LNL_Course_Proj_Part1

Anupriya Thirumurthy

8/22/2018

1. Problem Description

The business analytics group of a company is asked to investigate causes of malfunctions in technological process of one of the manufacturing plants that result in significant increase of cost for the end product of the business.

One of suspected reasons for malfunctions is deviation of temperature during the technological process from optimal levels. The sample in the provided file contains times of malfunctions in seconds since the start of measurement and minute records of temperature.

```
conditionalEcho<-F
```

2. Data

```
dataPath <- "/Users/anupriyathirumurthy/Documents/University/MScA_UoC/Courses/LinearAndNonLinearModels/I
Course.Project.Data<-read.csv(file=paste(dataPath,"MScA_LinearNonLinear_MalfunctionData.csv",sep="/"))
```

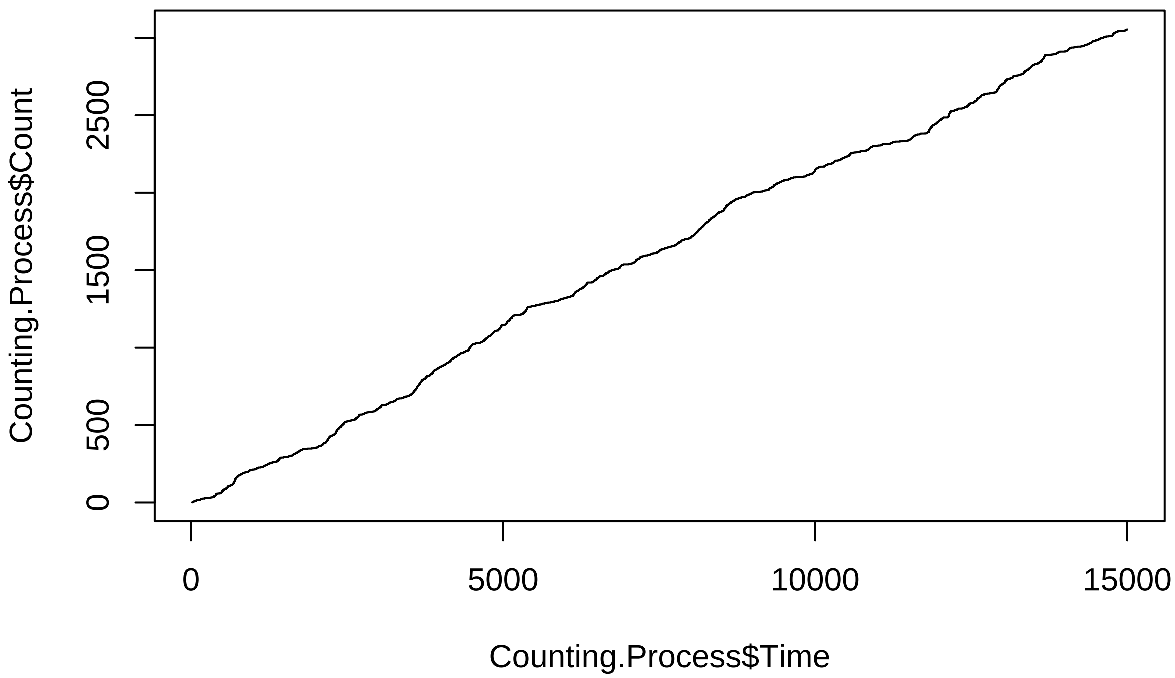
3. Creating Counting Process & Exploring Cumulative Intensity

Counting process is a step function of time that jumps by 1 at every moment of a new malfunction event being logged.

```
Counting.Process<-as.data.frame(cbind(Time=Course.Project.Data$Time,Count=1:length(Course.Project.Data$T
Counting.Process[1:20,]
```

```
##         Time Count
## 1   18.08567     1
## 2   28.74417     2
## 3   34.23941     3
## 4   36.87944     4
## 5   37.84399     5
## 6   41.37885     6
## 7   45.19283     7
## 8   60.94242     8
## 9   66.33539     9
## 10  69.95667    10
## 11  76.17420    11
## 12  80.48524    12
## 13  81.29133    13
## 14  86.18149    14
## 15  91.28642    15
## 16  91.75162    16
## 17  98.29452    17
## 18 142.58741    18
## 19 149.82484    19
## 20 151.58587    20
```

```r
plot(Counting.Process$Time,Counting.Process$Count,type="s")
```



The counting process trajectory looks pretty smooth and grows steadily.

