FinM 331/Stat 339 Financial Data Analysis, Winter 2010

Floyd B. Hanson, Visiting Professor

Email: fhanson@uchicago.edu

Master of Science in Financial Mathematics Program
University of Chicago

Lecture 3

6:30-9:30 pm, 18 January 2009, Ryerson 251 in Chicago

7:30-10:30 pm, 18 January 2009 at UBS in Stamford

7:30-10:30 am, 19 January 2009 at Spring in Singapore

3. Exploratory Data Analysis Tools, Continued:

3.1. Histograms — Bin/Bar Graphics of Data Distributions:

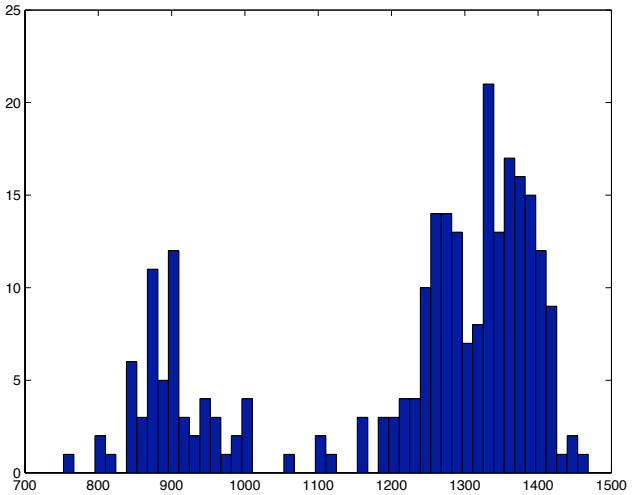
A **histogram function** produces **discrete graphical representation of a density functions** with bars (rectangles) for each bin (intervals) from empirical or computed data.

In **`MATLAB`**, vanilla or **`Statistical Toolbox`** versions, a typical format is

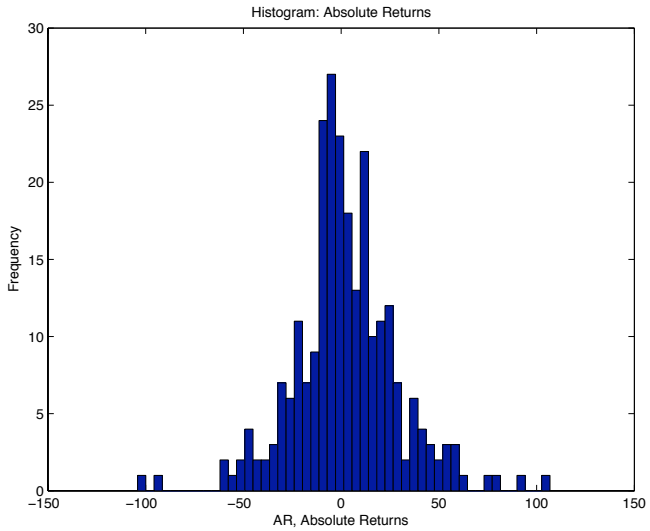
```
hist(Data,Nbins);
```
(3.1)

where **`Data`** is the vector or matrix data, **`NBins`** is the number of bins that data is divided into after it has been ordered by size. The function **`hist`** plots the frequency counts in each bin. In the plot, the x-axis is the **bin axis**, the y-axis is the **frequency axis** and the plot itself is a special kind of bar graph that represents the **quantile interval frequencies**. Since the frequencies are unscaled, the histogram is related to the density, but **lacks the normalization** that forces the density to satisfy the conservation of probability.

- ***Weak Examples Histograms for Financial Data:***



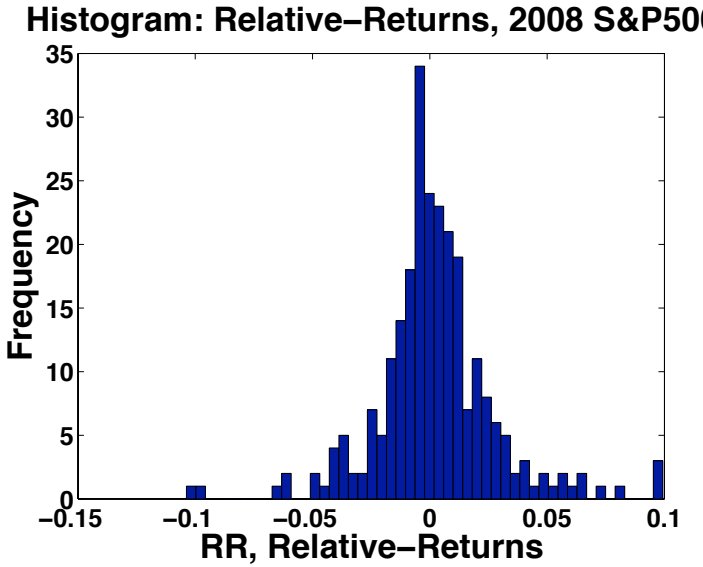
(a) S&P500 Adjusted Closings 2008.



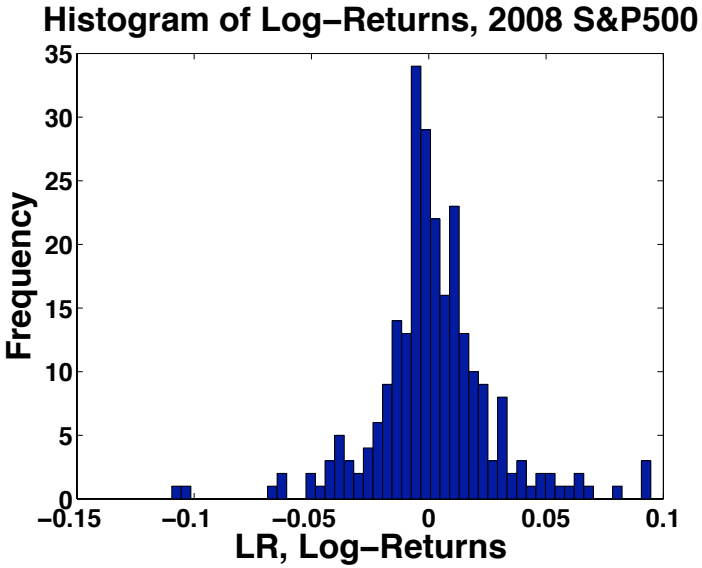
(b) S&P500 Absolute Returns 2008.

Figure 3.1: Histograms of **S&P500 Index** (quote $\hat{}$**GSPC**) data for the whole year 2008 using $\texttt{NBins} = 50$ bins for $n + 1 = 254$ closings. Subfigure 1(a) displays the **adjusted closings** S_i for $i = 1 : n + 1$ and is not a useful plot, but is a very poor plot since it lacks title and axes labels. Subfigure 1(b) displays the **absolute returns** $\mathrm{AR}_i = S_{i+1} - S_i$ for $i = 1 : n$ and is a somewhat better plot, but title, labels and axis numbers are hardly readable for a presentation.

- ***Better Examples Histograms for Financial Data:***



(a) S&P500 Absolute Returns 2008.



(b) S&P500 Log-Returns 2008.

Figure 3.2: Histograms of **S&P500 Index** (quote $\hat{}$**GSPC**) data for the whole year 2008 using $\texttt{NBins} = 50$ bins for $n + 1 = 254$ closings. Subfigure 2(a) displays the **relative returns** $\mathrm{RR}_i = S_{i+1}/S_i - 1$ for $i = 1:n$ and these two plots are much better as well as more professional for the space available. Subfigure 2(b) displays the **log-returns** $\mathrm{LR}_i = \log(S_{i+1}) - \log(S_i)$ for $i = 1:n$ and is a the preferred variable for plotting financial data.

- ***Histogram Code Examples*** (abbreviated) for Figures 3.1–3.2:

```
function histspc2008
% Get Histogram for S&P500 ^GSPC (Yahoo Finance) for Year 2008
%   Dates 2007/12/31-2008/12/31, Daily Adjusted Closings.
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
S = textread('GSPC2008adjC.txt','%f'); % Xcel.cvs deleted to 1 column.
L = length(S);
fprintf('\nmean(S)=%6.1f; std(S)=%5.1f; length(S)=%3i;',mean(S),std(S),L);
figure(1);
hist(S,50); % Note: Poor Toy or Amateur Plot;
%
figure(2);
AR = S(2:L)-S(1:L-1); % Note: Vector Subtraction of New - Old Prices;
fprintf('\nmean(AR)=%5.3f; std(AR)=%5.2f;',mean(AR),std(AR));
hist(AR,50);
title('Histogram: Absolute Returns'); % Note Not Much Better Plot ...
xlabel('AR, Absolute Returns'); % since Titles and Label Hard to Read
ylabel('Frequency');
%
figure(3); % Note: (S2-S1)/S1 = S2/S1-1; Saving Comp. Cost for 2X S1;
RR = S(2:L)./S(1:L-1)-1; % Note: Dot or Componentwise Division!
fprintf('\nmean(RR)=%8.6f; std(RR)=%7.5f;',mean(RR),std(RR));
hist(RR,50);
```

```
title('Histogram: Relative-Returns, 2008 S&P500'...
    ,'Fontsize',24,'FontWeight','Bold'); % Now Much Better Plot;
xlabel('RR, Relative-Returns','Fontsize',24,'FontWeight','Bold');
ylabel('Frequency','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',18,'FontWeight','Bold'); % gca = get current axis
%
figure(4);
LR = log(S(2:L))-log(S(1:L-1)); % Note: Vector Log Difference!
fprintf('\nmean(LR)=%8.6f; std(LR)=%7.5f;\n',mean(LR),std(LR));
hist(LR,50);
title('Histogram of Log-Returns, 2008 S&P500'...
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('LR, Log-Returns','Fontsize',24,'FontWeight','Bold');
ylabel('Frequency','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',18,'FontWeight','Bold'); % gca = get current axis


Output histspc2008.m:
mean(S)=1221.0; std(S)=191.7; length(S)=254;
mean(AR)=2.234; std(AR)=26.82;
mean(RR)=0.002256; std(RR)=0.02592;
mean(LR)=0.001921; std(LR)=0.02583;
>>
```

- ***More on Histograms:***

  The source for the **MATLAB** code on the previous two pages is at *Histogram histspc2008 code*.

