Data simulation

Luis Rojo-González

2020-02-05

Contents

List of Figures

List of Tables

1 Description of process for fitting input distribution

1.1 Statistical summary: graphical and moments

We can see that the input data has 4041 observations which are integers with 6 possible values (levels) with a negative exponential behavior between 0 and 5. Also, sample moments show that related function has a positive asymmetry, with a first order moment and a second one of 0.5543 and 0.9013, respectively; whereas summary shows a median and a mean equal to 0 and 0.5543, respectively, and a standard deviation of 0.7708 and a coefficient of variability (CV) of 0.7191.

Also, Table 1 shows absolute frequencies, Table 2 shows the summary of the data, Table 3 shows the first two sample moments and Figure 1 shows the behavior of the input data with observed frequencies.

```
#Importa datos
Data = data.frame(x = read_csv("Data24.txt"))
nrow(Data)
```

[1] 4041

```
sd(Data$x)
```

[1] 0.770805

```
mean(Data$x)/sd(Data$x)
```

[1] 0.719142

```
print(xtable(t(table(Data)), digits = 0, label = "tab:frequency",
             caption="Frequencies for each level in input data."),
             caption.placement = "top")
```

Table 1: Frequencies for each level in input data.

	0	1	2	3	4	5
1	2373	1205	373	72	17	1

```
print(xtable(t(summary(Data)), digits = 5, label = "tab:summary",
             caption="Summary of the input data."),
             caption.placement = "top")
```

Table 2: Summary of the input data.

	V1	V2	V3	V4	V5	V6
x	Min. :0.0000	1st Qu.:0.0000	Median :0.0000	Mean :0.5543	3rd Qu.:1.0000	Max. :5.0000

```
print(xtable(all.moments(Data, order.max = 2, central=FALSE),
             digits = 5, label = "tab:moments",
             caption ="First two sample moments of the input data."),
             caption.placement = "top") #moments
```

Table 3: First two sample moments of the input data.

	x
1	1.00000
2	0.55432
3	0.90126

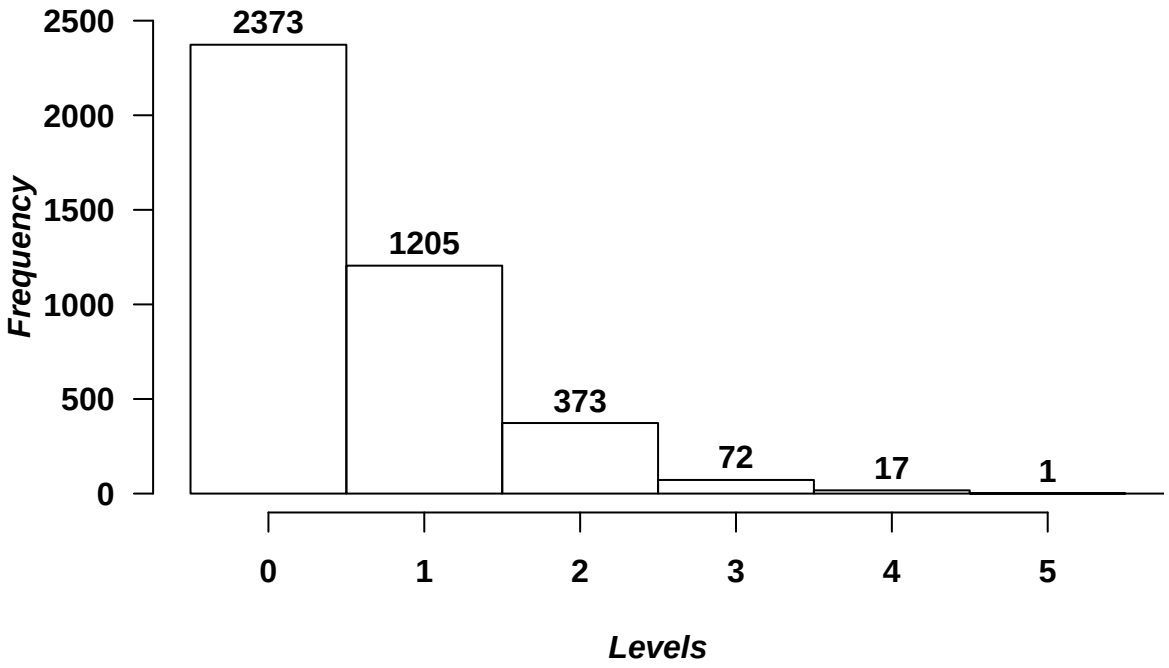

Figure 1: Histogram of input data.

```
par(mfrow = c(1, 1), las = 1, font = 2, font.lab = 4, font.axis = 2,
    cex = 1)
histogram = hist(Data$x, breaks = seq(-0.5, 6.5, by = 1),
                 xlab = "Levels", main = "", freq = TRUE,
                 ylim=c(0,2500), xlim = c(-0.5,5.5),
                 ylab = "Frequency")
text(histogram$mids, histogram$counts, histogram$counts,
     adj = c(0.5,-0.5), font = 2)
```

1.2 Selection of candidate distributions: matching theoretical moments to sample moments. Discussion based on histogram assessment

According to previous analysis we are able to discrimine those distributions where data comes from performing the analysis shown in Table 1.2

Table 4: Analysis of possible distributions.

