CQF Lecture One - Further Solutions

Explorations in Asset Returns

The Random Behaviour of Assets is not quite the typical solution to lecture exercises. Lecture One introduced a model, such as $R_i = \left[\mu\delta t + \sigma\sqrt{\delta t}\,\phi_i\right]$ and through the guided but self-implemented exercise questions you explore the robustness and time scaling of std deviation σ and then, empirical returns (historical) if they satisfy the assumptions about them. The later is done with histogram and Q-Q plot tools.

Beforehand, we consider the difference between log-returns and simple returns – in the context of **projection from the shorter to longer time period**. Simple returns are also known as arithmetic or linear. Log-returns are known as compound.

1. Multi-period projection with simple returns vs. log-returns, for the asset price S_t

$$R_t^s = \frac{S_{t+1}}{S_t} - 1 \qquad \text{vs.} \qquad R_t^c = \ln\frac{S_{t+1}}{S_t}$$

the conversion rule is below,

$$R_t^s = \exp\left(R_t^c\right) - 1$$

and can be modified to per period rate in the exponent $r \times n$, for example, monthly rate annualised as $R_{1M}^c \times 12$.

We ignore the difference between simple returns vs. log-returns computed from daily closing prices, because one day is sufficiently 'small time' period. The numerical difference is rarely in 2nd digit after the dot as a percentage, mostly 3rd or higher. However, remember that typical Monte-Carlo simulation step $dt = 0.01$ is $\tau = 2.52$ working days, which is considerably longer than one day.

For log-returns (compound rates) and SDE simulation,
A τ-period-log return is simply the sum of the single-period log-returns over τ/n periods, because it is additive in the exponent:

$$\begin{aligned} R_{t+\tau} &= R_t + R_{t+1} + R_{t+2} + \cdots + R_{t+n-1} \\ \\ R_{1Y} &= R_{1\text{M}} + R_{2\text{M}} + \cdots + R_{12\text{M}} \end{aligned}$$

where $R_{1\text{M}}$ is monthly return for $[0, 1\text{M}]$, $R_{2\text{M}}$ is monthly return for $[1\text{M}, 2\text{M}]$ and so on to $R_{12\text{M}}$ monthly return for $[11\text{M}, 12\text{M}]$.

Exercises and Solutions edited by CQF Faculty, Richard.Diamond@fitchlearning.com

For linear (simple) returns and portfolio optimisation and simple interest rates, projecting from single periods over the longer time period involves multiplication as follows:

$$\begin{aligned} 1+R_{t+\tau} &= (1+R_t)\times(1+R_{t+1})\times\cdots\times(1+R_{t+n-1}) \\ 1+R_{1Y} &= (1+R_{1\mathrm{M}})\times(1+R_{2\mathrm{M}})\times\cdots\times(1+R_{12\mathrm{M}}) \\ 1+r_{1Y} &= (1+r_{1\mathrm{M}})\times(1+f_{2\mathrm{M}})\times\cdots\times(1+f_{12\mathrm{M}}) \end{aligned}$$

The projection rule here is equivalent to no arbitrage relationship between: a set of forward rates and the simple (spot) rate over the entire period.

$r_{1\mathrm{M}} \equiv f_{1\mathrm{M}}$ applies over $[0, 1\mathrm{M}]$, while $f_{12\mathrm{M}}$ is a *forward term rate* applied to the period $[11\mathrm{M}, 12\mathrm{M}]$, and preferably known at the start of the period. That applies in interest rates payoffs and became a non-trivial assumption in the post-LIBOR world of backward-looking rates, computed from historic transactions only.

Rules about asset returns in portfolio optimisation problems. Returns are not unitless – their implicit unit is time. Returns are always estimated over some timescale, most common is 'daily timescale' where for $R_{t+\tau}-R_t$ the $\tau = 1/252 \approx 0.004$.

The square-root volatility time-scaling $\sigma\sqrt{\delta t}$ applies under the assumption that compound returns are invariants – they behave identically and independently across time.

Markowitz mean- variance minimisation (MVP) objective function is defined for linear returns.

$$\underset{\boldsymbol{w}}{\operatorname{argmin}}\,\{\boldsymbol{w'\mu_\tau} - \boldsymbol{\lambda w'\Sigma_\tau w}\}$$

where subscript in μ_τ and Σ_τ means their estimation from market data of respective frequency, eg daily $\tau = 1/252$.

In portfolio optimisation, linear returns typically aggregated across assets, and not time. Return for a portfolio of N assets is a weighted average of individual asset returns,

$$R_{t,\Pi} = w_1 R_{t,1} + w_2 R_{t,2} + w_3 R_{t,3} + \cdots + w_N R_{t,N}.$$

3. **Back to a histogram** as a discrete representation of probability density function $f_X(x)$, a *pdf* for the random variable capital X and its specific values small x. To build a histogram one engages in estimation of density,

$$f_X(x) = \frac{1}{Nh} n_j$$

where n_j is the varying number of observations in a bucket, h is our bandwidth – bucket/window size on the scale of real numbers $\mathbb{R}$, and N is the sample size.

n_j/N gives frequency, while n_j/Nh is 'a chunk' of density also called probability mass. If we set the window size so small that it includes only one observation, then for each chunk the *pdf* is $f(x) = 1/Nh$, where $1/h$ is a normalising parameter.

