An interpretation and comparison of Merton jump diffusion and geometric brownian motion processes as models for price movement of large cap U.S. technology equities.

Robert von der Schmidt, Princeton University Undergraduate Class of 2020
Graduate Advisor Utsav Popat

A project submitted to the Bendheim Center for Finance in partial fulfillment of the requirements for a certificate of proficiency in finance.

Abstract

The purpose of this paper is to test the relative efficacy of the Black-Scholes geometric brownian motion process and Merton jump diffusion process in explaining and predicting the price movement of the five largest (based on market capitalization) technology companies: Microsoft (MSFT), Amazon (AMZN), Apple (AAPL), Alphabet (GOOG), and Facebook (FB), whose combined market cap is $4.6 trillion USD [11]. All five companies are well-diversified firms with large business-to-consumer components whose earnings can be specifically affected a myriad of different external economic, social, and political factors. I begin with a brief introduction of Black-Scholes. I then derive the geometric brownian motion equation using Ito integrals, as many scholars have done more recently (as opposed to the original derivation through the scaling limit of a random walk). I illustrate how expectation and variance laws can give an explicit solution to the maximum likelihood estimate of drift and volatility parameters in the Black-Scholes model. I introduce and explain the Merton model, and I use numerical methods to maximize that model's probability density function from an initial guess. Kernel density estimation is used to compare empirical and theoretical distributions, Monte Carlo simulation is used to estimate the moments of the theoretical models, and the Akaike Information Criterion scores of both models are compared to see, particularly if there is additional explanatory power of the Merton model, if this added power warrants the additional parameters. This comparison is done both for time series data of one month and one year to examine how the relative strengths of the models change with the size of their forecasting windows.

Introduction

When Black and Scholes created their options pricing formula in their seminal 1973 paper, "The Pricing of Options and Corporate Liabilities", it was a revolution in options pricing and stock price forecasting because of the models' primary reliance on "observable variables" as opposed to "knowledge about investors' tastes or their beliefs about expected returns on the underlying common stock" [1,2]. Given the validity of five critical assumptions about frictionless markets, constant interest rates, lack of dividends, European style options contracts, and geometric brownian motion, analysts could price an option with nothing but strike price, time to expiry, and a history of price quotes [1]. This means of pricing and analysis was crucial to the explosion of the derivatives market that ultimately took place in the late seventies and entirety of the eighties as before its introduction, analysts had lacked robust means of valuation [3]. Within these critical assumptions necessary for the valid application of Black-Scholes, Robert Merton analyzed a hierarchy at which geometric brownian motion laid at the top. Even when the risk-free "interest rate is stochastic", "the stock pays dividends", and "the option is exercisable prior to expiration", Black-Scholes analysis would often remain valid [2]. Furthermore, as long as market activity proceeded in discrete increments that were small enough, which Merton believed was the case, he reasoned that the first assumption would essentially hold as well [2]. Thus, from a utilitarian sense, it seems that geometric brownian motion is often the most critical assumption to examine. From an existential pedagogic perspective, it is also probably the most important assumption to analyze. A closed form equation of either the stochastic differential equation of the stock price or the stock price itself is what allows us to say what is actually happening: what price movement involves. Intelligibility is crucial, especially in finance: mean-variance

efficiency and the capital asset pricing model, for example, are two systems that have been empirically outperformed by other explanations and models, yet are still widely employed in large part due to simplicity and intuitive appeal [4]. A regression of a stock against some market proxy is easier to make sense of than the other factors in the Fama-French model, and while we may want to minimize kurtosis independent of variance, variance is a better understood concept by most. Even outside of finance, we use approximations that we know are incomplete or inadequate, and its because of their pedagogic value or intelligibility – we talk about substances entropically tending towards eight valence electrons, when this is not always true (SF_6), we use finite perfect information games to explain firm competition, we take gravity's acceleration as a constant, ignoring jerk. The ethos that "all models are wrong, but some are useful" pervades, and so much of the value of a model comes from its ability not only to predict, but to explain. For this reason it's important to explore what geometric brownian motion *is,* and hopefully, if our parameter estimation has enough power, what Poisson jumps *are*. Merton's characterization of his Poisson jumps as representing "vibrations in price … due to the arrival of important new information about the stock that has more than a marginal effect … usually … specific to the firm or possibly its industry … arriving at discrete points in time" [2] is not necessarily incomplete, but it is still somewhat intangible, and to the extent it remains intangible it fails to realize its full potential.

Wiener Process

The first step to understanding Brownian motion, is to become acquainted with the Wiener process, which is characterized by the following properties [5]:

1. $W(0) = 0$;
2. $dW \sim N(0,dt)$ where $\sim$ means "is a random variable generated by", and $N(\mu,\sigma^2)$ is a normally distributed probability density function with mean μ and variance σ^2.
3. $\Delta W = N(0,\Delta t)$
4. $W_{t+\Delta t} - W_t$ is independent from all past values of W_x where $x \in (0,t)$.
5. $\forall t \in [0, \infty) \exists W_t$, which is to say the process is continuous

I will be simulating this process in R with very small time steps, so I'm technically relaxing property 5, and property 2 to the extent that differentials won't exist for this non-continuous process I generate, but again the time steps will be made small enough that the process appears continuous. Secondly despite the lack of continuity, I won't actually be approximating the process. It will be more like I am just displaying discrete, but still valid values of the continuous function, separated by very small amounts of time, as a consequence of property 3.

R-Code for Wiener process simulation

Because $\Delta W = N(0,\Delta t)$ and $W(0) = 0$, we can simulate the Wiener process exactly, as opposed to approximating it. We are able to recursively define $W(t)$ $\forall$ t, by saying that $W_{t+\Delta t} = W_t + N(0,\Delta t)$,. R happens to have a built-in random normally distributed variable function, rnorm(), where the first argument is the size of the sequence to be generated (in our case 1), and the second and third are the mean and standard deviation respectively (0 and 1 in the case of a Wiener process).

R-Code for Wiener process simulation *continued*

```
wiener = function(simulations)
{
 w = 1:simulations;
 w[1] = 0;
 for (i in 2:simulations) {
  w[i] = w[i-1] + rnorm(1,0,1);
 }
 return(w)
}
```

Technically, it's computationally inefficient to be using for loops like this in R, but I use them to keep the code more language agnostic.

