Input Data Analysis and Distribution Fitting with R

Lidia Montero

September 2016

1 Introduction to (Univariate) Distribution Fitting

I generate a sequence of 5000 numbers distributed following a Weibull distribution with:

1. c=location=10 (shift from origin),
2. b=scale = 2 and
3. a=shape = 1

```
sample<- rweibull(5000, shape=1, scale = 2) + 10
```

The Weibull distribution with shape parameter a and scale parameter b has density given by

f(x) = (a/b) (x/b)^(a-1) exp(- (x/b)^a) for x > 0

The cumulative distribution function is *F(x) = 1 - exp(- (x/b)^a) on x > 0*.

Theoretical moments for Weibull distributions are:

1. Mean(X)=E(X) = c+b Γ(1 + 1/a)
2. Var(X) = b^2 * (Γ(1 + 2/a) - (Γ(1 + 1/a))^2)

1.1 Summarize data

Don't forget to validate uncorrelated sample data :

- acf() Autocorrelation function is fast and easy in R.
- Use durbinWatsonTest() for an inferential option.
- Calculate central and plain moments (up to order 4) using method all.moments() in library(moments)

```
summary(sample)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   10.59   11.41   12.03   12.83   25.92
```

```
library(moments)
all.moments(sample,4,central=F)
```

```
## [1]     1.00000    12.03262   148.89875  1906.48619 25430.78743
```

```
all.moments(sample,4,central=T)
```

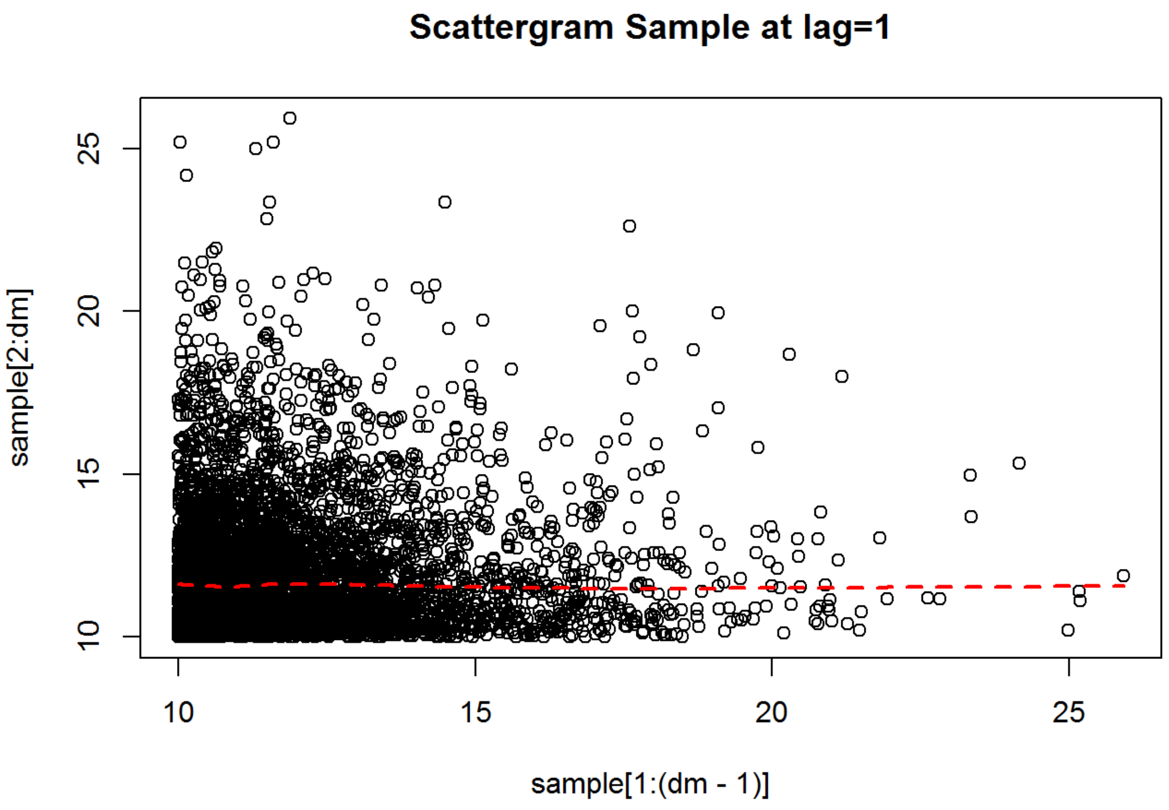
```
## [1] 1.000000e+00 7.023715e-16 4.114827e+00 1.581992e+01 1.324174e+02
```

```
dm=5000 #sample size
```

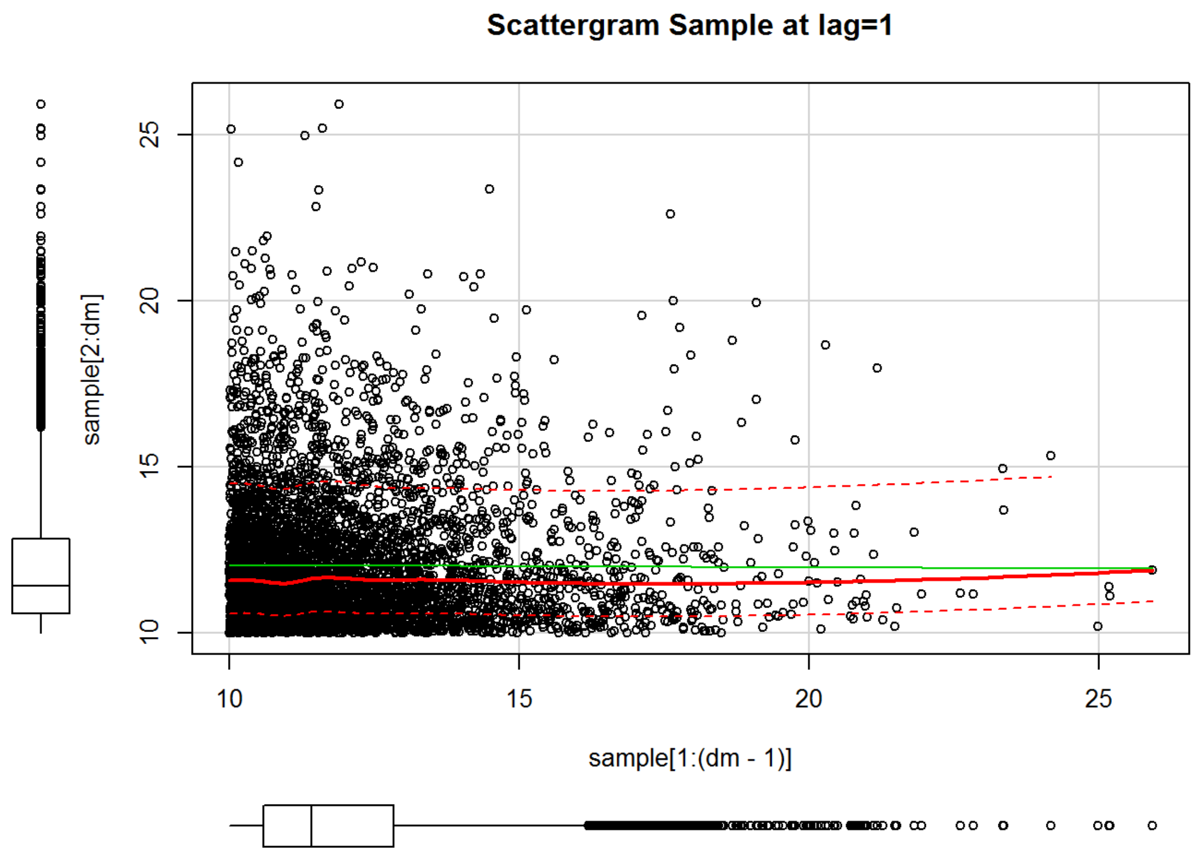
1.2 Autocorrelation Function

- An scattergram for data(1:(m-1)) vs data(2:m) is also valid and check for a flat smoother
- Default scatterplot() in library(car) contains linear adjustment and smoothers directly
- acf() is easy to interpret

```
library(car)
plot(sample[1:(dm-1)],sample[2:dm],main="Scattergram Sample at lag=1")
lines(lowess(sample[1:(dm-1)],sample[2:dm],f=0.5),col=2,lwd=2,lty=2)  # Ov
erlap smoother
```



```
scatterplot(sample[1:(dm-1)],sample[2:dm],main="Scattergram Sample at lag=
1")
```