  The number of bins, **NBins**, should be selected carefully: not too large causing raggedness from too many empty bins or too small so that it looks like a uniform distribution.

  Alternatively, the bin mark locations can be specified in a vector, say **BinLoc** which can be set manually such as letting **BinLoc=a:bw:b** be the bin location vector given a data range **[a,b]** and bin width **bw**, then using

  ```
  hist(Data,BinLoc);
  ```
  (3.2)

  and **MATLAB** should detect whether you are using a bin count or bin location vector. The range can be determined by the commands **a=min(Data)** and **b=max(Data)** if **Data** is a vector, but in rare instances some leeway outside this range might be needed.

- ***What About Cumulative Histogram for Discrete Distributions?***

  The function **`hist`** with output arguments,

  ```
  [BinCount,BinLoc]=hist(Data,Nbins);
  ```
  (3.3)

  along with the **cumulative sum function**, **`cumsum`**,

  ```
  CumBinCnt=cumsum(BinCount);
  ```
  (3.4)

  and the more powerful **bar graph function**, **`bar`**,

  ```
  bar(BinLoc,CumBinCnt);
  ```
  (3.5)

  can also generate discrete analogs of the Cumulative Distribution Function (**CDF**). The easier to use data import function **`load`** is also used in the sample code ***Histogram histspc2008dist code*** following **Figure 3.3**.

- ***Cumulative Histogram Figure for Financial Data:***

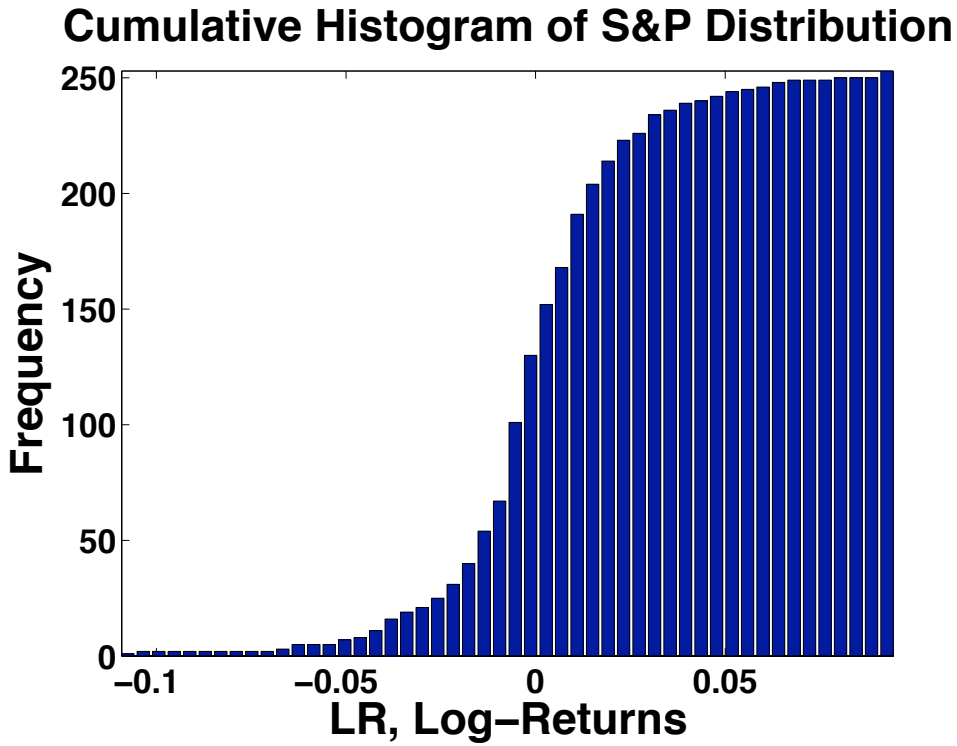


Figure 3.3: Cumulative Histogram of **S&P500 Index** (quote $\hat{}$**GSPC**) data for the whole year 2008 using $\texttt{NBins} = 50$ bins for $n+1 = 254$ closings. Figure displays **cumultative log-returns** where the ith log-return is $\mathbf{LR}_i = \log(S_{i+1}) - \log(S_i)$ for $i = 1:n$. Notice how `axis tight` command made a nicer and more compact figure than the previous histograms.

- ***Cumulative Histogram Code Example*** (abbreviated) for Fig. 3.3:

```
function histspc2008dist
% Get Cumulative Histogram for S&P500 ^GSPC (Yahoo Finance) for Year 2008
%   Dates 2007/12/31-2008/12/31, Daily Adjusted Closings.
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
% Another simpler import data file method!
load -ASCII S08.mat; % Note: Change GSPC2008adjC.txt name for load function.
L = length(S08);
%
LR = log(S08(2:L))-log(S08(1:L-1)); % Note: Vector Log Difference!
fprintf('\nmean(LR)=%8.6f; std(LR)=%7.5f;',mean(LR),std(LR));
[BinCount,BinLoc]=hist(LR,50); % Get Bin Count and Location Output!
%
figure(5);
CumBinCnt = cumsum(BinCount); % cumsum does the Cumulative Sums
bar(BinLoc,CumBinCnt); axis tight;
title('Cumulative Histogram of S&P Distribution'...;
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('LR, Log-Returns','Fontsize',24,'FontWeight','Bold');
ylabel('Frequency','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',18,'FontWeight','Bold'); % gca = get current axis
```

3.2. Kernel Smoothing Functions:

The **main disadvantage of the histogram is that it is ragged or nonsmooth**, more so the smaller the bin width. The estimation of the density, without normalization of frequencies, is done from the bar height over the bins. This nonsmooth feature can be fixed, in part if the corresponding width is not too small, using a **kernel smoother**, where the **kernel $K(x)$** is a density or something similar that integrates to one, e.g., a standard normal or uniform or triangular density. In order to accommodate the complexity of **real sample data**, a set of of n observations, $[X_i]_{n\times 1}$, is approximated by an **estimated smoother** $\widehat{f}_X^{(s)}(x)$. The smoother is a **sum or mixture** of n terms using this kernel evaluated by a scaled variable centered about each X_i,

$$\widehat{f}_X^{(s)}(x) = \frac{1}{n \cdot x_{bw}} \sum_{i=1}^{n} K\left(\frac{x - X_i}{x_{bw}}\right), \tag{3.6}$$

where x_{bw} is the **scaling or bandwidth** that would be the standard deviation if the kernel were the standard normal or Gaussian density.

- ***More Kernel Smoothing:***

  Note that the extra division by x_{bw} follow from conservation of probability, i.e.,

$$1 = \int_{-\infty}^{+\infty} K(z)dz = \int_{-\infty}^{+\infty} K\left(\frac{x - X_i}{x_{bw}}\right)\frac{dx}{x_{bw}}, \qquad (3.7)$$

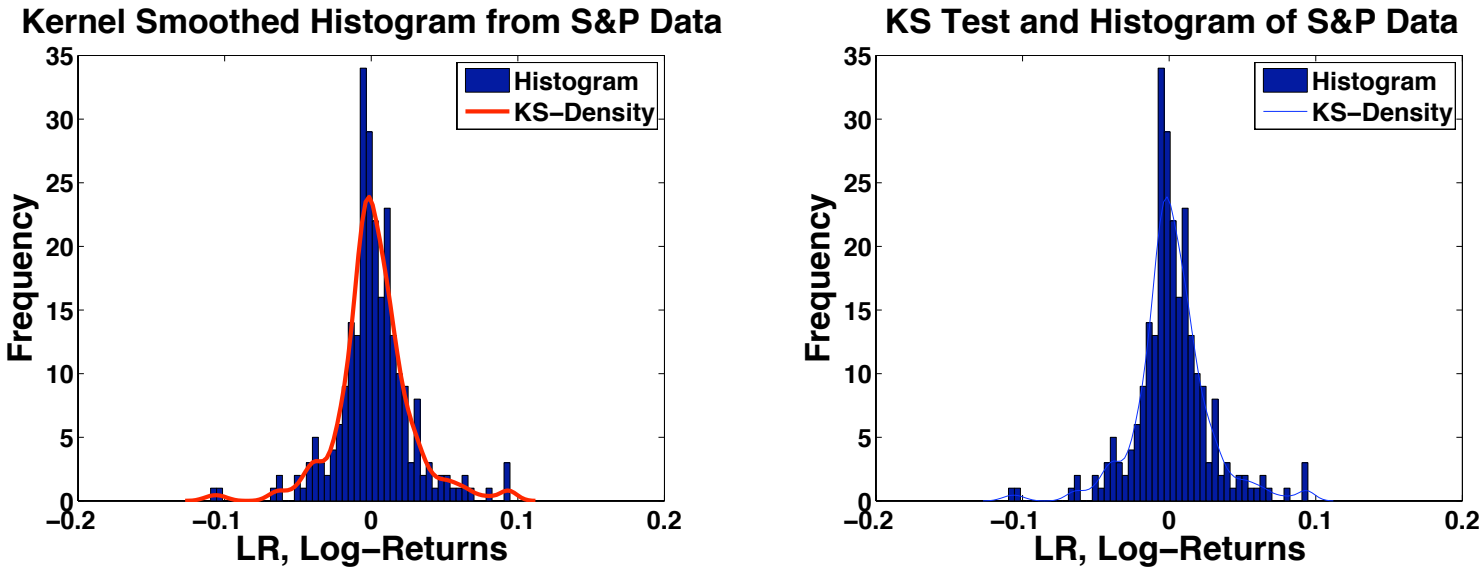
  where the calculus change of variables $z = (x - X_i)/x_{bw}$ has been used and an infinite domain is assumed, but a finite domain function, like the uniform density, can be used with compact support.

  In the **`Statistical Toolbox`**, **`ksdensity`** is the **kernel smoothing density estimation** function. A typical format would be the output-input form,

```
[fs,xs] = ksdensity(X);
```
(3.8)

  where **`X`** is the sample data **`n`**-vector, **`fs`** is the output = **`n`**-vector of estimated density values at the output **`n`**-vector of points **`xs`**. For a sample plot of an estimated density, use a professionally produced version of **`plot(xs,fs);`**, but see the following Figure 3.4.