Stochastic Differential Equation (SDE) for Geometric Brownian Motion

Using the concepts of both a wiener and time differential, we can define the stochastic (including random variables) differential equation (SDE) that characterizes geometric brownian motion of stocks.

Let S be the price of a stock, dS be an infinitely small change in that price, μ_s be the "drift" of a stock, σ_s^2 be the "volatility", dt be a time differential, and dW be a wiener process differential.

$dS = S(\mu_s\, dt + \sigma_s\, dW)$

Ito's Product Rule and Lemma

A branch of calculus, Stochastic Calculus, largely created and revolutionized by Kiyosi Ito [6], will now help us uncover an explicit solution from the stochastic differential equation. First of all, Ito's product rule will help us evaluate products of differentials of the Wiener process and time, dW and dt.

Ito's Product Rule [6]

$dW^2 = dt$
$dWdt = 0$
$dt^2 = 0$

Corollary to Ito's Product Rule [6]

Let X be a stochastic process described by the following SDE

$dX = \mu_x dt + \sigma_x dW$

Then $dX(dt) = (\mu_x dt + \sigma_x dW)dt = \mu_x dt^2 + \sigma_x dWdt = 0$
And $dX(dX) = (\mu_x dt + \sigma_x dW)(\mu_x dt + \sigma_x dW) = \mu_x^2\, dt^2 + 2\mu_x\, \sigma_x dWdt + \sigma_x^2\, dW^2 = \sigma_x^2 dt$

$\therefore\ dX^2 = \sigma_x^2 dt$ and $dXdt = 0\ \forall\ dX$ of the form $dX = \mu_x dt + \sigma_x dW$

Ito's Lemma [6]

Ito's Lemma allows us to find the differential of a function S(X(t),t) where $dX = \mu_x dt + \sigma_x dW$, by applying Ito's product rules to the Taylor series expansion of a stochastic function. The multivariable Taylor series expansion of S(X(t),t) would be as follows (from the normal rules of multivariable calculus with deterministic functions):

$dS = \partial S/\partial t * dt + \partial S/\partial X * dX. + \frac{1}{2} * \partial^2 S/\partial t^2 * dt^2 + \frac{1}{2} * \partial^2 S/\partial X^2 * dX^2 + 2 * \partial^2 S/\partial X \partial t * dXdt + \ldots$

The … denotes higher order terms that all go to zero according to Ito's product rules and corollaries, as they include the products of dt and dX with dt^2 (= 0), dX^2 ($=\sigma_x^2 dt$), and dXdt (=0). Products of dt^2 and dXdt with anything will obviously be zero, but so will products of dX^2 w/ dt and dX as they will be equal to $dX^2 dt = \sigma_x^2 dt^2 = 0$ or $dX^2 dX = \sigma_x^2 dXdt = 0$. Of the four remaining terms, we can also eliminate $\frac{1}{2} * \partial^2 S/\partial t^2 * dt^2$ (because $dt^2 = 0$) and $2 * \partial^2 S/\partial X \partial t * dXdt$ (because dXdt = 0).

$dS = \partial S/\partial t * dt + \partial S/\partial X * dX + \frac{1}{2} * \partial^2 S/\partial X^2 * dX^2$

Applying $dX^2 = dt$ allows us to condense this equation to

$dS = (\partial S/\partial t + \frac{1}{2} * \sigma_x^2 \partial^2 S/\partial X^2) dt + \partial S/\partial X * dX$

Which is the more general form of Ito's lemma. For our purposes, we will use the more specific form when X is the wiener process, W, which means that $\sigma_x^2 = 1$, so that

$dS = (\partial S/\partial t + \frac{1}{2} * \partial^2 S/\partial W^2) dt + \partial S/\partial W * dW$ for a function S of W(t) and t

Explicit Solution to Geometric Brownian Motion SDE

$dS = S(\mu_s\, dt + \sigma_s^2\, dW)$, then S is a function of W(t) and t. and dS is also equal to

$(\partial S/\partial t + \frac{1}{2} * \partial^2 S/\partial W^2) dt + \partial S/\partial W * dW$

$\therefore\ S(\mu_s\,dt + \sigma_s\,dW) = S\mu_s dt + S\sigma_s\,dW = (\partial S/\partial t + \ ½*\partial^2 S/\partial W^2)dt + (\partial S/\partial W)dW$

$\therefore\ S\sigma_s = \partial S/\partial W$ (I)

$S\mu_s = \partial S/\partial t + \ ½*\partial^2 S/\partial W^2$ (II)

We note that $S(0) = 0$ by definition and that this means that $S(t) \neq 0\ \forall\ t$, and this will allow us to freely divide expressions by S.

$\partial S/\partial W = S\sigma_s$ (from I)

$(1/S)\partial S = \sigma_s \partial W$

$\int(1/S)\partial S = \int\sigma_s \partial W$

$Ln(S) = \sigma_s W + f(t)$

$S = e^{\sigma s W + f(t)}$ (III)

Now, we use the chain rule to differentiate S, first with respect to t

$\partial S/\partial t = f'(t)S$ (IV)

And then with respect to W

$\partial S/\partial W = \sigma_s S$

$\partial^2 S/\partial W^2 = \sigma_s^2 S$ (V)

$S\mu_s = \partial S/\partial t + \ ½*\partial^2 S/\partial W^2 = f'(t)S + \ ½\sigma_s^2 S$ (from II, IV, and V)

$\mu_s = f'(t) + \ ½\sigma_s^{2.}$

$f'(t) = \mu_s - ½\sigma_s^2$

$f(t) = \int(\mu_s - ½\sigma_s^2)\ dt = (\mu_s - ½\sigma_s^2)t + C$ (VI)

$S(W(t),t) = e^{\sigma s W + (\mu_s - ½\sigma_s 2)t + C}$ (from III and VI)

Initial value of a stock, $S_0 = S(W(0),0) = e^C$, **so the explicit solution to the stochastic differential equation of a stock moving in geometric brownian motion is**

$S(W(t),t) = S_0\ e^{\sigma s W + (\mu_s - ½\sigma_s 2)t}$

While this proof follows directly from Taylor Expansion and Ito's Product Rules and Lemma, much of the specific form of its form is adapted from Chapter seven of an "Introduction to Stochastic Processes" by Edward P.C. Kao [7].