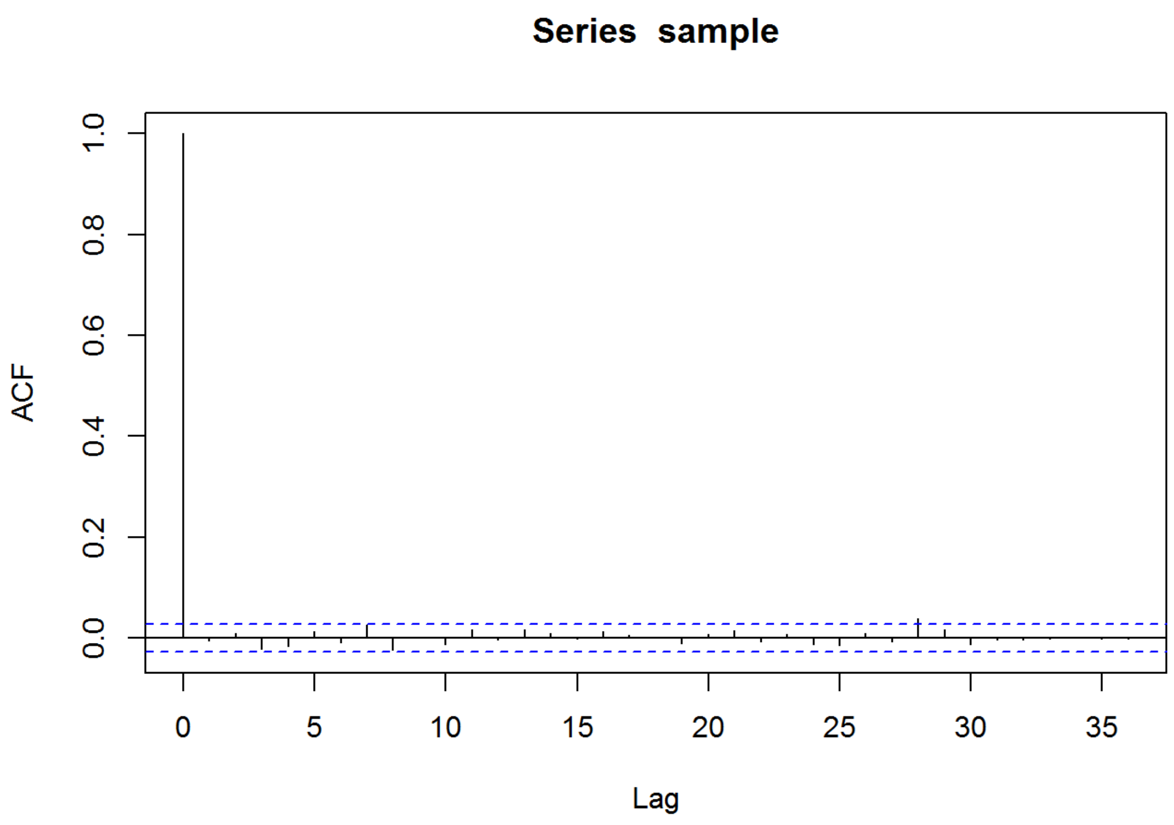
```
library(lmtest)
```

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

```
dwtest(sample~1)  # OK for residuals in library car and lmtest
```

```
##
##  Durbin-Watson test
##
## data:  sample ~ 1
## DW = 2.0122, p-value = 0.6674
## alternative hypothesis: true autocorrelation is greater than 0
```

```
## Also ?durbinWatsonTest  # in library car
acf(sample)
```



2 Plot data

2.1 Histogram: Equal length intervals

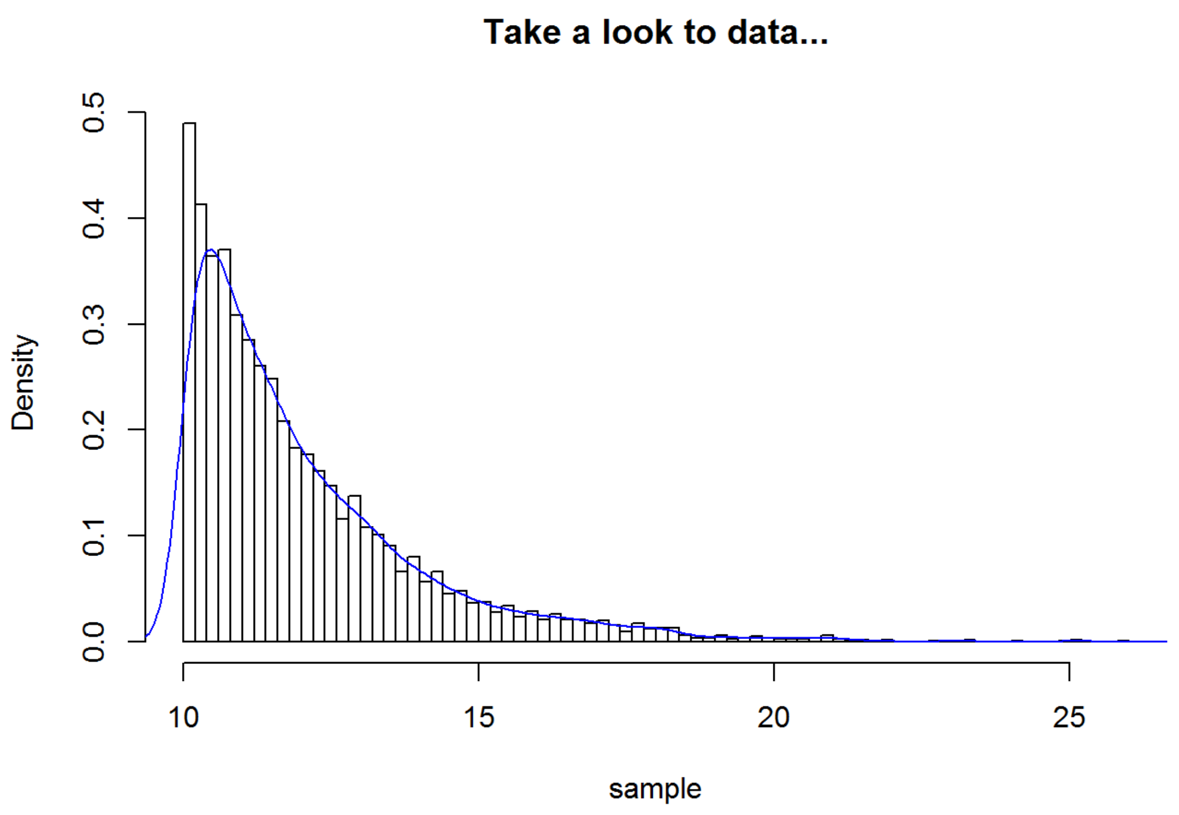
Non suitable for distribution fitting Chi-squared Test

```
hist(sample,freq=F,breaks=100,main="Take a look to data...")
m=mean(sample);std=sd(sample);m;std
```

```
## [1] 12.03262
```

```
## [1] 2.028707
```

```
lines(density(sample),col="blue")
```



Overlap some candidate distributions to fit data: normal (unlikely) and exponential (defined by rate parameter)

The exponential distribution with rate λ and location c has density *f(x)* = $\lambda * exp(-\lambda(x-c))$ for x > c. The exponential cumulative distribution function with rate λ and location c is *F(x) = 1 - exp(-*λ*(x-c))* on x > c.

Theoretical moments for exponential distributions are:

1. Mean(X)=E(X) = c+1/λ
2. Var(X) = (1/λ)^2

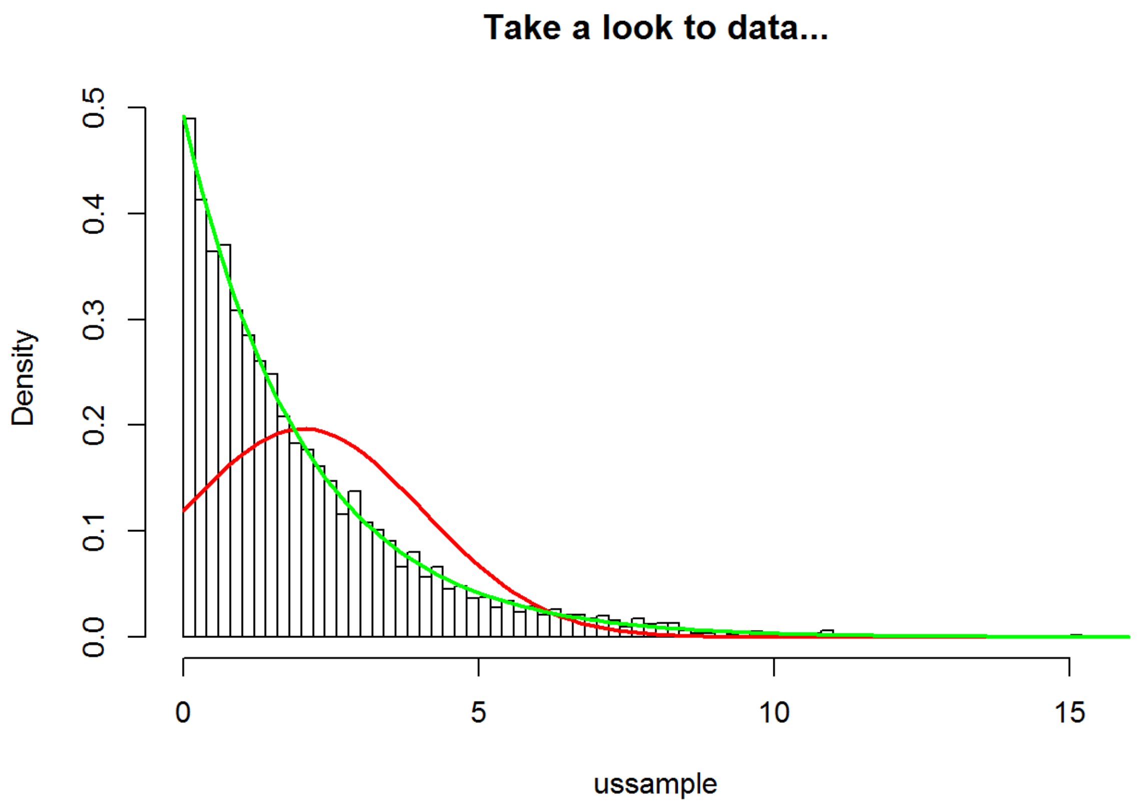
Location parameter c has to be estimated externally: for example, using the minimum, and for overlaped distributions should consider non-shifted distribution candidates.

```
#ussample<-sample-min(sample)+epsilon
ussample<-sample-10
hist(ussample,freq=F,breaks=100,main="Take a look to data...")
m=mean(ussample);std=sd(ussample);m;std
```

```
## [1] 2.032619
```

```
## [1] 2.028707
```

```
curve(dnorm(x,m,std),col="red",lwd=2,add=T)
curve(dexp(x,rate=1/m),col="green",lwd=2,add=T)
```



```
all.moments(ussample,4,central=F)
```

```
## [1]   1.000000   2.032619   8.246368  49.309399 380.113990
```

```
all.moments(ussample,4,central=T)
```

```
## [1]  1.000000e+00 -1.920242e-16  4.114827e+00  1.581992e+01  1.324174e+
02
```

