

## **Linear Regression Diagnostics**

Shaillay Kumar Dogra  
Scientific Editor – QSAR World  
[editor@qsarworld.com](mailto: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 from page 3 through page 6. It is also accessible directly from the webpage in .py format.***

### **Discussion:**

Various Linear Regression Diagnostics computed by this script are:

- 1) Ordinary Residuals
- 2) Standardized Residuals
- 3) Jack-Knifed Residuals
- 4) Leverage
- 5) Leverage Status
- 6) Cook's Distance

The formulae for each of these have been given in the comments section of the script and the same have been taken from the notes given in (Rodríguez) and compared with notes of (Guha). User is advised to go through these notes to both understand the concepts and the relevant cut-offs for various formulae.

The Input columns required are:

- “Actual” values column
- “Predicted” values column
- “Standard Error” column

In all possibility, column 1 should already be present in your spreadsheet. When you run predictions on some test set, column 2 and 3 get automatically appended to the spreadsheet.

You will also need to type-in the number of descriptors used in the MLR model, which you ran to obtain the predictions, when prompted by a dialog-box.

On successful completion of the script, the above mentioned columns get appended to the spreadsheet. Also, a scatter-plot and a histogram are launched. This can be used to look at say, Williams plot for example, which is a plot of standardized residuals versus leverage values (Eriksson et al, Gramatica) or simply a residuals plot (Rodríguez), which is a plot of the residuals versus the fitted values. One may also look at the standardized or jack-knifed residuals versus the fitted values. There should not be any specific pattern in these plots (Rodríguez) such as:

i) trend: negative residuals observed at small predicted values and positive residuals observed at large predicted values, which would imply non-linearity in data

ii) non-uniform spread: clustered residuals observed for small predicted values and dispersed residuals observed for large predicted values, which would imply heteroscedasticity.

Leverage values for the training set compound indicate those compounds that may have (potential) influence on model parameters (fit may be strongly dependent on these). For the test compounds these indicate the applicability domain of the model (Eriksson et al, Gramatica).

Cook's distance combines residuals and leverage in a single measure that indicates (actual) influence of a compound on the model parameters (Rodríguez).

The squared Cook's distance should be  $< 1.0$  (Tropsha et al, Rodríguez) and the standardized residual should be  $< 2.5$  (Tropsha et al),  $< 2.0$  (Eriksson et al) and  $< 3.0$  (Gramatica). The warning leverage is set at  $3*k/n$ , where  $n$  is the number of compounds in the training set and  $k$  is the number of descriptors in the model (Eriksson et al, Tropsha et al, Gramatica). High leverage has also been defined as  $> 2*k/n$  (Rodríguez).

#### References:

[Rodríguez] "Chapter 2. Linear Models for Continuous Data", Germán Rodríguez, 1993-2000, pp. 49-58. <http://data.princeton.edu/wws509/notes/default.htm>

[Guha] <http://cheminfo.informatics.indiana.edu/~rguha/writing/notes/stats/node6.html>

[Tropsha] The Importance of Being Earnest: Validation is the Absolute Essential for Successful Application and Interpretation of QSPR Models, Alexander Tropsha, Paola Gramatica, and Vijay K. Gombar, QSAR Comb. Sci. 22 (1), 69-77, 2003.  
<http://www3.interscience.wiley.com/cgi-bin/abstract/104525640/ABSTRACT>

[Eriksson] Methods for reliability and uncertainty assessment and for applicability evaluations of classification- and regression-based QSARs, Lennart Eriksson, Joanna Jaworska, Andrew P Worth, Mark T D Cronin, Robert M McDowell, and Paola Gramatica, Environ Health Perspect., 111(10): 1361-1375, 2003.  
<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1241620>

[Gramatica] Principles of QSAR models validation: internal and external, Paola Gramatica, QSAR Comb. Sci., 2007, 26(5), 694-701.  
<http://www3.interscience.wiley.com/cgi-bin/abstract/114182462/ABSTRACT>

