Market data *vs* Gaussian and Power Tails

The R package quantmod is installed through the following command:

```
install.packages("quantmod")
```

```
library(quantmod)
getSymbols("STI",from="1990-01-03",to="2015-02-01",src="yahoo")
stock.rtn=diff(log(Ad(`STI`)))
returns <- as.vector(stock.rtn)
m=mean(returns,na.rm=TRUE)
s=sd(returns,na.rm=TRUE)
times=index(stock.rtn)
n = sum(is.na(returns))+sum(!is.na(returns))
x=seq(1,n)
y=rnorm(n, m, s)
plot(times,returns,pch=19,cex=0.03,col="blue", ylab="X", xlab="n", main = '')
segments(x0 = times, x1 = times, y0 = 0, y1 = returns,col="blue")
points(times,y,pch=19,cex=0.3,col="red", ylab="X", xlab="n", main = '')
```

```
getSymbols("DJIA",from="1990-01-03",to="2015-02-01",src="FRED")
stock.rtn=diff(log(`DJIA`))
returns <- as.vector(stock.rtn)
m=mean(returns,na.rm=TRUE)
s=sd(returns,na.rm=TRUE)
times=index(stock.rtn)
n = sum(is.na(returns))+sum(!is.na(returns))
x=seq(1,n)
y=rnorm(n, m, s)
plot(times,returns,pch=19,cex=0.03,col="blue", ylab="X", xlab="n", main = '')
segments(x0 = times, x1 = times, y0 = 0, y1 = returns,col="blue")
points(times,y,pch=19,cex=0.3,col="red", ylab="X", xlab="n", main = '')
```

The next figures illustrate the mismatch between the distributional properties of market *vs* Gaussian returns.

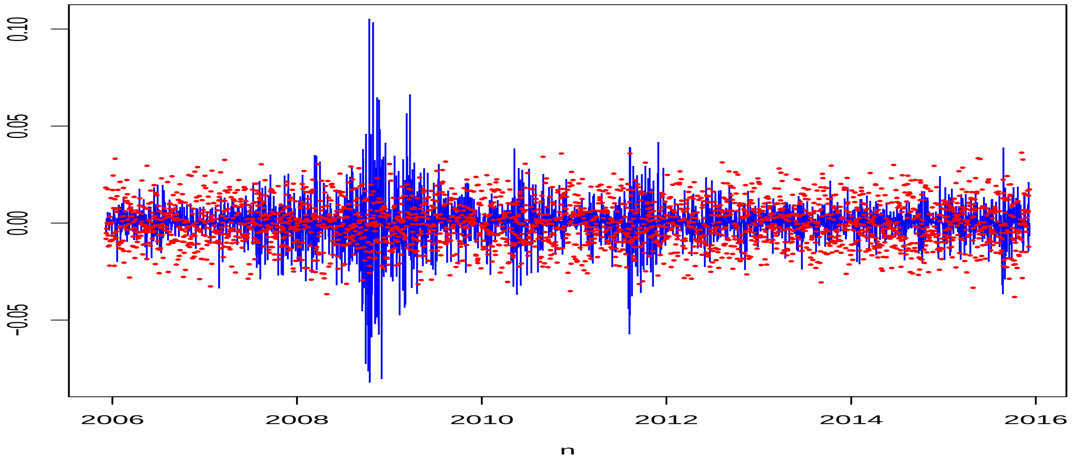

Figure 0.1: Market returns *vs* normalized Gaussian returns.

```
stock.ecdf=ecdf(as.vector(stock.rtn))
x <- seq(-12*s, 12*s, length=100)
px <- pnorm((x-m)/s)
plot(stock.ecdf, xlab = 'Sample Quantiles', col="blue",ylab = '', main = 'Empirical Cumulative Distribution')
lines(x, px, type="l", lty=2, col="red",xlab="x value",ylab="Density", main="Gaussian cdf")
legend("topleft", legend=c("Empirical cdf", "Gaussian cdf"),col=c("blue", "red"), lty=1:2, cex=0.8)
```