Distribution	Status	Observation
Bernoulli	Rejected	This distributions works only with one observation of two possible outputs.
Binomial	Rejected	This distributions works only with possible outputs.
Hypergeometric	Rejected	We are not able to recorgnize proportions of classes in input.
Negative binomial	Possible	
Discrete uniform	Rejected	Data is not uniform.
Poisson	Possible	

According to that analysis we have two possible distributions to fit our data: i) Negative binomial and ii) Poisson with probability functions:

$$X \sim NB(r,p) \mapsto P(X=k|r,p) = \binom{k+r-1}{k}(1-p)^r p^k \tag{1}$$

where r represents how much events happends with probability p of occurence, $p \in \{0,1\}$.

$$X \sim Pois(\lambda) \mapsto P(X=k|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \tag{2}$$

where λ is the parameter that measure events per time period.

Another analysis which is convenient to perform is based on histogram visualization, which gives us the behaviour of the data and, combined with moments obtained above, should clarify even more the possible distribution.

On the this way, for these two possible candidates their respectively moments are, for Negative binomial (note that we are considering two first sample moments):

$$\bar{x} = \frac{r(1-p)}{p}, \quad \bar{x^2} = \frac{r(1-p)}{p^2} \tag{3}$$

and, for Poisson:

$$\bar{x} = \lambda \tag{4}$$

In case of Poisson distribution it is easy to compute which is the value of the parameter that is equal to the first moment value, i.e. the mean value of the data, so we can say that $\hat{\lambda}_{MM} = 0.5543$. Whereas for Negative binomial distribution, we can use both first sample moment and second sample moment which give us an equation system which results are: $r = 7.7188$ and $p = 0.9330$.

```
par(mfrow = c(1, 1), las = 1, font = 2, font.lab = 4,
    font.axis = 2, cex = 1)
histogram = hist(Data$x, breaks = seq(-0.5, 6.5, by = 1),
                 xlab = "Levels", main = "", freq = FALSE,
                 ylim = c(0,0.65), xlim = c(-0.5,5.5),
                 ylab = "Probability")
points(0:5, dpois(0:5, 0.5543), col = 'red', pch = 15)
points(0:5, dnbinom(0:5, 7.7188, 0.9330), col = 'blue', pch = 16)
legend("topright", c("Poisson","Negative binomial "),
       col = c('red','blue'), pch = c(15,16), bty = "n",
       title = "Distribution")
text(histogram$mids, round(histogram$density,5),
     round(histogram$density,5), adj = c(0.5, -1.3), font = 2)
```

In fact, the data provided has a negative exponential behaviour such as we claim above, and using obtained parameters both distributions are quite close to each other and the data such Figure 2. That is to say, both distributions are still candidates.

1.3 Estimation of model parameters: Maximum Likelihood estimation

In this section we perform the parameter estimation by using Maximum Likelihood Estimation (MLE) in each possible distribution. On this way, we consider N iid observations (k_1, k_2, ..., k_N) where the maximum of the log-likelihood function l gives

for Negative binomial (Adamidis, 1999)

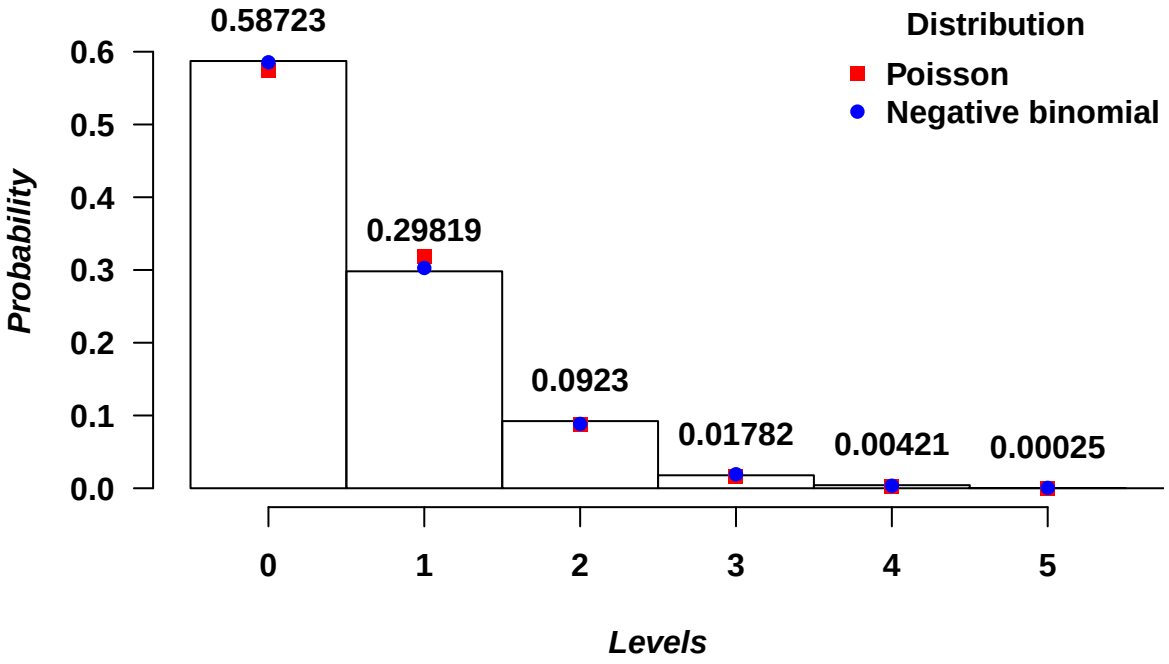

Figure 2: Probability histogram fitted by method of moments.