Character of malfunctions and the reasons causing them

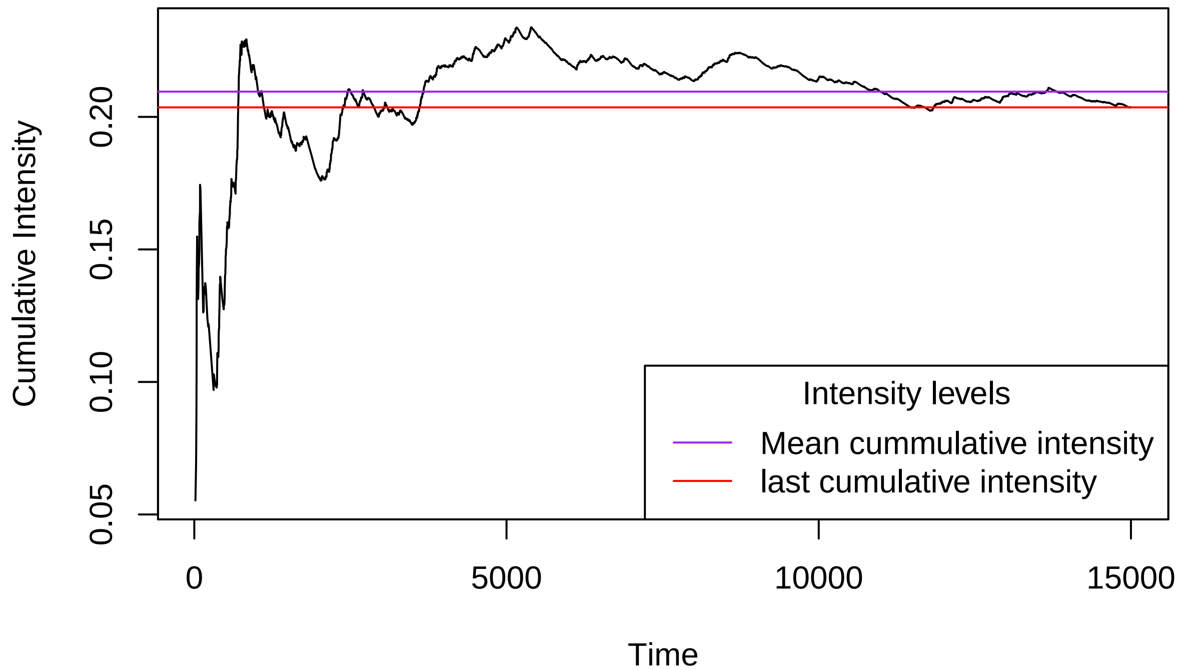
The above graph shows Time Vs Malfunction Counts.

We could infer that the number of malfunctions is directly proportional to the length of the period during which they are observed. The graph has a stead increase. We also notice that the rate at which events occur is constant. These inferences on the counts makes us suspicious that it might be a Poisson model.

3.1 Exploring cumulative intensity of the process.

Cumulative intensity = Nt/t, where, Nt is the number of events during the time interval [0,t] For our data t is the sequence of time stamps and Nt is the count up until t.

```r
plot(Counting.Process$Time,Counting.Process$Count/Counting.Process$Time,type="l",ylab="Cumulative Inten
abline(h=Counting.Process$Count[length(Counting.Process$Count)]/
         Counting.Process$Time[length(Counting.Process$Time)], col = "red")
abline(h=mean(Counting.Process$Count/Counting.Process$Time), col = "purple")
legend("bottomright", legend=c("Mean cummulative intensity", "last cumulative intensity"),
       col=c("purple", "red"), lty=1, cex=1.0, title="Intensity levels")
```



The cumulative intensity seems to converge to a stable level.

```
c(Last.Intensity=Counting.Process$Count[length(Counting.Process$Count)]/
         Counting.Process$Time[length(Counting.Process$Time)],
  Mean.Intensity=mean(Counting.Process$Count/Counting.Process$Time))
```

```
## Last.Intensity Mean.Intensity
##      0.2036008      0.2095305
```

4. Checking for over-dispersion.

Checking for overdispersion by calculating the one-minute event counts and temperatures.

Looking at the first 20 rows of the data.

```
Course.Project.Data[1:29,]
```

```
##          Time Temperature
## 1    18.08567    91.59307
## 2    28.74417    91.59307
## 3    34.23941    91.59307
## 4    36.87944    91.59307
## 5    37.84399    91.59307
## 6    41.37885    91.59307
## 7    45.19283    91.59307
## 8    60.94242    97.30860
## 9    66.33539    97.30860
## 10   69.95667    97.30860
## 11   76.17420    97.30860
## 12   80.48524    97.30860
## 13   81.29133    97.30860
## 14   86.18149    97.30860
## 15   91.28642    97.30860
## 16   91.75162    97.30860
```

```
## 17  98.29452    97.30860
## 18 142.58741    95.98865
## 19 149.82484    95.98865
## 20 151.58587    95.98865
## 21 156.37781    95.98865
## 22 161.97298    95.98865
## 23 172.42610    95.98865
## 24 174.79452    95.98865
## 25 184.07435   100.38440
## 26 209.82744   100.38440
## 27 223.35757   100.38440
## 28 230.02632   100.38440
## 29 252.39556    99.98330
```

```
# The time column is in seconds.
```

Note that the first 7 rows (events) occurred during the first minute. The temperature measurement for the first minute was 91.59307 degree F.

The following 10 rows happen during the second minute and the second minute temperature is 97.3086 degree F.

The third minute had 7 events at temperature 95.98865 degree F.

The fourth minute had 4 events at 100.3844 degree F.

And the following fifth minute had only 1 event at 99.9833 degree F.

