mjon_238_assignment_3

Michael Jones

31/01/2022

```
source("prime.poisson.data.generator.R")
```

```
## The first ten prime numbers: 2 3 5 7 11 13 17 19 23 29
```

Question 1)

a) Equations:

The model we are fitting has the following equation:

$$log(y_i) = \beta_0 + \beta_1 \times center_i + \beta_2 \times ratio_i + log(size_i)$$

Where y_i is the ith observations number of primes in the sequences of consecutive numbers. $center_i$ is the ith observations median of the elements in its sequence. $ratio_i$ is the ith observations ratio of x_n/x_1.

b)

```
poisson.fit = glm(y ~ center + ratio, family = "poisson",
offset = log(size), data = prime.poisson.df)

# Hypothesis test 1 & 2 -----
anova(poisson.fit, test = "Chisq") # two p-values
```

```
## Analysis of Deviance Table
##
## Model: poisson, link: log
##
## Response: y
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                     999     606.31
## center  1   50.107       998     556.20 1.456e-12 ***
## ratio   1   10.996       997     545.21  0.000913 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Hypothesis test 3 -----
1-pchisq(poisson.fit$deviance, poisson.fit$df.residual) # one p-value
```

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

Hypothesis Test 1:

- H0: $\beta_1 = 0$, the median of the elements in a sequence of consecutive numbers does not effect the number of primes in that sequence.
- Conclusion: We reject the H0, $\beta_1 \neq 0$, the median of the elements in a sequence of consecutive numbers does effect the number of primes in that sequence.

Hypothesis Test 2:

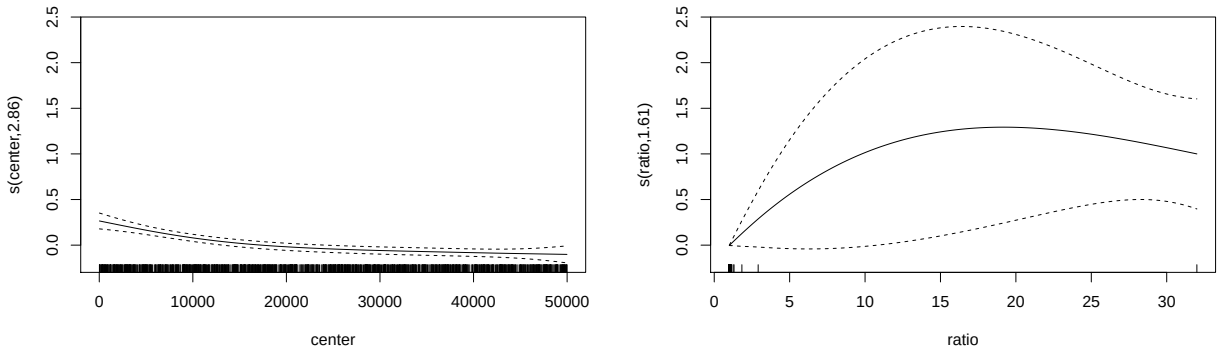
- H0: $\beta_2 = 0$, the ratio of x_n/x_1 does not effect the number of primes in a sequence of consecutive numbers.
- Conclusion: We reject the H0, $\beta_2 \neq 0$, the ratio of x_n/x_1 has an effect on the number of primes in a sequence of consecutive numbers.

Hypothesis Test 3:

- H0: The model is correct.
- Conclusion: We accept the null hypothesis, the model is correct.

c)

```
gam.pois.fit = gam(y ~ s(center) + s(ratio), family = "poisson",
offset = log(size), data = prime.poisson.df)

plot(gam.pois.fit)
```



The *center* variable may have a non-linear relationship. This is evident through the fitted line method. We can draw a straight line through the dotted lines. Given this *center* might need a transformation, but

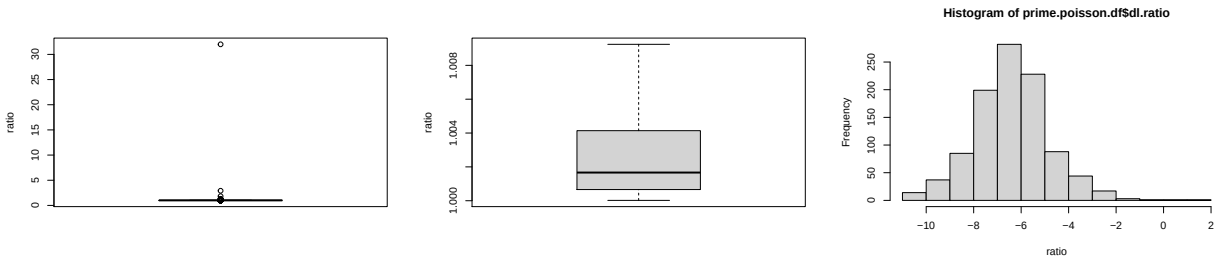
we must do more research/experimentation because the fitted-line method does not confirm we have a non-linear variable, it just suggests we might.