R Code For Geometric Brownian Motion Simulation

Different stocks have different initial prices, drifts, and volatilities, so the whole price is really a function $S(W(t), t, \mu_s, \sigma_s^2)$. The only argument that is a function itself, fortunately happens to be capable of an exact simulation, thus we can also simulate geometric brownian motion exactly.

```
gbm = function(s0, mu, sig, time,simulations_per_period)
{
  n = time*simulations_per_period;
  stock = 1:n;
  w = wiener(n);
  for (i in 1:n) {
    stock[i] = s0*exp((mu/simulations_per_period- .5*sig^2/(simulations_per_period))*i +
sig*(1/simulations_per_period^.5)*w[i])};
  return(stock);
}
```

Maximum Likelihood Estimation of σ_s

Variance and expectation laws allow us to derive an explicit solution for the maximum likelihood estimation of μ_s and σ_s.

We start off by taking the log of the geometric brownian motion equation.

$\ln(S) = \ln(S_0\, e^{\sigma s W+(\mu_s - \frac{1}{2}\sigma_s^2)t}) = \ln(S_0) + \sigma s W+(\mu s - \frac{1}{2}\sigma s2)t$

$\therefore\ \ln(S_{t+\Delta t}/S_t) = \ln(S_o) + \sigma_s W_{t+\Delta t}+(\mu_s - \frac{1}{2}\sigma_s^2)(t+\Delta t) - \ln(S_0) - \sigma_s W_t - (\mu_s - \frac{1}{2}\sigma_s^2)(t)$

Cancelling out pairs of terms that sum to 0, we get

$\ln(S_{t+\Delta t}/S_t) = \sigma_s(W_{t+\Delta t} - W_t) + (\mu_s - \frac{1}{2}\sigma_s^2)(\Delta t) = \sigma_s\, N(0,\Delta t) + (\mu_s - \frac{1}{2}\sigma_s^2)(\Delta t)$

Now, we take the variance of both sides to obtain our MLE of σ_s^2

$\mathrm{Var}[\ln(S_{t+\Delta t}/S_t)] = \mathrm{Var}[\sigma_s\, N(0,\Delta t) + (\mu_s - \frac{1}{2}\sigma_s^2)(\Delta t)] = \mathrm{Var}[\sigma_s\, N(0,\Delta t)] + \mathrm{Var}[(\mu_s - \frac{1}{2}\sigma_s^2)(\Delta t)] + \mathrm{Cov}[\sigma_s\, N(0,\Delta t), (\mu_s - \frac{1}{2}\sigma_s^2)(\Delta t)] = \mathrm{Var}[\sigma_s\, N(0,\Delta t)] + 0 + 0 = \sigma_s^2 \mathrm{Var}[N(0,\Delta t)] = \sigma_s^2 \Delta t$

$\therefore\ \sigma_s = (\mathrm{Var}[\ln(S_{t+\Delta t}/S_t)]/\Delta t)^{.5}$, which becomes an approximation when variance is the sample variance.

R Code for $S_{t+\Delta t}/S_t$

Because $S_{t+\Delta t}/S_t$ is used in our MLE of σ_s^2 and μ_s for stocks assumed to move in geometric brownian motion, I write a function that calculates these returns for the entire quote history of a stock.

```
returns = function(s)
{
  r = 1:(length(s) - 1);
  for (i in 1:(length(s) - 1))
  {
    r[i] = s[i+1] / s[i];
  }
  return(r);
}
```

R Code For MLE of σ_s^2

```
#var() returns the sample variance of a dataset
getmu = function(s, t)
{
dt = t/length(s);
r = returns(s);
x = (2*mean(log(r))+var(log(r)))/2/dt;
return(x);
}
```

Maximum Likelihood Estimation of μ_s

Next, we take expectation of both sides to obtain our MLE of μ_s

$$E[\ln(S_{t+\Delta t}/S_t)] = E[\sigma_s N(0,\Delta t) + (\mu_s - \tfrac{1}{2}\sigma_s^2)(\Delta t)] = E[\sigma_s N(0,\Delta t)] + E[(\mu_s - \tfrac{1}{2}\sigma_s^2)(\Delta t)]$$
$$= 0 + E[(\mu_s - \tfrac{1}{2}\sigma_s^2)(\Delta t)$$
$$= (\mu_s - \tfrac{1}{2}\sigma_s^2)(\Delta t)$$

$\therefore\ \mu_s = E[\ln(S_{t+\Delta t}/S_t)] + \tfrac{1}{2}\sigma_s^2 = (2*\ E[\ln(S_{t+\Delta t}/S_t)] + Var[\ln(S_{t+\Delta t}/S_t)]/\Delta t)/2$, which again becomes an estimate as we can only find sample means and variances of log stock price returns $\ln(S_{t+\Delta t}/S_t)$

R Code for MLE of μ_s

```
getsig = function(s,t)
{
  dt = t/length(s);
  r = returns(s);
  x = (var(log(r))/dt)^.5;
  return(x);}
```

simgbm Method for Stock Price Data

My getmu() and getsig() functions can now be nested into a function that takes stock price history as its argument, the time length of that history, the forecast window, the desired simulations per forecast, and a Boolean to take into account whether I want a time series plot or just a vector of predictions.

```
simgbm = function(s,t,h,simulations_per_period, plot)
{
  mu = getmu(s,t);
  sig = getsig(s,t);
  s0 = s[1];
  if(plot == TRUE) return(ts.plot(gbm(s0,mu,sig,h,simulations_per_period)))
  else return(gbm(s0,mu,sig,h,simulations_per_period))
}
```

We are going to use this simulation function to compare the distribution of the geometric brownian motion model with that of the empirical data through kernel density estimation, Monte Carlo simulation, and comparison of each distributions first four moments, but we also want to compare the empirical distribution and moments with the Merton jump diffusion model as well, so before actually fitting and simulating over any of the three time steps for the five equities, we will introduce and discuss the Merton jump diffusion model.

