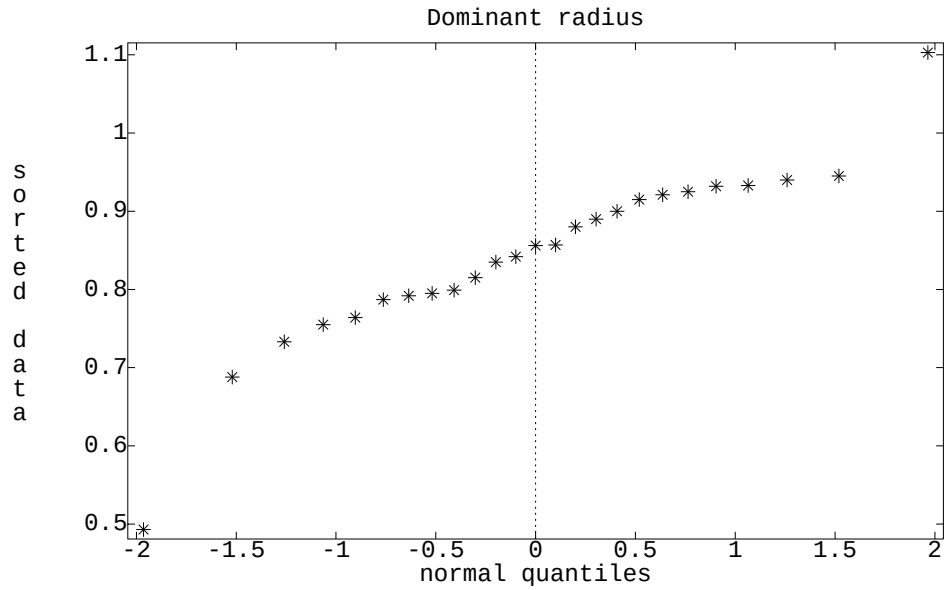
Statistics 5041
10. Assessing Normality

Gary W. Oehlert
School of Statistics
313B Ford Hall
612-625-1557
gary@stat.umn.edu

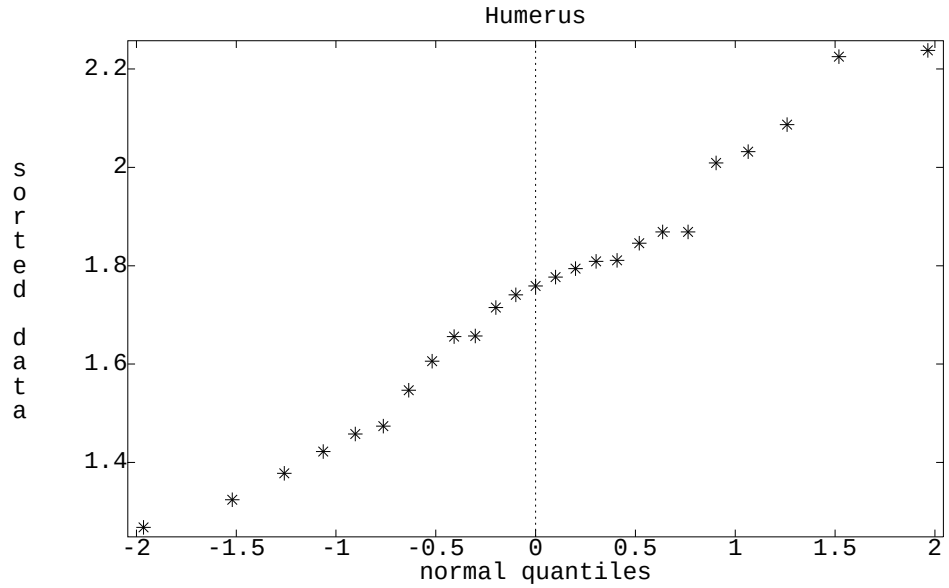
Many procedures assume normality.
Some procedures fall apart if the data aren't normal, whereas others can take a lot of abuse and keep going.
In either case, it's nice to know how close to normal the data are.
Assessing normality is an example of a *goodness-of-fit* problem.
Goodness-of-fit is a difficult problem.
There are *many* approaches. We will emphasize a graphical approach and the numerical follow-ons to the graphs.
Start with the univariate case.
Q-Q plots. Q-Q is short for "quantile-quantile." When testing normality, these are also known as normal probability plots and (locally) as rankit plots.
Make pairs (x_r, y_r), where x_r is the rth quantile of some theoretical distribution, and y_r is the rth quantile of the data.
Then make a plot of the pairs. If the data come from the theoretical distribution, the points on the plot should fall on a line with slope 1 and intercept 0. Plot will be linear if y quantiles are scaled x quantiles plus a constant: $y = \sigma x + \mu$.
Usually we choose levels r something like $i/(n+1)$ or $(i-.5)/n$ or $(i-3/8)/(n+1/4)$ for $i = 1, 2, \ldots, n$, so that the data quantiles are just the ordered data $y_{(1)} \le y_{(2)} \le \ldots \le y_{(n)}$.
For a continuous theoretical distribution with cumulative $F()$, the theoretical quantiles are just $F^{-1}(r_i)$.
It generally makes little difference which version of r_i we use.

```
Cmd> readdata("",dr,r,dh,h,du,u)
Read from file "/cdrom/T1-8.DAT"
Column 1 saved as REAL vector dr
Column 2 saved as REAL vector r
Column 3 saved as REAL vector dh
Column 4 saved as REAL vector h
Column 5 saved as REAL vector du
Column 6 saved as REAL vector u

Cmd> q1 <- invnor((run(25)-.5)/25)

Cmd> q2 <- invnor(run(25)/26)

Cmd> q3 <- invnor((run(25)-3/8)/(25.25))

Cmd> cor(hconcat(q1,q2,q3))
(1,1)            1      0.99912      0.99991
(2,1)      0.99912            1       0.9996
```