The GAM plots do not provide much information for the *ratio* variable. This is because we do **not** have symmetry in the data. The vast majority of observations are between 1 and 1.1, there are very few data points larger than this. The dotted-line bands are also very wide. For those reasons we will not use the fitted line method.

d)

```
# New variable dl.ratio

prime.poisson.df$dl.ratio = log(log(prime.poisson.df$ratio))
```

```
boxplot(prime.poisson.df$ratio, ylab = "ratio")
boxplot(prime.poisson.df$ratio, ylab = "ratio", outline=FALSE)
hist(prime.poisson.df$dl.ratio, xlab = "ratio")
```

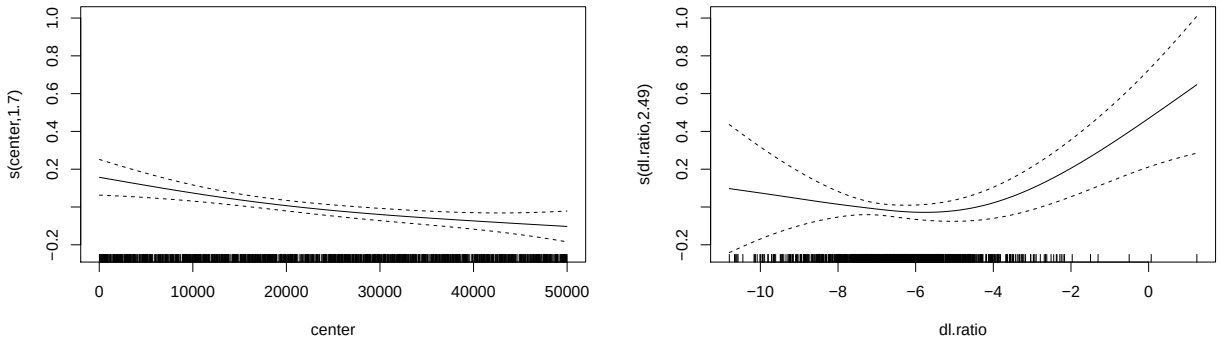


The first plot shows that the ratio variable is very **right-skewed**. the second and third plots show the Log-Log transformation reduces skewness so the observations seem approximately normal.

e)

Including the dl.ratio variable

```
gam.pois.fit2 = gam(y ~ s(center) + s(dl.ratio), family = "poisson",
offset = log(size), data = prime.poisson.df)

plot(gam.pois.fit2)
```



The first plot, regarding the *center* variable, now seems very linear.

The second plot, regarding the *dl.ratio* variable, still seems non-linear. However, the non-linear areas (left and right poitns on the graph), still have very few data points. So its still not clear how reliable it is to assumed dl.ratio is non-linear.

f)

```
detach("package:mgcv", unload = TRUE)
library(VGAM)
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
vgam.pois.fit <- VGAM::vgam(y ~ s(center) + s(dl.ratio),
                            family = 'poissonff',
                            offset = log(size),
                            data = prime.poisson.df)

summary(vgam.pois.fit)
```

```
##
## Call:
## VGAM::vgam(formula = y ~ s(center) + s(dl.ratio), family = "poissonff",
##     data = prime.poisson.df, offset = log(size))
##
## Name of additive predictor: loglink(lambda)
##
## (Default) Dispersion Parameter for poissonff family:   1
##
## Residual deviance:  525.4312 on 991.093 degrees of freedom
##
## Log-likelihood: -1725.63 on 991.093 degrees of freedom
##
## Number of Fisher scoring iterations:  6
##
```

```
## DF for Terms and Approximate Chi-squares for Nonparametric Effects
##
##             Df Npar Df Npar Chisq   P(Chi)
## (Intercept)  1
## s(center)    1     3.0     5.8003 0.121087
## s(dl.ratio)  1     2.9    14.2106 0.002410
```

First Examining the center variable

- The null hypothesis is H0: $f_j(center_j) = \beta_j \times center_j$, the variable follows a linear relationship.
- Given the p-value 0.121 we accept the null hypothesis. $f_j(center_j) = \beta_j \times center_j$

Now Examining the dl.ratio variable

- The null hypothesis is H0: $f_j(dl.ratio_j) = \beta_j \times dl.ratio_j$, the variable is linear.
- Given the p-value is 0.002, we reject the null hypothesis $f_j(dl.ratio_j) \neq \beta_j \times dl.ratio_j$, dl.ratio is non-linear.

