LNL_Course_Project

Patrick Kelly

Tuesday, March 17, 2015

Course Assignment. Part 1

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

The file `MScA_LinearNonLinear_CourseProject.csv` contains time stamps of events expressed in seconds.

Read and prepare the data.

```
Course.Project.Data<-read.csv(file="C:/Users/Patrick/Documents/R/UChicago/Linear_NonLinear/MScA_LinearN

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
```

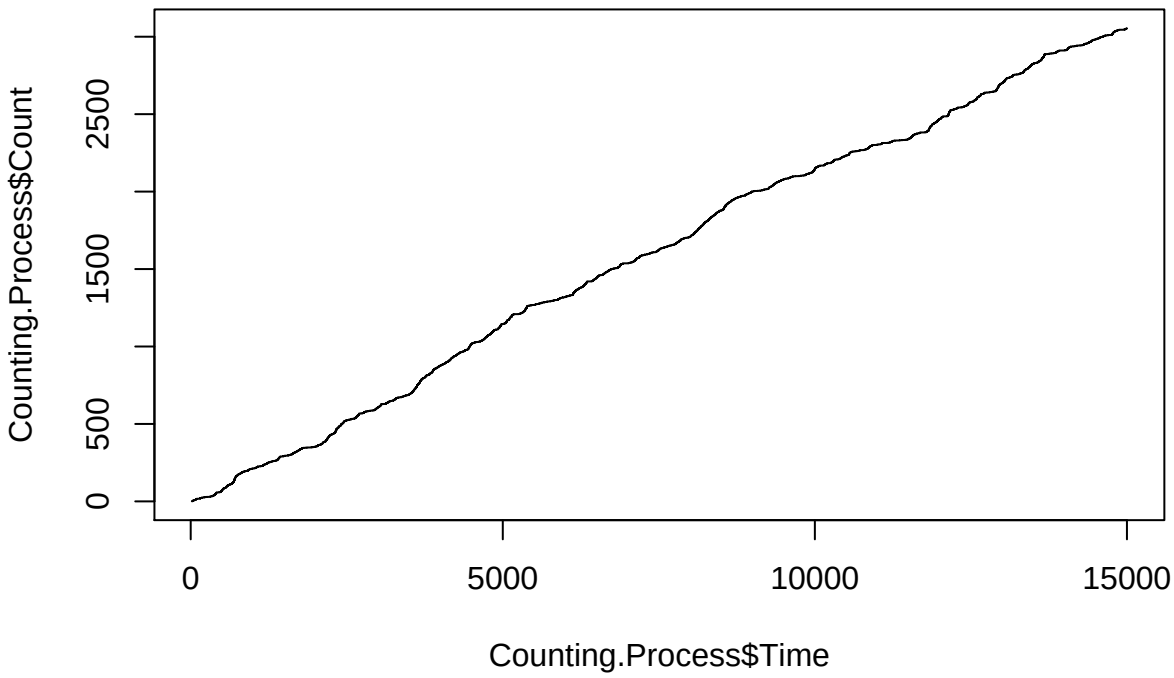
3. Create Counting Process, Explore Cumulative Intensity

Counting Process is a step function that jumps by 1 at every moment of new event.

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

Counting.Process[1:20,]
```

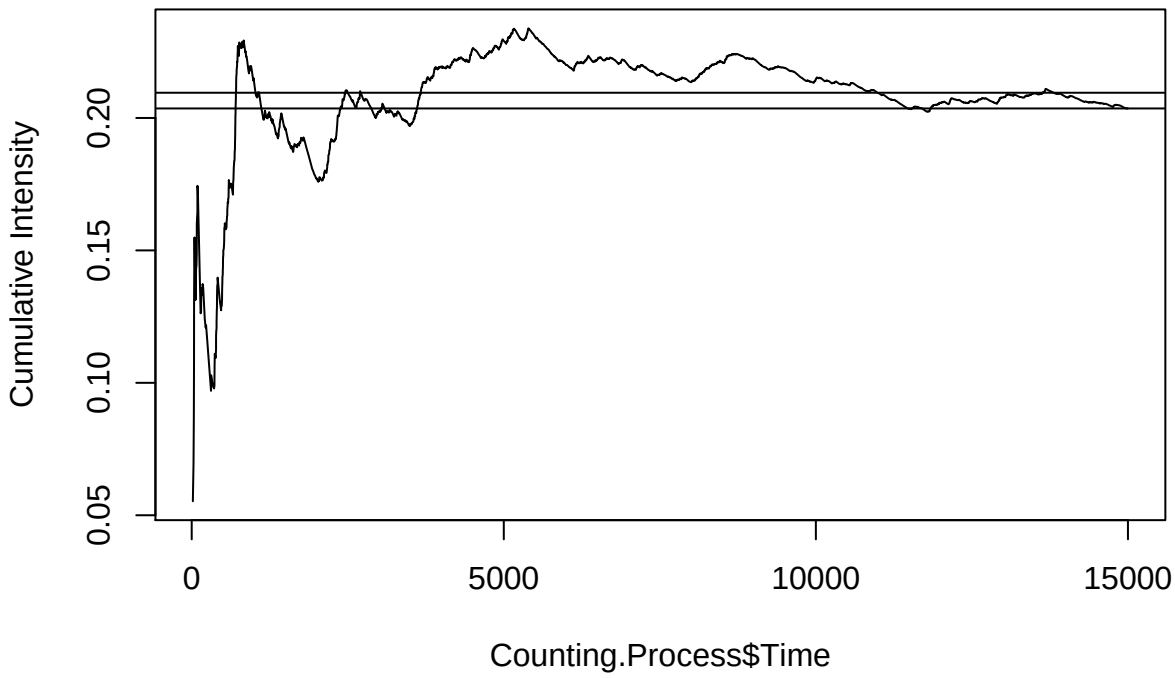
```
##         Time Count
## 1   18.08567     1
## 2   28.74417     2
## 3   34.23941     3
## 4   36.87944     4
## 5   37.84399     5
## 6   41.37885     6
## 7   45.19283     7
## 8   60.94242     8
## 9   66.33539     9
## 10  69.95667    10
## 11  76.17420    11
## 12  80.48524    12
## 13  81.29133    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")
```



3.1 Explore cumulative intensity of the process

```
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))
```



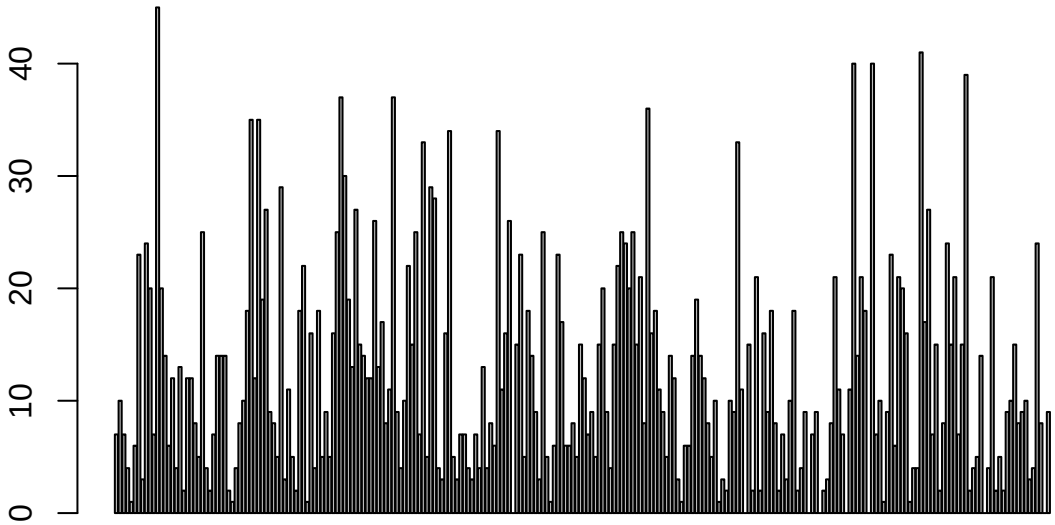

