Python 金融工程套件 QuantLib 介紹

衍生商品評價與風險計算

Python Taipei

2019/1/24

希奇資本 技術長

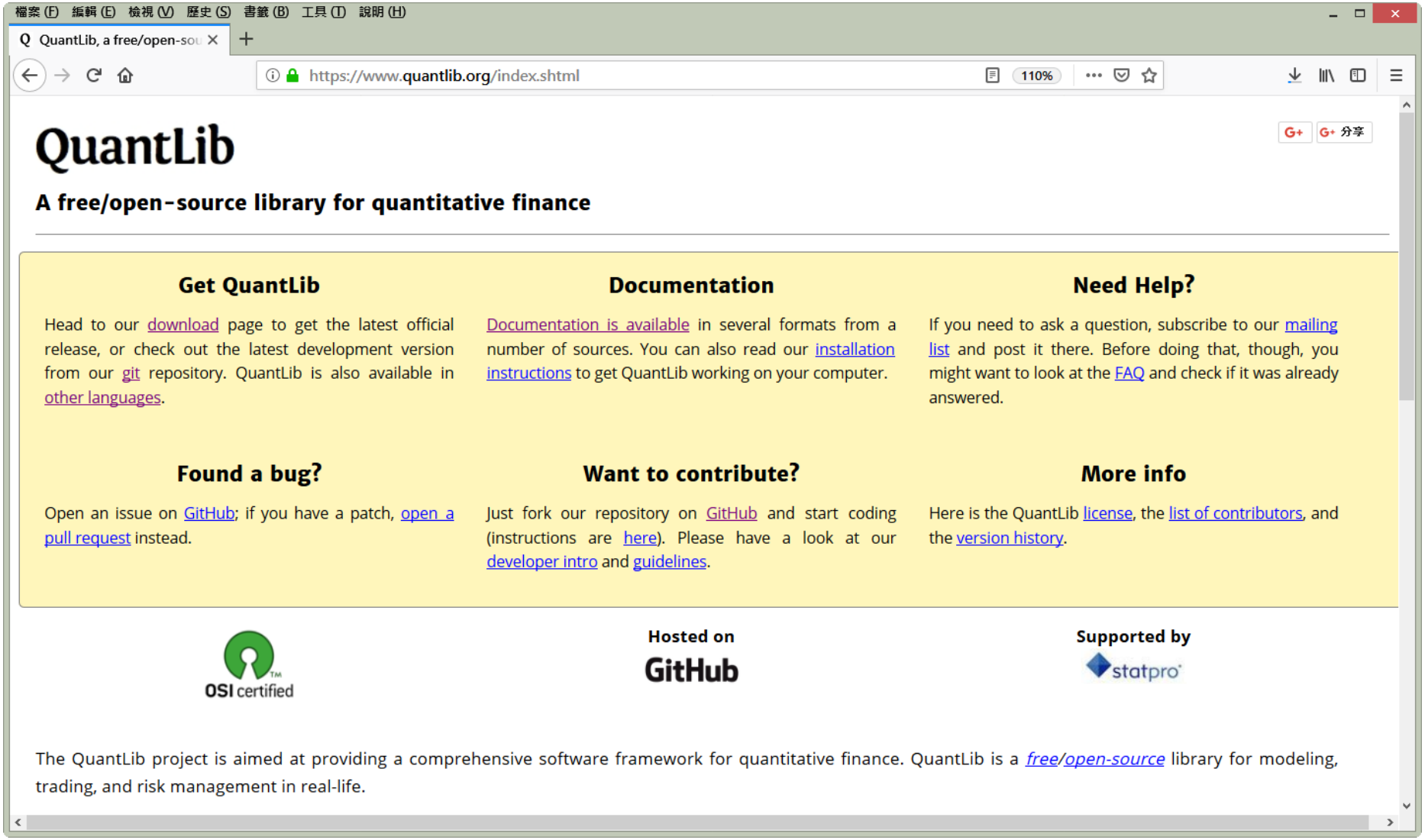
董夢雲 博士

目　錄

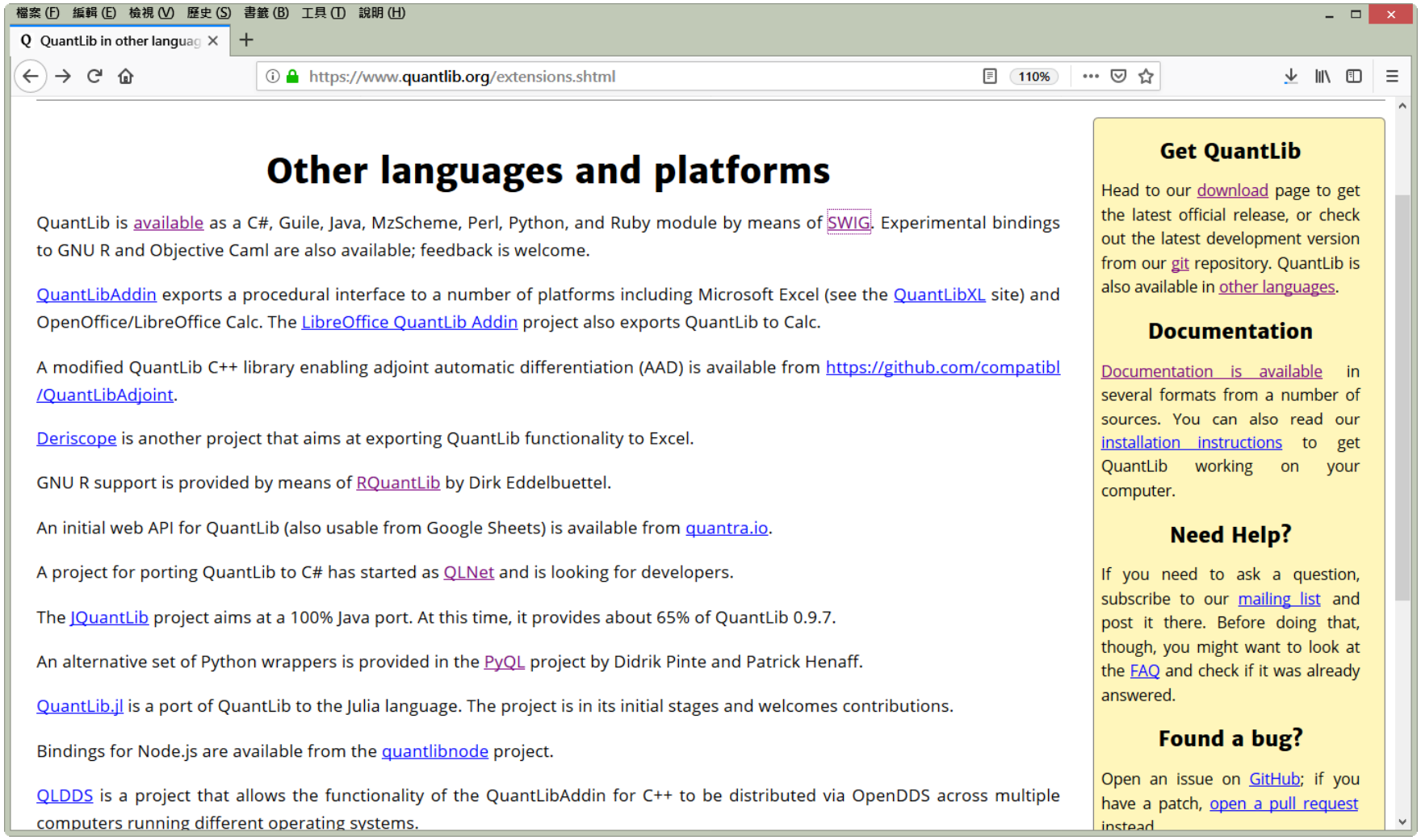
零、QuantLib 與 QuantLib-Python 安裝

(一)QuantLib 程式庫的由來

- QuantLib 是一個開放源碼軟體程式庫(Open-Source Software Library)。
  - 目的在提供對金融工具評價和相關主題有興趣的軟體開發人員，適合的開發工具。
  - 可在 QuantLib 的網站上，https://www.quantlib.org，了解專案的相關內容，2018/09/19。


檔案 (F) 編輯 (E) 檢視 (V) 歷史 (S) 書籤 (B) 工具 (T) 說明 (H)
QuantLib, a free/open-sou
https://www.quantlib.org/index.shtml
110%
QuantLib
A free/open-source library for quantitative finance
Get QuantLib
Head to our download page to get the latest official release, or check out the latest development version from our git repository. QuantLib is also available in other languages.
Documentation
Documentation is available in several formats from a number of sources. You can also read our installation instructions to get QuantLib working on your computer.
Need Help?
If you need to ask a question, subscribe to our mailing list and post it there. Before doing that, though, you might want to look at the FAQ and check if it was already answered.
Found a bug?
Open an issue on GitHub; if you have a patch, open a pull request instead.
Want to contribute?
Just fork our repository on GitHub and start coding (instructions are here). Please have a look at our developer intro and guidelines.
More info
Here is the QuantLib license, the list of contributors, and the version history.
OSI certified
Hosted on
GitHub
Supported by
statpro
The QuantLib project is aimed at providing a comprehensive software framework for quantitative finance. QuantLib is a free/open-source library for modeling, trading, and risk management in real-life.

- QuantLib 是以 C++語言開發的程式庫，以此為基礎被轉寫成其他不同的語言。
  - 包括 C#、Java、Perl、Python、R、Ruby 與 Scheme。



(二)QuantLib 的歷史

- 由一群在 Cabota Banca Intesa 的利率衍生商品交易台工作的數量分析專家們，於西元 2000 年時所開創的一個財務金融程式庫開發專案。
  - 目前這些專家們已經成立了一家公司，該公司之前稱之為 RiskMap，現在則命名為 StatPro Italia
- QuantLib 專案目前是由 Luigi Ballabio 與 Ferdinando Ametrano 兩位專家所領導
  - 當他們在實作 Black-Scholes 公式的時候，Ferdinando Ametrano 分享他在另一個結構化專案中的開發經驗。
  - 該專案允許一種新的合作方式，專案中的任何人可以改進、更正與開發一個免費的共用基礎框架 (Free Common-base Framework)。
    - ✓ 此一意見為 RiskMap 所支持並加以採用。
  - QuantLib 的創建者秉持著這樣的信念，公開的標準最適合於科學與技術的演進。
    - ✓ 尤其在學術界之內，好的實務技能與工具只有被教育界所接受，方可在長期的演化中勝出。

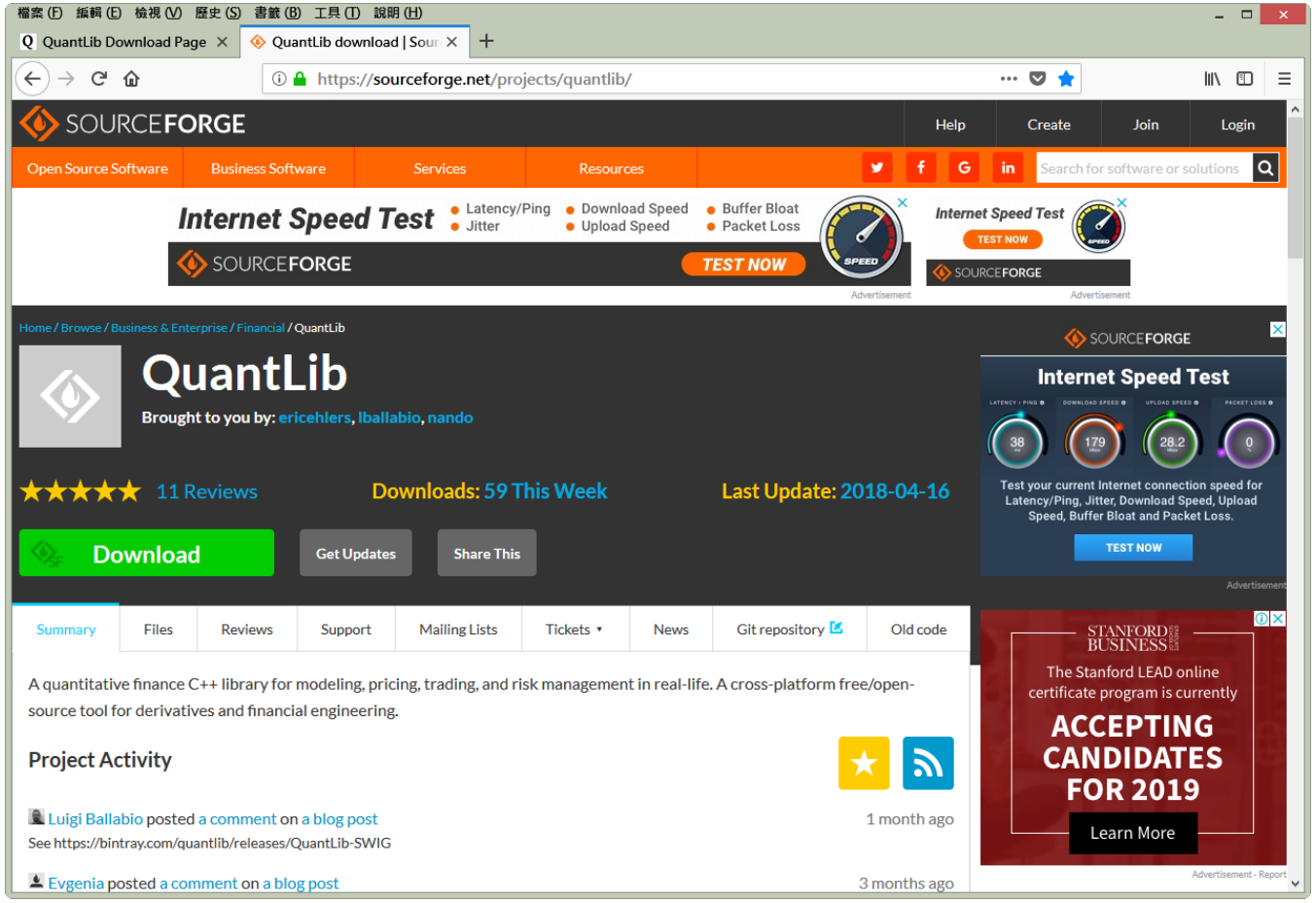
- 雖然最早專案的目標對象為學術界與實務界，然而最後卻是進一步促成這兩方人士進一步的交流。
  - QuantLib 所提供的工具，不論對實務上的實作或是進階的建模，都是相當有用的。
- 下表列示了 QuantLib 各版本釋出的時間。目前最新的版本是 1.13 版。
  - 除了 C++語言之外，其他語言的 QuantLib 專案也路續成立運作，
    - 例如使用 C#語言的 QLNet，https://sourceforge.net/projects/ qlnet/；
    - 使用 R 語言的 RQuantLib，https://dirk.eddelbuettel.com/code/ rquantlib.html；
    - 使用 Java 語言的 JQuantLib，https://www.jquantlib.org；
    - 使用 Python 語言的 PyQL，https://github.com/enthought/pyql；
    - 以及可配合 Excel 使用的增益集，本專案的子專案 QuantLibXL。

◆ QuantLib 各版本的演進與推出時間表

Version	Release date	Notes
0.1.1	Nov 21, 2000	
0.2.0	Sep 18, 2001	
0.3.4	Nov 21, 2003.	
0.3.7	Jul 23, 2004.	此版之後 QuantLib 需要 Boost 程式庫。
0.4.0	Feb 20, 2007.	
0.8.0	May 30, 2007.	版本的大幅更動，目的在加快收斂到 1.0 版。
0.9.0	Dec 24, 2007.	
0.9.9	Nov 2009.	
1.0.0	Feb 24, 2010	
1.0.1	Sep 17, 2010	
1.1.0	May 23, 2011	
1.2.0	March 6, 2012	

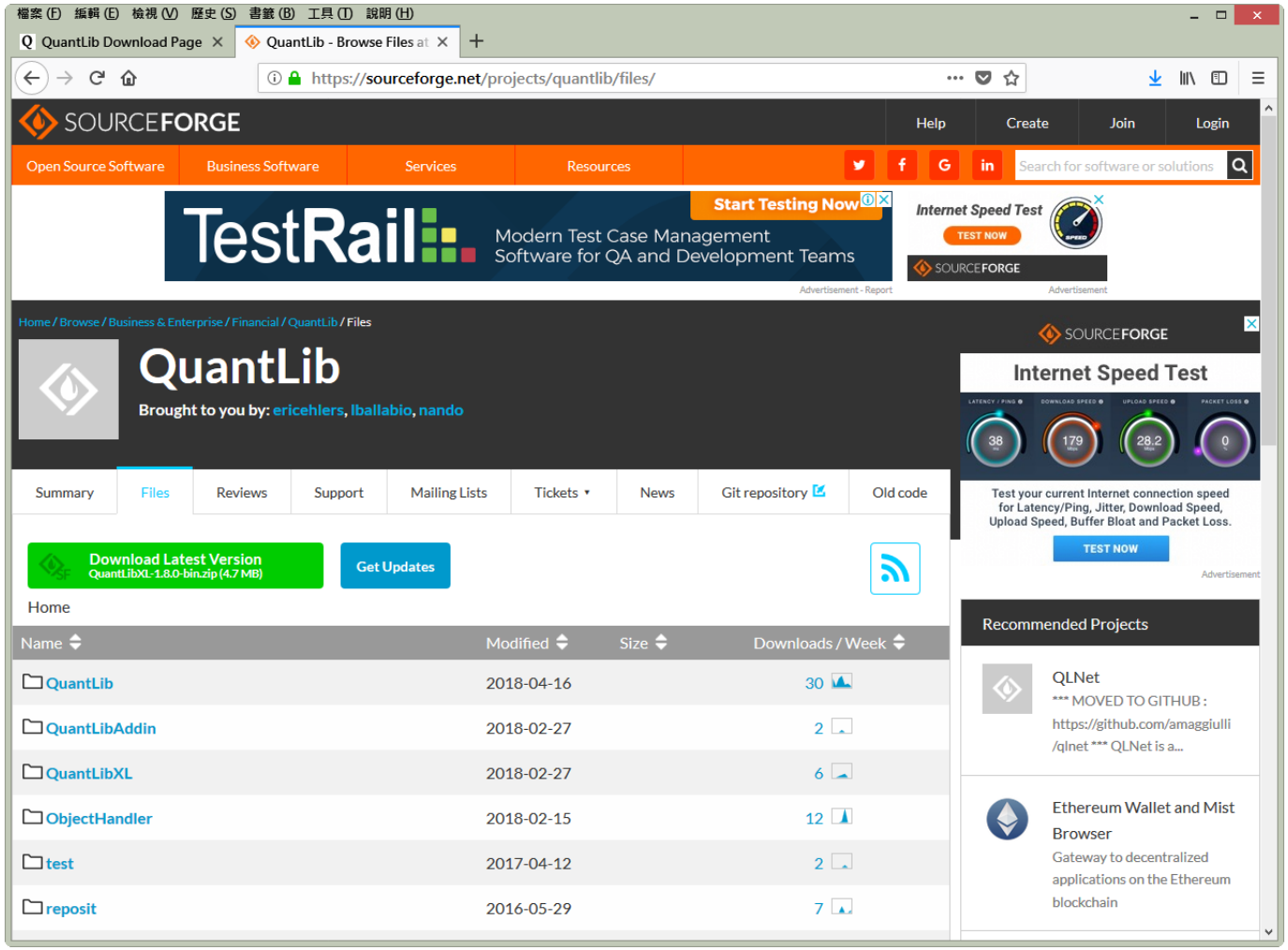
(三)QuantLib 程式庫的下載

- 可在 Open Source Software 網站內的 QuantLib 專案網頁下載，
  - http://sourceforge.net/projects/quantlib/，2018/09/16。

