2_Example_Accessing_4GL

August 18, 2018

1 Accessing FAME 4GL from Python

This example accesses FAME 4GL functionality from Python.

```
In [1]: import os
        import sys
        from __future__ import print_function

        import pandas as pd
        from pyhli import *
        import qoma_smuggler as qm
```

The Qoma utility function `open_hli()` opens the FAME environment and prints diagnostic information.

Below we use lower level FAME HLI functions. The `qomautils` package offers higher level composites of FAME HLI functions.
The `qomautils` package function `open_hli()` calls `cfmini()` and `cfmver()` to initialize the FAME environment and to obtain FAME HLI version information.

```
In [2]: if qm.open_hli()!=0:
            raise
```

```
Linux 4.9.0-4-amd64 (#1 SMP Debian 4.9.65-3+deb9u1 (2017-12-23)) x86_64
Python 3.6.5 | packaged by conda-forge | (default, Apr  6 2018, 13:39:56)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-15)]
NumPy 1.13.3 Pandas 0.23.4 FAME HLI 11.63000 pyhli 0.0.11 qoma-smuggler 0.0.2
```

FAME 4GL commands are easily passed to a FAME server using the FAME HLI function `cfmfame()`. Here we: * set a date range to the prior quarter (`date thisday(q)-1`) * set the frequency to BUSINESS (`freq b`) * open a data base, instructing FAME to overwrite any existing database named `tmp.db` (`open<acc over>tmp`) * create a time series x, specifying `double` storage (`series x : precision by date`) * update the object's decription attribute (`desc(x) = "U[0,1]"`) * update the object's documentation (`docu(x) = "..."`) * use the FAME 4GL function `uniform()` to obtain samples from the uniform distribution indexed by date * direct FAME 4GL output to a temporary file, overwriting old file if needed (`output<acc over>tmp.txt`) * set report

orientation to VERTICAL (time will appear vertically) * set report length to FULL (affects report pagination) * permit automatic time scale conversion (`conv on`) * request reports at `BUSINESS`, `WEEKLY(FRIDAY)`, `MONTHLY`, and `QUARTERLY` frequency. * FAME uses the object's OBSERVED attribute to properly reduce business data to lower frequency data. * instruct the FAME 4GL to close the output file (by directing output to terminal) * instruct FAME to close the database

Upon return to Python, we: * confirm the FAME HLI function `cfmfame()` returns code `HSUCC` (success) * display the temporary text file `tmp.txt`

```
In [3]: cmd = ['\
          date thisday(q)-1; \
          freq b; \
          open<acc over>tmp; \
          series x : precision by date; \
          desc(x) = "U[0,1]"; \
          docu(x) = "Uniformly distributed U[0,1] time-series."; \
          set x = uniform(date); \
          output<acc over>tmp.txt; \
          whats x; \
          show vert; length full; conv on;\
          freq b; title text "Frequency "+@freq; repo x; \
          freq w(fri); title text "Frequency "+@freq; repo x; \
          freq m; title text "Frequency "+@freq; repo x; \
          freq q; title text "Frequency "+@freq; repo x; \
          output terminal; \
          close tmp\
          ']
        cfmfame ([-1], cmd)
        qm.print_file('tmp.txt')
```

```
                                    X

                                 U[0,1]

Class:  SERIES                                  DB name:  TMP
Type:   PRECISION                               Created:  18-Aug-18
Index:  DATE:BUSINESS                           Updated:  18-Aug-18

First Value at:  2-Apr-18                       Observed: SUMMED
Last Value at:  29-Jun-18                       Basis:    BUSINESS

Uniformly distributed U[0,1] time-series.



Frequency BUSINESS
```