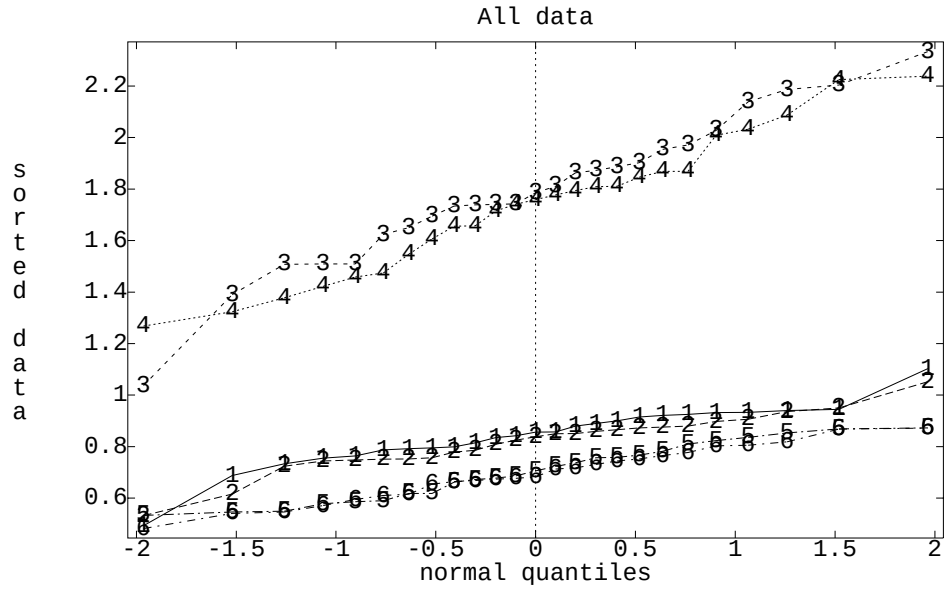
```
(3,1)      0.99991       0.9996            1

Cmd> cor(hconcat(q3,rankits(q3)))
(1,1)            1            1
(2,1)            1            1

Cmd> ord <- grade(dr)

Cmd> plot(q3,dr[ord],\
xlab:"normal quantiles",\
ylab:"sorted data",\
title:"Dominant radius")
```




```
Cmd> plot(rankits(h),h,\
xlab:"normal quantiles",\
ylab:"sorted data",title:"Humerus")
```




```
Cmd> yall <- hconcat(dr,r,dh,h,du,u)

Cmd> syall <- sort(yall)

Cmd> chplot(q3,syall,lines:T,\
xlab:"normal quantiles",\
ylab:"sorted data",title:"All data")
```

