Linear & NonLinear Project

Adham Suliman

July 17, 2018

1. & 2) & 3)

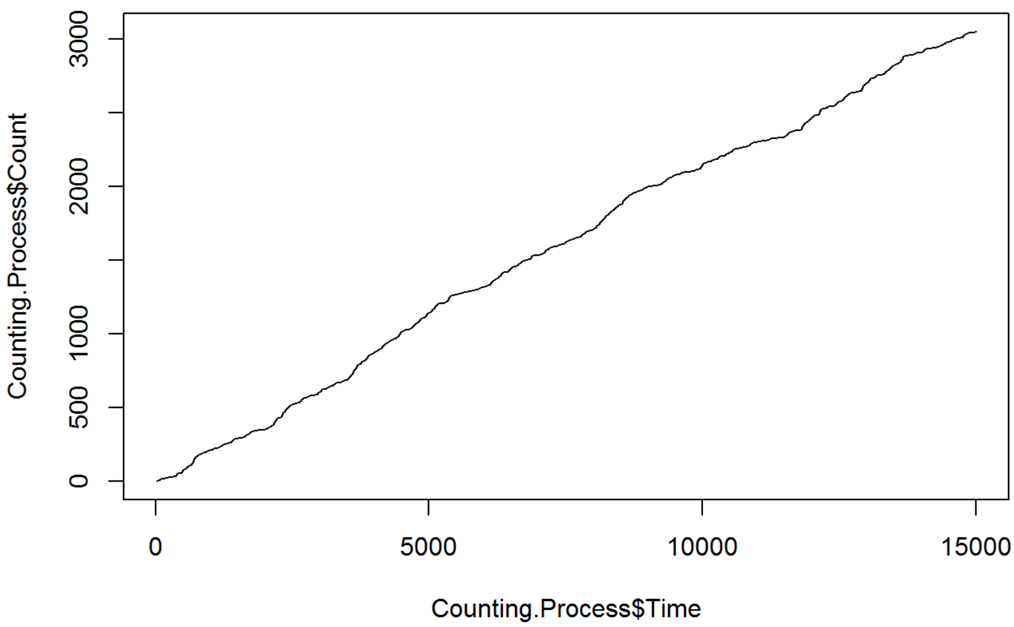
```
Course.Project.Data <- read.csv('LinearNonLinear_MalfunctionData.csv', header = T)
Counting.Process<-as.data.frame(cbind(Time=Course.Project.Data$Time,Count=1:length(Course.Project.Data$Time)
))
Counting.Process[1:20,]
```

```
##         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
```

```
max(Counting.Process$Time)
```

```
## [1] 14999.94
```

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



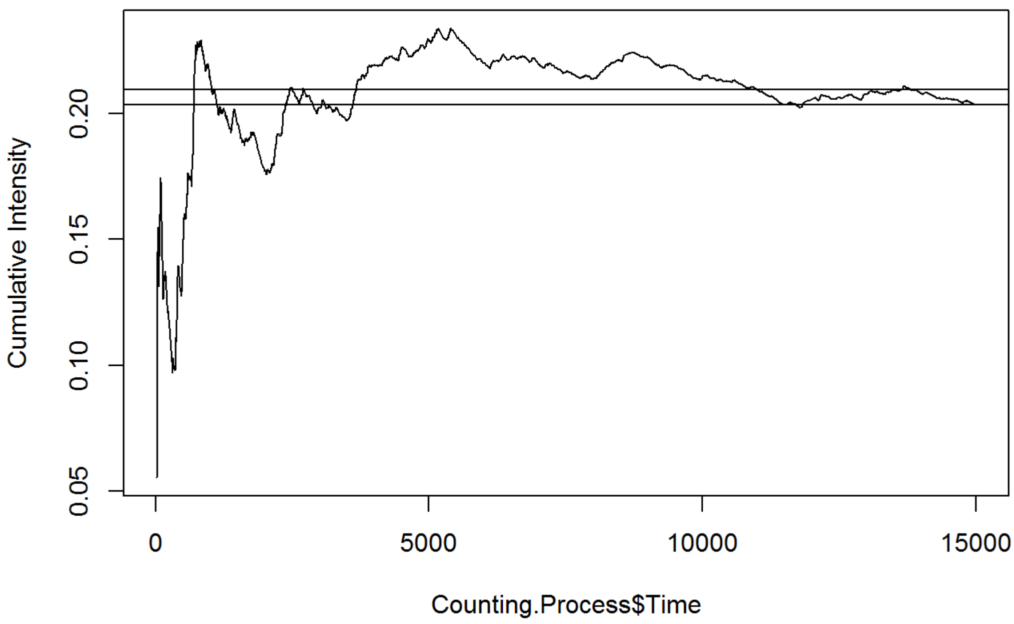
The counting process

trajectory looks pretty smooth and grows steadily.

What does it tell you about the character of malfunctions and the reasons causing them? This tells me that as time goes on, there seems to be a pretty constant count which could mean an average rate or “intensity” of the issue occuring over time.

3.1)

```
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))
```

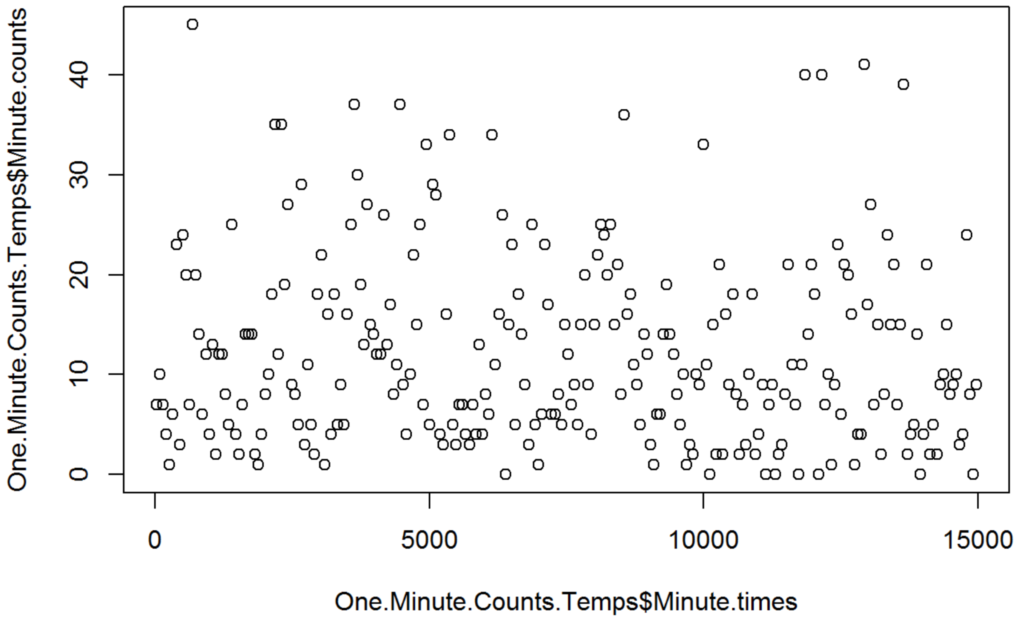



```
c(Last.Intensity=Counting.Process$Count[length(Counting.Process$Count)]/
         Counting.Process$Time[length(Counting.Process$Time)],
  Mean.Intensity=mean(Counting.Process$Count/Counting.Process$Time))
```

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

4.

```
Course.Project.Data$Minute <- floor(Course.Project.Data$Time/60)
Course.Project.Data1 <-
  Course.Project.Data%>%
    dplyr::select(Minute, Temperature)%>%
    dplyr::group_by(Minute) %>%
    dplyr::summarise(Minute.counts = c(n()),Miunute.Temps = c(mean(Temperature)))
One.Minute.Counts.Temps<-   Course.Project.Data1%>%
                            dplyr::mutate(Minute.times=c((Minute*60)+30))%>%
                            dplyr::select(Minute.times, Minute.counts, Miunute.Temps)
NCourse.Project.Data <-vector()
for (x in 1:250){
  NCourse.Project.Data[x]<-(30+(60*(x-1)))
}
NCourse.Project.Data <- as.data.frame(matrix(NCourse.Project.Data,ncol=1),col.names="Minute.times")
colnames(NCourse.Project.Data)="Minute.times"
One.Minute.Counts.Temps <-
  NCourse.Project.Data %>%
  left_join(One.Minute.Counts.Temps,by="Minute.times")
One.Minute.Counts.Temps$Minute.counts[is.na(One.Minute.Counts.Temps$Minute.counts)] <- 0
plot(One.Minute.Counts.Temps$Minute.times,One.Minute.Counts.Temps$Minute.counts)
```



Poisson Data

```
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
```

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

```
## [1] 272
```

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

```
## (Intercept)
##       1.947
```

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 the one minute event counts Do you see signs of over-dispersion? Alpha is greater than 0 due to the signicant p-value below which shows over disperison.

```
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
```

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

```
##
##  Overdispersion test
##
## data:  GLM.model
## z = 8.5747, p-value < 2.2e-16
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##   7.377975
```

Q: Does the test show overdispersion? A: Yes it does because the alpha is above 0 and pour p-value is significant which shows over dispersion.