Constructing a data frame of one-minute counts and the corresponding temperatures

```
library(magrittr)

suppressMessages(library(plyr))

Minute.Temps <- Course.Project.Data$Temperature
One.Minute.Counts.Temps <-
  Course.Project.Data %>% cbind(Count=Counting.Process$Count) %>%
    ddply( ~Minute.Temps, function(minute_frame) {
      data.frame(Minute.times = unique(floor(minute_frame$Time/60)*60+30), # The column Minute.Times co
                 Minute.counts = nrow(minute_frame))
    })

# Reorder by time recorded
One.Minute.Counts.Temps <- One.Minute.Counts.Temps[order(One.Minute.Counts.Temps$Minute.times), c(2,3,1)

rownames(One.Minute.Counts.Temps) <- NULL
colnames(One.Minute.Counts.Temps) <- c('Minute.times','Minute.counts','Minute.Temps' )

unique(diff(One.Minute.Counts.Temps$Minute.times))
```

```
## [1]  60 120
```

```
ExtraMinutes<- data.frame(Minute.times = NULL, Minute.counts = NULL, Minute.Temps = NULL);

for (i in 2:nrow(One.Minute.Counts.Temps)) {
    if((One.Minute.Counts.Temps$Minute.times[i] -
```

```r
        One.Minute.Counts.Temps$Minute.times[i-1]) != 60) {
      ExtraRow <- data.frame(Minute.times = (One.Minute.Counts.Temps$Minute.times[i] +
                                       One.Minute.Counts.Temps$Minute.times[i-1])/2,
                             Minute.counts = 0,
                             Minute.Temps =NA)
      ExtraMinutes <- rbind(ExtraMinutes, ExtraRow)
    }
}

One.Minute.Counts.Temps <- rbind(One.Minute.Counts.Temps, ExtraMinutes)
One.Minute.Counts.Temps <- One.Minute.Counts.Temps[order(One.Minute.Counts.Temps$Minute.times),]
head(One.Minute.Counts.Temps)
```

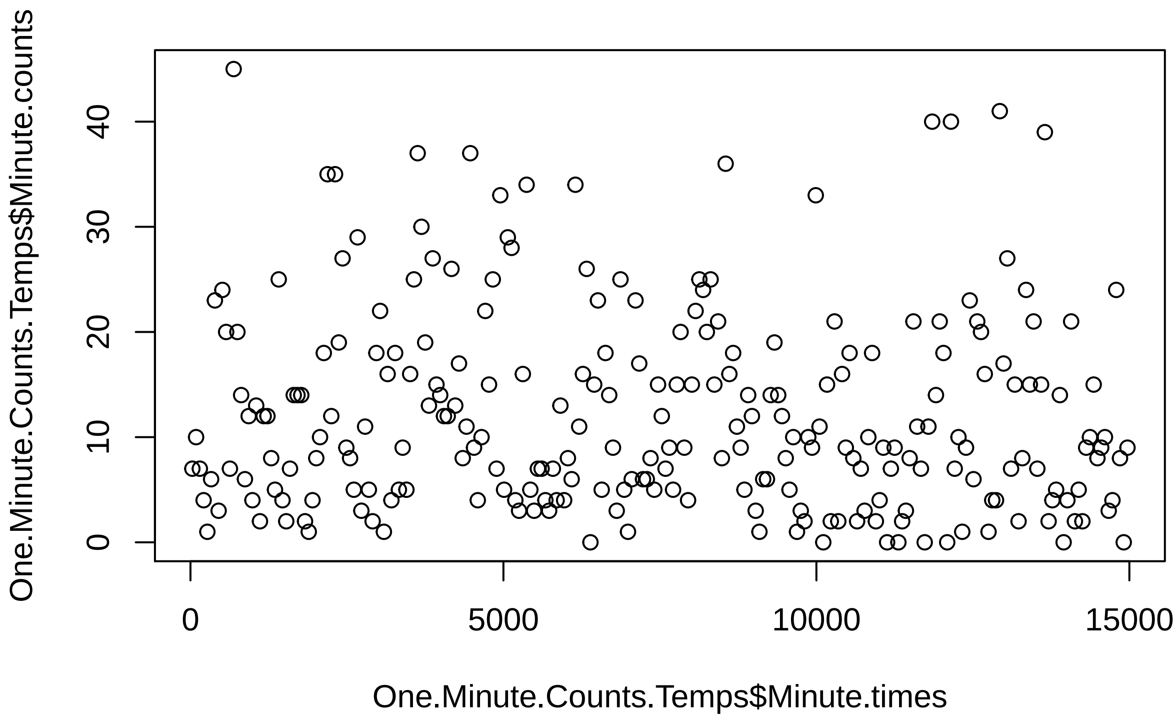
```
##   Minute.times Minute.counts Minute.Temps
## 1           30             7     91.59307
## 2           90            10     97.30860
## 3          150             7     95.98865
## 4          210             4    100.38440
## 5          270             1     99.98330
## 6          330             6    102.54126
```

```r
head(One.Minute.Counts.Temps)
```

```
##   Minute.times Minute.counts Minute.Temps
## 1           30             7     91.59307
## 2           90            10     97.30860
## 3          150             7     95.98865
## 4          210             4    100.38440
## 5          270             1     99.98330
## 6          330             6    102.54126
```

Plot of the One-Minute Counts

```r
plot(One.Minute.Counts.Temps$Minute.times,One.Minute.Counts.Temps$Minute.counts)
```



4.1 Methods for Testing Over-Dispersion

4.1.1 A quick and rough method.

Looking at the output of glm() and comparing the residual deviance with the number of degrees of freedom. If the assumed model is correct, deviance is asymptotically distributed as Chi-squared2 with degrees of freedom n k where n is the number of observations and k is the number of parameters. For a 2 distribution the mean is the number of degrees of freedom n k. If the residual deviance returned by glm() is greater than n k, then it might be a sign of over-dispersion.

