Course Project

Brian Ritz

Saturday, February 14, 2015

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 significantly increased cost to the end product of the business. One of suspected reasons for malfunctions is deviation of temperature during the 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.

2. Data – Read in and prepare the data:

```
Course.Project.Data<-read.csv(file="../input/MScA_LinearNonLinear_MalfunctionData.csv")
Course.Project.Data<-as.data.frame(Course.Project.Data)
Course.Project.Data[1:20,]
```

```
##         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
```

```
dim(Course.Project.Data)
```

```
## [1] 3054    2
```

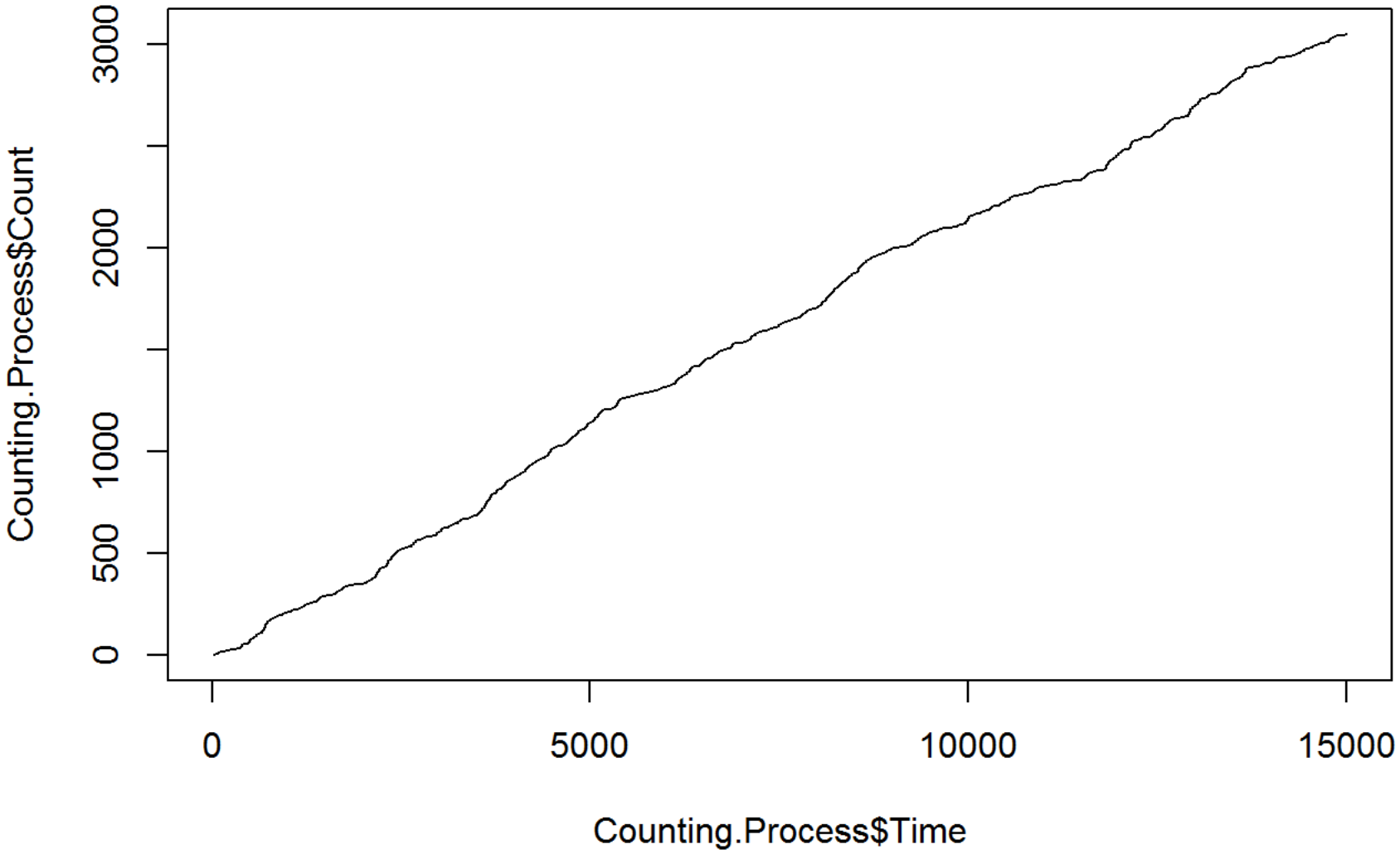
3. Create Counting Process, Explore Cumulative Intensity