4.1.3 Test against negative binomial distribution Definitely not an NB model.

```
GLM.NM<-glm.nb(Minute.counts~1,One.Minute.Counts.Temps)
summary(GLM.NM)
```

```
##
## Call:
## glm.nb(formula = Minute.counts ~ 1, data = One.Minute.Counts.Temps,
##     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
```

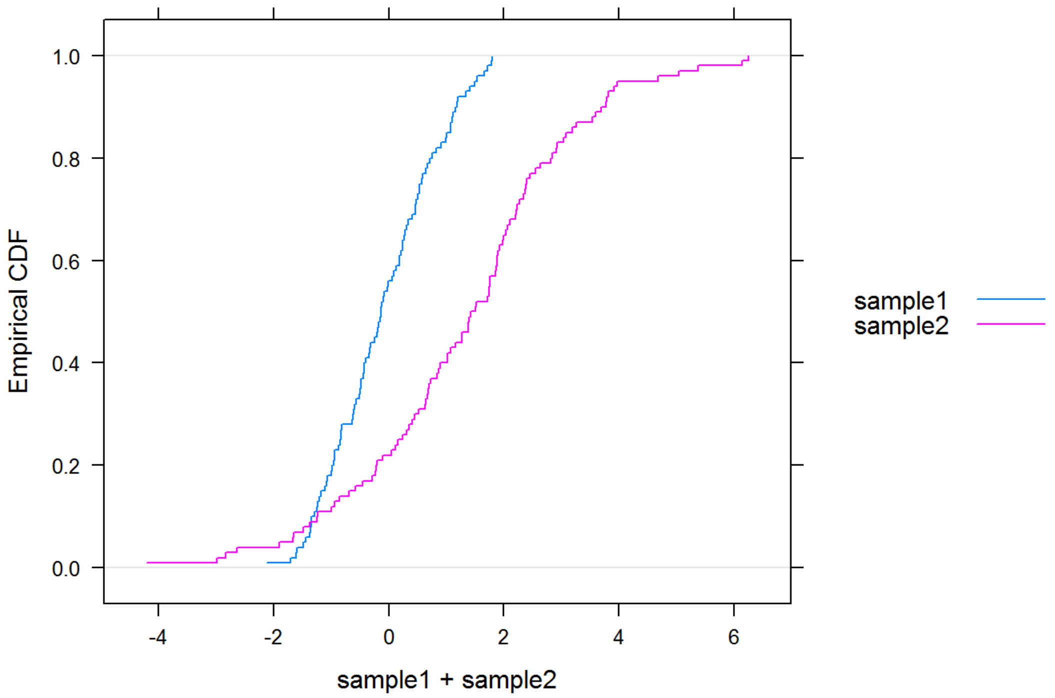
```
odTest(GLM.NM)
```

```
## 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
```

Does this test show overdispersion? Yes it does due to the significant P-value.

5. Find the distribution of Poisson Intensity 5.1 Kolmlgrorov-Sminrnov Test

```
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 = 0.48, p-value = 1.972e-10
## alternative hypothesis: two-sided
```

What does this output tell you about equivalence of the two distributions? Because the p-value is significant, these samples have come from different populations.

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

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sample1
## D = 0.072997, p-value = 0.6609
## alternative hypothesis: two-sided
```

What does this output tell you? Because the p-value is not significant, the 1st samples does come from a population with a normal distribution N(0,1).

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

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  sample2
## D = 0.45719, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Check equivalence of the empirical distribution of sample2 and theoretical distribution Norm(0,1). The second sample comes from a population that doesn't have a normal distribution N(0,1).

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

```
Time.diff <- diff(Counting.Process$Time)
intensity <-1:length(Counting.Process$Time)/(Counting.Process$Time)
mean.intensity <- mean(intensity)
KS.Test.Event.Interval <- ks.test(Time.diff,"pexp",rate=mean.intensity)
c(KS.Test.Event.Interval$statistic,p.value=KS.Test.Event.Interval$p.value)
```

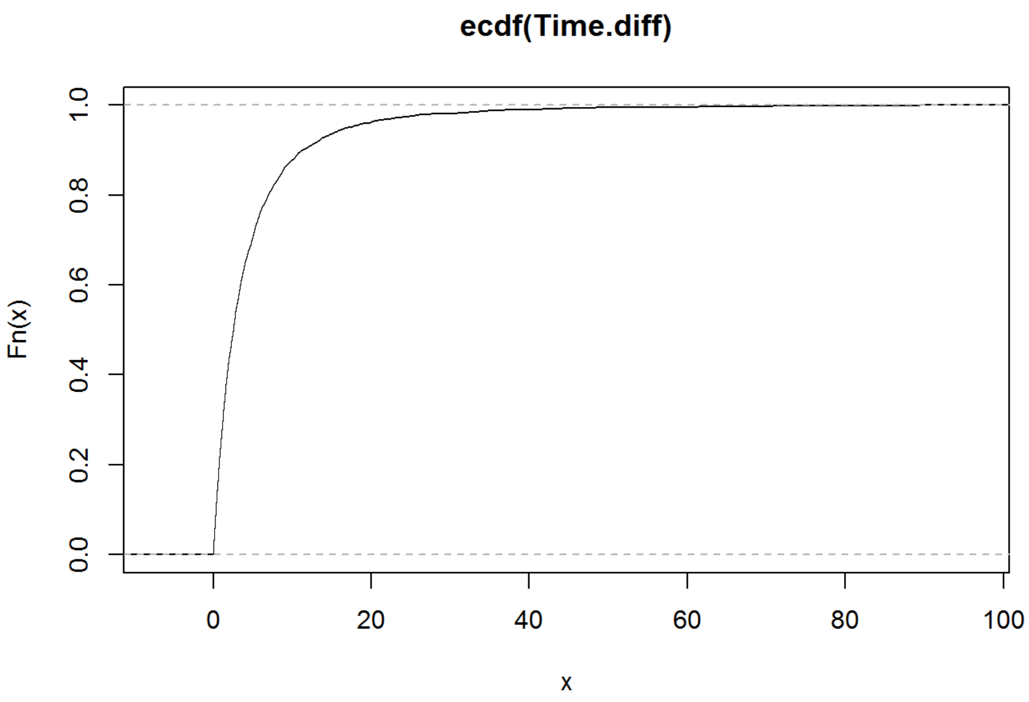
```
##         D   p.value
## 0.0950615 0.0000000
```

theoretical exponential distribution with parameter equal to average intensity.