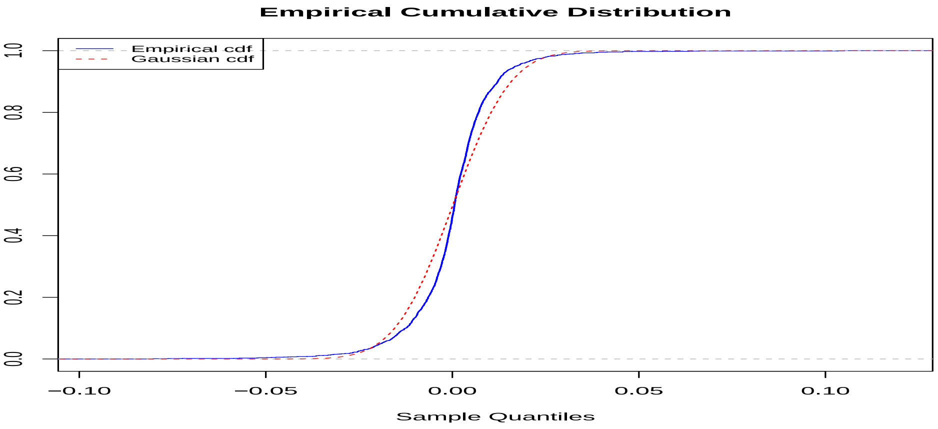

Figure 0.2: Empirical *vs* Gaussian CDF.

The following graph is obtained with the `qqnorm(returns);qqline(returns)` command.

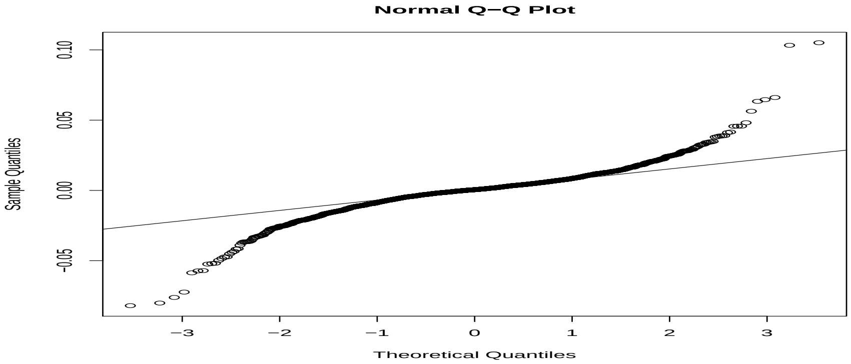

Figure 0.3: Quantile-Quantile plot.

```
x <- seq(-0.25, 0.25, length=100)
qx <- dnorm(x,mean=m,sd=s)
stock.dens=density(returns,na.rm=TRUE)
plot(stock.dens, xlab = 'x', col="red",ylab = '', main = 'Empirical Probability Density Function')
lines(x, qx, type="l", lty=2, col="blue",xlab="x value",ylab="Density", main="Gaussian cdf")
legend("topleft", legend=c("Empirical pdf", "Gaussian pdf"),col=c("red", "blue"), lty=1:2, cex=0.8)
```

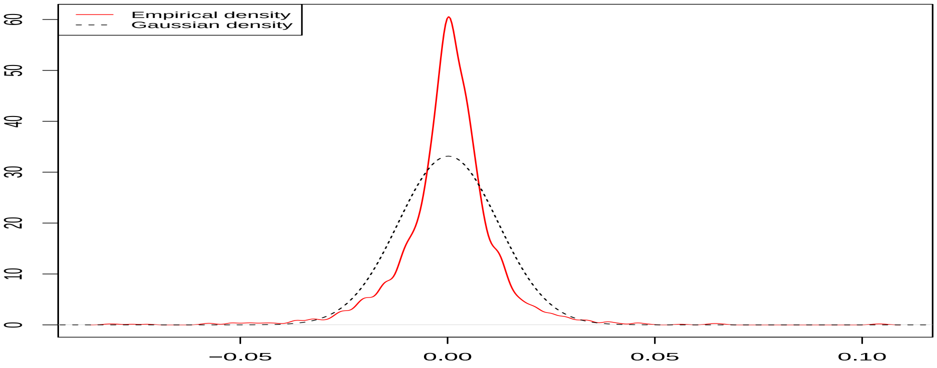

Figure 0.4: Empirical density vs normalized Gaussian density.

