

Average Descriptors

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 takes off from where the Boltzmann-probability script ends [qsarworld]. It uses the p-values computed by the Boltzmann-probability script for weighted averaging of descriptors.

As an input, it requires a column ('p-value') that is needed for weighted averaging of descriptors. This 'p-value' column is used as the 'weights'. It also prompts the user to provide the set of descriptors that need to be averaged. The unselected descriptors are left *as is* (and not averaged by using the p-value). For the unselected descriptors, the first value of the set of conformers is taken as the representative value in the 'output' dataset.

Specifically, the script prompts for 3 sets of columns:

- a) columns that need to be scaled
- b) column with which to scale (p-value)
- c) conformer-ID column

Ensure "Attribute Type" is 'categorical' for 'conformer-ID' column for certain scripting assumptions to work.

Output is a dataset named as "Averaged-Descriptors" that contains un-averaged descriptors (for unselected columns) and weight-averaged descriptors (for selected columns). The naming of the averaged columns reflects the fact that they were scaled by some given column.

References:

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

Cite this as:

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

```
##
##
## sarchitect designer 2.2 script for 'weighted averaging' of
## descriptors
##
## Shailay Kumar Dogra
## editor@qsarworld.com
## 10 Oct 2006
##
## Requires a column ('p-value') that is needed for weighted averaging
## of descriptors. This 'p-value' column is the 'weights'. Requires
## the user to provide the set of descriptors that need to be scaled ## (by p-value). The
## remaining descriptors are left as is (and not
## scaled by any value). First value of the set of conformers is taken
## as the representative value in the 'output' dataset.
##
##
## Asks for following columns as inputs:
##     -> set of columns that need to be scaled,
##     -> column that has the values with which to scale (p-value),
##     -> conformer-ID column.
##
##
## Ensure "Attribute Type" is 'categorical' for 'conformer-ID' column
## (else "dataset[mseq_col].getRowIndicesOfCategory(catIdx)" will
## fail)
##
##
## Output is a dataset (named "Averaged-Descriptors")
## containing unaveraged descriptors (for unselected columns)
## and weight-averaged descriptors (for selected columns).
##
##
##
```

```
import script
from script.dataset import *
from script.algorithm import *
from script.project import *
from script.omega import createComponent, showDialog
from javax.swing import *
from math import *
from java.lang import Boolean
from com.strandgenomics.cube.dataset import ColumnFactory
from com.strandgenomics.chem.project import ChemProject
```

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

```

##-----
## PANEL TO GET DESCRIPTOR-COLUMN SELECTION
def getInputColumns():
    helptext=""
    To weight-average descriptors, select desired columns.
    Ensure selecting numerical columns only!
    Also, don't select the weights (p-value) column.
    ""
    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 Descriptors for Averaging",
description="Set of Descriptors",components=[helpPanel,colPanel])
    result=showDialog(finalPanel)
    return result['columns']

##-----
## PANEL TO GET p-value COLUMN
def getPcolumn():
    helptext=""
    Provide the weights (p-value) column:
    ""
    helpPanel=createComponent(type="ui",id="help",
description="",component=textarea(helptext))
    colPanel = createComponent(type="column", id="pvalue", description="Select p-
value",dataset=script.project.getActiveDataset())
    finalPanel= createComponent(type="group",id="Provide p-value column", description="p-
value column?",components=[helpPanel,colPanel])
    result=showDialog(finalPanel)
    return result['pvalue']

##-----
## PANEL TO GET mseq COLUMN
def getMseqcolumn():
    helptext=""
    Provide the conformer-ID column:
    ""
    helpPanel=createComponent(type="ui",id="help",
description="",component=textarea(helptext))
    colPanel = createComponent(type="column", id="mseq", description="Select Conformer-
ID column",dataset=script.project.getActiveDataset())
    finalPanel= createComponent(type="group",id="Provide Conformer-ID column",
description="Conformer-ID column?",components=[helpPanel,colPanel])
    result=showDialog(finalPanel)
    return result['mseq']

##-----
## GET INDICES OF NOT-SELECTED COLUMNS
def notSelectedColumns(columns):
    selected = [ ]
    for i in range(columns.getSize()):
        selected.append(columns.get(i))
        #print columns.get(i), ":", dataset[columns.get(i)].getName()
    #print selected