```
#check intensity
c(Last.Intensity=Counting.Process$Count[length(Counting.Process$Count)]/
         Counting.Process$Time[length(Counting.Process$Time)],
  Mean.Intensity=mean(Counting.Process$Count/Counting.Process$Time))
```

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

4. Check for overdispersion

In order to do that create one-minute counts.

```
#Make 60 second windows of time -- determine the observed counts in each minute duration

# Event.Counts <- hist(ceiling(Counting.Process$Time/60), breaks=250)$counts
ecounts <- table(ceiling(Counting.Process$Time/60))
ecounts_2 <- data.frame(cbind(as.numeric(names(ecounts)),as.numeric(ecounts)))
colnames(ecounts_2) <- c("MINUTES","COUNTS")
ecounts_3 <- data.frame(MINUTES=c(1:250))
Event.Counts <- merge(x=ecounts_2, y=ecounts_3, by="MINUTES", all = TRUE)
Event.Counts$COUNTS[is.na(Event.Counts$COUNTS)] <- 0

barplot(Event.Counts$COUNTS)
```



4.1 Methods for Testing Overdispersion

4.1.1 A quick and rough 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 X2 distribution the mean is the number of degrees of freedom n???k. If the residual deviance returned by `glm()` is greated than n???k then it might be a sign of overdispersion.

Test the method on simulated Poisson data.

Note: Deviance has chisquared distribution (mean is the df)

```
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] 0
```

```
sum(replicate(1000,Test.Deviance.Overdispersion.Poisson(100,1)))
```

```
## [1] 887
```

```
#sum(replicate(1000,Test.Deviance.Overdispersion.Poisson(150,1)))
#sum(replicate(1000,Test.Deviance.Overdispersion.Poisson(200,1)))

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

```
## (Intercept)
##       1.964
```

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(1000,Test.Deviance.Overdispersion.NBinom(100,.2)))
```

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

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

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

```
##
## Call:  glm(formula = Event.Counts$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
```

Do you see signs of overdispersion?

Yes, we do observe signs overdisperesion because the null deviance (1798) is considerably higher than the degrees of freedom (248). When this ratio is > 1, as it is for this data, it indicates overdispersion.

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 Overdispersion in the Poisson Model. Journal of Econometrics, 46, 347-364.

In a Poisson model, the mean is E(Y)=lambda and the variance is V(Y)=lambda as well. They are equal. The test has a null hypothesis c=0 where Var(Y)=lambda+c * f(lambda), c<0 means underdispersion and c>0 means overdispersion. 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.

```r
library(AER)
```

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

```r
mysample<-rpois(100,1)
model<-glm(mysample~1,family=poisson)

mysample2<-rnbinom(100,2,0.2)
model2<-glm(mysample2~1,family=poisson)

#Disp.Test
dispersiontest(model)
```

```
##
##  Overdispersion test
##
## data:  model
## z = -0.671, p-value = 0.7489
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##  0.9268504
```

```r
dispersiontest(model2)
```

```
##
##  Overdispersion test
##
## data:  model2
## z = 4.8024, p-value = 7.841e-07
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##   5.163325
```

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)
library(pscl)
```

```
## 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
```

```
#fit the negative binomial and test which negative binomial it is likely to be --- if it is general neg

#small p-value = reject that it is poisson

# Test.Deviance.Overdispersion.Poisson2<-function(samplesize,param_lambda){
#   p_sample<-rpois(samplesize,param_lambda)
#   p_sample<-data.frame(p_sample_var = p_sample)
#   nbmodel<-glm.nb(p_sample_var~1,data=p_sample)
#   odtest_out <- odTest(nbmodel)
#   return(odtest_out)
# }
#
# Test.Deviance.Overdispersion.Poisson2(100,5)

  p_sample<-rpois(100,1)
  nbmodel<-glm.nb(p_sample~1)
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
  odtest_out <- odTest(nbmodel)
```

```
## 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 =  -5e-04 p-value = 0.5
```

```
# Test.Deviance.Overdispersion.NBinom2<-function(samplesize,param_prob){
#   rnsample<-rnbinom(samplesize,2,param_prob)
#   nbmodel<-glm.nb(rnsample~1)
#   odTest(nbmodel)
# }
#
# Test.Deviance.Overdispersion.NBinom2(100,.2)

  rnsample<-rnbinom(100,2,0.2)
  nbmodel<-glm.nb(rnsample~1)
  odTest(nbmodel)
```

```
## 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 =  173.8443 p-value = < 2.2e-16
```