#### Cite this as:

Dogra, Shaillay K., "Script for computing Linear Regression Diagnostics" from QSARWorld - free online resource for QSAR modeling. <http://www.qsarworld.com/virtual-workshop.php>

```

##
## sarchitect_designer_2.2 script to get "regression diagnostics"
##
## Shaillay Kumar Dogra
## 18 Oct 2006
## editor@qsarworld.com
##
##
## INPUTs required: (in that order)
##   -> "actual values" column
##   -> "predicted values" column
##   -> "standard error" column
##
##   -> "number of descriptors" used in LR model
##
##
## OUTPUTs:
##   -> 'Ordinary Residuals' column
##   -> 'Standardized Residuals' column
##   -> 'Jack-Knifed Residuals' column
##   -> 'Leverage' column
##   -> 'Leverage Status' column
##   -> 'Cook's Distance' column
##
##   -> launches scatterplot
##   -> launches histogram
##
##
## Mathematics:
##
## 'ordinary residuals' is (actual - predicted)
## 'standardized residuals' is (actual - predicted)/{SQRT(1 - h_i) * stdDev}
## 'jack-knifed residuals' is (std-resid_i) * SQRT{(n-p-1)/(n-p-(std-resid_i)2)}
## 'leverage' is (stdErr_i/stdDev)2
## 'leverage status' is 'high' if leverage > 3(p+1)/n
## 'Cook's distance' is ((stdErr_i)2 * h_i) / (1-h_i)*(p+1)
##
##
## Now, Standard-error (reported in sarchitect regression results) is:
##   stdErr_i = stdDev * sqrt(h_i)
##   => h_i = (stdErr_i/stdDev)2
##
## Also, (stdDev)2 = RSS / dfE
## where - RSS = SUM(actual - predicted)2
##       - dfE = (n-p-1) [when LR is done withIntercept]
##       or dfE = (n-p) [when LR is done withoutIntercept]
##
## 'n' - no. of compounds, 'p' - no. of descriptors used in LR model
##
## Since division by 'dfE' (instead of 'n') is a bias correction in the estimation of
## (stdDev)2 or 'variance' we will stay conservative and always use dfE = (n-p-1)
## whether LR model happens to be 'withIntercept' or 'withoutIntercept'.
##

```

```

##          Thus, 'standardized residuals' become -
##
##          'ordinary residuals' / [ SQRT( (stdDev)2 - (stdErri)2 ) ]
##
##
##
##
import script
from script.dataset import *
from script.omega import createComponent, showDialog
from javax.swing import *
from math import *
from script.view import *

##-----
def textarea(text):
    t = JTextArea(text)
    t.setBackground(JLabel().getBackground())
    return t

##-----
## PANEL TO GET COLUMN SELECTION
def getInputColumns():
    helptext=""
    Select 'actual', 'predicted' and 'standard-error' columns.
    Make 'actual' column the first selection, 'predicted' column the second selection and so on.
    ""
    helpPanel=createComponent(type="ui",id="help",
description="",component=textarea(helptext))
    colPanel = createComponent(type="ColumnSelector",id="columns", description="Select
Columns",dataset=script.project.getActiveDataset())
    finalPanel= createComponent(type="group",id="Provide Actual and Predicted Columns",
description="Select Columns",components=[helpPanel,colPanel])
    result=showDialog(finalPanel)
    return result['columns']

##-----
## PANEL TO GET NUMBER OF DESCRIPTORS USED IN MODEL
def getNumDescriptors():
    p = createComponent(type="int", id="name", description="No. of Descriptors used in
model?",value="0")
    result=showDialog(p)
    return result

##-----
## COMPUTE SQUARE OF STANDARD DEVIATION (VARIANCE)
def StdDevSqr():
    ## compute sum of squares of residuals
    RSS = 0
    for i in residual_col:
        RSS = RSS + (i*i)

    degree_of_freedom = (no_of_compounds - no_of_descriptors - 1)
    sqr_stdDev = (RSS / degree_of_freedom)
    return sqr_stdDev

