

Normal Probability Plot

Shaillay Kumar Dogra
Scientific Editor – QSAR World
editor@qsarworld.com

Notes:

1. This Jython script works in Sarchitect Designer version 2.2
2. Learn about Sarchitect Designer – <http://www.strandls.com/sarchitect/index.html>
3. Get Sarchitect – <http://www.strandls.com/sarchitect/freetrial.php>

The actual script follows this discussion. It is also accessible directly from the webpage in .py format.

Discussion:

Assumption of normal distribution of data is an important prerequisite for some statistical tests (parametric) and regression methods. This assumption can be tested by using various graphical methods like rankit plots, normal probability plots and tests like Shapiro-Wilk [SW] or Kolmogorov-Smirnov [KS] tests.

A simple way to test the normality assumption could be to just look at the distribution of data as a histogram and see if it assumes a bell-shaped distribution that is characteristic of a normal or Gaussian distribution [ND]. If this is not the case, some transformations like log transformation can be attempted to see if now the new distribution takes more of a bell shaped curve.

Also, parameters like skewness and kurtosis can be checked. Quantile distribution of values can be looked at to see if a given mass of distribution is falling below a given %ile position patterning the indicative trend of a normal distribution [ND].

For more discussion about normality tests read [GraphPad, Eng-Stat, wikipedia].

I am not clear if nomenclature-wise the normal-probability plot is the same as rankit plot or the Quantile-Quantile plot. Seemingly, there is hardly any difference between the rankit and the Q-Q plot unless sample size happens to be small.

I have also provided a script for generating the rankit plot, that runs in sarchitect designer 2.2, at QSAR-World [script]. The script provided here is for getting the Q-Q plot.

For more on normal probability and Q-Q plots read [NP-plot, norm-prob, QQ, Q-Q, Rodríguez]

Input to the script is a column containing continuous values for which normal-probability plot needs to be generated. A UI box prompts one to provide that column. A typical use may be to run the plot on residuals columns. Any of the input data columns can also be tested to check if data transformations are required to make the distribution adhere to normality.

Output is a child dataset that has in the spreadsheet all the columns that were 'marked' in the parent dataset, a "rankits" column, user-selected input column, and a dummy label column for connecting the points in the graph.

Script gives error if any of the 'marked' columns is provided as input.

References:

[SW] <http://www.itl.nist.gov/div898/handbook/prc/section2/prc213.htm>

[KS] <http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm>

[ND] http://en.wikipedia.org/wiki/Normal_distribution

[GraphPad] http://www.graphpad.com/library/BiostatsSpecial/article_197.htm

[Eng-Stat] <http://www.itl.nist.gov/div898/handbook/prc/section2/prc21.htm>

[wikipedia] http://en.wikipedia.org/wiki/Normality_test

[script] <http://www.qsarworld.com/virtual-workshop.php>

[NP-plot] http://en.wikipedia.org/wiki/Normal_probability_plot

[norm-prob] <http://www.itl.nist.gov/div898/handbook/eda/section3/normprpl.htm>

[QQ] http://en.wikipedia.org/wiki/Quantile-quantile_plot

[Q-Q] <http://www.itl.nist.gov/div898/handbook/eda/section3/qplot.htm>

[Rodríguez] "Chapter 2. Linear Models for Continuous Data", Germán Rodríguez, 1993-2000, pp. 57-58. <http://data.princeton.edu/wws509/notes/default.htm>

Cite this as:

Dogra, Shaillay K., "Script for getting Normal Probability plot" from QSARWorld – free online resource for QSAR modeling. <http://www.qsarworld.com/virtual-workshop.php>

```

##
##
## sarchitect designer 2.2 script to get 'Normal Probability plot'
##
## Shaillay Kumar Dogra
## Oct 20, 2006
## editor@qsarworld.com
##
##
## Input: A Continuous Column for which normal-probability graph needs
## to be plotted
##
##
## Output: A subset with ALL the MARKed columns + "Rankits" column +
## user selected column + a dummy string column for connecting points ## in the scatter plot
##
## A scatter plot b/w rankits and user-selected column
## (points are connected to their neighbours)
##
##
## References :
##   http://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm
##   http://www.itl.nist.gov/div898/handbook/eda/section3/normprpl.htm
##
##

import script
from script.dataset import *
from script.project import *
from script.omega import createComponent, showDialog
from javax.swing import *
from math import *
from script.view import *

##-----
## INVERSE OF NORMAL DISTRIBUTION FUNCTION
##
## taken from Ramesh's (CTO, Strand Life Sciences) "Normal Probability ## Plot" script
##
def norminverse(p):
    # Coefficients in rational approximations
    a = [-3.969683028665376e+01, 2.209460984245205e+02,-2.759285104469687e+02,
1.383577518672690e+02,-3.066479806614716e+01, 2.506628277459239e+00]
    b = [-5.447609879822406e+01, 1.615858368580409e+02,-1.556989798598866e+02,
6.680131188771972e+01,-1.328068155288572e+01];
    c = [-7.784894002430293e-03, -3.223964580411365e-01,-2.400758277161838e+00, -
2.549732539343734e+00,4.374664141464968e+00, 2.938163982698783e+00]
    d = [7.784695709041462e-03, 3.224671290700398e-01,2.445134137142996e+00,
3.754408661907416e+00];

    #Define break-points.
    plow = 0.02425;
    phigh = 1 - plow;