5. Find the distribution of Poisson intensity

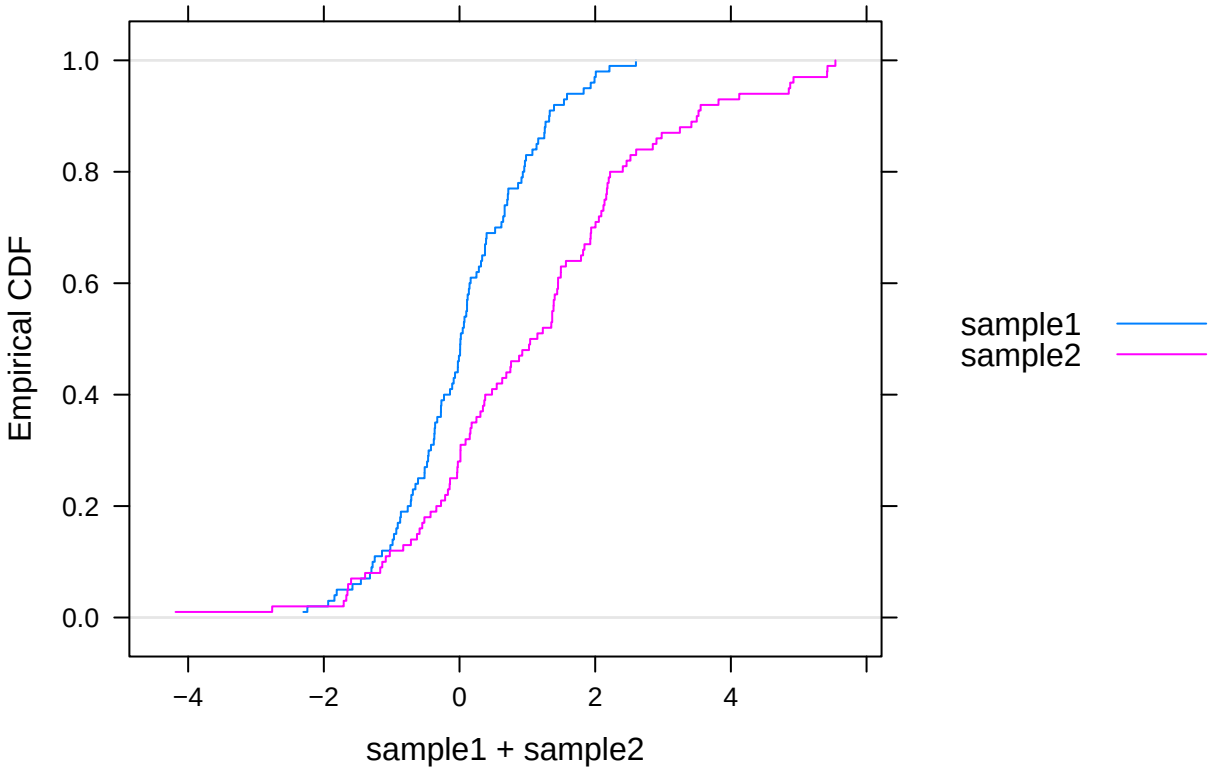
5.1. Kolmlgorov-Smirnov test

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

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

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

```
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'))
```



Check equivalence of empirical distributions for the two samples.

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

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sample1 and sample2
## D = 0.39, p-value = 4.959e-07
## alternative hypothesis: two-sided
```

Check eqiovalence of empirical distribution of `sample1` and theoretical distribution `Norm(0,1)`.

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

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sample1
## D = 0.053, p-value = 0.9413
## alternative hypothesis: two-sided
```

Check eqiovalence of 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 = 0.392, p-value = 8.993e-14
## alternative hypothesis: two-sided
```

5.2. Check the distribution for the entire period

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

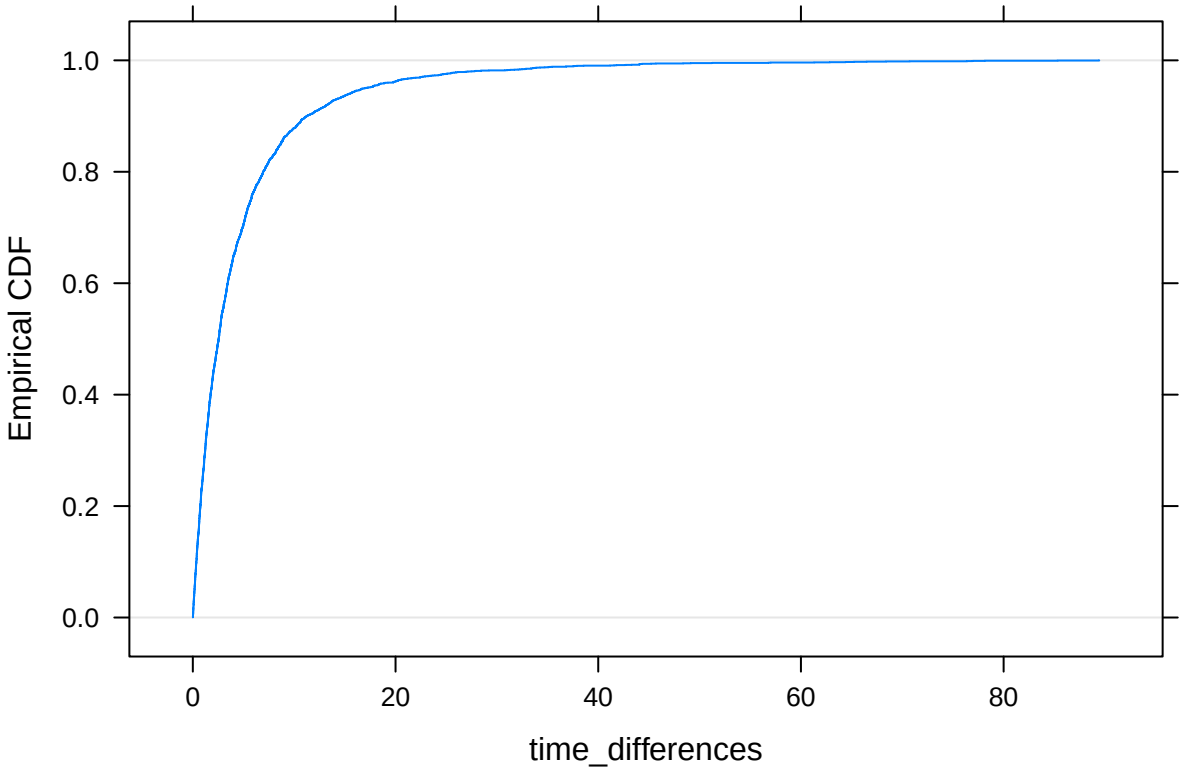
Hint: the empirical distribution should be estimated for time intervals between malfunctions.

Plot empirical cumulative distribution function for time intervals between malfunctions.

```r
#check ks test
fit1 <- fitdistr(Counting.Process$Time, "exponential")
ks.test(Counting.Process$Time, "pexp", fit1$estimate)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Counting.Process$Time
## D = 0.1597, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
#ks.test(Counting.Process$Time, "pexp", (1/mean(Counting.Process$Time)))

#fit2 <- fitdistr(time_differences, "exponential")
#ks.test(time_differences, "pexp", fit2$estimate)
# ks.test(time_differences, "pexp", rate=(1/mean(time_differences)))

#create vector of differences
time_differences <- diff(Counting.Process$Time)