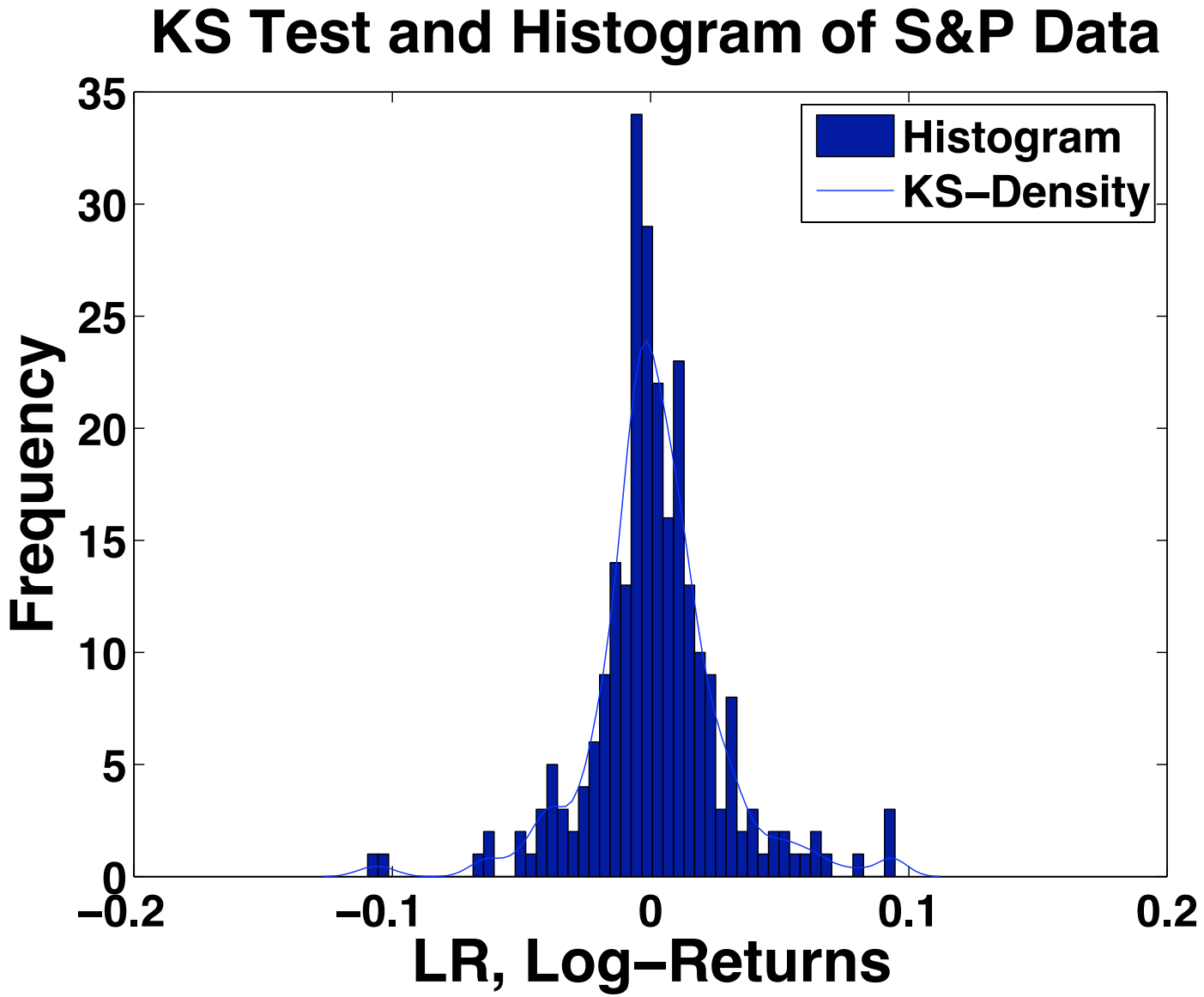
- ***Kernel Smoothing and Histogram for 2008 Log-Returns Figure:***



(a) Kernel Smoothing and Histogram. (b) Same without output arguments.

Figure 3.4: Estimated kernel smoothing density is compared to a prior histogram for **S&P500 Index** (quote^GSPC) **log-returns** $\mathbf{LR}_i = \log(S_{i+1}) - \log(S_i)$ for $i = 1:n$ for the whole year 2008. Note that the (a) graph uses **`ksdensity`** with output arguments and the **`plot`** and is quite readable, while the (b) graph uses only **`ksdensity`** without output arguments that automatically produces a plotted figure and the kernel estimation is essentially ***unreadable in the overlay onto the histogram***, but does not seem to be easily fixed.

- *Poor Kernel Smoothing and Histogram Subfigure (b) Enlarged:*



- ***Kernel Smoothing and Histogram Code Example for Figure 3.4:***

```
function histspc2008ks
% Get Kernel Smoothed Histogram for Log-Returns Density
% for 2008 S&P500 ^GSPC (Yahoo Finance) Data;
%   Dates 2007/12/31-2008/12/31, Daily Adjusted Closings.
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
% Another simpler import data file method!
load -ASCII S08; % Note: Change GSPC2008adjC.txt name for load function.
L = length(S);
figure(6);
LR = log(S(2:L))-log(S(1:L-1)); % Note: Vector Log Difference!
fprintf('\nmean(LR)=%8.6f; std(LR)=%7.5f;',mean(LR),std(LR));
hist(LR,50); hold on; % hold hist for adding ks-density;
[fz,z] = ksdensity(LR);
plot(z,fz,'-r','LineWidth',3); hold off;
title('Kernel Smoothed Histogram from S&P Data'...;
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('LR, Log-Returns','Fontsize',24,'FontWeight','Bold');
ylabel('Frequency','Fontsize',24,'FontWeight','Bold');
legend('Histogram','KS-Density');
set(gca,'Fontsize',18,'FontWeight','Bold'); % gca = get current axis
```

- ***Even More on Kernel Smoothing:***
  - The **default kernel is normal**, but other kernels can be requested by adding to the input arguments the **parameter-value pairs**, written as literals within single quotes, such as `,'kernel','box'` presumably for a uniform kernel or `,'kernel','triangular'`.
  - Similarly, the bandwidth parameter can be changed using the parameter `'width'` to reveal more features, but the **default normal kernel has an optimal width, i.e., the standard deviation**. The used width can be recovered by replacing the output set `[fs,xs]` to `[fs,xs,xbw]` where `xbw` is the used width (binwidth).
  - The output estimated function type can be specified pairing the parameter `'function'` with the value `'cdf'` for a CDF or with `'icdf'` for the inverse-cumulative-distribution function.
  - For more, see `Help` menu option in `MATLAB`, e.g., see the **GUI fitting tool** of the toolbox, the `dfitool` which helps you do a lot of the tasks you may need to do without real coding.

3.3 Quantiles of Ordered Statistics:

- ***Quantiles:***

  Quantiles are a way of summarizing a theoretical or empirical distribution function $F_X(x)$ by specifying the values of the distribution variate $x = q_i$ called the **ith-quantiles** for $i = 1:m-1$ that divide the distribution in to m quantile groups or **m-quantiles**, $[q_{i-1}, q_i]$ for $i = 1:m$, usually of even probabilities or frequencies $p_i \in [0,1]$ for $i = 1:m$. The extra **tile marks** are **minimum**, $q_0 = \min[x]$, and the **maximum**, $q_m = \max[x]$, which may be infinite in case of distributions such as the theoretical normal or exponential distributions. When convenient, we will refer to the complete set q_i as **tile marks** for $i = 0:m$. The quantiles are defined implicitly by

$$F_X(q_i) \equiv \text{Prob}[X \leq q_i] = p_i, \tag{3.9}$$

  for $i = 1:m$.

- ***Examples of Quantiles:***

  The set of both $q_0 = \min[x]$ and $q_m = \max[x]$ would define the most trivial of quantiles, the **1-quantile** or **uniquantile**. However, the simplest of quantiles that would be considered are the **quartiles** such that quartile probability vector is $\vec{p} = [0.0, 0.25, 0.5, 0.75, 1.0]$ defining the total quartile marks named the minimum, lower quarter, median, upper quarter and maximum, or

  $$q_i = F_X^{-1}(p_i) \tag{3.10}$$

  for $i = 0:4$, assuming an inverse distribution exists and is unique, both could be violated if there are jump and discontinuities.

  Another commonly used quantiles are the percentiles with $m = 100$, where $\vec{p} = [0.01 * i]_{100\times 1} = [0.01:0.01:1.00]$, the last in **MATLAB** loop construct notation with the percentile mark vector given by $\vec{q} = [F_X^{-1}(p_i)]_{100\times 1} = F_X^{-1}(\vec{p})$, in MATLAB vector notation, as in the quartile inversion formula above, but with $m = 100$. Recall that **MATLAB** has inverse distribution functions.

- ***MATLAB quantile function:***

  Recall that **MATLAB** has inverse distribution functions for its distributions, e.g., the **Statistics Toolbox** generic

  ```
  F_x=cdf('Name',Xvector,Parmvector);
  ```
  (3.11)

  there is the generic inverse distribution

  ```
  Finverse_p=icdf('Name',Pvector,Parmvector);
  ```
  (3.12)

  However, the Statistics Toolbox has a built-in function that does quantiles,

  ```
  QuantVector=quantile(DataVector,ProbVector);
  ```
  (3.13)

  where veciors `[DataVector,ProbVector,QuantVector]` are of size `[n,m,m]`, where `n` is sufficiently large to yield well-defined quartile marks. See **MATLAB help** for data sort and a not so simple quantile construction.[a]

[a] These Lecture 3 Notes come partly from **MATLAB** (registered trademark of Mathworks, Inc.) Help, testing, Carmona (2004) and the necessary Hanson's interpretations, motivations and experiences.