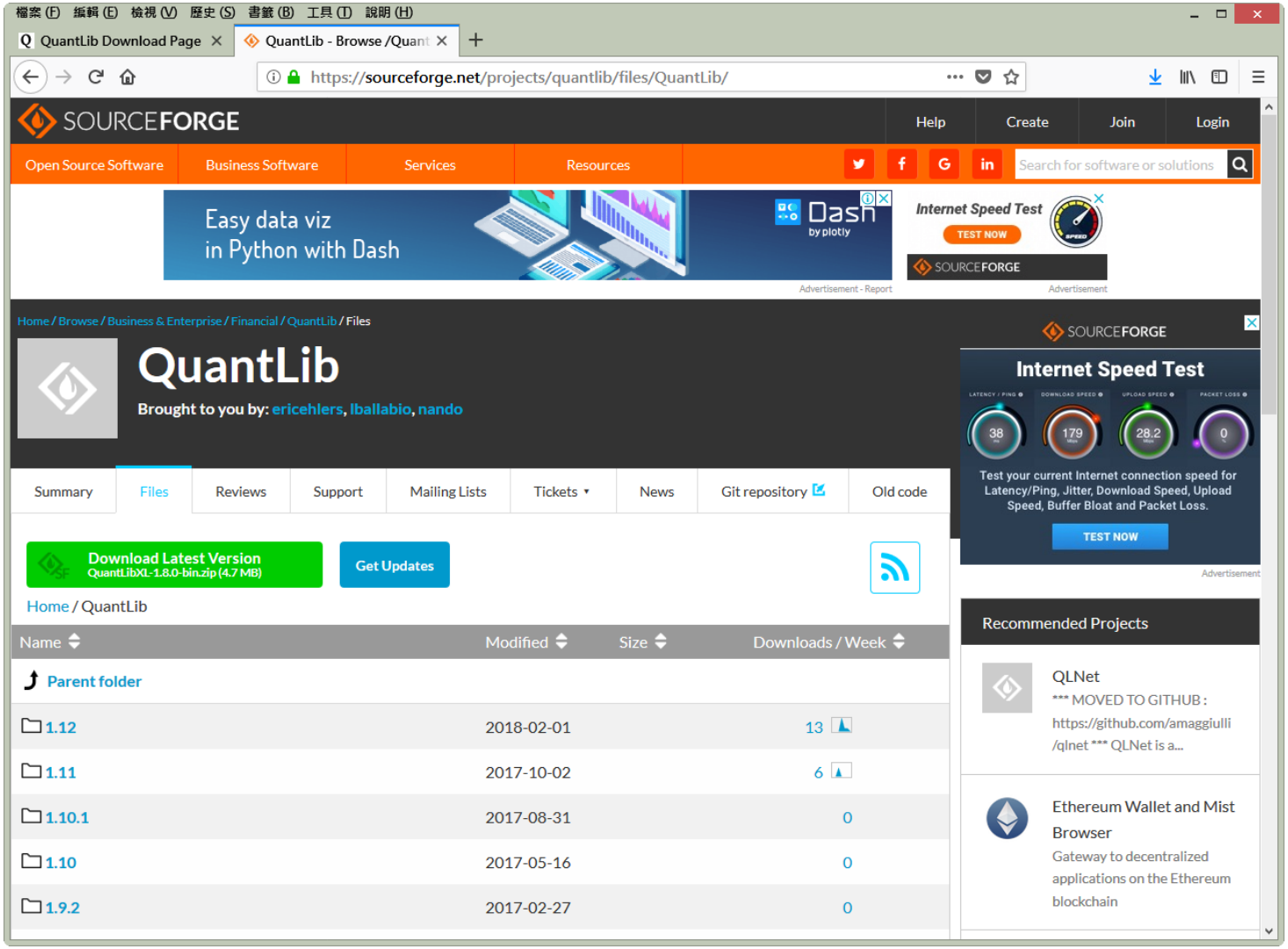


◆ 點 Files，可以看到下面目錄結構。

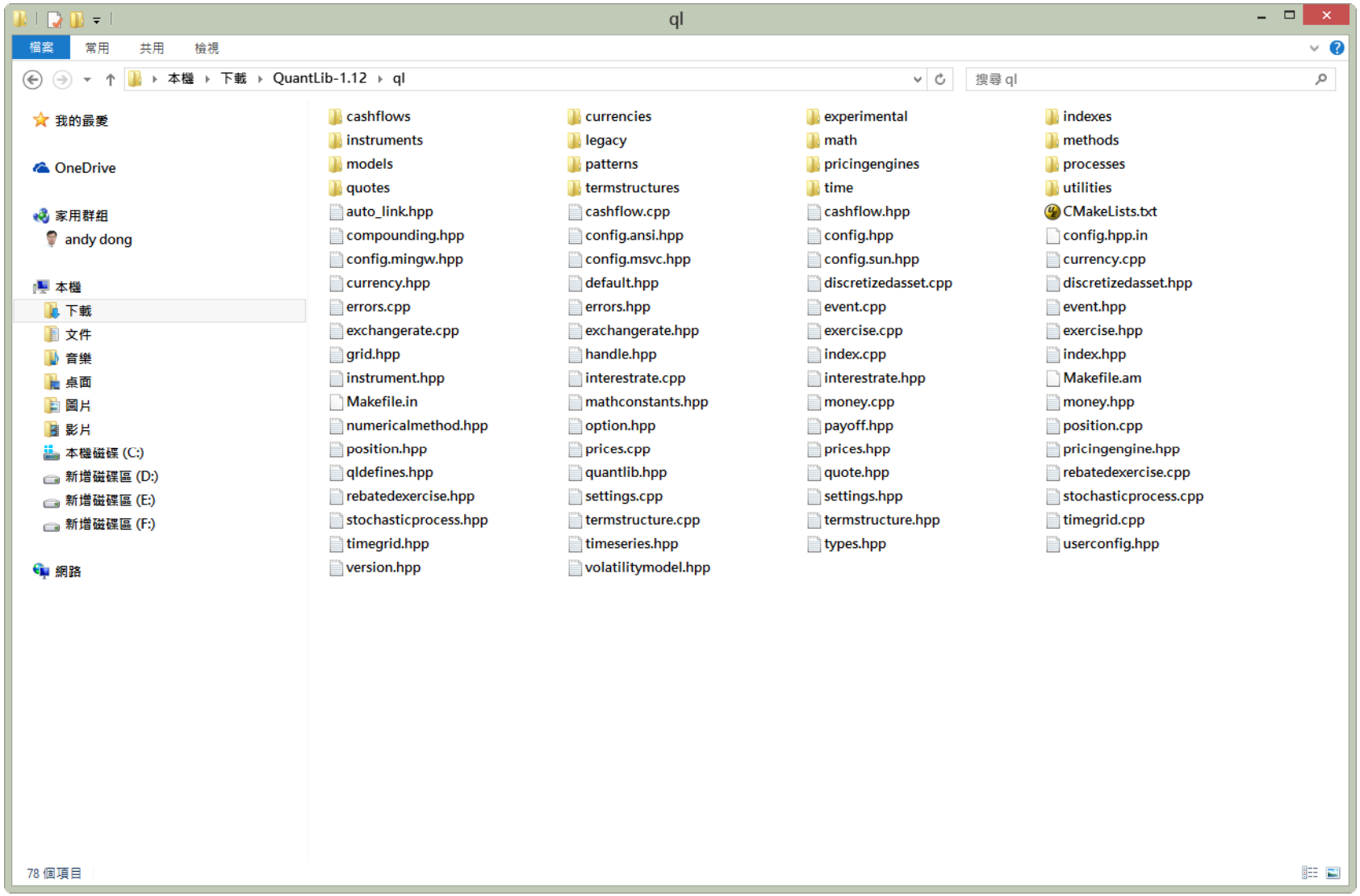
➢ 點選 QuantLib，



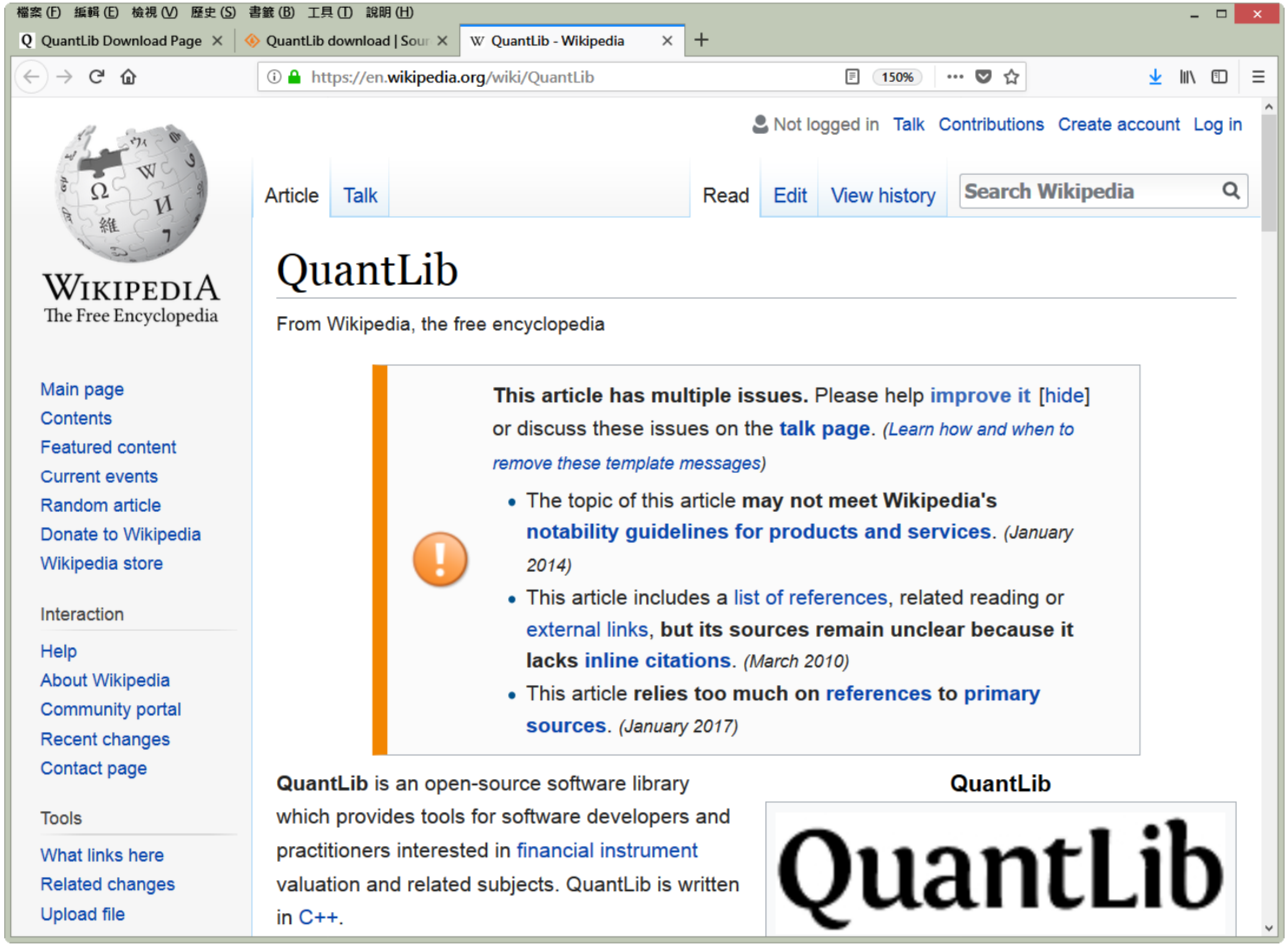
◆ 最新版 1.12 版。


QuantLib Download Page
QuantLib - Browse /Quant
https://sourceforge.net/projects/quantlib/files/QuantLib/
SOURCEFORGE
Help
Create
Join
Login
Open Source Software
Business Software
Services
Resources
Search for software or solutions
Easy data viz
in Python with Dash
Dash
by plotly
Internet Speed Test
TEST NOW
SOURCEFORGE
Advertisement - Report
Advertisement
Home / Browse / Business & Enterprise / Financial / QuantLib / Files
QuantLib
Brought to you by: ericehlers, lballabio, nando
Summary
Files
Reviews
Support
Mailing Lists
Tickets
News
Git repository
Old code
Download Latest Version
QuantLibXL-1.8.0-bin.zip (4.7 MB)
Get Updates
Home / QuantLib
Name
Modified
Size
Downloads / Week
Parent folder
1.12
2018-02-01
13
1.11
2017-10-02
6
1.10.1
2017-08-31
0
1.10
2017-05-16
0
1.9.2
2017-02-27
0
SOURCEFORGE
Internet Speed Test
38
179
28.2
0
Test your current Internet connection speed for Latency/Ping, Jitter, Download Speed, Upload Speed, Buffer Bloat and Packet Loss.
TEST NOW
Advertisement
Recommended Projects
QLNet
*** MOVED TO GITHUB : https://github.com/amaggiulli/qlnet *** QLNet is a...
Ethereum Wallet and Mist Browser
Gateway to decentralized applications on the Ethereum blockchain

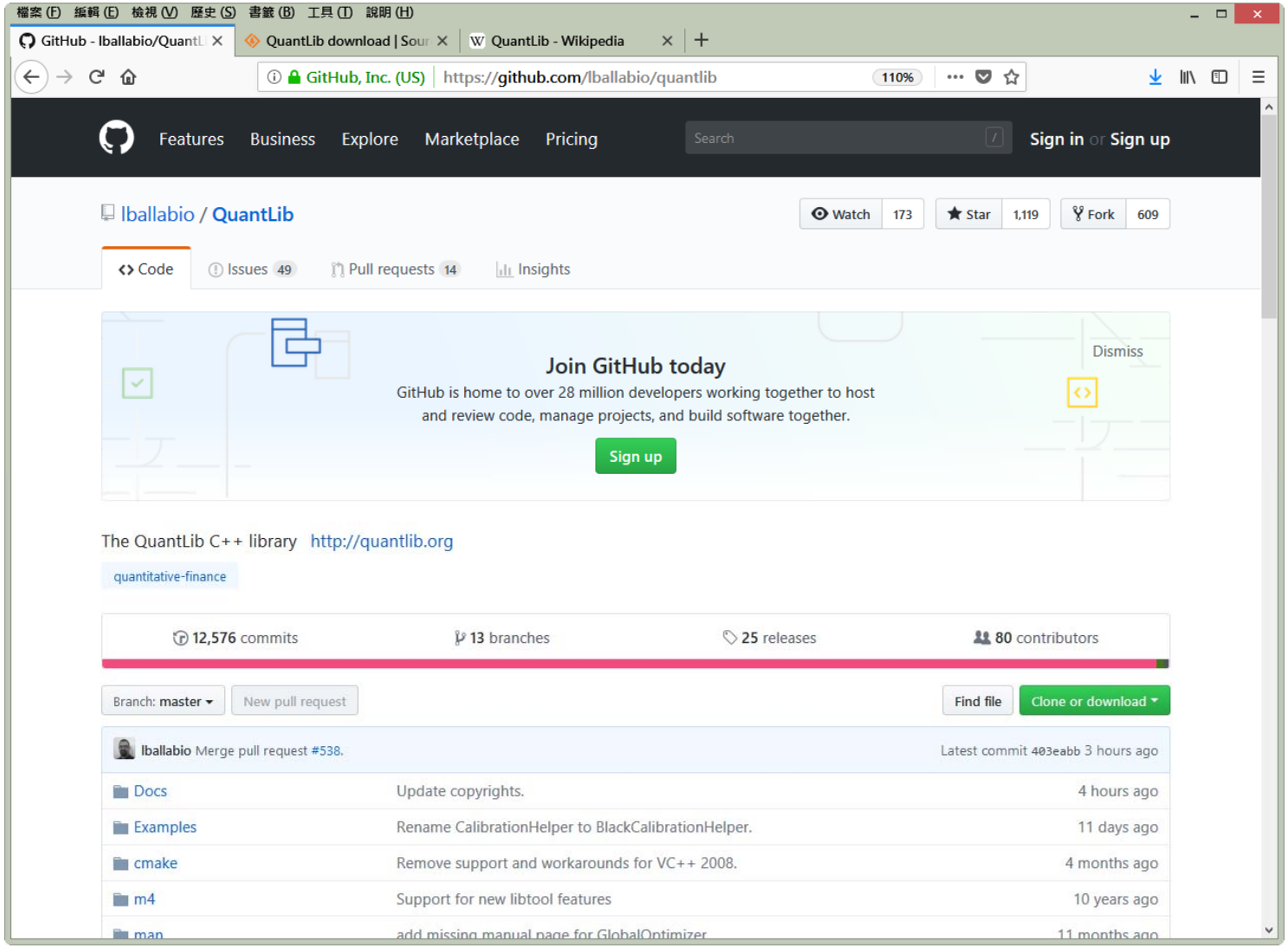
◆完整目錄


ql
檔案
常用
共用
檢視
本機 ▸ 下載 ▸ QuantLib-1.12 ▸ ql
搜尋 ql
我的最愛
OneDrive
家用群組
andy dong
本機
下載
文件
音樂
桌面
圖片
影片
本機磁碟 (C:)
新增磁碟區 (D:)
新增磁碟區 (E:)
新增磁碟區 (F:)
網路
cashflows
instruments
models
quotes
auto_link.hpp
compounding.hpp
config.mingw.hpp
currency.hpp
errors.cpp
exchangerate.cpp
grid.hpp
instrument.hpp
Makefile.in
numericalmethod.hpp
position.hpp
qldefines.hpp
rebatedexercise.hpp
stochasticprocess.hpp
timegrid.hpp
version.hpp
currencies
legacy
patterns
termstructures
cashflow.cpp
config.ansi.hpp
config.msvc.hpp
default.hpp
errors.hpp
exchangerate.hpp
handle.hpp
interestrate.cpp
mathconstants.hpp
option.hpp
prices.cpp
quantlib.hpp
settings.cpp
termstructure.cpp
timeseries.hpp
volatilitymodel.hpp
experimental
math
pricingengines
time
cashflow.hpp
config.hpp
config.sun.hpp
discretizedasset.cpp
event.cpp
exercise.cpp
index.cpp
interestrate.hpp
money.cpp
payoff.hpp
prices.hpp
quote.hpp
settings.hpp
termstructure.hpp
types.hpp
indexes
methods
processes
utilities
CMakeLists.txt
config.hpp.in
currency.cpp
discretizedasset.hpp
event.hpp
exercise.hpp
index.hpp
Makefile.am
money.hpp
position.cpp
pricingengine.hpp
rebatedexercise.cpp
stochasticprocess.cpp
timegrid.cpp
userconfig.hpp
78 個項目

◆ QuantLib 的 Wiki 網頁為，http://en.wikipedia.org/wiki/ QuantLib。



◆ Git 網站，2018/09/16。



(四)QuantLib 程式庫的內容

◆ QuantLib 程式庫內容豐富，包含的檔案上千個，可以概分 15 類模組，如下所示。

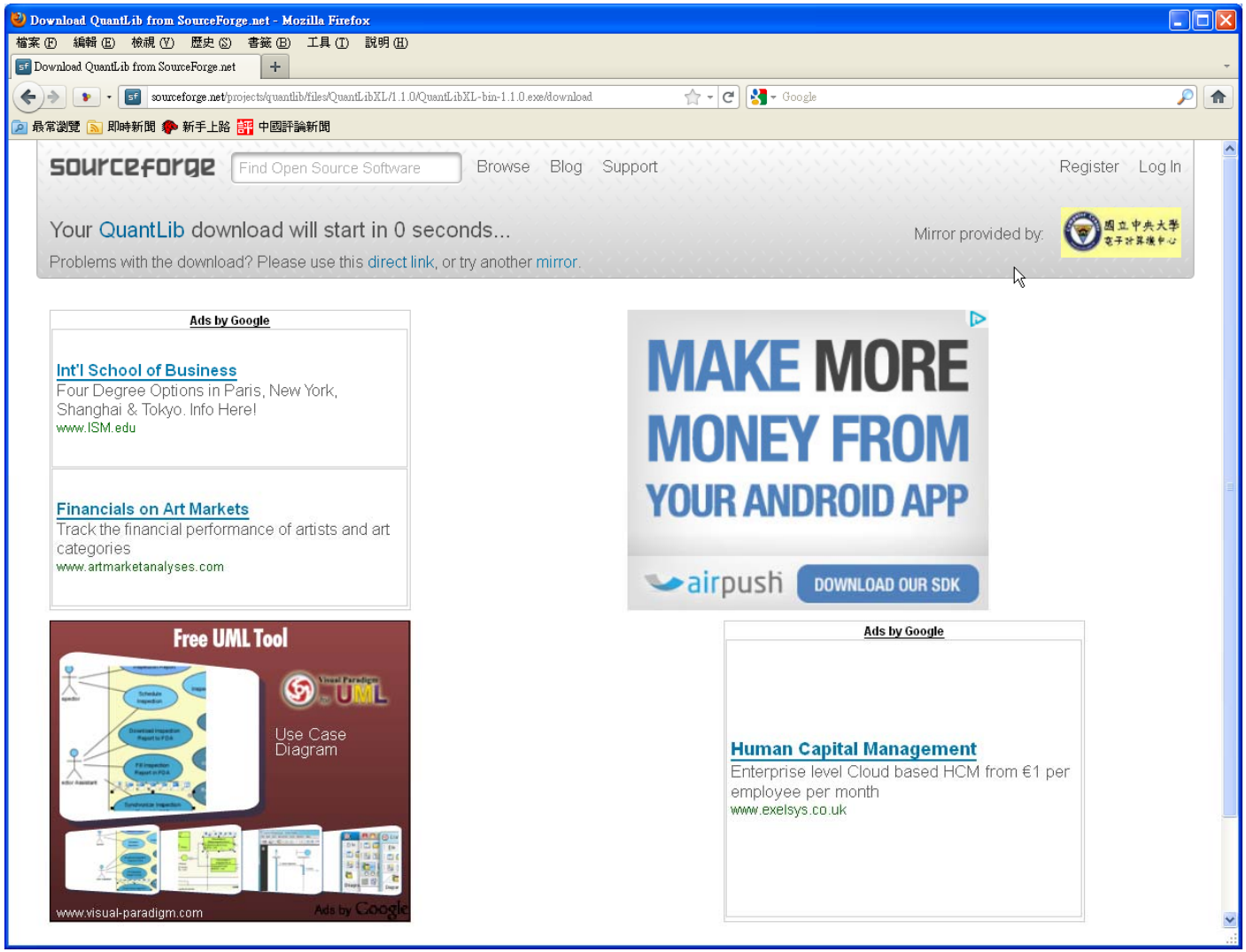
- 1.Numeric types：主要在定義各類資料的資料型別，例如，利率(Rate)、利差(Spread)、波動性(Volatility)是實數(Real)。
- 2.Currencies and FX rates：包括 66 種貨幣類別以及相關匯率轉換計算與管理的物件。
- 3.Date and time calculations：包括 37 種不同國家/地區日曆類別、7 種不同計息方式類別以及相關日期計算的物件。
- 4.Pricing engines：九大類評價引擎，包括 Asian option、Barrier option、Basket option、Cap/Floor、Cliquet option、Forward option、Quanto option、Swaption、Vanilla option 等。每一類都有解析解、二元樹、有限差分法與蒙地卡羅模擬的實作類別。
- 5.Finite-differences framework：三大類有限差分法的實作類別。
- 6.Short-rate modelling framework：包括單因子模型，Vasicel、CIR、Hull-White、Black-Karasinski、Extended CIR 等模型。雙因子模型的 G2 模型。

- 7.Financial instruments：40 多種的金融工具，含 Swap、Vanilla Option、Exotic Option、Stock、Forward、Cap、Floor、Color、Bond、Future、Callable Bond 以及相對應的 Quanto 產品、Inflation Bond。
- 8.Lattice methods：一維與二維的二元樹和三元樹模型。
- 9.Math tools：包括分配、積分、相關性、內差、矩陣、最適化、亂數、求解、統計等九類數學工具。
- 10.Monte Carlo framework：單變數與多變數的歐式與美式模擬方法物件。
- 11.Design patterns：Singleton、Observer/Observable、Lazy Object、Composite、Curiously Recurring Template、Acyclic Visitor 樣式。
- 12.Stochastic processes：幾何布朗運動、隨機波動模型、Square Root Process、Ornstein Uhlenbeck Process、Hull White Process、G2 Process。
- 13.Term structures：包括利率、波動性、信用與通膨的期限結構物件。
- 14.Models：分別包括 Equity Model、Short Rate Model、Volatility Model 中使用的相關模型，以及針對 Libor/Swap Market Model 涉及的金融工具與模擬方法的相關物件上百個，。
- 15.QuantLib macros：一些數值極限與除錯所需的 Macros。

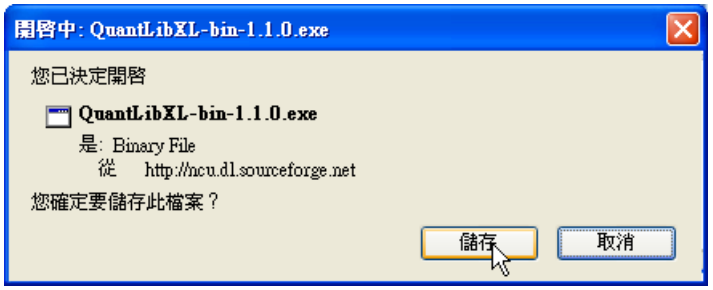
(五)QuantLibXL增益集的安裝

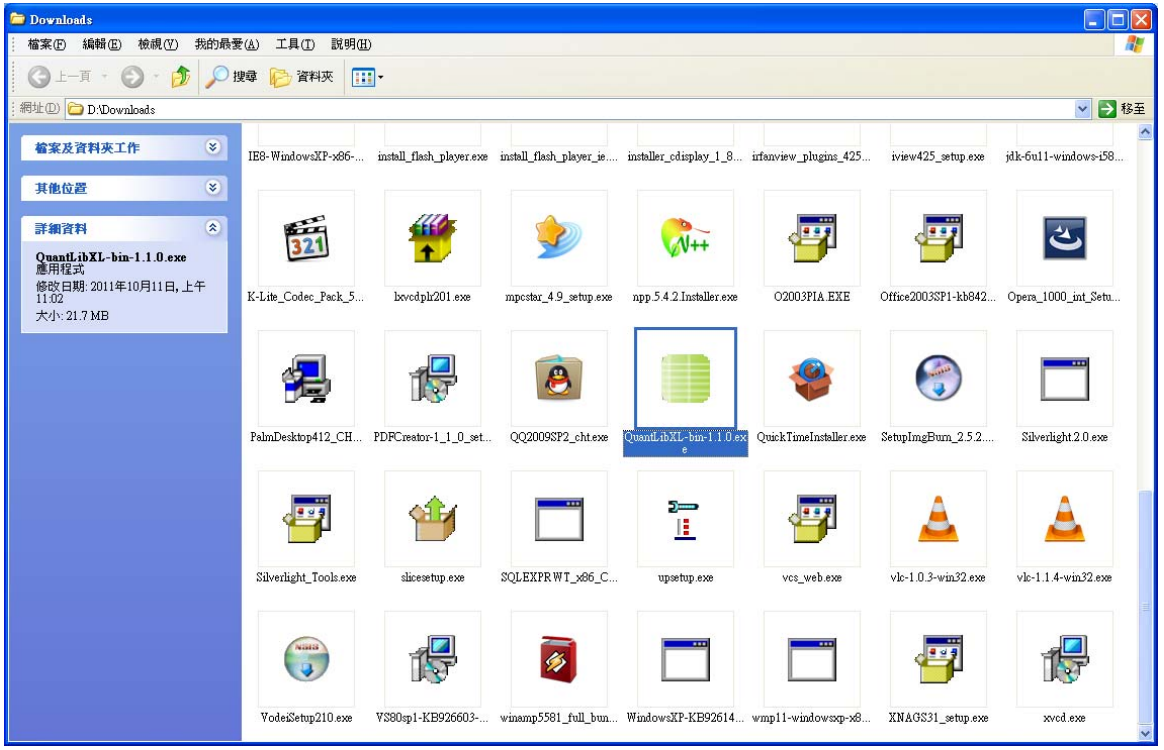
◆ 進入如下圖的網頁，http://quantlib.org/quantlibxl/，按右上方 QuantLibXL Download 的按鈕。