Merton Jump Diffusion Model

Merton proposes the follow modification to the geometric brownian motion SDE where some growth or decay factor (1+k) (where k is a random variable with expected value kbar + 1) is multiplied by the stock price and added to the price at any moment in time dt with jumps given by a Poisson process distribution with λ expected jumps per 1 unit time Poisson(λdt), leading to the SDE [2]

$$dS/S = (\mu_s - \lambda kbar)\, dt + \sigma_s\, dW + kbar*Poisson(\lambda dt)$$

The solution to this equation is considerably more rigorous, and because the purpose of this paper is to explore and interpret two models as opposed to comprehensively examining the methods of stochastic calculus, I jump (no pun intended) to the explicit solution

$S(W(t),t) = S_0\, e^{\sigma_s W + (\mu_s - kbar - \frac{1}{2}\sigma_s 2)t} * Y(n)$

Y(n) is referred to as a compound Poisson process. n is Poisson distributed with expected value λdt, Y(n) = 0 if n = 0, Π(1+k, 1, n) when n is greater than zero [2]. k is a random variable, and so far I've avoided talking about its distribution because both equations actually allows for any distribution of k [2]. However, Merton mostly spends his time discussing k as being log-normally distributing, i.e. $k \sim e^{N(\mu j, \sigma j\hat{}2)} - 1$, where subscript j denotes that these means and variances are for the jump process not the diffusion process [2]. Press also takes k to have this distribution, and it's often used in academia as well, so I proceed with this understanding of k [8,9]. The expectation of k, kbar, is found through the law of the unconscious statistician, which says that $E[g(X)] = \int g(x)f(x)dx$ from 0 to ∞ where f(x) is the probability density function of X, where g(x) is e^x and f(x) is the pdf of the normal distribution. It is mostly achieved by completing the square and rearranging terms so you have ∫f(x)dx from 0 to ∞, which is one, multiplied by a constant, which is then necessarily the expectation. This expected value is $e\hat{}(\frac{1}{2}\sigma_j^2 + \mu_j)$.

Now that we have defined the process for every random variable, I can create functions that simulate, first the compound Poisson process Y(n), and then the entire Merton jump diffusion process.

R Code for Compound Poisson Process, Y(n)

```
comppoi = function(simulations, lambda, mu_j, sig_j)
{
  y = 1:simulations;
  for (i in 1:simulations) { y[i] = 1;}
  y[1] = 1; #since we multiply by U, we want it's initial value to be unity
```

```
  for (i in 2:simulations) {
    x = 1;
    lam = rpois(1, lambda)
    if (lam > 0) {
    for (j in 1:lam) {x = x*exp(rnorm(1,mu_j,sig_j))}
    }
    y[i] = y[i-1]*x;
  }
  return(y);
}
```

Unlike the Wiener process, this simulation ends up approximating the process as opposed to replicating it. One advantage is that the sum of Poisson processes with different λ's is a single Poisson process whose λ is the sum of the individual λ's [2], but within my time steps I make no designation as to "where" the jump occurs. As the time steps approach zero, then this algorithm gets closer and closer to generate the jump at a single point, and so while the simulation is approximate, it is not biased.

R Code for Merton Jump Diffusion Simulation

```
mjd = function(s0, mu, sig,lambda, mu_j, sig_j, time,simulations_per_period)
{
  n = time*simulations_per_period;
  lambda = lambda/simulations_per_period;
  y = comppoi(n,lambda,mu_j,sig_j);
  kbar = exp(.5*sig_j^2+mu_j);
  stock = 1:n;
  w = wiener(n);
  for (i in 1:n) {
  stock[i] = s0*exp((mu/simulations_per_period - lambda*kbar/simulations_per_period -
.5*sig^2/(simulations_per_period))*i + sig*(1/simulations_per_period^.5)*w[i])*y[i]};
  return(stock);
}
```

MLE of μ_s, σ_s , λ, μ_j, σ_j for the Merton Jump Diffusion Model

In 1967, Press showed that there is no explicit solution for the MLE [8]. I provide a probability density function of the log-returns as a function of the five parameters. This was rigorously derived by Hanson in 2008 [10].

Probability(ln(return)) = sum(dpois(k,lambda*Δt) * dnorm(ln(return), (mu – 1/2*mu^2)Δt + mu_j*k, sig^2Δt + sig_j^2*k), k, infinity)

Where dpois(k, lambda) is the probability of a Poisson process with lambda generating k jumps, and dnorm(x, mu, sig) and sum(f(x), x, y, z) is the sum of f(x) as x goes from y to z.

R Code for Merton Jump Diffusion PDF

```
pdflogreturn = function(mu,sig,mu_j,sig_j,lam,t,quotes_per_period,logreturn)
{
  dt = t/quotes_per_period;
  sum = 0;
  for (i in 1:100) {
    sum = sum + (dpois(i,lam*dt)*dnorm(logreturn, (mu - 1/2*mu^2)*dt + mu_j*i, (sig^2*dt + sig_j^2*i)^.5))
  }
  return(sum);
}
```

Unfortunately, I had to cut off the summation at 100 for run-time purposes.

R Code for MLE of μ_s, σ_s , λ, μ_j, σ_j for the Merton Jump Diffusion Model

This Monte Carlo MLE is the weakest part of my anywhere. I couldn't figure out how to make a good first guess, and the algorithm seems to have run-time of somewhere between O(n) and $O(n^2)$, which makes it hard to run more than 100,000 trials (a sample size that would be fine for single variable optimization, but much less so for a five-dimensional one).

```
mle = function(trials,stock, bestguess, t)
{
  logreturn = mean(log(returns(stock)));
  quotes = length(stock);
  mu = bestguess[1];
  sig = bestguess[2];
  mu_j = bestguess[3];
  sig_j = bestguess[4];
  lt = bestguess[5];
  max = pdflogreturn(mu,sig,mu_j,sig_j,lt,t,quotes,logreturn);
  for (i in 1:trials) {
    imu = runif(1,-1,1);
    isig = runif(1,0,1);
    imu_j = runif(1,-1,1);
    isig_j = runif(1,0,1);
    ildt = runif(1,0,100);
    if(pdflogreturn(imu,isig,imu_j,isig_j,ildt,t,quotes,logreturn) >max) {mu = imu; sig = isig; mu_j = imu_j; sig_j =
isig_j; lt = ildt;}
  }
  return(c(mu,sig,mu_j,sig_j,lt))
}
```