Based on these conclusions. We only need to transform the dl.ratio variable.

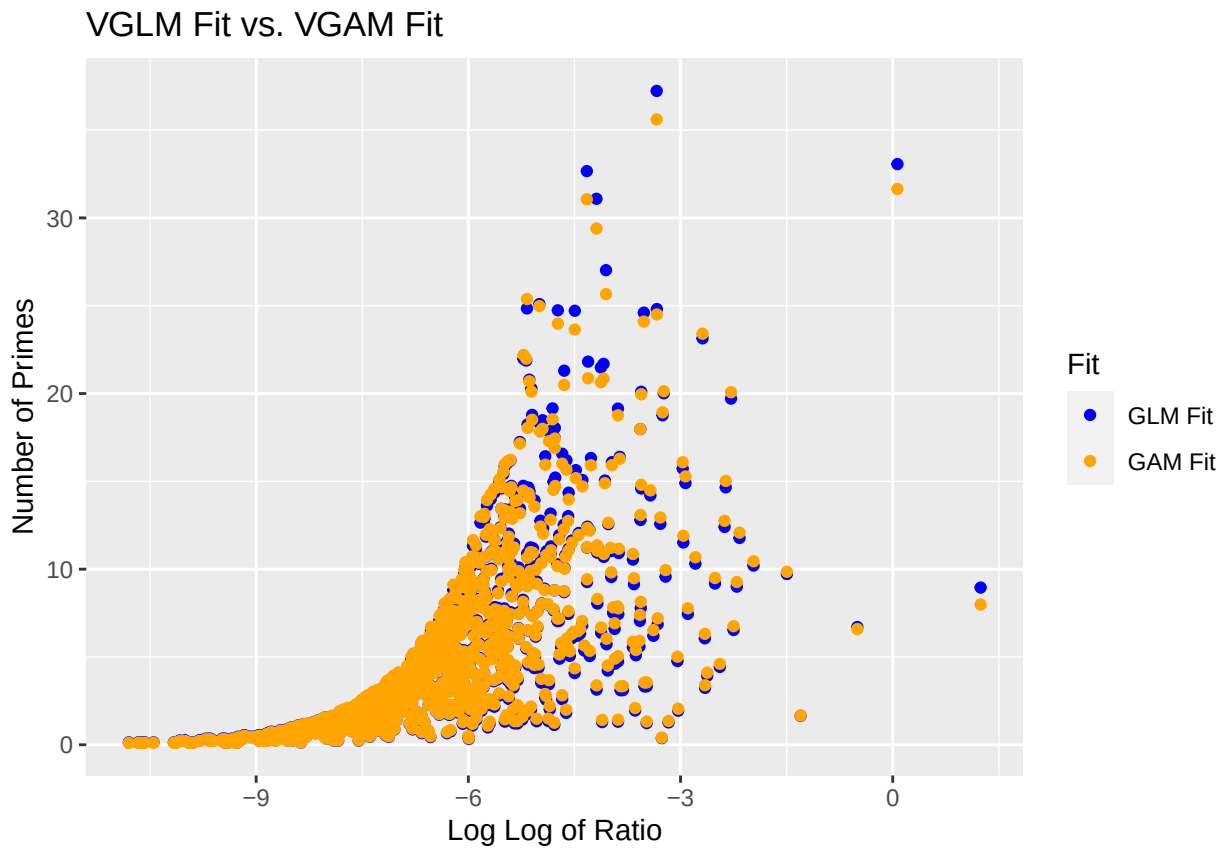
g)

It looked very quadratic, so I decided to fit a polynomial in the GLM model.

```
prime.poisson.df$dl.ratio2 = prime.poisson.df$dl.ratio^2

vglm.pois.fit <- VGAM::vglm(y ~ s(center) + s(dl.ratio) + s(dl.ratio2), family = 'poissonff', offset = l
summary(vglm.pois.fit)
```

```
##
## Call:
## VGAM::vglm(formula = y ~ s(center) + s(dl.ratio) + s(dl.ratio2),
##     family = "poissonff", data = prime.poisson.df, offset = log(size))
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.546e+00  1.047e-01 -14.769  < 2e-16 ***
## s(center)    -4.694e-06  1.331e-06  -3.526 0.000422 ***
## s(dl.ratio)   2.214e-01  4.100e-02   5.400 6.65e-08 ***
## s(dl.ratio2)  1.882e-02  4.001e-03   4.704 2.55e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Name of linear predictor: loglink(lambda)
##
## Residual deviance: 529.875 on 996 degrees of freedom
##
## Log-likelihood: -1727.852 on 996 degrees of freedom
##
## Number of Fisher scoring iterations: 5
##
## No Hauck-Donner effect found in any of the estimates
```

```r
ggplotCompare <- prime.poisson.df%>%
  select(c(y, dl.ratio))
ggplotCompare$`GLM Fit` <- fitted(vglm.pois.fit)
ggplotCompare$`GAM Fit` <- fitted(vgam.pois.fit)

ggplotCompare <- melt(ggplotCompare, id.vars = 'dl.ratio',
                      variable.name = 'Fit',
                      value.name = 'Number of Primes')%>%
  filter(!grepl("y",Fit))


ggplot(data = ggplotCompare,
       aes(dl.ratio)) +
  geom_point(aes(x= dl.ratio, y = `Number of Primes`, color = Fit))+
  scale_color_manual(values = c("GLM Fit" = 'blue','GAM Fit' = 'orange'))+
  xlab("Log Log of Ratio")+ylab("Number of Primes")+
  ggtitle("VGLM Fit vs. VGAM Fit")
```



