MetaOS R integration. A how-to guide.

Luis F. Canals (luisf.canals@gmail.com)

November 3, 2023

1 Abstract

2 MetaOS viewed from Python scripts.

The following example shows how Python scripts invokes a R code in the simplest way: load of R environment and R code, inizialization of R variables and objects, example of loop to call several times to a function R in the R-environment and new call at the end of the loop:

```
interpreteR = R(["correlation.r"])

interpreteR.eval("corre<-correlator()")
for i in range(1,200):
    interpreteR.eval("corre$memo(" + str(i) + "," + str(i) + ")")

print interpreteR.eval("corre$show()")

interpreteR.end()
```

As it's seen from the code, a R source code named *correlation.r* where a class *correlator* is defined. The class should have got the methods *memo* and *show*, as in this example:

```
correlator <- function() {
    xVals <- c()
    yVals <- c()

    memo <- function(x, y) {
        xVals <<- union(xVals, x)
        yVals <<- union(yVals, y)
    }


    show <- function() {
        r <- lm(xVals ~ yVals)
        return(r)
    }

    return(list(memo=memo, show=show))
}
```

3 An example

The following code (*rintegration.py*) uses the same described principle in the previous section but letting the name of R source containing the class as a runtime parameter:

```
#
# Predictor using R defined classes over BRIC40.
#

# TO DO: get symbols from file source
symbols = [ '1288.HK', '3988.HK', '0883.HK', '0939.HK', '2628.HK', '3968.HK', '0941.HK', '0688.HK', '0386.HK'

symbol1 = symbols[16]
symbol2 = symbols[0]

source = CSVUnorderedData.getInstance().reuters('BRIC40_1min.csv', symbols)

# R code: create predictor object
interpreteR = R([args[0]])
interpreteR.eval("predictor <- lsPredictor()")

# Bind R predictor to source events through an observer
class MyObserver(MarketObserver):
    def __init__(self):
        self.anyCoincidence = False;

    def update(self, ss, when):
        if symbol1 in ss and symbol2 in ss:
            self.anyCoincidence = True
            interpreteR.eval('predictor$learn(' \
                + str(market.getLastPrice(0, symbol1 + '-CLOSE')) + ','
                + str(market.getLastPrice(0, symbol2 + '-CLOSE')) + ')')

            strLine = Long.toString(when.getTimeInMillis()).encode('utf-8')
            strLine = strLine + ',' \
                    + Double.toString(market.getLastPrice(0,symbol1 + '-OPEN'))\
                            .encode('utf-8') + ','\
                    + Double.toString(market.getLastPrice(0,symbol1 + '-HIGH'))\
                            .encode('utf-8') + ',' \
                    + Double.toString(market.getLastPrice(0,symbol1 + '-LOW'))\
                            .encode('utf-8') + ','\
                    + Double.toString(market.getLastPrice(0,symbol1+'-CLOSE'))\
                            .encode('utf-8') + ','\
                    + Long.toString(market.getLastVolume(0,symbol1))\
                            .encode('utf-8')
            strLine = strLine + ',' \
                    + Double.toString(market.getLastPrice(0,symbol2 + '-OPEN'))\
                            .encode('utf-8') + ','\
                    + Double.toString(market.getLastPrice(0,symbol2 + '-HIGH'))\
                            .encode('utf-8') + ',' \
                    + Double.toString(market.getLastPrice(0,symbol2 + '-LOW'))\
                            .encode('utf-8') + ','\
                    + Double.toString(market.getLastPrice(0,symbol2+'-CLOSE'))\
                            .encode('utf-8') + ','\
                    + Long.toString(market.getLastVolume(0,symbol2))\
                            .encode('utf-8')
            print strLine

        if symbol1 in ss and not symbol2 in ss and self.anyCoincidence:
            y = interpreteR.evalDouble('predictor$predict(' \
                + str(market.getLastPrice(0, symbol1 + '-CLOSE')) + ')')

            strLine = Long.toString(when.getTimeInMillis()).encode('utf-8')
            strLine = strLine + ',' \
                    + Double.toString(market.getLastPrice(0,symbol1 + '-OPEN'))\
                            .encode('utf-8') + ','\
                    + Double.toString(market.getLastPrice(0,symbol1 + '-HIGH'))\
                            .encode('utf-8') + ',' \
                    + Double.toString(market.getLastPrice(0,symbol1 + '-LOW'))\
                            .encode('utf-8') + ','\
                    + Double.toString(market.getLastPrice(0,symbol1+'-CLOSE'))\
                            .encode('utf-8') + ','\
                    + Long.toString(market.getLastVolume(0,symbol1))\
                            .encode('utf-8')
            strLine = strLine + ',-,-,-,' + str(y) + ',-'
            print strLine


# Join everything together
market = SequentialAccessMarket(0.0, 5000)
source.addMarketListener(market)
source.addListener(MyObserver())

# Ready, steady, go
print 'milliseconds,' + symbol1 + '-OPEN,' + symbol1 + '-HIGH,' \
    + symbol1 + '-LOW,' + symbol1 + '-CLOSE,' + symbol1 + '-VOLUME,' \
    + symbol2 + '-OPEN,' + symbol2 + '-HIGH,' \
    + symbol2 + '-LOW,' + symbol2 + '-CLOSE,' + symbol2 + '-VOLUME'
```

```
source.run()

# Land down
interpreteR.end()
```

In this case, interface for R class has been modified, to create a simple predictor with two methods, *learn* and *predict*. An example of predictor based on linear regression might look like this:

From the example, we write down the interface all R-class should satisfy to be compatible with *rintegration.py* requirements (in pseudocode Java-like):

```
interface PredictorInR {
    // Learns a new association x->y
    void learn(double x, double y);

    // Tries to predict the value of y from x
    double predict(double x);
}
```

4 Usage from command line and visual interface

The code in *rintegration.py* can be invoked even from command line or from visual interface. In both cases, file acting as a source of prices, the main Python script (*rintegration.py* in this case) and the R file containing the source code of the class with the predictor following the interface described in the interface 3.

5 License terms