Introduction to R

```
> 1+1
[1] 2
> 2^3 # Two to the power 3
[1] 8

> 1:30
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30

> gamma(.5)^2      # Gamma(1/2) = Sqrt(Pi)
[1] 3.141593

> x <- 1           # Assigns the value 1 to x
> y <- 2
> x+y
[1] 3
> z <- x+y
> z
[1] 3
> x <- c(1,2,3,4,5,6)    #  Collect these numbers; x is now a vector

> z # No dynamic updating; it's not a spreadsheet
[1] 3
> x+y
[1] 3 4 5 6 7 8

> y = 1 + 2*x # Another way to do assignment
> cbind(x,y)
     x  y
[1,] 1  3
[2,] 2  5
[3,] 3  7
[4,] 4  9
[5,] 5 11
[6,] 6 13

> z <- y[x>4]           # z gets y such that x > 4
> z
[1] 11 13

> # If you put an array of integers inside the brackets, you get those
> # elements, in the order indicated.

> y[c(6,5,4,3,2,1)] # y in opposite order
[1] 13 11  9  7  5  3
> y[c(2,2,2,3,4)] # Repeats are okay
[1] 5 5 5 7 9
> y[7] # There is no seventh element.  NA is the missing value code
[1] NA
```

```
> # Random number generation
> # Transform a uniform by inverse CDF
> help(rexp)
```

Exponential {stats} R Documentation

The Exponential Distribution

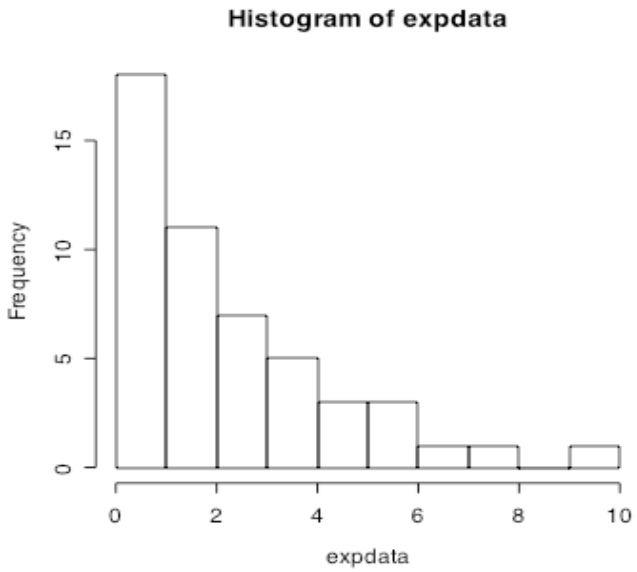
Description

Density, distribution function, quantile function and random generation for the exponential distribution with rate `rate` (i.e., mean `1/rate`).