3 List of Candidate distributions

Formulate the list of candidate distributions: for distributions with shape parameter, plot the distribution for several shape parameters, using massive R plot, as the ones suggested in the following example, that takes a gamma distribution as possible candidate,

3.1 Plot pdf's for variable location = 0, scala =1 for several shape parameter values

```
par( mfrow = c( 3,2 ) )
m.shape1 <-  rgamma(5000, shape=1, scale = 1)
mean( m.shape1);sd( m.shape1);var( m.shape1)
```

```
## [1] 0.9884699
```

```
## [1] 0.9980591
```

```
## [1] 0.996122
```

```
hist( m.shape1, freq=F )
x<-seq(0,max(m.shape1),100)
curve(dgamma(x,shape=1,scale=1),col=2,add=T)

m.shape0.5 <-  rgamma(5000, shape=0.5, scale = 1)
mean( m.shape0.5);var( m.shape0.5)^0.5;var( m.shape0.5)
```

```
## [1] 0.4800706
```

```
## [1] 0.6602639
```

```
## [1] 0.4359484
```

```
hist( m.shape0.5, freq=F )
x<-seq(0,max(m.shape0.5),100)
curve(dgamma(x,shape=0.5,scale=1),col=2,add=T)

m.shape2<-  rgamma(5000, shape=2, scale = 1)
mean( m.shape2);var( m.shape2)^0.5;var( m.shape2)
```

```
## [1] 1.965469
```

```
## [1] 1.385176
```

```
## [1] 1.918712
```

```
hist( m.shape2, freq=F )
x<-seq(0,max(m.shape2),100)
curve(dgamma(x,shape=2,scale=1),col=2,add=T)

m.shape3<-  rgamma(5000, shape=3, scale = 1)
mean( m.shape3);var( m.shape3)^0.5;var( m.shape3)
```

```
## [1] 2.994269
```

```
## [1] 1.716285
```

```
## [1] 2.945635
```

```
hist( m.shape3, freq=F )
x<-seq(0,max(m.shape3),100)
curve(dgamma(x,shape=3,scale=1),col=2,add=T)

m.shape5<-  rgamma(5000, shape=5, scale = 1)
mean( m.shape5);var( m.shape5)^0.5;var( m.shape5)
```

```
## [1] 4.976568
```

```
## [1] 2.251865
```

```
## [1] 5.070896
```

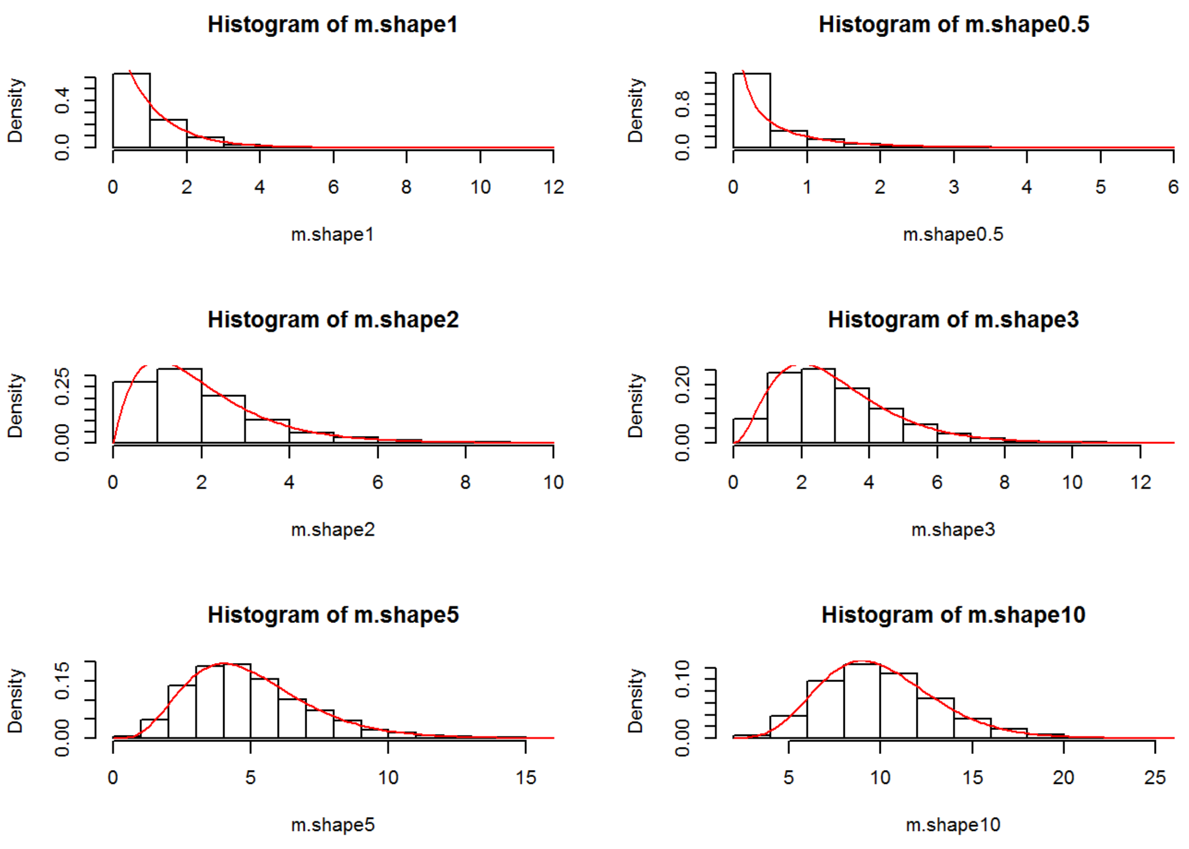
```
hist( m.shape5, freq=F )
x<-seq(0,max(m.shape5),100)
curve(dgamma(x,shape=5,scale=1),col=2,add=T)

m.shape10 <-  rgamma(5000, shape=10, scale = 1)
mean( m.shape10);var( m.shape10)^0.5;var( m.shape10)
```

```
## [1] 10.04493
```

```
## [1] 3.192368
```

```
## [1] 10.19121
```

```
hist( m.shape10, freq=F )
x<-seq(0,max(m.shape10),100)
curve(dgamma(x,shape=10,scale=1),col=2,add=T)
```

