

Boltzmann Probability

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:

This script computes Boltzmann-probability (of existence) for a conformer of a given set of conformers for a structure. It was written for some work wherein I wanted to get averaged descriptors for a set of conformers (for the given structure). In usual QSAR modeling scenario, one works with the descriptors of a single optimized structure. This does not affect the constitutional or topological (2D) descriptors but has influence over the conformational (3D) descriptors that are but naturally dependent on the conformation adopted by the molecule. I wanted to check if the QSAR models obtained by the two approaches – averaged descriptors from conformers versus descriptors from single optimized structure – are any different.

Details of this work with some interesting results are available at QSAR-World [qsarworld].

We obtained conformers for the compounds under study in MOE and used two fields from the generated files for computations – a column containing the Energy values and another with the identifier tag (that indicates all conformers of a structure with the same label). For this reason, in the script, variables dealing with conformer-IDs are named after 'mseq', a column name as in MOE for labeling structures.

For the purpose of averaging descriptors, I assigned weights to each of the conformers (that exist within certain Energy limits, as explained later) based on their probability of existence per the Boltzmann distribution. Another script does the averaging part. This one ends at computing the p-values to be used later by the averaging script.

The probability of a molecule to adopt a certain conformation may be calculated from a normalized Boltzmann distribution as:

$$P_j = \exp(-\Delta\text{Energy}_j/RT) / \sum \exp(-\Delta\text{Energy}_j/RT)$$

where ΔEnergy_j is the relative energy of the j-th conformation (in J/mol), R is the gas constant (8.314 J/mol/K), T is the temperature in Kelvin and N is the number of conformations with $\Delta\text{Energy}_j \leq 2.5$ kcal/mol

As mentioned above, as input the script needs two columns – a column containing the Energy values and another with labels for set of conformers. Various columns get appended as and when they are calculated - E_min, E_delta, E_exp_term, E_exp-sum, and p-value.

There is lot of explanation in the script as comments.

Possible improvements could be:

1. To not scan through all the data every time related set of conformers are required for some computation
2. sort at first go and then work with sorted data, which now has all the conformers together in a single block.
3. The way I run through the data uses 'range' function and assumes that conformer-IDs
(a) start from 1 and
(b) are continuous with no integer gaps in labels and run till maximum integer tag, this assumption worked fine with the data that was generated in MOE but may cause problems with other variations.
4. Lastly, possibly some of the computations could have been done in a single instance than creating a column with some intermittent value and then working further with this column.

References:

[qsarworld] Dogra, S. K., Das, A. and Subramanian K. 'Accounting for 3D descriptors of conformers in QSAR modeling' Oral presentation made at the 233rd National Meeting of the American Chemical Society, March 25-29, 2007 Chicago, IL. <http://www.qsarworld.com/qsar-poster-gallery.php?mm=9>

[Palm] Correlation of Drug Absorption with Molecular Surface Properties Palm K et. al. J. Pharm. Sci. 85, 1, Jan 1996, pp. 32-39.

Cite this as:

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

```
##
##
## sarchitect designer 2.2 script to get Boltzmann-probability values ## for a set of conformers of
the given structure
##
## Shaillay Kumar Dogra
## editor@qsarworld.com
## 25 July 2006
##
##
## Input:
## 1) a column containing Energy values for the conformers
## (based on which Boltzmann-probability will be calculated)
## 2) a column labeling set of conformers of a given structure
## (with an integer tag)
##
##
## Note: Ensure that 'DATA-TYPE' is float (and not string) for 'Energy' ## column is int (and not
string) for 'Conformer Identity' column
##
##
##
## Comments in PART- I, II & III in this script
##
##
##
##
##
##
## PART-I
##
## Asks users for two columns - 'Energy' and 'Conformer Identity'.
##
## The 'Conformer Identity' column that contains the ID tag for a set ## of conformers (belonging
to the same structure), can be any column ## with such a reference tag
##
## Script scans through dataset looking for same set of conformer-IDs ## (all conformers of a
given structure);
## finds the minimum value of Energy for same set of conformer-IDs
## (given conformers of a structure);
## Subtracts this minimum from Energy value of each conformer in the
## same set of conformer-IDs (given conformers of a structure)
##
## Thus,  $E_{\text{delta}} = (E_i - E_{\text{minimum}})$ 
##
## (Even though same set of conformer-IDs usually occur together as a ## block of rows in the
dataset, script scans through 'ALL' data not
## making this assumption. What if this was not the case, script should
## not break down. Possibly should have used some 'sort' function.)
##
##
## Appends 2 columns to dataset :  $E_{\text{min}}$ ,  $E_{\text{delta}}$ 
##
##  $E_{\text{min}} = \text{minimum}(\text{Energy})$  : computed for a set of conformer-IDs
##  $E_{\text{delta}} = (E_i - E_{\text{min}})$  : computed for a set of conformer-IDs
##
##
##
```

```

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

dataset=script.project.getActiveDataset()

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

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

## GET UNIQUE mseq ENTRIES ## may fail if intermediate numbers do not exist; did not test
this possibility
start = min(dataset[mseq_col])
stop = max(dataset[mseq_col])
total = len(dataset[mseq_col])
#print start, stop, total

## INITIATE REQUIRED COLUMNS
Energy_minimum = [0] * total
Energy_delta = [0] * total

for i in range(stop): ## assumption here that conformer-IDs start from "1"
    print "For conformer-ID:", i+1, "finding E_min"
    Energy_min = 10000000 # set some arbitrarily high value that may not possibly exist
    for j in range(total): # go through ALL data looking for particular conformer-ID
        #print i+1, j, dataset[Energy_col][j]
        #print dataset[mseq_col][j]
        if (dataset[mseq_col][j] == i+1): # when match happens
            Energy_now = dataset[Energy_col][j]
            #print "Conformer-ID:", i+1, "Row:", j, "matched-ID:",
dataset[mseq_col][j], "Energy_value:", dataset[Energy_col][j]
            #print i+1, Energy_now
            if (Energy_min < Energy_now):
                Energy_min = Energy_min
            elif (Energy_min > Energy_now):
                Energy_min = Energy_now
            #print "Conformer-ID:", i+1, "Energy_min:", Energy_min

    for j in range(total): # go through ALL data looking for particular conformer-ID
        if (dataset[mseq_col][j] == i+1): # when match happens
            Energy_minimum[j] = Energy_min