➢ 接著，網頁自動進入到圖 2.2 Sourceforge 下載的頁面。

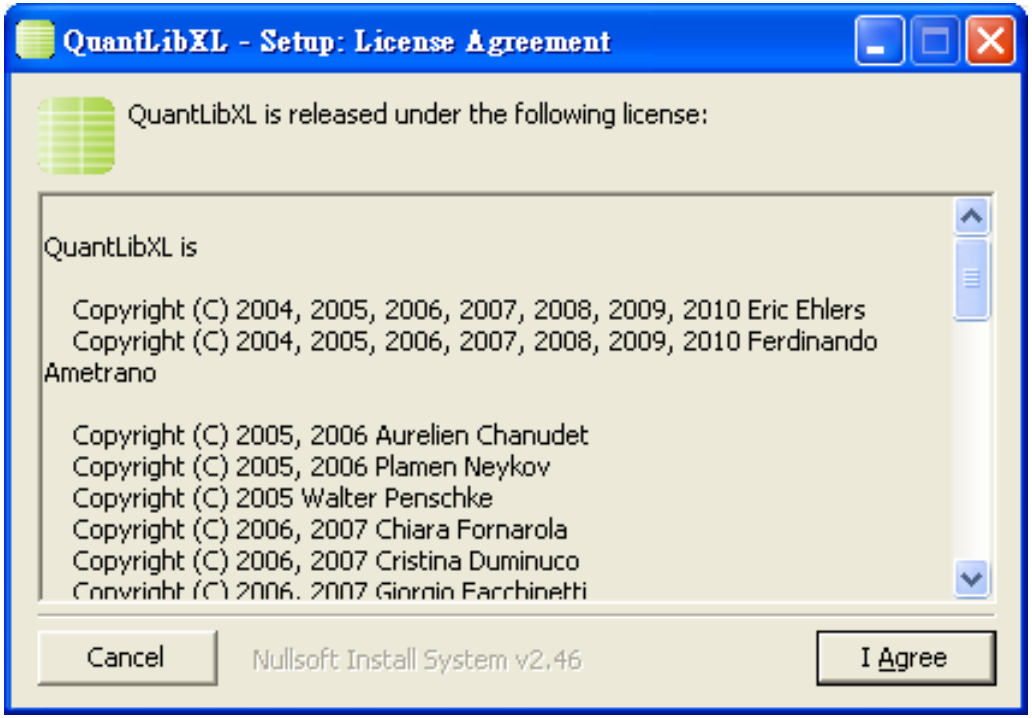


➢ 等待幾秒，自動出現圖 2.3 下載 QuantLibXL 安裝執行檔的對話盒，按儲存按鈕，作者將之儲存到 D:\Downloads\目錄下。

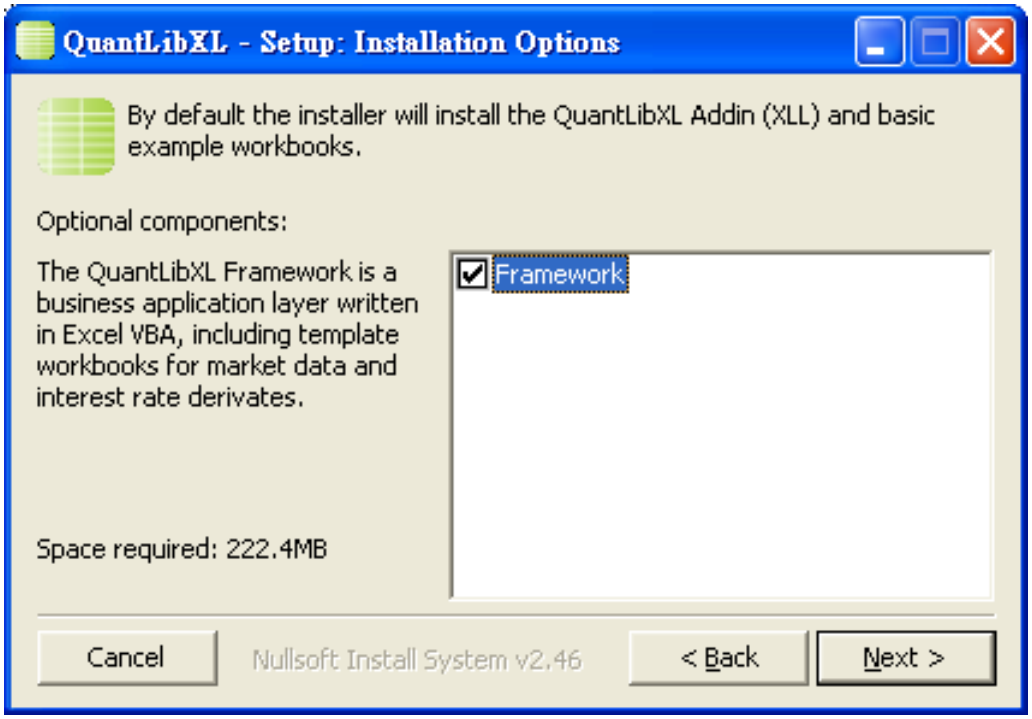




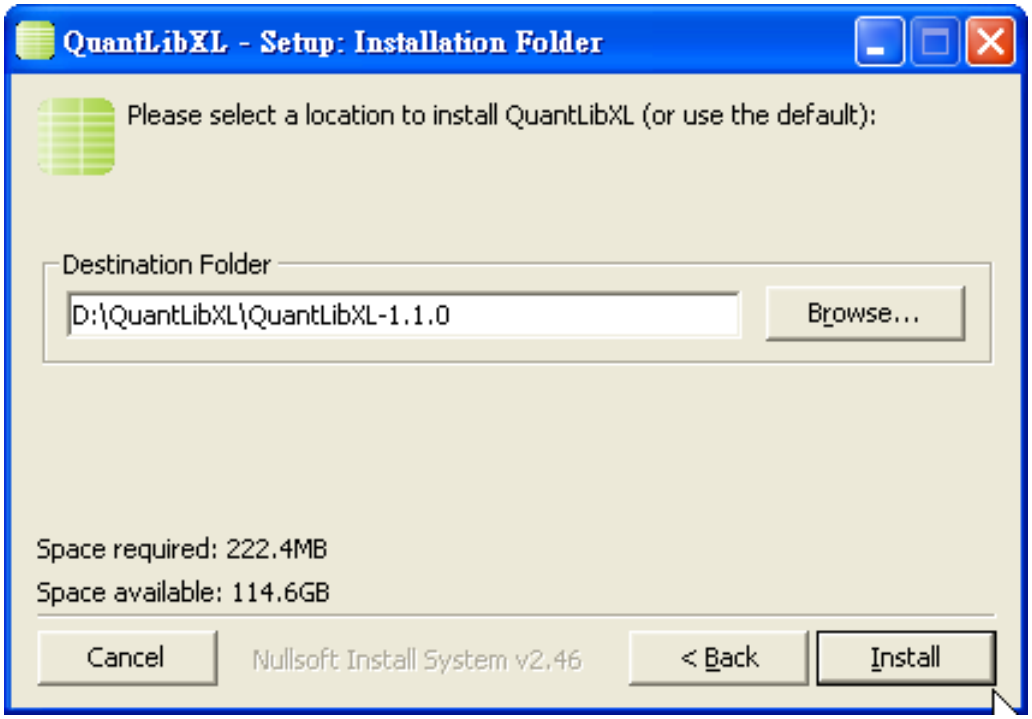
◆ 點選 QuantLibXL-bin-1.1.0.exe 執行之，出現 QuantLibXL 版權協議說明，按同意按鈕，



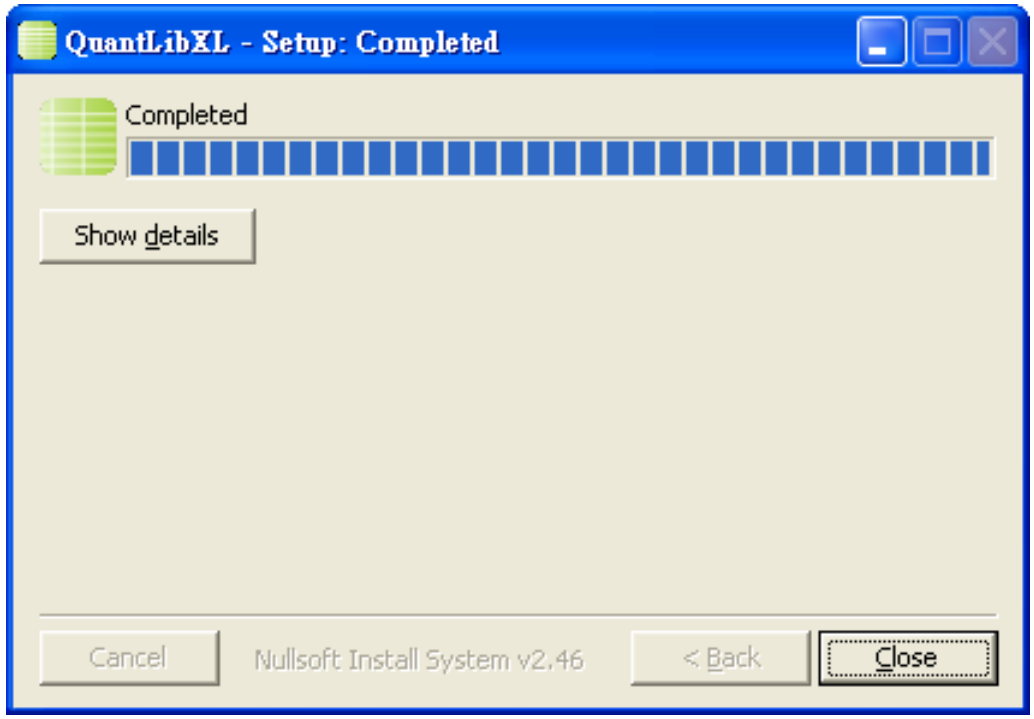
- 出現 QuantLibXL 安裝選項。
  - ✓ 此選項是詢問讀者是否要安裝 QuantLibXL Framework，它包含一些針對市場資料與利率衍生商品的樣板試算表，是以 Excel VBA 所撰寫的。讀者可以選取以安裝之，按 Next 往下執行。



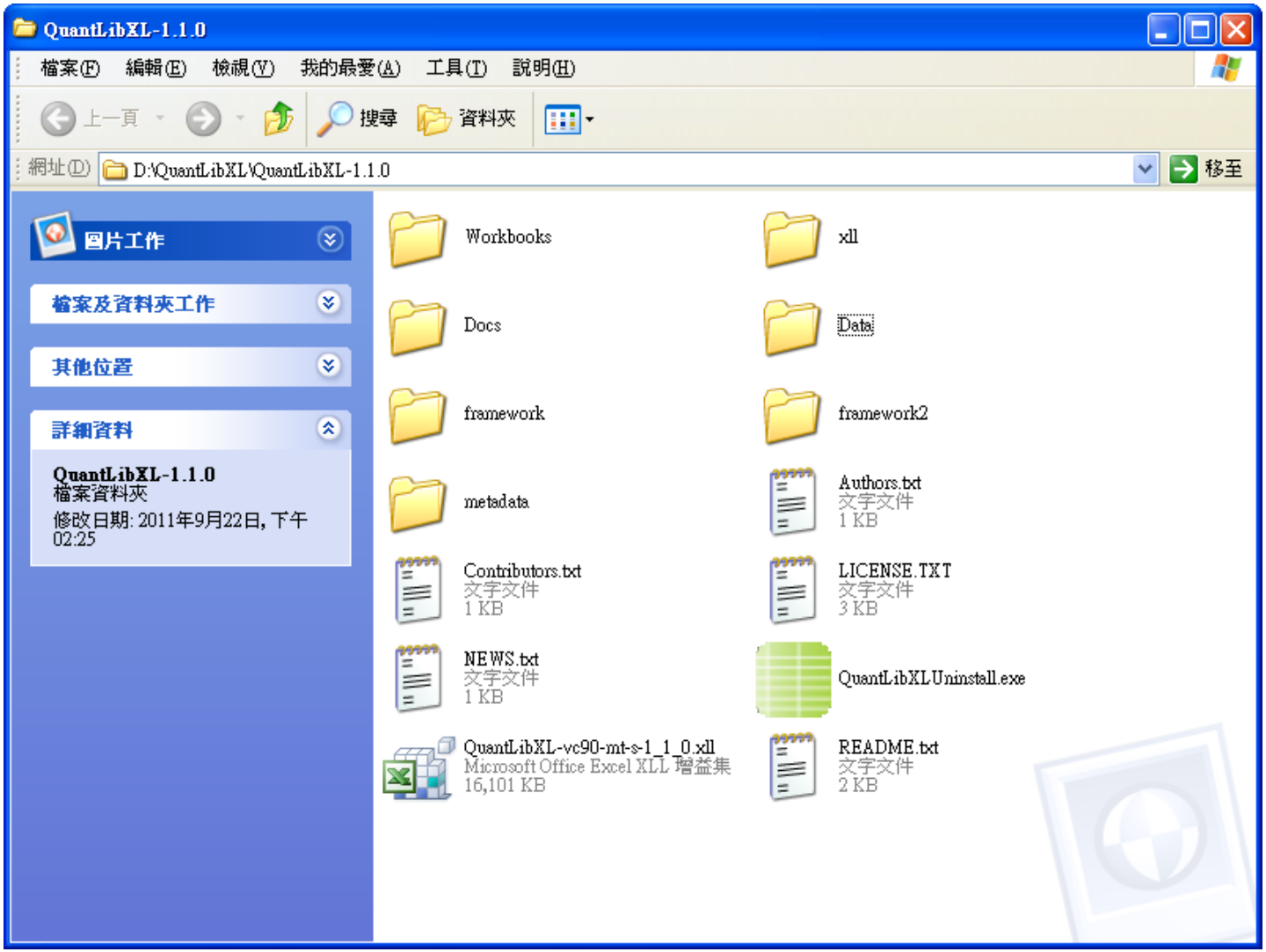
- 接著出現 QuantLibXL 安裝路徑設定。
  - ✓ 讀者可自行設定到想要的地方，作者將之設定到 D:\QuantLibXL\QuantLibXL-1.1.0。
  - ✓ 按安裝按鈕，便會開始安裝之。



- 安裝完成畫面

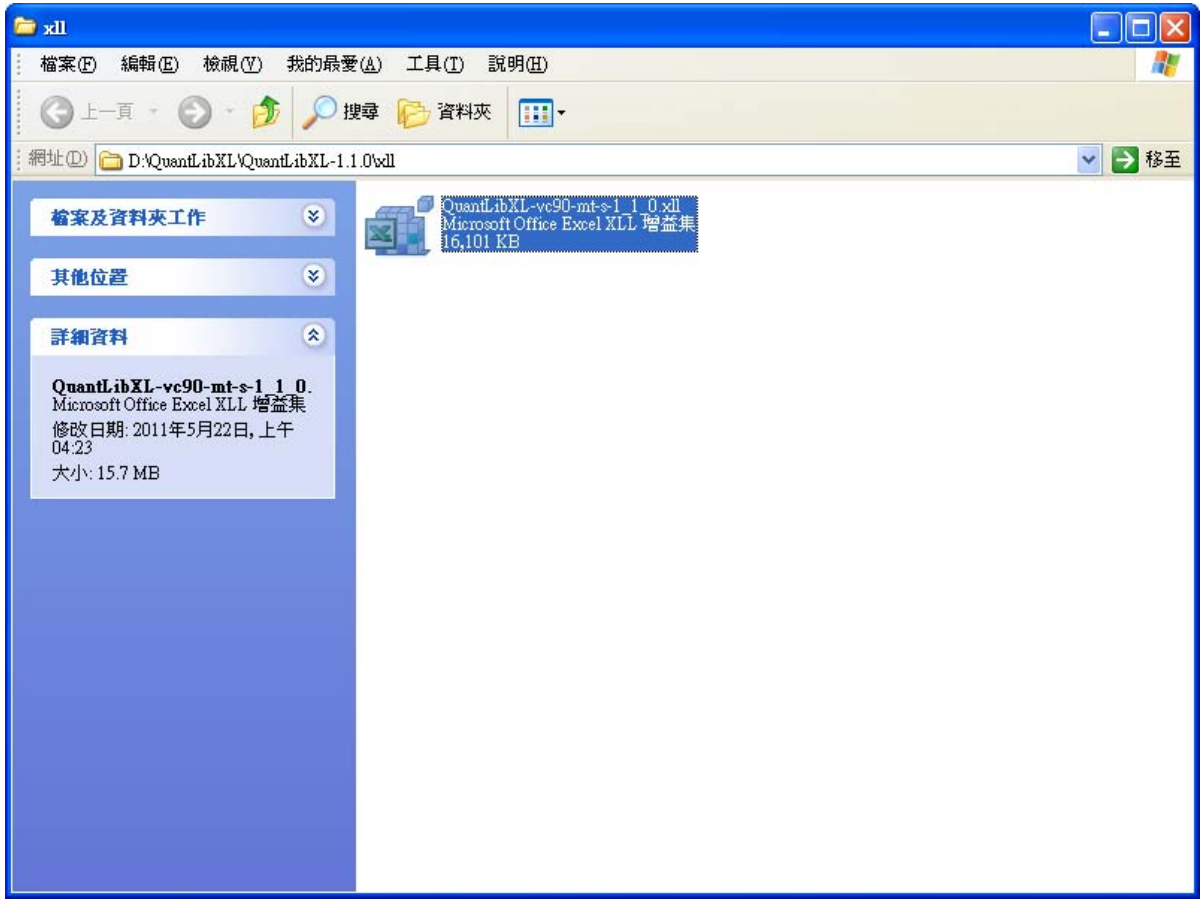

QuantLibXL - Setup: Completed
Completed
Show details
Cancel
Nullsoft Install System v2.46
< Back
Close

◆ 安裝的目錄與檔案

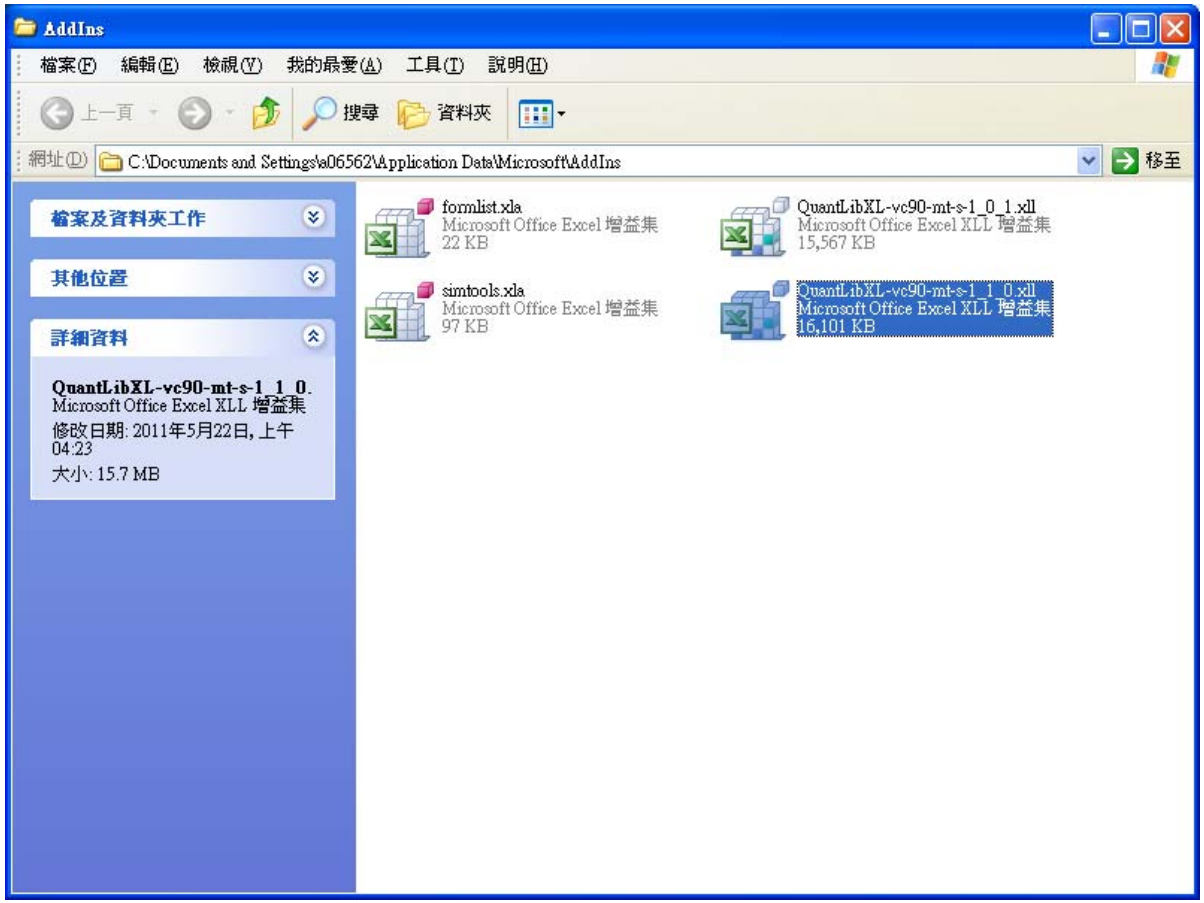

QuantLibXL-1.1.0
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 資料夾
網址(D) D:\QuantLibXL\QuantLibXL-1.1.0
移至
圖片工作
檔案及資料夾工作
其他位置
詳細資料
QuantLibXL-1.1.0
檔案資料夾
修改日期: 2011年9月22日, 下午 02:25
Workbooks
xll
Docs
Data
framework
framework2
metadata
Authors.txt
文字文件
1 KB
Contributors.txt
文字文件
1 KB
LICENSE.TXT
文字文件
3 KB
NEWS.txt
文字文件
1 KB
QuantLibXLUninstall.exe
QuantLibXL-vc90-mt-s-1_1_0.xll
Microsoft Office Excel XLL 增益集
16,101 KB
README.txt
文字文件
2 KB

(六)QuantLibXL的使用測試

- 首先，將增益集檔案，QuantLibXL-vc90-mt-s-1_1_0.xll，拷貝到正確的位置。
  - 增益集在路徑 D:\QuantLibXL \QuantLibXL-1.1.0\xll 下面