$$p = \frac{\sum_{i=1}^{N} k_i}{Nr + \sum_{i=1}^{N} k_i}, \quad \frac{\partial l(r,p)}{\partial r} = \left(\sum_{i=1}^{N} \psi(k_i + r) \right) - N\psi(r) + Nln\left(\frac{r}{r + \sum_{i=1}^{N} k_i / N} \right) \tag{5}$$

where $\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$ is the digamma function. And, for Poisson

$$\lambda = \frac{\sum_{i=1}^{N} k_i}{N} \tag{6}$$

Note that there is not a closed expression for parameter r in Negative Binomial distribution, although we can get it through a numerical approximation, we will use the *goodfit* function of the package *vcd* (Friendly & Meyer, 2015) to estimate it.

Also, it is important to say that λ estimator for Poisson distribution is exactly the same by both method of moments (MM) and MLE, which is $\hat{\lambda}_{MM} = \hat{\lambda}_{MLE} = 0.5543$. On the other hand, for Negative binomial parameter estimation we have: $\hat{r}_{MLE} = 7.7158$, $\hat{p}_{MLE} = 0.9330$ and $\hat{\lambda}_{MLE} = 0.5543$.

```
fit = data.frame(f = histogram$counts, x = histogram$mids)[1:6,]
fitting_nbinom = goodfit(fit, type = "nbinom", method= "ML")
cat("r = ", fitting_nbinom$par$size, "\n")
```

```
## r =  7.715794
```

```
cat("p = ", fitting_nbinom$par$prob, "\n")
```

```
## p =  0.9329694
```

```
fitting_poisson = goodfit(fit, type = "poisson", method = "ML")
cat("lambda = ", fitting_poisson$par$lambda, "\n")
```

```
## lambda =  0.5543182
```

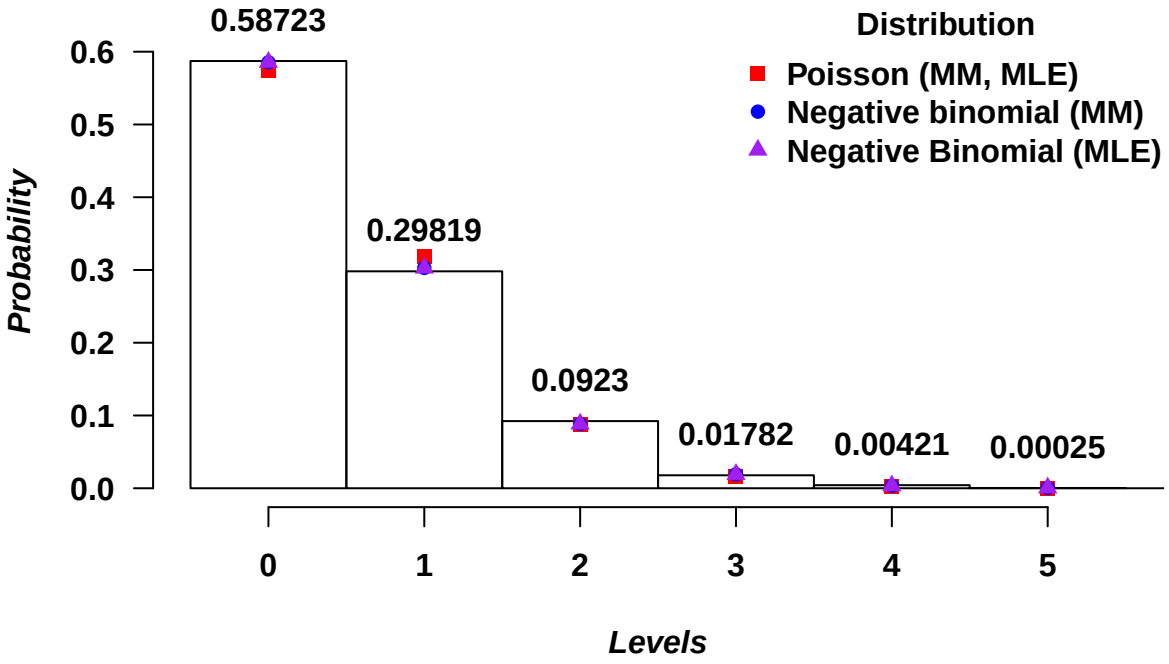

Figure 3: Histogram with fitted probability function overlapped.