	X
2-Apr-18	0.04
3-Apr-18	0.41
4-Apr-18	0.32
5-Apr-18	0.68
6-Apr-18	0.27
9-Apr-18	1.00
10-Apr-18	0.81
11-Apr-18	0.03
12-Apr-18	0.63
13-Apr-18	0.98
16-Apr-18	0.85
17-Apr-18	0.74
18-Apr-18	0.75
19-Apr-18	0.34
20-Apr-18	0.96
23-Apr-18	0.83
24-Apr-18	0.62
25-Apr-18	0.45
26-Apr-18	0.13
27-Apr-18	0.94
30-Apr-18	0.34
1-May-18	0.40
2-May-18	0.77
3-May-18	0.49
4-May-18	0.13
7-May-18	0.03
8-May-18	0.79
9-May-18	0.20
10-May-18	0.24
11-May-18	0.02
14-May-18	0.08
15-May-18	0.11
16-May-18	0.88
17-May-18	0.96
18-May-18	0.78
21-May-18	0.73
22-May-18	0.65
23-May-18	0.16
24-May-18	0.66
25-May-18	0.53
28-May-18	0.74
29-May-18	0.16
30-May-18	0.56
31-May-18	0.42

1-Jun-18	0.57
4-Jun-18	0.46
5-Jun-18	0.76
6-Jun-18	0.69
7-Jun-18	0.36
8-Jun-18	0.57
11-Jun-18	0.09
12-Jun-18	0.20
13-Jun-18	0.95
14-Jun-18	0.81
15-Jun-18	0.96
18-Jun-18	0.97
19-Jun-18	0.50
20-Jun-18	0.59
21-Jun-18	0.96
22-Jun-18	0.06
25-Jun-18	0.63
26-Jun-18	0.10
27-Jun-18	0.25
28-Jun-18	0.15
29-Jun-18	0.89

Frequency WEEKLY(FRIDAY)

	X
13-Apr-18	3.45
20-Apr-18	3.63
27-Apr-18	2.97
4-May-18	2.13
11-May-18	1.28
18-May-18	2.81
25-May-18	2.74
1-Jun-18	2.45
8-Jun-18	2.84
15-Jun-18	3.00
22-Jun-18	3.09
29-Jun-18	2.01

```
Frequency MONTHLY


            X
         -----

Apr 18   12.11
May 18   10.49
Jun 18   11.51
```

```
Frequency QUARTERLY


           X
       -----

18:2   34.11
```

The `qomautils` function `read_fame()` reads FAME data objects into a nested Python dictionary. At the top level, each FAME object name is mapped to a dictionary with entries `data` and `fame`. * As appropriate, the entry `data` maps to a single data value or to multiple data values in an array. * For FAME `SCALAR` objects, `data` maps to a value. * For FAME `SERIES` objects, `data` maps to an array of values. * The entry `fame` maps to FAME object meta data such as object class (`SCALAR` or `SERIES`), object data type, and index values for `data`.

We will use the `tmp` database constructed above.

```
In [4]: dbname = "tmp"
        famedata = qm.read_fame(dbname)
        print("read_fame() returned {0} FAME objects from {1}.\n".format(len(famedata),dbname))
```

```
read_fame() returned 1 FAME objects from tmp.
```

Once FAME data objects are loaded to a Python dictionary, it is easy to access information. First, a peek at the top level contents of the dictionary for the FAME data object `X` we constructed earlier in this notebook.

```
In [5]: x = famedata.get('X')
        print("data:\n{0}\n".format(x.get('data')))
        print("fame:\n{0}\n".format(x.get('fame')))
```

```
data:
[0.04464869573712349, 0.4106217622756958, 0.32020440697669983, 0.6753296852111816, 0.266079425
```

```
fame:
{'desc': 'U[0,1]', 'docu': 'Uniformly distributed U[0,1] time-series.', 'range': [9, 43895, 43
```