R Code for simmjd Function

This is a pretty straight-forward function. The only caveat is that I still couldn't figure out a good algorithm for initial guesses, so my best guess for initial values was that MJD μ_s and σ_s would be close to BS μ_s and σ_s that positive and negative jumps would occur symmetrically such that $\mu_j = 0$ and $\sigma_s = .05$, and that they would occur, on average, once a time step (this is a pretty arbitrary guess as my time steps will vary between seconds and days, so my only real rationale was to pick the smallest natural number).

```
simmjd = function(trials, s, t, h, simulations_per_period, plot)
{
  x = mle(trials,s,c(getmu(s,t),getsig(s,t),0.1,.2,10), t)
  s0 = s[1];
  mu = x[1];
  sig = x[2];
  mu_j = x[3];
  sig_j = x[4];
  lam = x[5];
  if(plot == TRUE) return(ts.plot(mjd(s0,mu,sig,mu_j,sig_j, lam, h,simulations_per_period)))
  else return(mjd(s0,mu,sig,mu_j,sig_j, lam, h,simulations_per_period))
}
```

Goodness of Fit

Now that we have all of our MLE method and simulation methods, we can finally compare the two models to empirical distributions. Probability density functions will be estimated through the smoothing process, kernel density estimation, and I will install the only external package for this project, Lukasz Komsta's moments package, so that I can use the two function skewness() and kurtosis() [17]. I will then use Lukasz's package to create some summary statistic functions. I'm embedding the Monte Carlo simulations into two summary statistic methods for both models, but for kernel density estimation (whose purpose is mostly as a visual aid anyway) I will run each process once. Note: The graphs do not correspond to the Monte Carlo moment data. Unfortunately, MLE of MJD was too computationally intensive to be able to

nest that within another Monte Carlo simulation. I'm also aware that there are 252 trading days in a year, but I've used 256 because it was easier to partition that way (I factored this into any relevant arguments or methods so I don't understate the time).

R Code for Summary Statistic Functions

Function for empirical log-returns summary statistics

```
sumstat = function(stock) {
  lr = log(returns(stock))
  x = 1:4;
  x[1] = mean(lr);
  x[2] = var(lr);
  x[3] = skewness(lr);
  x[4] = kurtosis(lr);
  return(x)
}
```

Function for Black-Scholes geometric brownian motion log-returns summary statistics

```
simgbmsumstat = function(trials,stock,ismonth) {
  x = 0;
  if(ismonth) {
    for (i in 1:trials) {
      x = x + sumstat(simgbm(stock,23/52,23/52,52,FALSE)) }}
  else {for (i in 1:trials) {
    x = x + sumstat(simgbm(stock,256/252,256/252,256,FALSE))}}
  x = x/trials;
  return(x);}
```

Function for Merton jump diffusion log-returns summary statistics

```
simmjdsumstat = function(trials,stock,ismonth) {
x = 0;
if(ismonth) {
for (i in 1:trials) {
 x = x + sumstat(simmjd(1000,stock,23/52,23/52,52,FALSE))}}
else {
for (i in 1:trials)
{ x = x + sumstat(simmjd(1000,stock,256/252,256/252,256,FALSE)) } }
x = x/trials;
return(x);}
```

Short Term Goodness of Fit

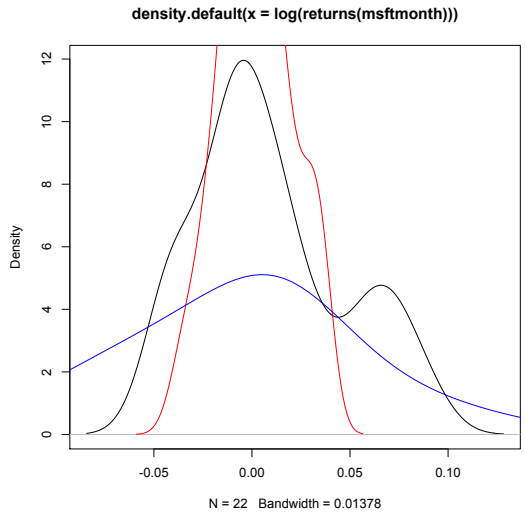
Reiterated Note: The graphs do not correspond to the Monte Carlo moment data. Unfortunately, MLE of MJD was too computationally intensive to be able to nest that within another Monte Carlo simulation.

For the short term, we look at stock price data between March 17th and April 17th of 2020.

MSFT [12]

	mean,	variance,	skewness,	and kurtosis of log-returns
Empirical:	0.008983887	0.001450231	0.543404617	2.338826842.
Monte Carlo Estimation (10,000 trials) of GBM:	0.009006361	0.001441189	0.001932815	2.739333059
Monte Carlo Estimation (100 trials) of MJD:	-0.04444880	0.28019737	-0.00719729	3.94780889

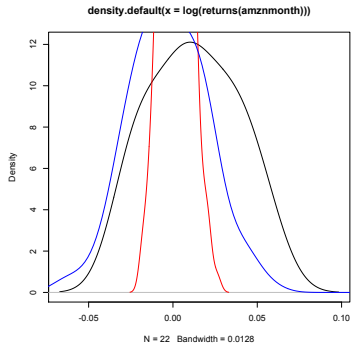
In this case, all four of the MJD's central moments are less accurate than Geometric Brownian Motion, and the MLE estimates that there are 64 jumps per month. Black is the KDE of the empirical log returns, red is for GBM and blue is for MJD. The plot illustrates the radically higher kurtosis and variance. And the AIC score for GBM is in fact lower (166 vs 186)



AMZN [13]

	mean,	variance,	skewness,	and kurtosis of log-returns
Empirical:	0.0124029406	0.0006961273	0.0635019505	1.9545168931.
Monte Carlo Estimation (10,000 trials) of GBM:	0.0121876187	0.0006833264.	-0.0017579236	2.7407545596
Monte Carlo Estimation (100 trials) of MJD:	-0.02371200	0.28344853	0.03905755	3.78368053

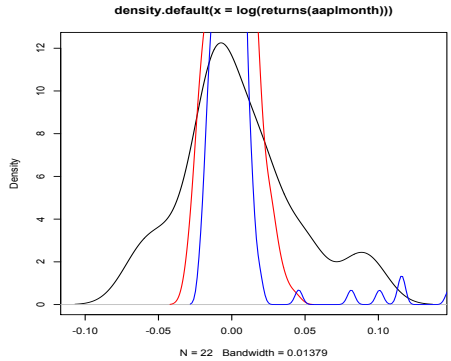
In this case, MJD has a closer skewness, but its other three moments are farther off. It is also considerably more leptokurtotic. However, this single simulation of the three models shows and MJD that looks much closer, so perhaps the discrepancy has less to do with the underlying process and just the parameter estimation. Finally, GBM has considerably lower AIC (158 vs 273).