On the other hand, power tail densities can provide a better fit of empirical densities, as shown in Figure 0.5.

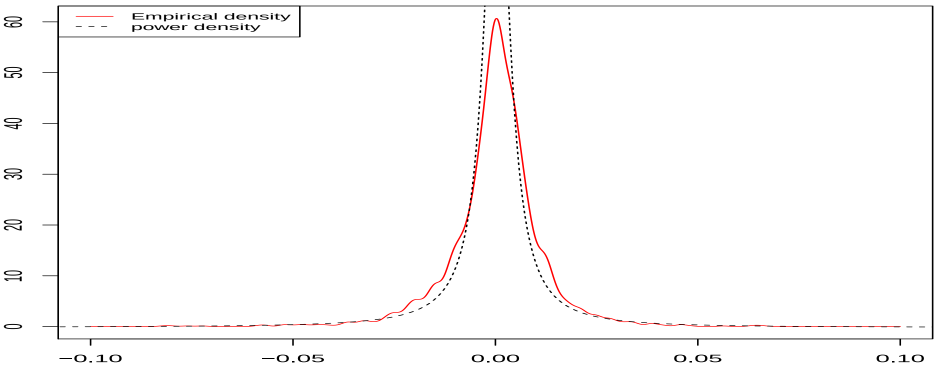

Figure 0.5: Empirical density vs power density.

The above fitting of empirical density with a power law is obtained through the following R script.

```
install.packages("pracma")
x <- seq(-0.25, 0.25, length=1000)
stock.dens=density(returns,na.rm=TRUE, from = -0.1, to = 0.1, n = 1000)
library(pracma)
a<-rationalfit(stock.dens$x, stock.dens$y, d1=2, d2=2)
plot(stock.dens$x,stock.dens$y, type = "l",xlab = '', col="red",ylab = '', main = '')
lines(x, (a$p1[3]+a$p1[2]*x+a$p1[1]*x^2)/(a$p2[3]+a$p2[2]*x+a$p2[1]*x^2),type="l",
    lty=2,col="blue",xlab="x value",ylab="Density",main="")
legend("topleft", legend=c("Empirical density", "power density"),col=c("red", "blue"),
    lty=1:2, cex=0.8)
```

The output of the command `rationalfit` is

```
$p1
[1] -0.184717249 -0.001591433  0.001385017


$p2
[1]  1.000000e+00 -6.460948e-04  1.314672e-05
```

which yields a rational fraction of the form

$$\begin{aligned} x \quad \longmapsto \quad & \frac{0.001385017 - 0.001591433 \times x - 0.184717249 \times x^2}{1.314672 \ 10^{-5} - 6.460948 \ 10^{-4} \times x + x^2} \\ \simeq \quad & -0.184717249 - \frac{0.001591433}{x} + \frac{0.001385017}{x^2}, \end{aligned}$$

which approximates the empirical density of DJIA returns in the least squares sense.

