Normalp

Prof. Massimiliano Giacalone

10/02/2017

library('normalp')  
  
#Let's compute the density for a x value with mu=0, sigmap=2 and p=2  
dnormp(2, p=2)

## [1] 0.05399097

#Let's compute the distribution for a q value with mu=1, sigmap=2 and p=2   
pnormp(0.7, mu=1, sigmap=2, p=2)

## [1] 0.4403823

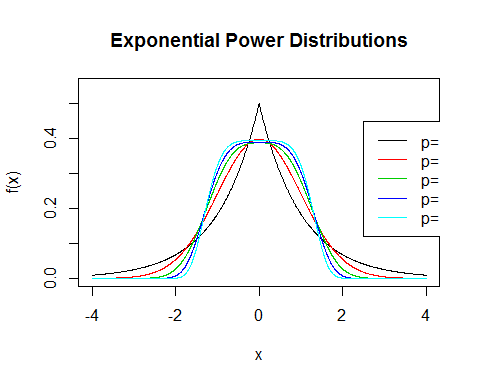
#Let's compute the quantile for a probability value pr=0.4 with mu=3, sigmap=2 and p=2   
qnormp(0.4, mu=3, sigmap=2, p=1.5)

## [1] 2.51149

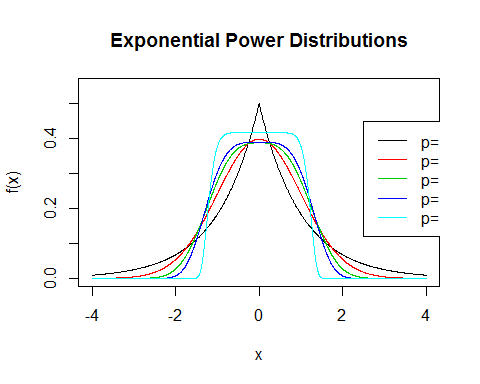
#In the following example, we generate a random sample of size n=30 from an exponential power distribution   
#with mu=2, sigmap=3 and p=1.5  
rnormp(30,2,3,1.5)

## [1] 8.0053821 8.1042743 -2.3606352 5.2568087 1.9550047 2.8596691  
## [7] 1.7795360 3.6183377 1.1902721 6.9011483 4.8606972 3.5129063  
## [13] 3.3935536 1.4716770 6.4289855 1.1359680 -4.6231760 2.3544251  
## [19] -2.8170461 3.9082953 3.3320523 -1.3695038 -2.3745950 8.2578143  
## [25] 6.7712646 2.6541339 -0.3799379 -1.1570932 -2.2119725 -2.2464491

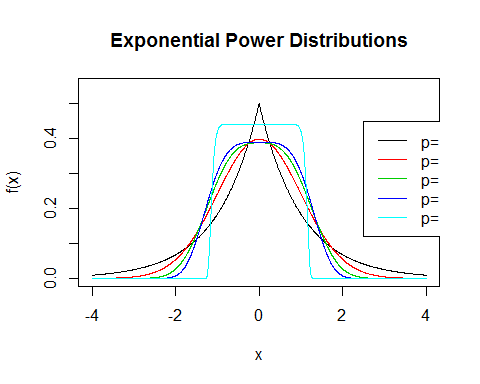
#With the following command, we plot 5 different distributions with p=1, 2, 3, 4, 5, 10, 20, 30, 50 (the last one will be the  
#uniform distribution)   
graphnp(c(1:4,5))



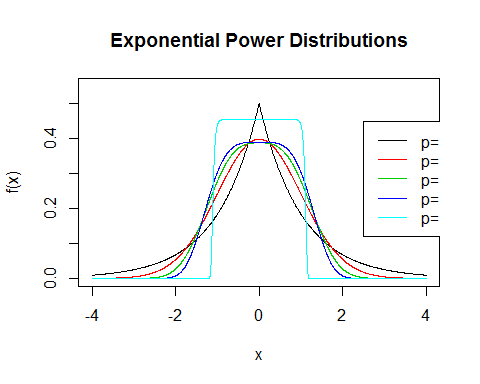
graphnp(c(1:4,10))



