Exploring Higher Order Risk Premia Using High Frequency Data

R/Finance Chicago 2016

Jerzy Pawlowski *jp3900@nyu.edu*

NYU Tandon School of Engineering

May 20, 2016

NYU | TANDON SCHOOL OF ENGINEERING

Investor *Utility*, *Risk Aversion*, *Prudence* and *Temperance*

The *expected utility* hypothesis states that investor risk and return preferences are based on the expected value of the *utility* of their wealth, instead of on the level of wealth,

Investor risk and return preferences depend on the signs of the derivatives of the *utility* function,

A *utility* function with a positive first derivative implies a preference for higher *returns* (first moment), while a negative second derivative implies risk aversion and a preference for lower *volatility* (second moment),

A positive third derivative implies *prudence*, or a preference for higher *skewness* (third moment), while a negative fourth derivative implies *temperance*, or a preference for lower *kurtosis* (fourth moment),

Investors with a logarithmic *utility* of wealth base their preferences on the percentage change of wealth, instead of the absolute change, and have a preference for larger odd moments and smaller even moments,

$$u(w) = u(w_0) + \mathrm{d}w \frac{\mathrm{d}u}{\mathrm{d}w} + \mathrm{d}w^2 \frac{1}{2} \frac{\mathrm{d}^2 u}{\mathrm{d}w^2} + \mathrm{d}w^3 \frac{1}{3!} \frac{\mathrm{d}^3 u}{\mathrm{d}w^3} + \mathrm{d}w^4 \frac{1}{4!} \frac{\mathrm{d}^4 u}{\mathrm{d}w^4} + \dots$$

$$\mathrm{d}E[u] = \alpha_1 \mathit{mean} + \alpha_1 \mathit{variance} + \alpha_1 \mathit{skew} + \alpha_4 \mathit{kurtosis} + \dots$$

mean: $\bar{w} = \frac{1}{k} \sum_{i=1}^{k} w_i$

variance: $\hat{\sigma}^2 = \frac{1}{k-1} \sum_{i=1}^{k} (w_i - \bar{w})^2$

skew: $\hat{s} = \frac{1}{k-1} \sum_{i=1}^{k} (\frac{w_i - \bar{w}}{\hat{\sigma}})^3$

kurtosis: $\hat{k} = \frac{1}{k-1} \sum_{i=1}^{k} (\frac{w_i - \bar{w}}{\hat{\sigma}})^4$

Investor Preferences and Empirical Return Distributions

Investor preference for higher *returns* and for lower *volatility* is expressed by maximizing the *Sharpe* ratio,

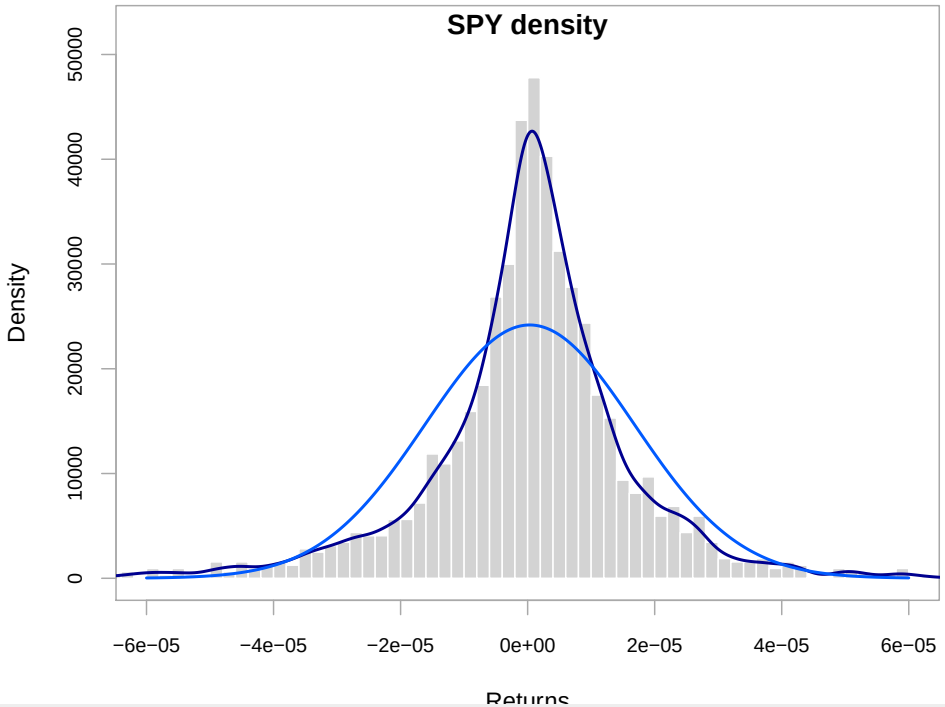
Stock price *momentum* refers to the fact that stocks with high past returns tend to have high future returns, and vice versa,