#check the culm dist plot
ecdfplot(~time_differences)
```

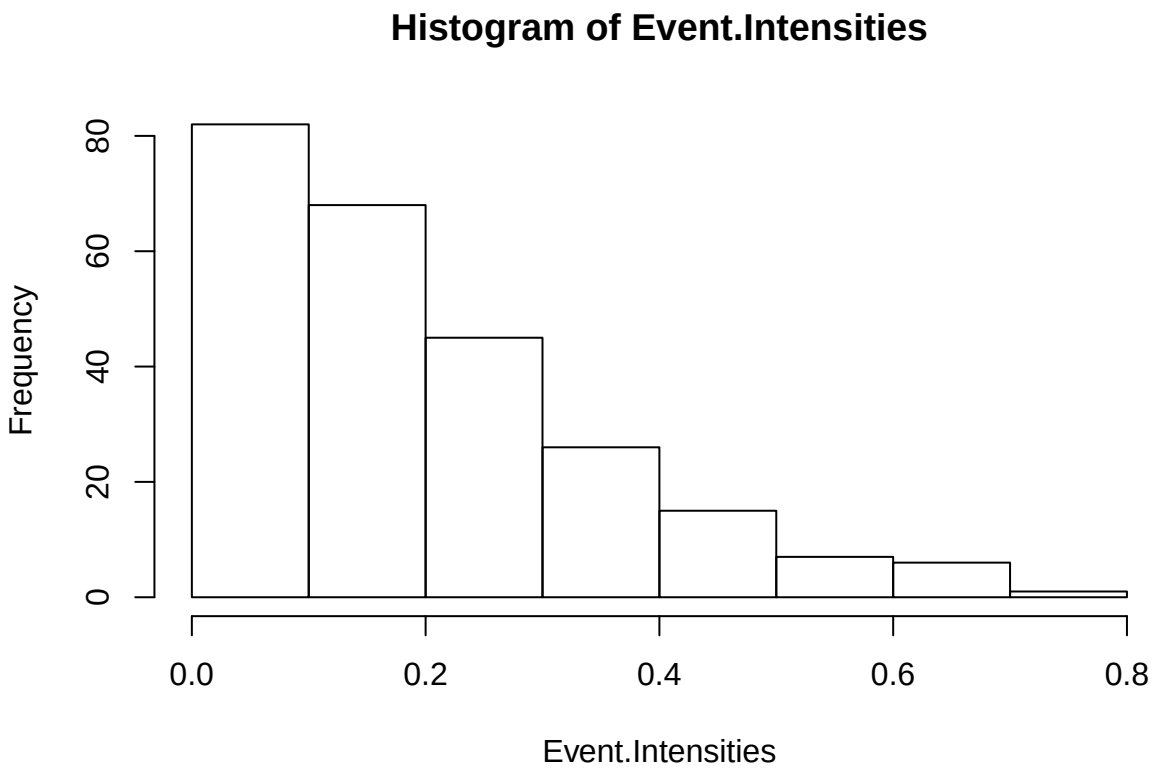


5.3. Check distribution of one-minute periods

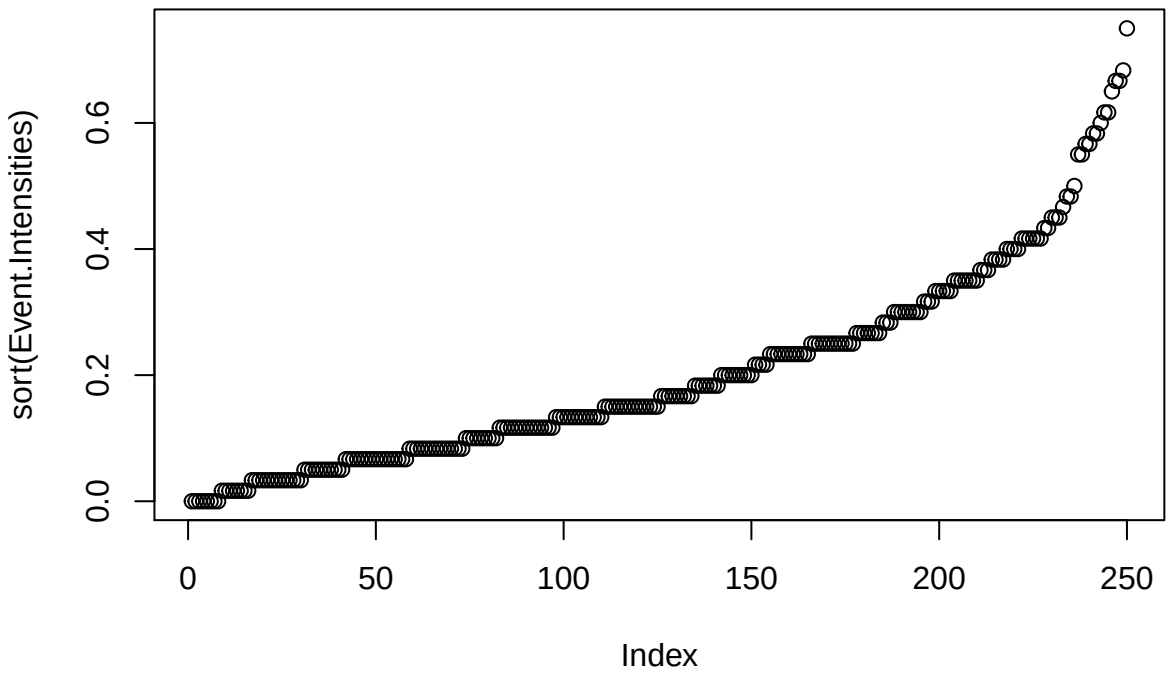
Use at least 5 different candidates for distribution of Poisson intensity of malfunctions.

Find one-minute intensities.

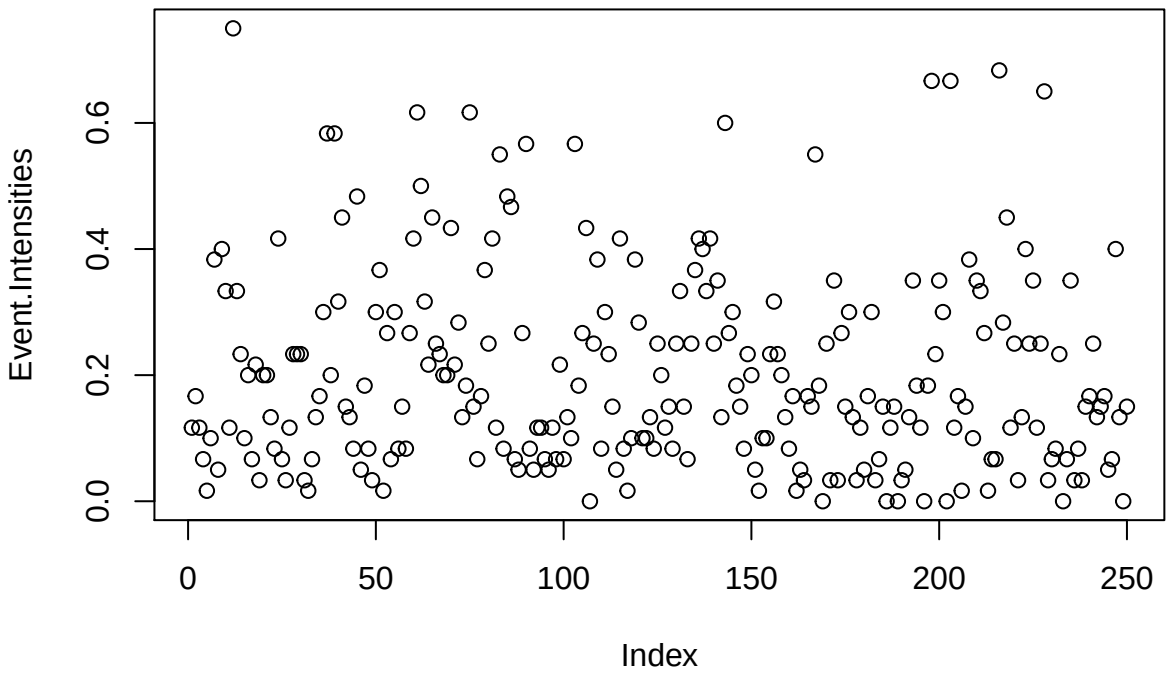
```
#create a cumulative counts
Event.Counts <- cbind.data.frame(Event.Counts, CUM.COUNTS = cumsum(Event.Counts$COUNTS))