Test the method on simulated Poisson data.

The function simulates a sample from Poisson distribution, estimates parameter which is simultaneously the mean value and the variance, then it checks if DevianceDeg.Freedom 1 belongs to the interval (1.96,1.96]. If yes, the result is 1. Otherwise it is 0.

```
Test.Deviance.Overdispersion.Poisson<-function(Sample.Size,Parameter.Lambda){
  my.Sample<-rpois(Sample.Size,Parameter.Lambda)
  Model<-glm(my.Sample~1,family=poisson)
  Dev<-Model$deviance
  Deg.Fred<-Model$df.residual
  (((Dev/Deg.Fred-1)/sqrt(2/Deg.Fred)>-1.96)&((Dev/Deg.Fred-1)/sqrt(2/Deg.Fred)<=1.96))*1
}
Test.Deviance.Overdispersion.Poisson(100,1)
```

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

Now repeat the call of the function 300 times to see how many times it returns one and how many times zero.

```
sum(replicate(300,Test.Deviance.Overdispersion.Poisson(100,1)))
```

```
## [1] 257
```

We see that the Poisson distribution passes the test.

The estimate of the parameter given by glm() is e^Coefficient:

```
exp(glm(rpois(1000,2)~1,family=poisson)$coeff)
```

```
## (Intercept)
##       2.033
```

Perform the same test on negative binomial data

```
Test.Deviance.Overdispersion.NBinom<-function(Sample.Size,Parameter.prob){
  my.Sample<-rnbinom(Sample.Size,2,Parameter.prob)
  Model<-glm(my.Sample~1,family=poisson)
  Dev<-Model$deviance
  Deg.Fred<-Model$df.residual
  (((Dev/Deg.Fred-1)/sqrt(2/Deg.Fred)>-1.96)&((Dev/Deg.Fred-1)/sqrt(2/Deg.Fred)<=1.96))*1
}
sum(replicate(300,Test.Deviance.Overdispersion.NBinom(100,.2)))
```

```
## [1] 0
```

We see that the over-dispersed negative binomial distribution sample never passes the test.

Now apply the test to the one-minute event counts.

```
GLM.model<-glm(One.Minute.Counts.Temps$Minute.counts~1,family=poisson)
GLM.model
```

```
##
## Call:  glm(formula = One.Minute.Counts.Temps$Minute.counts ~ 1, family = poisson)
##
## Coefficients:
## (Intercept)
##       2.503
##
## Degrees of Freedom: 249 Total (i.e. Null);  249 Residual
## Null Deviance:       1799
## Residual Deviance: 1799  AIC: 2789
```

Signs of over-dispersion?

I see signs of over-dispersion because this is not a good model as we can see that the residual deviance is is higher than the number of degrees of freedom.

4.1.2 Regression test by Cameron-Trivedi

The test implemented in AER is described in Cameron, A.C. and Trivedi, P.K. (1990). Regression-based Tests for Over-dispersion in the Poisson Model. Journal of Econometrics, 46, 347 364.

In a Poisson model, the mean is E(Y)=?? and the variance is V(Y)=?? as well. They are equal. The test has a null hypothesis c=0 where Var(Y)=??+c???f(??), c<0 means under-dispersion and c>0 means over-dispersion. The function f(.) is some monotonic function (linear (default) or quadratic). The test statistic used is a t statistic which is asymptotically standard normal under the null.

```
suppressMessages(library(car))
suppressWarnings(library(AER))
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Loading required package: survival
```

```
Disp.Test <- dispersiontest(object = GLM.model, alternative = 'two.sided')
Disp.Test
```

```
##
##  Dispersion test
##
## data:  GLM.model
## z = 8.5747, p-value < 2.2e-16
## alternative hypothesis: true dispersion is not equal to 1
## sample estimates:
## dispersion
##   7.377975
```

Overdispersion?

Yes, the p-value is extremely small, thus we reject the null hypothesis that its poisson(The hypothesis H0 : ???dispersion=1???) and accept the alternative hypothesis of dispresion being not equal to 1. We also see that the estimated dispersion is around 7 confirming that there are signs of over dispersion.

4.1.3 Test against Negative Binomial Distribution

The null hypothesis of this test is that the distribution is Poisson as particular case of Negative binomial against Negative Binomial.