I plotted the fitted values for the GLM and GAM models. The closeness of the points indicate that the GLM Model does a good job of replicating the GAM model.

h) Deviance testing

I will do a deviance goodness of fit chi-squared test with an null hypothesis:

H0: The model is appropriate.

```r
#Print Deviance
deviance(vglm.pois.fit)
```

```
## [1] 529.875
```

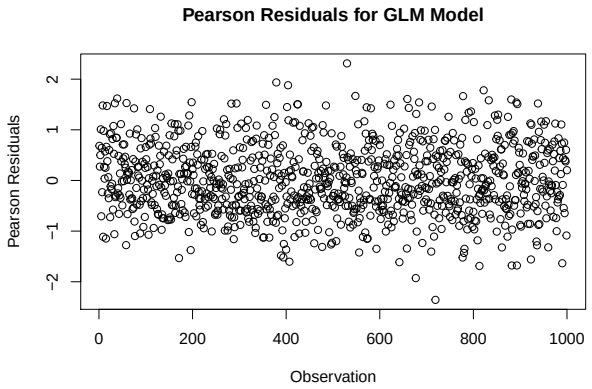
```r
#Chi Square Test
1-pchisq(deviance(vglm.pois.fit), 996)
```

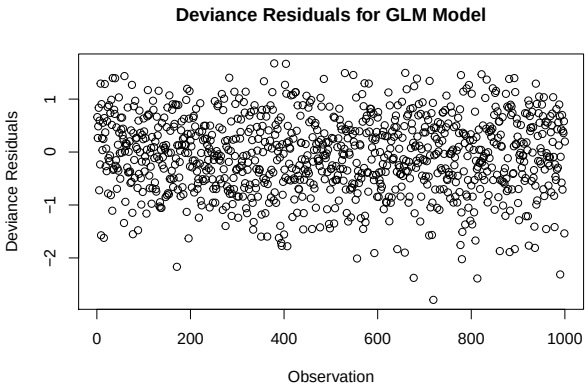
```
## [1] 1
```

Given a p-value of approximately 1, we accept the null hypothesis, **the model is appropriate**

I fit two plots, deviance and pearson residuals.

```r
#Residual Plot
plot(residuals(vglm.pois.fit, type = 'pearson'),
     xlab = "Observation",
     ylab = "Pearson Residuals")
title(main = "Pearson Residuals for GLM Model")

plot(residuals(vglm.pois.fit, type = 'deviance'),
     xlab = "Observation",
     ylab = "Deviance Residuals")
title(main = "Deviance Residuals for GLM Model")
```





Both pearson and deviance residuals have a zero mean and a constant variance. All values are inside the interval [-3,3], suggesting both residuals have approximate standard normal distribution (N(0,1)).

For the reasons above, the model is appropriate.