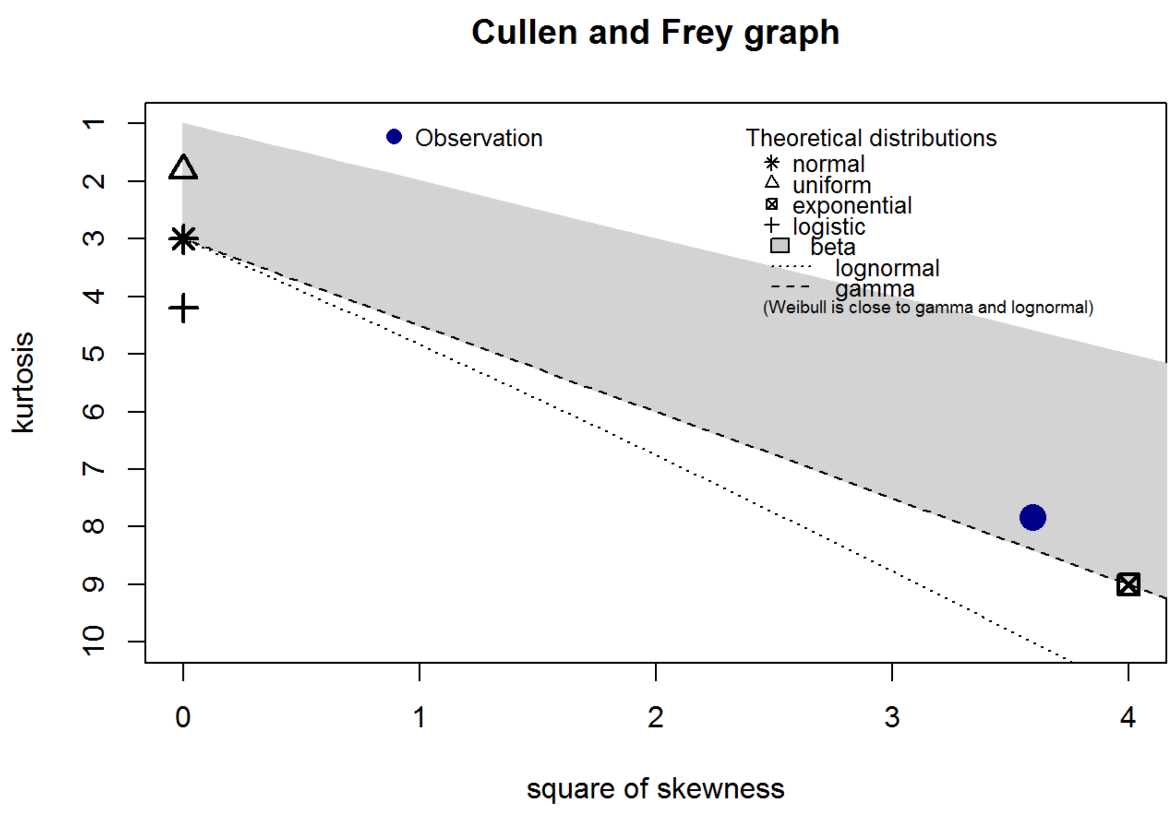


3.2 Method descdist() in fitdistplus package

```
library(fitdistrplus)
```

```
## Loading required package: MASS
```

```
descdist(ussample)
```



```
## summary statistics
## ------
## min:  0.0003954973   max:  15.915
## median:  1.407226
## mean:  2.032619
## estimated sd:  2.028707
## estimated skewness:  1.895864
## estimated kurtosis:  7.826658
```

4 Fit distribution

To fit: use fitdistr() method in MASS package. Pay attention to supported distributions and how to refer to them (the name given by the method) and parameter names and meaning.

For discrete data use goodfit() method in vcd package: estimates and goodness of fit provided together

```
library(MASS)

res<-fitdistr(ussample,"weibull",lower=0.001)
res$estimate
```

```
##     shape     scale
## 0.9973711 2.0303540
```

```
# ?fitdistr   Check it

library(vcd) # For discrete data
```

```
## Loading required package: grid
```

```
# gf <- goodfit(d27x, type = "binomial", par = list(prob = 0.8, size = 7))
# summary(gf)
# plot(gf)

#Method fitdist() in fitdistplus package
library(fitdistrplus)
fitg<-fitdist(ussample,"gamma")
fitw<-fitdist(ussample,"weibull")
```

5 Goodness of Fit

5.1 Histogram: Variable length intervals

- Histogram with breaks defined using quartiles of theoretical candidate distributions.
- Non Equal length intervals defined by empirical quartiles are more suitable for distribution fitting Chi-squared Test, since degrees of freedoms for Chi-squared Tests are guaranteed.

```
sequence<-seq(0,1,by=0.02)
qualist<-quantile(ussample,sequence)
sequence;qualist
```

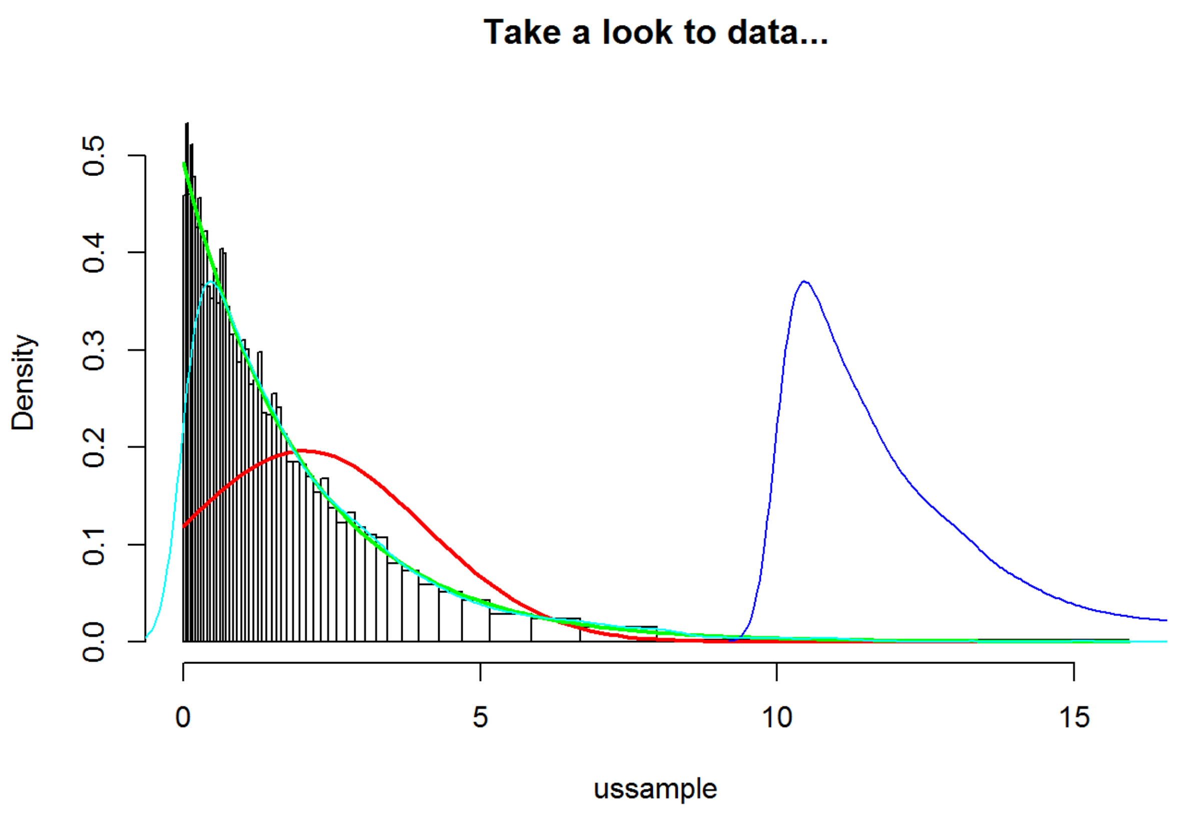
```
##  [1] 0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22 0.24
0.26
## [15] 0.28 0.30 0.32 0.34 0.36 0.38 0.40 0.42 0.44 0.46 0.48 0.50 0.52
0.54
## [29] 0.56 0.58 0.60 0.62 0.64 0.66 0.68 0.70 0.72 0.74 0.76 0.78 0.80
0.82
## [43] 0.84 0.86 0.88 0.90 0.92 0.94 0.96 0.98 1.00
##           0%           2%           4%           6%           8% 
## 3.954973e-04 4.394407e-02 8.151098e-02 1.250477e-01 1.641681e-01 
##          10%          12%          14%          16%          18% 
## 2.060015e-01 2.529657e-01 2.968658e-01 3.513416e-01 3.987952e-01 
##          20%          22%          24%          26%          28% 
## 4.536233e-01 5.102732e-01 5.624885e-01 6.199457e-01 6.694348e-01 
##          30%          32%          34%          36%          38% 
## 7.195240e-01 7.775978e-01 8.409661e-01 9.041210e-01 9.738403e-01 
##          40%          42%          44%          46%          48% 
## 1.038386e+00 1.104926e+00 1.180532e+00 1.254849e+00 1.322035e+00 
##          50%          52%          54%          56%          58% 
## 1.407226e+00 1.492833e+00 1.571155e+00 1.654192e+00 1.747934e+00 
##          60%          62%          64%          66%          68% 
## 1.855825e+00 1.964084e+00 2.073619e+00 2.191610e+00 2.321625e+00 
##          70%          72%          74%          76%          78% 
## 2.440747e+00 2.585901e+00 2.749210e+00 2.899563e+00 3.070045e+00 
##          80%          82%          84%          86%          88% 
## 3.250987e+00 3.438255e+00 3.687353e+00 3.961713e+00 4.300690e+00 
##          90%          92%          94%          96%          98% 
## 4.691573e+00 5.162643e+00 5.862118e+00 6.690737e+00 7.969253e+00 
##         100% 
## 1.591500e+01
```

```
hist(ussample,freq=F,breaks=qualist,main="Take a look to data...")
m=mean(ussample);std=sd(ussample);m;std
```

```
## [1] 2.032619
```

```
## [1] 2.028707
```

```
curve(dnorm(x,m,std),col="red",lwd=2,add=T)
curve(dexp(x,rate=1/m),col="green",lwd=2,add=T)
lines(density(sample),col="blue")
lines(density(ussample),col="cyan")
```