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

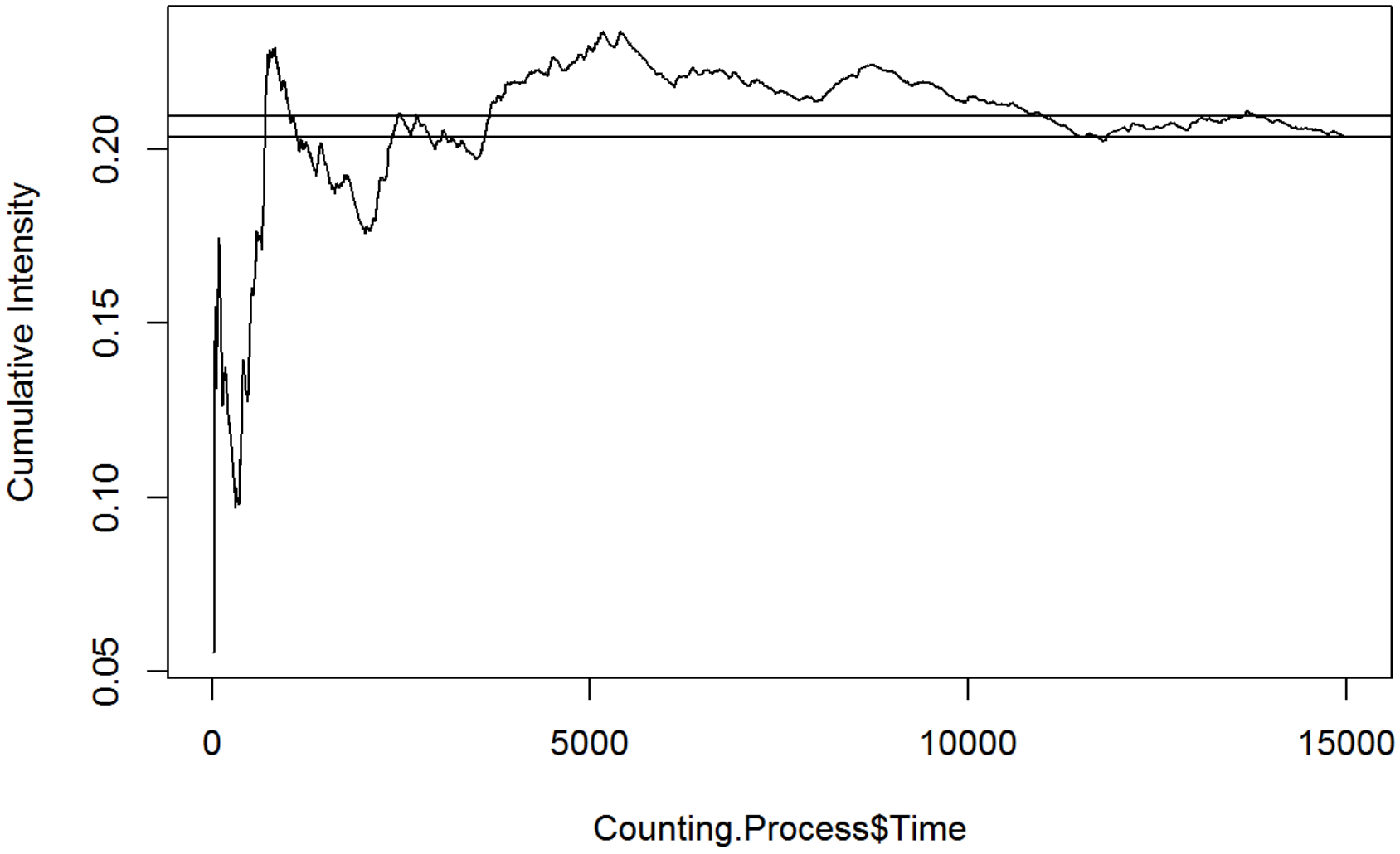
```
plot(Counting.Process$Time,Counting.Process$Count,type="s")
```



3.1 Explore the Cumulative intensity of the process

Cumulative intensity is calculated as the number of events between time zero and t divided by t.

```
plot(Counting.Process$Time,Counting.Process$Count/Counting.Process$Time,type="l",ylab="Cumulative Intensity
")
abline(h=Counting.Process$Count[length(Counting.Process$Count)]/
         Counting.Process$Time[length(Counting.Process$Time)])
abline(h=mean(Counting.Process$Count/Counting.Process$Time))
```