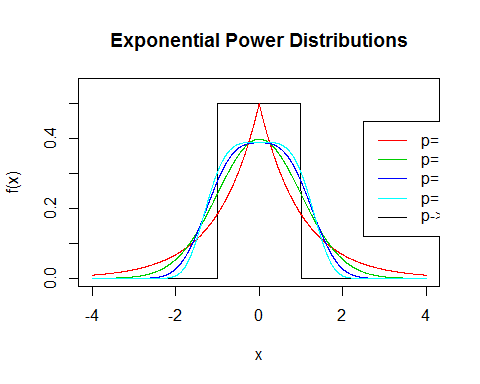
graphnp(c(1:4,20))



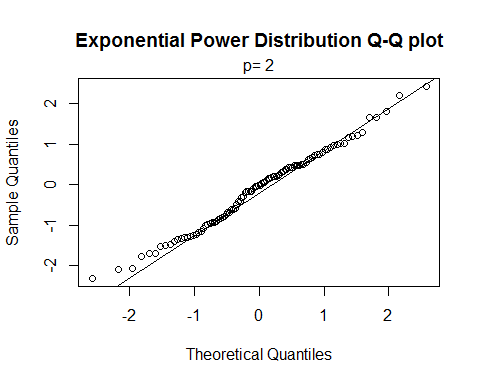
graphnp(c(1:4,30))



graphnp(c(1:4,50))



#and an exponential power distribution Q-Q plot for a sample of size n=100:  
x=rnormp(100, p=2)  
qqnormp(x, p=2)  
qqlinep(x, p=2)



###Estimating the exponential power distribution parameters###  
  
#Let's estimate the shape parameter p from a vector of observations:  
x=rnormp(200,1,2,4)  
p=estimatep(x, 1, 2)  
p

## [1] 3.127864

#Let's estimate {inserire mu, sigmap } and p from a vector of observations:  
x=rnormp(200,1,2,3)  
parameters=paramp(x)  
parameters

## Mean Mp Sd Sp p   
## 1.004110 1.030457 1.794412 2.004770 2.855000   
##   
## no.conv = FALSE

#Let's compute the values of theoretical and empirical indices of kurtosis, with p known or estimated  
kurtosis(p=1.5)

## VI B2 Bp   
## 1.303127 3.761954 2.500000

kurtosis(x, p=1.5, value='parameter')

## VI B2 Bp   
## 1.208993 2.355632 1.852478

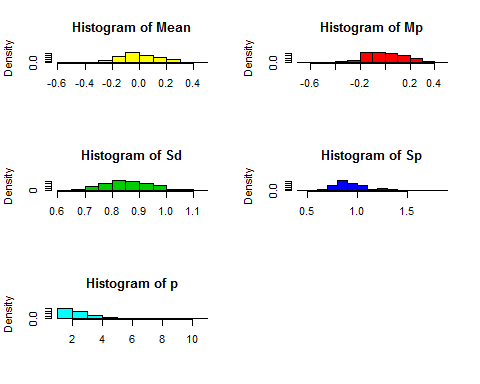
kurtosis(x)

## VI B2 Bp   
## 1.207764 2.345899 3.503474

#Let's perform a simulation plan for m=200 samples of size n=30, with mu=0, sigmap=1, p=3, by showing   
#the histogram for each set of estimates   
sim=simul.mp(30, 200, p=3)  
sim

## Mean Mp Sd Sp p  
## Mean 0.001880339 -0.007492459 0.85687655 0.95915706 3.374070  
## Variance 0.024985574 0.027781130 0.00868146 0.04562682 7.531195  
##   
## Number of samples with a difficult convergence: 10

par(mfrow=c(3,2))  
plot(sim)  
par(mfrow=c(1,1))



###Fitting a linear regression model with random errors distributed according to an exponential power distribution###  
  
#We consider an example that has the only purpose to show the use of the implemented functions to fit a   
#linear regression model when we have to assume that the random errors are distributed according to an exponential  
#power distribution  
  
dataset=read.table('C:/Users/mario/Documents/GIACALONE/NORMALP/movie.txt', header = TRUE)  
dataset

## Gross Videos  
## 1 1.10 57.18  
## 2 1.13 26.17  
## 3 1.18 92.79  
## 4 1.25 61.60  
## 5 1.44 46.50  
## 6 1.53 85.06  
## 7 1.53 103.52  
## 8 1.69 30.88  
## 9 1.74 49.29  
## 10 1.77 24.14  
## 11 2.42 115.31  
## 12 5.34 87.04  
## 13 5.70 128.45  
## 14 6.43 126.64  
## 15 8.59 107.28  
## 16 9.36 190.80  
## 17 9.89 121.57  
## 18 12.66 183.30  
## 19 15.35 204.72  
## 20 17.55 112.47  
## 21 17.91 162.95  
## 22 18.25 109.20  
## 23 23.13 280.79  
## 24 27.62 229.51  
## 25 37.09 277.68  
## 26 40.73 226.73  
## 27 45.55 365.14  
## 28 46.62 218.64  
## 29 57.70 286.31  
## 30 58.51 254.58