## Method fitdist() in fitdistplus package

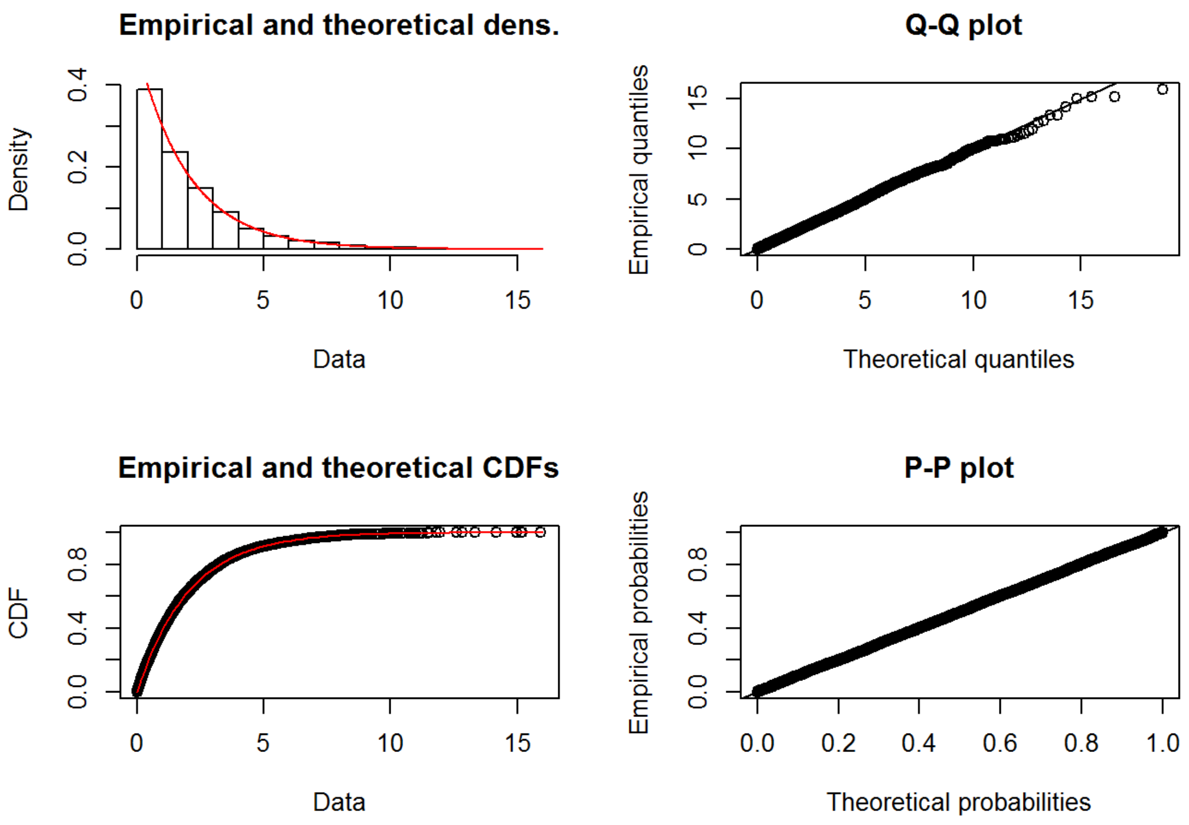
```
library(fitdistrplus)
# descdist(ussample)
fitg<-fitdist(ussample,"gamma")
fitw<-fitdist(ussample,"weibull")

summary(fitg)
```

```
## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters :
##        estimate Std. Error
## shape 0.9950789 0.01751549
## rate  0.4895212 0.01106380
## Loglikelihood:  -8546.587   AIC:  17097.17   BIC:  17110.21
## Correlation matrix:
##           shape      rate
## shape 1.0000000 0.7788053
## rate  0.7788053 1.0000000
```

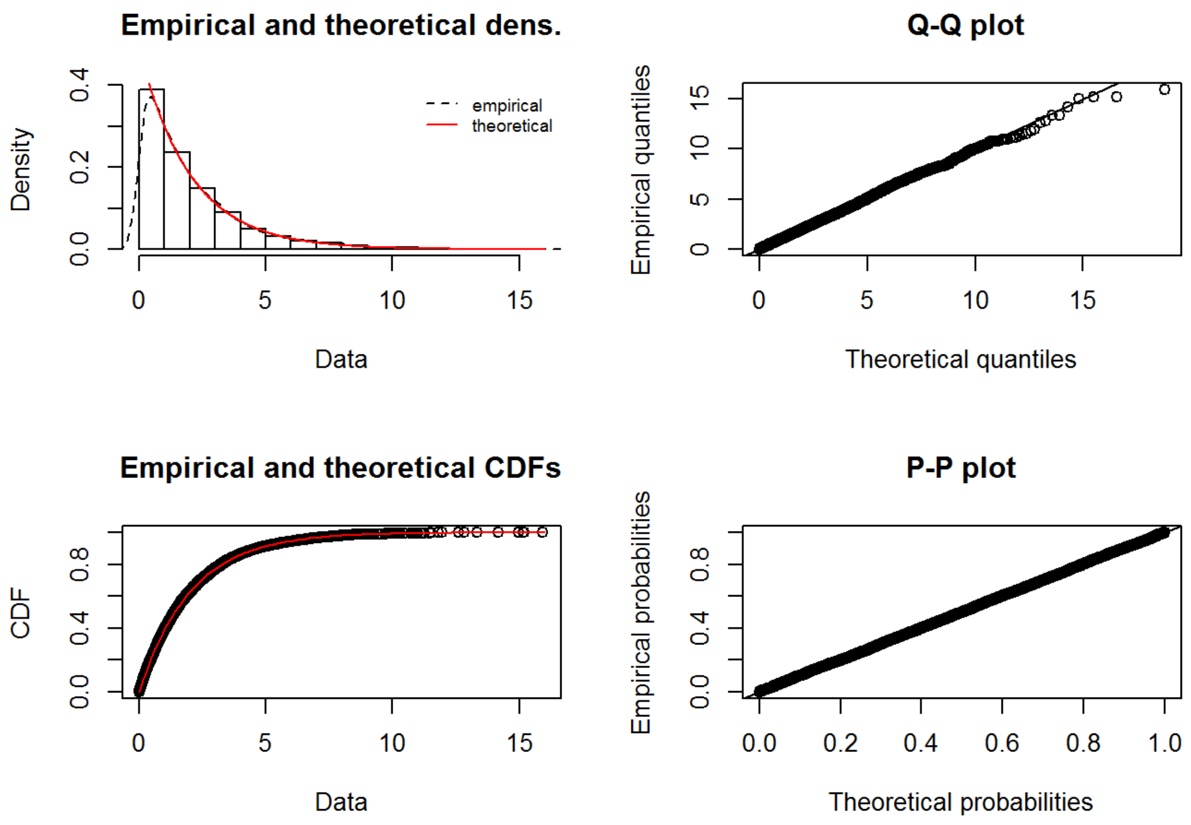
```
summary(fitw)
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters :
##        estimate Std. Error
## shape 0.9973664 0.01101032
## scale 2.0301398 0.03030799
## Loglikelihood:  -8546.598   AIC:  17097.2   BIC:  17110.23
## Correlation matrix:
##           shape     scale
## shape 1.0000000 0.3131775
## scale 0.3131775 1.0000000
```