```

```

#Rational approximation for lower region:
if ( p < plow ):
    q = sqrt(-2*log(p));
    return (((((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5])
*1.0/(((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1);

#Rational approximation for upper region:
if ( phigh < p ):
    q = sqrt(-2*log(1-p));
    return -
((((c[0]*q+c[1])*q+c[2])*q+c[3])*q+c[4])*q+c[5])*1.0/(((d[0]*q+d[1])*q+d[2])*q+d[3])*q+1);

#Rational approximation for central region:
q = p - 0.5;
r = q*q;
return (((((a[0]*r+a[1])*r+a[2])*r+a[3])*r+a[4])*r+a[5])*q
*1.0/(((b[0]*r+b[1])*r+b[2])*r+b[3])*r+b[4])*r+1);

##-----
## COMPUTE RANKITS
def rankits():
    sample_size = dataset.getRowCount()
    n = sample_size
    rankits = [ ]

    ## for computing the first rankit
    val = (1.0 - (0.5**(1.0/n)))
    rankits.append(norminverse(val))

    #for computing the middle rankits
    for i in range(1, sample_size-1):
        val = (i-0.3175)/(n+0.365)
        rankits.append(norminverse(val))

    ## for computing the last rankit
    val = (0.5**(1.0/n))
    rankits.append(norminverse(val))

    return rankits

##-----
## GET POINTS IN SCATTER-PLOT CONNECTED
def setConnectColumn(view, connect_idx, order_idx):
    c = view.connectBy
    c['connect'].columnIndex = connect_idx
    c['order'] = order_idx
    view.connectBy = c

##-----

## MAIN

dataset = script.project.getActiveDataset()

```

```
sample_size = dataset.getRowCount()

## COLUMN CHOICE DROPDOWN-BOX
p = createComponent(type="column", id="name", description="Which
Column?", dataset=script.project.getActiveDataset())

resid_col_idx = showDialog(p)
indices = dataset[resid_col_idx].getRowIndicesInSortedOrder(1)

rankit_list = rankits()

## Here we are shuffling the various rankits and putting them in ordered pairing
## with the values in the above-selected column; this gives us 'sorted' pairing

ordered_rankit = [0.0] * sample_size
i = 0
for idx in indices:
    ordered_rankit[idx] = rankit_list[i]
    i = i+1

## Just to get a "connectColumn" to be able to join points in the scatter-polt later
dummy_label = ["ABC"] * sample_size

## DEFINE A NEW CHILD-DATASET
rowIndices = [i for i in range(sample_size)]
colIndices = []
script.project.addSubsetChild(rowIndices, colIndices, name="Normal-Probability-Set")
subset=script.project.getActiveDataset()

rankitsCol=createFloatColumn("Rankits", ordered_rankit)
subset.addColumn(rankitsCol)

subset.addColumn(dataset[resid_col_idx])

ConnectCol=createStringColumn("Dummy Labels", dummy_label)
subset.addColumn(ConnectCol)

script.view.Table().show()

## SCATTER PLOT
total_cols = subset.getColumnCount()
y_idx = (total_cols - 2) ## second last column we appended was the user selected column
ConnectIdx = (total_cols - 1) ## last column we appended was the dummy label column
x_idx = subset.index("Rankits")
OrderIdx = x_idx
plot = ScatterPlot(title = 'Normal Probability Plot', yaxis=y_idx, xaxis=x_idx)

setConnectColumn(plot, ConnectIdx, OrderIdx)

plot.show()
```

```
## REPORT COMPLETION  
parent=script.tool.getTool().getFrame()  
mesg = "Done With Script Execution."  
JOptionPane.showMessageDialog(parent,mesg,"STATUS!",JOptionPane.INFORMATION_MESS  
AGE)
```

```
##  
## DONE  
##
```

End of Document