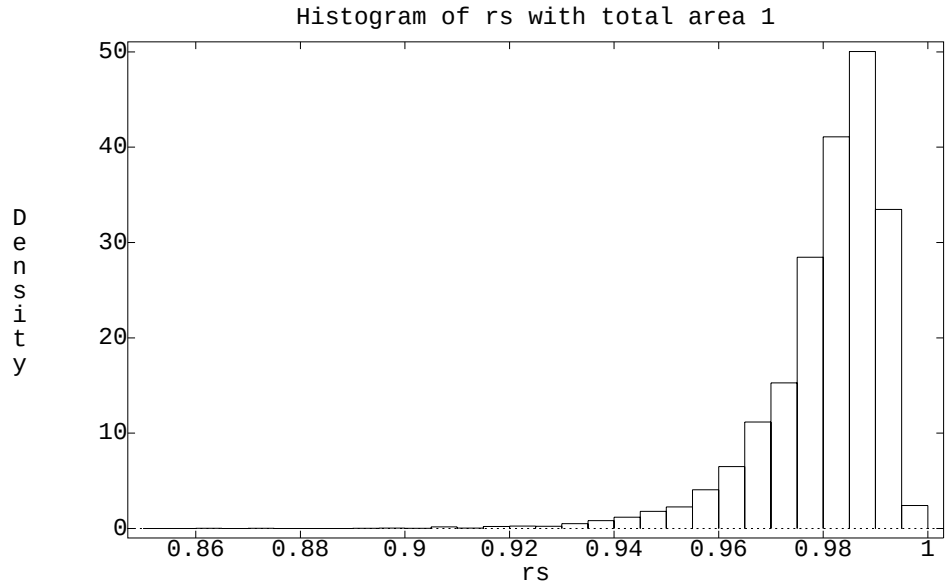


As a test of normality, consider the correlation of the normal quantiles x_r and the data quantiles y_r. Large values of the correlation are consistent with normality; small values are inconsistent.

```
Cmd> cor(q3,syall)[1,]
(1,1)     1   0.948  0.970  0.983  0.991
(1,6) 0.984   0.995
```

How small is too small?
No simple answer, so simulate the distribution.

```
Cmd> rs <- rep(0,10000)

Cmd> for(i,run(10000))
x <- sort(rnorm(25))
rs[i] <- cor(q3,x)[2];;


Cmd> min(rs)
(1)      0.86155

Cmd> max(rs)
(1)      0.99734

Cmd> hist(rs,run(.85,1,.005))
```




```
Cmd> obsr <- cor(q3,syall)[1,-1];obsr
(1,1)     1   0.948  0.970  0.983  0.991
(1,6) 0.984   0.995

Cmd> sum(rs<obsr)/10000
(1,1)  0.023  0.144   0.472  0.861  0.503
(1,6)  0.984
```

These are the simulated p-values for testing normality for the six variables.
I am not completely happy with many tests of normality, including this one.
When n is small, they have low power.

When n is big, they can detect deviations from normality that might be safe to ignore.
They need to be used in conjuction with judgement and other methods, such as assessment of the QQ plot.
What do we do for multivariate normal?
Usual approach is to see if properties known to hold for MVN hold for these data.
Normal marginals, normal conditionals, normal linear combinations, linear regressions of each variable on other variables, constant conditional variance, chi-squared distances, etc.
When we test normality for each variable, we are testing marginal normality. Recall the p-values:

```
Cmd> sum(rs<obsr)/10000
(1,1)  0.023  0.144   0.472  0.861  0.503
(1,6)  0.984
```

Should we reject normality because variable one has a p-value of .023?
Not necessarily.
We have a *multiple testing* situation (also called multiple comparisons or simultaneous inference).
In this problem we have p null hypotheses H_{0i} (normality of variable i) and an overall null hypothesis H_0 which is true if all the H_{0i} are true. If we test each H_{0i} at level α, then