```

```

##-----
## COMPUTE STANDARDIZED RESIDUALS (INTERNALLY STUDENTIZED RESIDUALS)
def IntStdResid():
    stdDev_sqr = StdDevSqr()
    int_standardized_residual = [ ]
    for idx in range(no_of_compounds):
        denominator = sqrt((stdDev_sqr)-(std_err[idx]*std_err[idx]))
        std_resid = (residual_col[idx] / denominator)
        int_standardized_residual.append(std_resid)

    return int_standardized_residual

##-----
## COMPUTE JACK-KNIFED RESIDUALS (EXTERNALLY STUDENTIZED RESIDUALS)
def ExtStdResid():
    int_std_resid = dataset.getColumn("Standardized Residuals")
    ext_standardized_residual = [ ]
    for val in int_std_resid:
        std_resid = val * sqrt( (no_of_compounds - no_of_descriptors - 1) /
(no_of_compounds - no_of_descriptors - (val * val)) )
        ext_standardized_residual.append(std_resid)

    return ext_standardized_residual

##-----
## COMPUTE LEVERAGE
def Leverage():
    stdDev_sqr = StdDevSqr()
    threshold = (3.00 * (no_of_descriptors + 1.00) / (no_of_compounds))
    leverage = [ ]
    warning = [ ]
    for idx in range(no_of_compounds):
        lvrg = ( std_err[idx] * std_err[idx] ) / stdDev_sqr
        leverage.append(lvrg)
        if (lvrg > threshold):
            warning.append("HIGH")
        else:
            warning.append("OK")

    return leverage, warning

##-----
## COMPUTE COOK'S DISTANCE
def CooksDist():
    StdResidCol = dataset.getColumn("Standardized Residuals")
    LvrgCol = dataset.getColumn("Leverage")
    cooks_distance = [ ]
    for idx in range(no_of_compounds):
        numtr = (StdResidCol[idx] * StdResidCol[idx]) * LvrgCol[idx]
        denmtr = (1 - LvrgCol[idx]) * (no_of_descriptors + 1)
        distance = (numtr/denmtr)

        cooks_distance.append(distance)

    return cooks_distance

```

```
##-----  
  
## MAIN  
  
dataset=script.project.getActiveDataset()  
  
## GET REQUIRED COLUMNS  
columns=getInputColumns()  
  
## get 'n'  
no_of_compounds = dataset.getRowCount()  
  
## get 'p'  
no_of_descriptors = getNumDescriptors()  
  
actual_value = dataset[columns.get(0)]  
pred_value = dataset[columns.get(1)]  
std_err = dataset[columns.get(2)]  
  
residual_col = (actual_value - pred_value)  
dataset.addColumn(createFloatColumn("Ordinary Residuals", residual_col))  
  
std_residual_col = IntStdResid()  
dataset.addColumn(createFloatColumn("Standardized Residuals", std_residual_col))  
  
jack_knife_resid_col = ExtStdResid()  
dataset.addColumn(createFloatColumn("Jack-Knifed Residuals", jack_knife_resid_col))  
  
(leverage_col, warning_col) = Leverage()  
dataset.addColumn(createFloatColumn("Leverage", leverage_col))  
dataset.addColumn(createStringColumn("Leverage Status", warning_col))  
  
cooks_dist_col = CooksDist()  
dataset.addColumn(createFloatColumn("Cook's Distance", cooks_dist_col))  
  
## SCATTER PLOT  
xcol = dataset.index(dataset[columns.get(1)])  
ycol = dataset.index("Ordinary Residuals")  
plot = ScatterPlot(yaxis = ycol, xaxis = xcol)  
colorIdx = dataset.index("Leverage Status")  
plot.colorBy.columnIndex = colorIdx  
plot.show()  
  
## HISTOGRAM  
col = dataset.index("Ordinary Residuals")  
Histogram(column = col).show()  
  
## REPORT COMPLETION  
parent=script.tool.getTool().getFrame()  
mesg = "Done With Script Execution."  
JOptionPane.showMessageDialog(parent,mesg,"STATUS!",JOptionPane.INFORMATION_MESSAGE)  
  
----- End of Document -----
```