```
Time.diff <- diff(Counting.Process$Time)
ks.test(Time.diff,"pexp",mean(Time.diff),sd(Time.diff))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Time.diff
## D = 0.78677, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

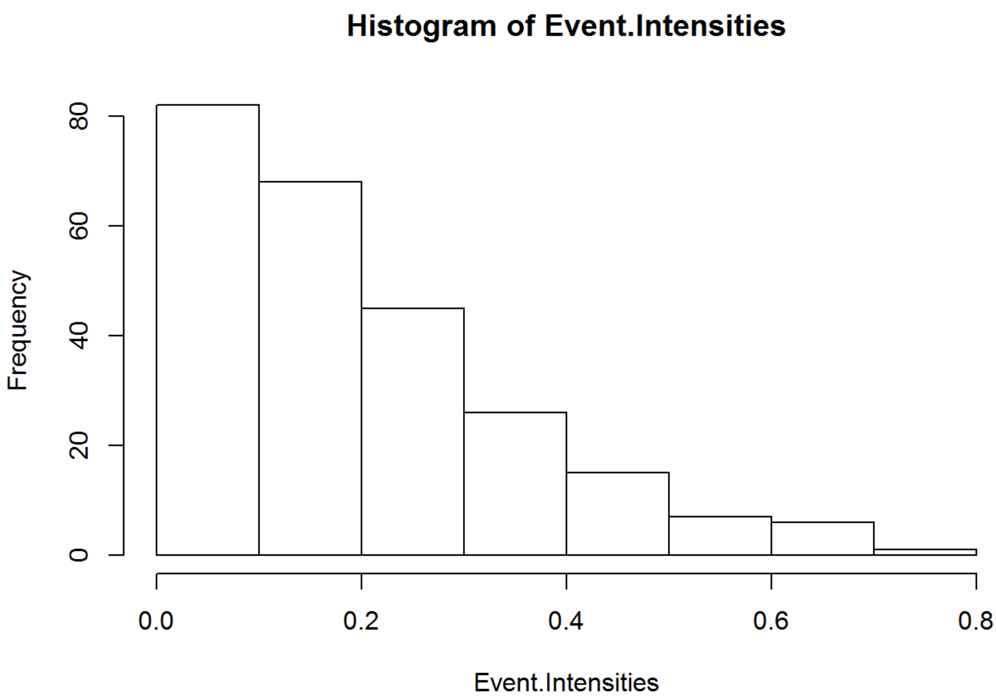
```
plot(ecdf(Time.diff))
```




```
order(One.Minute.Counts.Temps$Minute.counts,decreasing=T)
```

```
##   [1]  12 216 198 203 228  61  75 143  37  39  90 103  83 167  62  45  85  86  41  65 218  70 106
##  [24]  24  60  81 115 136 139   9 137 223 247   7 109 119 208  51  79 135 141 172 193 200 210 225
##  [47] 235  10  13 131 138 211  40  63 156  36  50  55 111 145 176 182 201  72 120 217  53  59  89
##  [70] 105 144 174 212  66  80 108 125 130 134 140 170 220 224 227 241  14  28  29  30  67 112 149
##  [93] 155 157 199 232  18  64  71  99  16  20  21  38  68  69 126 150 158  47  74 104 146 168 194
## [116] 197   2  35  78 161 165 181 205 240 244  42  57  76 113 128 132 147 166 175 185 188 207 239
## [139] 243 250  22  34  43  73 101 123 142 159 177 192 222 242 248   1   3  11  27  82  93  94  97
## [162] 127 179 187 195 204 219 226   6  15 102 118 121 122 153 154 209  23  44  48  56  58  84  91
## [185] 110 116 124 129 148 160 231 237   4  17  25  33  54  77  87  95  98 100 133 184 214 215 230
## [208] 234 246   8  46  88  92  96 114 151 163 180 191 245  19  26  31  49 164 171 173 178 183 190
## [231] 221 229 236 238   5  32  52 117 152 162 206 213 107 169 186 189 196 202 233 249
```

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



What distribution does this histogram remind you of? This distributino reminds me of a poissoin distribution.

```
start <- seq(0,5,.2)
(Fitting.Normal <- fitdistr(Event.Intensities,densfun="normal"))
```

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

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

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

Logs

```
Event.Intensities1 <- Event.Intensities + 10e-15
(Fitting.Logs<-fitdistr(Event.Intensities1,densfun="lognormal"))
```

```
##     meanlog       sdlog   
##   -2.8474770    5.4111039 
##  ( 0.3422283) ( 0.2419919)
```

```
Fitting.Logs
```

```
##     meanlog       sdlog
##   -2.8474770    5.4111039
##  ( 0.3422283) ( 0.2419919)
```

```
KS.LogNormal <- ks.test(Event.Intensities1,"plnorm",meanlog= -2.8474770 ,sd= 5.4111039)
```

```
## Warning in ks.test(Event.Intensities1, "plnorm", meanlog = -2.847477, sd = 5.4111039): ties should
## not be present for the Kolmogorov-Smirnov test
```

```
KS.LogNormal
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities1
## D = 0.39525, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Beta

```
mu <- mean(Event.Intensities)
var <- var(Event.Intensities)
estBetaParams <- function(mu, var) {
  alpha <- ((1 - mu) / var - 1 / mu) * mu ^ 2
  beta <- alpha * (1 / mu - 1)
  return(params = list(alpha = alpha, beta = beta))
}
x <- estBetaParams(mu,var)
x[2]
```

```
## $beta
## [1] 4.340912
```

```
KS.beta <- ks.test(Event.Intensities,"pbeta", shape1=1.109756,shape2=4.340912)
```

```
## Warning in ks.test(Event.Intensities, "pbeta", shape1 = 1.109756, shape2 = 4.340912): ties should
## not be present for the Kolmogorov-Smirnov test
```

Exponential distribution should be closer but it’s not.

```
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
```

```
names(Fitting.Exponential)
```

```
## [1] "estimate" "sd"       "vcov"     "n"        "loglik"
```

```
KS.Exp <- ks.test(Event.Intensities,"pexp",rate=Fitting.Exponential$estimate[1])
```

```
## Warning in ks.test(Event.Intensities, "pexp", rate = Fitting.Exponential$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
```

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

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

What do you conclude from these tests? These tests are telling that the data doesn't fit a normal distribution nor an exponential distribution because of the significant p values.

Try to fit gamma distribution directly using fitdistr()

```
#Fitting.Gamma <- fitdistr(Event.Intensities,densfun="gamma")
```

NANs are prouced, but why? Since there is a negative value within the data, the gamma distribution cannot be utilized because it can take only positive values.

Intensity is gamma with a normal distribution.

```
first.moment <- mean(Event.Intensities)
second.moment <- var(Event.Intensities)*(length(Event.Intensities))/(length(Event.Intensities)+1)
beta <- first.moment/second.moment
alpha <- first.moment*beta
(Moments.Rate <- beta)
```

```
## [1] 8.132182
```

```
(Moments.Shape <- alpha)
```

```
## [1] 1.655712
```

```
KS.Test.Moments.Gamma <- ks.test(Event.Intensities,"pgamma",rate=beta, shape=alpha)
```

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

```
KS.Test.Moments.Gamma
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Event.Intensities
## D = 0.057807, p-value = 0.3737
## alternative hypothesis: two-sided
```

What distribution for the one-minute intensity of malfunctions do you choose? In the end, I would choose the beta distribution due to it's high P value.

What distribution of one-minute malfunctions counts follow from your choice? If I couldn't use the beta distribution, I would choose the gamma distribution.

```
rbind(KS.Moments=c(KS.Test.Moments.Gamma$statistic,P.Value=KS.Test.Moments.Gamma$p.value),
      KS.beta=c(KS.beta$statistic,P.Value=KS.beta$p.value),
      KS.LogNormal=c(KS.LogNormal$statistic,P.Value=KS.LogNormal$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   0.05780734 0.3736786147
## KS.beta      0.05239816 0.4985637180
## KS.LogNormal 0.39524722 0.0000000000
## KS.Exp       0.11523305 0.0026158124
## KS.Normal    0.13260203 0.0003039941
```