$$\alpha \leq \tilde{\alpha} = P(\text{reject } \mathrm{H}_0) \leq p\alpha$$

$\tilde{\alpha}$ tends to be closer to $p\alpha$ for small α, but high correlations make $\tilde{\alpha}$ closer to α.
The *Bonferroni* adjustment says to reject H_0 if the smallest individual p-value is less than α/k, or equivalently if k times the smallest p-value is less than α, where k is the number of tests.
The *Bonferronizes* the test.
In our example $6 \times .023 = .138$ is not very small, and we would not take the marginal results as strong evidence against multivariate normality.
One common recommendation is to check the normality of $\mathbf{U}'x$, where $\mathbf{U}$ is the matrix of eigenvectors for $\mathbf{V}$, the variance matrix of x.
This rotates the data to axes of the "ellipse" and does normal testing marginally down each axis.
The word ellipse is in quotes above, because we might not have elliptical point clouds for nonnormal data.

```
Cmd> V <- tabs(yall,covar:T)

Cmd> U <- eigen(v)$vectors

Cmd> w <- yall%*%U; sw <- sort(w)

Cmd> obsrw <- cor(q3,sw)[1,-1];obsrw
(1,1)   0.990   0.968   0.985   0.970   0.987
(1,6)   0.986

Cmd> sum(rs<obsrw)/10000
(1,1)   0.808   0.121   0.571   0.149   0.650
(1,6)   0.605
```

If $x \sim \mathrm{N}_p(\mu, \Sigma)$, then

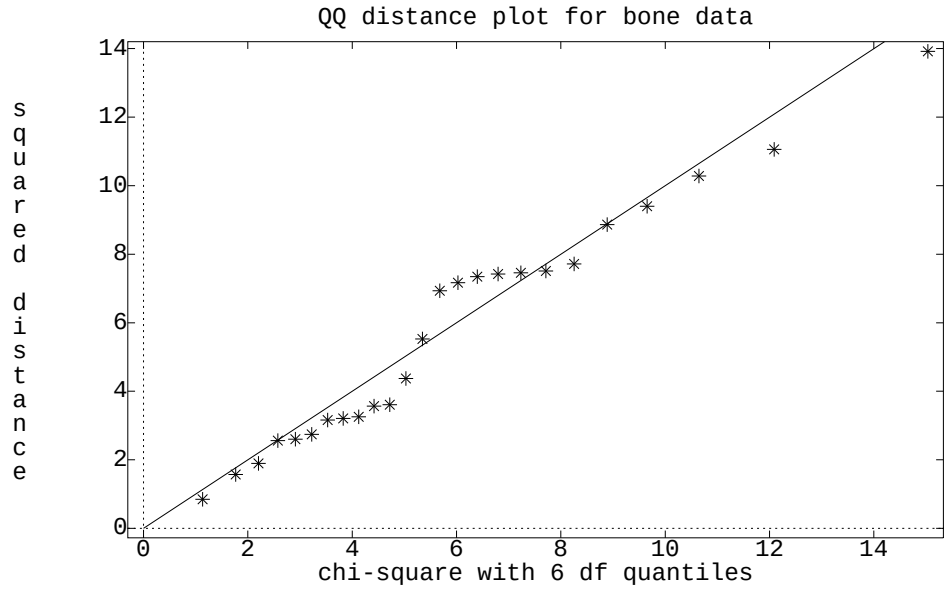
$$(x-\mu)'\Sigma^{-1}(x-\mu) \sim \chi^2_p$$

If $n-p$ is large, then

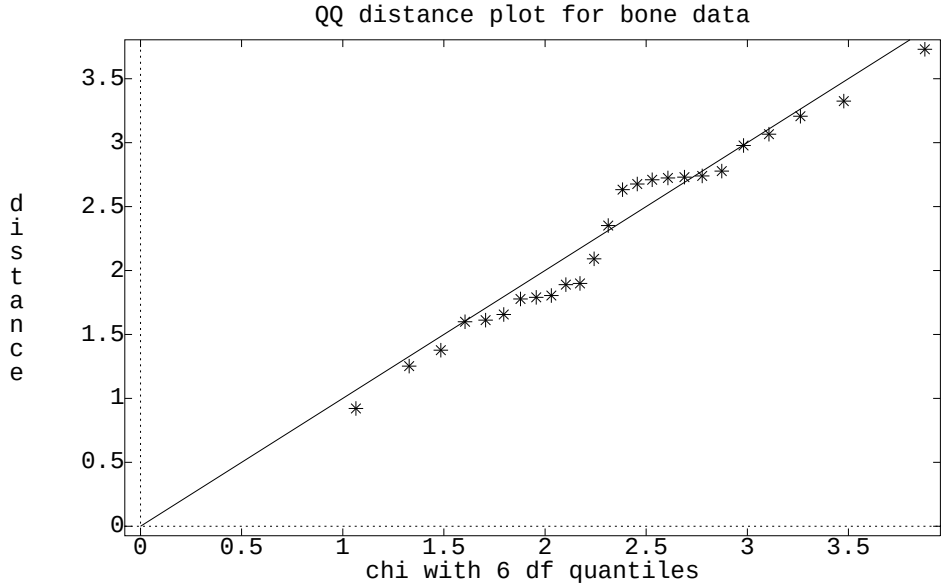
$$d_i^2 = (\vec{\mathbf{X}}_i - \overline{\mathbf{x}})'\mathbf{S}^{-1}(\vec{\mathbf{X}}_i - \overline{\mathbf{x}})$$

