_count~log_gdp_per_cap+comm_soviet, data = Olympic_v2))
olympic.lmX3X4 = PRESS(lm(log_count~host+comm_soviet, data = Olympic_v2))
olympic.lmX1X2X3 = PRESS(lm(log_count~log_pop+log_gdp_per_cap+host, data = Olympic_v2))
olympic.lmX1X2X4 = PRESS(lm(log_count~log_pop+log_gdp_per_cap+comm_soviet, data = Olympic_v2))
olympic.lmX2X3X4 = PRESS(lm(log_count~log_gdp_per_cap+host+comm_soviet, data = Olympic_v2))
olympic.lmX1X3X4 = PRESS(lm(log_count~log_pop+host+comm_soviet, data = Olympic_v2))
olympic.lm_press <- rbind(olympic.lmX1$stat,
                          olympic.lmX3$stat,
                          olympic.lmX2$stat,
                          olympic.lmX4$stat,
                          olympic.lmX1X3$stat,
                          olympic.lmX1X2$stat,
                          olympic.lmX1X4$stat,
                          olympic.lmX2X3$stat,
                          olympic.lmX3X4$stat,
                          olympic.lmX2X4$stat,
                          olympic.lmX1X2X3$stat,
                          olympic.lmX1X2X4$stat,
                          olympic.lmX1X3X4$stat,
                          olympic.lmX2X3X4$stat,
                          olympic.lm$stat)
#Summary
Diagnostics = cbind(olympic.leapCP$which, Cp=round(olympic.leapCP$Cp,2), aR2=round(olympic.leapR2a$adjr
      AIC=matrix(unlist(lapply(allModelsResults.lm, function(x) round(extractAIC(x),2))), ncol=2, byrow=
#PRESS wasn't showing as column name
colnames(Diagnostics) = c("1","2","3","4","Cp","aR2","AIC","PRESS")
par(mfrow=c(2,2))
plot(olympic.lmfinal)
par(mfrow=c(2,2))
plot(olympic.nb)
kable(Diagnostics, caption = 'Model Selection for Linear Model')
kable(cbind(which, Parameters = olympic.nb_leap$size, Cp.nb, AIC.nb), caption = 'Model Selection Diagnos
par(mfrow=c(1,2))
plot(x=Olympic_v2$count, y = exp(olympic.lmfinal$fitted.values), col='blue', pch=20, xlab='Actual',
     ylab='Predicted', main='OLS Linear Regression')
abline(a=0,b=1, col='red')
plot(x=Olympic_v2$count, y = olympic.nb$fitted.values, col='blue', pch=20, xlab='Actual',
     ylab='Predicted', main='Negative Binomial Regression')
abline(a=0,b=1, col='red')
df_20<-data.frame(matrix(c(2008.82, 442.44, 279.47, 2319.23,  480.74, 675.54), nrow = 2, byrow = T))
colnames(df_20)<-c('AIC', 'MSE', 'MSE (out of sample)')
rownames(df_20)<-c('Linear', 'Negative Binomial')
kable(df_20, caption = 'Linear vs. GLM Model Comparison (Absolute Scale)')
df_133<-data.frame(matrix(c(
'-9.63 (0.638)', '-10.04 (0.67)','-11.23 (0.254)','-10.17 (0.647)','-10.57 (0.641)',
'0.408 (0.032)','0.412 (0.034)','0.521 (0.014)','0.402 (0.029)','0.452 (0.033)',
```

```
'0.443 (0.029)','0.464 (0.031)','0.505 (0.010)','0.498 (0.032)','0.506 (0.028)',
'0.867 (0.190)','0.829 (0.199)','0.311 (0.042)','0.693 (0.159)','0.608 (0.153)',
'1.033 (0.105)','1.04 (0.110)','1.02 (0.037)','1.03 (0.098)','1.08 (0.097)'
), ncol = 5, byrow = T))
colnames(df_133)<-c(
'Linear',
'Robust (Bisquare)',
'Poisson',
'Neg Binom',
'Neg Binom (fixed effects')