```
write.csv(One.Minute.Counts.Temps,file="OneMinuteCountsTemps.csv",row.names=FALSE)
Part2.Data<-read.csv("OneMinuteCountsTemps.csv")
head(Part2.Data)
```

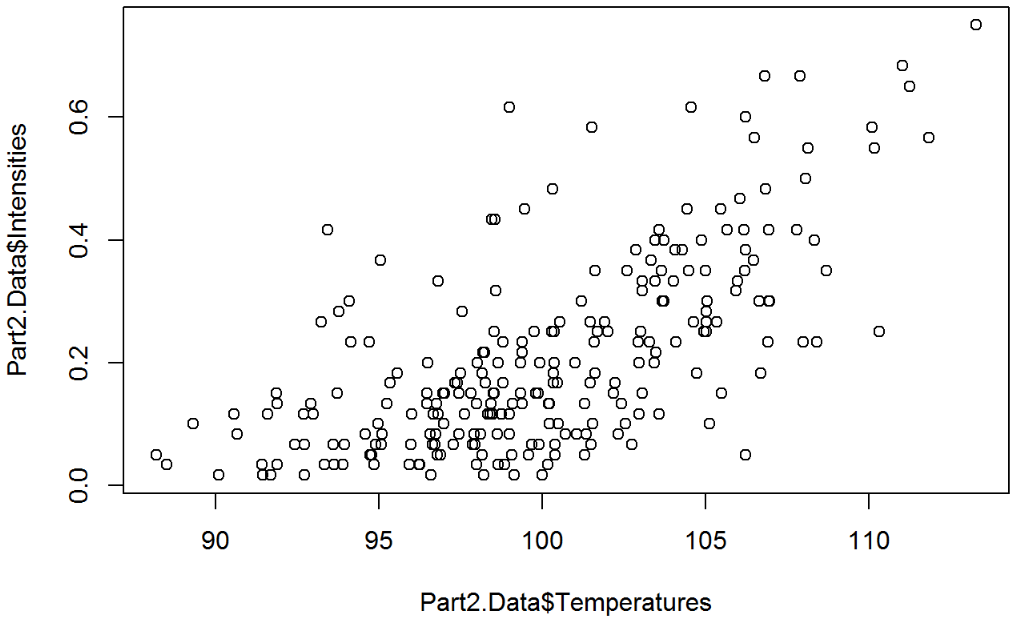
```
##   Minute.times Minute.counts Miunute.Temps
## 1           30             7      91.59307
## 2           90            10      97.30860
## 3          150             7      95.98865
## 4          210             4     100.38440
## 5          270             1      99.98330
## 6          330   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

```
Part2.Data<-Part2.Data[complete.cases(Part2.Data),]
```

```
Part2.Data<-as.data.frame(cbind(Part2.Data,Part2.Data[,2]/60))
colnames(Part2.Data)<-c("Times","Counts","Temperatures","Intensities")
head(Part2.Data)
```

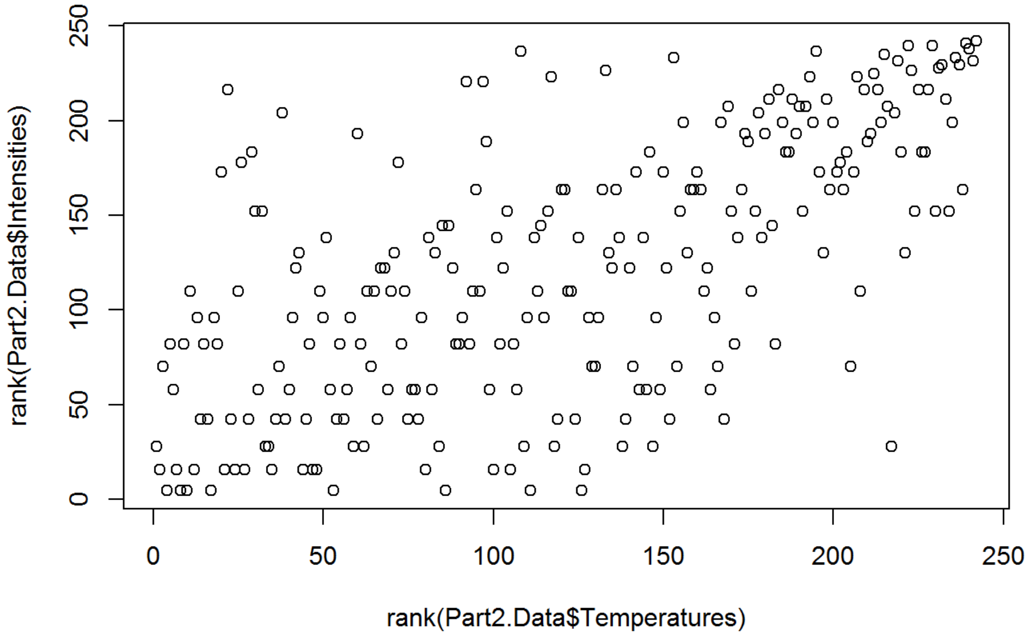
```
##   Times Counts Temperatures Intensities
## 1    30      7     91.59307  0.11666667
## 2    90     10     97.30860  0.16666667
## 3   150      7     95.98865  0.11666667
## 4   210      4    100.38440  0.06666667
## 5   270      1     99.98330  0.01666667
## 6   330      6    102.54126  0.10000000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

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



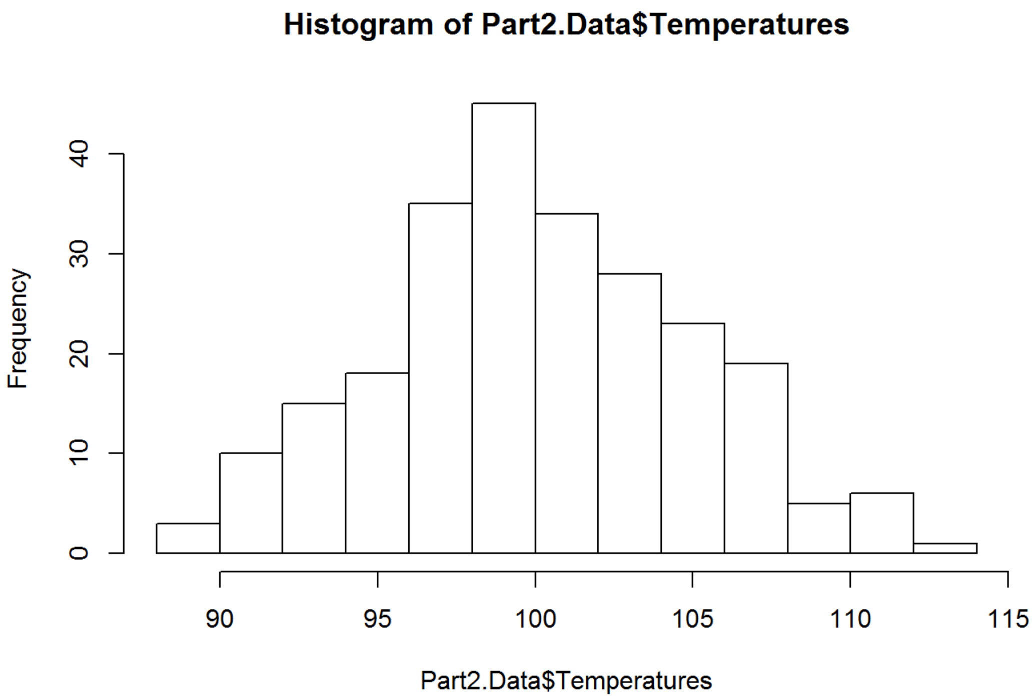
Interpret the plot. What type of relationship do you observe? There initialy seems to be a high amount of data points in the temperatures between 95 to 105 where intensity is between 0 and .3.

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



What type of dependency you see in the empirical copula? It seems as temperature increases, so does intensity but there data does seem to be quite sparsed.

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