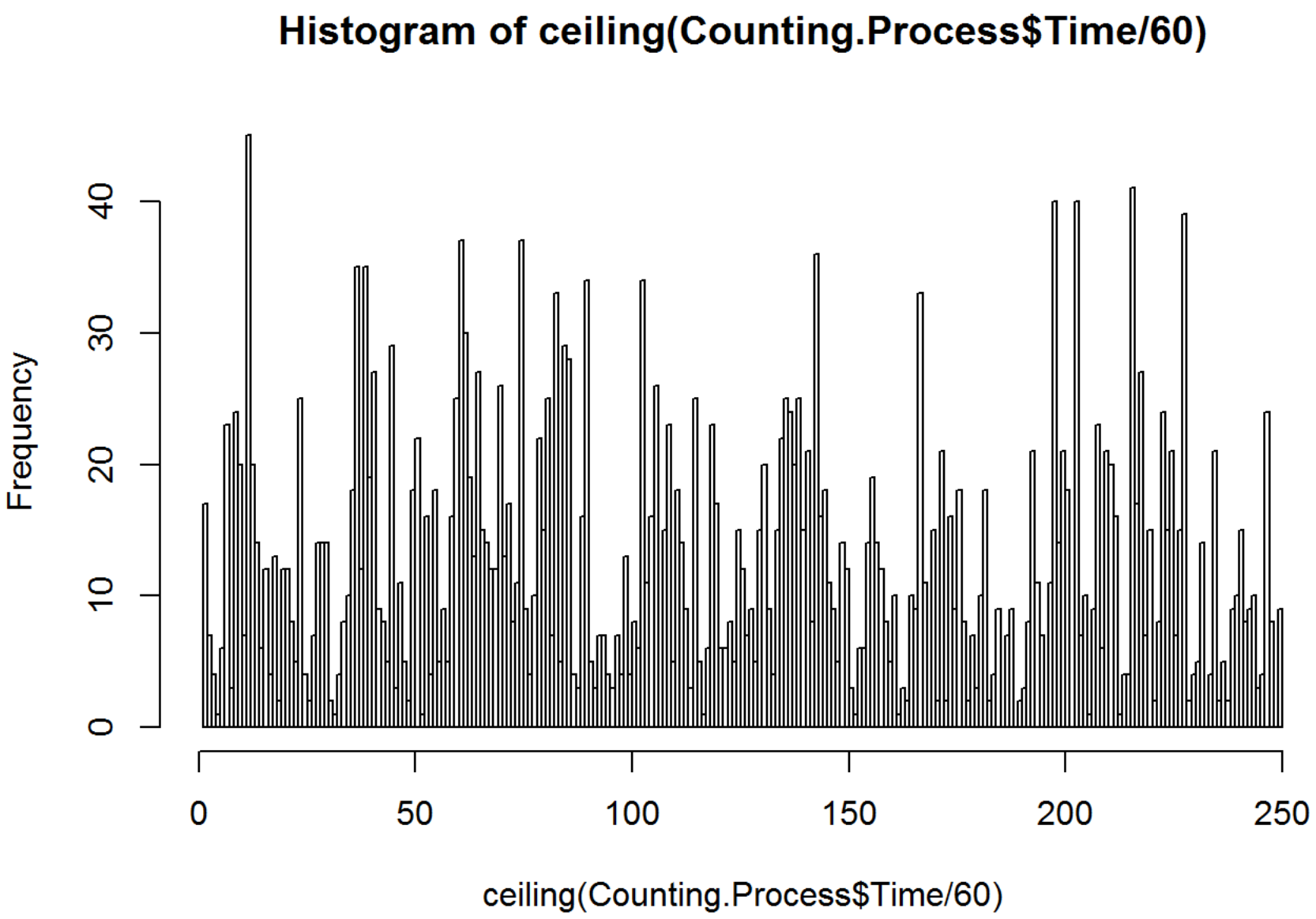

```
c(Last.Intensity=Counting.Process$Count[length(Counting.Process$Count)]/
         Counting.Process$Time[length(Counting.Process$Time)],
  Mean.Intensity=mean(Counting.Process$Count/Counting.Process$Time))
```