With these parameters we are able to plot points which are obtained from their respectively probability functions such as follows:

```
par(mfrow = c(1, 1), las = 1, font = 2, font.lab = 4,
    font.axis = 2, cex = 1)
histogram = hist(Data$x, breaks = seq(-0.5, 6.5, by = 1),
                 xlab = "Levels", main = "", freq = FALSE,
                 ylim = c(0,0.65), xlim = c(-0.5,5.5),
                 ylab = "Probability")
points(0:5, dpois(0:5, 0.5543), col = 'red', pch = 15) #MM
points(0:5, dnbinom(0:5, 7.7188, 0.9330), col = 'blue', pch = 16) #MM
points(0:5, dnbinom(0:5, 7.7158, 0.9330), col = 'purple', pch = 17)
legend("topright", c("Poisson (MM, MLE)",
                     "Negative binomial (MM)",
                     "Negative Binomial (MLE) "),
       col = c('red','blue','purple'), pch = c(15,16,17),
       bty = "n", title = "Distribution")
text(histogram$mids, round(histogram$density,5),
     round(histogram$density,5), adj = c(0.5, -1.3), font = 2)
```

Finally, such as Figure 3 shows probability distributions seem to be suitable to fit the data.

1.4 Goodness of fit: Cramer-von Mises and Chi-Squared Tests

Once possible model and their respectively parameter is already identified, is time to do statistically tests to claim that the data comes from that model. On this way, we perform the Chi-Squared test to compare those models with our data aims to find the best one which is statistically significative whith a threshole of 5% and Cramer-von Mises test. Note that we could perform the Kolmogorov-Smirnov test, but due to we are working on discrete data it does not properly to apply it (Arnold & Emerson, 2011; Massey Jr, 1951), instead of that test we will perform the Cramer-von Mises test (Anderson, 1962; Darling, 1957), which is which is

suitable for discrete data.

As we have one level with a frequency less than 5, we must merge two last rows to accomplish test conditions (Cochran, 1952).

```
# Arrange to have 5 observations at least
fit[5,] = fit[5,]+fit[6,]
fit = fit[1:5,]
```

Thus, Chi-Squared test is defined as follows

$$\chi^2_{obs} = \sum_{i=1}^{N} \frac{(f_i - E_i)^2}{E_i} \sim \chi^2_{k-t-1} \tag{7}$$

where f_i and E_i define relative frequency and expected frecuency, respectively; whereas k is the number of levels and t is the number of estimated parameters. Also, we define the null hypothesis as $H_o : X \sim f(x|\theta)$ (i.e. the variable distributes according to function f with parameter θ) and the confidence region (CR) as $CR = \{X|\chi^2_{obs} < \chi^2_{k-t-1,1-\alpha}\}$. So, if χ^2_{obs} belongs to CR we do not have proofs to refuse the null hyphothesis.

And, Cramer-von Mises test is

Let two samples $x_1, x_2, \ldots, x_N$ and $y_1, y_2, \ldots, y_M$ in increasing order. Be $r_1, r_2, \ldots, r_N$ the ranks of x in combined sample, and be $s_1, s_2, \ldots, s_M$ the ranks of y in combined sample

$$T = N\omega^2 = \frac{U}{NM(N+M)} - \frac{4NM-1}{6(M+N)}, \quad U = N\sum_{i=1}^{N}(r_i - i)^2 + M\sum_{j=1}^{M}(s_j - j)^2 \tag{8}$$

then if T is greather than tabulated, samples do not com from the same distribution.

```
# Crea columnas con valores esperados por cada funcion
fit$Pois = dpois(0:4, 0.5543) #Poisson MM
fit$Pois[5] = 1-sum(fit$Pois[1:4]) # suma cola derecha
fit$Pois = fit$Pois*4041 # frecuencia absoluta
fit$NB.MM = dnbinom(0:4, 7.7188, 0.9330) #NB MM
fit$NB.MM[5] = 1-sum(fit$NB.MM[1:4]) # suma cola derecha
fit$NB.MM = fit$NB.MM*4041
fit$NB.MLE = dnbinom(0:4, 7.7158, 0.9330) #NB MLE'
fit$NB.MLE[5] = 1-sum(fit$NB.MLE[1:4]) # suma cola derecha
fit$NB.MLE = fit$NB.MLE*4041
fit$x = c(0:3,">4")

print(xtable(fit,
      caption = "Histogram of input data and expected values.",
      align = 'crcccc', digits = 5, label = "tab:label"),
      caption.placement = "top")
```

Table 5: Histogram of input data and expected values.

	f	x	Pois	NB.MM	NB.MLE
1	2373	0	2321.45045	2365.98116	2366.47345
2	1205	1	1286.77999	1223.58987	1223.36880
3	373	2	356.63107	357.38588	357.19837
4	72	3	65.89353	77.57175	77.50712
5	18	>4	10.24495	16.47133	16.45226

```
# ---------------- Poisson--------
# Test chi-squared
chi.sq.pois = with(fit, sum(((f-Pois)^2)/(Pois)))
chis.sq.pois.stat = qchisq(0.95, 5-1-1)
cat("Chi-Squared test \n")
```

Chi-Squared test

```
cat("Poisson probability \n", "Statistic", chi.sq.pois,
    "Critical value", chis.sq.pois.stat, "\n")
```

Poisson probability Statistic 13.52963 Critical value 7.814728

```
(cvm.pois = cvm.test(Data$x,
                     ecdf(rep(c(0:5),
                              round(dpois(0:5, 0.5543)*4041, 1)))))
```