- ***Recall Kernel Smoothing `ksdensity` and Histogram `hist` for 2008 Log-Returns Figure:***

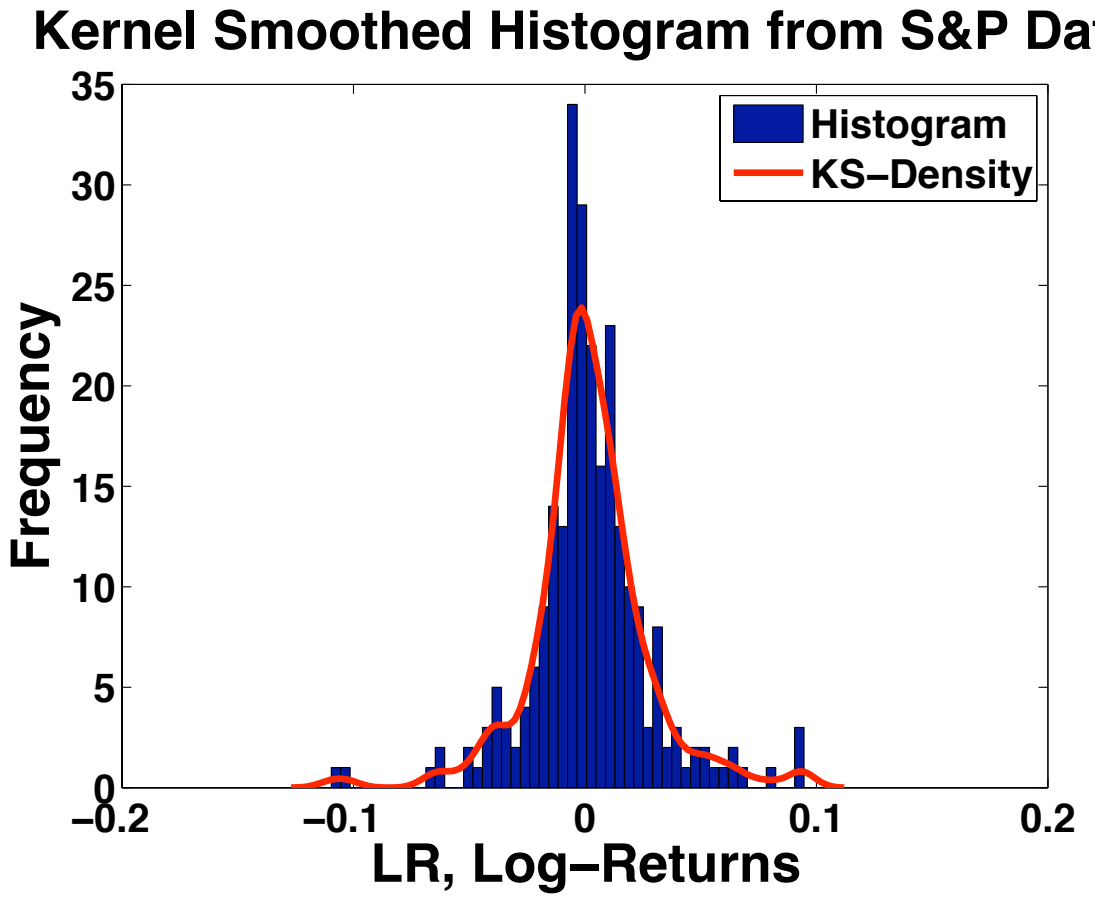


Figure 3.5: Note the ***two kernel bumps in the two tails*** made by two little normal kernels fitted to the **tail nonnormal behavior** of the **normal kernel mixture**.

- ***MATLAB Code for `quantile` Application to S&P500 2008 Log-Returns:***

```
function quantilespc2008
% Get Quantiles for Log-Returns Density of 2008 S&P500;
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
load -ASCII S08.mat; %
NS = length(S08);
fprintf('\nquantilespc2008 Output for Log-Returns, 1/13/2009:');
LR = log(S08(2:NS))-log(S08(1:NS-1)); % Note: Vector Log Difference!
fprintf('\n\nmean(LR) = %8.6f; std(LR) = %7.5f; NS = %4i;'...
    ,mean(LR),std(LR),NS);
fprintf('\nskew(LR) = %7.5f > 0; kurtosis(LR) = %5.3f > 3;'...
    ,skewness(LR),kurtosis(LR));
P = [0,0.001,0.01,0.025,0.05,0.25,0.50,0.75,1.00];
Q = quantile(LR,P);
fprintf('\n\nRisk Probabilities P = \n[%5.3f,%5.3f,%5.3f,%5.3f];'...
    ,P(1,2:5));
fprintf('\n\nRisk Log-Return Quantiles Q = ');
fprintf('\n[%5.3f,%5.3f,%5.3f,%5.3f];',Q(1,2:5));
fprintf('\n\nQuartile Probabilities P = ');
fprintf('\n[%5.3f,%5.3f,%5.3f,%5.3f,%5.3f];',P(1,[1,6:9]));
fprintf('\n\nQuartiles Log-Return Q = ');
fprintf('\n[%5.3f,%5.3f,%5.3f,%5.3f,%5.3f];',Q(1,[1,6:9]));
```

- *MATLAB `quantile` Application to S&P500 Log-Returns Output:*

```
quantilespc2008 Output for Log-Returns,1/13/2009:

mean(LR)=0.001921; std(LR)=0.02583; NS= 254;
skew(LR)=0.03657>0; kurtosis(LR)=6.680>3;

Risk Probabilities
 P=[0.001,0.010,0.025,0.050];

Risk Log-Return Quantiles
 Q=[-0.110,-0.067,-0.050,-0.039];

Quartile Probabilities
 P=[0.000,0.250,0.500,0.750,1.000];

Quartiles Log-Return
 Q=[-0.110,-0.008,0.000,0.013,0.095];
 >>
```

- ***Quantile versus Quantile (Q-Q) Plots:***

  A **quantile-quantile plot** or **Q-Q plot** is used to **calculate and compare sample quantiles of to distributions against each other**, e.g., the quantiles of a sample distribution against the same quantiles of a theoretical distribution. The closeness of the plot to a linear plot $(Y = X)$ is a qualitative measure of how close the distributions are to each other.

  In the `Statistics Toolbox` the **Q-Q** plot function is called `qqplot`. As with most `MATLAB` function its arguments comes in several forms,

  ```
  qqplot(Xvector);
  ```
  (3.14)

  is used when you have a sample vector **Xvector** that you want to compare its quantiles with a normal distribution and you want `qqplot` to simulate the theoretical normal. However, if you want to use your own pair of samples, `(Xvector,Yvector)`, then use

  ```
  qqplot(Xvector,Yvector);
  ```
  (3.15)

If you also want to use your own quantiles, `Qvector`, then use

```
qqplot(Xvector,Yvector,Qvector);
```
(3.16)

but the second version may be more appropriate for a lot of applications.

The **first example** uses this second form to test the sample distribution of our **2008 S&P500 Index log-returns LR**, a row-vector of length `NLR` against the a **simulated normal distribution** having the same mean (`meanLR`) and standard deviation (`stdLR`),

```
Xnorm,normrnd(meanLR,stdLR,1,NLR);
```
(3.17)

outputting a row-vector of length,

```
NLR=length(LR);
```
(3.18)

- ***Q-Q Plot Comparing 2008 Log-Returns Against Corresponding Normal Distribution:***

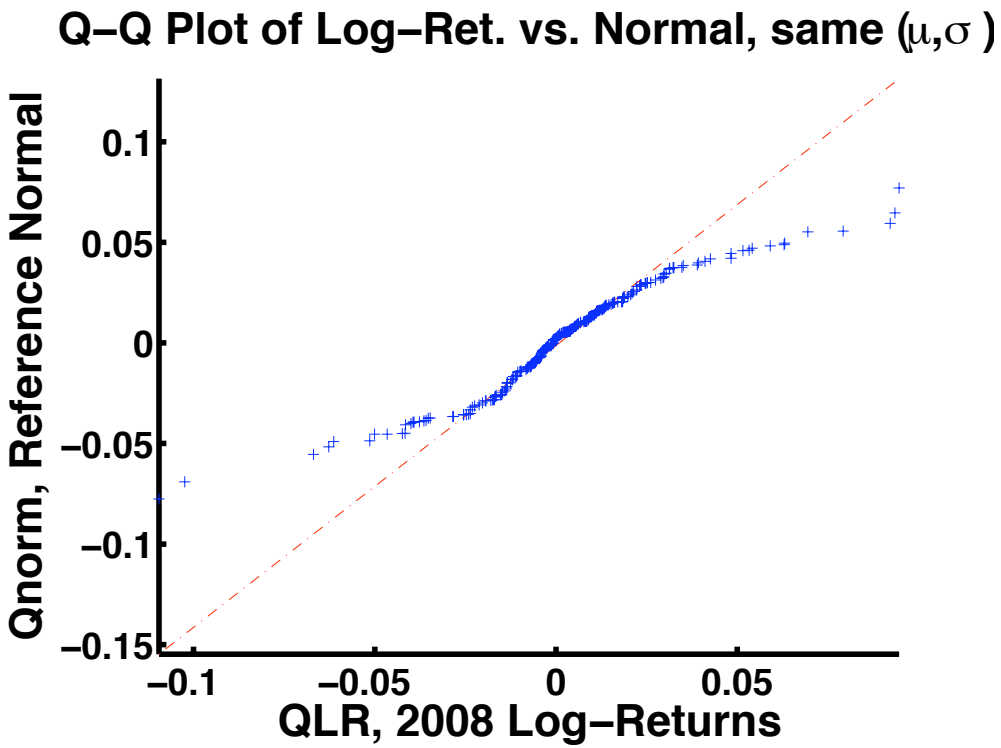


Figure 3.6: Q-Q Plot of **S&P500 Index log-returns** for the whole year 2008 compared to a simulated normal distribution with the **same** mean (μ) and standard deviation or volatility (σ). ***Look at those really fat tails!***