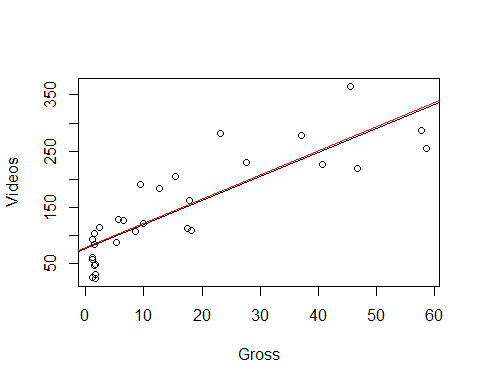
attach(dataset)  
res.lmp=lmp(Videos~Gross)  
res.lmp

##   
## Call:  
## lmp(formula = Videos ~ Gross)  
##   
## Coefficients:  
## (Intercept) Gross   
## 78.235 4.317

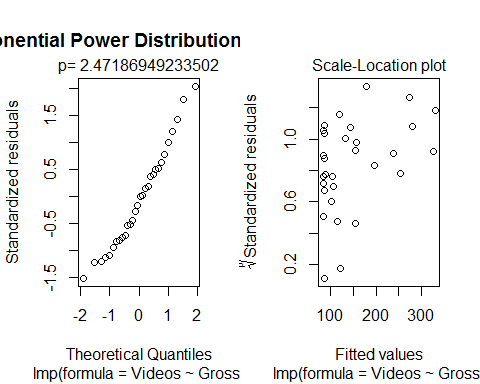
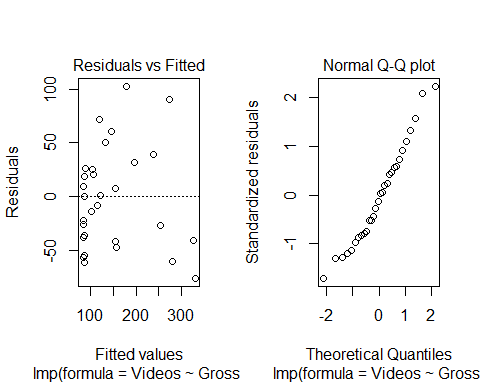
#Let's report a summary of the mian results with a plot showing the 2 straight lines derived by using the   
#OLS methods and the lmp() method:  
  
summary(res.lmp)

##   
## Call:  
## lmp(formula = Videos ~ Gross)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -76.224 -40.236 -3.907 26.374 102.710   
##   
## Coefficients:  
## (Intercept) Gross   
## 78.235 4.317   
##   
## Estimate of p  
## `2.471869`  
##   
## Power deviation of order p: 51.69

plot(Videos~Gross)  
abline(lm(Videos~Gross))  
abline(res.lmp, col='red')



#Let's perform an analysis of residuals:  
par(mfrow=c(1.5,2))  
plot(res.lmp)



#Let's perform a simulation plan for m=200 samples of size n=30 for a linear regression model with only one   
#regressor, by showing the histogram for each set of estimates  
sim=simul.lmp(30,200,1,2,1,1,3)  
sim

## (intercept) x1 Sp p  
## Mean 1.0264802 1.9712919 0.91180743 3.012432  
## Variance 0.1151872 0.3411297 0.03726107 6.326452  
##   
## Number of samples with a difficult convergence: 1

summary(sim)

## Results:  
## (intercept) x1 Sp p  
## Mean 1.0264802 1.9712919 0.91180743 3.012432  
## Variance 0.1151872 0.3411297 0.03726107 6.326452  
##   
## Coefficients:  
## (intercept) x1   
## 1 2   
##   
## Formula:  
## y ~ +x1  
## <environment: 0x00000000073de4b0>  
##   
## Number of samples:  
## `200`  
##   
## Value of p:  
## `3`  
##   
## Number of samples with problems on convergence  
## `1`

par(mfrow=c(1.5,2))  
plot(sim)