#calculate intensities
Event.Intensities <- Event.Counts$COUNTS / 60
hist(Event.Intensities)#, breaks=20)
```



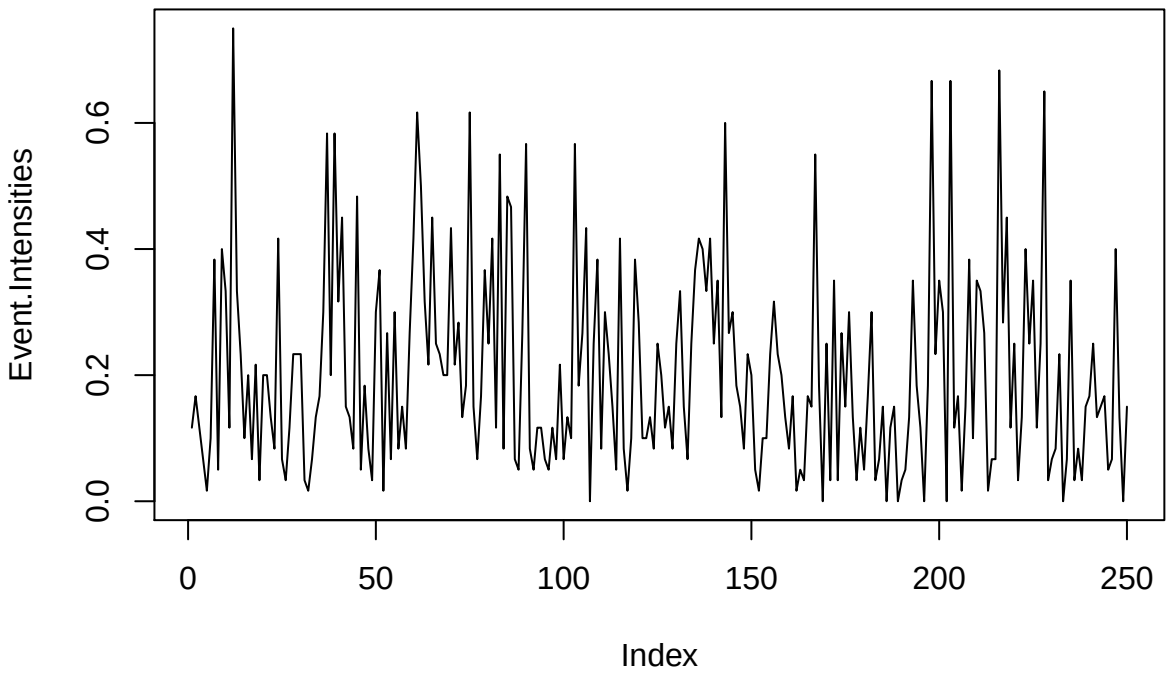

```
plot(sort(Event.Intensities))
```




```
plot(Event.Intensities)
```




```
plot(Event.Intensities, type = "l")
```



Fit 5 different distributions to `Event.Intensities` using `fitdistr()` from MASS.

Recommendation: start with fitting normal and exponential distributions first.

```
#fit normal distribution
Fitting.Normal <- fitdistr(Event.Intensities, "normal")
Fitting.Normal
```

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

```
#fit exponential distribution
Fitting.Exponential <- fitdistr(Event.Intensities, "exponential")
Fitting.Exponential
```

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

Test the fitted distributions with Kolmogorov-Smirnov test.

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

```
## Warning in ks.test(Event.Intensities, "pnorm", mean =
## 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
```

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

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

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

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

Try to fit gamma distribution directly

Estimate parameters of gamma distribution using method of moments.

```
xbar <- mean(Event.Intensities)
n <- length(Event.Intensities)
v <- var(Event.Intensities)* (n-1)/n

Moments.Rate <- xbar/v
Moments.Shape <- (xbar)^2/v
```

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

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

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

```
KS.Test.Moments
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities
## D = 0.0578, p-value = 0.3736
## alternative hypothesis: two-sided
```

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

```r
#fit poisson dist
Fitting.Poisson <- suppressWarnings(fitdistr(Event.Intensities,"Poisson"))
Fitting.Poisson
```

```
##      lambda
##   0.20360000
##  (0.02853769)
```

```r
KS.Candidate.4 <- ks.test(Event.Intensities,"ppois",lambda=Fitting.Poisson$estimate[1])
```

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

```r
KS.Candidate.4
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities
## D = 0.8158, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
#fit geometric dist
Fitting.Geom <- fitdistr(Event.Intensities,"geometric")
Fitting.Geom
```

```
##       prob
##   0.83084081
##  (0.02161203)
```

```r
KS.Candidate.5 <- ks.test(Event.Intensities,"pgeom", prob = Fitting.Geom$estimate[1])
```

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

```r
KS.Candidate.5
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities
## D = 0.8308, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Collect all estimated distributions together and make your 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 0.3736123382
## KS.Candidate.4 0.81578862 0.0000000000
## KS.Candidate.5 0.83084081 0.0000000000
## KS.Exp         0.11523305 0.0026158124
## KS.Normal      0.13260203 0.0003039941
```

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

The Gamma distribution. This decision is based on the fits and KS tests above - observe that the gamma fit produces the only passing pvalue from the above test, indicating its gamma distribution fit

What is the resulting distribution of the counts of one-minute malfunctions for your choice?

The resulting distribution of the counts of one-minute malfunctions is a negative-binomial distribution.

Course Assignment. Part 2

Explore possible types of dependence between one-minute counts and temperature.

Read the data and create a data frame with one-minute breaks counts and temperature measurements.

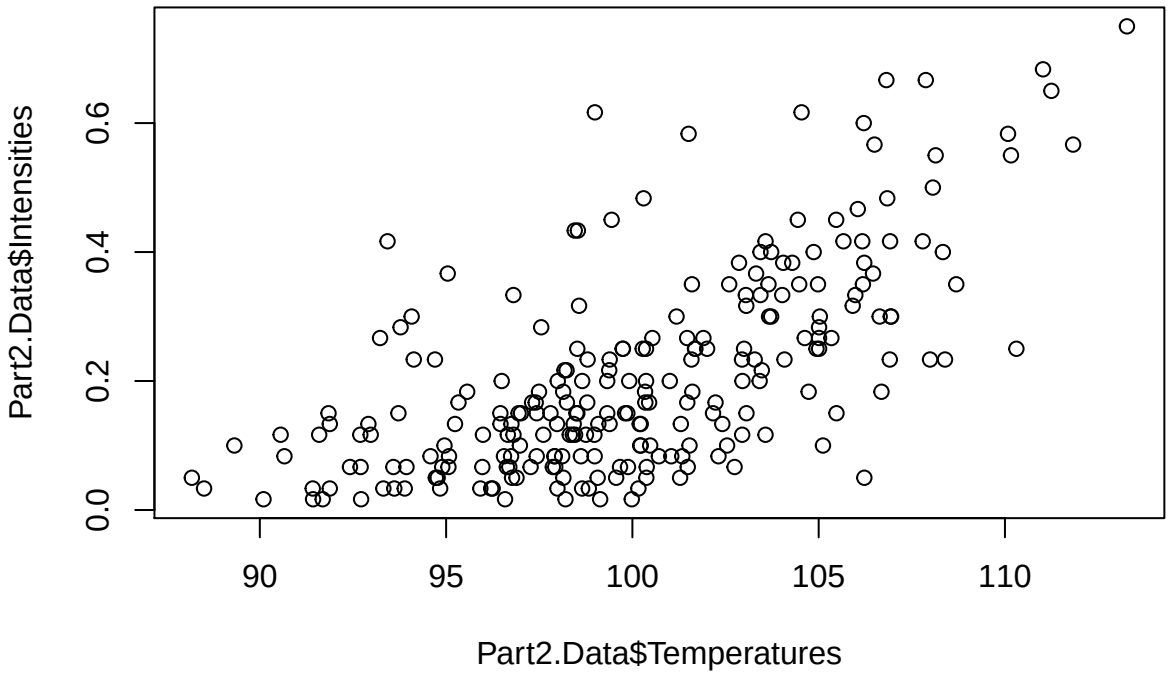
Create data frame with necessary data.