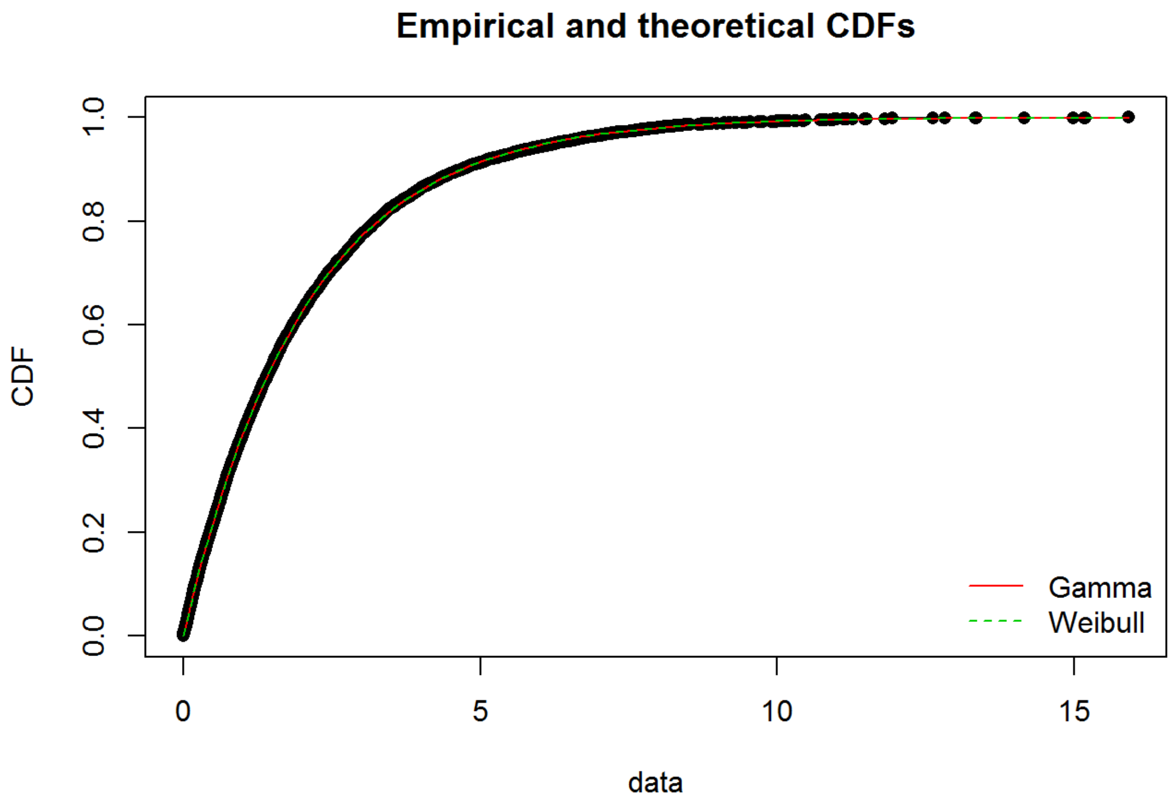
```
plot(fitg)
```



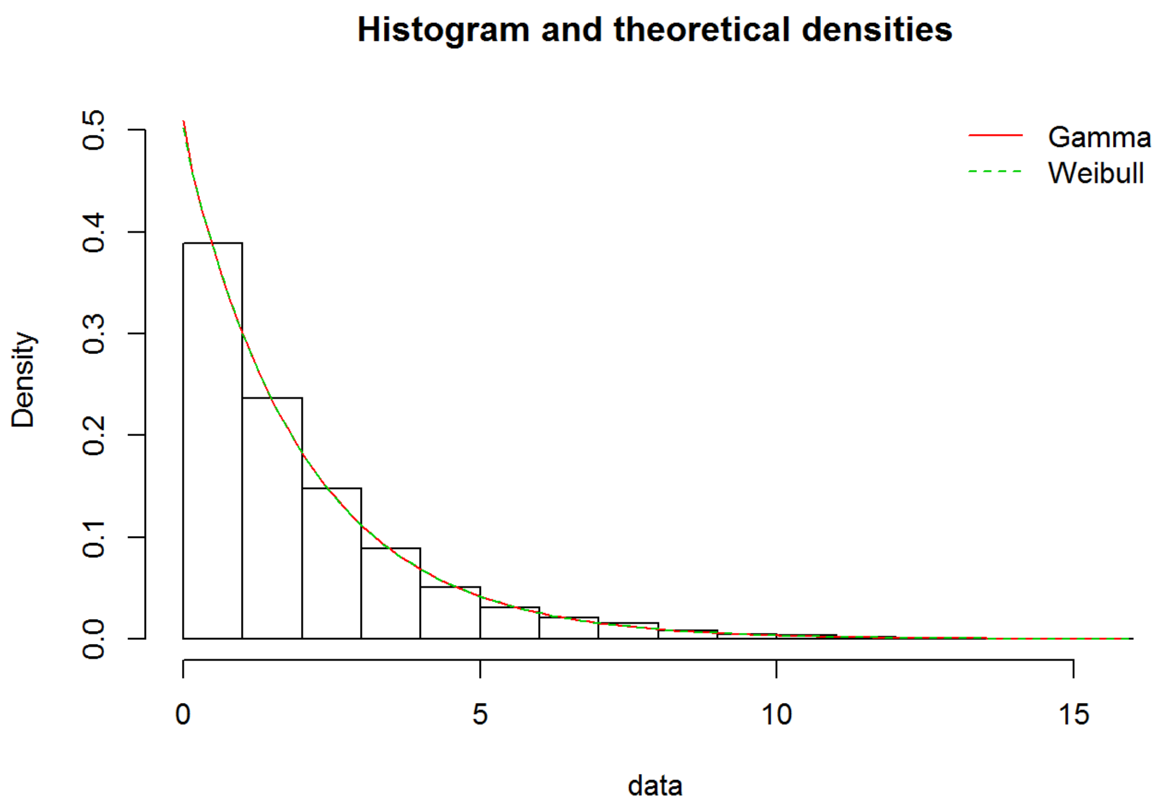
```
plot(fitg, demp = TRUE)
```



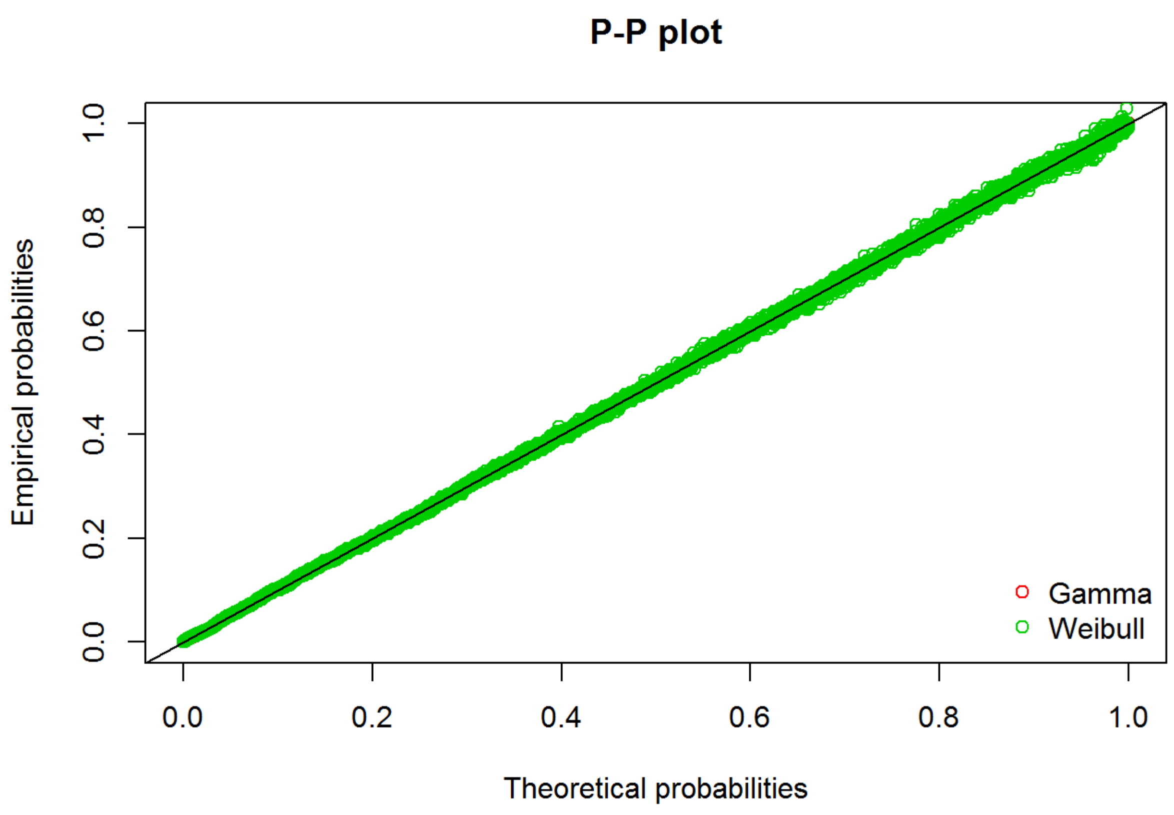
```
cdfcomp(list(fitg,fitw), legendtext=c("Gamma","Weibull"))
```



