Stats and Probability Information

April 21, 2014

Continuous and marginal distributions

The marginal distribution of x in a two-variable distribution is equal to the sum of the joint distribution over y.

$$Pr(X = x) = \sum_y Pr(X = x, Y = y) = \sum_y Pr(X = x|Y = y)Pr(Y = y) \tag{1}$$

From Wikipedia
For the continuous case

$$p_X(x) = \int_y p_{X,Y}(x, y)dy = \int_y p_{X|Y}(x|y)p_Y(y)dy \tag{2}$$

There are three related distributions: the marginal, the joint and the conditional.

1 Mixture Model

This is a probabilistic model that relates some random variables to some other variables. The model has sub-populations. The properties of the sub-population are different from those of the parent. The sub-populations may not be observable. For example, the distribution of returns may be different in different sub-population or regime.

A *mixture distribution* is the probability distribution of a random variable whose values are derived from an underlying set of random variables. The *mixture components* are individual distributions with *mixture weights*. Even in cases where the mixture comonents have a normal distribution, the mixture distribution is likely to be non-normal. Mixture models are used to

understand the sub-population when there is only access to the information about the pooled population.

The mixture model will be comprised of N random varibles distributed according to K components, with each component belonging to the same distribution. The k mixture weights sum to one. Each component will have parameters (mean and variance in the case of normal distribution).

The method will try to estimate the all the parameters of the model from the data. The underlying data is known (x_i); the number of mixture components is set (K); the parameters of the distribution of each mixture component ($\theta_{i=1...K}$); mixture weight ($\Phi_{i=1...K}$); $\mathbf{\Phi}$ K-dimensional vector summing to 1; $F(x|\theta)$ probability distribution of observations parameterised on θ; α shared hyperparameter for component weights; β shared hyperparameter for mixture weights; $H(\theta|\alpha)$ prior probability distribution of component parameters;

2 Adjusted R squared

Adjusted R squared applied a penalty to the basic R squard to account for additional variables. The equartion is

$$R_A^2 = 1 - \left[\frac{(n-1)}{(n-k)}\right][1 - R^2] \tag{3}$$

Adding a regressor to the equation will increase (reduce)) the $R_A^"$ when the absolute value of the t-statistic is greater (less) than one. Adding a group of regressors to the model will reduce (increase) the $R_A^"$ when the absolute value of the F-statistic is greater than one.

Proof http://davegiles.blogspot.com/2014/04/proof-of-result-about-adjusted.html

3 Monte Carlo Simulation

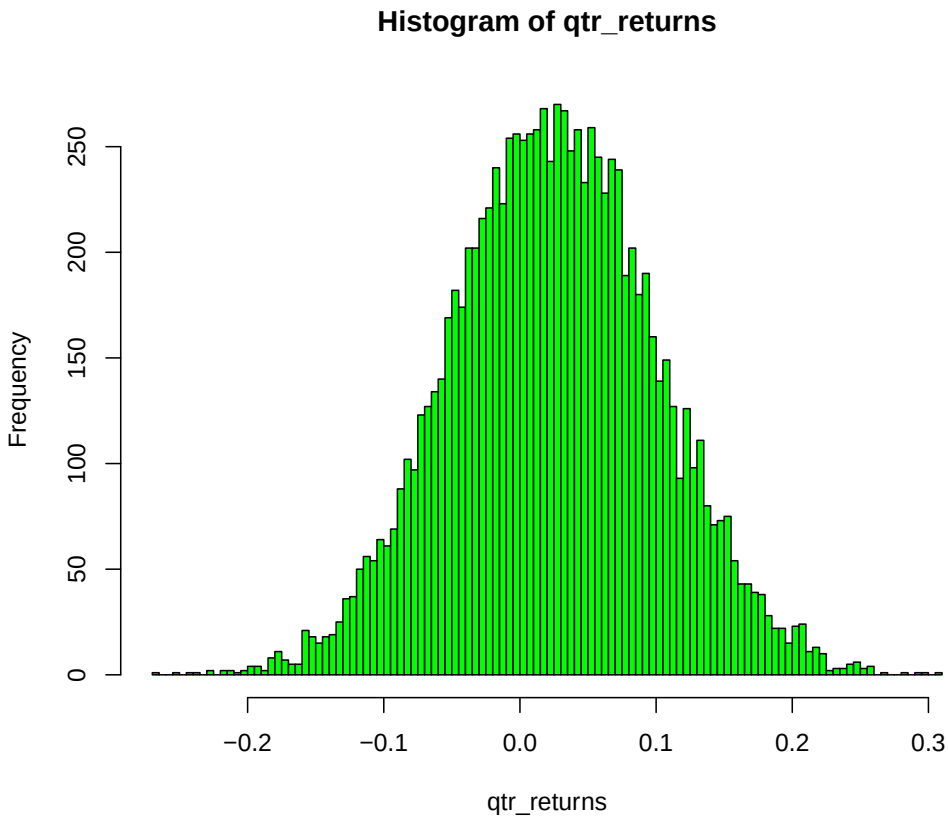
This comes from Revoluitionary Analytics. The analysis is in annual terms.