```
Part2.Data<-read.csv(file="C:/Users/Patrick/Documents/R/UChicago/Linear_NonLinear/MScA_LinearNonLinear_
Part2.Data<-as.data.frame(cbind(Part2.Data,Part2.Data[,2]/60))
colnames(Part2.Data)<-c("Times","Counts","Temperatures","Intensities")
head(Part2.Data)
```

```
##   Times Counts Temperatures Intensities
## 1    30      7     91.59307  0.11666667
## 2    90     10     97.30860  0.16666667
## 3   150      7     95.98865  0.11666667
## 4   210      4    100.38440  0.06666667
## 5   270      1     99.98330  0.01666667
## 6   330      6    102.54126  0.10000000
```

Visualize the data.

```
plot(Part2.Data$Temperatures,Part2.Data$Intensities)
```

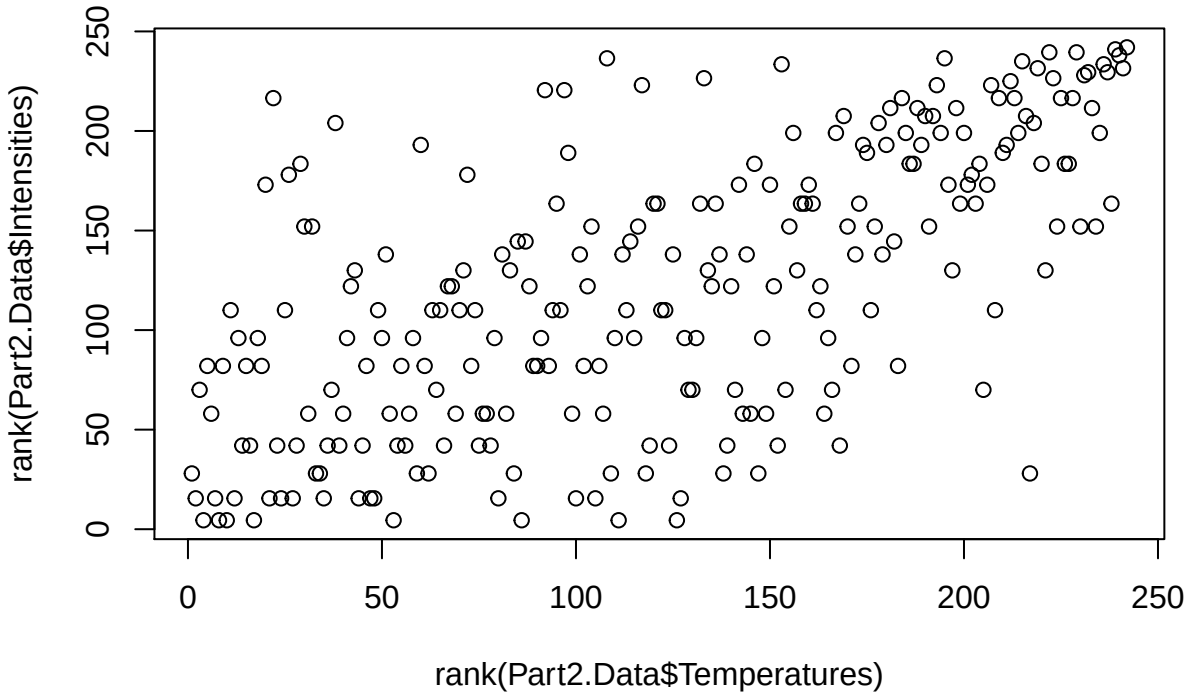


Interpret the plot. What type of relationship you observe?

The plot above defintiely suggest a positive relationship between the temperature and intensity.

Analyze empirical copula.

```
plot(rank(Part2.Data$Temperatures),rank(Part2.Data$Intensities))
```



What type of dependency you see in the empirical copula?

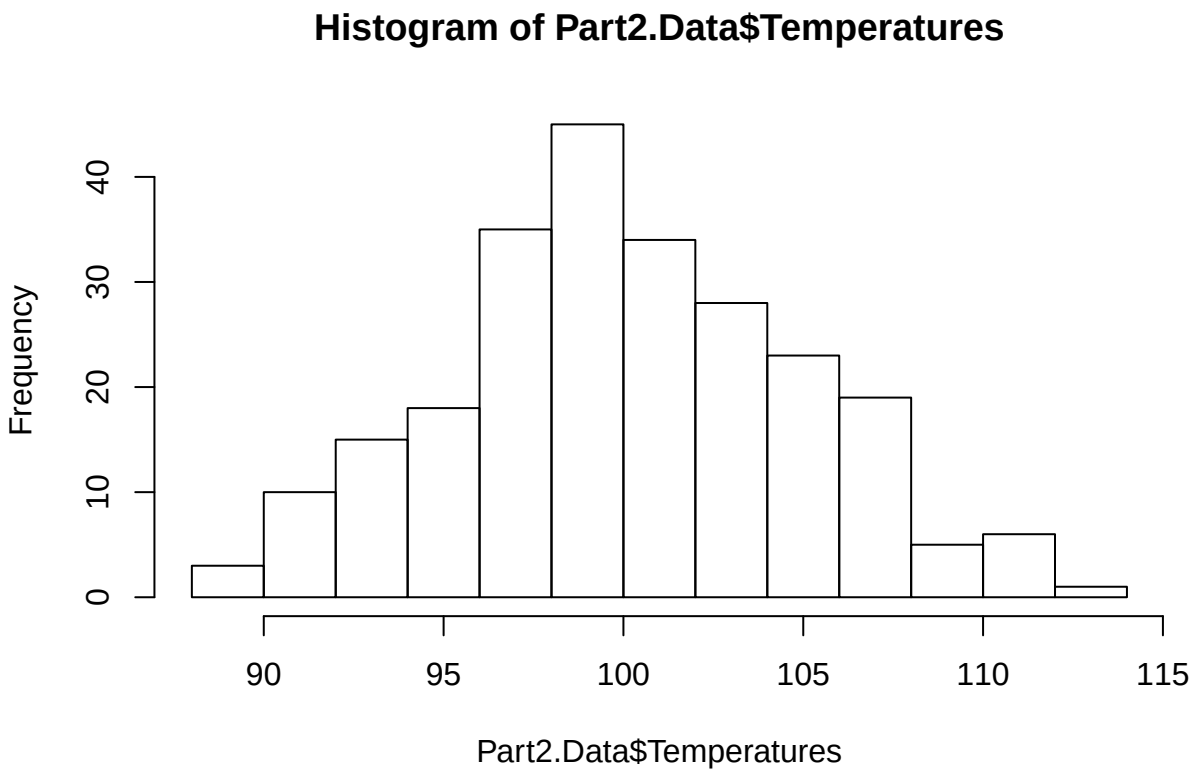
The empirical copula follows the findings from above that there is a positive relationship in the data.

What is the distribution of temperatures? Load package `MASS` to estimate distributions

```
library(MASS)
```

Observe the histogram

```
hist(Part2.Data$Temperatures)
```



Estimate and test normal distribution.