- ***MATLAB Code for `qqplot` Comparison of 2008 S&P500 Log-Returns to Normal Distribution with Same Basic Statistics:***

```
function qqplotspc2008
% Get Q-Q Plots for Log-Returns Density of 2008 S&P500;
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
load -ASCII S08.mat;
NS = length(S08);
fprintf('\nqqplotspc2008 Output for Log-Returns, 1/13/2009:');
LR = log(S08(2:NS))-log(S08(1:NS-1)); % Note: Vector Log Difference!
meanLR = mean(LR); stdLR = std(LR); NLR = length(LR);
fprintf('\nmean(LR) = %8.6f; std(LR) = %7.5f; NLR = %4i;'...
    ,meanLR,stdLR,NLR);
fprintf('\nskew(LR) = %7.5f > 0; kurtosis(LR) = %5.3f > 3;'...
    ,skewness(LR),kurtosis(LR));
Xnorm = normrnd(meanLR,stdLR,1,NLR);
figure(2);
qqplot(Xnorm,LR); axis tight;
title('Q-Q Plot of Log-Ret. vs. Normal, same (\mu,\sigma )'...;
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('QLR, 2008 Log-Returns','Fontsize',24,'FontWeight','Bold');
ylabel('Qnorm, Reference Normal','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',20,'FontWeight','Bold','LineWidth',3);
```

- ***Second `qqplot` Example Comparing 2008 Log-Returns With a Simulated Jump-Diffusion:***

  Since the closeness of the 2008 S&P500 log-returns (**`NLR-vector LR`**) to the normal distribution simulation, with same overall mean **`meanLR = mean(LR)`** and standard deviation **`stdLR= std(LR)`** is poor for both positive and negative log-return tails, a simplified jump-diffusion simulation will be tested as the reference distribution,

  ```
  XJD = Xnorm+Xiid.*Xpois;
  ```
  (3.19)

  Be sure to note that the **`.*`** element by element multiplication is used in the compound Poisson product of **`Xiid`** and **`Xpois`**.

  The discretized continuous diffusion component of the jump-diffusion will be the same as for the prior reference, simulated normal **`NLR`**-vector,

  ```
  Xnorm = normrnd(meanLR,stdLR,1,NLR);
  ```
  (3.20)

For the Poisson counting process **`Xpois`**, it will be assumed that the Poisson jump parameter **`Lambda=lambda*Dt`** to be sufficiently small so the **zero-one law, two-state Bernoulli process** can be used, since the multi-jump case would be too complicated to use at this stage of the course. However, the **`Statistic Toolbox`** does not include a Bernoulli RNG, but the **built-in Binomial RNG `binornd`** is the same as what would be the Bernoulli RNG if the order **`n=1`** and the one-state probability **`p=Lambda`** are used. The **`binomial disrtibution`** is given by

$$\begin{aligned} \mathrm{Bino}(k;n,p) \equiv \mathrm{Prob}[Z^{(\mathrm{bin,n})}=k] &\equiv \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k} \\ &= \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}, \end{aligned} \tag{3.21}$$

for $k = 0:n$.

So, in the Bernoulli case with parameters (`n=1,p=Lambda`)

$$\mathrm{Ber}(k;\Lambda) \equiv \texttt{Bino}(k;1,\Lambda) = \frac{1}{k!(1-k)!}\Lambda^k(1-\Lambda)^{1-k}$$
$$= \left\{ \begin{array}{ll} 1-\Lambda, & k=0 \\ \Lambda, & k=1 \end{array} \right\}, \tag{3.22}$$

which is the 0-1 Poisson jump law. Hence, the simulated Poisson/Bernoulli approximate component has the form

```
Xpois = binornd(1,Lambda,1,NLR);
```
(3.23)

subject to the condition that `Lambda=lambda*Dt` be small, which is not the same as requiring the time-step `Dt` be small.

Since we do not have enough data to have any idea about the distribution of the amplitude of the compound Poisson process, we will take it as a uniform distribution on the same range as the data, i.e., `(a,b)=(minLR,maxLR)`, so

```
Xiid = unifrnd(minLR,maxLR,1,NLR);
```
(3.24)

- ***Q-Q Plot Comparing 2008 Log-Returns Against a Simple Jump-Diffusion Distribution:***

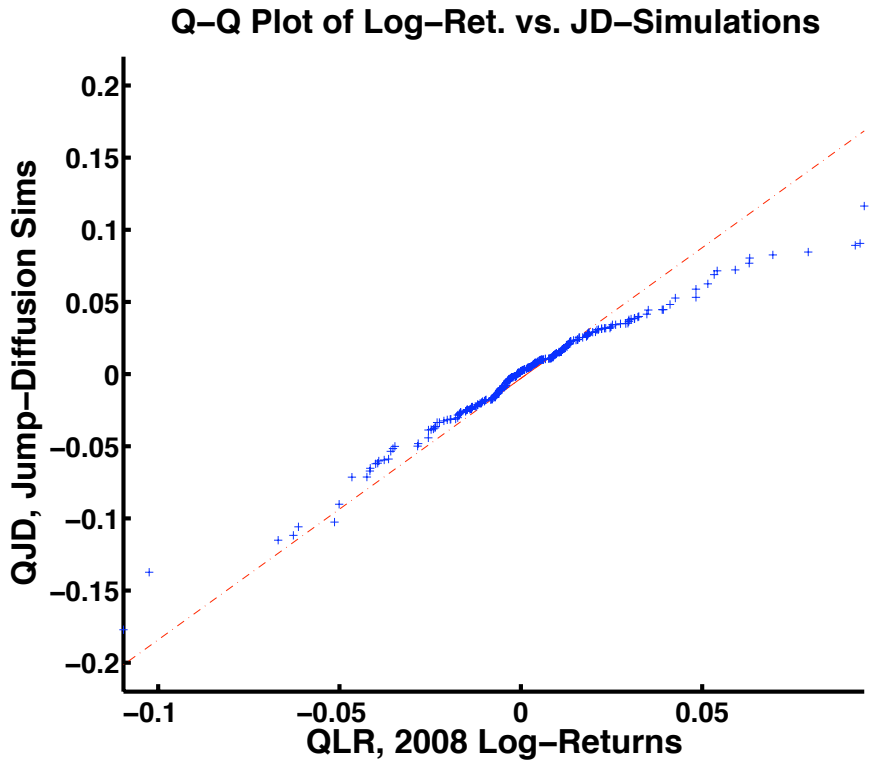


Figure 3.7: Q-Q Plot of **S&P500 Index log-returns** for the whole year 2008 compared to a simulated jump-diffusion distribution with the **same** mean (μ), volatility (σ) and uniform jump-amplitude range (a, b). ***Closer on both tails, considering the different*** `Qnorm` ***and*** `QJD` ***scales. A better parameter fit for QJD may help?***

- ***Q-Q Plot Comparing 2008 Log-Returns Against a Normal Distribution at Same Y-Scale:***

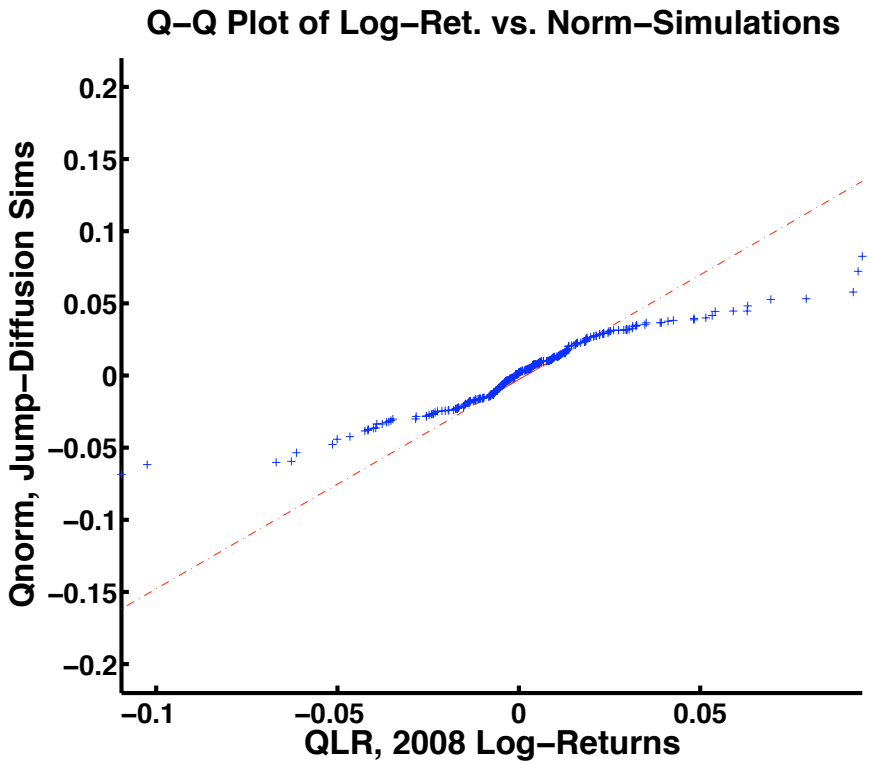


Figure 3.8: Q-Q Plot of **S&P500 Index log-returns** against the corresponding normal simulation for the whole year 2008 at the same scale as the simulated jump-diffusion distribution for comparison.