The references are: A. Colin Cameron and Pravin K. Trivedi (1998) Regression analysis of count data. New York: Cambridge University Press. Lawless, J. F. (1987) Negative Binomial and Mixed Poisson Regressions. The Canadian Journal of Statistics. 15:209-225.

Required packages are MASS (to create a negative binomial object with glm.nb) and pscl the test function is odTest.

```
library("MASS")
GLM.model.nb <- glm.nb(One.Minute.Counts.Temps$Minute.counts ~ 1)
summary(GLM.model.nb)
```

```
##
## Call:
## glm.nb(formula = One.Minute.Counts.Temps$Minute.counts ~ 1, init.theta = 1.747516571,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6951  -0.9389  -0.2977   0.4958   2.0931
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  2.50275    0.05115   48.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.7475) family taken to be 1)
##
##     Null deviance: 278.5  on 249  degrees of freedom
## Residual deviance: 278.5  on 249  degrees of freedom
## AIC: 1746.7
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  1.748
##           Std. Err.:  0.179
##
##  2 x log-likelihood:  -1742.697
```

Theta is so small which confirms that its negative binomial and there is overdispersion

```r
suppressMessages(library(pscl))
odTest(GLM.model.nb)
```

```
## Likelihood ratio test of H0: Poisson, as restricted NB model:
## n.b., the distribution of the test-statistic under H0 is non-standard
## e.g., see help(odTest) for details/references
##
## Critical value of test statistic at the alpha= 0.05 level: 2.7055
## Chi-Square Test Statistic =  1044.585 p-value = < 2.2e-16
```

Overdispersion?

Yes, p-value is so small and we reject the hypothesis, this shows overdispersion.

5. Find the distribution of Poisson intensity.

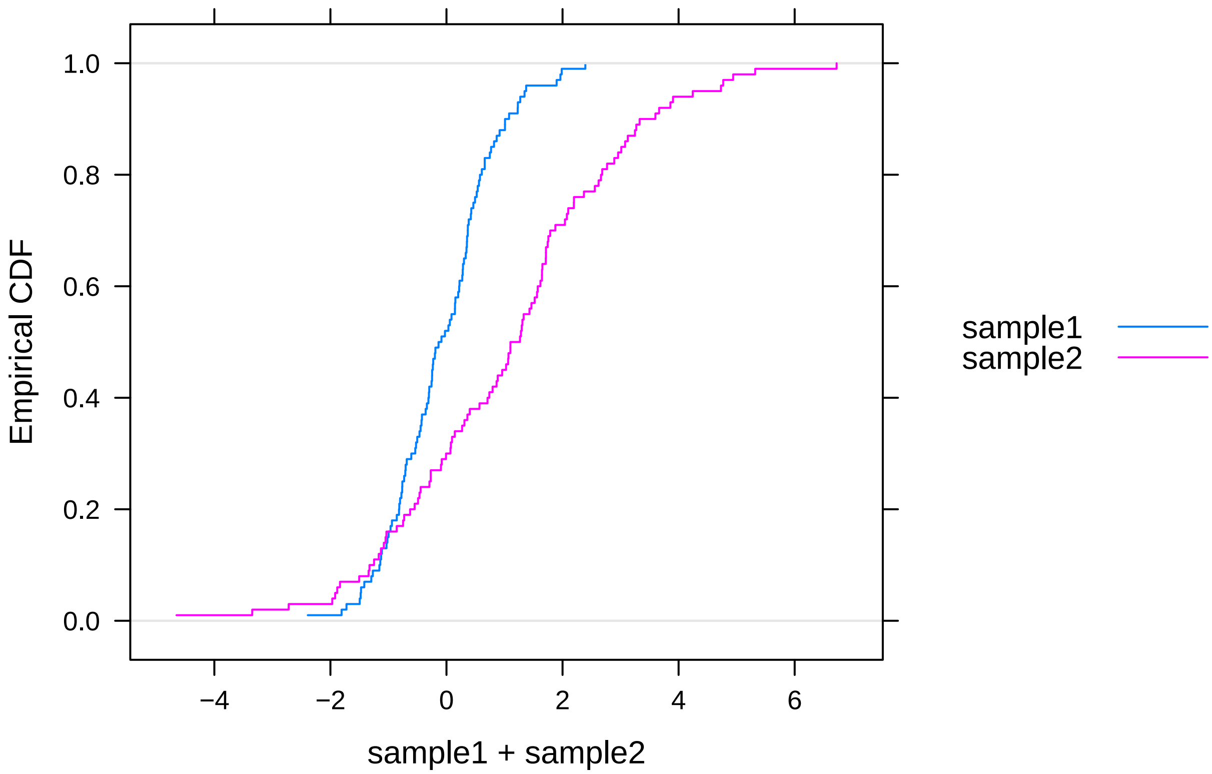
5.1. Kolmlgorov-Smirnov test.