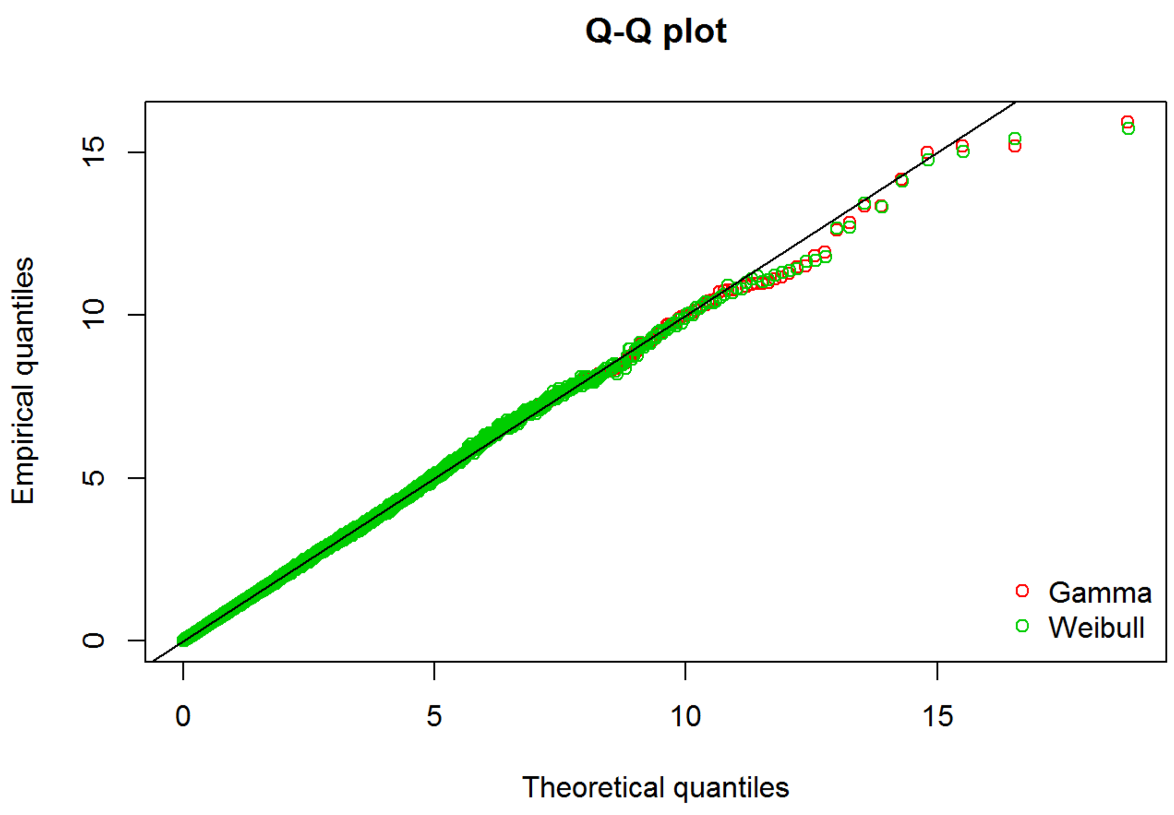
```
denscomp(list(fitg,fitw), legendtext=c("Gamma","Weibull"))
```



```
ppcomp(list(fitg,fitw), legendtext=c("Gamma","Weibull"))
```



```
qqcomp(list(fitg,fitw), legendtext=c("Gamma","Weibull"))
```



```
gofstat(list(fitg,fitw), fitnames=c("Gamma","Weibull"))
```

```
## Goodness-of-fit statistics
##                                    Gamma     Weibull
## Kolmogorov-Smirnov statistic 0.005389694 0.005259483
## Cramer-von Mises statistic   0.016380448 0.016222121
## Anderson-Darling statistic   0.176133193 0.175976428
##
## Goodness-of-fit criteria
##                                  Gamma  Weibull
## Aikake's Information Criterion 17097.17 17097.20
## Bayesian Information Criterion 17110.21 17110.23
```

5.2 Validate matching between empirical moments and theoretical moments

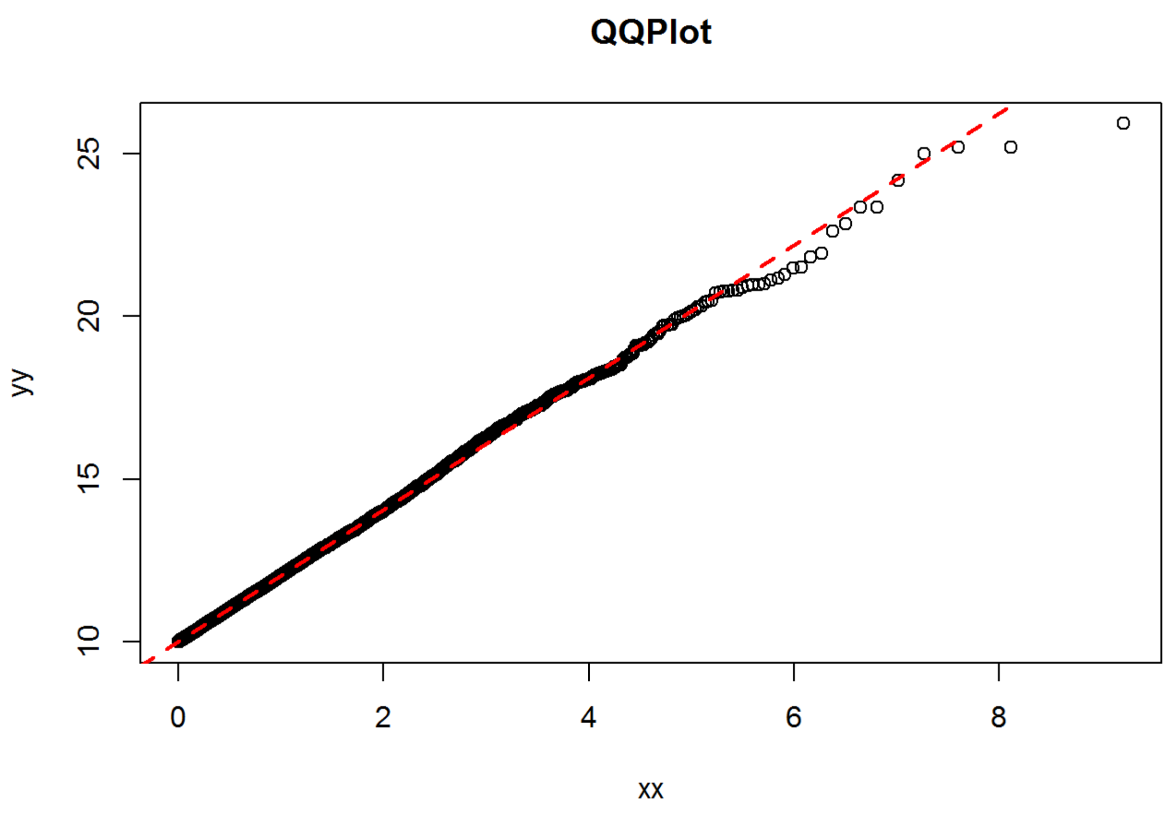
For each candidate distributions calculate up to degree 4 theoretical moments and check central and absolute empirical moments.Previously, you have to estimate parameters and calculate theoretical moments, using estimated parameters. Good matching should exists for any of the candidate distributions between theoretical and empirical moments.

5.3 Goodness of Fit: Graphical Assessment

5.3.1 Known shape

Location and scale parameter estimates are returned as coefficient of linear regression in QQPlot. Check versus fitdistr estimates for distribution parameters

```
xx <- qweibull( ppoints( sample ) , shape=1)
yy<- sort( sample )
mm <- lm( yy ~ xx )
plot(xx,yy,main="QQPlot")
abline(mm,col="red", lwd=2, lty=2)
```



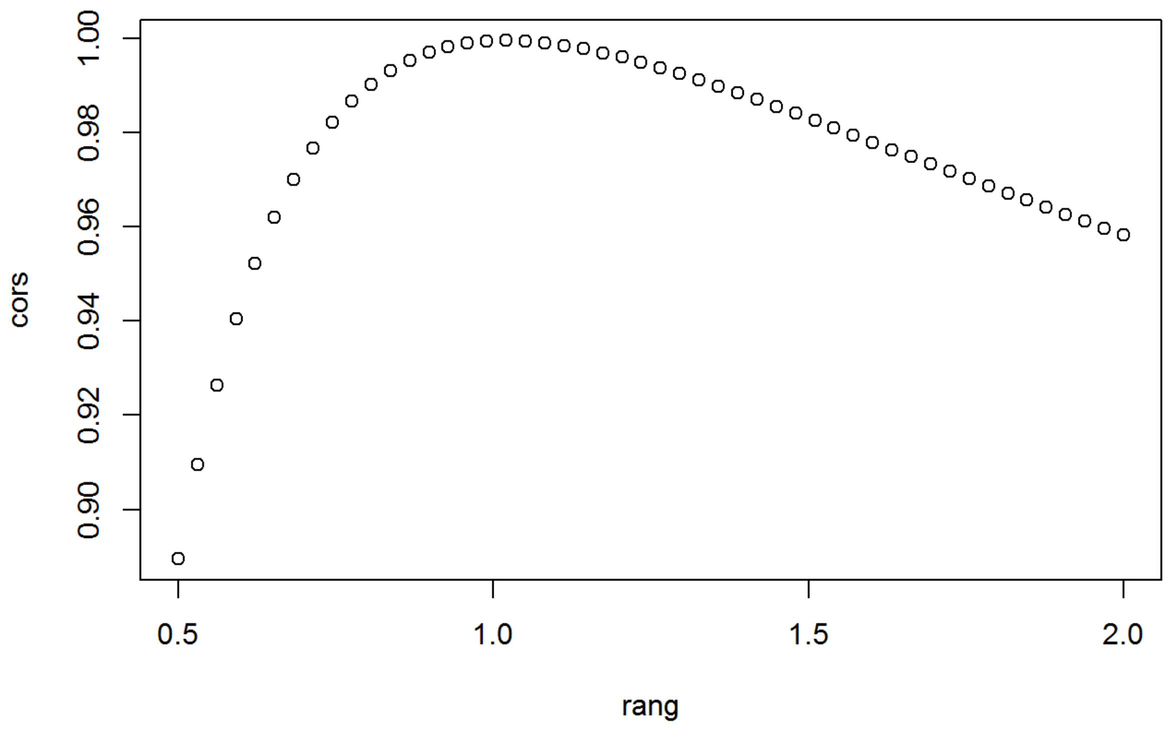
```
cors<-cor(yy,xx);cors
```

```
## [1] 0.9994532
```