```

```

list      ## mseq, p-value get selected in addition to 'columns'; need to add p-value to 'selected'

      ## for mseq, no need as we need its (unaveraged) entry unlike p-value
      ## 'pvalue' already got using "getPcolumn()" before this method gets called

      selected.append(pvalue)

      total = range(dataset.getColumnCount())
      for i in selected:
          total.remove(i)
      notselected = total
      return notselected

##-----

## MAIN

dataset=script.project.getActiveDataset()

## GET REQUIRED COLUMNS

columns=getInputColumns()
#for i in range(columns.getSize()):
#    print columns.get(i), ":", dataset[columns.get(i)].getName()

pvalue=getPcolumn()

mseq_col = getMseqcolumn()

ColsNotSelected = notSelectedColumns(columns) ## notSelectedColumns defined above

total_ids = dataset[mseq_col].getCategoryCount()
uniqcol = [ ]

for idx in ColsNotSelected:
    uniqrow = [ ]
    for catIdx in range(total_ids):
        row_set = dataset[mseq_col].getRowIndicesOfCategory(catIdx)
        row_id = min(row_set)
        uniqrow.append(dataset[idx][row_id])

    if (dataset[idx].getDatatype()=='string'): ## 'Id' column in 'unselected columns'
        addCol = ColumnFactory.createStringColumn(dataset[idx].getName(), uniqrow)
        (addCol.getMetaData()).setMark('Identifier')
        uniqcol.append(addCol)
    elif (dataset[idx].getDatatype()=='object'): ## 'Structure' column in 'unselected columns'
        addCol = ColumnFactory.createObjectColumn(dataset[idx].getName(), uniqrow)
        (addCol.getMetaData()).setMark(ChemProject.RAWSTRUCTURE)
        uniqcol.append(addCol)

```

```

        else:
            uniqcol.append(script.dataset.createFloatColumn(dataset[idx].getName(),
            uniqrow))

    unscaledset = uniqcol

## SCALE COLUMNS BY p-value
scaledColumns = [ ]
for i in range(columns.getSize()):
    scaledColumns.append(dataset[columns.get(i)] * dataset[pvalue])

total_cols = len(scaledColumns)
averagedset = [ ]

for colldx in range(total_cols):

    averagedcol = [ ]
    for catldx in range(total_ids):
        row_set = dataset[mseq_col].getRowIndicesOfCategory(catldx)

        avg_rows = 0
        for i in row_set:
            #print i+1, scaledColumns[colldx][i]
            avg_rows = avg_rows + scaledColumns[colldx][i]    ## for a set of
conformers

        #print catldx, avg_rows
        averagedcol.append(avg_rows)    ## append averaged set of conformers in a
column

    averagedset.append(script.dataset.createFloatColumn(scaledColumns[colldx].getName()
, averagedcol))    ## append columns with averaged rows to get dataset

#print averagedset

## DEFINE A NEW CHILD-DATASET
rowIndices = [i for i in range(total_ids)]
colIndices = [ ]
script.project.addSubsetChild(rowIndices,    colIndices    ,    name="Averaged-Descriptors",
addMarkedColumns=0)
subset=script.project.getActiveDataset()

for i in range(len(unscaledset)):
    addcol = unscaledset[i]
    subset.addColumn(addcol)

for i in range(len(averagedset)):
    addcol = averagedset[i]
    subset.addColumn(addcol)

```

```
script.view.Table().show()
```

```
## REPORT COMPLETION
```

```
parent=script.tool.getTool().getFrame()
```

```
mesg = "Done With Script Execution."
```

```
JOptionPane.showMessageDialog(parent,mesg,"STATUS!",JOptionPane.INFORMATION_MESSAGE)
```

End of Document