$$\mu\Delta t + \sigma Z\sqrt{\Delta t} \tag{4}$$

where μ is the drift or average annual return, Z is a standard Normal random variable, t is measured in years so for monthly returns Δt equals $\frac{1}{12}$.

```
n <- 10000
# Fixing the seed gives us a consistent set of simulated returns
set.seed(106)
```

```r
z <- rnorm(n)  # mean = 0 and sd = 1 are defaults
mu <- 0.1
sd <- 0.15
delta_t <- 0.25
# apply to expression (*) above
qtr_returns <- mu * delta_t + sd * z * sqrt(delta_t)
hist(qtr_returns, breaks = 100, col = "green")
```



Now the descriptive statistics can be uncovered from the simulated results.

```r
stats <- c(mean(qtr_returns) * 4, sd(qtr_returns) * 2)  # sqrt(4)
names(stats) <- c("mean", "volatility")
stats
```

```
##       mean volatility
##    0.09901    0.14976
```

This is the basic model. It would also be possible to simulate two variables and to include some relationship between the two in the analysis. It would also be possible to simulate an asset in two different regimes. A Monte-Carlo Markov Model (MCMM) would require another set of μ and σ inputs as well as a transition matrix of the probabilities that there is a switch from one regime to another.

4 Generalised Lambda Distribution

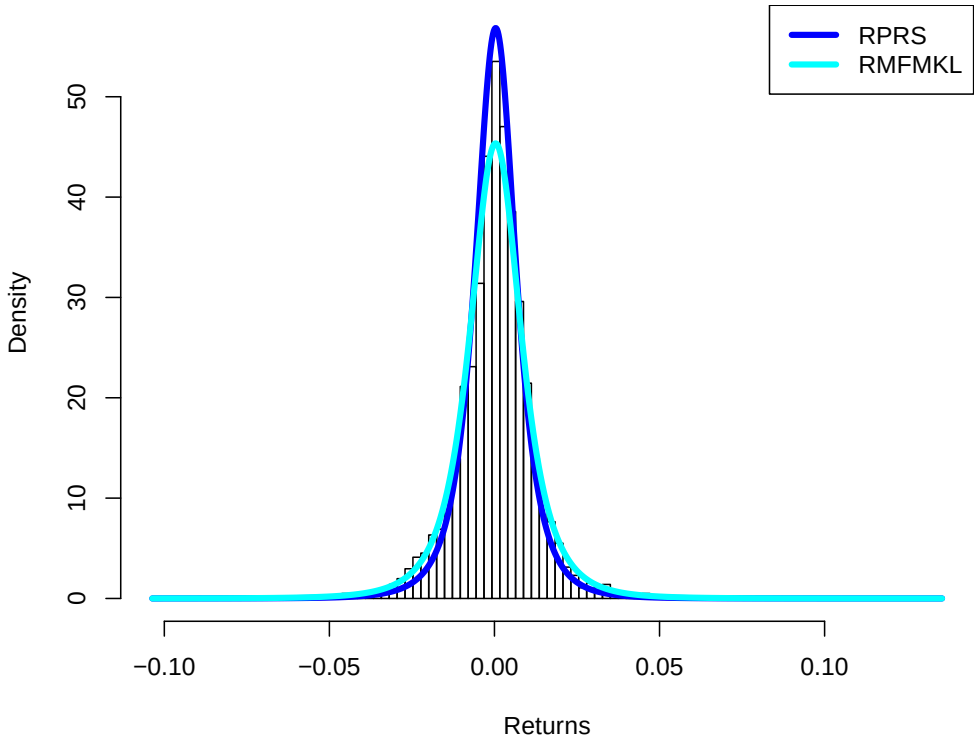
This is from Revolutionary Analytics. The four parameters λ_1, λ_2, λ_3 and λ_4 indicate the location, scale, skew and kurtosis of the distribution.

```
require(GLDEX)
require(quantmod)
getSymbols("SPY", from = "1994-02-01")

## [1] "SPY"

SPY.Close <- SPY[, 4]  # Closing prices
SPY.vector <- as.vector(SPY.Close)
# Calculate log returns
sp500 <- diff(log(SPY.vector), lag = 1)
sp500 <- sp500[-1]  # Remove the NA in the first position
# Set normalise='Y' so that kurtosis is calculated with reference to
# kurtosis = 0 under Normal distribution
fun.moments.r(sp500, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  0.0002633  0.0001532 -0.0960976  9.5131553
```