- ***MATLAB Code for `qqplot` Comparison of 2008 S&P500 Log-Returns to Jump-Diffusion Simulation with Some Similar Statistics, But Not All (`XJD` is called `Xsims` in the code):***

```
function qqplotsims2008w10
% Get Q-Q Plots for Log-Returns Density of 2008 S&P500 & JD-Sims;
%   Revised Winter 2010
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
load -ASCII S08.mat;
NS = length(S08);
fprintf('\nqqplotsims2008w10 Log-Returns Output  (%s):\n',datestr(now));
LR = log(S08(2:NS))-log(S08(1:NS-1)); % Note: Vector Log Difference!
meanLR = mean(LR); stdLR = std(LR); NLR = length(LR);
fprintf('\nmean(LR) = %8.6f; std(LR) = %7.5f; NLR = %4i;'...
    ,meanLR,stdLR,NLR);
minLR = min(LR); maxLR = max(LR);
fprintf('\nminLR = %7.5f; maxLR = %5.3f;',minLR,maxLR);
fprintf('\nskew(LR) = %7.5f > 0; kurtosis(LR) = %5.3f > 3;'...
    ,skewness(LR),kurtosis(LR));
Xnorm = normrnd(meanLR,stdLR,1,NLR);
fprintf('\nXnorm: mean=%8.6f; std=%7.5f;',mean(Xnorm),std(Xnorm));
fprintf('\nXnorm: min=%7.5f; max=%7.5f;',min(Xnorm),max(Xnorm));
Dt = 1/NLR; lambda = 37; Lambda = lambda*Dt;
```

```
fprintf('\nDt = %7.5f; lambda = %5.3f; Lambda = %5.3f;'...
    ,Dt,lambda,Lambda);
Xpois = binornd(1,Lambda,1,NLR); % small Lambda Bernoulli approximation;
Xiid = unifrnd(minLR,maxLR,1,NLR);
XJD = Xnorm + Xiid.*Xpois;
%
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
%
figure(3); nfig = 3;
qqplot(LR,XJD);
axis([minLR maxLR -0.22 0.22])% axis tight;
title('Q-Q Plot of Log-Ret. vs. JD-Simulations'...;
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('QLR, 2008 Log-Returns','Fontsize',24,'FontWeight','Bold');
ylabel('QJD, Jump-Diffusion Sims','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',20,'FontWeight','Bold','LineWidth',3);
set(gcf,'Color','White','Position' ...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]); %[l,b,w,h]
%
figure(2); nfig = 2;
qqplot(LR,Xnorm);
axis([minLR maxLR -0.22 0.22])% axis tight;
title('Q-Q Plot of Log-Ret. vs. Norm-Simulations'...;
```

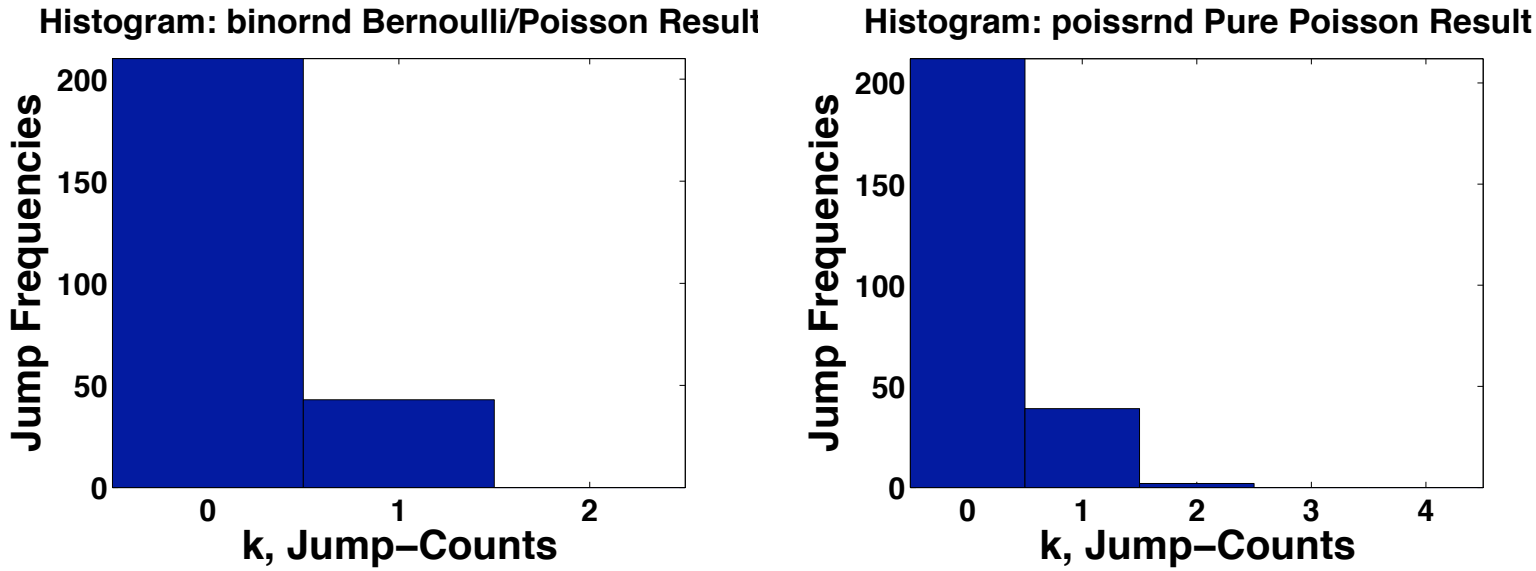
```
    ,'Fontsize',24,'FontWeight','Bold');
xlabel('QLR, 2008 Log-Returns','Fontsize',24,'FontWeight','Bold');
ylabel('Qnorm, Jump-Diffusion Sims','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',20,'FontWeight','Bold','LineWidth',3);
set(gcf,'Color','White','Position' ...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]); %[l,b,w,h]
fprintf('\n');
%
======================OUTPUT=========================

qqplotsims2008w10 Output for Log-Returns (14-Jan-2010 22:50:05):

mean(LR) = 0.001921; std(LR) = 0.02583; NLR =  253;
minLR = -0.10957; maxLR = 0.095;
skew(LR) = 0.03657 > 0; kurtosis(LR) = 6.680 > 3;
Xnorm: mean=0.002821; std=0.02448;
Xnorm: min=-0.06449; max=0.05894;
Dt = 0.00395; lambda = 37.000; Lambda = 0.146;
>>
```

- ***Postscript — Comments on `qqplot` Result and Limited Improvement of Simple Jump-Diffusion Simulations:***
  1. The `XJD` simulation does better than than `Xnorm` alone in matching the `LR` quartiles, except for the most extremes jumps.
  2. Note that for the uniform jump-amplitude distribution there are a **lot of small jumps around the center** that are indistinguishable from the continuous, normal fluctuations in the log-returns. Thus, a large jump rate does not mean a large number of crash and rally type jumps.
  3. However, making `lambda` any larger, than the `lambda=37` used so `Lambda=0.146<1`, would seriously violate the Bernoulli (0-1) approximation and make 2 or more jumps more likely. Of course, including data between closing helps, if it is readily available and is for users with special access.
  4. If there are more then one jump, then compound Poisson needs the sum form. See later, perhaps.

- ***Comparison Test of binornd and poissrnd:***



(a) Poisson-Bernoulli `binornd` Test. (b) General Poisson tbbtpoissrnd Test.

Figure 3.9: Histograms for comparing the 0-1 jump binorad RNG and the general Poisson jump count `poissrnd`. The values uses the Poisson parameter `Lambda=0.01462` and the 2008 log-return sample count `NLG=253`. Notice the different distributions for each RNG, although most jump counts are either 0 or 1, the `poissrnd` generate a few double jumps, but this is one sample and rare extra counts are possible.

- ***MATLAB Code for Comparing Bernoulli (`bino` with `n=1`) and Full `poissrnd`:***

```
function PoissBinoRndTestS08W10
% Get Q-Q Plots for Log-Returns Density of 2008 S&P500 & JD-Sims;
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
load -ASCII S08.mat;
NS = length(S08);
fprintf('\nPoissBinoRndTests Output for Log-Returns, 1/14/2009:');
LR = log(S08(2:NS))-log(S08(1:NS-1)); % Note: Vector Log Difference!
meanLR = mean(LR); stdLR = std(LR); NLR = length(LR);
fprintf('\nmean(LR) = %8.6f; std(LR) = %7.5f; NLR = %4i;'...
    ,meanLR,stdLR,NLR);
fprintf('\nskew(LR) = %7.5f > 0; kurtosis(LR) = %5.3f > 3;'...
    ,skewness(LR),kurtosis(LR));
Dt = 1/NLR; lambda = 37; Lambda = lambda*Dt;
fprintf('\nDt = %8.6f; lambda = %4.1f; Lambda = %7.4f;'...
    ,Dt,lambda,Lambda);
minLR = min(LR); maxLR = max(LR);
fprintf('\nminLR = %8.5f; maxLR = +%7.5f;',minLR,maxLR);
Xbino = binornd(1,Lambda,1,NLR); % small Lambda Bernoulli approximation;
Xpoiss = poissrnd(Lambda,1,NLR); % note renaming former from Xpois
fprintf('\nXbino(1,1:12)=[%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i];'...
```

```
    ,Xbino(1,1:12));
fprintf('\nXpoiss(1,1:12)=[%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i,%1i];'...
    ,Xpoiss(1,1:12));
figure(4);
hist(Xbino,0:2);
axis tight;
title('Histogram: binornd Bernoulli/Poisson Result'...
    ,'Fontsize',20,'FontWeight','Bold');
xlabel('k, Jump-Counts','Fontsize',24,'FontWeight','Bold');
ylabel('Jump Frequencies','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',18,'FontWeight','Bold');
figure(5);
hist(Xpoiss,0:4);
axis tight;
title('Histogram: poissrnd Pure Poisson Result'...
    ,'Fontsize',20,'FontWeight','Bold');
xlabel('k, Jump-Counts','Fontsize',24,'FontWeight','Bold');
ylabel('Jump Frequencies','Fontsize',24,'FontWeight','Bold');
set(gca,'Fontsize',18,'FontWeight','Bold');
fprintf('\n');
```

Technique Alert: Note too that there is a change in the **hist** function, from previous uses with **Nbins=50** to a **binvector= 1:K**, where **K** is the likely maximum jump **count+J** where **J** is an extra count to verify a correctly identified maximum), centered on integer counts of **binvector**, otherwise a confusing graph is produced, with the integer appears anywhere on its bin-interval, when only integer value matter on the **x**-axis.[a]

```
====================OUTPUT=======================
PoissBinoRndTests Output for Log-Returns, 1/14/2009:
mean(LR) = 0.001921; std(LR) = 0.02583; NLR =  253;
skew(LR) = 0.03657 > 0; kurtosis(LR) = 6.680 > 3;
Dt = 0.003953; lambda = 37.0; Lambda =  0.1462;
minLR = -0.10957; maxLR = +0.09470;
Xbino(1,1:12)  = [0,0,0,0,0,0,0,0,1,0,0,0];
Xpoiss(1,1:12) = [1,0,0,0,0,0,0,0,0,0,0,0];
```

[a] Is this Financial Engineering Mathematics? {⌣}

3.4. Value at Risk (VaR)[a]*:*

Value at Risk (VaR) formulas are used or misused for statistical ***estimate of risk*** in order to estimate the ***amount of capital held in reserve (liquidity)*** in case there are ***large losses (drawdowns or shortfalls)***. There are many versions of **VaR** and **VaR has been heavily criticized and blamed** for recent financial and general economic problems. There are arguments about whether the problems stem from the **VaR** model itself or the underlying statistical probability model that is used to quantitatively evaluate the **VaR** formulas, especially **assuming the normal probability model in very nonnormal environments**. An alternative to the modeling approach is the use of historical simulations.