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

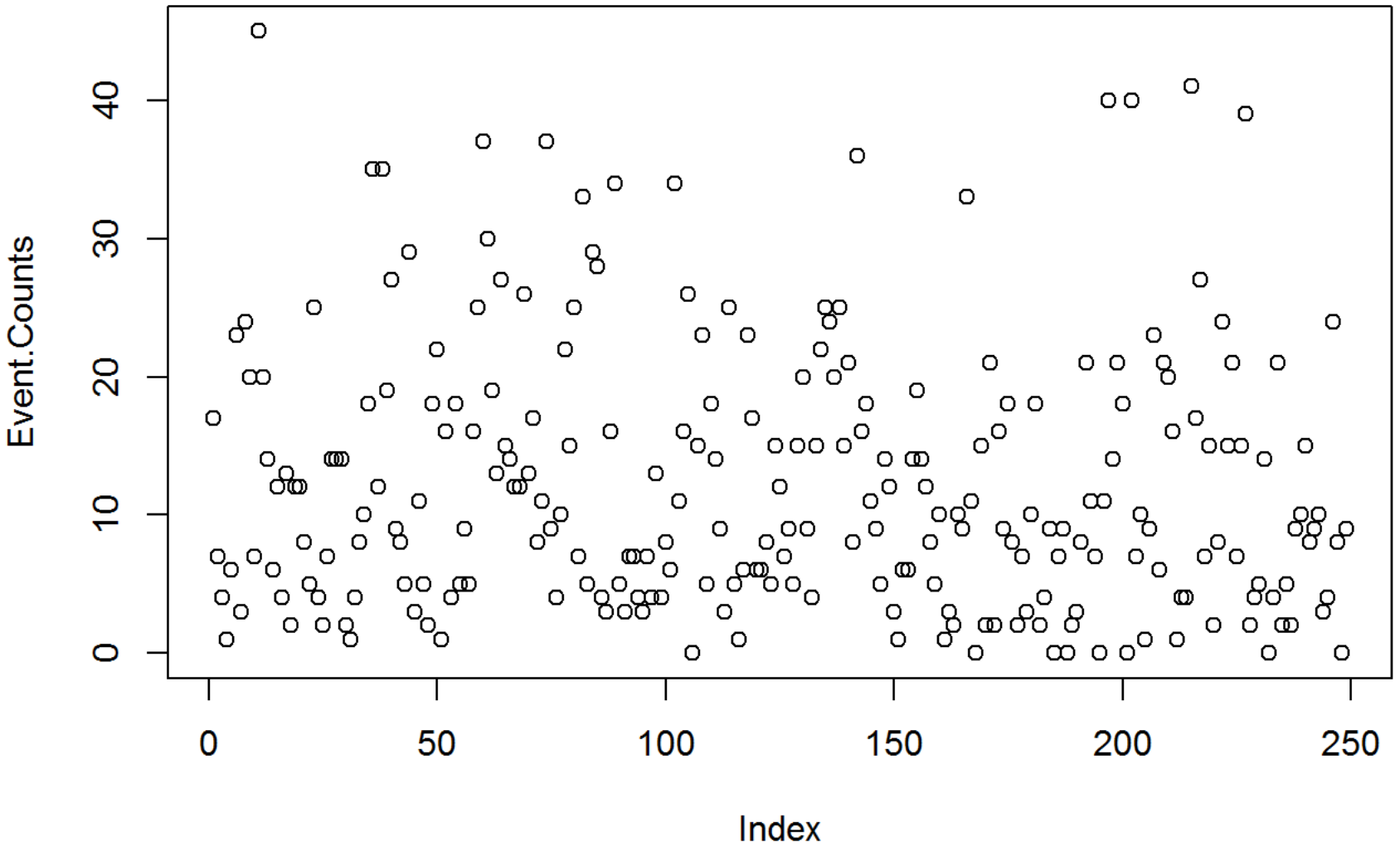
4. Check for overdispersion

Make 60 minute windows – how many counts are there in each one minute window…one dot is one observation for a 60 minute period.

```
Event.Counts<-hist(ceiling(Counting.Process$Time/60), breaks=15000/60)$counts
```




```
plot(Event.Counts)
```



4.1 Methods for testing overdispersion:

4.1.1 Quick and dirty method. Look at the output of glm() and compare the residual deviance with the number of degrees of freedom. If the assumed model is correct deviance is asymptotically distributed as Chi-squared (X2) with degrees of

freedom n-k where n is the number of observations and k is the number of parameters. For Chi-squared 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 overdispersion.

if no overdipersion – then expectation of deviance should be near the residual degress of freedom – if we divide both and there is no overdispersion, then we should get something near 1.

Test the method on simulated Poisson data.

```
# use distribution that we know is poisson and test function will give us what we think
Test.Deviance.Overdispersion.Poisson<-function(Sample.Size,Parameter.Lambda){
  my.Sample<-rpois(Sample.Size,Parameter.Lambda) # get poisson dist
  Model<-glm(my.Sample~1,family=poisson) # fit poisson wth only intercept
  Dev<-Model$deviance  # get deviance
  Deg.Fred<-Model$df.residual # get deg freedom

  # create confidence interval -- and retrun if the deg freedom is within this confidence interval
  (((Dev/Deg.Fred-1)/sqrt(2/Deg.Fred)>-1.96)&((Dev/Deg.Fred-1)/sqrt(2/Deg.Fred)<=1.96))*1
}

# return 1 if confidence interval does cover "0" in the interval, 0 if it does not
Test.Deviance.Overdispersion.Poisson(100,1)
```

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

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

```
## [1] 263
```

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

```
## (Intercept)
##       2.006
```

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
```

Apply the test to one minute counts

```
GLM.model<-glm(Event.Counts~1,family=poisson)
GLM.model
```

```
##
## Call:  glm(formula = Event.Counts ~ 1, family = poisson)
```

```
##
## Coefficients:
## (Intercept)
##       2.507
##
## Degrees of Freedom: 248 Total (i.e. Null);  248 Residual
## Null Deviance:       1798
## Residual Deviance: 1798  AIC: 2785
```

The residual deviance is about 6 times n-k (the degrees of freedom). This is a signal of overdispersion.

4.1.2 Regression test by Cameron-Trivedi

Use the AER package to test the hypothesis that the variance is equal to the mean. This first tests that dispersiontest gives us the expected results. Then we use the dispersiontest to test GLM.model.

```
library(AER)
```

```
## Warning: package 'AER' was built under R version 3.1.3
```

```
## Loading required package: car
```

```
## Warning: package 'car' was built under R version 3.1.3
```

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

```
## Warning: package 'lmtest' was built under R version 3.1.3
```

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

```
## Warning: package 'zoo' was built under R version 3.1.3
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
```

```
## Warning: package 'sandwich' was built under R version 3.1.3
```

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

```
Test.Deviance.Overdispersion.Poisson.AER<-function(Sample.Size,Parameter.Lambda){
  my.Sample<-rpois(Sample.Size,Parameter.Lambda) # get poisson dist
  Model<-glm(my.Sample~1,family=poisson) # fit poisson wth only intercept
  Disp.Test <- dispersiontest(Model)
  return(Disp.Test)
}
```

```
Test.Deviance.Overdispersion.Poisson.AER(100, .2)
```

```
##
##  Overdispersion test
##
## data:  Model
## z = -0.703, p-value = 0.759
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##  0.9266667
```

```
Disp.Test <- dispersiontest(GLM.model)
Disp.Test
```

```
##
##  Overdispersion test
##
## data:  GLM.model
## z = 8.6081, p-value < 2.2e-16
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##   7.374756
```

Given the p-value of < 2.2e-16, we reject the null hypothesis that the mean is equal to the variace, this is consistent with

our earlier findings.

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.

```
library(MASS)
library(pscl)
```

```
## Warning: package 'pscl' was built under R version 3.1.3
```

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

Test the validitiy of the odtest.

Use odTest to test the glm.nb model created by the mass package

```
GLM.model.nb<-glm.nb(Event.Counts~1)
```

```
GLM.model.nb
```

```
##
## Call:  glm.nb(formula = Event.Counts ~ 1, init.theta = 1.741886365,
##     link = log)
##
## Coefficients:
## (Intercept)
##       2.507
##
## Degrees of Freedom: 248 Total (i.e. Null);  248 Residual
## Null Deviance:       277.5
## Residual Deviance: 277.5     AIC: 1742
```