With the proper 'kernel smoothing' methods small bandwidth setting h will produce a histogram that repeats the plotted data (low smoothness), while sufficiently high bandwidth will smooth the representation into a symmetric histogram as if the returns are Normally distributed.

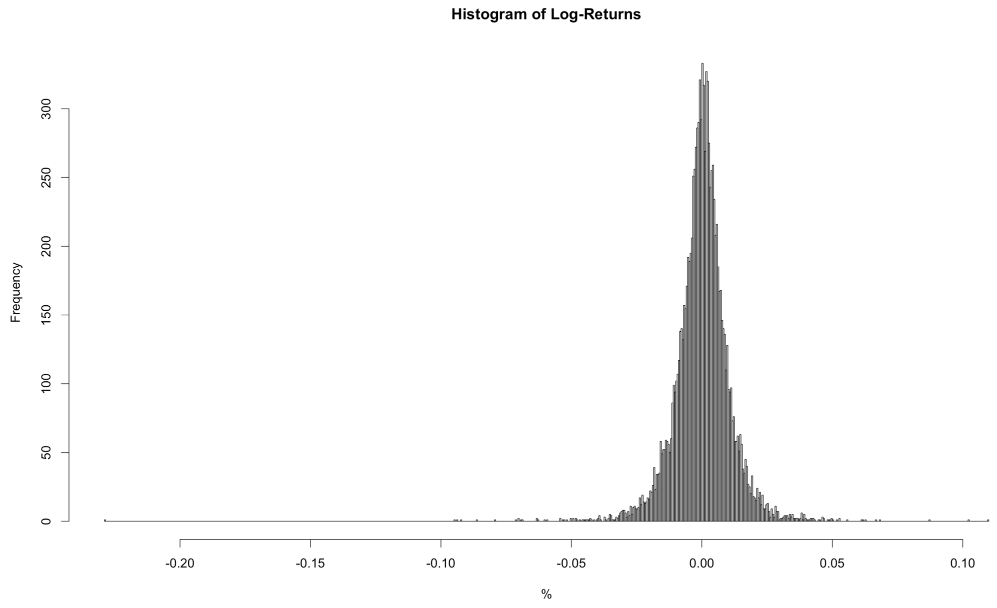

Figure 1: Histogram of SPX log-returns, built in R environment (code below).

1000 breakpoints were used. It is recommended that you experiment by setting the number of breakpoints higher (gives low smoothness) and lower but not too low.

4. Q-Q plot is straightforward to build from the first principles: follow the transformations and arrange the data in matching columns. Implementing the steps, you will obtain a result alike the following table. 'Historic' stands for the actual S&P 500 return, matched by 'Scaled' return (Z-score), indexes and the Standard Normal Percentile which corresponds to the cumulative probability given by i/N. Notice one past negative return that was in excess of 21 standard deviations!

Historic	Scaled	i	i/N	Standard
-0.22900	-21.20462	1	0.00009	-3.74534
-0.09470	-8.78146	2	0.00018	-3.56758
-0.09354	-8.67429	3	0.00027	-3.45987
-0.09219	-8.54970	4	0.00036	-3.38162
-0.08642	-8.01584	5	0.00045	-3.31983
-0.07922	-7.35036	6	0.00054	-3.26858
...	...	...	...	...

PDF $1/N$	CDF i/N
0.00009	0.00009
0.00009	0.00018
0.00009	0.00027
0.00009	0.00036
0.00009	0.00045
0.00009	0.00054
...	...

Table 1: Left: Inputs for a Q-Q plot. Right: Empirical PDF and CDF

Q-Q Plot step-by-step

(a) Scale empirical log-returns r_t to be the Normal variables $Z_t = \frac{r_t - \mu}{\sigma}$. For a market index the average daily return $\mu \approx 0$, that is increasingly valid for a large sample.

(b) Sort the scaled returns in the ascending order and create an index column $i = 1..N$.

(c) For each observation, the individual probability density (probability mass) is $1/N$ and the cumulative density (percentage of observations below this) is i/N.

(d) The standardised percentile is obtained with the inverse Normal CDF $\Phi^{-1}(i/N)$.

Plot the scaled returns (Z-scores) from step (a) against the theoretical percentiles from step (d). For the perfectly Normal log-returns the Q-Q plot would be a straight line.

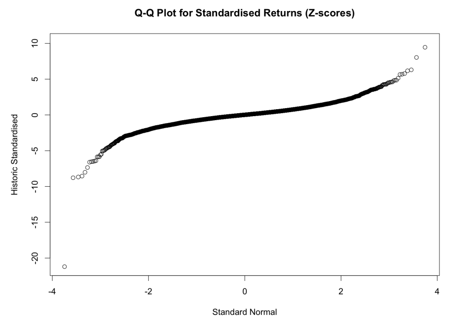

Figure 2: Q-Q Plot built step by step.