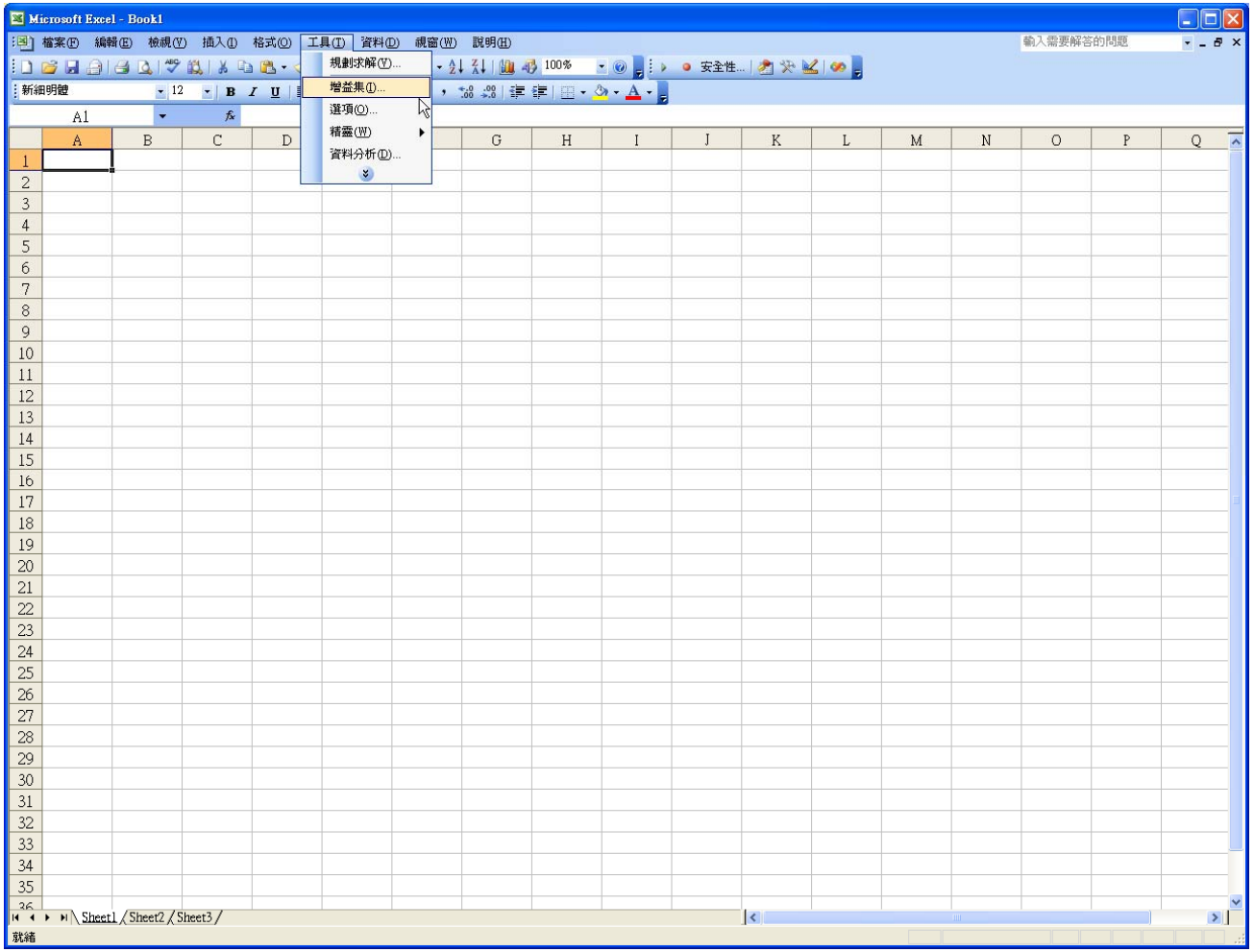


- 拷貝到 Excel 增益集的預設目錄下，C:\Documents and Settings\a06562\ Application Data\Microsoft\ AddIns。
  - ✓ 在 Windows 7 作業系統下，拷貝到 Excel 增益集的預設目錄下，C:\Users\andydong\AppData\Roaming\ Microsoft\AddIns。



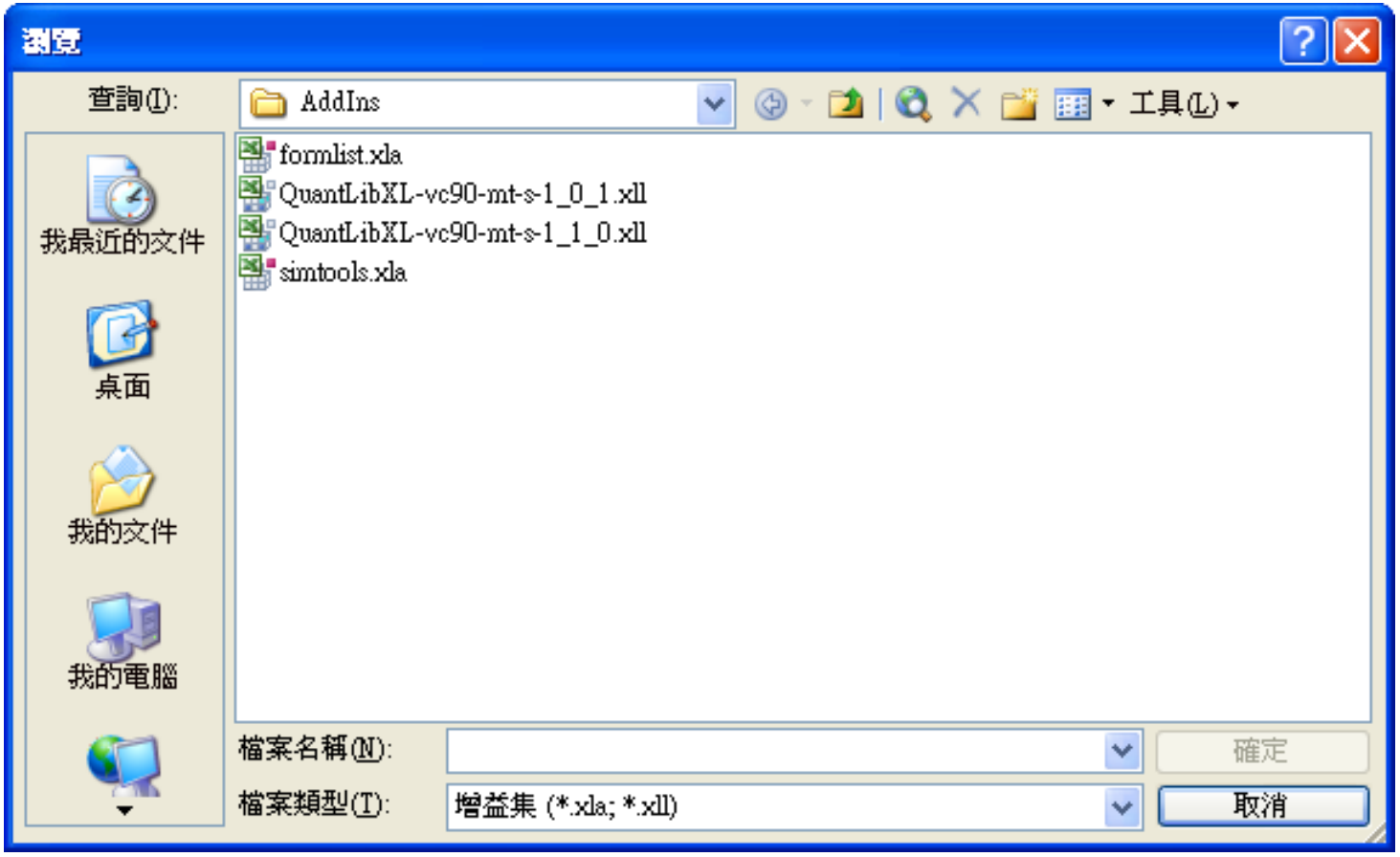
◆ 其次，開啟增益集。

➢ 開啟 Excel 新活頁簿，從《工具》選單中點選《增益集》。



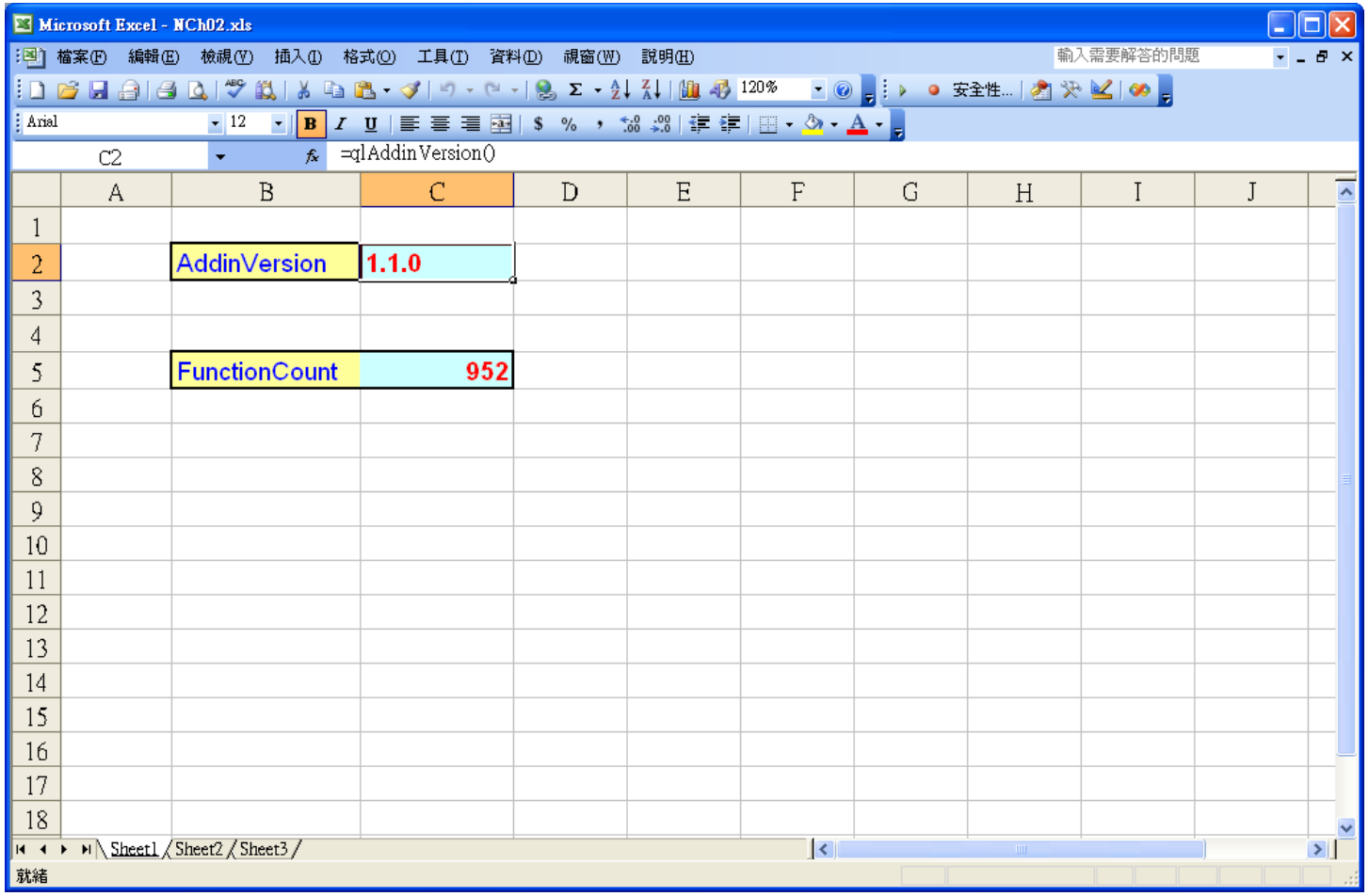
- 可以看到 QuantLibAddin 1.1.0 已經出現在選項之中，勾選之即可。

- 如果 QuantLibXL 增益集沒有放在預設的目錄下，則讀者必須使用《瀏覽》按鈕，找尋所放置的檔案位置。



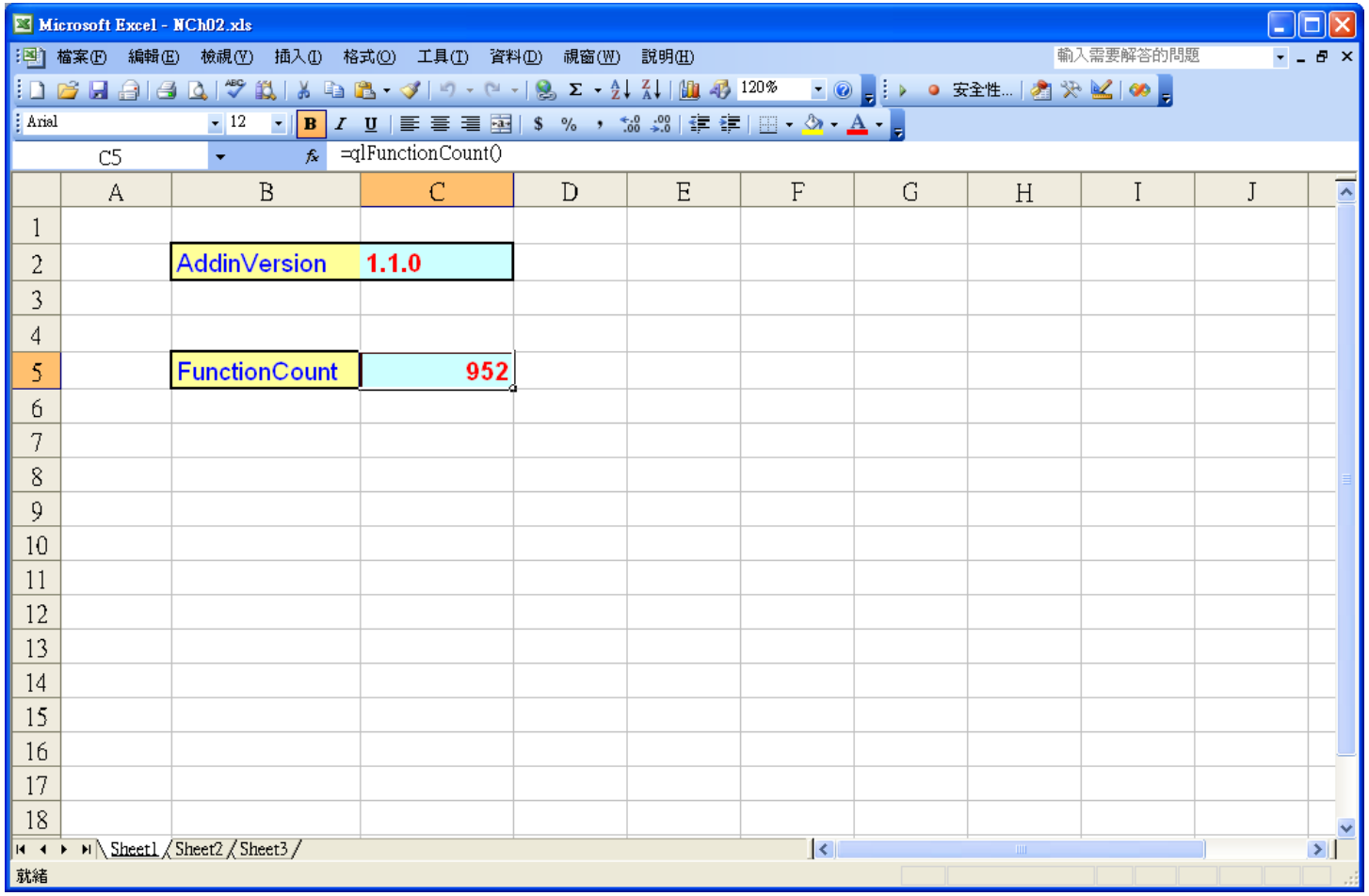
◆ 增益集版本函數

- ➢ 增益集版本的函數，qlAddinVersion()，可以傳回我們安裝的增益集之版本資訊。
  - ✓ 在 B3 格輸入，=qlAddinVersion()，便可傳回，1.1.0。

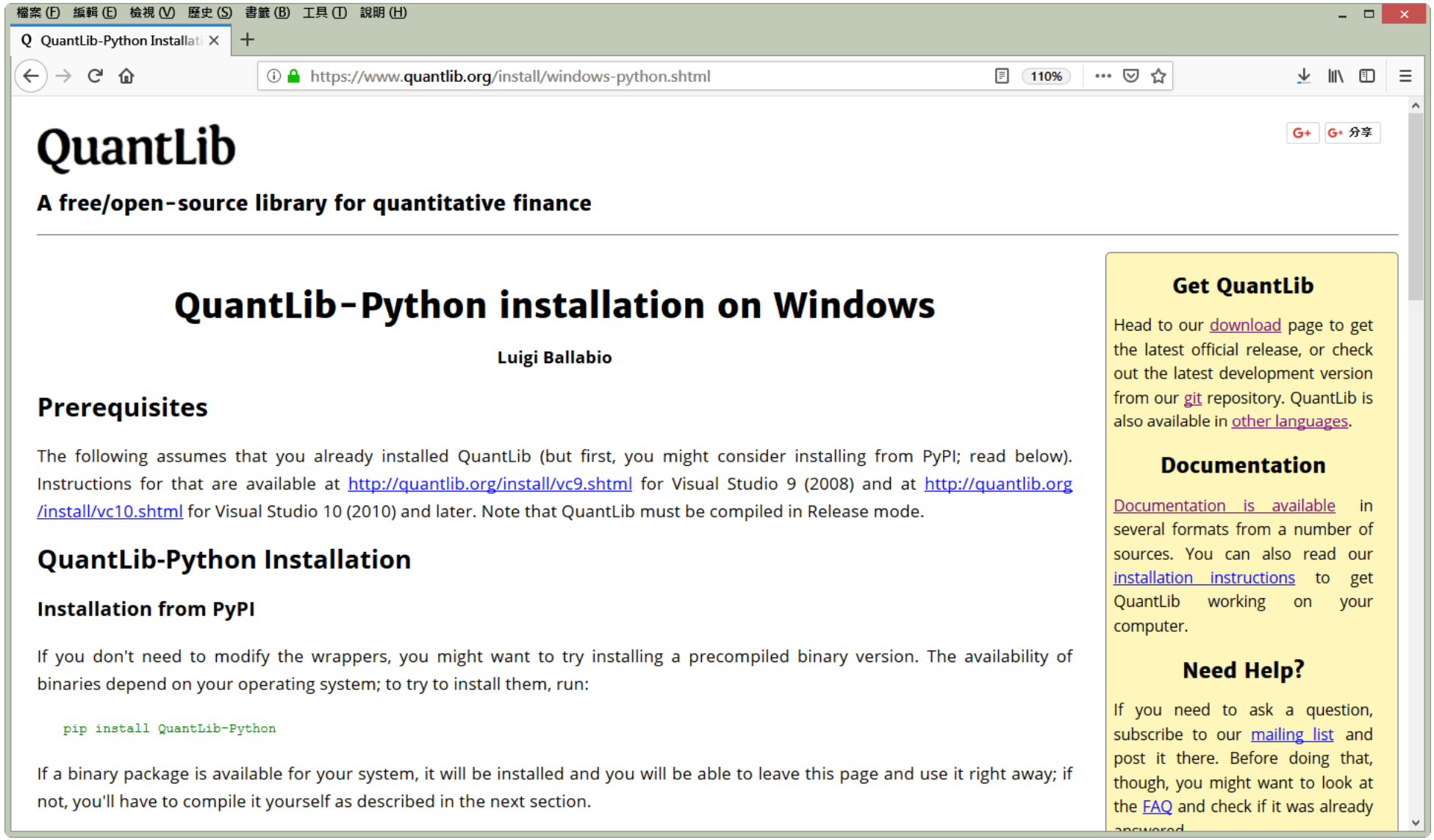


◆ 增益集中函數數量函數

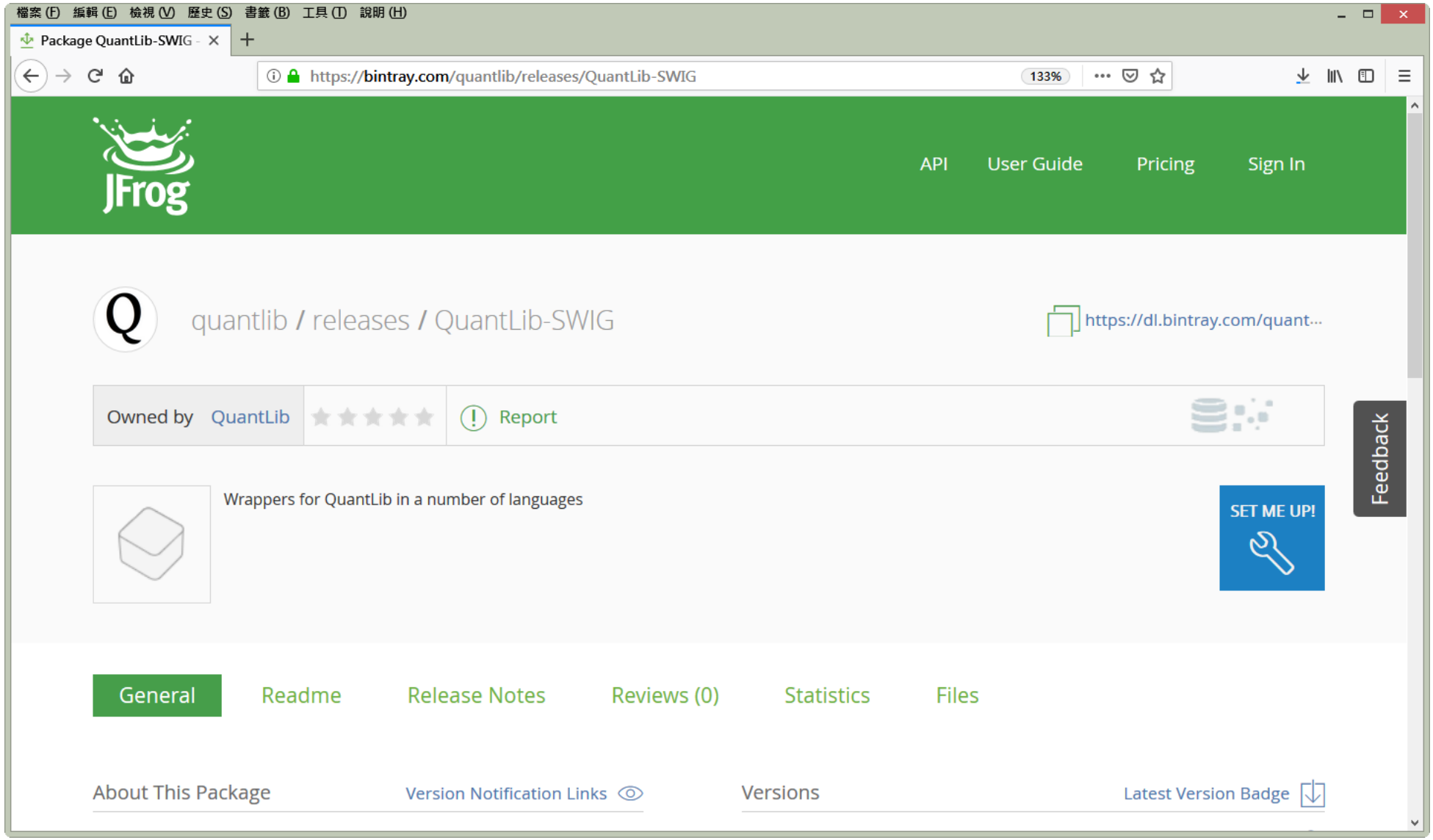
- ➢ 計算增益集中函數數量的函數，qlFunctionCount()，可傳回增益集中函數數量
  - ✓ 在 B7 格輸入，=qlFunctionCount()，便可傳回，952。在 1.0.1 版本中，提供了 898 個函數。