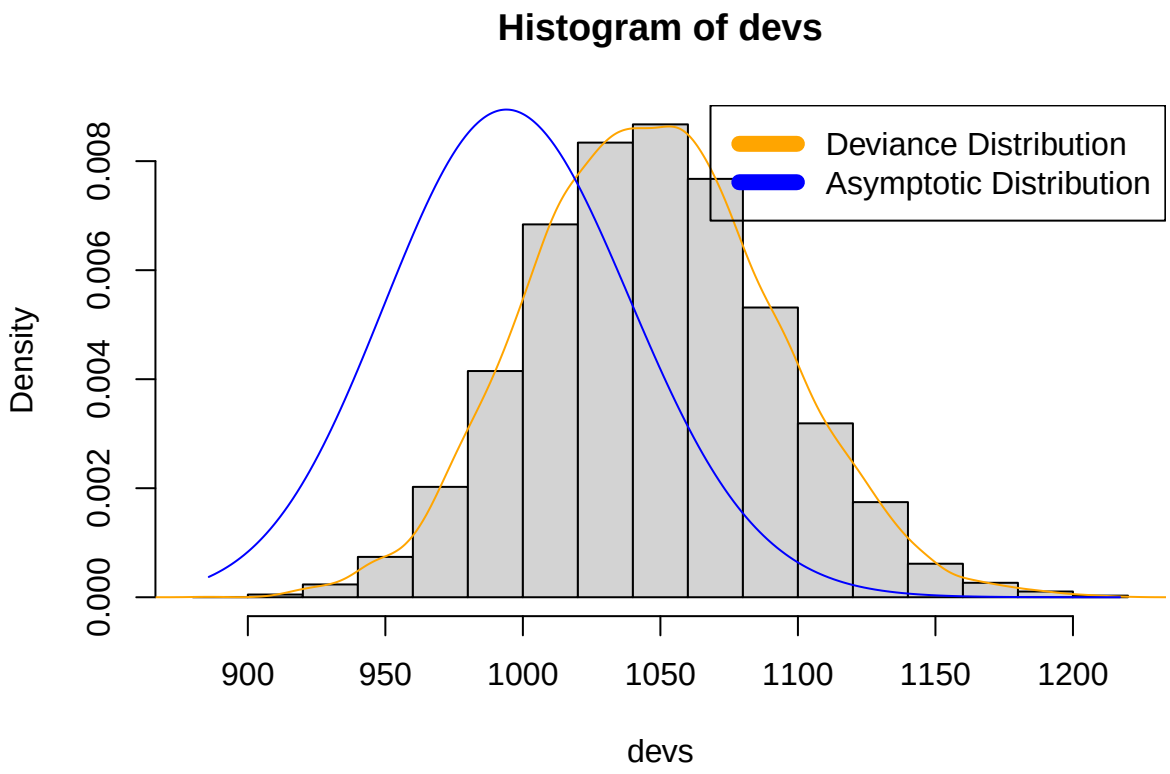
i)

Bootstrapping

```r
#Number of Sims
Nsim = 10000
nr = nrow(prime.poisson.df)
```

```r
#Generate Betas
betahat <- coef(vglm.pois.fit)
b0 <- betahat[1]
b1 <- betahat[2]
b2 <- betahat[3]
b3 <- betahat[4]

#Input X's
x1 <- prime.poisson.df$center
x2 <- prime.poisson.df$dl.ratio
x3 <- prime.poisson.df$dl.ratio2
logSize <- log(prime.poisson.df$size)
means <- exp(b0 + b1*x1 + b2*x2 + b3*x3 + logSize)

#Create Empty Vector of est.betas/mean/dev
est.b0 <- numeric(Nsim)
est.b1 <- numeric(Nsim)
est.b2 <- numeric(Nsim)
est.b3 <- numeric(Nsim)
est.mean <- numeric(Nsim)
devs = numeric(Nsim)

#For Loop
for (i in 1:Nsim) {

y_sim = rpois(nr, means)
fit <- vglm(y_sim ~ s(x1) + s(x2) + s(x3),
            family = 'poissonff',
            offset = log(size),
            data = prime.poisson.df)

devs[i] = deviance(fit)
est.b0[i] <- coef(fit)[1]
est.b1[i] <- coef(fit)[2]
est.b2[i] <- coef(fit)[3]
est.b3[i] <- coef(fit)[4]

est.mean[i] <- exp(est.b0[i] + est.b1[i]*x1 + est.b2[i]*x2 + est.b3[i]*x3 + logSize)

}
```

```r
hist(devs, freq = FALSE)
lines(density(devs), col="orange")
ds <- seq(min(devs), max(devs), length= 1000)
lines(ds, dchisq(ds, df=996), col="blue")
legend("topright", c("Deviance Distribution",
                     "Asymptotic Distribution"),
       col=c("orange", "blue"), lwd=8)
```



The distribution of the deviance is slightly different from the asymptotic chi-squared distribution (χ^2_{996}).

The asymptotic χ^2_{996} density curve is to the left relative to the deviance curve. This suggests the deviance estimates do not follow a chi-sq distribution and out GOF deviance test is inappropriate. Our deviance test-stat is likely even smaller than we estimated.