AAPL[14]

	mean,	variance,	skewness,	and kurtosis
Empirical:	0.005086541	0.001600851	0.532190707	3.076316368.
Monte Carlo Estimation (10,000 trials) of GBM:	0.004939500	0.001571287	0.003577597	2.718919716
Monte Carlo Estimation (100 trials) of MJD	-0.00309087	0.31264448	0.15787746	3.77222779

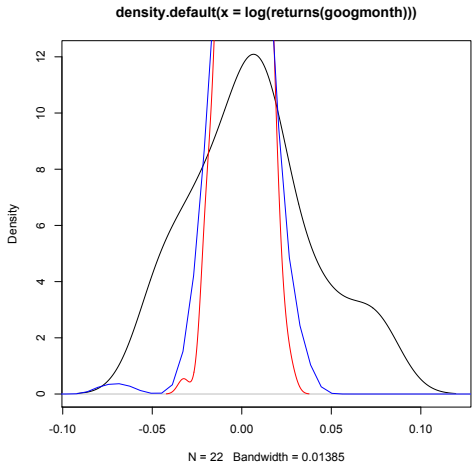
MJD has a closer skewness, but its other three moments are farther off, and it's kurtosis is high . However, *MJD* does has the lower AIC (173 vs 195).



GOOG [15]

	mean,	variance,	skewness,	and kurtosis
Empirical:	0.006192990	0.001196659	0.361126644	2.613310069.
Monte Carlo Estimation (10,000 trials) of GBM	0.005951110	0.001183489	-0.029559300	2.747292262
Monte **Carlo** Estimation (100 trials) of MJD	-0.04969167	0.27806658	0.07832069	3.77925135

MJD has a closer skewness, with three other moments farther off, and it's kurtosis is high, again around 3.7. GBM has the lower AIC (176 vs 202).

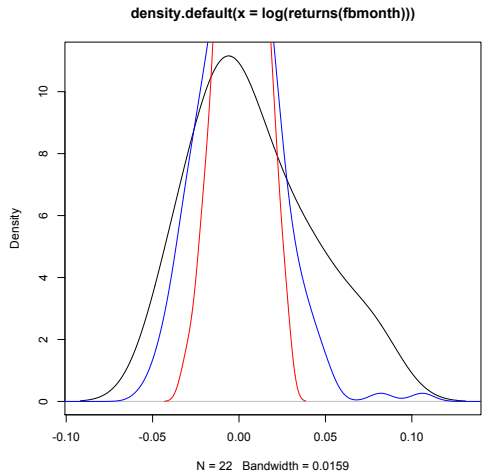


The main takeaway from the graph, however, is that both processes look much more similar to each other than the empirical data.

FB [16]

	mean,	variance,	skewness,	and kurtosis
Empirical:	0.008271118	0.001226116	0.528010835	2.480173992
Monte Carlo Estimation (10,000 trials) of GBM	0.008337799	0.001230209	-0.006410072	2.775389578
Monte Carlo Estimation (100 trials) of MJD	-0.02293837	0.32788756	0.07602602	3.96078344

MJD has a closer skewness, with three other moments farther off, and it's kurtosis is high. However, it is not around 3.7 anymore. Interestingly, MJD has the lower AIC (160 vs 169). It's also interesting to note that MJD looks closer to the empirical distribution in this single simulation.



Before looking at the goodness of fit for the long term, we observe that geometric brownian motion does a markedly better job of recreating moments 1,2, and 4 than a merton jump diffusion process fit (although this jump process was fit with significantly low power because of the computational efficiency issues). It's interesting how poor geometric brownian motion is at capturing skew in these large cap technology companies, especially positive skew. It is also is generally less leptokurtotic. There were some occasional glimpses of hope for the Merton jump diffusion model, (e.g. two higher AIC's and two graphs that seemed to demonstrate closer correspondence between merton jump diffusion and reality). I suspect the weakness of Merton Jump diffusion in this process is because of my poor MLE algorithm as opposed to the strength of the model, and that's useful to note – without very precise optimization, and thus parameter estimation, Merton Jump diffusion is considerably inaccurate.

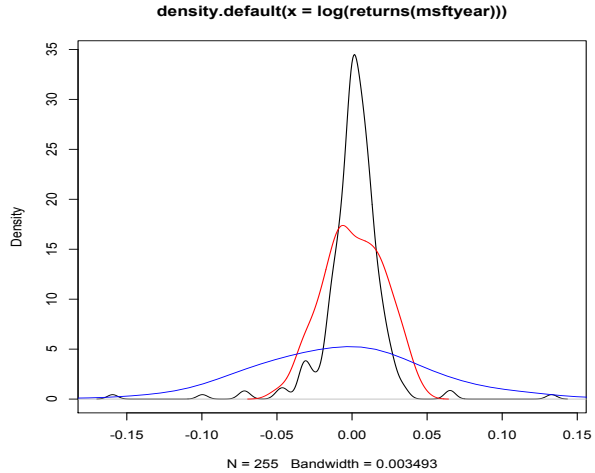
Long Term Goodness of Fit

From March 11th, 2019 to March 16th, 2020

MSFT [12]

	mean,	variance,	skewness,	and kurtosis of log-returns
Empirical:	0.0006883195	0.0004700251	-1.3093511328	21.3390786793
Monte Carlo Estimation (10,000 trials) of GBM:	0.0007399786	0.0001086195	-0.0019789178	2.9853218806
Monte Carlo Estimation (100 trials) of MJD:	-0.0002281761	0.0125974091	0.0694807435	14.3057727567

The Merton jump diffusion model is able to replicate the high kurtosis, but nothing else in this case, and GBM has a lower AIC (2086 vs 2149). The log-return distribution for the Merton jump diffusion process looks radically different, but MJD was a poor model of Microsoft over one month as well.