```
Fit.Temp.Normal <- fitdistr(Part2.Data$Temperatures,"normal")
Fit.Temp.Normal
```

```
##       mean           sd
##   100.0698530     4.8124839
##  (  0.3093582) (  0.2187493)
```

```
ks.test(Part2.Data$Temperatures,"pnorm",
        mean=Fit.Temp.Normal$estimate[1],sd=Fit.Temp.Normal$estimate[2])
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Part2.Data$Temperatures
## D = 0.0489, p-value = 0.6089
## alternative hypothesis: two-sided
```

Fit a copula

Select a parametric copula appropriate for this case.

Fit the copula and use it for simulation of rare events.

```
library(copula)

# cop1 <- normalCopula()
cop1 <- gumbelCopula()
p1 <- pobs(na.omit(Part2.Data[,3:4]), ties.method = "average")

Copula.Fit <- fitCopula(cop1,p1)

Copula.Fit
```

```
## fitCopula() estimation based on 'maximum pseudo-likelihood'
## and a sample of size 242.
##       Estimate Std. Error z value Pr(>|z|)
## param   1.8775     0.1146   16.38   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## The maximized loglikelihood is  74.18
## Optimization converged
## Number of loglikelihood evaluations:
## function gradient
##       11        4
```

Run longer simulation to observe more tail events Use the parameter estimates of the gamma distribution fitted to intensities.

```
Gamma.Rate<-8.132
Gamma.Shape<-1.656
```

and the parameter estimates of the normal distribution fitted to temperatures.

```
Fit.Temp.Normal$estimate
```

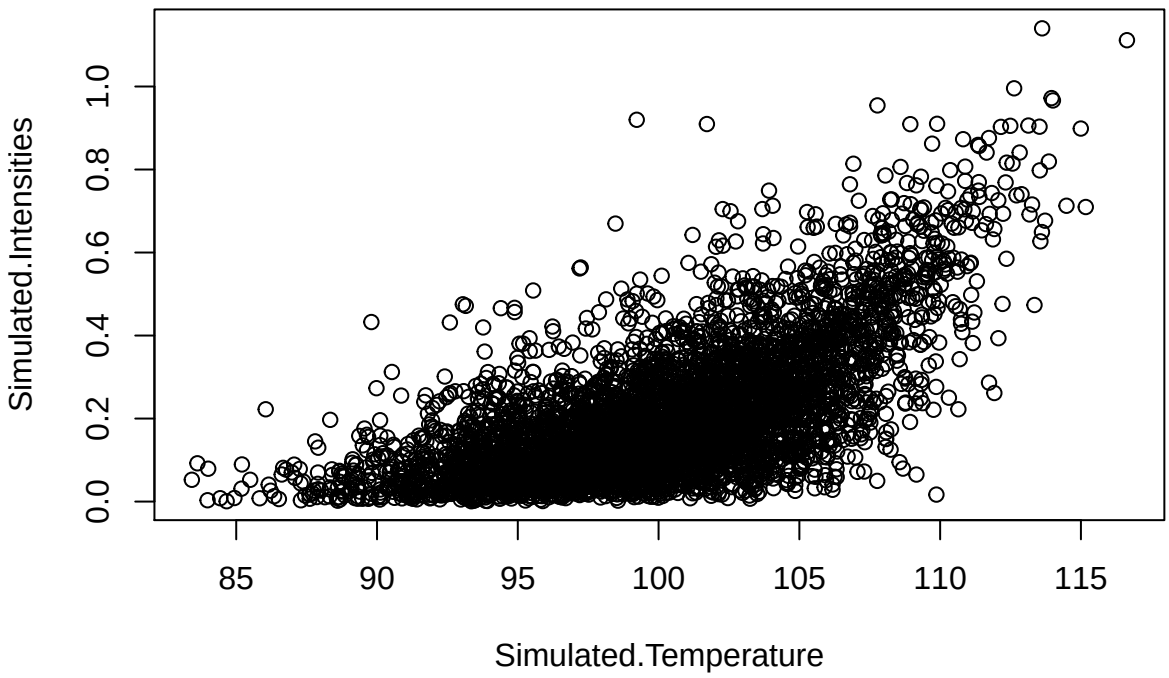
```
##       mean         sd
## 100.069853   4.812484
```

Simulate 5000 pairs of intensities and temperatures using the estimated copula.

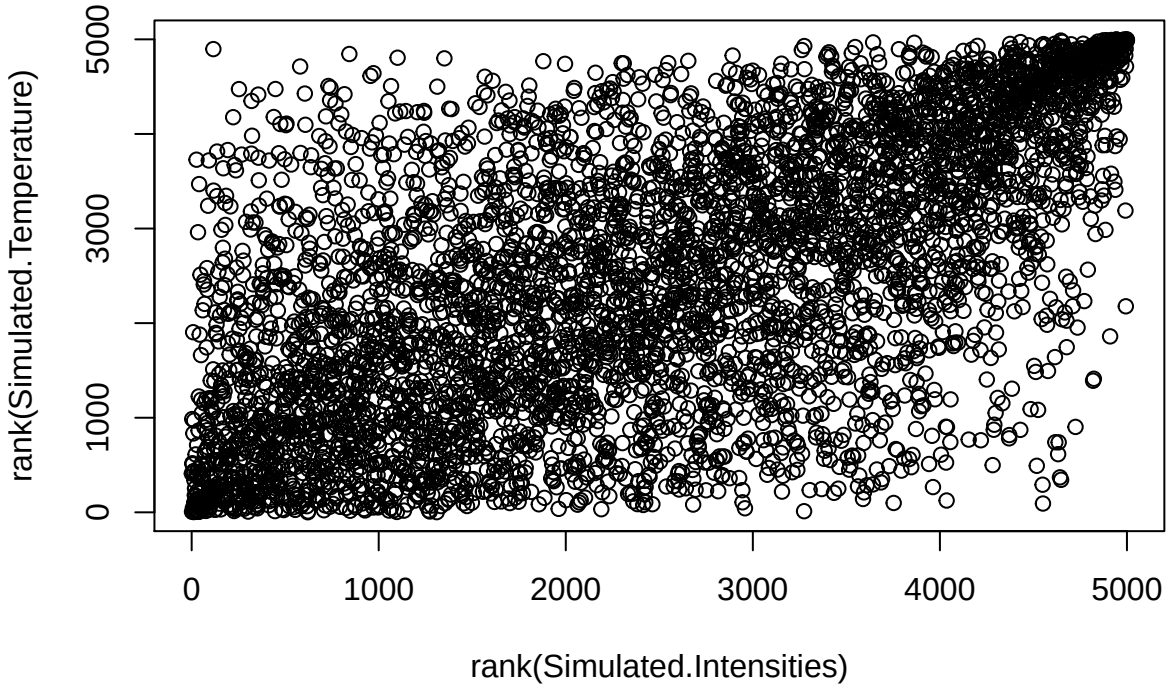
```
Defined.Copula <- gumbelCopula(param = 1.8775, dim = 2)

Simulated.Copula<-rCopula(5000,Defined.Copula)
Simulated.Intensities<-qgamma(Simulated.Copula[,1],shape=Gamma.Shape,scale=1/Gamma.Rate)
Simulated.Temperature<-qnorm(Simulated.Copula[,2],Fit.Temp.Normal$estimate[1],Fit.Temp.Normal$estimate[
```

Plot the simulated variables and their empirical copula.