Kolmogorov-Smirnov test is used to test hypotheses of equivalence between two empirical distributions or equivalence between one empirical distribution and one theoretical distribution.

```r
suppressWarnings(library(lattice))
suppressWarnings(library(latticeExtra))
```

```
## Loading required package: RColorBrewer
```

```r
sample1=rnorm(100)
sample2=rnorm(100,1,2)
Cum.Distr.Functions <- data.frame(sample1,sample2)
ecdfplot(~ sample1 + sample2, data=Cum.Distr.Functions, auto.key=list(space='right'))
```



Checking equivalence of empirical distributions for the two samples.

```r
ks.test(sample1,sample2)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sample1 and sample2
## D = 0.45, p-value = 3.21e-09
## alternative hypothesis: two-sided
```

Equivalence of the two distributions?

p value is so small and we reject the null hypothesis of the Kolmogorov-Smirnov Test. Hence, It suggests that the two samples are not sampled from the same distribution.

Checking equivalence of empirical distribution of sample1 and theoretical distribution Norm(0,1).

```r
ks.test(sample1,"pnorm",mean=0,sd=1)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sample1
## D = 0.085042, p-value = 0.4647
## alternative hypothesis: two-sided
```

The null hypothesis that the two samples are drawn from the same distribution cannot be rejected at this level of p-value.

Check equivalence of the empirical distribution of sample2 and theoretical distribution Norm(0,1).

```r
ks.test(sample2,"pnorm",mean=0,sd=1)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sample2
## D = 0.39783, p-value = 3.586e-14
## alternative hypothesis: two-sided
```

p value is so small and we reject the null hypothesis of the Kolmogorov-Smirnov Test. Hence, It suggests that the two samples are not sampled from the same distribution.

5.2. Checking the distribution for the entire period.

Apply Kolmogorov-Smirnov test to Counting.Process$Time and theoretical exponential distribution with parameter equal to average intensity.

Here, the empirical distribution should be estimated for time intervals between malfunctions.

```r
Time.Intervals <- diff(Counting.Process$Time)
Mean.Intensity<- mean(Counting.Process$Count/Counting.Process$Time)
(KS.Test.Event.Intervals <- ks.test(Time.Intervals, "pexp", Mean.Intensity))
```

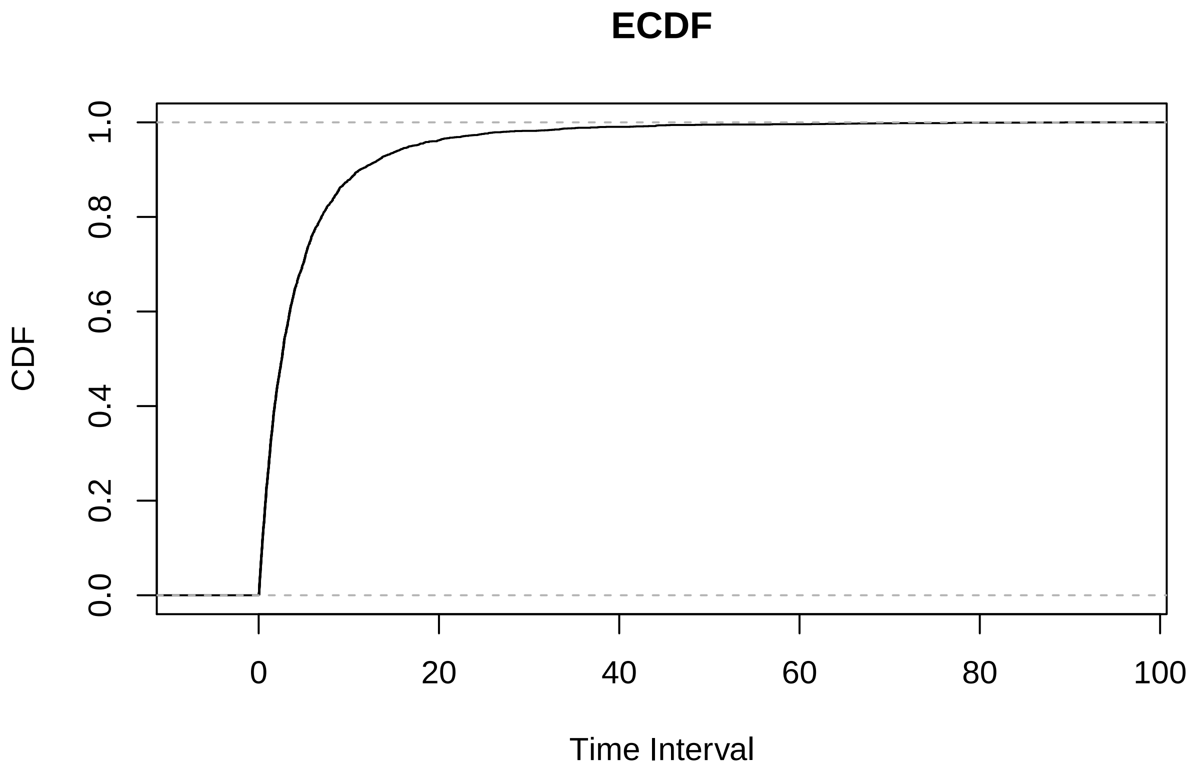
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Time.Intervals
## D = 0.095061, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
c(KS.Test.Event.Intervals$statistic,p.value=KS.Test.Event.Intervals$p.value)
```

```
##         D   p.value
## 0.0950615 0.0000000
```

Ploting empirical cumulative distribution function for time intervals between malfunctions.