```
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
```

```
Part2.Norm <- fitdistr(Part2.Data$Temperatures,"normal")
Part2.Norm$n
```

```
## [1] 242
```

```
ks.test(Part2.Data$Temperatures,"pnorm",mean=100.069853 ,sd=4.812484  )
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  Part2.Data$Temperatures
## D = 0.048912, p-value = 0.6089
## alternative hypothesis: two-sided
```

data intensities

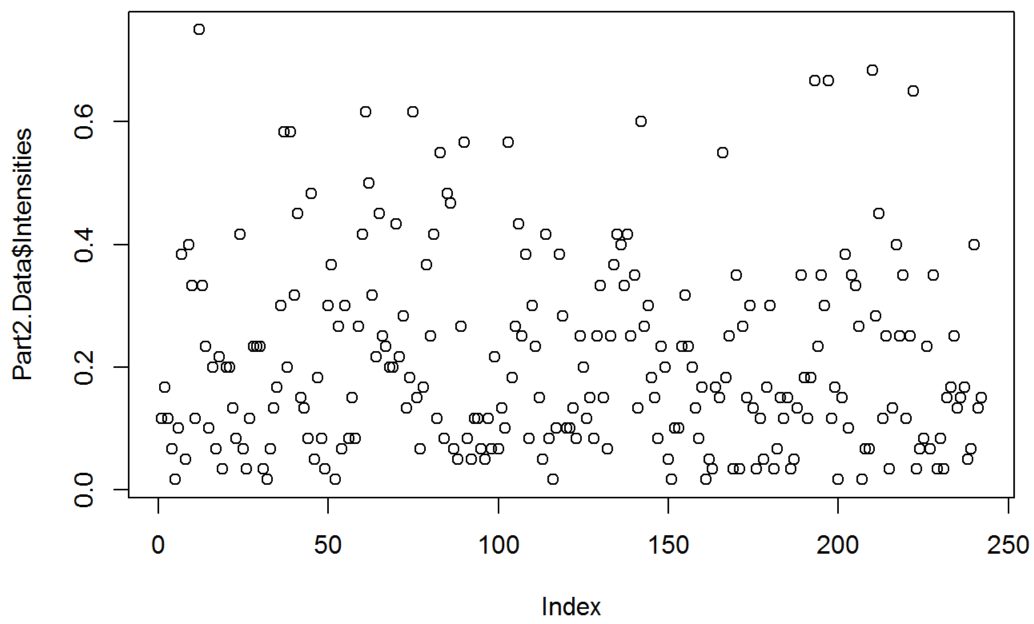
```
Fitting.Copula <- cbind(pobs(Part2.Data$Temperatures, ties.method = "average"),pobs(Part2.Data$Intensities,
ties.method = "average"))
Gumbelcopula<-gumbelCopula(dim=2)
Copula.Fit<-fitCopula(Gumbelcopula,
                              pobs(Fitting.Copula,ties.method = "average"),
                              method = "ml",
                              optim.method = "BFGS",
                              optim.control = list(maxit=1000))

summary(Copula.Fit)
```

```
## Call: fitCopula(copula, data = data, method = "ml", optim.method = "BFGS",
##     optim.control = ..3)
## Fit based on "maximum likelihood" and 242 2-dimensional observations.
## Gumbel copula, dim. d = 2
##       Estimate Std. Error
## alpha    1.877      0.099
## The maximized loglikelihood is 74.18
## Optimization converged
## Number of loglikelihood evaluations:
## function gradient
##       11        4
```

```
Copula.Fit@estimate
```

```
## [1] 1.877455
```

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



https://stats.stackexchange.com/questions/90729/generating-values-from-copula-using-copula-package-in-r

```
Fitting.Copula <- cbind(pobs(Part2.Data$Temperatures, ties.method = "average"),pobs(Part2.Data$Intensities,
ties.method = "average"))
Gumbelcopula<-gumbelCopula(dim=2)
(Copula.Fit<-fitCopula(Gumbelcopula,
                                pobs(Fitting.Copula,ties.method = "average"),
                                method = "ml",
                                optim.method = "BFGS",
                                optim.control = list(maxit=1000)))
```

```
## Call: fitCopula(copula, data = data, method = "ml", optim.method = "BFGS",
##     optim.control = ..3)
## Fit based on "maximum likelihood" and 242 2-dimensional observations.
## Copula: gumbelCopula
## alpha
## 1.877
## The maximized loglikelihood is 74.18
## Optimization converged
```

```
summary(Copula.Fit)
```

```
## Call: fitCopula(copula, data = data, method = "ml", optim.method = "BFGS",
##     optim.control = ..3)
## Fit based on "maximum likelihood" and 242 2-dimensional observations.
## Gumbel copula, dim. d = 2
##       Estimate Std. Error
## alpha    1.877      0.099
## The maximized loglikelihood is 74.18
## Optimization converged
## Number of loglikelihood evaluations:
## function gradient
##       11        4
```

```
set.seed(8301735)
Simulated.Copula <- as.data.frame(rCopula(copula=gumbelCopula(Copula.Fit@estimate, dim=2), n=250))
Simulated.Copula
```

```
##               V1          V2
## 1   0.346294799 0.279039922
## 2   0.997172131 0.992229381
## 3   0.617043299 0.019520077
```

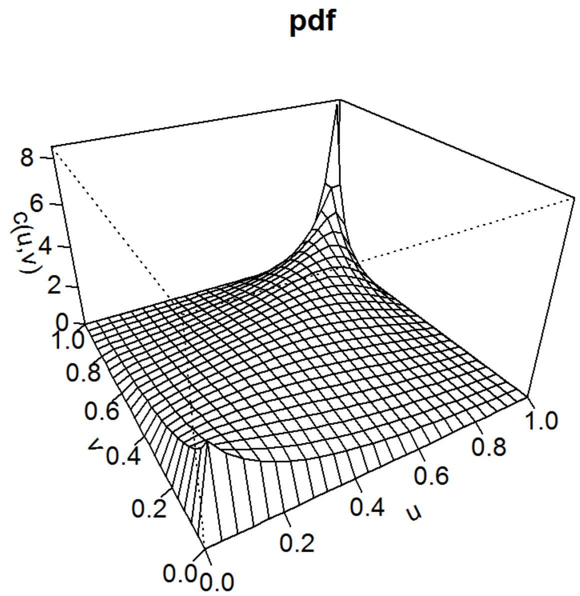
```
## 4   0.259385911 0.650830834
## 5   0.270713152 0.027871947
## 6   0.889982559 0.571358980
## 7   0.066110585 0.182890724
## 8   0.107686727 0.519409285
## 9   0.174084135 0.057772586
## 10  0.883338242 0.986061524
## 11  0.484448369 0.161447133
## 12  0.036373304 0.441940142
## 13  0.071734919 0.041168541
## 14  0.553966883 0.734009767
## 15  0.911372278 0.896681713
## 16  0.451324353 0.456238931
## 17  0.260699761 0.488324668
## 18  0.069794134 0.235534249
## 19  0.034087851 0.056572140
## 20  0.038971572 0.179960052
## 21  0.598372551 0.830605495
## 22  0.317778308 0.893942225
## 23  0.888525850 0.772224198
## 24  0.876170025 0.757074080
## 25  0.107540518 0.168576621
## 26  0.733493823 0.772071733
## 27  0.997287721 0.989353221
## 28  0.301781325 0.082714274
## 29  0.954254848 0.962999037
## 30  0.285038849 0.165979563
## 31  0.403090345 0.076256378
## 32  0.645124729 0.160697826
## 33  0.105289411 0.160321658
## 34  0.132595994 0.494041836
## 35  0.317081838 0.131602818
## 36  0.137762855 0.287327393
## 37  0.443768165 0.056579028
## 38  0.949075426 0.611099492
## 39  0.268700433 0.667986777
## 40  0.760405095 0.789516579
## 41  0.251139813 0.344854323
## 42  0.367915345 0.308020507
## 43  0.454789868 0.874486316
## 44  0.342607238 0.240798234
## 45  0.568934380 0.104565289
## 46  0.640065062 0.259939366
## 47  0.860498970 0.808327958
## 48  0.194798105 0.109639468
## 49  0.777644118 0.913489633
## 50  0.190313540 0.069054858
## 51  0.272439974 0.135127469
## 52  0.652139639 0.361091793
## 53  0.208002619 0.507944713
## 54  0.057171737 0.045306302
## 55  0.380310256 0.194585189
## 56  0.673724252 0.839219835
## 57  0.936417204 0.879231489
## 58  0.453455603 0.321425232
## 59  0.317959580 0.176589624
## 60  0.848136578 0.931204745
## 61  0.513045460 0.540847322
## 62  0.118653793 0.374282271
## 63  0.953245854 0.991251027
## 64  0.998450592 0.997083562
## 65  0.750936291 0.766497903
## 66  0.374496993 0.782559428
## 67  0.325793266 0.410983833
## 68  0.055211675 0.754305158
## 69  0.001389858 0.213389195
## 70  0.032047772 0.014793247
## 71  0.410798854 0.305235863
## 72  0.096852102 0.221267846
## 73  0.869480513 0.811375669
## 74  0.553959193 0.402743532
## 75  0.415478448 0.908425787
## 76  0.658439684 0.795733862
```