---

[a]See Hull (2006) Chapt. 18, Carmona (2004) p. 25ff., and Wilmott (2000) for other background information.

- ***VaR — One Formulation in Words:***

  Suppose we have a financial portfolio, say of stocks, bonds, derivatives and other financial instruments (in general, this could include all the firm's investments), with **current total book value** of V_i at time t_i and for the firm and this portfolio we use a ***level of confidence*** c, a probability of confidence. This probability is usually expressed as its complement which is the **level of risk (loss)** $\alpha = 1 - c$. Say, an ***alpha level*** is 0.01 or alternately in per cent is 1%, so the confidence interval is $100c\% = 100(1-\alpha)\% = 99\%$, The firm expects no less than its specified level of risk, usually specified by regulatory agencies. The particular insurance or protection question is what amount of **reserve capital** RC_N is required to guarantee that the **portfolio's book value** V_{i+N} will not be in the red (has net loss) at the next time t_{i+1} the book is reevaluated with **time horizon** $T \equiv t_{i+N} - t_i$ (often measured in days N) with at least α probability. The RC_N is usually some multiple of V_i.

- ***VaR — One Formulation in Mathematics:***

  Let $\Delta V_N = V_{i+N} - V_i$ be the potential loss (a gain has no risk problem here) value of the portfolio in N days, then the reserve capital RC_N is determined by

$$\mathrm{Prob}[V_i + \Delta V_N + \mathrm{RC}_N < 0] = \alpha. \tag{3.25}$$

  One way the **Value of Risk** can be defined as the **current portfolio value plus the reserved capital**, i.e.,

$$\mathrm{VaR}_N \equiv V_i + \mathrm{RC}_N. \tag{3.26}$$

  Thus, $\mathrm{VaR}_N = \mathrm{VaR}_N(\alpha)$ such that

$$\mathrm{Prob}[\Delta V_N < -\mathrm{VaR}_N(\alpha)] = \alpha \tag{3.27}$$

  and as a confidence interval using **complementary probability**,

$$\mathrm{Prob}[\Delta V_N \geq -\mathrm{VaR}_N(\alpha)] = (1-\alpha), \tag{3.28}$$

  which means a $100(1-\alpha)\%$ level of confidence that the portfolio return will be greater than or equal to $-\mathrm{VaR}_N(\alpha)$, the cutoff separation from the tail of greater losses.

- ***Relative VaR and Log-Reformulation:***[a]

  Another formulation compatible with **relative returns** or **log-returns** uses the relative ***VaR*** with respect to current portfolio value V_i,

$$\mathbf{RVaR}_N(\alpha) = \frac{\mathbf{VaR}_N(\alpha)}{V_i}, \tag{3.29}$$

  provided $V_i > 0$, so

$$\mathbf{Prob}[\Delta V_N/V_i \geq -\mathbf{RVaR}_N(\alpha)] = 1 - \alpha. \tag{3.30}$$

  Since we showed previously that the log-return as asymptotic to the relative return for small values , we have the $\mathbf{Log-VaR}$,

$$\mathbf{Prob}[\log(V_{i+N}/V_i) \geq -\mathbf{LVaR}_N(\alpha)] = 1 - \alpha. \tag{3.31}$$

  This formula could very well be taken as a practical definition of $\mathbf{RVaR}_N(\alpha)$ when calculating with available underlying log-return (**LR**) distribution data. Given the underlying **LR** distribution, $\mathbf{LVaR}_N(\alpha)$ is called the **negative α–quantile of the log-return distribution**, according to Carmona (2004).

[a] In some references the kind of **VaR** may not be clearly defined, so needs to be identified by usage.

- ***Log-VaR Calculations with Log-Return distributions:***

  Recall the log-return $LR_N = \Delta_N \log(V_i) \equiv \log(V_{i+N}/V_i)$, so then the probability complement form for **LVaR** is

$$\mathbf{Prob}[\mathbf{LR}_N < -\mathbf{LVaR}_N(\alpha)] = \alpha \tag{3.32}$$

  and assuming there is a nicely behaved, invertible log-return distribution $F_{\mathbf{LR}}(x)$ for the application and the risk level α is reasonable,

$$F_{\mathbf{LR}}(-\mathbf{LVaR}_N(\alpha)) = \alpha. \tag{3.33}$$

  Inverting and dropping the index i yields in general the **Log-VaR**,

$$\mathbf{LVaR}_N(\alpha) = -F^{-1}_{\mathbf{LR}}(\alpha). \tag{3.34}$$

  If that $F_{\mathbf{LR}}$ is a normal distribution,

$$\mathbf{LVaR}^{(n)}_N(\alpha) = -\left(F^{(n)}_{\mathbf{LR}}\right)^{-1}(\alpha). \tag{3.35}$$

Theorem 3.1. LVaR $\sqrt{N}$ Factor from Daily Basis to N-days:

Let the daily log-returns be IID, distributed normally, with zero-mean[a] *and* $\boldsymbol{\sigma^2 k\Delta t}$*-variance, where* $\boldsymbol{\Delta t}$ *is one trading day in years and* $\boldsymbol{k}$ *is an integer, then for the* ***log-VaR*** *at risk level* $\boldsymbol{\alpha}$ *satisfies*[b]

$$\mathbf{LVaR}_N(\alpha) = \sqrt{N} \cdot \mathbf{LVaR}_1(\alpha). \tag{3.36}$$

However, using real data, such as the **S&P500 (SP)**, might be better, so then the **`Statistics Toolbox quantile`** function can be used instead,

$$\mathbf{QVaR}_N(\alpha) \overset{\text{eg}}{=} \mathbf{LVaR}_N^{(\mathbf{sp})}(\alpha) = -\mathtt{quantile}\left(\mathbf{LR}_N^{(\mathbf{sp})}, \alpha\right). \tag{3.37}$$

---

[a] Often, the log-return mean of the data is extremely small and often the mean is set to zero for a convenient fast trick, else over very long periods a non-zero mean estimate would lead to a monotonic bias with monotonic growth of the expected price of the asset. Another assumption that is often made is that the log-returns are IID independent of the number of trading days.

[b] Proof is left to the student. **Caution: This does not follow from prior discrete models.**

- ***LVaR Code Comparing Normal and Real Data Versions:***

```
function LVaR2008
% Get LogVaR for Normal & Quantile Data Log-Returns of 2008 S&P500;
clc
% Get S&P500 ^GSPC Adjusted Closings for 2008 From Yahoo Finance;
load -ASCII S08.mat; % Change GSPC2008adjC.txt name for load function.
NS = length(S08);
fprintf('\nLVaR2008.m Output for S&P 2008 Log-Returns, 1/15/2009:');
fprintf('\nHere LVaRnorm = -NormInv; assuming normally distributed;');
LR = log(S08(2:NS))-log(S08(1:NS-1)); % Note: Vector Log Difference!
meanLR = mean(LR); stdLR = std(LR); NLR = length(LR);
fprintf('\nmean(LR) = %8.6f; std(LR) = %7.5f; NLR = %4i;'...
    ,meanLR,stdLR,NLR);
fprintf('\nskew(LR) = %7.5f > 0; kurtosis(LR) = %5.3f > 3;'...
    ,skewness(LR),kurtosis(LR));alpha = [0.001,0.01,0.02,0.05];
LVaRquant = -quantile(LR,alpha);
LVaRnorm = -norminv(alpha,meanLR,stdLR); % Same Mean and Std;
fprintf('\n\n        Risk level alpha = [%6.4f,%6.4f,%6.4f,%6.4f];'...
    ,alpha);
fprintf('\nConfidence level 1-alpha = [%6.4f,%6.4f,%6.4f,%6.4f];'...
    ,1-alpha);
fprintf('\n\nLog-Return Quantile LVaR = ');
fprintf('[%6.4f,%6.4f,%6.4f,%6.4f];',LVaRquant);
fprintf('\n\n  Log-Return Normal LVaR = ');
fprintf('[%6.4f,%6.4f,%6.4f,%6.4f];',LVaRnorm);
```

```
fprintf('\n\n  Normal LVaR RelDiff= ');
fprintf('[%6.2f%%,%6.2f%%,%6.2f%%,%6.2f%%];'...
    ,(LVaRnorm./LVaRquant-1)*100);
fprintf('\nwhere Normal RelDiff = (LVaRnorm./LVaRquant-1)*100;');
fprintf('\n');

======================OUTPUT=========================

LVaR2008.m Output for S&P 2008 Log-Returns, 1/15/2009:
Here LVaRnorm = -NormInv; assuming normally distributed;
mean(LR) = 0.001921; std(LR) = 0.02583; NLR =  253;
skew(LR) = 0.03657 > 0; kurtosis(LR) = 6.680 > 3;

        Risk level alpha = [0.0010,0.0100,0.0200,0.0500];
Confidence level 1-alpha = [0.9990,0.9900,0.9800,0.9500];
Log-Return Quantile LVaR = [0.1096,0.0668,0.0558,0.0394];
  Log-Return Normal LVaR = [0.0779,0.0582,0.0511,0.0406];
  Normal LVaR RelDiff= [-28.91%,-12.92%, -8.30%,  2.89%];
where Normal RelDiff = (LVaRnorm./LVaRquant-1)*100;
>>
```

Note: The difference of the normal `LVaR` from the `quantile` `LVaR=QVaR` grows larger the bigger are the losses.

