Econophysics

Juan Camilo Henao Londono

Universitaet Duisburg-Essen, Duisburg, Germany

Abstract

Document to set some rules to name files, folders, python functions and jupyter notebooks in the [Find a good name for the econophysics project]. Basically, the style guide for code.

Keywords: Econophysics, Python, Jupyter, [Project name]

1. Files and folders names

In general all the files and folder will be created inside of one of three possible folders

```
/data_name_kind[_year]/
```

In this case data_name can be itch or taq, kind can be algorithms, data or plot, and year is the year of the data. Year is optional as algorithm is a general folder and do not depend of the year. Then, Examples of folders names are

```
/taq_data_2008/
/itch_algorithms
```

When a folder is created by a function, its name must follow the following sintaxis

```
/function_name[_time_step]/
```

In this case, function_name is the name of the function that creates the folder and time_step is the time step taken to do the analysis. time_step is optional as it is only used in the ITCH data. Examples of folders names are

```
/taq_midpoint_data/
/itch_cross_response_data_1ms/
```

Finally, when a file is created by a function, it name must follow the following sintaxis

```
/function_name_year_month_day_ticker(s)[_time_step].ext
```

In this case, function name is the name of the function that creates the file; year, month and day is the date of the data, ticker is the stock analyzed, time_step is the time step taken to do the analysis and ext is the extension of the file. In ticker can be used one or two tickers depending on the function. time_step is optional as it is only used in the ITCH data. For functions that save more than one file, it must be used the name of the variable saved after the function name. Examples of files names are

```
/itch_cross_response_data_20160307_aapl_msft_10ms.pickle
/taq_midpoint_data_ask_20080310_aapl.pickle
/itch_self_response_data_20160307_aapl_1ms.pickle
```

2. Functions

2.1. Names

The functions take the name of the data and a name related with its functionality

```
data_name_function_name_type(params)
```

In this case data_name can be itch or taq, function_name is the name related with its functionality and type can be data or plot depending on what is the result of the function

```
taq_midpoint_data(ticker, year, month, day)
```

2.2. Parameters

The order of the parameters must be

```
params -> ticker, year, month, day [,tau_val, tau_step]
```

When only one ticker is used in the function, the parameter must be "ticker". If two tickers are used in the function, they must be named "ticker_i" and "ticker_j". tau_val and tau_step are optional as they are used only with the ITCH data.

```
itch_cross_response_data(ticker_i, ticker_j, year, month, day, tau_val, t_step)

itch_zero_correlation_model_data(ticker, year, month, day, tau_val, t_step)
```

2.3. Body

The body of the functions must follow the following order

```
def function_name():
    '''Docstrings: description and parameters explanation'''
    # Opening message
    print('TAQ / ITCH data')
    print('Function name')
    print('Processing data for the stock' + ticker + 'the ' + year + '.'
          + month + '.' + day)
    # or
    print('Processing data for the stock i ' + ticker_i + 'and stock j '
           + ticker_j + 'the ' + year + '.' + month + '.' + day)
    print('Time step: ' + t_step + 'ms')
    # Body
    # Saving data
```

Examples of functions with one or two stocks are

```
def cross_response_data(ticker_i, ticker_j, day, tau_val, t_step):
    """
    Obtain the cross response function using the midpoint log returns of
    ticker i and trade signs of ticker j during different time lags. The data
    is adjusted to use only the values each t_step ms
        :param ticker_i: string of the abbreviation of the midpoint stock to
         be analized (i.e. 'AAPL')
        :param ticker_j: string of the abbreviation of the trade sign stock to
         be analized (i.e. 'AAPL')
        :param year: string of the year to be analized (i.e '2008')
        :param month: string of the month to be analized (i.e '07')
```

```
        :param day: string of the day to be analized (i.e '07')
        :param tau_val: maximum time lag to be analyzed
        :param t_step: time step in the data in ms
    """

    if (ticker_i == ticker_j):

        return None

    else:

        print('ITCH data')
        print('Cross response function data')
        print('Processing data for the stock i ' + ticker_i + ' and stock j ' +
            ticker_j + ' the ' + year + '.' + month + '.' + day)
        print('Time step: ', t_step, 'ms')

        # Body

        return None

def zero_correlation_model_data(ticker, year, month, day, tau_val, t_step):
    """
    Obtain the cross response function using the midpoint log return of
    ticker i and random trade signs during different time lags. The data is
    adjusted to use only the values each t_step ms
        :param ticker: string of the abbreviation of the midpoint stock to
         be analized (i.e. 'AAPL')
        :param day: string of the day to be analized (i.e '07')
        :param tau_val: maximum time lag to be analyzed
        :param t_step: time step in the data in ms
    """

    print('ITCH data')
    print('Zero correlation model data')
    print('Processing data for the stock ' + ticker + ' and a random'
          + ' trade sign array the ' + year + '.' + month + '.' + day)
    print('Time step: ', t_step, 'ms')

    # Body

    return None
```

In general the code must follow the PEP8 – Style guide for Python code.