```
## 77  0.500836043 0.177773716
## 78  0.130732654 0.106418497
## 79  0.728538258 0.852964246
## 80  0.506675652 0.346848585
## 81  0.403298707 0.629401476
## 82  0.832650449 0.076076257
## 83  0.871363018 0.903385116
## 84  0.065991802 0.060336663
## 85  0.445598316 0.879416196
## 86  0.098689577 0.189565939
## 87  0.668126578 0.754923937
## 88  0.458781060 0.145391182
## 89  0.616431715 0.443456491
## 90  0.259097933 0.327504692
## 91  0.730065572 0.163266355
## 92  0.557631152 0.317416078
## 93  0.068532221 0.218659056
## 94  0.115281361 0.172396854
## 95  0.027752579 0.741416460
## 96  0.575933466 0.824566096
## 97  0.156754877 0.106585171
## 98  0.541532786 0.481062727
## 99  0.806167096 0.597349549
## 100 0.402087114 0.553718457
## 101 0.190958069 0.608116278
## 102 0.775272388 0.311561955
## 103 0.604865162 0.743569435
## 104 0.402323055 0.719873189
## 105 0.630530759 0.622330896
## 106 0.286804480 0.124848219
## 107 0.895228878 0.977903124
## 108 0.457403485 0.204075542
## 109 0.019012019 0.113302441
## 110 0.347121181 0.326088664
## 111 0.924273251 0.956132090
## 112 0.302643831 0.015504721
## 113 0.295327831 0.203679201
## 114 0.011634054 0.657250128
## 115 0.038694121 0.144764824
## 116 0.623539379 0.443042042
## 117 0.414168145 0.514396115
## 118 0.258503363 0.828638456
## 119 0.300207878 0.227086371
## 120 0.114565619 0.011527819
## 121 0.688129488 0.282178466
## 122 0.033312174 0.187025380
## 123 0.481654377 0.813641936
## 124 0.001295225 0.007634664
## 125 0.388263415 0.074220854
## 126 0.384896269 0.477730364
## 127 0.512750342 0.095197523
## 128 0.104899905 0.112143560
## 129 0.999663790 0.997337416
## 130 0.424814466 0.208653109
## 131 0.850785189 0.906961675
## 132 0.243503710 0.082212315
## 133 0.332952391 0.501024742
## 134 0.614142163 0.064234089
## 135 0.512008780 0.658044816
## 136 0.726432962 0.727007923
## 137 0.831474741 0.762327439
## 138 0.361176648 0.709568185
## 139 0.818923090 0.513360373
## 140 0.514335895 0.341335508
## 141 0.012815634 0.206161948
## 142 0.714056479 0.752363658
## 143 0.057016525 0.397479355
## 144 0.937796579 0.989328463
## 145 0.656339132 0.446326698
## 146 0.429358808 0.532604335
## 147 0.780332827 0.335831855
## 148 0.909961591 0.428400186
```

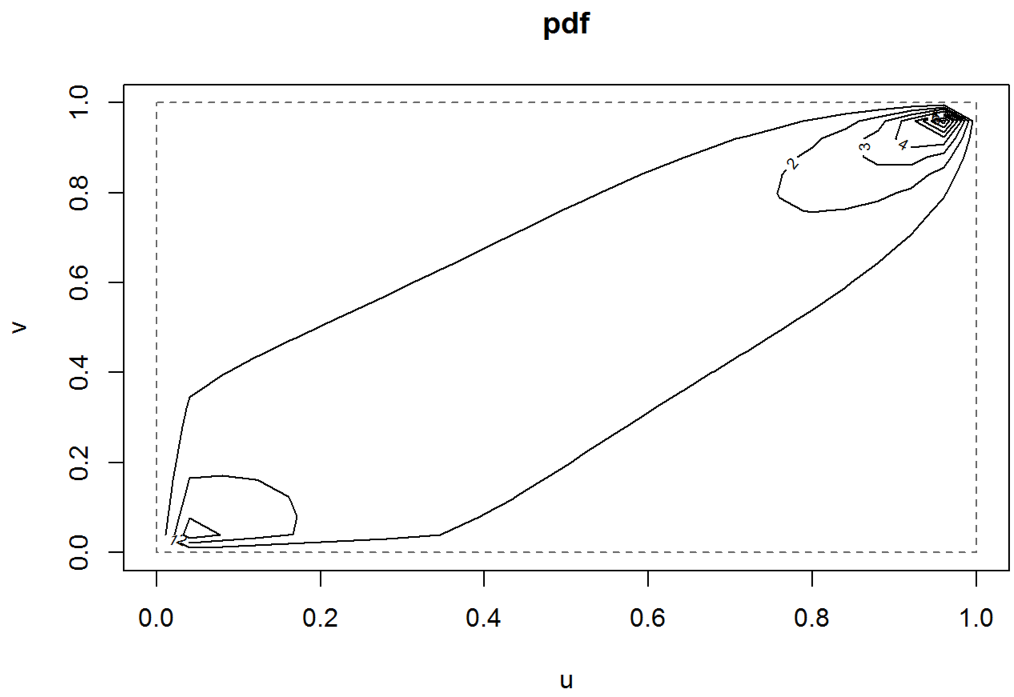
```
## 149 0.853885117 0.854618306
## 150 0.125915516 0.335436619
## 151 0.674414150 0.728580537
## 152 0.575326605 0.544839683
## 153 0.492399756 0.761673937
## 154 0.129200037 0.072709677
## 155 0.463383877 0.980597997
## 156 0.558399877 0.574285225
## 157 0.945130219 0.840834077
## 158 0.881137270 0.761086282
## 159 0.486699709 0.821690720
## 160 0.447624655 0.429804159
## 161 0.020861733 0.003650346
## 162 0.364095775 0.519227457
## 163 0.505845012 0.339425551
## 164 0.769899018 0.808811371
## 165 0.904011687 0.924558871
## 166 0.138711509 0.341455797
## 167 0.156830423 0.268411997
## 168 0.522202697 0.506267220
## 169 0.221645707 0.036108822
## 170 0.625334606 0.917612661
## 171 0.266936184 0.317186422
## 172 0.832821183 0.660347274
## 173 0.429464012 0.143629820
## 174 0.605314188 0.649099231
## 175 0.054387878 0.022649230
## 176 0.755646040 0.483061137
## 177 0.695940414 0.696213508
## 178 0.472080113 0.315531163
## 179 0.245444329 0.203595434
## 180 0.047595605 0.366907606
## 181 0.262282368 0.285670622
## 182 0.956975703 0.919597270
## 183 0.588160932 0.673302014
## 184 0.434829350 0.241365353
## 185 0.667142100 0.869916425
## 186 0.847527358 0.809397906
## 187 0.624674068 0.040164227
## 188 0.027760090 0.190357201
## 189 0.688352183 0.431352332
## 190 0.008062198 0.006384882
## 191 0.341602206 0.472928550
## 192 0.277975453 0.483515143
## 193 0.050369720 0.506341692
## 194 0.637908158 0.329920228
## 195 0.936923777 0.922430074
## 196 0.410699085 0.327373201
## 197 0.480166262 0.952745735
## 198 0.380469551 0.602995930
## 199 0.309092956 0.426867719
## 200 0.228672082 0.152666056
## 201 0.645131010 0.161050575
## 202 0.003933024 0.091629976
## 203 0.518840875 0.520477303
## 204 0.472784696 0.081789289
## 205 0.480197294 0.163450804
## 206 0.273492228 0.415314266
## 207 0.569764223 0.060069685
## 208 0.510323457 0.416085480
## 209 0.195807477 0.375886621
## 210 0.197995060 0.114624732
## 211 0.213154338 0.058979585
## 212 0.781245760 0.955899395
## 213 0.650924751 0.693425955
## 214 0.211928103 0.134100289
## 215 0.437054261 0.379083302
## 216 0.520008486 0.247978282
## 217 0.575416121 0.183024244
## 218 0.936352507 0.961271328
## 219 0.413688720 0.189479796
## 220 0.552612441 0.764121181
## 221 0.129377091 0.649275765
```