The meta data contained in the `fame` Python dictionary specifies things such as the date range of the `data`.

```
In [6]: meta = x.get('fame')
        rng = meta.get('range')
        print("pandas range:\n{0}\n".format(qm.to_pandas_range(rng)))
        print("description     {0}".format(meta.get('desc')))
        print("documentation   {0}".format(meta.get('docu')))
        print("fame class      {0}".format(qm.class_to_string(meta.get('class'))))
        print("fame data type  {0}\n".format(qm.type_to_string(meta.get('type'))))
```

```
pandas range:
DatetimeIndex(['2018-04-02', '2018-04-03', '2018-04-04', '2018-04-05',
               '2018-04-06', '2018-04-09', '2018-04-10', '2018-04-11',
               '2018-04-12', '2018-04-13', '2018-04-16', '2018-04-17',
               '2018-04-18', '2018-04-19', '2018-04-20', '2018-04-23',
               '2018-04-24', '2018-04-25', '2018-04-26', '2018-04-27',
               '2018-04-30', '2018-05-01', '2018-05-02', '2018-05-03',
               '2018-05-04', '2018-05-07', '2018-05-08', '2018-05-09',
               '2018-05-10', '2018-05-11', '2018-05-14', '2018-05-15',
               '2018-05-16', '2018-05-17', '2018-05-18', '2018-05-21',
               '2018-05-22', '2018-05-23', '2018-05-24', '2018-05-25',
               '2018-05-28', '2018-05-29', '2018-05-30', '2018-05-31',
               '2018-06-01', '2018-06-04', '2018-06-05', '2018-06-06',
               '2018-06-07', '2018-06-08', '2018-06-11', '2018-06-12',
               '2018-06-13', '2018-06-14', '2018-06-15', '2018-06-18',
               '2018-06-19', '2018-06-20', '2018-06-21', '2018-06-22',
               '2018-06-25', '2018-06-26', '2018-06-27', '2018-06-28',
               '2018-06-29'],
              dtype='datetime64[ns]', freq='B')

description     U[0,1]
documentation   Uniformly distributed U[0,1] time-series.
fame class      SERIES
fame data type  PRECISION
```

```
In [7]: print(qm.meta_to_string(famedata,'X'))
        qm.get(famedata,'X')

SERIES X : PRECISION BY DATE(BUSINESS) 2Apr2018 to 29Jun2018
U[0,1]
-- documentation:
Uniformly distributed U[0,1] time-series.
--
```

```
Out[7]: 2018-04-02    0.044649
        2018-04-03    0.410622
        2018-04-04    0.320204
        2018-04-05    0.675330
        2018-04-06    0.266079
        2018-04-09    0.996716
        2018-04-10    0.806511
        2018-04-11    0.028181
        2018-04-12    0.633366
        2018-04-13    0.983093
        2018-04-16    0.848262
        2018-04-17    0.735095
        2018-04-18    0.746435
        2018-04-19    0.337060
        2018-04-20    0.959828
        2018-04-23    0.828680
        2018-04-24    0.624528
        2018-04-25    0.448333
        2018-04-26    0.130775
        2018-04-27    0.937249
        2018-04-30    0.344642
        2018-05-01    0.400653
        2018-05-02    0.769589
        2018-05-03    0.489387
        2018-05-04    0.127211
        2018-05-07    0.030986
        2018-05-08    0.789088
        2018-05-09    0.199574
        2018-05-10    0.237343
        2018-05-11    0.021365
                        ...
        2018-05-21    0.733900
        2018-05-22    0.654142
        2018-05-23    0.161222
        2018-05-24    0.662791
        2018-05-25    0.525599
        2018-05-28    0.737679
```

```
2018-05-29    0.163477
2018-05-30    0.557531
2018-05-31    0.419681
2018-06-01    0.570980
2018-06-04    0.455391
2018-06-05    0.760141
2018-06-06    0.691697
2018-06-07    0.356612
2018-06-08    0.572802
2018-06-11    0.085750
2018-06-12    0.200033
2018-06-13    0.954174
2018-06-14    0.806625
2018-06-15    0.955017
2018-06-18    0.973731
2018-06-19    0.502207
2018-06-20    0.593441
2018-06-21    0.963412
2018-06-22    0.060191
2018-06-25    0.627482
2018-06-26    0.095392
2018-06-27    0.245204
2018-06-28    0.146123
2018-06-29    0.892807
Freq: B, Name: X, Length: 65, dtype: float64
```

In [8]:
```
if qm.close_hli()!=0:
    raise
```

In [9]:
```
os.remove("tmp.txt")
os.remove("tmp.db")
```