```r
plot(ecdf(diff(Counting.Process$Time)), xlab = "Time Interval", ylab = "CDF", main = "ECDF")
```

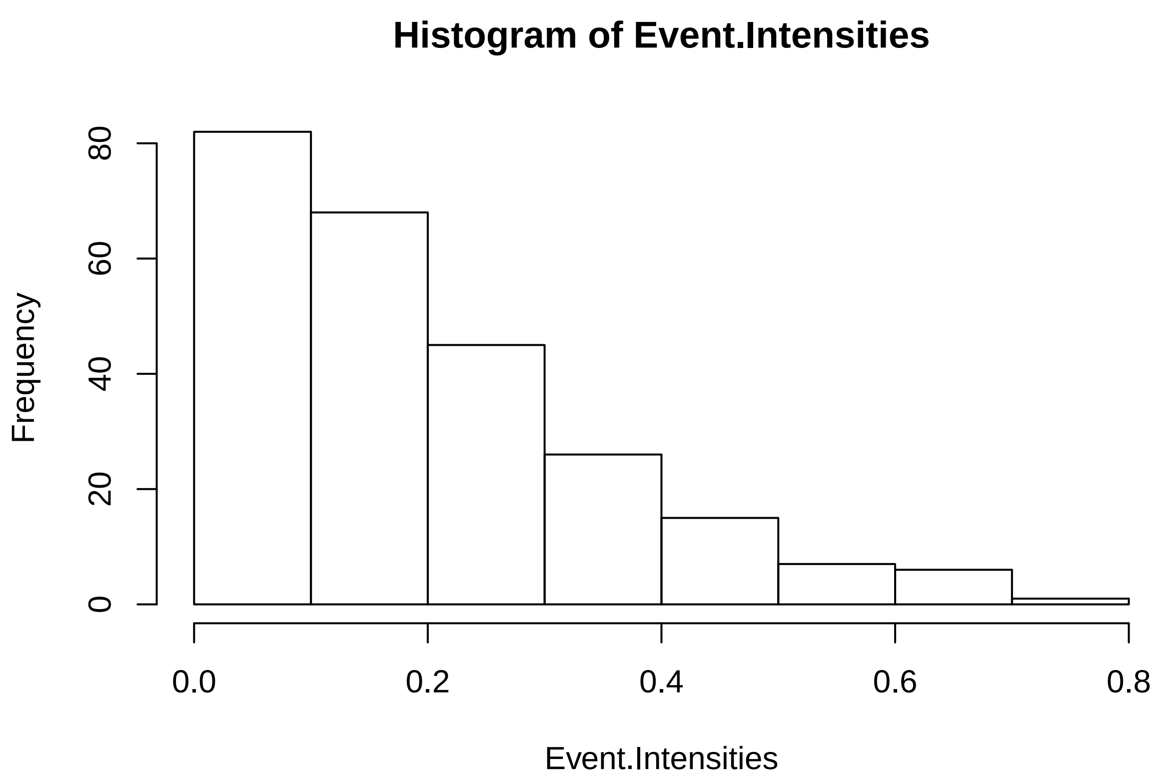


5.3. Checking distribution of one-minute periods

Use at least 5 different candidates for distribution of Poisson intensity of malfunctions.

Find one-minute intensities Event.Intensities. Hint. One-minute intensity by definition is the number of events per unit of time (second).

```
Event.Intensities <- One.Minute.Counts.Temps$Minute.counts/60
hist(Event.Intensities)
```



I guess for the distribution of time intervals between events, this histogram suggest a gamma or exponential distribution.

Suggesting 5 candidates for the distribution.

Fiting each of you 5 candidate distributions to Event.Intensities using fitdistr() from MASS.

Starting with fitting normal and exponential distributions first.

```
Fitting.Normal <- fitdistr(Event.Intensities, densfun = "normal")
Fitting.Normal
```

```
##       mean           sd
##   0.203600000   0.158227459
##  (0.010007183) (0.007076147)
```

```
Fitting.Exponential <- fitdistr(Event.Intensities, densfun = "exponential")
Fitting.Exponential
```

```
##      rate
##   4.9115914
##  (0.3106363)
```

Testing the fitted distributions with Kolmogorov-Smirnov test.

```
KS.Normal <- ks.test(Event.Intensities, "pnorm", Fitting.Normal$estimate[1], Fitting.Normal$estimate[2]
```

```
## Warning in ks.test(Event.Intensities, "pnorm",
## Fitting.Normal$estimate[1], : ties should not be present for the
## Kolmogorov-Smirnov test
```

```
c(KS.Normal$statistic,P.Value=KS.Normal$p.value)
```

```
##            D      P.Value
## 0.1326020316 0.0003039941
```

```r
KS.Normal
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities
## D = 0.1326, p-value = 0.000304
## alternative hypothesis: two-sided
```

The null hypothesis that the two samples are drawn from the same distribution is rejected at this level of p-value. Thus, two samples are drawn from different distribution.