Now fit the GLD with the function fun.data.fit.mm. There are warnings but these can be ignored.

```
spLambdaDist = fun.data.fit.mm(sp500)
spLambdaDist

##             RPRS     RMFMKL
## [1,]  3.846e-04   0.000321
## [2,] -4.228e+01 203.501581
## [3,] -1.675e-01  -0.169657
## [4,] -1.640e-01  -0.161483
```

```
fun.plot.fit(fit.obj = spLambdaDist, data = sp500, nclass = 100, param = c("rs",
    "fmkl"), xlab = "Returns")
```



Now it is possible to generrate simualted results using the function rgl(). Lambdas need to be identified.

```
lambda_params_rs <- spLambdaDist[, 1]
lambda1_rs <- lambda_params_rs[1]
lambda2_rs <- lambda_params_rs[2]
lambda3_rs <- lambda_params_rs[3]
lambda4_rs <- lambda_params_rs[4]
lambda_params_fmkl <- spLambdaDist[, 2]
lambda1_fmkl <- lambda_params_fmkl[1]
lambda2_fmkl <- lambda_params_fmkl[2]
lambda3_fmkl <- lambda_params_fmkl[3]
lambda4_fmkl <- lambda_params_fmkl[4]
```

Now generate simulations of each variety.

There are problems with the rgl function. I am not sure what this does. It is 10 million simulations. I think that the rgl just uses extra hardware to make the change. It may be useful to re-do this last section using a different method.

```
require(gld)

## Loading required package:  gld

require(GLDEX)

## Loading required package:  GLDEX
## Loading required package:  cluster
##
## Attaching package:  'GLDEX'
##
## The following objects are masked from 'package:gld':
##
##    dgl, pgl, qdgl, qgl, rgl, starship, starship.adaptivegrid,
##    starship.obj

# RS version:
set.seed(100)  # Set seed to obtain a reproducible set
rs_sample <- rgl(n = 1e+07, lambda1 = lambda1_rs, lambda2 = lambda2_rs, lambda3 =
    lambda4 = lambda4_rs, param = "rs")
# Moments of simulated returns using RS method:
fun.moments.r(rs_sample, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  2.633e-04  9.774e-05 -1.043e-01  9.955e+00

# Moments calculated from market data:
fun.moments.r(sp500, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  0.0002633  0.0001532 -0.0960976  9.5131553

# FKML version:
set.seed(100)  # Set seed to obtain a reproducible set
fmkl_sample <- rgl(n = 1e+05, lambda1 = lambda1_fmkl, lambda2 = lambda2_fmkl,
    lambda3 = lambda3_fmkl, lambda4 = lambda4_fmkl, param = "fmkl")
# Moments of simulated returns using FMKL method:
fun.moments.r(fmkl_sample, normalise = "Y")
```

```
##       mean   variance   skewness   kurtosis
##  0.0002403  0.0001547 -0.0862244  8.5839659

# Moments calculated from market data:
fun.moments.r(sp500, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  0.0002633  0.0001532 -0.0960976  9.5131553
```

Compare the moments to the S&P500 market data

```
fun.moments.r(rs_sample, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  2.633e-04  9.774e-05 -1.043e-01  9.955e+00

fun.moments.r(sp500, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  0.0002633  0.0001532 -0.0960976  9.5131553

fun.moments.r(fmkl_sample, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  0.0002403  0.0001547 -0.0862244  8.5839659

fun.moments.r(sp500, normalise = "Y")

##       mean   variance   skewness   kurtosis
##  0.0002633  0.0001532 -0.0960976  9.5131553
```