Cramer-von Mises - W2

data: Data$x W2 = 0.33289, p-value = 0.09506 alternative hypothesis: Two.sided

```
# ------------ Negative binomial (moments) -------
chi.sq.nb.mm = with(fit, sum(((f-NB.MM)^2)/(NB.MM)))
chis.sq.nb.mm.stat = qchisq(0.95, 5-2-1)
cat("Chi-Squared test \n")
```

Chi-Squared test

```
cat("Negative binomial probability (MM) \n", "Statistic",
    chi.sq.nb.mm, "Critical value", chis.sq.nb.mm.stat, "\n")
```

Negative binomial probability (MM) Statistic 1.527507 Critical value 5.991465

```
(cvm.nb = cvm.test(Data$x,
                   ecdf(rep(c(0:5),
                            round(dnbinom(0:5,
                                          7.7188,
                                          0.9330)*4041, 1)))))
```

Cramer-von Mises - W2

data: Data$x W2 = 0.012146, p-value = 0.879 alternative hypothesis: Two.sided

```
# ------------ Negative binomial (likelihood) ---------
# Test chi-squared
chi.sq.nb.mle = with(fit, sum(((f-NB.MLE)^2)/(NB.MLE)))
chis.sq.nb.mle.stat = qchisq(0.95, 5-2-1)
cat("Chi-Squared test \n")
```

Chi-Squared test

```
cat("Negative binomial probability (MLE) \n", "Statistic",
    chi.sq.nb.mle, "Critical value",
    chis.sq.nb.mle.stat, "\n")
```

Negative binomial probability (MLE) Statistic 1.529735 Critical value 5.991465

```
(cvm.nb.1 = cvm.test(Data$x,
                     ecdf(rep(c(0:5),
                              round(dnbinom(0:5,
```

```
                                        7.7158,
                                        0.9330)*4041,1)))))
```

```
Cramer-von Mises - W2
```

data: Data$x W2 = 0.012146, p-value = 0.879 alternative hypothesis: Two.sided

According to this, tests support the Negative binomial model as the one which is statistically robust to the data with a 5% of significance[1]. Now, the problem is such that we have two possible models, nevertheless we can use the average value of them because of their parameters are quite closer to each other, it means they are practically the same. Finally, we can say that our data distributes as a Negative binomial such that $X \sim NB(r = 7.7173,\ p = 0.9330)$, where considered r is the mean of the estimations by both methods.

2 Generation of random variables

2.1 Proposed algorithm for the generation of samples for the distribution fitted/reported in Deliverable 1-Part 1

To generate a Negative binomial random variable we use the fact that this is a sum of r independent Geometric random variables, fact which is really convenient to avoid the combinatorial number of the Negative binomial distribution. So, as we get above in section 1.4, we have parameters $r = 7.7173$[2] and $p = 0.9330$. Also, to generate Geometric random variable we will use the Inverse Transformation method, for which we will need a uniform random variable between 0 and 1 and its cumulative density function (cdf).

First of all, the cdf of a Geometric distribution is

$$P(X \leq x) = 1 - (1-p)^{x+1} \tag{9}$$

where $x \in \{0, 1, 2, \dots\}$. Thus, for Inverse Transformation method we have to use a uniform random number u, $u \sim Uniform$, and the cdf defined by equation (9) such that

$$x = \frac{ln(1-u)}{ln(1-p)} - 1, x^* = \sum_{i=1}^{r} x \tag{10}$$

where x is the generated geometric number and x^* the sum of r of that numbers. Also, we consider the range of our data to impose a filter such that when generated number is discarded when it is out of that bounds.

```
set.seed(12345)
nb.sim = function(n, r, p, lim.inf, lim.sup, u.inf, u.sup){
  sim = data.frame()
  contador = 0
  while (contador < n) {
    u = runif(r, u.inf, u.sup)
    x = sum(round((log(1-u)/log(1-p))-1,0))
    if(x %in% lim.inf:lim.sup){
      sim[contador+1,1] = x
      contador = contador + 1
    } else{contador = contador}
  }
```

[1] Note that although for Poisson model Cramer-von Mises test says that we do not have evidence to refuse the null hyphotesis Chi-Squared test does.

[2] On this case r must be an integer number but we hold it as it is to keep working, but considering that this number will be rounded to the nearest integer. i.e. $r = 7$.

```
  return(sim)
}
```

Although we have a proposed algorithm to generate our sample according to our fitted distribution, we depend on parameter u which is an uniform random number. Nevertheless, as we stated above, we know that the input data has a negative exponential behavior, for which we could find an uniform random number u but a convenient one. That it to say, let equation 10 which generate through the inverse transformation method and let some e^{-u} such that our generator has a negative exponential behavior. That is

$$e^{-u} = \frac{ln(1-u)}{ln(1-p)} - 1 \Longleftrightarrow p = 1 - e^{\frac{ln(1-u)}{e^{-u}+1}}$$