5.3.2 UNKnown shape

Use standarized distributions - Identifies shape giving the best fit (alternative to ML estimation). As a subproduct location and scale parameters are also estimated, so you do not need to unshift your data.

```
rang <- seq(0.5,2,length=50)
cors<-rep(0,50)

for (i in 1:50)
{
xx <- qweibull( ppoints( sample ) , shape=rang[ i ] )
yy<- sort( sample )
mm <- lm( yy ~ xx )
cors[ i ]<-cor(yy,xx);cors
}
plot(rang,cors)
```



```
# Goodness of Fit: Histogram needs estimates for parameters
sequence<-seq(0,1,by=0.02)
qualist<-qweibull(shape=res$estimate[1],scale=res$estimate[2],sequence)
sequence;qualist
```

```
##  [1] 0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22 0.24
0.26
## [15] 0.28 0.30 0.32 0.34 0.36 0.38 0.40 0.42 0.44 0.46 0.48 0.50 0.52
0.54
## [29] 0.56 0.58 0.60 0.62 0.64 0.66 0.68 0.70 0.72 0.74 0.76 0.78 0.80
0.82
## [43] 0.84 0.86 0.88 0.90 0.92 0.94 0.96 0.98 1.00
```

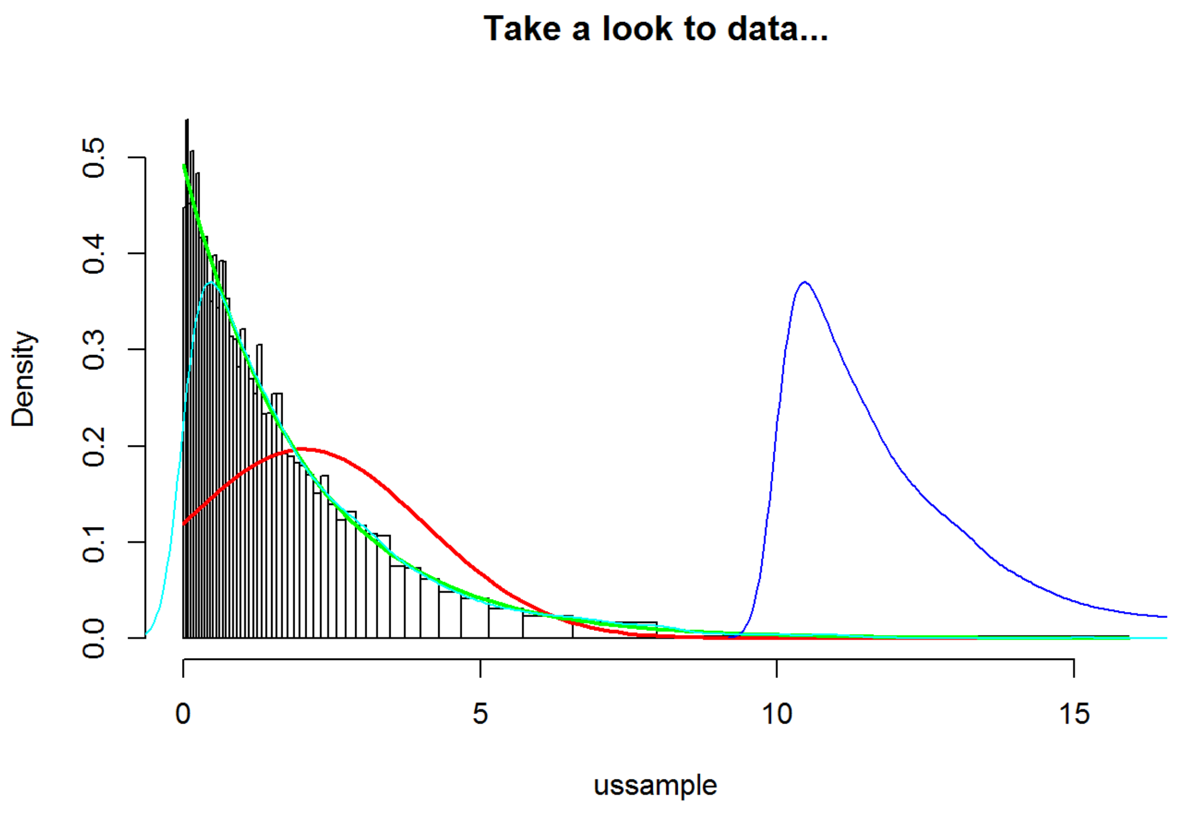
```
##  [1] 0.00000000 0.04059894 0.08218727 0.12471091 0.16818922 0.21265402
##  [7] 0.25814354 0.30470080 0.35237305 0.40121176 0.45127272 0.50261636
## [13] 0.55530810 0.60941881 0.66502535 0.72221122 0.78106732 0.84169277
## [19] 0.90419592 0.96869554 1.03532213 1.10421952 1.17554676 1.24948028
## [25] 1.32621656 1.40597523 1.48900288 1.57557761 1.66601471 1.76067348
## [31] 1.85996592 1.96436748 2.07443082 2.19080353 2.31425107 2.44568735
## [37] 2.58621568 2.73718507 2.90026900 3.07757865 3.27183018 3.48660053
## [43] 3.72673446 3.99902496 4.31342174 4.68535172 5.14066239 5.72781537
## [49] 6.55562671 7.97140072        Inf
```

```
qualist[51]<-max(ussample)

hist(ussample,freq=F,breaks=qualist,main="Take a look to data...")
m=mean(ussample);std=sd(ussample);m;std
```

```
## [1] 2.032619
```

```
## [1] 2.028707
```

```
curve(dnorm(x,m,std),col="red",lwd=2,add=T)
curve(dexp(x,rate=1/m),col="green",lwd=2,add=T)
lines(density(sample),col="blue")
lines(density(ussample),col="cyan")
```



5.4 Goodness of Fit: Tests

5.4.1 Kolmogorov-Smirnoff Test for continuous data

Beware of using the proper names in R for distribution parameters

```
#ks.test(ussample,"pweibull")  # Doesn't work
ks.test(ussample,"pweibull", scale=res$estimate[2], shape=res$estimate[1]
)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  ussample
## D = 0.0052914, p-value = 0.999
## alternative hypothesis: two-sided
```

5.4.2 Cramer von Miess Test for discrete data

For discrete data (discrete version of KS Test). library(dgof) includes cvm.test() Cramer von Miess test, discrete version of KS Test.

5.4.3 Chi Squared Test

Chi Squared Test - It requires manual programming using non-constant length intervals (defined by quartiles). Valid for discrete or continuous data.

```
sequence<-seq(0,1,by=0.02)
qualist<-qweibull(shape=res$estimate[1],scale=res$estimate[2],sequence)
sequence;qualist
```

```
##  [1] 0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22 0.24
0.26
## [15] 0.28 0.30 0.32 0.34 0.36 0.38 0.40 0.42 0.44 0.46 0.48 0.50 0.52
0.54
## [29] 0.56 0.58 0.60 0.62 0.64 0.66 0.68 0.70 0.72 0.74 0.76 0.78 0.80
0.82
## [43] 0.84 0.86 0.88 0.90 0.92 0.94 0.96 0.98 1.00
```

```
##  [1] 0.00000000 0.04059894 0.08218727 0.12471091 0.16818922 0.21265402
##  [7] 0.25814354 0.30470080 0.35237305 0.40121176 0.45127272 0.50261636
## [13] 0.55530810 0.60941881 0.66502535 0.72221122 0.78106732 0.84169277
## [19] 0.90419592 0.96869554 1.03532213 1.10421952 1.17554676 1.24948028
## [25] 1.32621656 1.40597523 1.48900288 1.57557761 1.66601471 1.76067348
## [31] 1.85996592 1.96436748 2.07443082 2.19080353 2.31425107 2.44568735
## [37] 2.58621568 2.73718507 2.90026900 3.07757865 3.27183018 3.48660053
## [43] 3.72673446 3.99902496 4.31342174 4.68535172 5.14066239 5.72781537
## [49] 6.55562671 7.97140072        Inf
```

```
qualist[51]<-max(ussample)
dsample<-cut(ussample,breaks=qualist)
iobs<-as.vector(table(dsample))  # Obs in the groups defined by nb interva
ls-percentiles
pexp<-as.vector(rep(1/50,50))    # Expected probability for each groups: s
ample size/nb intervals
iexp<-length(ussample)*pexp
iobs;pexp;iexp
```

```
##  [1]  91 112  96 110  99 110  97  86 102  92  90 105  93 109 112 104  9
5
## [18]  97  91 107 101  96  94 117  93  97 110 115  91  94  95  99  99  9
3
## [35] 111  98  93 107 104 105 114  90  99  96  90  94  92  96 119 100
```

```
##  [1] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
0.02
## [15] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
0.02
## [29] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
0.02
## [43] 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
```

```
##  [1] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 10
0
## [18] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 10
0
## [35] 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100 100
```

```
X2<-sum(((iobs-iexp)^2)/iexp);X2
```

```
## [1] 33.66
```

```
#chisq.test( sample, "pweibull", shape= 2, scale=1 )   #Doesn't work
chisq.test( iobs,p=pexp ) # Works
```

```
##
##  Chi-squared test for given probabilities
##
## data:  iobs
## X-squared = 33.66, df = 49, p-value = 0.9535
```