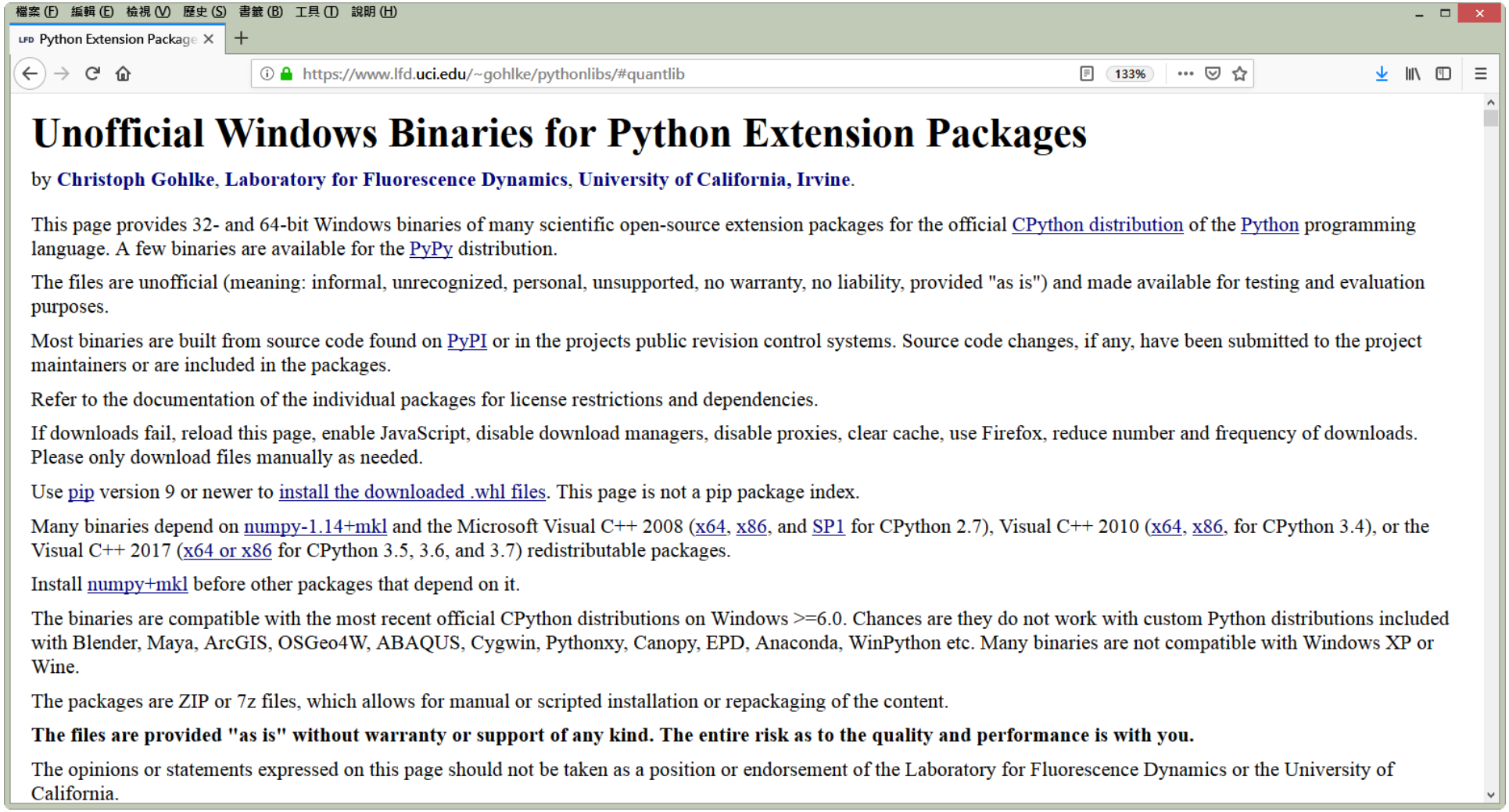
(七)QuantLib-Python Package 下載與安裝



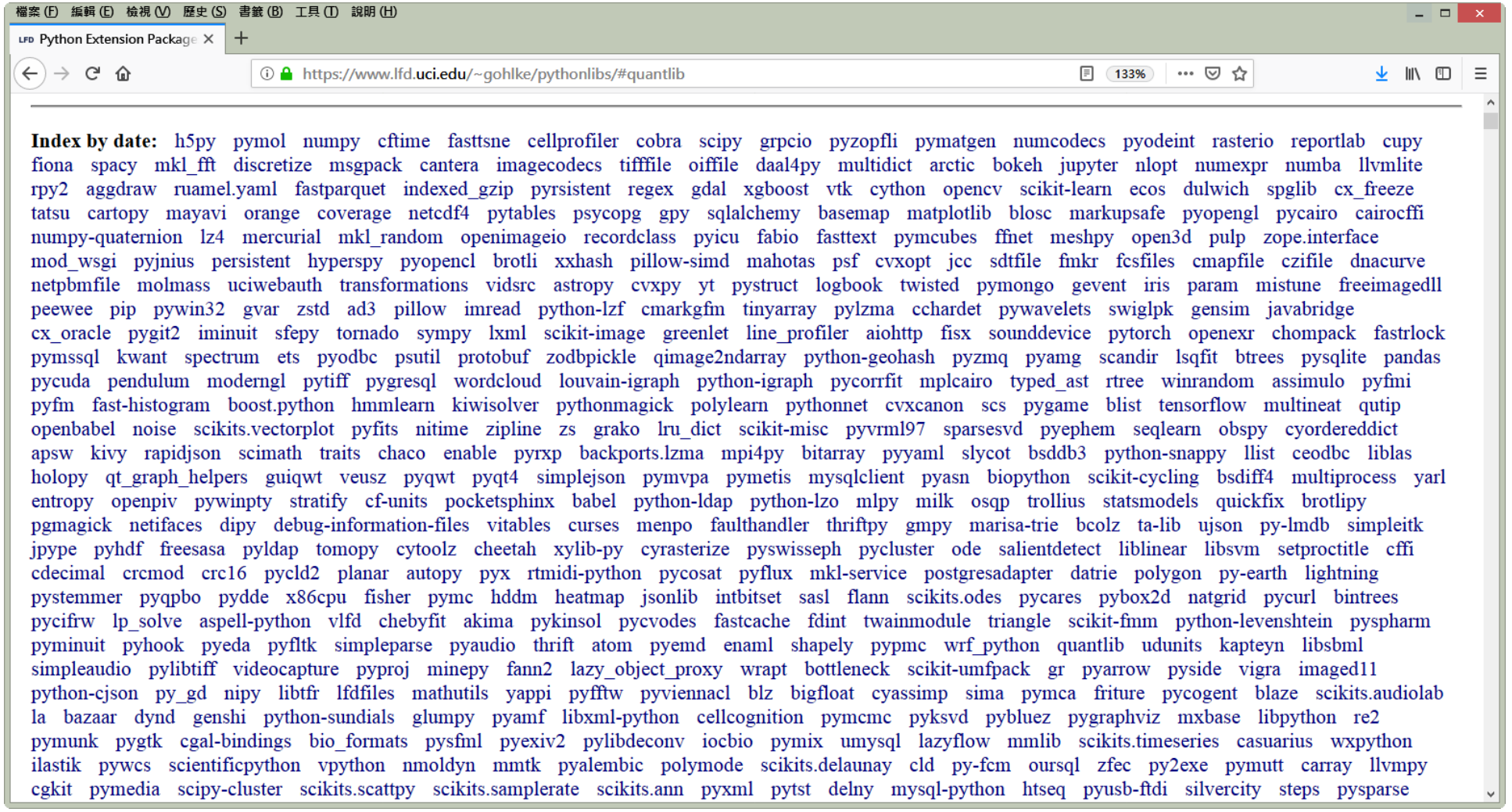
◆Source Code


檔案 (F) 編輯 (E) 檢視 (V) 歷史 (S) 書籤 (B) 工具 (T) 說明 (H)
Package QuantLib-SWIG
https://bintray.com/quantlib/releases/QuantLib-SWIG
133%
JFrog
API
User Guide
Pricing
Sign In
quantlib / releases / QuantLib-SWIG
https://dl.bintray.com/quant...
Owned by QuantLib
Report
Wrappers for QuantLib in a number of languages
SET ME UP!
Feedback
General
Readme
Release Notes
Reviews (0)
Statistics
Files
About This Package
Version Notification Links
Versions
Latest Version Badge

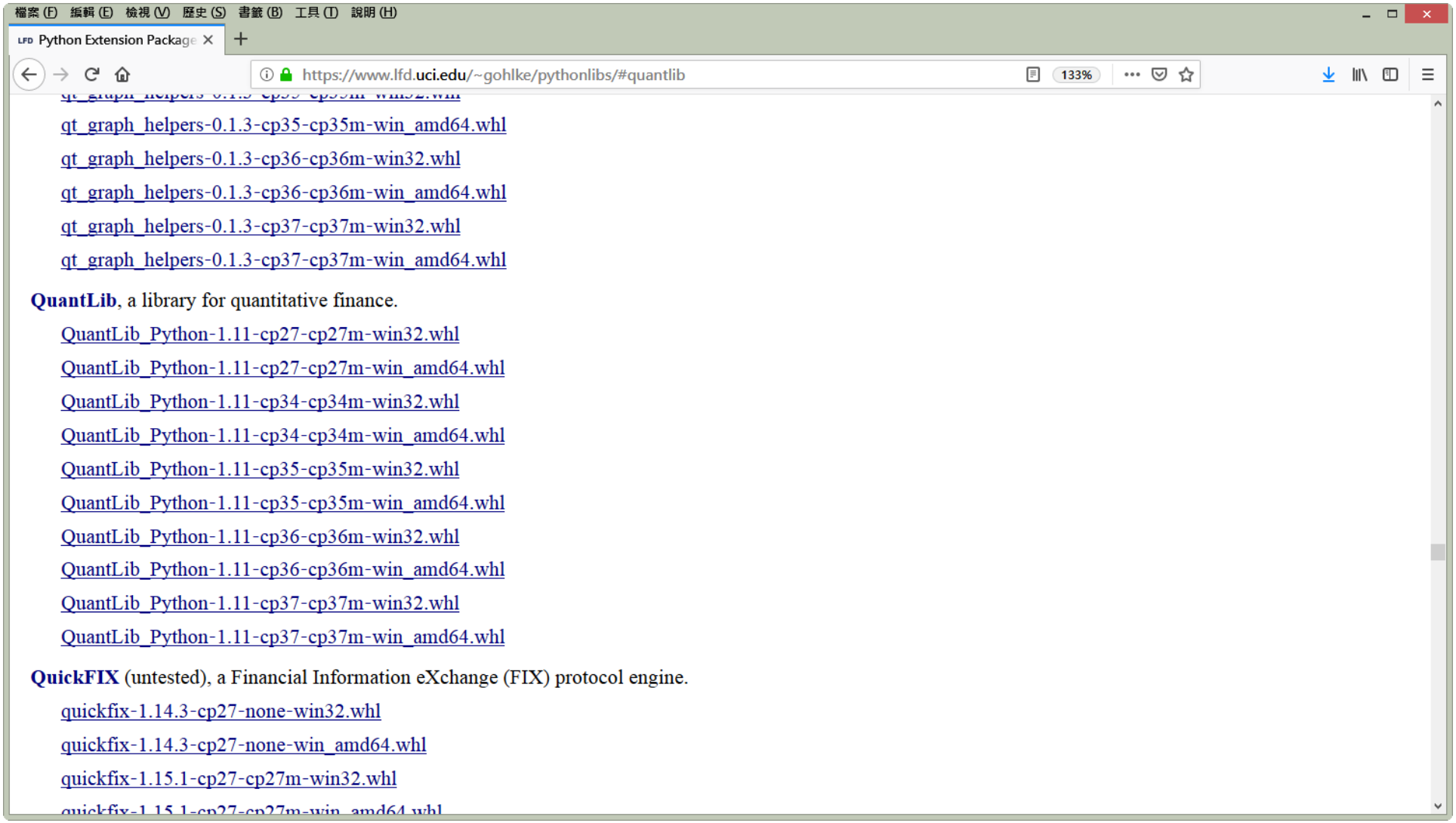
◆下載網址


檔案 (F) 編輯 (E) 檢視 (V) 歷史 (S) 書籤 (B) 工具 (T) 說明 (H)
Python Extension Package
https://www.lfd.uci.edu/~gohlke/pythonlibs/#quantlib
133%
Unofficial Windows Binaries for Python Extension Packages
by Christoph Gohlke, Laboratory for Fluorescence Dynamics, University of California, Irvine.
This page provides 32- and 64-bit Windows binaries of many scientific open-source extension packages for the official CPython distribution of the Python programming language. A few binaries are available for the PyPy distribution.
The files are unofficial (meaning: informal, unrecognized, personal, unsupported, no warranty, no liability, provided "as is") and made available for testing and evaluation purposes.
Most binaries are built from source code found on PyPI or in the projects public revision control systems. Source code changes, if any, have been submitted to the project maintainers or are included in the packages.
Refer to the documentation of the individual packages for license restrictions and dependencies.
If downloads fail, reload this page, enable JavaScript, disable download managers, disable proxies, clear cache, use Firefox, reduce number and frequency of downloads. Please only download files manually as needed.
Use pip version 9 or newer to install the downloaded .whl files. This page is not a pip package index.
Many binaries depend on numpy-1.14+mkl and the Microsoft Visual C++ 2008 (x64, x86, and SP1 for CPython 2.7), Visual C++ 2010 (x64, x86, for CPython 3.4), or the Visual C++ 2017 (x64 or x86 for CPython 3.5, 3.6, and 3.7) redistributable packages.
Install numpy+mkl before other packages that depend on it.
The binaries are compatible with the most recent official CPython distributions on Windows >=6.0. Chances are they do not work with custom Python distributions included with Blender, Maya, ArcGIS, OSGeo4W, ABAQUS, Cygwin, Pythonxy, Canopy, EPD, Anaconda, WinPython etc. Many binaries are not compatible with Windows XP or Wine.
The packages are ZIP or 7z files, which allows for manual or scripted installation or repackaging of the content.
The files are provided "as is" without warranty or support of any kind. The entire risk as to the quality and performance is with you.
The opinions or statements expressed on this page should not be taken as a position or endorsement of the Laboratory for Fluorescence Dynamics or the University of California.

➢ Package 索引

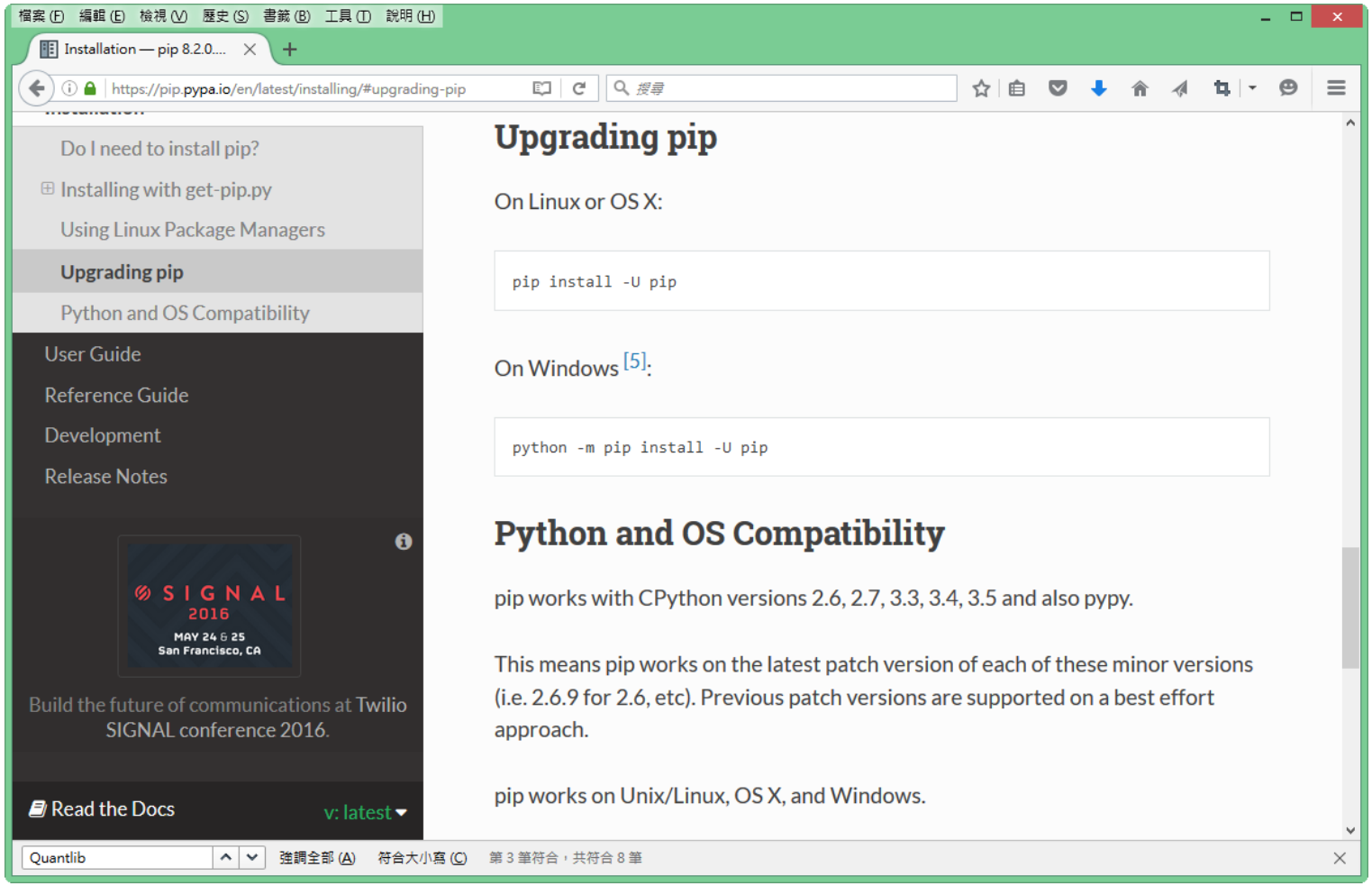

檔案 (F) 編輯 (E) 檢視 (V) 歷史 (S) 書籤 (B) 工具 (T) 說明 (H)
Python Extension Package
https://www.lfd.uci.edu/~gohlke/pythonlibs/#quantlib
133%
Index by date: h5py pymol numpy cftime fasttsne cellprofiler cobra scipy grpcio pyzopfli pymatgen numcodecs pyodeint rasterio reportlab cupy
fiona spacy mkl_fft discretize msgpack cantera imagecodecs tifffile oiffile daal4py multidict arctic bokeh jupyter nlopt numexpr numba llvmlite
rpy2 aggdraw ruamel.yaml fastparquet indexed_gzip pyrsistent regex gdal xgboost vtk cython opencv scikit-learn ecos dulwich spglib cx_freeze
tatsu cartopy mayavi orange coverage netcdf4 pytables psycopg gpy sqlalchemy basemap matplotlib blosc markupsafe pyopengl pycairo cairocffi
numpy-quaternion lz4 mercurial mkl_random openimageio recordclass pyicu fabio fasttext pymcubes ffnet meshpy open3d pulp zope.interface
mod_wsgi pyjnius persistent hyperspy pyopencl brotli xxhash pillow-simd mahotas psf cvxopt jcc sdtfile fmkr fcsfiles cmapfile czifile dnacurve
netpbmfile molmass uciwebauth transformations vidsrc astropy cvxpy yt pystruct logbook twisted pymongo gevent iris param mistune freeimagedll
peewee pip pywin32 gvar zstd ad3 pillow imread python-lzf cmarkgfm tinyarray pylzma cchardet pywavelets swiglpk gensim javabridge
cx_oracle pygit2 iminuit sfepy tornado sympy lxml scikit-image greenlet line_profiler aiohttp fisx sounddevice pytorch openexr chompack fastrlock
pymssql kwant spectrum ets pyodbc psutil protobuf zodbpickle qimage2ndarray python-geohash pyzmq pyamg scandir lsqfit btrees pysqlite pandas
pycuda pendulum moderngl pytiff pygresql wordcloud louvain-igraph python-igraph pycorrfit mplcairo typed_ast rtree winrandom assimulo pyfmi
pyfm fast-histogram boost.python hmmlearn kiwisolver pythonmagick polylearn pythonnet cvxcanon scs pygame blist tensorflow multineat qutip
openbabel noise scikits.vectorplot pyfits nitime zipline zs grako lru_dict scikit-misc pyvrml97 sparsesvd pyephem seqlearn obspy cyordereddict
apsw kivy rapidjson scimath traits chaco enable pyrxp backports.lzma mpi4py bitarray pyyaml slycot bsddb3 python-snappy llist ceodbc liblas
holopy qt_graph_helpers guiqwt veusz pyqwt pyqt4 simplejson pymvpa pymetis mysqlclient pyasn biopython scikit-cycling bsdiff4 multiprocess yarl
entropy openpiv pywinpty stratify cf-units pocketsphinx babel python-ldap python-lzo mlpy milk osqp trollius statsmodels quickfix brotlipy
pgmagick netifaces dipy debug-information-files vitables curses menpo faulthandler thriftpy gmpy marisa-trie bcolz ta-lib ujson py-lmdb simpleitk
jpype pyhdf freesasa pyldap tomopy cytoolz cheetah xylib-py cyrasterize pyswisseph pycluster ode salientdetect liblinear libsvm setproctitle cffi
cdecimal crcmod crc16 pycld2 planar autopy pyx rtmidi-python pycosat pyflux mkl-service postgresadapter datrie polygon py-earth lightning
pystemmer pyqpbo pydde x86cpu fisher pymc hddm heatmap jsonlib intbitset sasl flann scikits.odes pycares pybox2d natgrid pycurl bintrees
pycifrw lp_solve aspell-python vlfd chebyfit akima pykinsol pycvodes fastcache fdint twainmodule triangle scikit-fmm python-levenshtein pyspharm
pyminuit pyhook pyeda pyfltk simpleparse pyaudio thrift atom pyemd enaml shapely pypmc wrf_python quantlib udunits kapteyn libsbml
simpleaudio pylibtiff videocapture pyproj minepy fann2 lazy_object_proxy wrapt bottleneck scikit-umfpack gr pyarrow pyside vigra imaged11
python-cjson py_gd nipy libtfr lfdfiles mathutils yappi pyfftw pyviennacl blz bigfloat cyassimp sima pymca friture pycogent blaze scikits.audiolab
la bazaar dynd genshi python-sundials glumpy pyamf libxml-python cellcognition pymcmc pyksvd pybluez pygraphviz mxbase libpython re2
pymunk pygtk cgal-bindings bio_formats pysfml pyexiv2 pylibdeconv iocbio pymix umysql lazyflow mmlib scikits.timeseries casuarius wxpython
ilastik pywcs scientificpython vpython nmoldyn mmtk pyalembic polymode scikits.delaunay cld py-fcm oursql zfec py2exe pymutt carray llvmpy
cgkit pymedia scipy-cluster scikits.scattpy scikits.samplerate scikits.ann pyxml pytst delny mysql-python htseq pyusb-ftdi silvercity steps pysparse