and therefore, as our $p = 0.9330$, we can see that $u \to 1$. That is to say, we should generate an uniform random number on the premise that it has to be *reasonable big* to get a negative exponential behavior such as Figure 4 shows.

```
par(mfrow = c(2, 3), las = 1, font = 2, font.lab = 4,
    font.axis = 2, cex = 1)
instances = seq(from = 0, to = 0.9,  by = 0.18)
experiment = data.frame(c(1:1000))
for (i in instances) {
  experiment = cbind(experiment,
                     nb.sim(1000, 7.7173, 0.9330, 0, 5, i, 1))
}
plot(as.vector(table(experiment[,2])), col = 'black', pch = 16,
     xlab = "Levels", ylab = "Frequency",
     main = paste("0")) # 0,1
plot(as.vector(table(experiment[,3])), col = 'black', pch = 16,
     xlab = "Levels", ylab = "Frequency",
     main = paste("0.18")) # 0.18,1
plot(as.vector(table(experiment[,4])), col = 'black', pch = 16,
     xlab = "Levels", ylab = "Frequency",
     main = paste("0.36")) # 0.36,1
plot(as.vector(table(experiment[,5])), col = 'black', pch = 16,
     xlab = "Levels", ylab = "Frequency",
     main = paste("0.54")) # 0.54,1
plot(as.vector(table(experiment[,6])), col = 'black', pch = 16,
     xlab = "Levels", ylab = "Frequency",
     main = paste("0.72")) # 0.72,1
plot(as.vector(table(experiment[,7])), col = 'black', pch = 16,
     xlab = "Levels", ylab = "Frequency",
     main = paste("0.9")) # 0.9,1
```

2.2 Exploratory Data Analysis for a sample of size 5000 following the fitted distribution illustrating a correct behavior of the proposed generator

On the other hand, when we have already defined the model with we will work; an algorithm to generate a random variable which distributes as the proposed model is performed by using the inverse transformation method and those parameters previously estimated, where an analysis of the necessary parameters are carried out finding that for a correct behavior we need an specific bound for uniform random variable. It was proposed a value for that bound based on the coefficient of variability of the input data giving a correct behavior, even although we could work with another and get best results, such Figure 6 shows with a bound of 0.75 instead of 0.7191 used.