```
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 =  1044.6131 p-value = < 2.2e-16
```

Again, we see that the p-value is very small – indicating overdispersion.

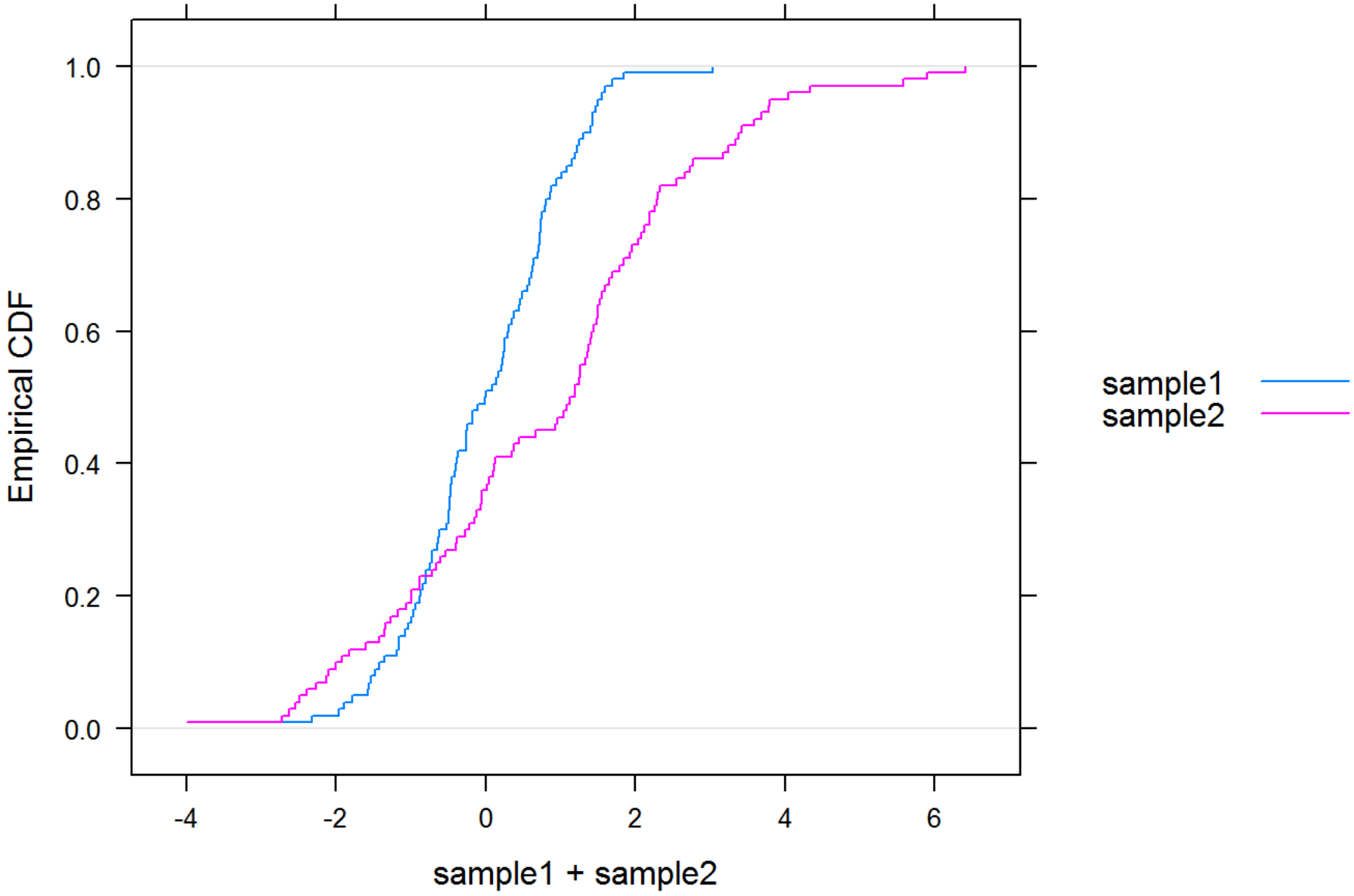
5. Distribution of the Poisson Intensity Kolmlgorov-Smirnov test – tests if samples come from same distribution

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

```
## Warning: package 'latticeExtra' was built under R version 3.1.3
```

```
## 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'))
```




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

```
##
```

```
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sample1 and sample2
## D = 0.37, p-value = 2.267e-06
## alternative hypothesis: two-sided
```

P-value is again very small – this is another indication that we may have overdispersion.

Check eqiovalence of empirical distribution of sample1 and theoretical distribution Norm(0,1). The null hypothesis is that the distributions are the same.

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

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sample1
## D = 0.0649, p-value = 0.7938
## alternative hypothesis: two-sided
```

There is weak evidence that sample 1 is different from a normal distribution with mean 0 and standard deviation 1.

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

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

```
##
##  One-sample Kolmogorov-Smirnov test
```

```
##
## data:  sample2
## D = 0.3833, p-value = 3.473e-13
## alternative hypothesis: two-sided
```