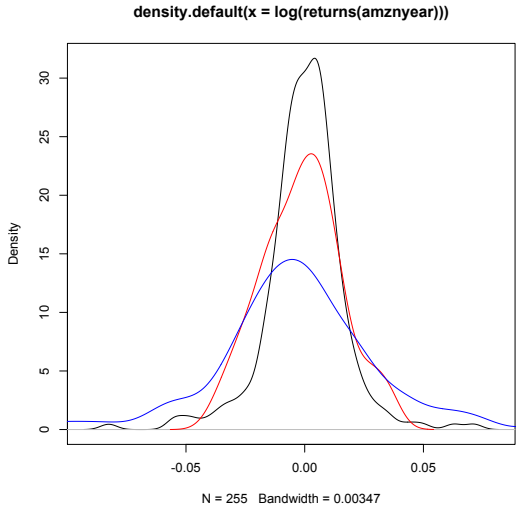


AMZN [13]

	mean,	variance,	skewness,	and kurtosis
Empirical:	3.744031e-05	2.816034e-04.	-3.554555e-01	7.555980e+00
Monte Carlo Estimation (10,000 trials) of GBM:	0.0007164013	0.0001081573	0.0007718511	2.9675627260
Monte Carlo Estimation				

(100 trials) of MJD	0.007534251	0.133915417	0.003269395	13.932726829

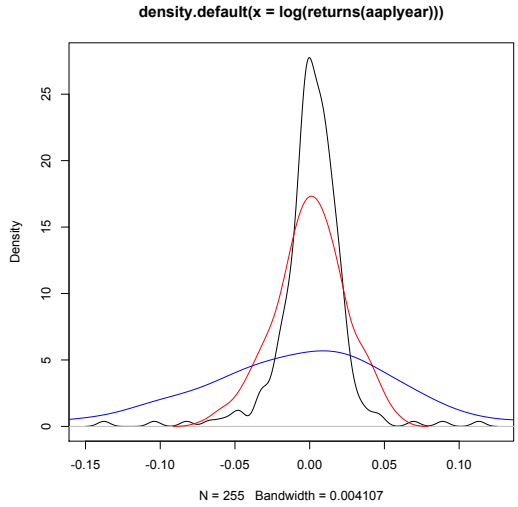
The Merton jump diffusion has worse estimates of all four moments, and the Black-Scholes geometric brownian motion model has a lower AIC (3019 vs 3085). It's clear from that the plot that the MJD model has much larger second and fourth moments than the empirical distribution. Although GMB does better, the distribution of log-returns under the MJD model in the plot looks somewhat similar to those of the empirical distribution and the geometric brownian motion model.



AAPL [14]

	mean,	variance,	skewness,	and kurtosis
Empirical:	0.0011443351	0.0005202553	-0.9441280468	12.8532696586
Monte Carlo Estimation (10,000 trials) of GBM:	0.0011753205	0.0005151078	0.0020952492	2.9809303175
Monte Carlo Estimation (100 trials) of MJD	0.002728591	0.160642485	0.313797661	12.985112131

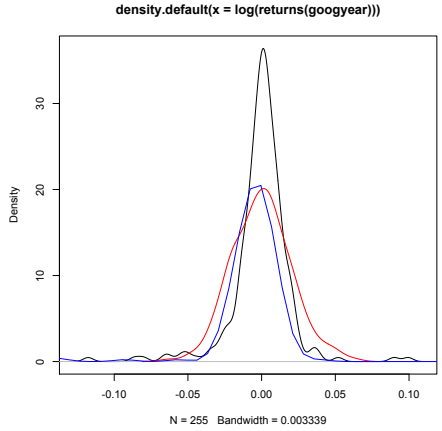
The Merton jump diffusion model produces a very similar kurtosis, but too much variance, the wrong direction of skewness, and a slightly worse estimate of the mean log-returns. GMB has an AIC of 2356 compared to MJD's 2619.



GOOG [15]

	mean,	variance,	skewness,	and kurtosis
Empirical:	-0.0003752019	0.0004067570	-0.8637024434	12.8375083627
Monte Carlo Estimation (10,000 trials) of GBM:	-0.0003707753	0.0004007185	-0.0100067690	2.9831672744
Monte Carlo Estimation (100 trials) of MJD	0.004167406	0.144482281	-0.091291494	13.599225873

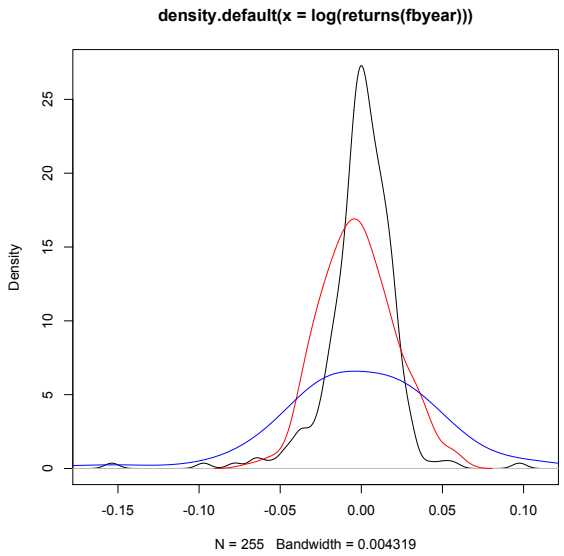
The Merton jump diffusion model produces a similar kurtosis and a better estimate of the skewness in the case of GOOG. Remarkably, both models produce an AIC of 3155. Lastly, the distributions look very similar.



FB [16]

	mean,	variance,	skewness,	and kurtosis
Empirical:	-0.0006406046	0.0004933777	-1.6705654424	14.1953780674
Monte Carlo Estimation (10,000 trials) of GBM:	-0.0006439601	0.0004847060	-0.0086410570	2.9781939715
Monte Carlo Estimation (100 trials) of MJD	0.006220536	0.174478808	0.316072487	13.932173734

The Merton jump diffusion model produces a better kurtosis estimate, but that is it. The GBM produces a slightly lower AIC (2046 vs 2060).