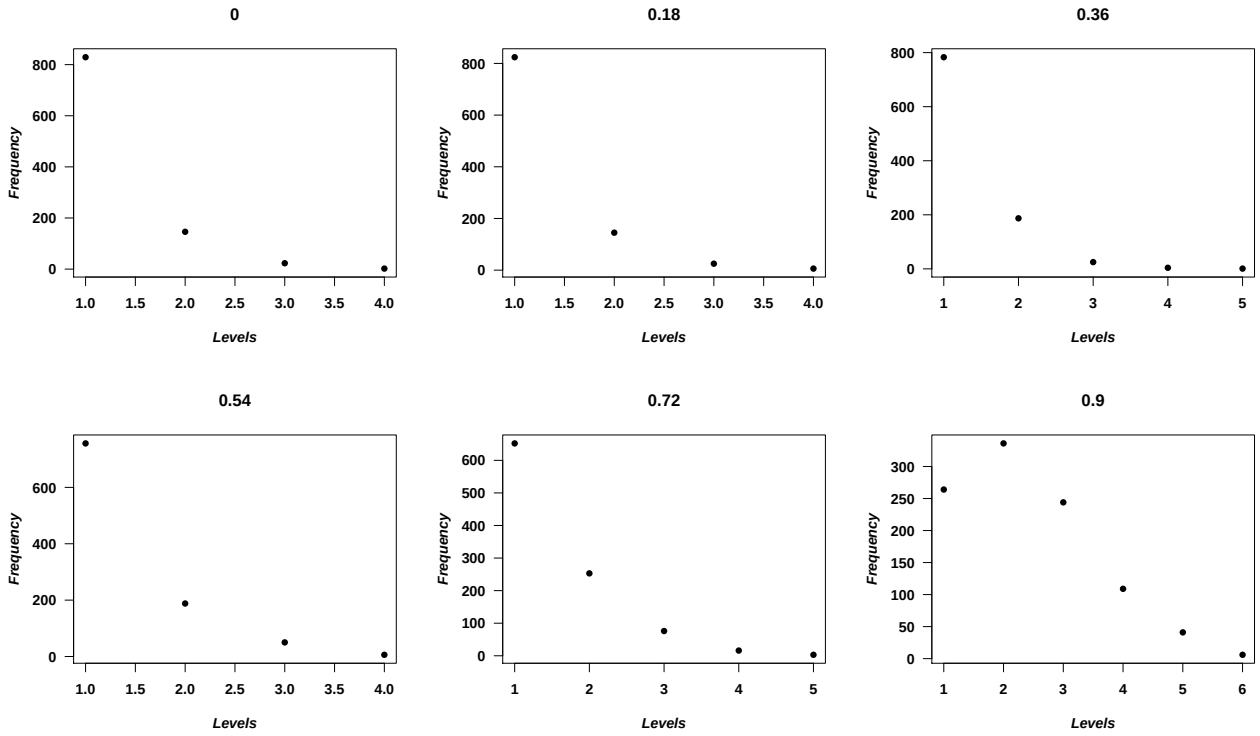

Figure 4: Experiments over lower bound of an uniform random variable to generate a Negative binomial variable with 1000 observations. Titles represent the lower bound of the considered uniform random variable.

```
n.sim = 5000
lim.inf = 0
lim.sup = 5
u.inf = 0.7191
u.sup = 1
r.sim = 7.7173
p.sim = 0.9330

sim = nb.sim(n.sim, r.sim, p.sim, lim.inf, lim.sup, u.inf, u.sup)
```

```
par(mfrow = c(1, 2), las = 1, font = 2, font.lab = 4,
    font.axis = 2, cex = 1)
histogram.sim = with(sim, hist(V1,
                               breaks = seq(min(V1)-0.5,
                                            max(V1)+5, by = 1),
                               xlab = "Levels", main = "",
                               freq = TRUE,
                               ylim = c(0,0.8*n.sim),
                               xlim = c(min(V1)-0.5,max(V1)+0.5),
                               ylab = "Frequency"))
text(histogram.sim$mids, round(histogram.sim$counts,5),
     round(histogram.sim$counts,5), adj = c(0.5, -1), font = 2)
histogram.sim = with(sim, hist(V1,
                               breaks = seq(min(V1)-0.5,
                                            max(V1)+5, by = 1),
                               xlab = "Levels", main = "",
```

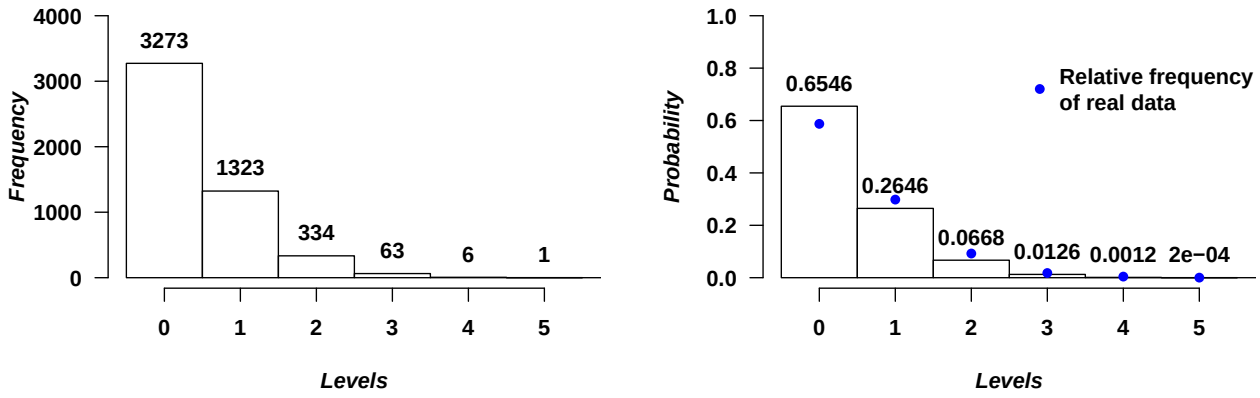

Figure 5: Histogram of simulated data with a lower bound to u of 0.7191.

```
                                freq = FALSE, ylim = c(0,1),
                                xlim = c(min(V1)-0.5,max(V1)+0.5),
                                ylab = "Probability"))
text(histogram.sim$mids, round(histogram.sim$density,5),
     round(histogram.sim$density,5), adj = c(0.5, -1), font = 2)
points(0:6, histogram$density, col = 'blue', pch = 16) #Real
legend("topright", c("Relative frequency \nof real data"),
       col = c('blue'), pch = c(16), bty = "n", title = "")
```

```
print(xtable(t(table(sim)), digits = 0,
             label = "tab:frequency.sim",
            caption="Frequencies for each level in generated data."),
      caption.placement = "top")
```

Table 6: Frequencies for each level in generated data.

	0	1	2	3	4	5
1	3273	1323	334	63	6	1

```
print(xtable(t(summary(sim)), digits = 5,
             label = "tab:summary.sim",
             caption="Summary of the generated data."),
      caption.placement = "top")
```

Table 7: Summary of the generated data.

	V1	V2	V3	V4	V5	V6
V1	Min. :0.0000	1st Qu.:0.0000	Median :0.0000	Mean :0.4418	3rd Qu.:1.0000	Max. :5.0000

```
print(xtable(all.moments(sim, order.max = 2, central=FALSE),
             digits = 5, label = "tab:moments.sim",
       caption ="First two sample moments of the generated data."),
      caption.placement = "top") #moments
```

```
sd(sim$V1)
```

[1] 0.6887

Table 8: First two sample moments of the generated data.

	x
1	1.00000
2	0.44180
3	0.66940

```
mean(sim$V1)/sd(sim$V1)
```

[1] 0.6414985

As we can see, proposed algorithm has a correct behavior according to input data. Even more, both samples has a similar mean value, 0.5543 for input data and 0.4418 for generated one. Nevertheless, the standard deviation and the CV of the generated variable are less than the input data values.

3 Conclusions

This work presents an input data analysis and their respectively fitting process to a known probability model according to the kind of the input data. Throughout the process two possible models were found by a sample moments matching and histograms analysis. Those fitted models showed a good behavior related to input data, nonetheless just one of them was statistically correct according to Chi-Squared and Cramer-von Mises goodness-of-fit tests.