➢ QuantLib 檔案


檔案 (F) 編輯 (E) 檢視 (V) 歷史 (S) 書籤 (B) 工具 (T) 說明 (H)
Python Extension Package
https://www.lfd.uci.edu/~gohlke/pythonlibs/#quantlib
133%
qt_graph_helpers-0.1.3-cp35-cp35m-win_amd64.whl
qt_graph_helpers-0.1.3-cp36-cp36m-win32.whl
qt_graph_helpers-0.1.3-cp36-cp36m-win_amd64.whl
qt_graph_helpers-0.1.3-cp37-cp37m-win32.whl
qt_graph_helpers-0.1.3-cp37-cp37m-win_amd64.whl
QuantLib, a library for quantitative finance.
QuantLib_Python-1.11-cp27-cp27m-win32.whl
QuantLib_Python-1.11-cp27-cp27m-win_amd64.whl
QuantLib_Python-1.11-cp34-cp34m-win32.whl
QuantLib_Python-1.11-cp34-cp34m-win_amd64.whl
QuantLib_Python-1.11-cp35-cp35m-win32.whl
QuantLib_Python-1.11-cp35-cp35m-win_amd64.whl
QuantLib_Python-1.11-cp36-cp36m-win32.whl
QuantLib_Python-1.11-cp36-cp36m-win_amd64.whl
QuantLib_Python-1.11-cp37-cp37m-win32.whl
QuantLib_Python-1.11-cp37-cp37m-win_amd64.whl
QuickFIX (untested), a Financial Information eXchange (FIX) protocol engine.
quickfix-1.14.3-cp27-none-win32.whl
quickfix-1.14.3-cp27-none-win_amd64.whl
quickfix-1.15.1-cp27-cp27m-win32.whl

◆升級 pip

```
C:\>python -m pip install -U pip
```



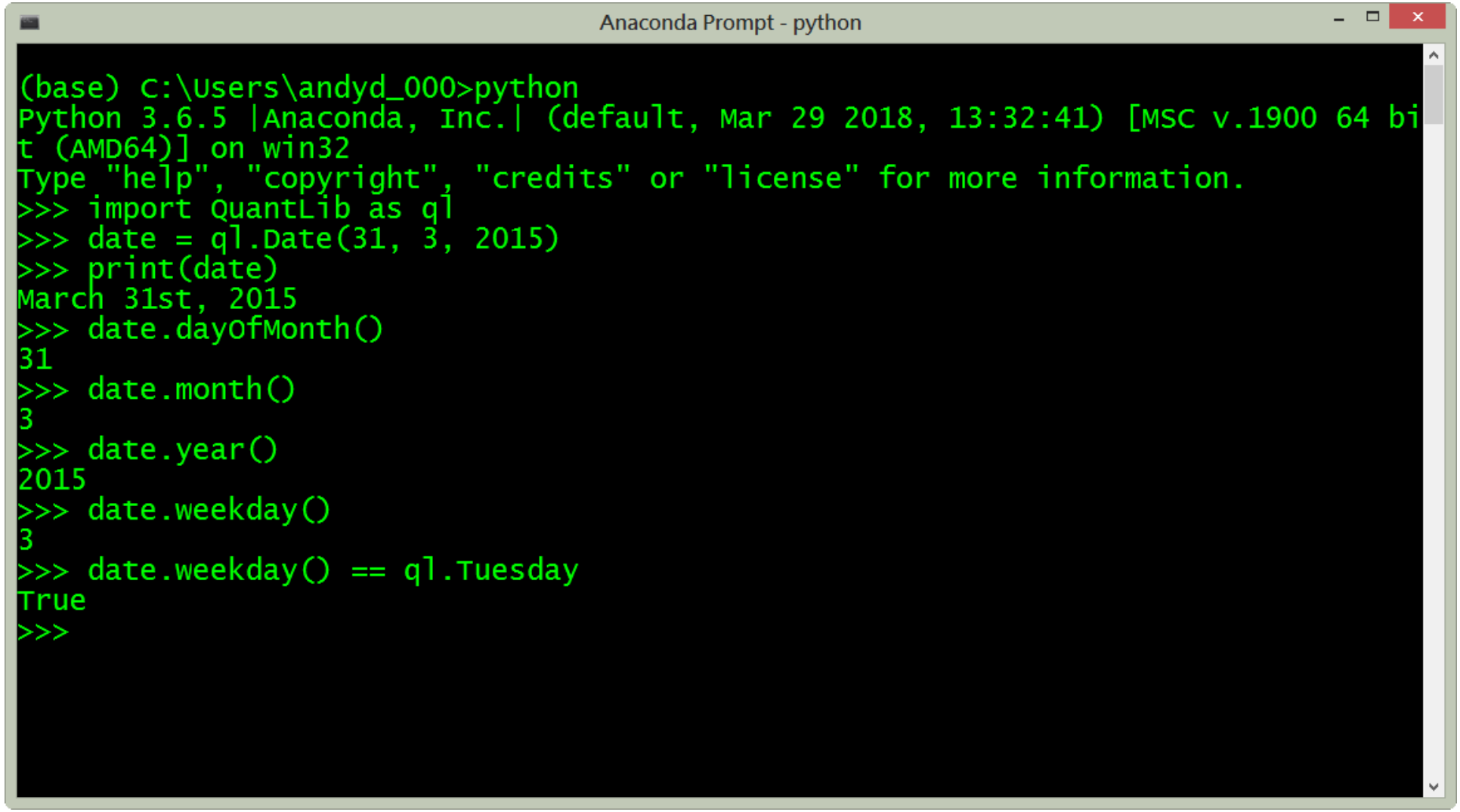
◆安裝 QuantLib_Python Package，whl 格式

```
C:\>pip install QuantLib_Python-1.11-cp36-cp36m-win_amd64.whl
```

◆最簡單方式

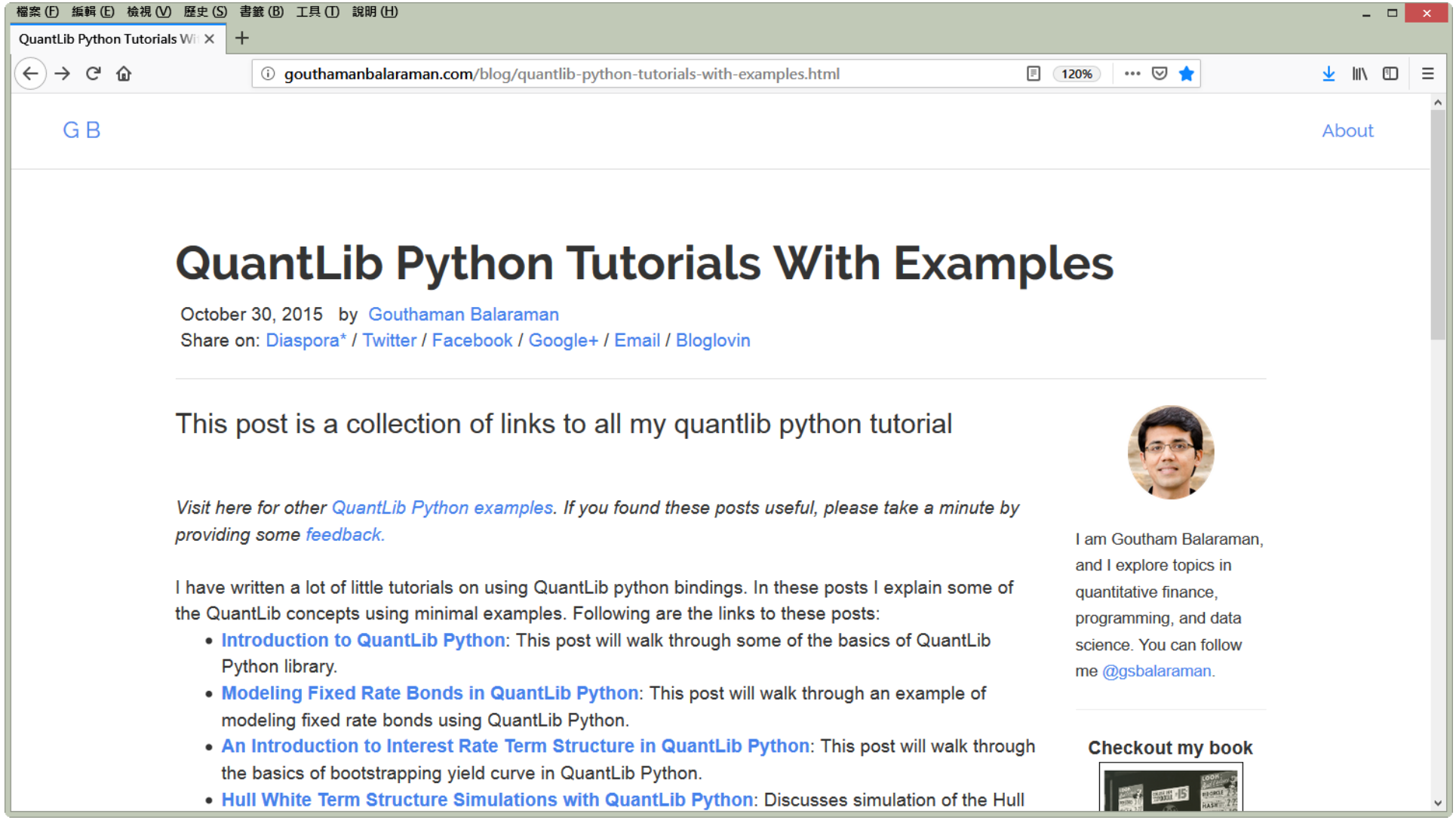
```
C:\pip install QuantLib-Python
```

(八)QuantLib Package 使用測試



◆主要參考網站


檔案 (F) 編輯 (E) 檢視 (V) 歷史 (S) 書籤 (B) 工具 (T) 說明 (H)
Introduction to QuantLib Pyth
gouthamanbalaraman.com/blog/quantlib-basics.html
120%
G B
About
Introduction to QuantLib Python
March 24, 2015 by Goutham Balaraman
Share on: Diaspora* / Twitter / Facebook / Google+ / Email / Bloglovin
This post will walk through some of the basics of QuantLib Python library.
Visit here for other QuantLib Python examples. If you found these posts useful, please take a minute by providing some feedback.
I installed the latest version of QuantLib (V1.5) and the python wrapper to QuantLib. My experiments lately have been to get a feel for the QuantLib API. The library itself is so extensive, that it is rather hard for a new comer to get going. In this post we will look into some of the basic classes and functionality in QuantLib.
Let us import QuantLib as:
import QuantLib as ql
I am Goutham Balaraman, and I explore topics in quantitative finance, programming, and data science. You can follow me @gsbalaraman.
Checkout my book


檔案 (F) 編輯 (E) 檢視 (V) 歷史 (S) 書籤 (B) 工具 (T) 說明 (H)
QuantLib Python Tutorials Wi
gouthamanbalaraman.com/blog/quantlib-python-tutorials-with-examples.html
120%
G B
About
QuantLib Python Tutorials With Examples
October 30, 2015 by Gouthaman Balaraman
Share on: Diaspora* / Twitter / Facebook / Google+ / Email / Bloglovin
This post is a collection of links to all my quantlib python tutorial
Visit here for other QuantLib Python examples. If you found these posts useful, please take a minute by providing some feedback.
I have written a lot of little tutorials on using QuantLib python bindings. In these posts I explain some of the QuantLib concepts using minimal examples. Following are the links to these posts:
Introduction to QuantLib Python: This post will walk through some of the basics of QuantLib Python library.
Modeling Fixed Rate Bonds in QuantLib Python: This post will walk through an example of modeling fixed rate bonds using QuantLib Python.
An Introduction to Interest Rate Term Structure in QuantLib Python: This post will walk through the basics of bootstrapping yield curve in QuantLib Python.
Hull White Term Structure Simulations with QuantLib Python: Discusses simulation of the Hull
I am Goutham Balaraman, and I explore topics in quantitative finance, programming, and data science. You can follow me @gsbalaraman.
Checkout my book

一、QuantLib 基礎類別

◆Date, Period, Calendar 與 Schedule 等類別，

- 在產生工具(Instrument)、模型(Model)與期限結構(Term Structure)方面，成為 QuantLib 的基本構件。

```
In [1]:from QuantLib import *
       import pandas as pd
```

(一)Date & Period Class

◆日期物件以建構子 Date(day, month, year)產生。

- 與 Python datetime 物件實例化不同，先日、中月、後年。

```
In [2]: date = Date(31, 3, 2015)

        print(date)

Out[2]: March 31st, 2015
```

◆可以 month(), dayOfMonth() and year()等方法取得其欄位。

- weekday()方法取得其星期值。

```
In [3]: print("%d-%d-%d" %(date.month(), date.dayOfMonth(), date.year()))

Out[3]: 3-31-2015

In [4]: date.weekday() == Tuesday

Out[4]: True
```

◆Date 物件可以進行算數運算，如前進日、周、月等，

◆如星期、月等周期時間，可以 Period 物件表示，

- 周期物件建構子為 Period(num_periods, period_type)。
  - ✓ num_periods 為整數，表示周期數目，
  - ✓ period_type 表周期單位，可為 Weeks, Months and Years。

```
In [5]: type(date+1)
Out[5]: QuantLib.QuantLib.Date
In [6]: print("Add a day : {0}".format(date + 1))
        print("Subtract a day : {0}".format(date - 1))
        print("Add a week : {0}".format(date + Period(1, Weeks)))
        print("Add a month : {0}".format(date + Period(1, Months)))
        print("Add a year : {0}".format(date + Period(1, Years)))
```

```
Out[6]:Add a day : April 1st, 2015
       Subtract a day : March 30th, 2015
       Add a week : April 7th, 2015
       Add a month : April 30th, 2015
       Add a year : March 31st, 2016
```

◆the Date object 可以進行邏輯運算

```
In [7]: print(date == Date(31, 3, 2015))
        print(date > Date(30, 3, 2015))
        print(date < Date(1, 4, 2015))
        print(date != Date(1, 4, 2015))
Out[7]: True
        True
        True
        True
```

◆Date 物件用於設定評價日，工具的發行日與到期日。

◆Period 物件用於設定期限(tenors)，例如債息的支付或建立支付時程(payment schedules)。

(二)Calendar Class

◆上述 Date 運算沒有考慮 holidays。

- 然而不同工具需考慮特定交易所與國家的假日。Calendar 類別實作主要交易所得這些功能。

```
In [8]: date = Date(31, 3, 2015)
        us_calendar = UnitedStates()
        italy_calendar = Italy()
        period = Period(60, Days)
        raw_date = date + period
        us_date = us_calendar.advance(date, period)
        italy_date = italy_calendar.advance(date, period)
        print("Add 60 days: {0}".format(raw_date))
        print("Add 60 business days in US: {0}".format(us_date))
        print("Add 60 business days in Italy: {0}".format(italy_date))
Out[8]: Add 60 days: May 30th, 2015
        Add 60 business days in US: June 24th, 2015
        Add 60 business days in Italy: June 26th, 2015
```

◆addHoliday 與 removeHoliday 方法可用來加入或移除特定日曆的假日。

◆businessDaysBetween 方法可用來計算兩日期間的假日數目。

```
In [9]: us_busdays = us_calendar.businessDaysBetween(date, us_date)
        italy_busdays = italy_calendar.businessDaysBetween(date, italy_date)
        print("Business days US: {0}".format(us_busdays))
        print("Business days Italy: {0}".format(italy_busdays))
Out[9]: Business days US: 60
        Business days Italy: 60
```

◆特定交易評價時需觀察一個以上的日曆，

- QuantLib 有 JointCalendar 類別，合併兩個以上的日曆。

```
In [10]: joint_calendar = JointCalendar(us_calendar, italy_calendar)
         joint_date = joint_calendar.advance(date, period)
         joint_busdays = joint_calendar.businessDaysBetween(date, joint_date)
         print("Add 60 business days in US-Italy: {0}".format(joint_date))
         print("Business days US-Italy: {0}".format(joint_busdays))
Out[10]: Add 60 business days in US-Italy: June 29th, 2015
         Business days US-Italy: 60
```

(三)Schedule類別

◆Schedule 物件在產生債息與贖回時程上是必要的工具。有下述的建構子：

```
Schedule(const Date& effectiveDate, const Date& terminationDate, const Period& tenor,
    const Calendar& calendar, BusinessDayConvention convention,
    BusinessDayConvention terminationDateConvention, DateGeneration::Rule rule,
    bool endOfMonth, const Date& firstDate = Date(), const Date& nextToLastDate = Date())

Schedule(const std::vector<Date>&, const Calendar& calendar,
    BusinessDayConvention rollingConvention)
```

```
In [11]: effective_date = Date(1, 1, 2015)

         termination_date = Date(1, 1, 2016)

         tenor = Period(Monthly)

         calendar = UnitedStates()

         business_convention = Following

         termination_business_convention = Following

         date_generation = DateGeneration.Forward

         end_of_month = False

         schedule = Schedule(effective_date, termination_date, tenor, calendar,
             business_convention, termination_business_convention, date_generation,
             end_of_month)

         pd.DataFrame({'date': list(schedule)})
```

```
Out[11]:
```

	date
0	January 2nd, 2015
1	February 2nd, 2015
2	March 2nd, 2015
3	April 1st, 2015
4	May 1st, 2015
5	June 1st, 2015
6	July 1st, 2015
7	August 3rd, 2015
8	September 1st, 2015
9	October 1st, 2015
10	November 2nd, 2015
11	December 1st, 2015
12	January 4th, 2016

◆Schedule 物件包含介於 effective_date 與 termination_date 間的日期，tenor 說明週期 Period 為 Monthly。

- calendar 物件用來決定 holidays。
- 此處使用慣例為 following。因此假日被排除於日期中。

◆Schedule 類別可以處理不規律的日期產生。

- 兩個額外的參數，firstDate 與 nextToLastDate，搭配 forward 與 backward 日期產生規則，可以產生 short 或 long stub 的支付時程。

◆例如，下述 firstDate 與 backward 產生規則，產生一個 January 15, 2015 為前端的 short stub。

```
In [12]: # short stub in the front
         effective_date = Date(1, 1, 2015)
         termination_date = Date(1, 1, 2016)
         first_date = Date(15, 1, 2015)
         schedule = Schedule(effective_date, termination_date, tenor, calendar,
             business_convention, termination_business_convention, DateGeneration.Backward,
             end_of_month, first_date)
         pd.DataFrame({'date': list(schedule)})
Out[12]:
         date
         0 January 2nd, 2015
         1 January 15th, 2015
```

```
2 February 2nd, 2015
3 March 2nd, 2015
4 April 1st, 2015
5 May 1st, 2015
6 June 1st, 2015
7 July 1st, 2015
8 August 3rd, 2015
9 September 1st, 2015
10 October 1st, 2015
11 November 2nd, 2015
12 December 1st, 2015
13 January 4th, 2016
```

◆使用 nextToLastDate 參數配合 forward 產生規則，產生一個在時程尾端的 short stub。

```
In [13]: # short stub at the back
         effective_date = Date(1, 1, 2015)
         termination_date = Date(1, 1, 2016)
         penultimate_date = Date(15, 12, 2015)
         schedule = Schedule(effective_date, termination_date, tenor, calendar,
             business_convention, termination_business_convention, DateGeneration.Forward,
             end_of_month, Date(), penultimate_date)
         pd.DataFrame({'date': list(schedule)})
```

Out[13]:

	date
0	January 2nd, 2015
1	February 2nd, 2015
2	March 2nd, 2015
3	April 1st, 2015
4	May 1st, 2015
5	June 1st, 2015
6	July 1st, 2015

```
7 August 3rd, 2015
8 September 1st, 2015
9 October 1st, 2015
10 November 2nd, 2015
11 December 1st, 2015
12 December 15th, 2015
13 January 4th, 2016
```

◆使用 backward 產生規則，搭配 firstDate，允許我們在前端產生一個 long stub。

```
In [14]: # long stub in the front
         first_date = Date(1, 2, 2015)
         effective_date = Date(15, 12, 2014)
         termination_date = Date(1, 1, 2016)
        schedule = Schedule(effective_date, termination_date, tenor, calendar,
            business_convention, termination_business_convention, DateGeneration.Backward,
            end_of_month, first_date)
        pd.DataFrame({'date': list(schedule)})
Out[14]:
        date
        0 December 15th, 2014
        1 February 2nd, 2015
        2 March 2nd, 2015
        3 April 1st, 2015
        4 May 1st, 2015
        5 June 1st, 2015
        6 July 1st, 2015
```

```
7 August 3rd, 2015
8 September 1st, 2015
9 October 1st, 2015
10 November 2nd, 2015
11 December 1st, 2015
12 January 4th, 2016
```

◆使用 nextToLastDate 參數與 forward 日期產生規則，可以用來在尾端產生 long stub 的時程。

```
In [15]: # long stub at the back
         effective_date = Date(1, 1, 2015)
         penultimate_date = Date(1, 12, 2015)
         termination_date = Date(15, 1, 2016)
         schedule = Schedule(effective_date, termination_date, tenor, calendar,
             business_convention, termination_business_convention, DateGeneration.Forward,
             end_of_month, Date(), penultimate_date)
         pd.DataFrame({'date': list(schedule)})
Out[15]:
         date
         0 January 2nd, 2015
         1 February 2nd, 2015
         2 March 2nd, 2015
         3 April 1st, 2015
         4 May 1st, 2015
         5 June 1st, 2015
```

```
6 July 1st, 2015
7 August 3rd, 2015
8 September 1st, 2015
9 October 1st, 2015
10 November 2nd, 2015
11 December 1st, 2015
12 January 15th, 2016
```

◆亦可由一串日期來產生 Schedule。

```
In [16]: dates = [Date(2,1,2015), Date(2, 2,2015), Date(2,3,2015), Date(1,4,2015), Date(1,5,2015),
                  Date(1,6,2015), Date(1,7,2015), Date(3,8,2015), Date(1,9,2015), Date(1,10,2015),
                  Date(2,11,2015), Date(1,12,2015), Date(4,1,2016)]

         rolling_convention = Following schedule = Schedule(dates, calendar, rolling_convention)

         pd.DataFrame({'date': list(schedule)})

Out[16]:

         date

         0 January 2nd, 2015

         1 February 2nd, 2015

         2 March 2nd, 2015

         3 April 1st, 2015

         4 May 1st, 2015

         5 June 1st, 2015

         6 July 1st, 2015

         7 August 3rd, 2015

         8 September 1st, 2015

         9 October 1st, 2015
```

```
10 November 2nd, 2015
11 December 1st, 2015
12 January 4th, 2016
```

(四)Interest Rate

◆InterestRate 類別可用來儲存利率與複利類型(compounding type)，計日方式(day count)與複利頻率(frequency of compounding)。

➢ 下例為 5.0%利率，年複利，使用 Actual/Actual 計日方式。

```
In [17]: annual_rate = 0.05
         day_count = ActualActual()
         compound_type = Compounded
         frequency = Annual
         interest_rate = InterestRate(annual_rate, day_count, compound_type, frequency)
         print(interest_rate)
Out[17]: 5.000000 % Actual/Actual (ISDA) Annual compounding
```

◆compound_factor 表示複利次數，下例為 2 次，因頻率為年，表示 2 年投資。

```
In [18]: t = 2.0
         print(interest_rate.compoundFactor(t))
         print((1+annual_rate)*(1.0+annual_rate))
Out[18]: 1.1025
         1.1025
```

◆discountFactor 提供 compoundFactor 的倒數。可以用來算現值。

```
In [19]: print(interest_rate.discountFactor(t))
         print(1.0/interest_rate.compoundFactor(t))
Out[19]: 0.9070294784580498
         0.9070294784580498
```

◆一個利率可以使用 equivalentRate 方法，轉換為其他複利方式與複利頻率的利率。

```
In [20]: new_frequency = Semiannual
         new_interest_rate = interest_rate.equivalentRate(compound_type, new_frequency, t)
         print(new_interest_rate)
Out[20]: 4.939015 % Actual/Actual (ISDA) Semiannual compounding
```

◆interest_rate 與 new_interest_rate 的折現因子皆相同。

```
In [21]: print(interest_rate.discountFactor(t))
         print(new_interest_rate.discountFactor(t))
Out[21]: 0.9070294784580498
         0.9070294784580495
```