```
## 222 0.483406409 0.124640055
## 223 0.845764079 0.989894190
## 224 0.257035433 0.191943058
## 225 0.544792866 0.222054567
## 226 0.938533201 0.653926171
## 227 0.284342485 0.471243034
## 228 0.192515105 0.512119067
## 229 0.708638879 0.256203151
## 230 0.644266174 0.691958491
## 231 0.022494432 0.082467557
## 232 0.955267032 0.908984022
## 233 0.583265824 0.508326189
## 234 0.038639628 0.243757367
## 235 0.545428782 0.452583523
## 236 0.211175189 0.098317189
## 237 0.260897780 0.628421656
## 238 0.758170311 0.671253193
## 239 0.455266204 0.401976980
## 240 0.292001213 0.220047480
## 241 0.864043637 0.484360690
## 242 0.610785122 0.127772752
## 243 0.580441478 0.676647074
## 244 0.964204272 0.884034463
## 245 0.485969736 0.661506996
## 246 0.853547470 0.880471408
## 247 0.389515538 0.344212043
## 248 0.160170189 0.126158432
## 249 0.086382786 0.563810495
## 250 0.917273320 0.966115851
```

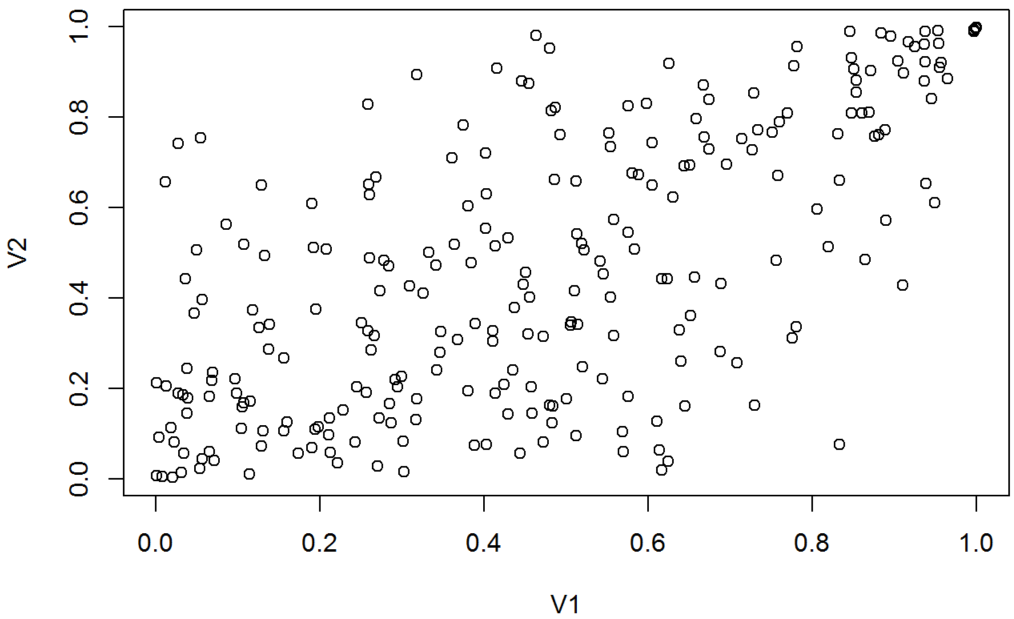
```
persp(gumbelCopula(Copula.Fit@estimate), dCopula, main="pdf",xlab="u", ylab="v", zlab="c(u,v)")
```



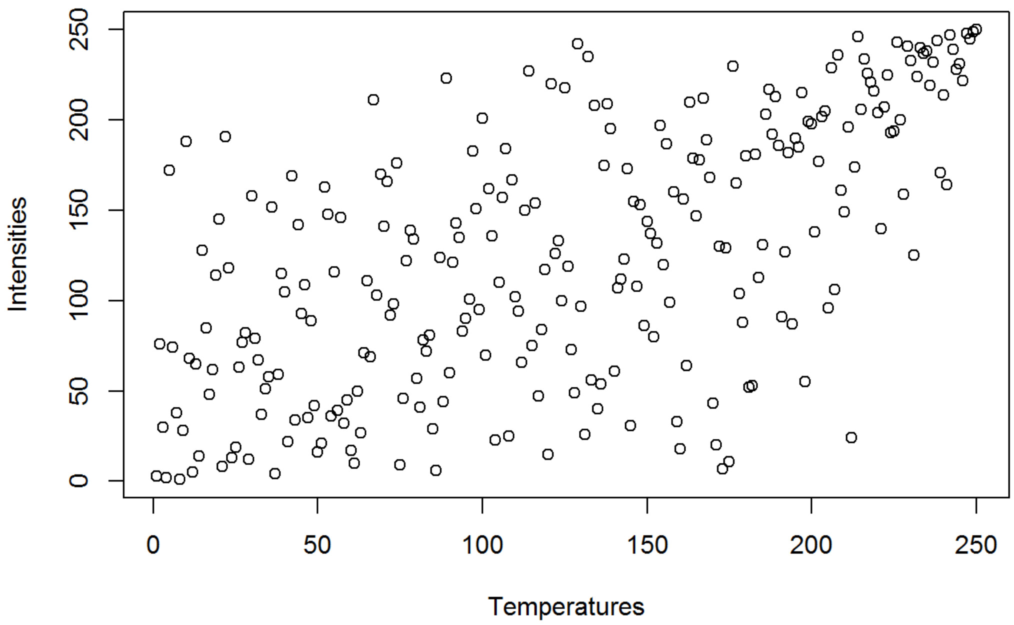

```
contour(gumbelCopula(Copula.Fit@estimate),dCopula, main="pdf",xlab="u", ylab="v")
```



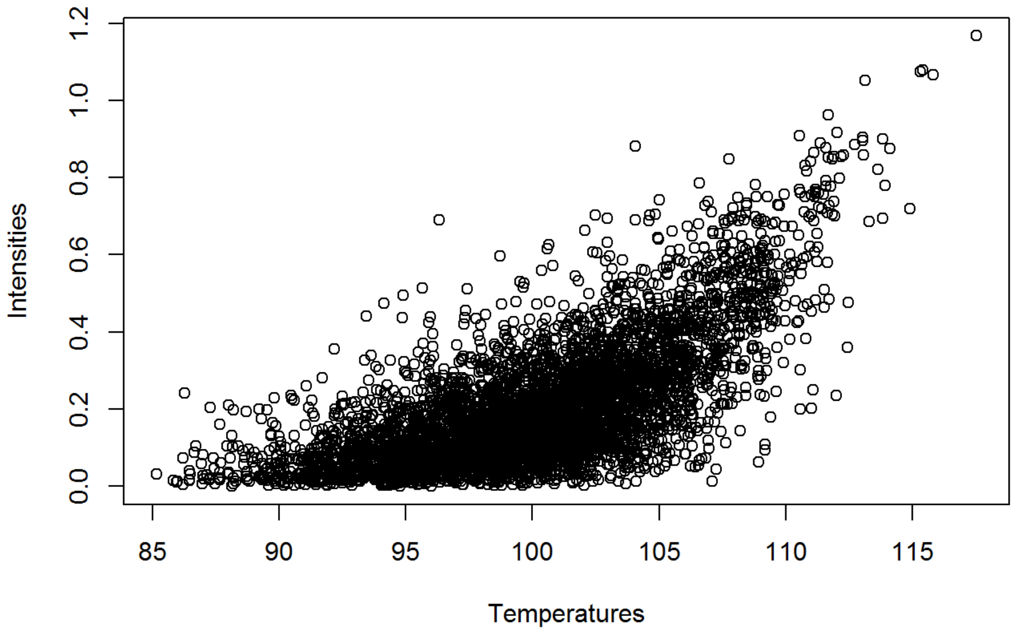