```

```

Energy_subtract = Energy_min
Energy_delta[j] = dataset[Energy_col][j] - Energy_subtract
#print "Conformer-ID:", i+1, "row:", j, Energy_delta[j]

## end of loop

## APPEND COLUMNS
new_column=createFloatColumn("E_min", Energy_minimum)
dataset.addColumn(new_column)

new_column=createFloatColumn("E_delta", Energy_delta)
dataset.addColumn(new_column)

##
##
## PART - II
##
##
## Convert Energy from kcal to joules
## Compute exp(-Energy_Delta/R*T)
##
## All these computations could have been done in above loop computing
## 'Energy_delta'
##
## Appends a column to data-set - 'E_exp_term'
##
##
##

R = 8.314 # R is the gas constant 8.314 J/mol/K
T = 300 # T is the temperature in Kelvin

constant = (R * T)

## INITIATE REQUIRED COLUMNS
Energy_joules = [0] * total
Energy_exp_term = [0] * total

for i in range(total):
    if (i%100 == 0): ## print after every 100th row
        print "For row:", i, "computing E_exp term"
        Energy_joules[i] = Energy_delta[i] * 1000 * 4.184 ## 1 cal=4.184 joule
        Energy_exp_term[i] = exp ( (-Energy_joules[i]) / (constant) )

## APPEND COLUMNS
#new_column=createFloatColumn("E_joules", Energy_joules)
#dataset.addColumn(new_column)

new_column=createFloatColumn("E_exp_term", Energy_exp_term)
dataset.addColumn(new_column)

##
##

```

```

## PART-III
##
##
## The probability of a molecule to adopt a certain conformation may be
## calculated from a normalized Boltzmann distribution as:
##
##  $P_j = \exp(-\text{Energy\_delta}_j/RT) / \text{SIGMA} \exp(-\text{Energy\_delta}_j/RT)$ 
##
## where  $_j$  stands for the j-th term
##
## where Energy_delta_j is the relative steric energy of the j-th
## conformation (J/mol)
## R is the gas constant (8.314 J/mol/K)
## T is the temperature in Kelvin
## N is the number of conformations with Energy_delta_j <= (le)
## 2.5 kcal/mol
##
##
## Reference:
## Correlation of Drug Absorption with Molecular Surface Properties
## Palm K et. al.
## J. Pharm. Sci. 85, 1, Jan 1996, pp. 32-39
##
##
## Appends 2 columns to dataset - 'E_exp-sum', 'p-value'
##
##
##
##
## Excluding 'Energy_delta_j' > 2.5 kcal/mol
## These shouldn't be taken into account in the computations;
## this implies conformers that have higher energy than that can exist ## at 300 K are ignored
## from the computation of p-value OR
## in other words, conformers that deviate from the minimum energy
## conformer by more than a certain energy amount, that cannot come
## from thermal energy at 300 K, are disregarded from the computations
##
Probability = [-100] * total ## ensure not to initiate to a value b/w 0 and 1
Energy_exp_summated = [-10000] * total ## ensure not to initiate to 0

for i in range(stop):
    print "For Conformer-ID:", i+1, "Summating E_exp term"
    Energy_exp_sum = 0
    for j in range(total): # go through ALL data looking for particular conformer-ID
        if (dataset[mseq_col][j] == i+1): # if match happens
            if (Energy_delta[j] <= 2.5): # energy difference is in acceptable limits
                Energy_exp_sum = Energy_exp_sum + Energy_exp_term[j]
            else: pass

    for j in range(total): # go through ALL data looking for particular conformer-ID
        if (dataset[mseq_col][j] == i+1): # if match happens
            if (Energy_delta[j] <= 2.5): # energy difference is in acceptable limits
                Energy_exp_summated[j] = Energy_exp_sum

```

```
else: Energy_exp_summated[j] = -11111111 # some arbitrary value set
as a flag

## APPEND COLUMN
new_column=createFloatColumn("E_exp-sum", Energy_exp_summated)
dataset.addColumn(new_column)

##
## COMPUTE PROBABILITY
##
## see formula in notes above
##
## Don't need to check for "mseq" tag (conformer-ID) now as
## 'Energy_Delta', 'Energy_exp_term', 'Energy_exp_summated' have all
## been previously computed checking for "mseq" tags (conformer-IDs)
##

for i in range(total):
    if (i%100 == 0):
        print "For row:", i, "Computing probability"
    if (Energy_delta[i] <= 2.5): # energy difference is in acceptable limits
        Probability[i] = Energy_exp_term[i]/Energy_exp_summated[i]
    else: Probability[i] = 0

## APPEND COLUMNS
new_column=createFloatColumn("p-value", Probability)
dataset.addColumn(new_column)

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

##
## END
##
```

End of Document