◆InterestRate 類別的 impliedRate 方法，接受 compound factor 為輸入，傳出 implied rate。

impliedRate 方法是靜態方法。equivalentRate 方法有喚起 impliedRate 方法，進行計算。

二、金融工具與選擇權評價引擎

(一)Setup

◆載入 QuantLib 模組，設定全域評價日。

```
In [1]: from QuantLib import *

In [2]: today = Date(7, March, 2014)

        Settings.instance().evaluationDate = today
```

(二)The instrument

◆以歐式選擇權為工具的範例，建立一個選擇權契約只需要，

- 它的 payoff (一個 call 選擇權，執行價格 strike 為 100)
- 三個月後到期的歐式執行日期
- 市場資料之後傳入

```
In [3]: option = EuropeanOption(PlainVanillaPayoff(Option.Call, 100.0),
                                EuropeanExercise(Date(7, June, 2014)))
```

(三)第一種定價方法：Black-Scholes解析公式

◆收集市場資料。

- 使用 SimpleQuote 來包裝市場資料，標的價格為 100，無風險利率 1%，波動性為 20%。

```
In [4]: u = SimpleQuote(100.0)
        r = SimpleQuote(0.01)
        sigma = SimpleQuote(0.20)
```

◆將報價資料包裝成 Black-Scholes process 的物件。

- 首先，建立水平的利率與波動性曲線。

```
In [5]: riskFreeCurve = FlatForward(0, TARGET(), QuoteHandle(r), Actual360())
        volatility = BlackConstantVol(0, TARGET(), QuoteHandle(sigma), Actual360())
```

◆然後，實例化程序

- 需要 underlying value 與 curves。
- 數入資料儲存在 handles 中，方便之後的改變。

```
In [6]: process = BlackScholesProcess(QuoteHandle(u), YieldTermStructureHandle(riskFreeCurve),
        BlackVolTermStructureHandle(volatility))
```

◆有了程序，便可建構 engine。

```
In [7]: engine = AnalyticEuropeanEngine(process)
```

◆將引擎設定給 option，便可計算。

```
In [8]: option.setPricingEngine(engine)
In [9]: print(option.NPV())
Out[9]: 4.155543462156206
```

◆根據工具與引擎，可以要求其他結果，此例為 Greeks.

```
In [10]: print(option.delta())
         print(option.gamma())
         print(option.vega())
Out[10]: 0.5302223303784392
         0.03934493301271913
         20.109632428723106
```

(四)Market changes

◆市場資料儲存於 Quote 的實例之中，當它有異動時會自行通知選擇權。

- 我們無須明白告知該選擇權需要重新計算，一旦我們重設新值給標的資產，只需再向選擇權要 NPV，便可得到異動的新價值。

```
In [11]: u.setValue(105.0)
         print(option.NPV())
Out[11]: 7.27556357927846
```

◆為求顯示此效果，可借此畫出選擇權價值對標的資產價格的作圖。

```
In [12]: %matplotlib inline
         import numpy as np
         import matplotlib.pyplot as plt
```

◆取用 80 到 120 的陣列，將其設定標的資產的價值，收集相對應的選擇權價值，畫出結果。

```
In [13]: plt.style.use("seaborn-whitegrid")
         fig = plt.figure()
         ax = plt.axes()
         plt.title("Black-Scholes Option Value Curve")
         plt.xlabel("Underlying Value")
         plt.ylabel("Call Value")
         xs = np.linspace(80.0, 120.0, 400)
         ys = []
         for x in xs:
            u.setValue(x)
            ys.append(option.NPV())
         plt.plot(xs, ys)
         plt.show()
```

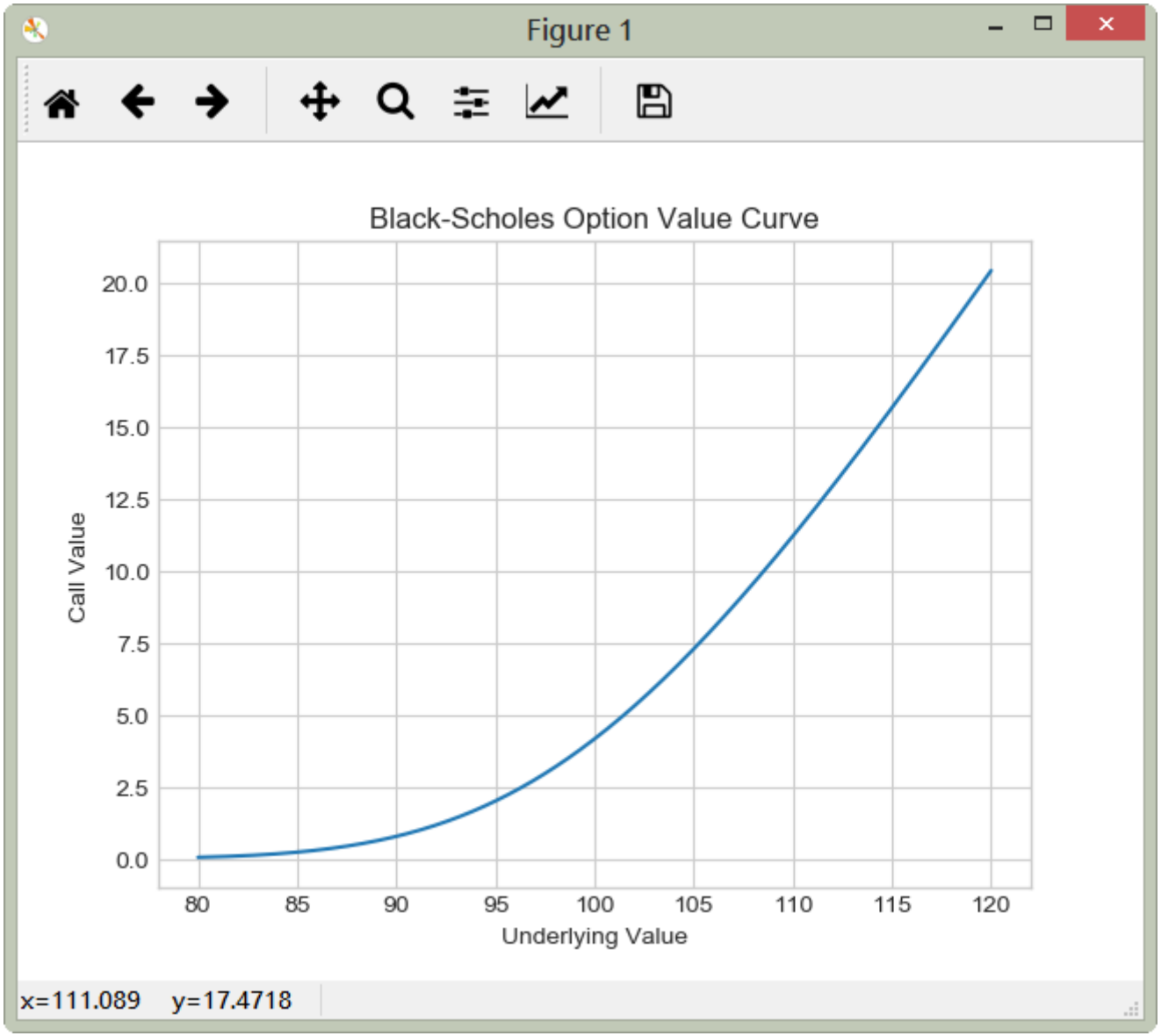


Figure 1
Black-Scholes Option Value Curve
Call Value
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
80
85
90
95
100
105
110
115
120
Underlying Value
x=111.089 y=17.4718

◆其他市場資料亦會影響價值。

```
In [14]: u.setValue(105.0)
         r.setValue(0.01)
         sigma.setValue(0.20)
In [15]: print(option.NPV())
Out[15]: 7.27556357927846
```

◆無風險利率的效果

```
In [16]: r.setValue(0.03)
In [17]: print(option.NPV())
Out[17]: 7.624029148527754
```

◆波動性的效果

```
In [18]: sigma.setValue(0.25)
In [19]: print(option.NPV())
Out[19]: 8.531296969971573
```

(五)Date變動

◆我們也可推進評價日期，看看價值的變動。首先，三個月後到期，標的資產 105。

```
In [20]: u.setValue(105.0)
         r.setValue(0.01)
         sigma.setValue(0.20)
         print(option.NPV())
Out[20]: 7.27556357927846
```

◆推進一個月評價。

```
In [21]: Settings.instance().evaluationDate = Date(7, April, 2014)
In [22]: print(option.NPV())
Out[22]: 6.560073820974377
```

◆繪圖示之。

```
In [23]: ys = []
         for x in xs:
             u.setValue(x)
             ys.append(option.NPV())
         plt.plot(xs, ys, '--')
         plt.show()
```

◆在到期日時，傳回價值為預設值：0。

```
In [24]: Settings.instance().evaluationDate = Date(7, June, 2014)
In [25]: print(option.NPV())
Out[25]: 0.0
```

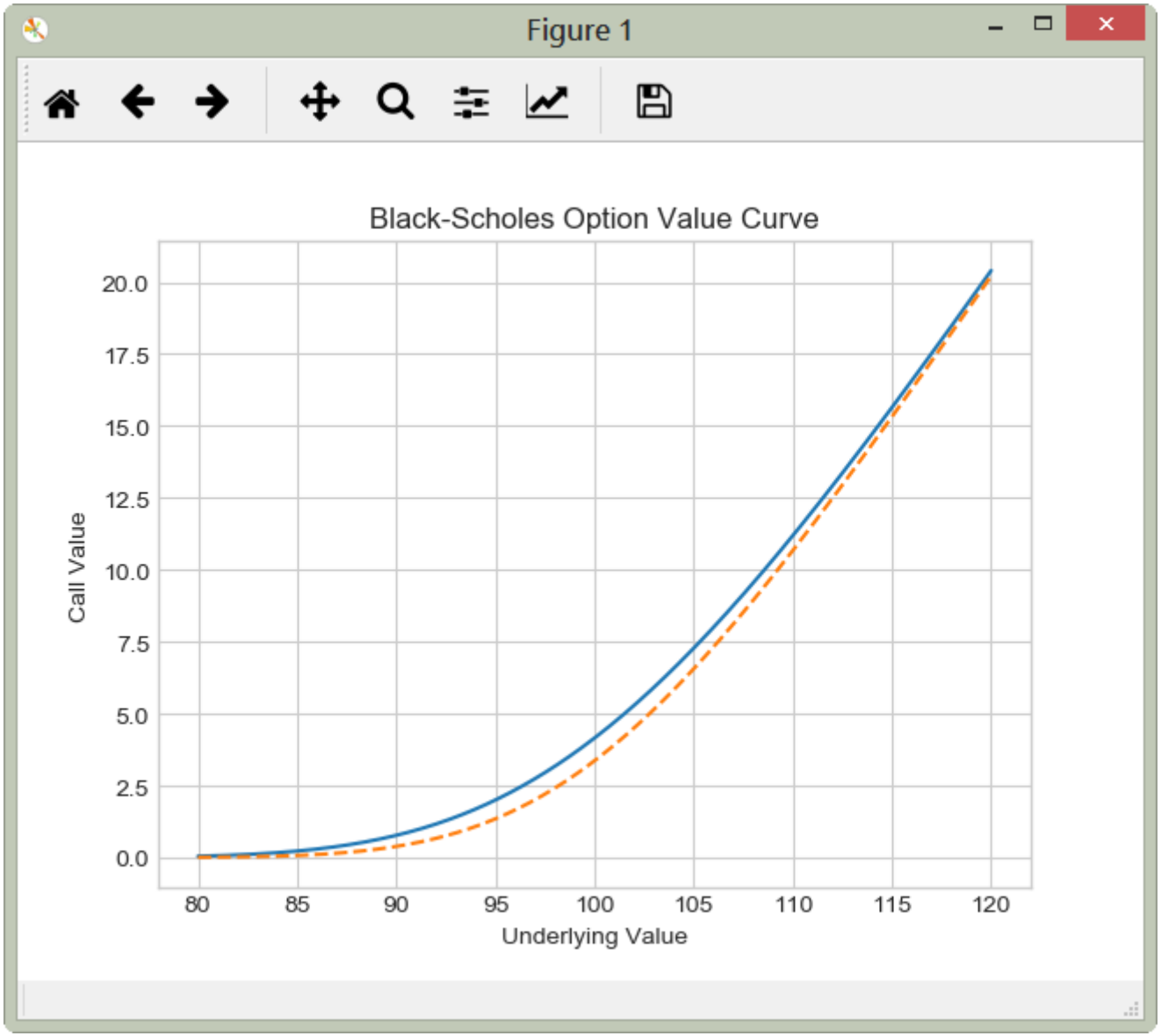


Figure 1
Black-Scholes Option Value Curve
Call Value
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
80
85
90
95
100
105
110
115
120
Underlying Value

(六)Other pricing methods

◆pricing-engine 機制允許我們使用不同評價方法，首先推回原來資料。

```
In [26]: Settings.instance().evaluationDate = today
         u.setValue(105.0)
         r.setValue(0.01)
         sigma.setValue(0.20)
In [27]: print(option.NPV())
Out[27]: 7.27556357927846
```

◆其，使用 Heston model 來進行評價。實例化一個相應的類別，輸入想要的參數。

```
In [28]: model = HestonModel(HestonProcess(YieldTermStructureHandle(riskFreeCurve),
           YieldTermStructureHandle(FlatForward(0, TARGET(), 0.0, Actual360())),
           QuoteHandle(u), 0.04, 0.1, 0.01, 0.05, -0.75))
```

◆傳給對應引擎，並設定給選擇權。

```
In [29]: engine = AnalyticHestonEngine(model)
         option.setPricingEngine(engine)
```

◆根據新模型，計算選擇權 NPV。

```
In [30]: print(option.NPV())
Out[30]: 7.295356086978629
```

三、Greeks 數值計算

(一)Setup

◆導入 QuantLib 模組，設定評價日。

```
In [1]: from QuantLib import *

In [2]: today = Date(8, October, 2014)

        Settings.instance().evaluationDate = today
```

(二)較複雜的Exotic Option

◆使用 knock-in barrier option 阻隔選擇權。

```
In [3]: option = BarrierOption(Barrier.UpIn, 120.0, # barrier
        0.0, # rebate
        PlainVanillaPayoff(Option.Call, 100.0),
        EuropeanExercise(Date(8, January, 2015)))
```

◆市場資料為 u，r，sigma。包入報價 Quote 中。

```
In [4]: u = SimpleQuote(100.0)
        r = SimpleQuote(0.01)
        sigma = SimpleQuote(0.20)
```

◆建立 flat curves 與 process 供 engine 使用。建立 term structures，使其可與評價日一並移動。

```
In [5]: riskFreeCurve = FlatForward(0, TARGET(), QuoteHandle(r), Actual360())
        volatility = BlackConstantVol(0, TARGET(), QuoteHandle(sigma), Actual360())
In [6]: process = BlackScholesProcess(QuoteHandle(u), YieldTermStructureHandle(riskFreeCurve),
        BlackVolTermStructureHandle(volatility))
```

◆最後，建立 engine(使用解析公式)設給 option。

```
In [7]:  option.setPricingEngine(AnalyticBarrierEngine(process))
```

◆可以計算價格。

```
In [8]: print(option.NPV())
Out[8]: 1.3657980739109867
```

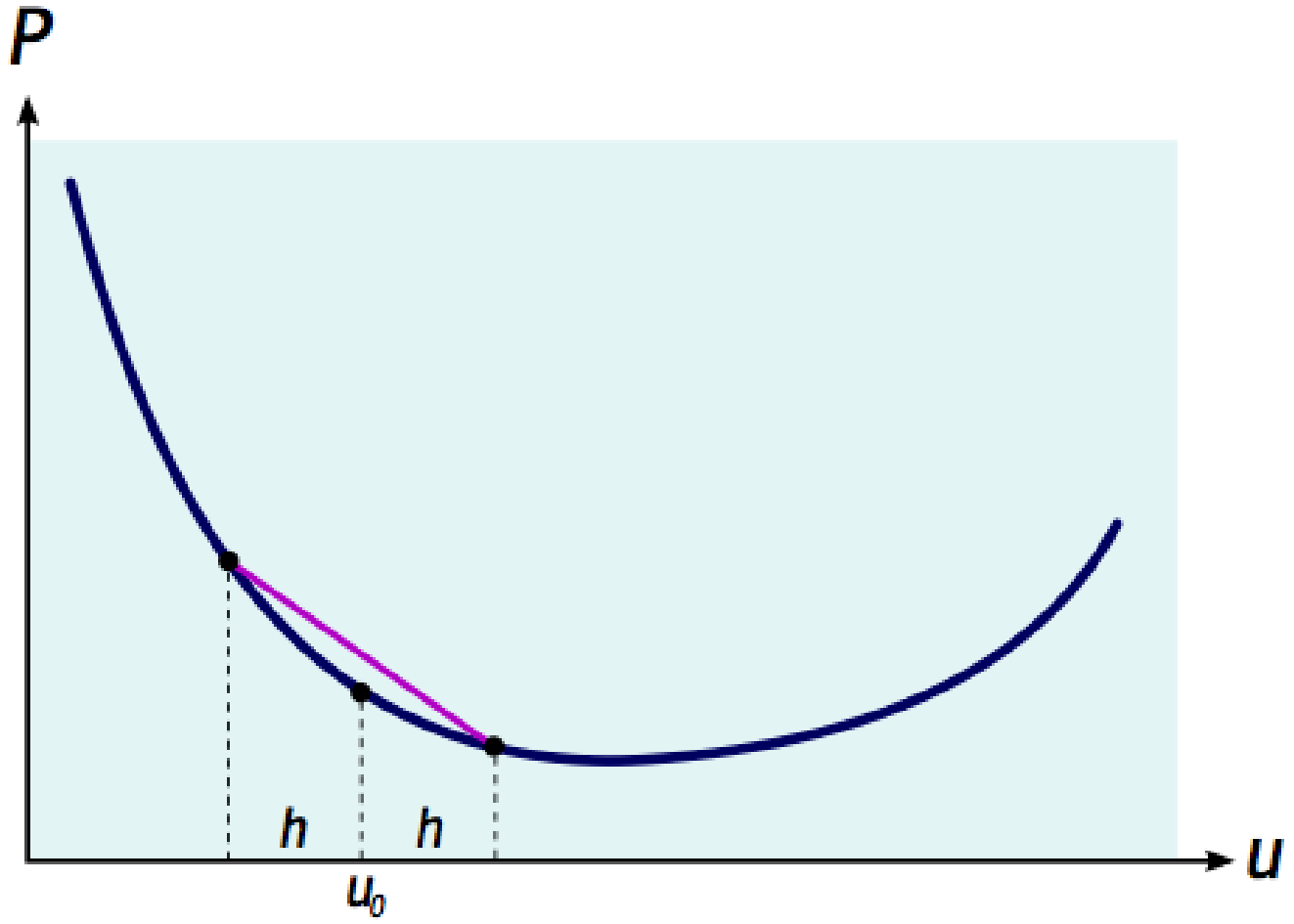
◆如果要計算 Greeks，會出現問題。

```
In [9]: print(option.delta())

Out[9]: ---------------------------------------------------------------------------
        RuntimeError Traceback (most recent call last)

        <ipython-input-9-dcaa26b2b456> in <module>()----> 1 print(option.delta())

        /usr/local/lib/python3.6/dist-packages/QuantLib/QuantLib.py in delta(self)
        11432
        11433 def delta(self):
        > 11434 return _QuantLib.BarrierOption_delta(self)
        11435
        11436 def gamma(self):

        RuntimeError: delta not provided
```

◆此 engine 不提供 delta，故產生 error。

(三)數值計算

◆quant 如何解決? 使用數值微分來近似。



◆近似公式為：

$$\Delta = \frac{P(u_0+h) - P(u_0-h)}{2h} \qquad \Gamma = \frac{P(u_0+h) - 2P(u_0) + P(u_0-h)}{h^2}$$

◆首先，紀錄當前價值。

```
In [10]: u0 = u.value() ; h = 0.01

In [11]: P0 = option.NPV() ; print(P0)

Out[11]: 1.3657980739109867
```

◆其次，增加 underlying 價格，取得價值。

```
In [12]: u.setValue(u0+h)

         P_plus = option.NPV() ; print(P_plus)

Out[12]: 1.3688112201958083
```

◆再次，減少 underlying 價格。

```
In [13]: u.setValue(u0-h)
         P_minus = option.NPV() ; print(P_minus)
Out[13]: 1.3627900998610207
```

◆最後，設定 underlying value 回目前價值。

```
In [14]: u.setValue(u0)
```

◆使用公式，計算 Greeks。

```
In [15]: Delta = (P_plus - P_minus)/(2*h)
         Gamma = (P_plus - 2*P0 + P_minus)/(h*h)
         print(Delta)
         print(Gamma)
Out[15]: 0.3010560167393761
         0.05172234855521651
```

◆此法適用於任何的 Greek。

$$\frac{\partial P}{\partial x} = \frac{P(x_0 + h) - P(x_0)}{h}$$

◆舉例，Rho 與 Vega：

```
In [16]: r0 = r.value() ; h = 0.0001
         r.setValue(r0+h) ; P_plus = option.NPV()
         r.setValue(r0)
         Rho = (P_plus - P0)/h ; print(Rho)
Out[16]: 6.531038494277386
In [17]: sigma0 = sigma.value() ; h = 0.0001
         sigma.setValue(sigma0+h) ; P_plus = option.NPV()
         sigma.setValue(sigma0)
         Vega = (P_plus - P0)/h ; print(Vega)
Out[17]: 26.52519924198904
```

◆Theta 的計算稍有不同。

```
In [18]: Settings.instance().evaluationDate = today+1
         P1 = option.NPV()
         h = 1.0/365
         Theta = (P1-P0)/h ; print(Theta)
Out[18]: -10.770888399441302
```

四、市場報價與債券價格

◆此例中，將顯示如何避免當多重報價需要更新時，可能產生的陷阱。

```
In [1]: import numpy as np

In [2]: from QuantLib import *
        import matplotlib.pyplot as plt
In [3]: today = Date(17, October, 2016)
        Settings.instance().evaluationDate = today
```

(一)Setting the stage

◆配合 QuantLib C++範例，產生一個 Bond Curve，

- Nelson-Siegel model 配湊出的資料，下面為到期年限與債息利率。

```
In [4]: data = [ (2, 0.02), (4, 0.0225), (6, 0.025), (8, 0.0275), (10, 0.03), (12, 0.0325),
                 (14, 0.035), (16, 0.0375), (18, 0.04), (20, 0.0425), (22, 0.045), (24, 0.0475),
                 (26, 0.05), (28, 0.0525), (30, 0.055)]
```

◆使用相同起始日、頻率、慣例。所有債券皆價格 100。

```
In [5]: calendar = TARGET()
        settlement = calendar.advance(today, 3, Days)
        quotes = []
        helpers = []
        for length, coupon in data:
            maturity = calendar.advance(settlement, length, Years)
            schedule = Schedule(settlement, maturity, Period(Annual), calendar,
                ModifiedFollowing, ModifiedFollowing, DateGeneration.Backward, False)
            quote = SimpleQuote(100.0)
```

```
        quotes.append(quote)
        helpers.append(FixedRateBondHelper(QuoteHandle(quote), 3, 100.0, schedule, [coupon],
            SimpleDayCounter(), ModifiedFollowing))
    curve = FittedBondDiscountCurve(0, calendar, helpers, SimpleDayCounter(),
        NelsonSiegelFitting())
```

◆下圖為折現因子對時間做圖，以年為單位。

```
In [6]: sample_times = np.linspace(0.0, 30.0, 301)
        sample_discounts = [ curve.discount(t) for t in sample_times ]
        fig = plt.figure()
        plt.title("Discount Function Curve of Bond")
        plt.xlabel("Time in Years")
        plt.ylabel("Discount Function")
        plt.style.use('seaborn-whitegrid')
        xs = sample_times
        ys = sample_discounts
        plt.plot(xs, ys)
        plt.show()
```