On the other hand, when we have already defined the model with we will work; an algorithm to generate a random variable which distributes as proposed model is performed by using the inverse transformation method and those parameters previously estimated, where an analysis of the necessary parameters are revised finding that for a correct behavior we need an specific bound for uniform random variable to carried out our algorithm. It was proposed a value for that bound based on the coefficient of variability of the input data giving a correct behavior, even although we could work with another and get best results, such as Figure 6 shows with a bound of 0.75 instead of 0.7191 used.

Finally, we can say that a data can comes from many probability distributions because of one distribution can be obtained from another with a suitable parameter setting; but it must be necessary to analyze and consider which kind of data about is, whether it is bounded or not and perform suitable statistic tests to support or to discard our candidates; on the other hand, to generate a random variable it must be necessary consider exploratory analysis to generate it, and how is our distribution constructed is helpful if our model has parts as combinatorial numbers.

```
u.inf.1 = 0.75
sim.2 = nb.sim(n.sim, r.sim, p.sim, lim.inf, lim.sup, u.inf.1, u.sup)
```

```
par(mfrow = c(1, 2), las = 1, font = 2, font.lab = 4,
    font.axis = 2, cex = 1)
histogram.sim.2 = with(sim.2, hist(V1,
                                   breaks = seq(min(V1)-0.5,
                                                max(V1)+5, by = 1),
                                   xlab = "Levels", main = "",
                                   freq = TRUE,
                                   ylim = c(0,0.8*n.sim),
                                   xlim = c(min(V1)-0.5,
                                            max(V1)+0.5),
                                   ylab = "Frequency"))
text(histogram.sim.2$mids, round(histogram.sim.2$counts,5),
     round(histogram.sim.2$counts,5), adj = c(0.5, -1), font = 2)
```

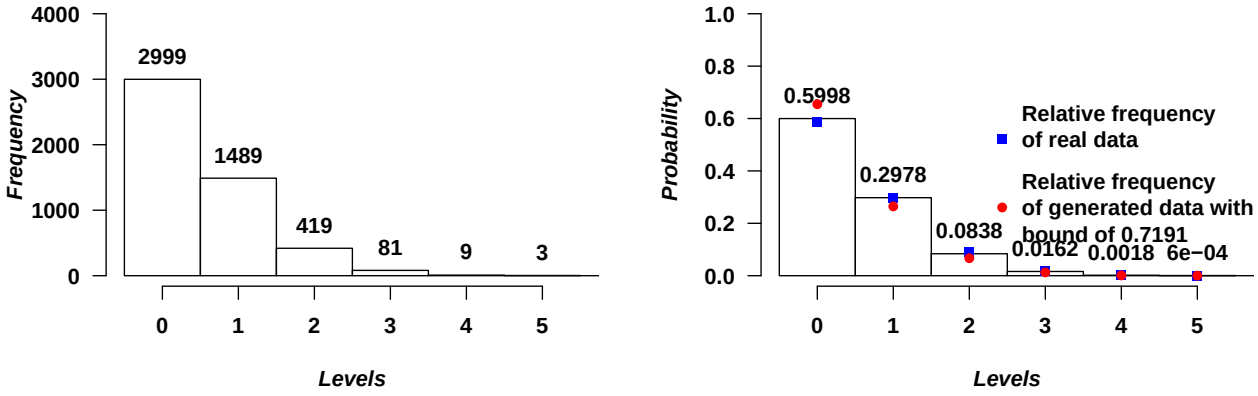

Figure 6: Histogram of simulated data with a lower bound to u of 0.75.

```
histogram.sim.2 = with(sim.2, hist(V1,
                                   breaks = seq(min(V1)-0.5,
                                                max(V1)+5, by = 1),
                                   xlab = "Levels", main = "",
                                   freq = FALSE, ylim = c(0,1),
                                   xlim = c(min(V1)-0.5,
                                            max(V1)+0.5),
                                   ylab = "Probability"))
text(histogram.sim.2$mids, round(histogram.sim.2$density,5),
     round(histogram.sim.2$density,5), adj = c(0.5, -1), font = 2)
points(0:6, histogram$density, col = 'blue', pch = 15) #Real
points(0:9, histogram.sim$density, col = 'red', pch = 16) #simulated1
legend("topright", c("Relative frequency \nof real data\n",
    "Relative frequency \nof generated data with \nbound of 0.7191"),
    col = c('blue','red'), pch = c(15,16), bty = "n", title = "")
```

References

Adamidis, K. (1999). Theory & methods: An em algorithm for estimating negative binomial parameters. *Australian & New Zealand Journal of Statistics*, *41*(2), 213–221.

Anderson, T. W. (1962). On the distribution of the two-sample cramer-von mises criterion. *The Annals of Mathematical Statistics*, 1148–1159.

Arnold, T. B., & Emerson, J. W. (2011). Nonparametric goodness-of-fit tests for discrete null distributions. *R Journal*, *3*(2).

Cochran, W. G. (1952). The $\chi 2$ test of goodness of fit. *The Annals of Mathematical Statistics*, 315–345.

Darling, D. A. (1957). The kolmogorov-smirnov, cramer-von mises tests. *The Annals of Mathematical Statistics*, *28*(4), 823–838.

Friendly, M., & Meyer, D. (2015). *Package vcd*. https://cran.r-project.org/web/packages/vcd/vcd.pdf.

Massey Jr, F. J. (1951). The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, *46*(253), 68–78.