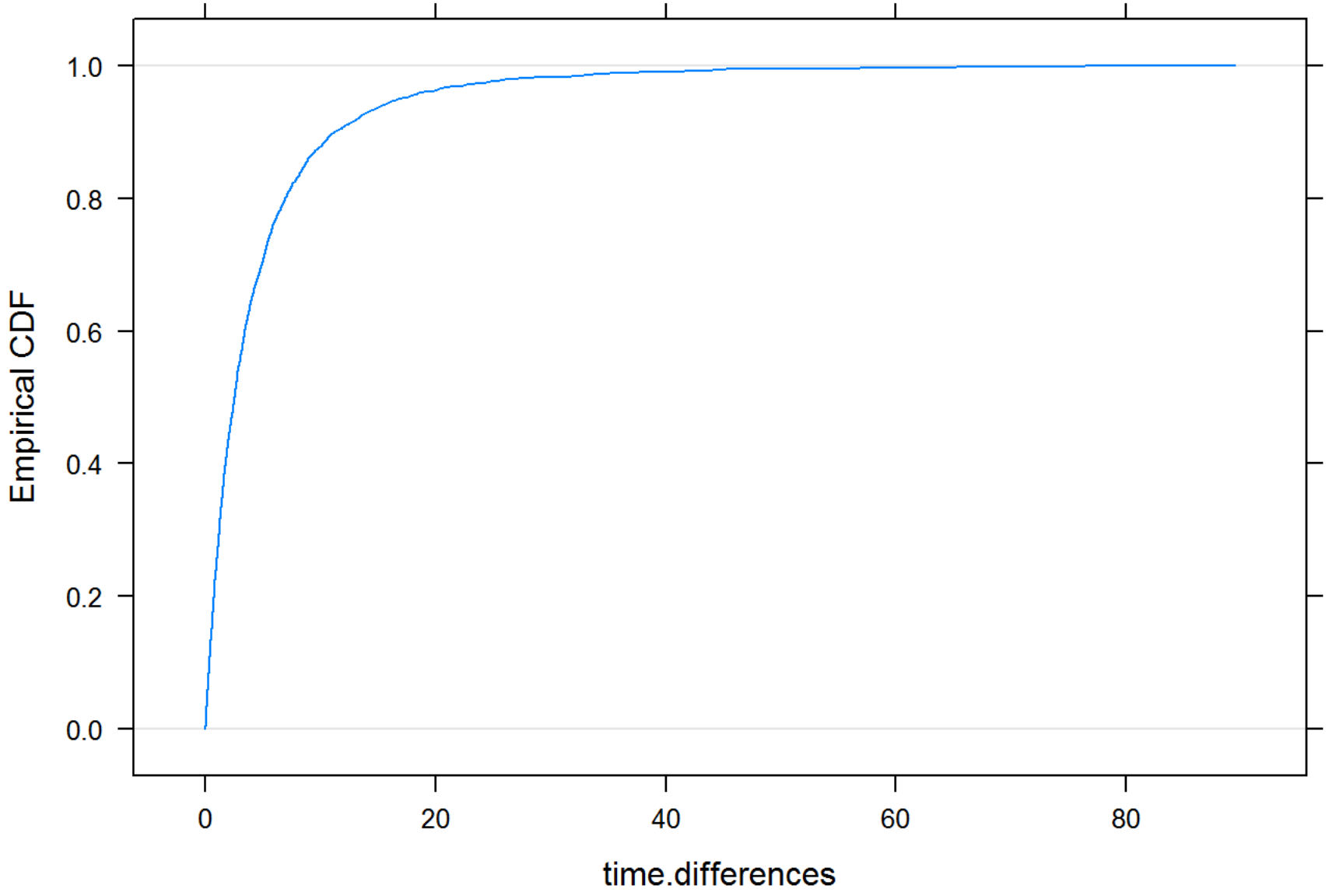
It appears that sample2 is significantly different from the normal distribution.

5.2. Check the distribution for the entire period. if it is a poisson distribution, then the time intervals between malfunctions should be exponential

```
time.differences <- diff(Counting.Process$Time)
ks.test(time.differences, "pexp", rate=1/mean(time.differences))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  time.differences
## D = 0.1029, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
# plot a CDF of the time differences
ecdfplot(~ time.differences)
```