Discussion

First of all, the computational weakness of my maximum likelihood estimation, and thus the poorness of the parametrization of the Merton jump diffusion process from stock price time series data only allows me to make speculative inferences about the model's efficacy as opposed to definite conclusions. However, the geometric brownian motion process was parametrized with the exact MLE solutions, and thus definite claims can be made about its absolute (as opposed to relative) efficacy in modeling price movement in large cap technology stocks. Despite the

weakness of my MJD parametrization, the model seemed to model stock price movement possibly better in the short term for AMZN and FB, looking at moments, AIC's, and density distributions. It also appeared slightly more accurate than GBM for GOOG in the long run (more accurate moments, an equal AIC, and a distribution closer to the black line representing the empirical distribution).

Fundamentally, it's interesting that Apple and Microsoft were the two companies who were not modelled well by Merton jump diffusion over either window (one month or one year). While both companies are technology companies, they aren't web service companies as Amazon, Facebook, and Google are. Perhaps there is something about either our expectations of web service company earnings or their actual realization that lends itself better to the incorporation of a compound Poisson process. In some sense, this makes a lot of sense as consumers of these companies have no cost of switching services, where as changing one's computer or operating system generally entails time and selling equipment at a discount. I would like to do more work exploring the goodness of fit for Merton jump diffusion as a model for other non-subscription, service businesses. However, to more concretely examine what the Poisson jumps truly are, a much better method of parametrization needs to be devised. With a better MLE algorithm and thus better parametrization, further work could explore correlation between google trends data and jump height and frequency, but this relies on a much greater level of confidence in λ, μ_j, and σ_j. With better parametrization, it would be interesting to see if the MJD still fails to reproduce similar variance – this was the most inaccurate moment for this process, often being off by an order of magnitude.

Log-returns of geometric brownian motion with static drift and volatility coefficients are going to be normally distributed, and the problem with the normal distribution is that the

skewness is always 0 and the kurtosis is always 3. Analysis of the empirical data shows that there is almost *always* significant skewness and a kurtosis that seems to hover around 3 on a one month window but that increases with the size of the window. In the long-run, we can almost definitively rule out this type of geometric brownian motion as a completely accurate process for modeling these sorts of equities, but it is very useful, and relatively intuitive, so that doesn't mean we should disregard it. A further point of study is whether the log-returns of the more general geometric brownian motion with non-static drift and volatility can avoid this trap and produce different values for skewness and kurtosis.

Looking again just at the empirical log-returns, it's interesting Four out of the five companies had significant negative skew between 3/12/20 and 3/16/20, where as those same companies had significant positive skew between 3/17/20 and 4/17/20. Is it just because of the global pandemic that the sign of the skew flipped?

Conclusion

1. Geometric brownian motion with static drift and volatility does an excellent job at modeling the mean and variance of log-returns, but a poor job adjusting for non-zero skewness and excess kurtosis.
2. Empirical log-returns for large cap tech companies seem to have a kurtosis that is proportional to some power of time, a property they share with the Merton jump diffusion processes. Further work can be done to examine other processes with kurtoses that are functions of time.
3. The weakness of drawing uniformly distributed random variables on best-guess intervals limited our ability to analyze the efficacy of the Merton jump diffusion process more

generally, and also to materially interpret the Poisson process. As there is no explicit solution [8] , better numerical techniques need to be applied to this parametrization through MLE. With better parametrization, it will be interesting to see if the Merton jump diffusion process stops overestimating Var(log-returns).

4. More analysis needs to be done in regards to the effect of a time window on skewness of actual stock log-returns.
5. Large companies with service business models that entail minimal costs of switching may be modeled well by the Merton jump diffusion with the compound Poisson process

This paper represents my own work in accordance with University regulations.

Robert von der Schmidt

Works Cited

1. Black, Fischer, and Myron Scholes. “The Pricing of Options and Corporate Liabilities.” Journal of Political Economy, vol. 81, no. 3, 1973, pp. 637–654., doi:10.1086/260062.
2. Merton, Robert C. “Option Pricing When Underlying Stock Returns Are Discontinuous.” Journal of Financial Economics, vol. 3, no. 1-2, 1976, pp. 125–144., doi:10.1016/0304-405x(76)90022-2.
3. Duffie, Darrell. “Black, Merton and Scholes - Their Central Contributions to Economics.” Scandinavian Journal of Economics, vol. 100, no. 2, 1998, pp. 411–424., doi:10.1111/1467-9442.00110.
4. Levy, Haim. “The CAPM Is Alive and Well: A Review and Synthesis.” European Financial Management, vol. 16, no. 1, 2010, pp. 43–71., doi:10.1111/j.1468-036x.2009.00530.x.
5. Shreve, Steven E. Stochastic Calculus for Finance II: Continuous-Time Models. Springer, 2010. p.114
6. Itō Kiyoshi, and Henry P. McKean. Diffusion Processes and Their Sample Paths. Springer, 1996.
7. Kao, Edward P. C. An Introduction to Stochastic Processes. Dover Publications, 2020. Chapter 7.
8. Press, S. James. “A Compound Events Model for Security Prices.” The Journal of Business, vol. 40, no. 3, 1967, p. 317., doi:10.1086/294980.
9. Yuh-Dauh Lyuu, National Taiwan University, Course Lecture. https://www.csie.ntu.edu.tw/~lyuu/finance1/2015/20150513.pdf
10. Hanson, Floyd B., and John J. Westman. “Stochastic Analysis of Jump-Diffusions for Financial Log-Return Processes.” *Stochastic Theory and Control Lecture Notes in Control and Information Sciences*, pp. 169–183., doi:10.1007/3-540-48022-6_13.
11. “Stock Research and Analysis.” Macrotrends, www.macrotrends.net/stocks/research.
12. https://finance.yahoo.com/quote/MSFT/history?p=MSFT
13. https://finance.yahoo.com/quote/AMZN/history?p=AMZN
14. https://finance.yahoo.com/quote/AAPL/history?p=AAPL
15. https://finance.yahoo.com/quote/GOOG/history?p=GOOG
16. https://finance.yahoo.com/quote/FB/history?p=FB
17. “Package Moments.” CRAN, cran.r-project.org/web/packages/moments/index.html.