3.5 Shortfall Statistics: Other Risk Measures Beyond VaR: [a]

- ***Basel Accords on Financial Institutions:***

  These are a series of agreements about the responsibilities of international financial institutions to control their exposures to risk (see Hull's (2006) so-called ***"Traders' Bible"*** Chapter 18, for a short sidebar summary; also Wikipedia, but since it is usually anonymous, check further). *"The first pillar (namely **Basel I**) deals with maintenance of regulatory capital calculated for three major components of risk that a bank faces: **credit risk, operational risk and market risk**. . . . **Basel II** uses a "three pillars" concept **(1) minimum capital requirements (addressing risk), (2) supervisory review and (3) market discipline . . .** "* (Wikipedia, time ordered).

[a] As previously, this lecture will be a hybrid of Carmona's (2004) and Hanson's (2000-09) financial data analysis, but with the former more clarified.

- ***Profit & Loss Or Loss & Profit Distributions:***

  Since the **portfolio value returns ΔV_j** for $j = i : i+N$ can range from positive to negative, their distribution is both a profit and loss (***P&L***) distribution. Since our main risk is the losses, let the **random variable X represent the data $-\Delta V_N$** to emphasize the losses and let $F_X(x)$ be the distribution of X, the **"Loss" distribution**. Note that if the RV Y represents the **profit $+\Delta V_N$**, then $Y = -X$ with **"Profit" distribution $F_Y(y)$**, so by the **law for changing densities**, say on (a, b) under changes of variables, conserving probability

$$1 = \int_a^b f_Y(y)dy = + \int_{-b}^{-a} f_X(x)dx, \tag{3.38}$$

  which is equivalent to using the absolute value of the **Jacobian of the transformation**, i.e., $f_X(x) = f_Y(y)|dy/dx|$ and similarly for the distribution at least in the continuous case when we can assume the densities exist.

- ***Shortfall Distribution:***

  Again let α be the level of risk corresonding to the given **Value at Risk** $\mathbf{VaR}_N(\alpha)$, then the **Shortfall distribution (SF)**, for the shortfall or loss $X = -\Delta V_N$, is defined as a **conditional loss distibution**,

$$F_X^{(\mathrm{sf})}(x;\alpha) = \mathbf{Prob}[X \le x \mid X > \mathbf{VaR}_N(\alpha)]. \tag{3.39}$$

  This can be used to get an estimate of the actual loss. The form of the conditioning is explained by the original definition of $\mathbf{VaR}_N(\alpha)$. Let the **loss** data $X = -\Delta V_N$ be chosen as an **RV**, so

$$\begin{aligned} \alpha &\stackrel{\mathrm{def}}{=} \mathbf{Prob}[\Delta V_N < -\mathbf{VaR}_N(\alpha)] \\ &= \mathbf{Prob}[-\Delta V_N > \mathbf{VaR}_N(\alpha)] \\ &= \mathbf{Prob}[X > \mathbf{VaR}_N(\alpha)]. \end{aligned} \tag{3.40}$$

  However, the shortfall distribution, since conditional, is related to a conditional truncation of the "**Loss**" distribution $F_X(x)$ with renormalization to conserve of probability.

In terms of the differential distribution (called measure in abstract, but $dF_X(x) = f_X(x)dx$ in usual practice),

$$dF_X^{(\mathrm{sf})}(x;\alpha) = \left\{ \begin{array}{ll} \frac{dF_x(x \mid x > \mathrm{VaR}_N(\alpha))}{\mathrm{Prob}[X > \mathrm{VaR}_N(\alpha)]}, & x > \mathrm{VaR}_N(\alpha) \\ 0, & else \end{array} \right\}, \quad (3.41)$$

so, probability is conserved,

$$\begin{aligned} \int_{-\infty}^{+\infty} dF_X^{(\mathrm{sf})}(x;\alpha) &= \int_{\mathrm{VaR}_N(\alpha)}^{+\infty} dF_X^{(\mathrm{sf})}(x;\alpha) \\ &= \tfrac{1}{\alpha} \int_{\mathrm{VaR}_N(\alpha)}^{+\infty} dF_X(x) \qquad (3.42) \\ &\equiv \tfrac{1}{\alpha} \mathrm{Prob}[X > \mathrm{Var}_\alpha] = 1. \end{aligned}$$

Hence, the Expected Shortfall $\mathrm{ES}(\alpha)$ is

$$\begin{aligned} \mathrm{ES}(\alpha) &\equiv \mathrm{E}[X \,|\, X > \mathrm{VaR}_N(\alpha)] \\ &= \int_{-\infty}^{+\infty} x dF_X^{(\mathrm{sf})}(x;\alpha) \qquad (3.43) \\ &= \tfrac{1}{\alpha} \int_{\mathrm{VaR}_N(\alpha)}^{+\infty} x dF_X(x), \end{aligned}$$

completing what should be a more coherent explanation of Carmona's (p. 27) less than credible presentation.

- ***Choice of Loss Distribution $F_X(x)$:***

  The difficulty with the shortfall distribution theory is the choice of the loss distribution $F_X(x)$ such that the distribution is consistent with appropriate financial models.

  * **"Normal" Choice:** If the distribution is **normally distributed** with mean μ and variance σ^2 then the estimated expected shortfall can be calculated by **MATLAB**, for instance, since

$$
\begin{aligned}
ES^{(n)}(\alpha) &= \frac{1}{\alpha}\int_{\mathrm{VaR}_N(\alpha)}^{+\infty} x dF_X^{(n)}(x;\mu,\sigma^2) \\
&= \frac{1}{\alpha\sqrt{2\pi}}\int_{\frac{\mathrm{VaR}_N(\alpha)-\mu}{2\sigma^2}}^{+\infty}(\mu+\sigma z)\exp(-z^2/2) \\
&= \frac{\mu}{\alpha}F_X^{(n)}\left(-\frac{\mathrm{VaR}_N(\alpha)-\mu}{\sigma};0,1\right) \\
&\quad + \frac{\sigma}{\alpha\sqrt{2\pi}}\exp\left(-\frac{(\mathrm{VaR}_N(\alpha)-\mu)^2}{2\sigma^2}\right).
\end{aligned}
\tag{3.44}
$$

For the appropriate financial model with a normal distribution often means the **log-diffusion (LD)** which has the form **(take for given)** that scales for N-steps (refer to Theorem 3.1),

$$\log(V_{i+N}/V_i) = \log(1 - X/V_i) = \mu_{ld} N\Delta t + \sigma\sqrt{N\Delta t}\, Z_i, \quad (3.45)$$

where $\mu_{ld} \equiv \mu - \sigma^2/2$ is the log-diffusion mean that here follows from $Z_i^2 \simeq \mathrm{E}[Z_i] = 1$ and Z_i is zero-mean and unit-variance. **(Note that (3.45) does not follow precisely from the discrete multiplicative form.)** The dilemma with Eq. (**3.45**) and the expected shortall estimate is that the interest is not in small losses X or relative ones $\mathbf{Xrel} \equiv X/V_i$, but in large ones, otherwise use $\log(1 - X/V_i) \simeq -X/V_i$, which is then normally distributed.

However, since the data has non-normal fat tails leads to questions of validity for the normal distribution anyway, we might just as well use $\mathbf{Xrel}$ with $\mathbf{RVaR}_N(\alpha)$ or the log-loss $\mathbf{Xlog} \equiv \log(V_{i+N}/V_i)$ with $\mathbf{LVaR}_N(\alpha)$ in the formulas on pages 50-52.

*** NonNormal Data Quantile Choice:** Recall the 2008 S&P500 Index data displayed in Figure 3.4 on page 13 to emphasize the two bumps in the tails with the histogram and the ks-density, **illustrating very nonnormal behavior**. Although another choice is the kernel-smoothed ks-density, here we choose the use of quantiles and the Quantile $\mathbf{VaR}$ or $\mathbf{QVaR}_N(\alpha)$ in Eq. (3.37) used in the `LVaR200.m` code on page 46. Given reduced market log-return N-data $\mathbf{LR}$ we can calculate $\mathbf{QVaR}_N(\alpha)$ for the given α using (3.37) replacing the $\mathbf{VaR}_N(\alpha)$ in the **Expected Shortfall Equation** (3.43).

Next, the log-return CDF $F_X(x)$ is approximated by the quantile distribution,

```
Q=quantile(LR,P);
```
(3.46)

where `1×(m+1)` quantile interval probabilities `P=0:DP:1;` with `m+1` constant steps `DP=1/m;`, assumed to be sufficiently small for reasonable numerical accuracy.

Moments or expectations of a function $g(x)$ can be numerically approximated by selecting an x-value in each quantile bin with probability **DP**, i.e.,

$$\begin{aligned}\int_{-\infty}^{-\infty} g(x)dF_X(x) &\simeq \sum_{i=1}^{m} g(Q_{i^*})\int_{Q_i}^{Q_{i+1}} dF_X(x)\\ &\simeq \mathtt{DP}*\sum_{i=1}^{m} g(Q_{i^*}).\end{aligned} \tag{3.47}$$

For instance, one estimate of the g-approximation is that of the mid-point, $Q_{i^*}=Q_{i+0.5}\simeq(Q_i+Q_{i+1})/2$.

Thus, the **quantile approximation of the expected shortfall** is

$$QES^{(n)}(\alpha)\simeq \frac{\mathtt{DP}}{\alpha}\sum_{i=1}^{m} g^{(es)}((Q_i+Q_{i+1})/2), \tag{3.48}$$

where

$$g^{(es)}(x)\equiv\begin{Bmatrix} x, & \mathrm{QVaR}_N(\alpha)\le x\le Q_{m+1}\\ 0, & else\end{Bmatrix}. \tag{3.49}$$

One can also check probability conservation using $g^{(es)}(1)$.

** Reminder: Lecture 3 Homework Posted in Chalk Assignments,*
due by Lecture 4 in Chalk Assignments!

*** Summary of Lecture 3:**

1. **Histograms**
2. **Kernel Smoothing**
3. **Quantiles and QQplots**
4. **VaR(alpha) variations: RVaR, LVaR, QVaR=QLVaR**
5. **Expected Shortfall Estimates ES(alpha)**