Retrieving option price data with R

The following code adapted from

https://mktstk.wordpress.com/2014/12/29/start-trading-like-a-quant-download-option-chains-from-google-finance-in-r/

retrieves option price data from *e.g.*

http://www.google.com/finance/option_chain?q=AAPL

```r
install.packages("RCurl")
install.packages("jsonlite")
```

```r
library(RCurl)
library(jsonlite)
```

```r
getOptionQuote <- function(symbol){
    output = list()
    goc = 'http://www.google.com/finance/option_chain?q='
    url = paste(goc, symbol, '&output=json', sep = "")
    x = getURL(url)
    fix = fixJSON(x)
    json = fromJSON(fix)
    numExp = dim(json$expirations)[1]
    for(i in 1:numExp){
        y = json$expirations[i,]$y
        m = json$expirations[i,]$m
        d = json$expirations[i,]$d
        expName = paste(y, m, d, sep = "_")
        if (i > 1){
          url=paste(goc,symbol,'&output=json&expy=',y,'&expm=',m,'&expd=',d,
              sep="")
          json = fromJSON(fixJSON(getURL(url)))
        }
        output[[paste(expName, "calls", sep = "_")]] = json$calls
        output[[paste(expName, "puts", sep = "_")]] = json$puts
    }
    return(output)
}
```

The next function relies on the JSON (JavaScript Object Notation) format.

```
fixJSON <- function(json_str){
    stuff = c('cid','cp','s','cs','vol','expiry','underlying_id','underlying_price','p','c','oi'
        ,'e','b','strike','a','name','puts','calls','expirations','y','m','d')
     for(i in 1:length(stuff)){
        replacement1 = paste(',"', stuff[i], '":', sep = "")
        replacement2 = paste('\\{"', stuff[i], '":', sep = "")
        regex1 = paste(',', stuff[i], ':', sep = "")
        regex2 = paste('\\{', stuff[i], ':', sep = "")
        json_str = gsub(regex1, replacement1, json_str)
        json_str = gsub(regex2, replacement2, json_str)
     }
     return(json_str)
}
```

```
aapl_opt = getOptionQuote("AAPL")
plot(aapl_opt$"2017_1_20_calls"$strike, aapl_opt$"2017_1_20_calls"$p, type = "s",
     main = "Price by Strike")
```

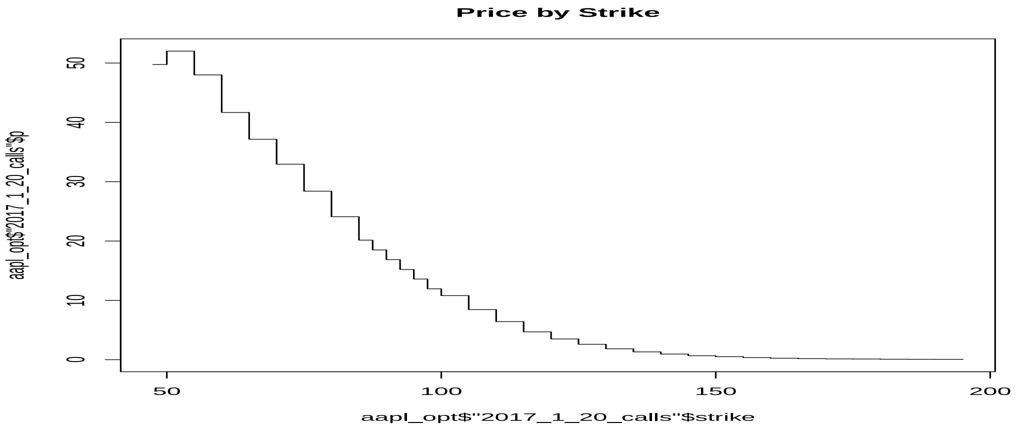

Figure 0.6: Option prices plotted against strikes.

```
aapl_opt = getOptionQuote("AAPL")
data<-as.data.frame(do.call(rbind,aapl_opt))
aapl_data<-data[,-c(1,2,4,6,8,9,10,11)]
head(aapl_data)
```

```
goog_opt = getOptionQuote("GOOG")
data<-as.data.frame(do.call(rbind,goog_opt))
goog_data<-data[,-c(1,2,4,6,8,9,10,11)]
head(goog_data)
```

Exporting option price data

```
write.table(goog_data, file = "goog")
write.csv(goog_data, file = "goog.csv")
install.packages("xlsx")
library(xlsx)
write.xlsx(goog_data, file = "goog.xlsx")
```