Because of our indexing $i = 1..N_{obs}$, the percentile axe X is essentially a scale of Standard Normal variables. For example, the standard normal variable (Z-score) 1.645 corresponds to an observation below which 95% of the observations reside.

Q-Q Plot with R libraries, ready functions Time series experiments are best conducted in the suitable environment. Most of the student and analytical tools (Eviews, Stata, SPSS, SAS) package the modelling into standardised statistical tests and procedures. They allow you to modify parameters but rarely the procedure itself.

We have selected R environment for its convenience in manipulating the time series and intuitive facilities of programming language. It is also noticeable that statistical tests were implemented by specialists – the output is organised to present what is important. A quant job requires the active knowledge of the chosen libraries, e.g., 'zoo' objects for time series analysis, 'quantlib' main library of the functions for a financial quant, and 'urca' library for working with non-stationary time series (cointegration testing).

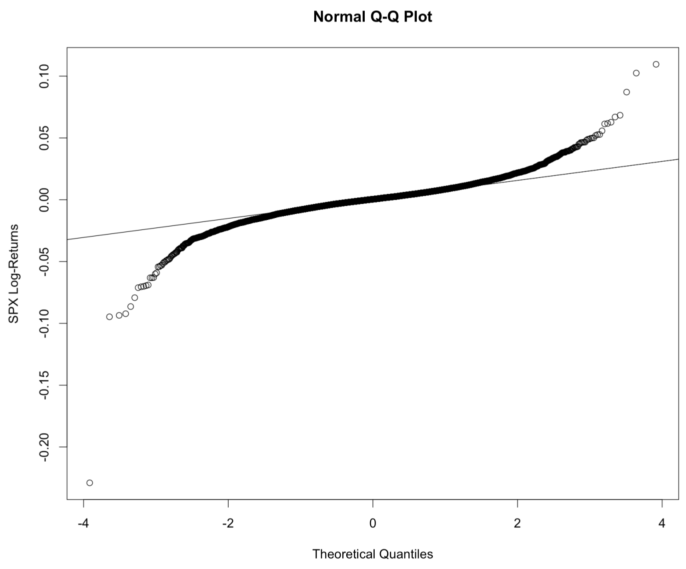

Figure 3: Q-Q Plot for SPX log-returns.

R Code: The code below reads from *SPX.xls* file distributed with Lecture One (saved as *.csv*). It selects certain sub-sample and plots a histogram and Q-Q plot.

```
#######################################################################
# 2015. Richard Diamond. Quieries to r.diamond@cqf.com                #
# CQF Lecture One                                                     #
#######################################################################

prices.zoo = read.zoo("SPX.csv", header=TRUE, sep=",", format = "%d/%m/%y")
prices.this = window(prices.zoo, start=as.Date("1950-01-03"), end=as.Date("2013-01-03"))
plot(prices.this$Close, main = "SPX Closing Prices", ylab = "Price (USD)", xlab = "Date")
```

```
# P1 Invariants. IID Check (histogram)
returns.this = diff(log(prices.this$Close))
hist(returns.this, breaks=1000, main = "Histogram of SPX Log-Returns", xlab="%")

# P2 Estimation. Normality Check (Q-Q plot)
qqnorm(returns.this, ylab = "SPX Log-Returns")
qqline(returns.this)
```

The following section of the R code replicates the steps that you might like to do in Excel in the first instance. It provides an example of matrix manipulation in R. Code comments abridged.

```
######################################################################
# 2015. Richard Diamond. Quieries to r.diamond@cqf.com               #
# CQF Lecture One. Q-Q Plot Step-by-step                             #
######################################################################

# P2.1 Empirical density
sreturns.this = (returns.this - mean(returns.this))/ sd(returns.this) #the long way
sreturns.this = scale(returns.this, center=TRUE, scale=TRUE) # each value is a Z-score

plot(sreturns.this, main = "SPX Standardised Returns (Z-scores)",
 ylab = "Standard Deviation", xlab = "Date")

# P2.2 Building a data matrix (table for presentation)
sreturns.N = length(sreturns.this)
sreturns.data = matrix(rep(NA, sreturns.N), sreturns.N, 3) #an empty structure
# dim(sreturns.data)

for(i in 1:sreturns.N)
  {
  sreturns.data[i,1]=i # This can be using seq(1:sreturns.N)
  sreturns.data[i,2]=i/sreturns.N
  sreturns.data[i,3]= qnorm(i/sreturns.N) # Can this be more efficient computationally?
  }
sreturns.data = cbind(sort(as.vector(sreturns.this)), sreturns.data)
# This is a tricky line: sorting a vector of scaled returns and appending it the data matrix

# P2.3 Q-Q Plot
plot(sreturns.data[,4],sreturns.data[,1], main = "Q-Q Plot for Standardised Returns (Z-scores)"
 ylab = "Historic Standardised", xlab = "Standard Normal")
```