This discrepancy in curves is likely due to the sparsity in our response (many observations are smaller than 5).

j)

```
#Simulation GOF
round(mean(devs > deviance(vglm.pois.fit)), 3)
```

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

We have a simulated deviance test statistic of 1.

k)

Inverting the confidence intervals for β_0

```
a = b0 - quantile(est.b0, prob=0.025)
b = quantile(est.b0, prob=0.975) - b0
exp(c(b0-b, b0+a))
```

```
## (Intercept) (Intercept)
##   0.1766769   0.2648526
```

The bootstrap confidence interval for β_0 is **(0.177, 0.265)**.

l)

Inverting the confidence interval for y

```
#Add new data
new_data_df$dl.ratio <- log(log(new_data_df$ratio))
new_data_df$dl.ratio2 <- new_data_df$dl.ratio^2

#Fit X's
x1New <- new_data_df$center
x2New <- new_data_df$dl.ratio
x3New <- new_data_df$dl.ratio2
logSizeNew <- log(new_data_df$size)

#Generate Observed Mean
meansNew <- exp(b0 + b1*x1New + b2*x2New + b3*x3New + logSizeNew)

#Bootstrap mean
est.meansNew <- exp(est.b0 + est.b1*x1New +  est.b2*x2New  +  est.b3*x3New + logSizeNew)

#Generate Inverted Newidence interval
a = meansNew - quantile(est.meansNew, prob=0.025)
b = quantile(est.meansNew, prob=0.975) - meansNew
(c(meansNew-b, meansNew+a))
```

```
## (Intercept) (Intercept)
##    71.37645   116.52344
```

The mean number of primes in the sequence of consecutive numbers (with the data specified) is **(71.38, 116.52)**.

m)

Generating prediction interval

Noting the following formulas

$log(\mu_i) = \beta_0 + \beta_1 \times center_i + \beta_2 \times dl.ratio_i + \beta_3 \times dl.ratio_i^2$

$Y \sim Poisson(\mu_i)$

```
#Generate Bootstrap Y
mu <- est.meansNew
simY <- rpois(1000, lambda = mu)

a = meansNew - quantile(simY, prob=0.025)
b = quantile(simY, prob=0.975) - meansNew
(c(meansNew-b, meansNew+a))
```

```
## (Intercept) (Intercept)
##    64.94229   121.94229
```

The prediction interval for the mean number of primes in the sequence of consecutive numbers specified is **(64.94, 121.94)**.

Question 2)

Loading Data...

```
source("prime.logistic.data.generator.R")
```

```
## The first ten prime numbers: 2 3 5 7 11 13 17 19 23 29
```

```
detach("package:VGAM", unload = TRUE)
library(gam)
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.20
```

a)

Fitting a GAM

```
quasibinomial.fit.gam = gam(cbind(y.binomial,n.binomial-y.binomial)~cramer*s(center),
                            family = 'quasibinomial', data=prime.logistic.df)

summary(quasibinomial.fit.gam)
```