```
plot(Simulated.Copula)
```



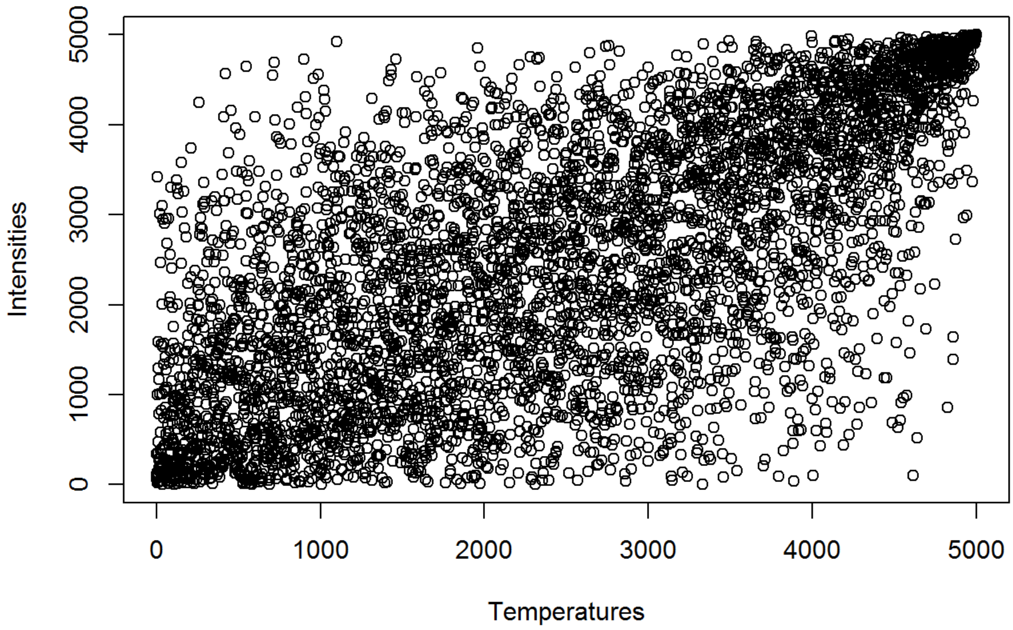

```
df <- data.frame(Temperatures = qnorm(Simulated.Copula[,1],mean=100.069853 ,sd=4.812484),
                 Intensities = qgamma(Simulated.Copula[,2],shape=1.655712, rate=8.132182/60 ))
Emperical.Copula <- as.data.frame(apply(df,2,rank))
plot(Emperical.Copula)
```




```
set.seed(8301735)
Simulated.Copula1 <- as.data.frame(rCopula(copula=gumbelCopula(Copula.Fit@estimate,dim=2), n=5000))
df1 <- data.frame(Temperatures = qnorm(Simulated.Copula1[,1],mean=100.069853 ,sd=4.812484),
                  Intensities = qgamma(Simulated.Copula1[,2],shape=1.655712, rate=8.132182 ))
plot(df1)
```




```
Emperical.Copula1 <- as.data.frame(apply(df1,2,rank))
plot(Emperical.Copula1)
```

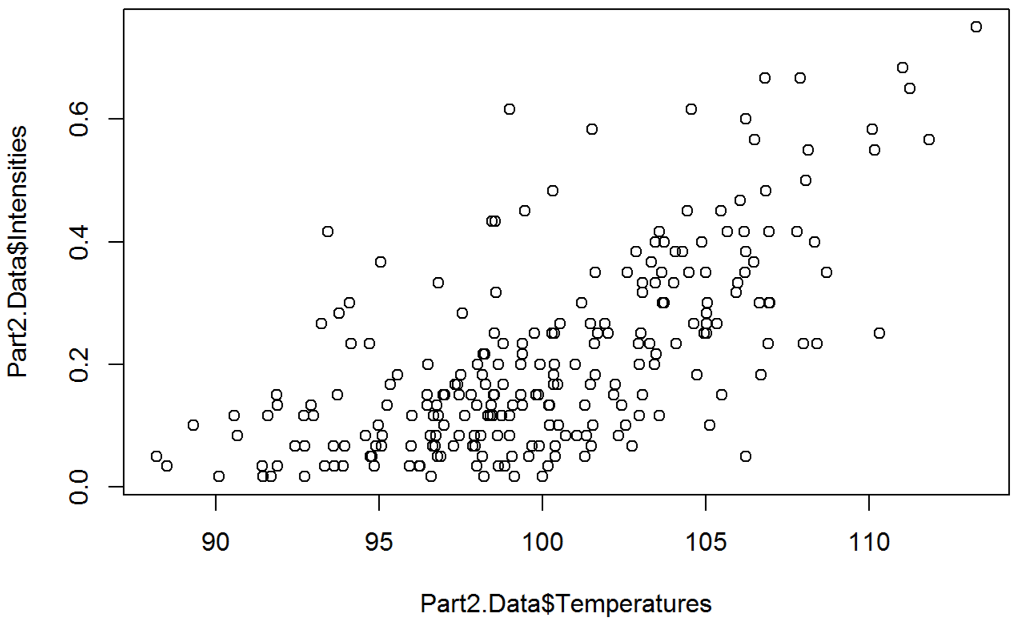


My numbers are slightly diferent than Yuri's below and I'm not exactly sure as to why. Have mercy! haaaha

```
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
```

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



```
NB.Fit.To.Sample <- glm.nb(Counts~Temperatures,Part2.Data)
NB.Fit.To.Sample$coefficients
```

```
##  (Intercept) Temperatures
##  -7.43137538   0.09843241
```

```
NB.Fit.To.Sample$deviance
```

```
## [1] 257.1937
```

```
NB.Fit.To.Sample$df.residual
```

```
## [1] 240
```

```
NB.Fit.To.Sample$aic
```

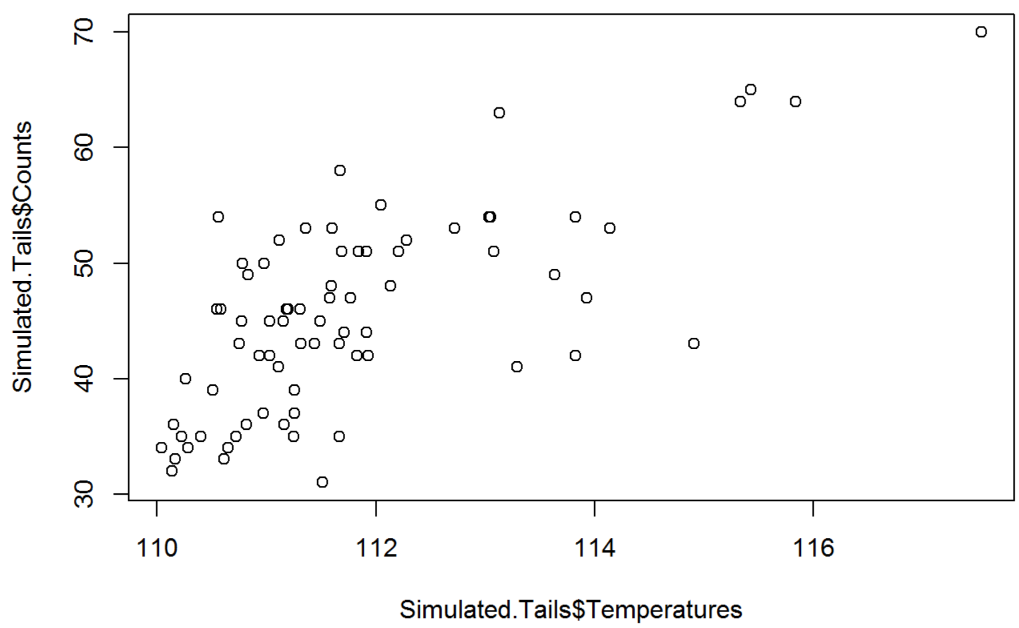
```
## [1] 1557.817
```

```
NB.Fit.To.Sample$theta
```

```
## [1] 4.202612
```

```
Simulated.Temperature <- qnorm(Simulated.Copula1[,1],mean=100.069853 ,sd=4.812484)
Simulated.Intensities <- qgamma(Simulated.Copula1[,2],shape=1.655712, rate=8.132182 )
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)
```



With the Simulated tails, there is a lot less overdispersion which means data towards very high temperatures has a high correlation with the Intensity.

```
NB.Simulated.Tails <- glm.nb(Counts~Temperatures,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
```

```
NB.Simulated.Tails$theta
```

```
## [1] 737219.5
```

Is there an alternative model that you would try to fit to the simulated tail data? I would also try fitting a poisson model which we will fit below What do both models tell you about the relationships between the temperature and the counts? They don't fit well because of the high theta, but the models do a moderatley good job of fitting the data.

```
Poisson.Fit <- glm(Counts~Temperatures,Simulated.Tails,family="poisson")
Poisson.Fit$deviance
```

```
## [1] 62.72328
```

```
summary(Poisson.Fit)$df
```

```
## [1]  2 73  2
```

```
Poisson.Fit$aic
```

```
## [1] 490.0239
```

```
dispersiontest(Poisson.Fit)
```

```
##
##  Overdispersion test
##
## data:  Poisson.Fit
## z = -1.43, p-value = 0.9236
## alternative hypothesis: true dispersion is greater than 1
## sample estimates:
## dispersion
##  0.8305643
```