Usage

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)
```

```
> expdata = rexp(50,rate=1/2) # Random sample from exponential distribution, mean=theta=2
> expdata
 [1] 0.4330015 5.7893762 0.9803759 0.7172530 2.2696433 4.0045302 3.3989651 0.3104736
 [9] 1.2026790 0.8543951 1.0438012 5.5095891 0.7587579 1.9263300 6.0660176 9.3017992
[17] 1.0910204 0.6551285 1.5747176 5.9417700 0.8464761 7.6684436 0.1107589 1.6787699
[25] 2.4744338 3.3470232 0.3209082 4.4307811 4.5510434 1.4316870 0.3457547 0.1302476
[33] 0.5777305 1.0898631 1.4467458 3.2472808 1.8113195 0.5090032 2.4633656 0.8972205
[41] 0.7562905 2.4623634 0.3413955 2.3122374 0.4166320 2.6279765 1.5072294 3.5732947
[49] 3.5449348 2.6472542
> mean(expdata) # The MLE
[1] 2.267962
> hist(expdata)
```



QQ Plots

The histogram looks pretty exponential, but let's try a QQ plot. Q stands for quantile. A quantile is like a percentile, but divided by 100 and applied to a probability distribution rather than a data set. The p-th quantile is the point with p of the distribution below it. So for example the 0.5 quantile is the median. The critical value of a typical F-test is the $1-\alpha$ quantile of the F distribution.

There is more than one kind of QQ plot. We will consider the one for judging whether a data set comes from a particular distribution.

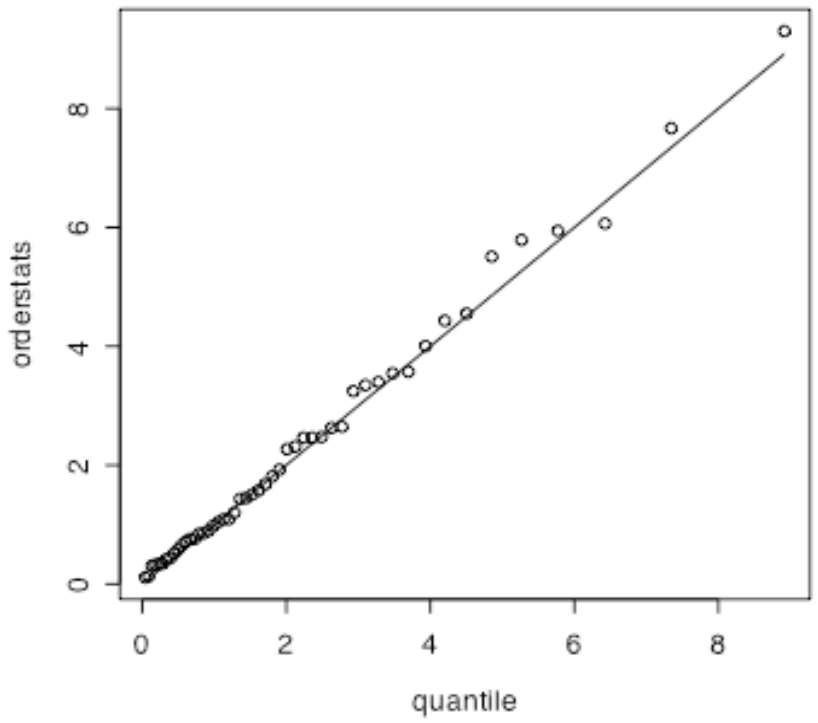
Definition: The order statistics of a sample are the data values arranged in order from lowest to highest. If the data are denoted $X_1, \ldots, X_n$, the order statistics are often written $X_{(1)}, \ldots, X_{(n)}$.

A QQ plot is a plot of the order statistics against the corresponding quantiles of the theoretical distribution. If the data really come from the distribution in question, the data value with p of the sample below it should line up with the p-th quantile, and the points in the plot should be close to the line $y=x$.

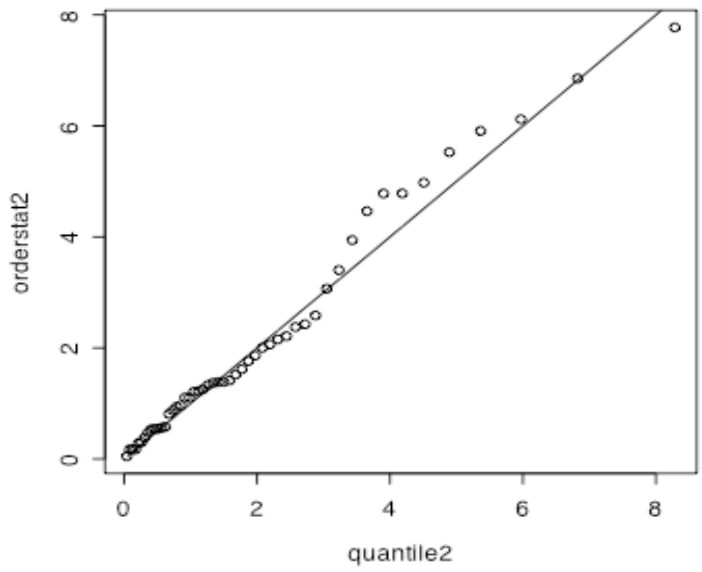
The standard way to do this is to plot the order statistics against the quantiles $k/(n+1)$ for $k = 1, \ldots, n$.

Yes, but the quantiles of what distribution? A typical distribution like the exponential is really a family of distributions, with parameter values that determine the member of the family. If the parameters are not given, it makes sense to estimate them from the data.

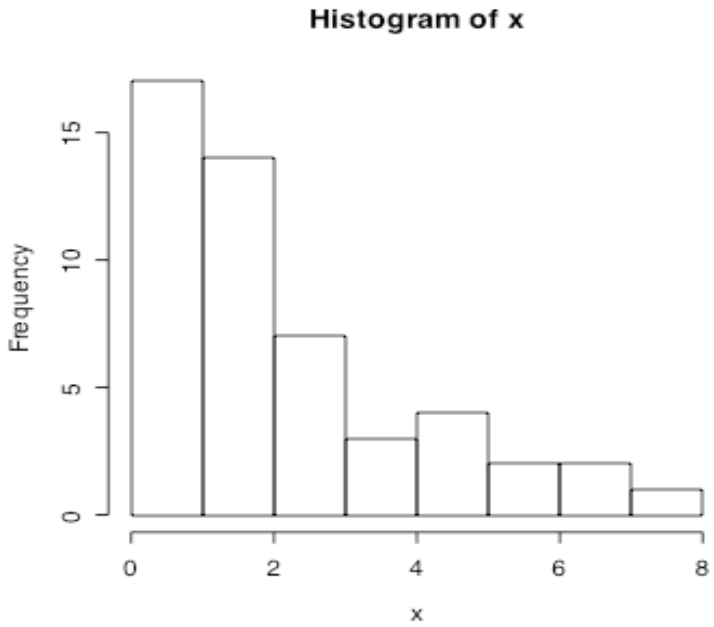
```
>
> orderstats <- sort(expdata)
> n <- length(expdata); thetahat <- mean(expdata)
> p = (1:n)/(n+1) # Vector of length n
> quantile = qexp(p, rate=1/thetahat)
> plot(quantile,orderstats)
> eq = quantile
> lines(quantile,eq,type='l') # That's a small L
```