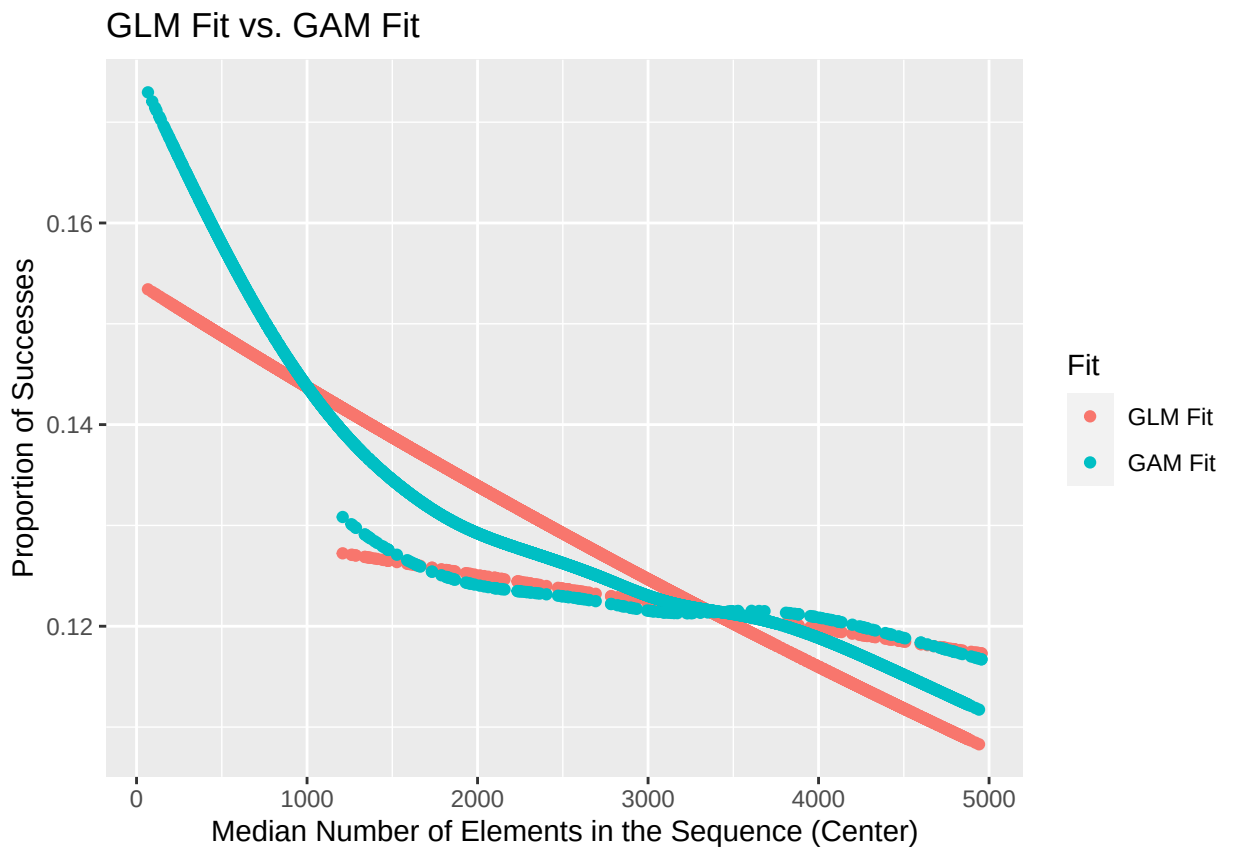
```
##
## Call: gam(formula = cbind(y.binomial, n.binomial - y.binomial) ~ cramer *
##     s(center), family = "quasibinomial", data = prime.logistic.df)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.05114 -0.25692 -0.02104  0.25158  1.49255
##
## (Dispersion Parameter for quasibinomial family taken to be 0.1714)
##
##     Null Deviance: 1309.293 on 2354 degrees of freedom
## Residual Deviance: 406.351 on 2348 degrees of freedom
## AIC: NA
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##                    Df Sum Sq Mean Sq   F value    Pr(>F)
## cramer              1   4.66    4.66   27.1693 2.027e-07 ***
## s(center)           1 788.90  788.90 4602.5519 < 2.2e-16 ***
```

```
## cramer:s(center)    1   1.38    1.38    8.0441  0.004604 **
## Residuals        2348 402.46    0.17
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                  Npar Df Npar F     Pr(F)
## (Intercept)
## cramer
## s(center)              3 239.09 < 2.2e-16 ***
## cramer:s(center)
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The anova tests for non-parametric effects has a p-value ≈ 0, this suggests a non-linear function of center will improve the model.

b)

```
quasibinomial.fit.glm = glm(cbind(y.binomial,n.binomial-y.binomial)~cramer*center,
                            family = 'quasibinomial', data=prime.logistic.df)


ggplotCompare <- prime.logistic.df%>%
  select(c( cramer, center))
ggplotCompare$`GLM Fit` <- fitted(quasibinomial.fit.glm)
ggplotCompare$`GAM Fit` <- fitted(quasibinomial.fit.gam)

ggplotCompare <- melt(ggplotCompare, id.vars = c('center', 'cramer'),
                      variable.name = 'Fit',
                      value.name = 'Prob')