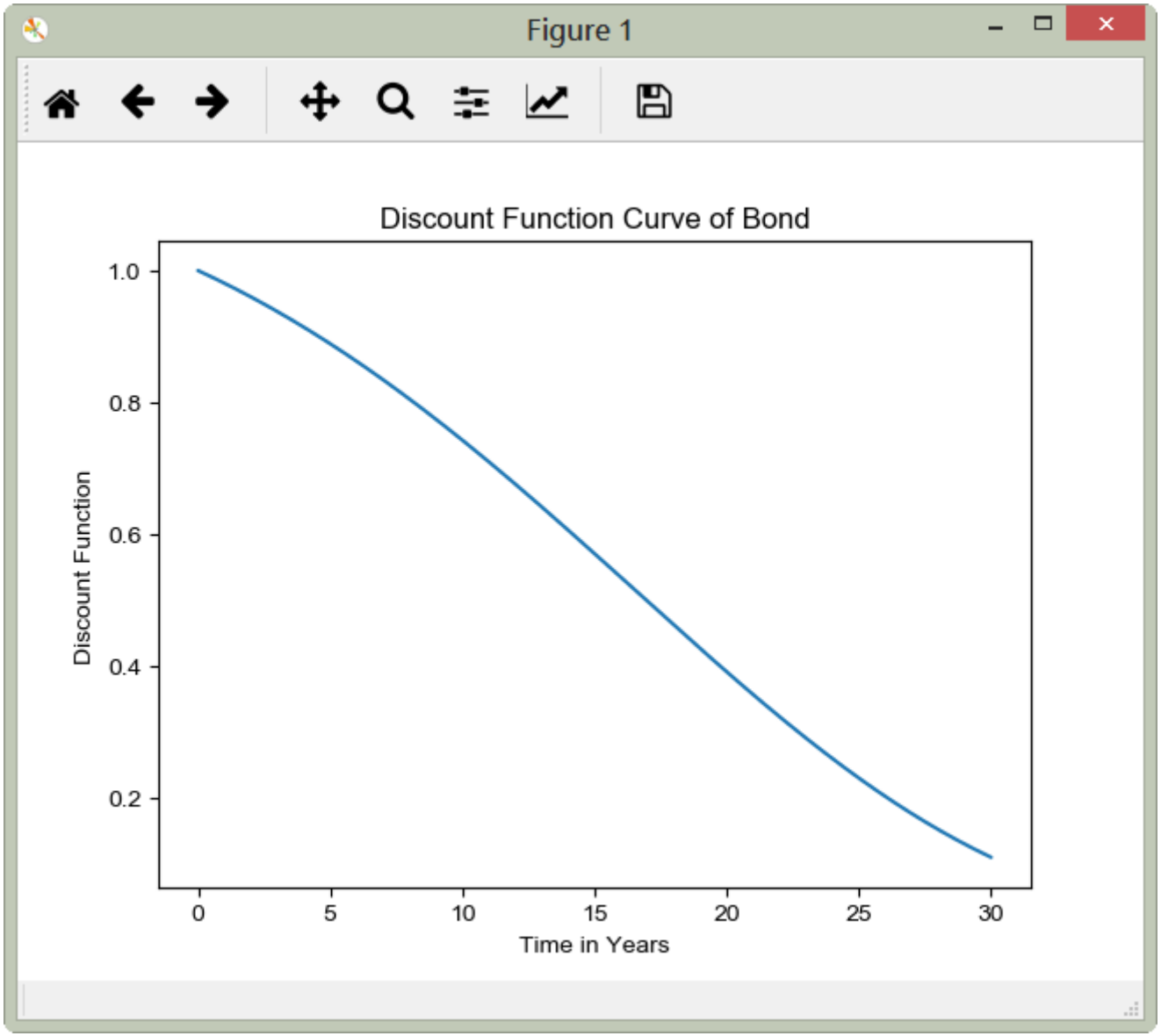


Figure 1
Discount Function Curve of Bond
Discount Function
1.0
0.8
0.6
0.4
0.2
0
5
10
15
20
25
30
Time in Years

◆使用此 Cureve，計算一債券的價格。

- 三年到期，100 元面值，4%債息，Actual/360 計息方式。

```
In [7]: schedule = Schedule(today, calendar.advance(today, 15, Years), Period(Semiannual),
            calendar, ModifiedFollowing, ModifiedFollowing, DateGeneration.Backward, False)
        bond = FixedRateBond(3, 100.0, schedule, [0.04], Actual360())
        bond.setPricingEngine(DiscountingBondEngine(YieldTermStructureHandle(curve)))
        print(bond.cleanPrice())
Out[7]: 105.77449628297312
```

(二) “It looked like a good idea at the time”

◆新增一觀察者，檢查債券是否過期，如果過期重算價格並輸出之。

- 在 Python 中，可定義一個被通知觸發的函數，此通知被傳遞給此觀察者，且將觀察者註冊此債券。
- 市場報價的任何變動，會通知 the helper，然後通知 the curve，然後通之 the pricing engine，the bond，最後我們的觀察者。

```
In [8]: prices = []
        def print_price():
            p = bond.cleanPrice()
            prices.append(p)
            print(p)
        o = Observer(print_price)
        o.registerWith(bond)
```

◆此函數會附加新價格到 List，作為之後價格歷史之用，看其作用。

```
In [9]: quotes[2].setValue(101.0)
Out[9]: 105.77449628297312
        105.8656042875337
```

◆此函數被呼叫兩次，這是由於多重繼承的毛病所造成的。Curve 發送兩次通知給工具。第一次，工具重算，但 Curve 沒有(因此價格不便)；第二次後，Curve 異動，價格改變。

- 此問題未來應修正，將價格改回來。

```
In [10]: quotes[2].setValue(100.0)
Out[10]: 105.8656042875337
         105.77449634664224
```

◆假設市場改變，債券價格皆走高至 101。異動所有報價。

```
In [11]: prices = []
         for q in quotes:
             q.setValue(101.0)
```

```
Out[11]: 105.77449634664224
         105.28388426272507
         105.28388426272507
         105.2186288679219
         105.2186288679219
         105.3195906444377
         105.3195906444377
         105.4878663448759
         105.4878663448759
         105.68032070200927
         105.68032070200927
         105.87580370787278
         105.87580370787278
         106.06201680440225
         106.06201680440225
         106.23044624497663
         106.23044624497663
         106.37409230798896
```

```
106.37409230798896
106.48708840758337
106.48708840758337
106.56505206364592
106.56505206364592
106.60570726105742
106.60570726105742
106.60980187075381
106.60980187075381
106.58011186582736
106.58011186582736
106.52070699740128
```

◆每一個異動發送通知，觸發重算。畫圖如下。

```
In [12]: unique_prices = prices[::2]+prices[-1::]
         plt.plot(unique_prices, '-');
         plt.show()
```

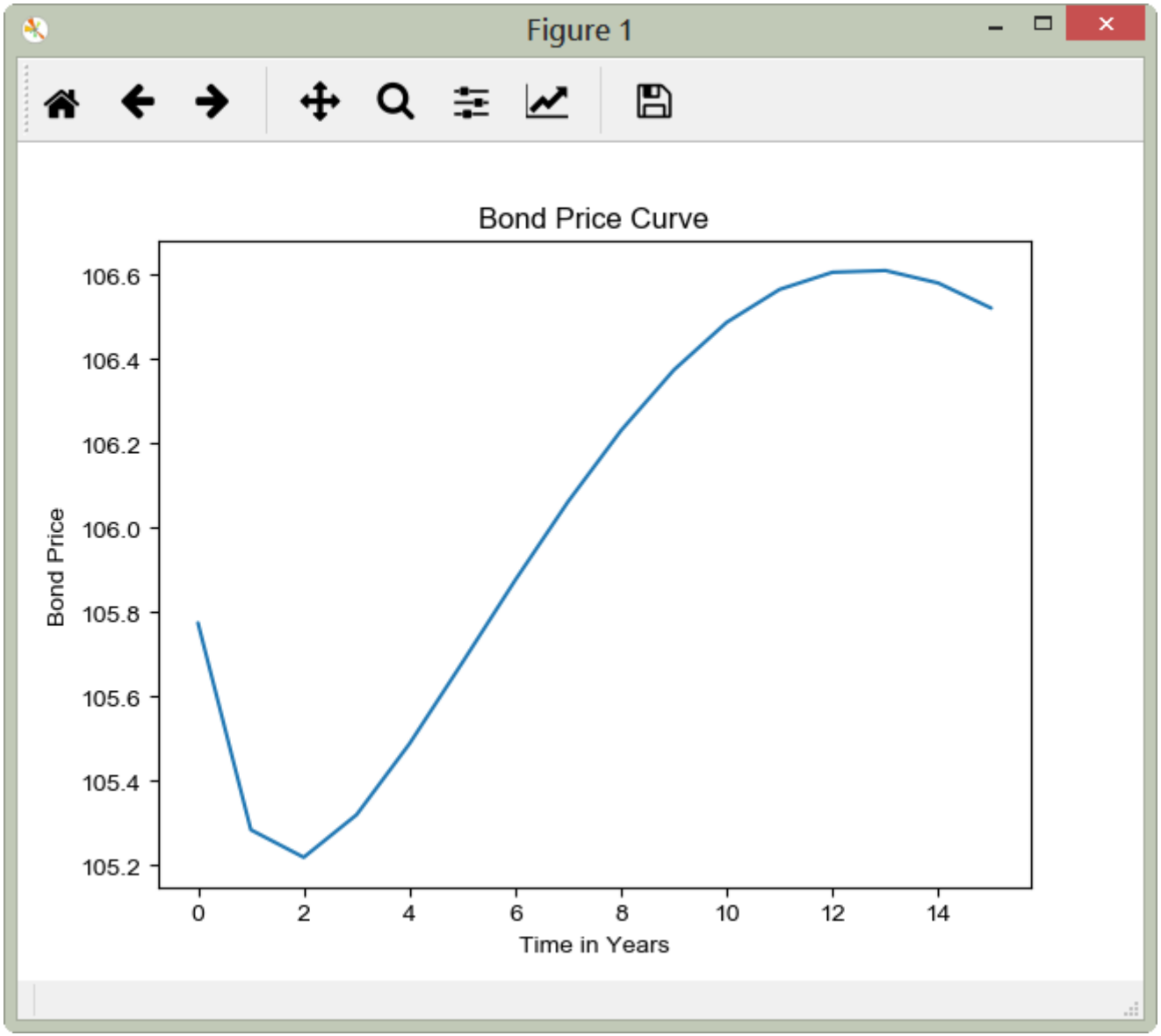


Figure 1
Bond Price Curve
106.6
106.4
106.2
106.0
105.8
105.6
105.4
105.2
Bond Price
0
2
4
6
8
10
12
14
Time in Years

◆第一個價格為原來的，最後價格為最後一個。中央的價格為不完全的變動，有些報價異動完成，有些沒有。因此那些價格是不正確的。

(三)Alternatives?

◆有些變通方法可以使用。例如，暫時凍結債券，防止通知前傳。

```
In [13]: bond.freeze()
```

◆如此，不會傳遞到觀察者。

```
In [14]: for q in quotes:
             q.setValue(101.5)
```

◆復原債券，發送一個通知。觸發一次重算，產生正確最後價格。

```
In [15]: bond.unfreeze()
Out[15]: 106.85839373944943
```

五、利率期限結構

(一)Setup

◆導入 QuantLib 模組，設定評價日。

```
In [1]: from QuantLib import *

In [2]: Settings.instance().evaluationDate = Date(3, October, 2014)
```

(二)明示Term Structure參考日

◆參考日是一 term structure 的起始日。它可為 evaluation date，但你也可能希望它自 spot date 開始。

◆第一種方法是有目前評價日的差距來表示，以“two business days after the evaluation date”定義其為 spot date；或是“no business days”來定義其為 evaluation date 本身。

- 此處藉由建立一個 swaps curve 來定義。

```
In [3]: helpers = [ SwapRateHelper(QuoteHandle(SimpleQuote(rate/100.0)),
                   Period(*tenor), TARGET(), Annual, Unadjusted, Thirty360(), Euribor6M())
                for tenor, rate in [((2,Years), 0.201), ((3,Years), 0.258), ((5,Years), 0.464),
                   ((10,Years), 1.151), ((15,Years), 1.588)] ]
```

- 注意 0 與 TARGET()引數，明示日數與使用決定營業日的日曆。

```
In [4]: curve1 = PiecewiseFlatForward(0, TARGET(), helpers, Actual360())
```

◆第二種方法事明確說明參考日期。

- 例如，ForwardCurve 類別接受特定日期向量與相對應的利率，內插這些資料。
- 第一個傳入的日期被認為是參考日，
- 由上述曲線取得節點資料，以之建立 ForwardCurve 實例。

```
In [5]: dates, rates = zip(*curve1.nodes())
In [6]: curve1.nodes()
Out[6]: ((Date(3,10,2014), 0.0019777694879293093),
         (Date(7,10,2016), 0.0019777694879293093),
         (Date(9,10,2017), 0.0036475517704509294),
         (Date(7,10,2019), 0.007660760701876805),
         (Date(7,10,2024), 0.018414773669420893),
         (Date(8,10,2029), 0.025311634328221498))
```

◆定義： zip([iterable, ...])

zip() 是 Python 的一個內建函數，它接受一系列可迭代的對象作為參數，將對象中對應的元素打包成一個個 tuple（元組），然後返回由這些 tuples 組成的 list（列表）。若傳入參數的長度不等，則返回 list 的長度和參數中長度最短的對象相同。利用*號操作符，可以將 list unzip（解壓）。

◆用法示例：參考下面的例子，對 zip()函數的基本用法就可以明白了：

```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> c = [4,5,6,7,8]
>>> zipped = zip(a,b)
>>> zipped
[(1, 4), (2, 5), (3, 6)]
>>> zip(a,c)
[(1, 4), (2, 5), (3, 6)]
>>> zip(*zipped)
[(1, 2, 3), (4, 5, 6)]
```

◆由這些資料建構出的曲線，與第一條是相同的。其參考日則是我們明示的 October 3rd (第一個傳入日期)。

```
In [7]: curve2 = ForwardCurve(dates, rates, Actual360())
```

◆兩條曲線定義於相同的日期區間。

```
In [8]: print("{0} to {1}".format(curve1.referenceDate(), curve1.maxDate()))
        print("{0} to {1}".format(curve2.referenceDate(), curve2.maxDate()))
Out[8]: October 3rd, 2014 to October 8th, 2029
        October 3rd, 2014 to October 8th, 2029
```

◆當我們詢問特定日期時，傳回相同利率。(例如，5 years)

```
In [9]: print(curve1.zeroRate(5.0, Continuous))
        print(curve2.zeroRate(5.0, Continuous))
Out[9]: 0.452196 % Actual/360 continuous compounding
        0.452196 % Actual/360 continuous compounding
```

◆或是特定日期。

```
In [10]: print(curve1.zeroRate(Date(7, September, 2019), Actual360(), Continuous))
         print(curve2.zeroRate(Date(7, September, 2019), Actual360(), Continuous))
Out[10]: 0.452196 % Actual/360 continuous compounding
         0.452196 % Actual/360 continuous compounding
```

◆匯出完整的利率曲線。

```
In [11]: from matplotlib.ticker import FuncFormatter
         import numpy as np
In [12]: times = np.linspace(0.0, 15.0, 400)
         rates = [ curve1.zeroRate(t, Continuous).rate() for t in times ]
         _, ax = utils.plot()
         ax.yaxis.set_major_formatter(FuncFormatter(lambda r,pos: utils.format_rate(r,2)))
         ax.plot(times, rates);
```

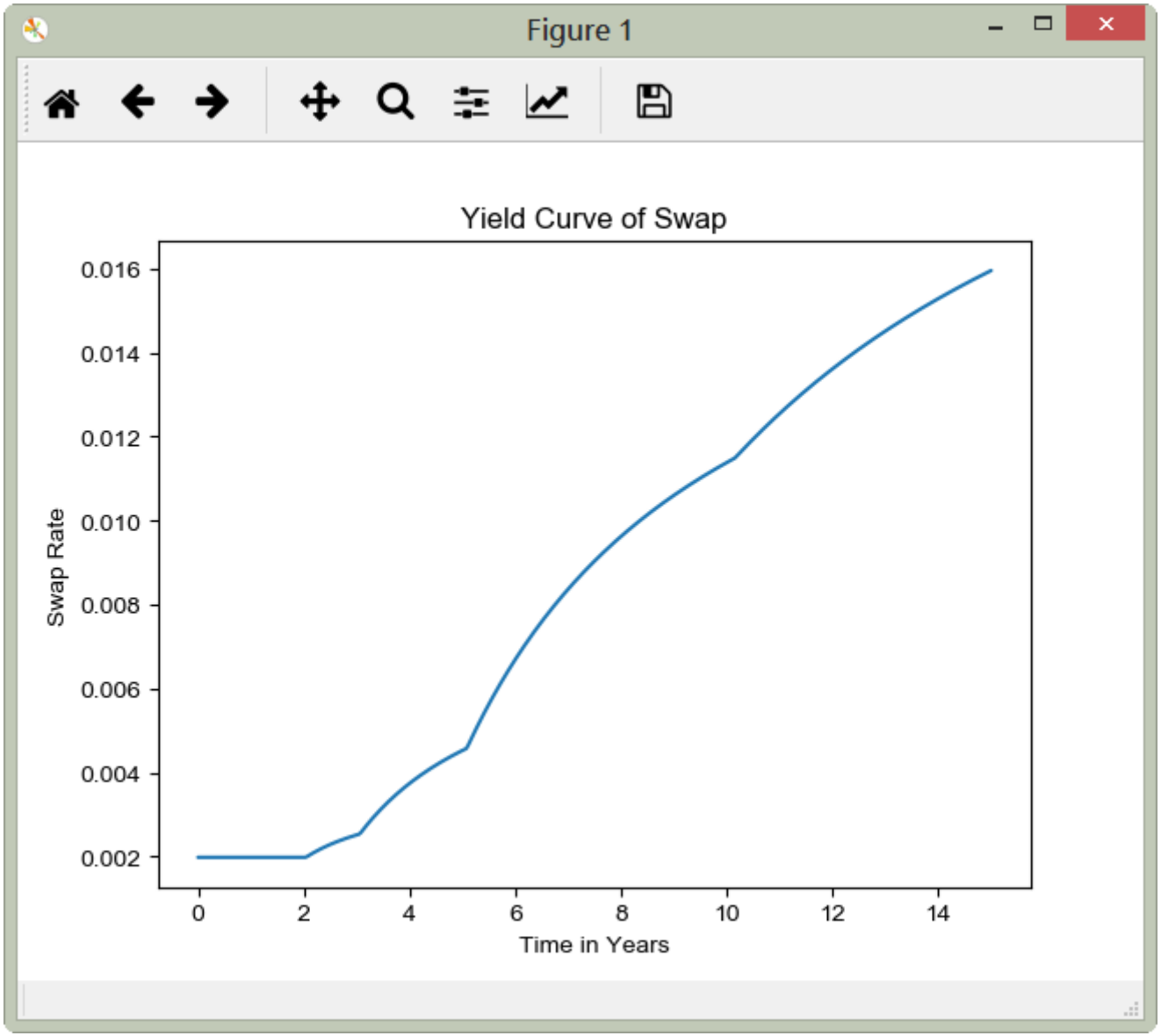


Figure 1
Yield Curve of Swap
Swap Rate
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0
2
4
6
8
10
12
14
Time in Years

(三)移動評價日

◆如果我們移動評價日，則此二曲線有何變化?

```
In [13]: Settings.instance().evaluationDate = Date(19, September, 2014)
```

◆如擬所預期，第一條曲線的參考日會變動，第二條則不會。我們可以看到第一條曲線定義的區間如何改變，第二條則否。

```
In [14]: print("{0} to {1}".format(curve1.referenceDate(), curve1.maxDate()))
         print("{0} to {1}".format(curve2.referenceDate(), curve2.maxDate()))
Out[14]: September 19th, 2014 to September 24th, 2029
         October 3rd, 2014 to October 8th, 2029
```

◆當然，利率也因之改變。

```
In [15]: print(curve1.zeroRate(5.0, Continuous))
         print(curve2.zeroRate(5.0, Continuous))
Out[15]: 0.452196 % Actual/360 continuous compounding
```

```
          0.452196 % Actual/360 continuous compounding
```

◆由於整條曲線往後移動數周，如果我們詢問一定時間後的利率，其值不變，如果詢問特定日期的利率，則自然不同。

```
In [16]: print(curve1.zeroRate(Date(7, September, 2019), Actual360(), Continuous))
         print(curve2.zeroRate(Date(7, September, 2019), Actual360(), Continuous))
Out[16]: 0.454618 % Actual/360 continuous compounding
         0.452196 % Actual/360 continuous compounding
```

(四)Notifications

◆最後，我們看看這兩條曲線的通知如何進行。製造兩個觀察者。

```
In [17]: def make_observer(i):
             def say():
                 s = "Observer %d notified" % i
                 print('-'*len(s))
                 print(s)
                 print('-'*len(s))
                 return Observer(say)
         obs1 = make_observer(1)
         obs2 = make_observer(2)
```

◆連結到一些報價，看看是否工作正常。

```
In [18]: q1 = SimpleQuote(1.0)
         obs1.registerWith(q1)
         q2 = SimpleQuote(2.0)
         obs2.registerWith(q2)
         q3 = SimpleQuote(3.0)
         obs1.registerWith(q3)
         obs2.registerWith(q3)
```

◆如果第一個報價變動，第一個觀察者會被通知。

```
In [19]: q1.setValue(1.5)
Out[19]: --------------------
         Observer 1 notified
         --------------------
```

◆如果第二個報價變動，第二個觀察者會被通知。

```
In [20]: q2.setValue(1.9)
Out[20]: -------------------
         Observer 2 notified
         -------------------
```

◆如果第三個報價變動，二個觀察者都會被通知。

```
In [21]: q3.setValue(3.1)
Out[21]: -------------------
         Observer 2 notified
         -------------------
         -------------------
         Observer 1 notified
         -------------------
```

◆將兩人分別連到兩條曲線，

```
In [22]: obs1.registerWith(curve1)
         obs2.registerWith(curve2)
```

◆評價日改變，只有第一人被通知。

```
In [23]: Settings.instance().evaluationDate = Date(23, September, 2014)
Out[23]: ------------------
         Observer 1 notified
         ------------------
```

金融研訓院 特約講師
證券暨投資分析人員合格(CSIA)
希奇資本 技術長(CTO)

董 夢 雲 財務博士

Mobil: (Taiwan)0988-065-751 (China)1508-919-2872
EMail: dongmy@ms5.hinet.net
Line ID/WeChat ID: andydong3137

專長

GPU 平行運算與財務工程，C#、.Net Framework、CUDA、OpenCL、C、C++。
外匯與利率結構商品評價實務，股權與債權及衍生商品評價實務。
風險管理理論與實務，資本配置與額度規劃。

經歷

中國信託商業銀行交易室研發科主管
凱基證券風險管理部主管兼亞洲區風險管理主管
中華開發金控、工業銀行風險管理處處長
永豐金控、商業銀行風險管理處處長
永豐商業銀行結構商品開發部副總經理

著作

金融選擇權：市場、評價與策略，第二版，1997，新陸書局。
財務工程與 Excel VBA 的應用：選擇權評價理論之實作，2005，證券暨期貨發展基金會。

翻譯

衍生性金融商品與內部稽核，2003，金融研訓院。