```r
KS.Exp <- ks.test(Event.Intensities, "pexp", Fitting.Exponential$estimate)
```

```
## Warning in ks.test(Event.Intensities, "pexp",
## Fitting.Exponential$estimate): ties should not be present for the
## Kolmogorov-Smirnov test
```

```r
c(KS.Exp$statistic,P.Value=KS.Exp$p.value)
```

```
##           D     P.Value
## 0.115233049 0.002615812
```

```r
KS.Exp
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities
## D = 0.11523, p-value = 0.002616
## alternative hypothesis: two-sided
```

The null hypothesis that the two samples are drawn from the same distribution is rejected at this level of p-value. Thus, two samples are drawn from different distribution.

Trying to fit gamma distribution directly using fitdistr()

```r
Event.Intensities.Mean <- mean(Event.Intensities)
Event.Intensities.Variance <- var(Event.Intensities)*(length(Event.Intensities)-1)/length(Event.Intensi
(Moments.Rate <- Event.Intensities.Mean/Event.Intensities.Variance)
```

```
## [1] 8.132313
```

```r
(Moments.Shape <- Event.Intensities.Mean^2/Event.Intensities.Variance)
```

```
## [1] 1.655739
```

Check gamma distribution with these parameters as a theoretical distribution using Kolmogorov-Smirnov test.

```r
KS.Test.Moments <- ks.test(Event.Intensities, "pgamma", Moments.Shape, Moments.Rate)
```

```
## Warning in ks.test(Event.Intensities, "pgamma", Moments.Shape,
## Moments.Rate): ties should not be present for the Kolmogorov-Smirnov test
```

```r
KS.Test.Moments
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities
```

```
## D = 0.05781, p-value = 0.3736
## alternative hypothesis: two-sided
```

The null hypothesis that the two samples are drawn from the same distribution cannot be rejected at this level of p-value.

Finding at least 2 more candidates and test them by Kolmogorov-Smirnov.

```
Event.Intensities.Nonzero <- ifelse(Event.Intensities==0,0.00001,Event.Intensities)

KS.Candidate.4.fit<-fitdistr(Event.Intensities.Nonzero,"lognormal")
KS.Candidate.4<-ks.test(Event.Intensities.Nonzero,"plnorm",KS.Candidate.4.fit$estimate[1],KS.Candidate.4
```

```
## Warning in ks.test(Event.Intensities.Nonzero, "plnorm", KS.Candidate.
## 4.fit$estimate[1], : ties should not be present for the Kolmogorov-Smirnov
## test
```

```
KS.Candidate.4
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities.Nonzero
## D = 0.22734, p-value = 1.198e-11
## alternative hypothesis: two-sided
```

The null hypothesis that the two samples are drawn from the same distribution is rejected at this level of p-value. Thus, two samples are drawn from different distribution.

```
minutes.intensities.vec <- unlist(One.Minute.Counts.Temps$Minute.counts)
minutes.intensities.vec <- ifelse(is.na(minutes.intensities.vec),0,minutes.intensities.vec)

KS.Candidate.5.fit<-fitdistr(minutes.intensities.vec,"poisson")
KS.Candidate.5<-ks.test(minutes.intensities.vec,"ppois",KS.Candidate.5.fit$estimate)
```

```
## Warning in ks.test(minutes.intensities.vec, "ppois", KS.Candidate.
## 5.fit$estimate): ties should not be present for the Kolmogorov-Smirnov test
```

```
KS.Candidate.5
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  minutes.intensities.vec
## D = 0.30752, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

The null hypothesis that the two samples are drawn from the same distribution is rejected at this level of p-value. Thus, two samples are drawn from different distribution.

Collecting all estimated distributions together and making the best choice.

```
rbind(KS.Moments=c(KS.Test.Moments$statistic,P.Value=KS.Test.Moments$p.value),
      KS.Candidate.4=c(KS.Candidate.4$statistic,P.Value=KS.Candidate.4$p.value),
      KS.Candidate.5=c(KS.Candidate.5$statistic,P.Value=KS.Candidate.5$p.value),
      KS.Exp=c(KS.Exp$statistic,P.Value=KS.Exp$p.value),
      KS.Normal=c(KS.Normal$statistic,KS.Normal$p.value))
```

```
##                         D      P.Value
## KS.Moments     0.05781047 3.736123e-01
## KS.Candidate.4 0.22733673 1.197920e-11
```

```
## KS.Candidate.5 0.30751542 0.000000e+00
## KS.Exp         0.11523305 2.615812e-03
## KS.Normal      0.13260203 3.039941e-04
```

What distribution for the one-minute intensity of malfunctions to choose?

Based on the ks.test performed, I choose the Gamma distribution for one-minute intensity of malfunction as the p-value of ks test was greater than 0.05, which indicates the Event.Intensities is from gamma distribution.

What distribution of one-minute malfunctions counts follow my choice?

It follows Negative binomial distribution, as shown in step 4 because the p-value of ks test was greater than 0.05.

Writing One.Minute.Counts.Temps to file OneMinuteCountsTemps.csv to continue working on Part 2.

```
write.csv(One.Minute.Counts.Temps,file="OneMinuteCountsTemps.csv",row.names=FALSE)
```