ggplot(data = ggplotCompare,
       aes(center)) +
  geom_point(aes(x= center, y = Prob, colour = Fit))+
  xlab("Median Number of Elements in the Sequence (Center)")+
  ylab("Proportion of Successes")+
  ggtitle("GLM Fit vs. GAM Fit")
```



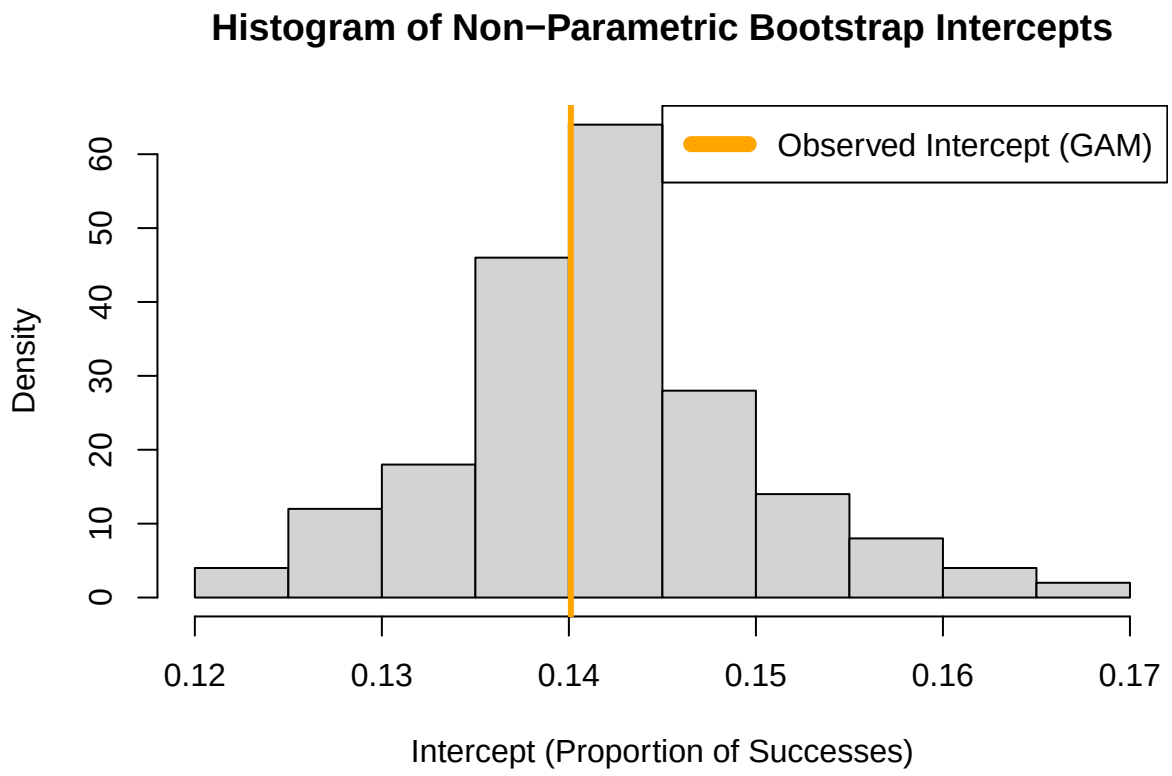
The GAM fit has a distinct non-linear effects that are absent from the GLM fit, this is evident for both when cramer is true and false.

c)

Non-Parametric Bootstrapping

```
n = nrow(prime.logistic.df)

Nsim = 100
est.beta0 <- numeric(Nsim)
est.p <- numeric(Nsim)

for (i in 1:Nsim) {
# Draw a random sample (with replacement) from these data:
id = sample(1:n, n, replace = TRUE)
BS.df = prime.logistic.df[id, ] # Bootstrap sample

mod_i = gam(cbind(y.binomial,n.binomial-y.binomial)~cramer*s(center),
                                family = 'quasibinomial', data=BS.df)
est.beta0[i] = coef(mod_i)[[1]]
est.p[i] <- predict(mod_i, BS.df)

}
```

```r
hist(plogis(est.beta0), freq = F, xlab = "Intercept (Proportion of Successes)",
     main = "Histogram of Non-Parametric Bootstrap Intercepts")
abline(v = plogis(coef(quasibinomial.fit.gam)[[1]]), col = "orange", lwd=3)
legend("topright", c("Observed Intercept (GAM)"),
       col=c("orange"), lwd=8)
```



Determining whether to use parametric of non-parametrics bootstraping comes down to which emulates the experiment better.

This experiment is to determine the proportion of elements in a sequence of consecutive numbers that are prime. Given elements can NOT repeat, this experiment is better emulated by parametric bootstraping, as opposed to non-parametric bootstrapping

d)

Non-parametric confidence interval

```r
#Fit X's

#Generate Observed Mean
pNew <- plogis(predict(quasibinomial.fit.gam, newdata = new_data_df))

#Generate Bootstrap Mean
est.p <- plogis(est.p)
```

```r
#Generate Inverted Newidence interval
a = pNew - quantile(est.p, prob=0.025)
b = quantile(est.p, prob=0.975) - pNew
(c(pNew-b, pNew+a))
```

```
##         1         1
## 0.1904545 0.2324931
```

The confidence interval for the proportion of primes in the sequence of consecutive numbers for which the corresponding explanatory variables are given by new.logistic.df.txt. is **(0.190, 0.232)**

e)

```r
#Generate Random Variables
yVal <- rbinom(n= 10000, size = 10000, est.p)

#Generate Proporotions
ySim <- yVal/10000

#Gnereate Prediction Intervals
a = pNew - quantile(ySim, prob=0.025)
b = quantile(ySim, prob=0.975) - pNew
(c(pNew-b, pNew+a))
```

```
##         1         1
## 0.1889938 0.2346963
```

The prediction interval for the proportion of primes in the sequence of consecutive numbers for which the corresponding explanatory variables are given by new.logistic.df.txt. is **(0.189, 0.235)**.

f)

The prediction interval is wider because it includes the error term.

For a basic model the confidence interval is for μ, where $\mu_i = \beta_0 + \beta_1 \times x_i$.

The prediction interval is for Y, where $Y = \mu_i + \epsilon_i$, where ϵ_i refers to the error term for the ith observation.

This extra term (ϵ) increases the prediction interval relative to the confidence interval.