```
plot(Simulated.Temperature,Simulated.Intensities)
```




```
plot(rank(Simulated.Intensities),rank(Simulated.Temperature))
```



Now we can use the simulated data to analyze the tail dependency. Select the simulated pairs wit intensity greater than 0.5 and temperature greater than 110. Use these data to fit negative binomial regression.

Use the initial sample of intensities and temperatures to fit the negative binomial regression for more regular ranges of intensity and temperature.

First, run fit the model to the sample.

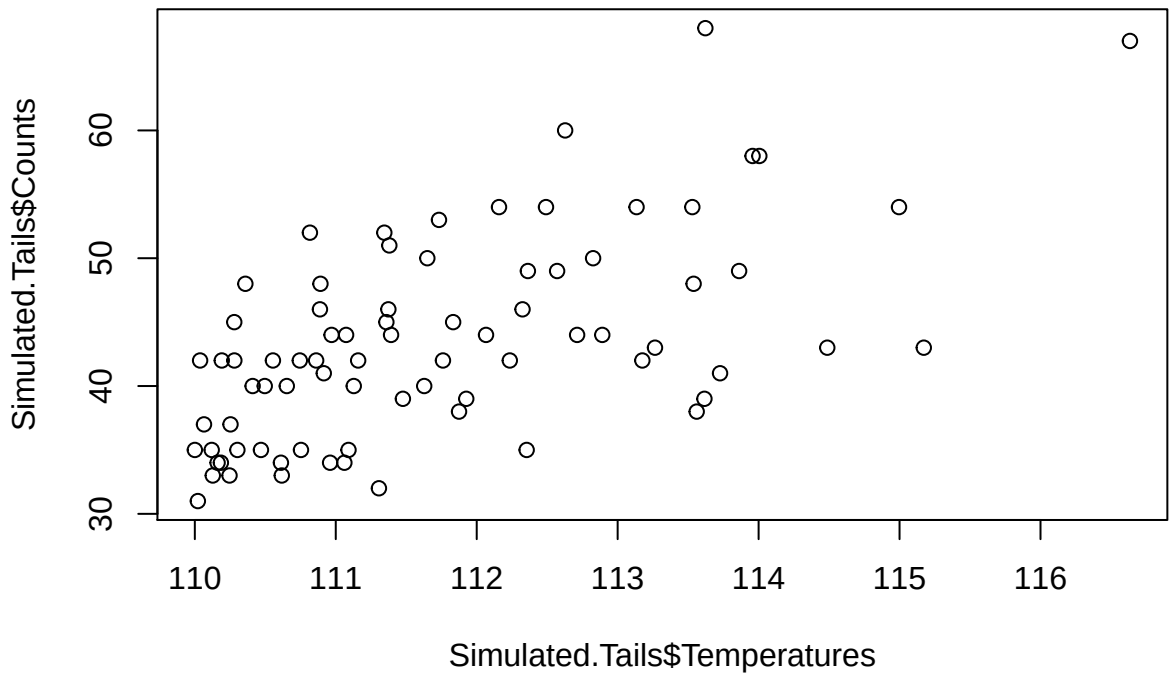
```
m1 <- glm.nb(formula = Counts ~ Temperatures, data = Part2.Data)
summary(m1)
```

```
##
## Call:
## glm.nb(formula = Counts ~ Temperatures, data = Part2.Data, init.theta = 4.202611755,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5358  -0.9162  -0.1509   0.4795   3.2190
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.431375   0.785456  -9.461   <2e-16 ***
## Temperatures  0.098432   0.007793  12.631   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for Negative Binomial(4.2026) family taken to be 1)
##
##     Null deviance: 431.60  on 241  degrees of freedom
## Residual deviance: 257.19  on 240  degrees of freedom
## AIC: 1557.8
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  4.203
##           Std. Err.:  0.549
##
##  2 x log-likelihood:  -1551.817
```

Create the simulated sample for tail events.

```
Simulated.Tails<-as.data.frame(
  cbind(round(Simulated.Intensities[(Simulated.Temperature>110)&(Simulated.Intensities>.5)]*60),
        Simulated.Temperature[(Simulated.Temperature>110)&(Simulated.Intensities>.5)]))
colnames(Simulated.Tails)<-c("Counts","Temperatures")

plot(Simulated.Tails$Temperatures,Simulated.Tails$Counts)
```




```
#model for the sample tail events
m2 <- glm.nb(formula = Counts ~ Temperatures, data = Simulated.Tails)
```

```
## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached

## Warning in theta.ml(Y, mu, sum(w), w, limit = control$maxit, trace =
## control$trace > : iteration limit reached
```

```
summary(m2)
```

```
##
## Call:
## glm.nb(formula = Counts ~ Temperatures, data = Simulated.Tails,
##     init.theta = 542648.0581, link = log)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.75387  -0.75422  -0.01717   0.58108   2.47028
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.38582    1.27861  -3.430 0.000603 ***
## Temperatures  0.07296    0.01143   6.386 1.71e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(542648.1) family taken to be 1)
##
##     Null deviance: 107.394  on 77  degrees of freedom
## Residual deviance:  67.911  on 76  degrees of freedom
## AIC: 510.62
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  542648
##           Std. Err.:  12206642
## Warning while fitting theta: iteration limit reached
##
##  2 x log-likelihood:  -504.621
```

Compare the summaries of the two models. Note that the parameter ?? estimated by glm.nb() defines the variance of the model as ??+??2/??, where ?? is the mean. In other words, ?? defines overdispersion.

What do the fitted parameters ?? tell you about both models

The higher theta values correspond with lower dispersion (or less over-dispersion) within the data. Therefore higher ?? values indicate that the distribution is closer to a Poisson distribution (rather than a neg. binomial). Since we see a much higher ?? value in the simulated tails and a low ?? value in the model of the entire data, we can conclude that the simulated tails data has less over-dispersion. This also intuitively makes sense given the model fit, as the extreme values are more likely to have a more predictable increase in malfunctions - in comparison to the variablility of occurances when the temperature is within a normal / non-extreme range.

Is there an alternative model that you would try to fit to the simulated tail data?

Based on the above explanation, I would try to fit a Poisson model as an alternative for the simulated tail data.

What do both models tell you about the relationships between the temperature and the counts?

Both models have similiar coefficients that indicate that there is a significant positive relationship between the temperature and malfunction counts.