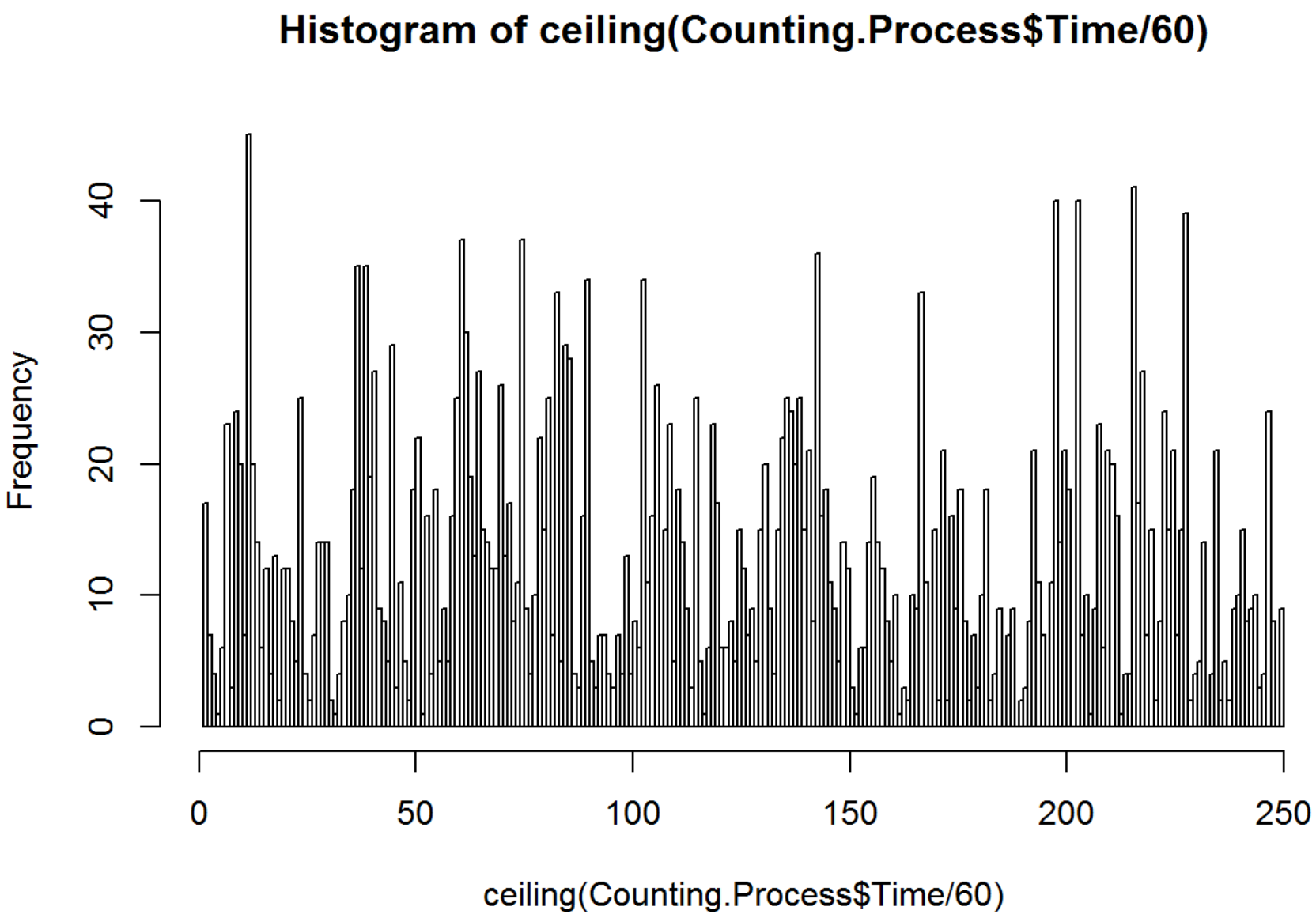


5.3. Check distribution of one-minute periods

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

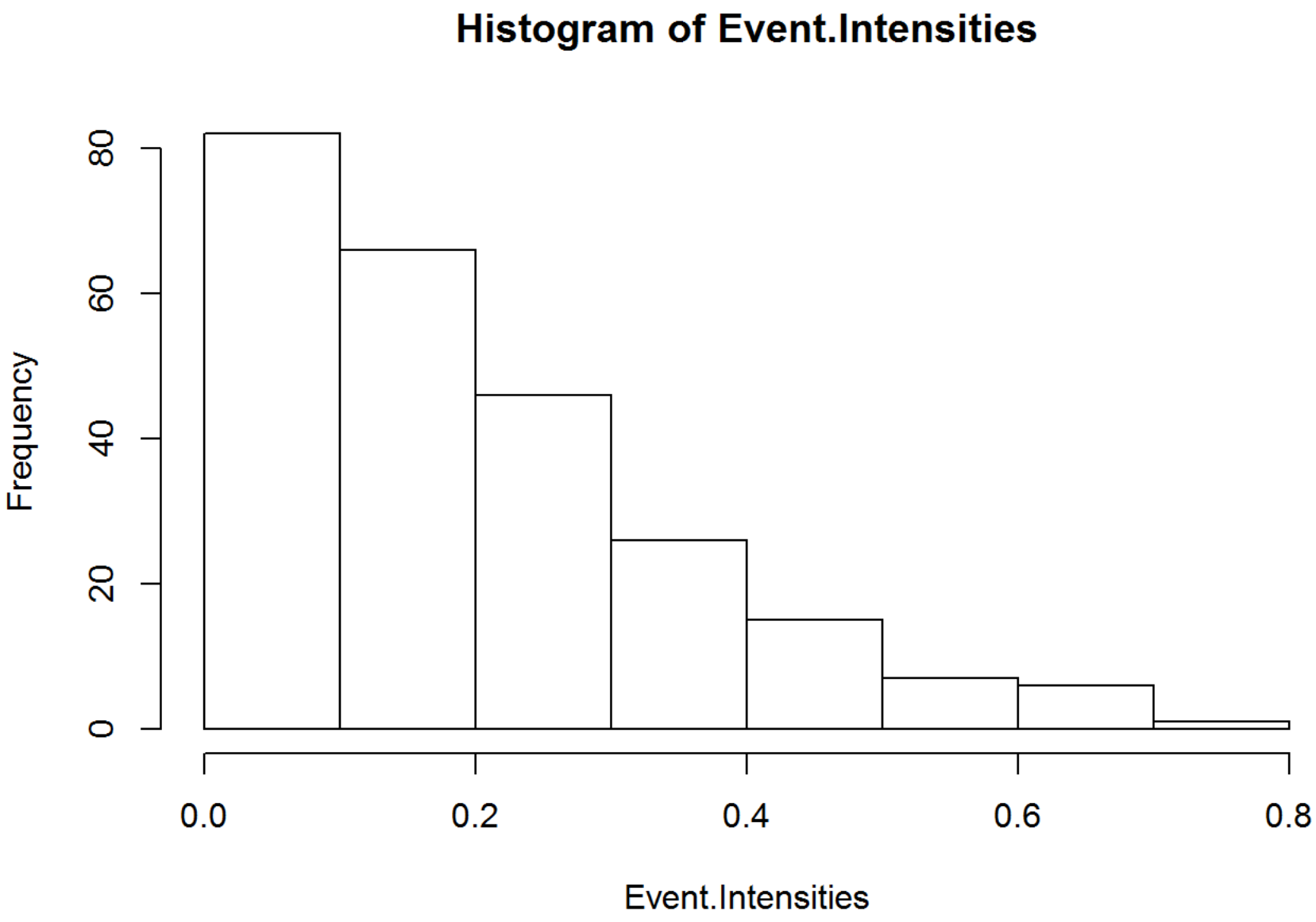
Find one-minute intensities.

```
Event.Intensities <-hist(ceiling(Counting.Process$Time/60), breaks=15000/60)$counts
```



```
Event.Intensities <-  Event.Intensities/60
```

```
# event intensities are the counts of observations in each 1 minute period
hist(Event.Intensities)
```