should also be approximately χ^2_p.
Use a Q-Q plot with data d_i^2 and horizontal values the percent points of χ^2_p. The square roots of these values will sometimes work better for small p.

```
Cmd> d <- distcomp(yall);d
 (1)   8.866   7.459   0.848   3.159   4.374
 (6)  10.283   3.569   3.208   3.254   2.600
(11)   1.570   9.402   1.897   2.742   5.529
(16)   7.511   6.931   2.560   7.714   7.170
(21)  11.057   7.345  13.922   3.607   7.422

Cmd> sd <- sort(d)

Cmd> q <- invchi((run(25)-.5)/25,6)

Cmd> plot(q,sd,\
xlab:"chi-square with 6 df quantiles",\
ylab:"squared distance",\
title:"QQ distance plot for bone data",
show:F)

Cmd> addlines(vector(0,15),vector(0,15),\
show:F)

Cmd> showplot(xmin:0,ymin:0)
```



```
Cmd> plot(q^.5,sd^.5,\
xlab:"chi with 6 df quantiles",\
ylab:"distance",\
title:"QQ distance plot for bone data",\
show:F)

Cmd> addlines(vector(0,4),vector(0,4),\
show:F)

Cmd> showplot(xmin:0,ymin:0)
```



We might want to build a test based on this plot, but we can't use correlation. We really want it to be on the line with intercept 0 and slope 1.

Try deviations from the line, or weighted deviations from the line. For example, `(sd-q)^2` or `(sd-q)^2/q`. I suspect that larger order statistics have more variance, so I may want to downweight them. Maybe dividing by `q` (more or less the expected value) will work.

Sample a bunch of chisquares, order them, and compute variances.

```
Cmd> D <- matrix(invchi(runi(25000),6),25)

Cmd> D <- sort(D)

Cmd> D <- D'

Cmd> tabs(D,var:T)
 (1)   0.286   0.329   0.340   0.375   0.402
 (6)   0.424   0.448   0.472   0.498   0.549
(11)   0.581   0.611   0.654   0.700   0.785
(16)   0.833   0.908   1.086   1.190   1.361
(21)   1.658   2.044   2.886   4.467  10.124
```

```
Cmd> tabs(D,var:T)/q
 (1)   0.252   0.186   0.154   0.146   0.138
 (6)   0.131   0.127   0.123   0.121   0.124
(11)   0.123   0.121   0.122   0.123   0.130
(16)   0.130   0.134   0.150   0.154   0.165
(21)   0.187   0.212   0.271   0.370   0.673

Cmd> tabs(D,var:T)/q^2
 (1)   0.222   0.106   0.070   0.057   0.047
 (6)   0.041   0.036   0.032   0.029   0.028
(11)   0.026   0.024   0.023   0.022   0.022
(16)   0.020   0.020   0.021   0.020   0.020
(21)   0.021   0.022   0.025   0.031   0.045

Cmd> D <- D'

Cmd> dout <- sum( (D-q)^2/q)

Cmd> dout <- vector(dout)

Cmd> sum( (sd-q)^2/q)
(1)       2.0092

Cmd> length(dout)
(1)         1000

Cmd> sum(dout > 2.009)/1000
(1)          0.8
```

This messing around with `q` as a scaling factor is not the best we can do.
We estimated the variance matrix of the order statistics during our simulation. We can make a better test using those variances.