rownames(df_133)<-c('Intercept','log(GDP/Cap)','log(Pop)','Host','Soviet/Comm')
kable(df_133, caption = 'Coefficient Comparison')
#Influential cases
olympic.lm_inf=influence.measures(olympic.lmfinal)$is.inf
idx=which(apply(olympic.lm_inf,1,any))
#Influential Cases
kable(Olympic[idx,])
#Inluential Cases by Test
kable(olympic.lm_inf[idx,])

kable(cbind('Count' = preds.nb$count, 'NB Predictions' = round(preds.nb$pred, 2), 'NB Interval' = paste
```

10 Appendix D

Table 10: Out-of-sample Predicted Values

Count	NB Predictions	NB Interval	Linear Predictions	Linear Intervals
121	110.48	(19, 269)	80.44	(14, 470)
70	290.9	(53, 711)	192.93	(33, 1142)
67	43.25	(7, 107)	34.35	(6, 200)
56	49.03	(8, 117)	30.99	(5, 177)
42	24.67	(4, 60)	16.25	(3, 92)
42	21.05	(3, 51)	14.02	(2, 79)
41	59.25	(10, 143)	45.34	(8, 264)
29	14.33	(2, 35)	10.12	(2, 57)
28	18.62	(3, 45)	12.45	(2, 71)
22	16.41	(2, 40)	11.31	(2, 64)
21	16.41	(2, 40)	11.07	(2, 63)
19	11.6	(1, 29)	8.35	(1, 47)
19	41.29	(7, 101)	30.47	(5, 177)
18	5.81	(0, 15)	4.49	(1, 25)
18	9.17	(1, 23)	6.7	(1, 38)
17	15.4	(2, 37)	10.45	(2, 59)
17	16.31	(2, 40)	11.57	(2, 66)
15	14.87	(2, 36)	10.94	(2, 62)
15	7.19	(1, 18)	5.5	(1, 31)
13	4.89	(0, 13)	3.25	(1, 18)
13	12.93	(2, 32)	8.81	(2, 50)
11	28.81	(5, 70)	19.65	(3, 112)
11	9.33	(1, 23)	6.92	(1, 39)
11	15.26	(2, 38)	10.2	(2, 58)
11	1.95	(0, 6)	1.53	(0, 9)
10	6.36	(0, 16)	4.67	(1, 26)
10	8.82	(1, 22)	5.86	(1, 33)
10	9.55	(1, 24)	7.37	(1, 42)
9	10.03	(1, 25)	7.35	(1, 42)
8	10.5	(1, 26)	6.84	(1, 39)
8	8.51	(1, 21)	5.7	(1, 32)
8	5.32	(0, 14)	3.38	(1, 19)
8	14.04	(2, 34)	9.19	(2, 52)
8	8.93	(1, 22)	6.65	(1, 38)
7	10.2	(1, 26)	7.66	(1, 43)
7	5.66	(0, 15)	4.36	(1, 25)
6	10.28	(1, 25)	6.76	(1, 38)
6	6.33	(0, 16)	4.65	(1, 26)
6	9.09	(1, 23)	6.69	(1, 38)
5	16.05	(2, 39)	10.22	(2, 58)
5	8.34	(1, 21)	5.75	(1, 32)
5	18.68	(3, 46)	13.19	(2, 75)
4	7.69	(1, 20)	5.93	(1, 34)
4	8.44	(1, 21)	6.79	(1, 39)
4	4.36	(0, 12)	3.32	(1, 19)
4	11.09	(1, 27)	7.51	(1, 42)
4	8.56	(1, 22)	6.75	(1, 38)
4	4.88	(0, 13)	3.81	(1, 22)

Count	NB Predictions	NB Interval	Linear Predictions	Linear Intervals
3	16.23	(2, 40)	9.88	(2, 56)
3	9.74	(1, 24)	6.27	(1, 35)
3	3.45	(0, 9)	2.5	(0, 14)
3	10.2	(1, 26)	7.61	(1, 43)
2	27.15	(4, 65)	15.03	(3, 86)
2	6.66	(1, 17)	4.5	(1, 25)
2	5.01	(0, 13)	3.91	(1, 22)
2	22.51	(3, 54)	14.44	(3, 82)
2	7.04	(1, 18)	5.45	(1, 31)
2	7.58	(1, 19)	5.66	(1, 32)
2	4.62	(0, 12)	3.59	(1, 20)
2	1.5	(0, 5)	1.33	(0, 8)
2	2.54	(0, 7)	2.09	(0, 12)
1	8.25	(1, 21)	6.16	(1, 35)
1	6.47	(0, 17)	4.96	(1, 28)
1	1.84	(0, 5)	1.27	(0, 7)
1	11.23	(1, 28)	6.94	(1, 39)
1	5.53	(0, 14)	3.76	(1, 21)
1	9.47	(1, 24)	6.06	(1, 34)
1	4.58	(0, 12)	3.34	(1, 19)
1	3.28	(0, 9)	2.4	(0, 14)
1	6.47	(0, 17)	4.76	(1, 27)
1	8	(1, 20)	5.95	(1, 34)
1	4.27	(0, 11)	3.12	(1, 18)
1	6.22	(0, 16)	5.13	(1, 29)
1	4.17	(0, 11)	3.21	(1, 18)
1	1.18	(0, 4)	0.85	(0, 5)
1	1.12	(0, 4)	0.93	(0, 5)
1	5.02	(0, 13)	4.02	(1, 23)
1	3.89	(0, 10)	3.12	(1, 18)
1	2.18	(0, 6)	1.8	(0, 10)
1	7.2	(1, 18)	5.52	(1, 31)
1	0.5	(0, 2)	0.47	(0, 3)