1st Distribution: Normal

```
normal.fitting <- fitdistr(Event.Intensities, "normal")
print(normal.fitting)
```

```
##       mean           sd
##   0.204417671   0.158510280
##  (0.010045181) (0.007103015)
```

2nd Distribution Exponential

```
exponential.fitting <- fitdistr(Event.Intensities, "exponential")
print(exponential.fitting)
```

```
##      rate
##   4.8919450
##  (0.3100144)
```

3rd Distribution Geometric

```
geometric.fitting <- fitdistr(Event.Intensities, "geometric")
print(geometric.fitting)
```

```
##       prob
##   0.83027676
##  (0.02167673)
```

4th Distribution Beta

```
beta.fitting <- fitdistr(Event.Intensities[Event.Intensities>0], "beta", list(shape1=.5, shape2=.5))
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
## Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
print(beta.fitting)
```

```
##     shape1      shape2
##   1.3699065   5.0524676
##  (0.1132025) (0.4753296)
```

5th Distribution Gamma

```
gamma.fitting <- fitdistr(Event.Intensities[Event.Intensities>0], "gamma",list(shape = 1, rate = 0.1), lowe
r = 0.001, upper=.999)
print(gamma.fitting)
```

```
##      shape         rate
##   0.51879748   0.99900000
##  (0.03914885) (0.11689734)
```

Test the fitted distributions with the Kolmogoraov-Smirnov Test Normal:

```
KS.Normal <- ks.test(Event.Intensities, "pnorm", mean=normal.fitting$estimate["mean"], sd=normal.fitting$es
timate["sd"])
```

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

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

```
##            D      P.Value
## 0.1323081679 0.0003273215
```

Exponential:

```
KS.exp <- ks.test(Event.Intensities, "pexp", rate=exponential.fitting$estimate["rate"])
```

```
## Warning in ks.test(Event.Intensities, "pexp", rate =
## exponential.fitting$estimate["rate"]): ties should not be present for the
## Kolmogorov-Smirnov test
```

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

```
##           D     P.Value
## 0.113629766 0.003224778
```

Geometric:

```
KS.geom <- ks.test(Event.Intensities, "pgeom", prob=geometric.fitting$estimate["prob"])
```

```
## Warning in ks.test(Event.Intensities, "pgeom", prob =
## geometric.fitting$estimate["prob"]): ties should not be present for the
## Kolmogorov-Smirnov test
```

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

```
##         D   P.Value
## 0.8302768 0.0000000
```

Beta:

```
KS.beta <- ks.test(Event.Intensities, "pbeta", shape1=beta.fitting$estimate["shape1"],  shape2=beta.fitting
$estimate["shape2"])
```

```
## Warning in ks.test(Event.Intensities, "pbeta", shape1 =
## beta.fitting$estimate["shape1"], : ties should not be present for the
```

```
## Kolmogorov-Smirnov test
```

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

```
##          D    P.Value
## 0.08538348 0.05300016
```

Gamma:

```
KS.gamma <- ks.test(Event.Intensities, "pgamma", shape=gamma.fitting$estimate["shape"], rate=gamma.fitting$
estimate["rate"])
```

```
## Warning in ks.test(Event.Intensities, "pgamma", shape =
## gamma.fitting$estimate["shape"], : ties should not be present for the
## Kolmogorov-Smirnov test
```

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

```
##         D   P.Value
## 0.2834438 0.0000000
```

Fit the gamma distribution directly.

```
gamma.mom <- function(x) {
  x.bar <- mean(x)
  n <- length(x)
```

```
  v <- var(x) * (n-1) / n

  l.est <- x.bar/v
  a.est<-(x.bar)^2/v
  # gamma distribution has mean shape*scale and variance shape*scale**2 + shape**2*scale**2

  return(list(rate=l.est, shape=a.est))

}
print(gamma.mom(Event.Intensities))
```

```
## $rate
## [1] 8.135862
##
## $shape
## [1] 1.663114
```

```
gamma.mom.estimates<- gamma.mom(Event.Intensities)
```

Ks- Test the method of moments gamma parameters against the known distribution

```
KS.gamma.mom <- ks.test(Event.Intensities, "pgamma", shape=gamma.mom.estimates$shape, rate=gamma.mom.estimates$rate)
```

```
## Warning in ks.test(Event.Intensities, "pgamma", shape =
## gamma.mom.estimates$shape, : ties should not be present for the
## Kolmogorov-Smirnov test
```

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

```
##          D    P.Value
## 0.06103669 0.00000000
```

This is an extremely low D statistic and p-value – we can conclude that the intensities follow a gamma distribution. This implies that the counts follow a negative binomial distribution. We will try to estimate that negative binomial distribution in part II