```
> # It looks good. Let's examine one that looks bad.
> x <- rchisq(n,2) # n independent chi-squares, df=expected value = 2
> thetahat2 = mean(x); thetahat2
[1] 2.107169
> orderstat2 = sort(x); quantile2 = qexp(p, rate=1/thetahat2)
> plot(quantile2, orderstat2)
> eq2 = quantile2
> lines(quantile2,eq2,type='l')
```




```
> # A cluster of order statistics is higher than it should be. Let's look at
> # a histogram.
> hist(x)
```



Maybe the QQ plot is more revealing.

Related to the QQ plots are plots of the order statistics against their expected values under the theoretical distribution. In the case of the normal distribution, these are called *normal probability plots*.

```
> kars = read.table("http://www.utstat.toronto.edu/~brunner/2101f11/data/mcars2.data")
> kars[1:10,] # Rows 1:10, all columns
   country   kpl weight length
1        1  5.04 2178.0 591.82
2        2 10.08 1026.0 431.80
3        1  9.24 1188.0 426.72
4        1  7.98 1444.5 510.54
5        1  7.98 1485.0 502.92
6        1  7.98 1485.0 502.92
7        3  9.66  972.0 436.88
8        1  7.56 1665.0 543.56
9        3  5.88 1539.0 487.68
10       1 10.92 1003.5 431.80


> modelkar = lm(kpl~weight+length,data=kars)
> summary(modelkar)

Call:
lm(formula = kpl ~ weight + length, data = kars)

Residuals:
     Min       1Q   Median       3Q      Max
-2.97979 -0.84152 -0.08533  0.84027  5.04694

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 24.785719   2.693092   9.203 7.04e-15 ***
weight      -0.001616   0.001410  -1.146  0.25455
length      -0.028268   0.009270  -3.049  0.00296 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.643 on 97 degrees of freedom
Multiple R-Squared: 0.6291,   Adjusted R-squared: 0.6215
F-statistic: 82.26 on 2 and 97 DF,  p-value: < 2.2e-16

> weight = kars$weight; kpl = kars$kpl
> plot(weight,kpl)
```