Stock price *momentum* has been observed, that is driven by approximately one-year of past returns:
Eugene Fama and Kenneth French, *Dissecting Anomalies*
Asness et al., *Fact, Fiction and Momentum Investing*

The question then is can momentum also work on shorter, intraday time scales?

Also, do higher moments (skew, kurtosis) have predictive power as well?
Amaya et al., *Does Realized Skewness Predict the Cross-Section of Equity Returns?*

Stocks typically have negative skew and excess kurtosis, the opposite of what investors prefer,




```
> mean_rets <- mean(re_turns[, 1])  # calculate
> sd_rets <- sd(re_turns[, 1])  # calculate stan
> # calculate skew and kurtosis
> (sum(((re_turns[, 1] - mean_rets)/sd_rets)^3))
> library(PerformanceAnalytics)
> chart.Histogram(re_turns[, 1], main="",
+   xlim=c(-6e-5, 6e-5),
+   methods = c("add.density", "add.normal"))
> # add title
> title(main=paste(sym_bol, "density"), line=-1)
```

Package HighFreq for Managing High Frequency Data

Package HighFreq contains functions for managing high frequency *TAQ* and *OHLC* market data:

- reading and writing data from files,
- managing time zones and alligning indices,
- chaining and joining time series,
- scrubbing bad data points,
- converting *TAQ* data to *OHLC* format,
- aggregating data to lower frequency,

HighFreq is inspired by the package highfrequency, and follows many of its conventions,

HighFreq depends on packages xts, quantmod, lubridate, and caTools,

The function scrub_agg() scrubs a single day of *TAQ* data, aggregates it, and converts it to *OHLC* format,

The function save_scrub_agg() loads, scrubs, aggregates, and binds multiple days of *TAQ* data for a single symbol, and saves the *OHLC*

```
> # install package "HighFreq" from github
> install.packages("devtools")
> library(devtools)
> install_github(repo="algoquant/HighFreq")
> # load package "HighFreq"
> library(HighFreq)
> # set data directories
> data_dir <- "C:/Develop/data/hfreq/src/"
> output_dir <- "C:/Develop/data/hfreq/scrub/"
> # define sym_bol
> sym_bol <- "SPY"
> # load a single day of TAQ data
> sym_bol <- load(
+   file.path(data_dir,
+       paste0(sym_bol, "/2014.05.02.",
+              sym_bol, ".RData")))
> # scrub, aggregate single day of TAQ data to O
> ohlc_data <- scrub_agg(taq_data=get(sym_bol))
> # aggregate TAQ data for symbol, save to file
> save_scrub_agg(sym_bol,
+          data_dir=data_dir,
+          output_dir=output_dir,
+          period="minutes")
```

High Frequency *OHLC* Data

Aggregating high frequency *TAQ* data into *OHLC* format with lower periodicity allows for data compression while maintaining some information about volatility,

```
> # load package "HighFreq"
> library(HighFreq)
> # define sym_bol
> sym_bol <- "SPY"
> # load OHLC data
> output_dir <- "C:/Develop/data/hfreq/scrub/"
> sym_bol <- load(
+   file.path(output_dir,
+       paste0(sym_bol, ".RData")))
> inter_val <-
+   "2013-11-11 09:30:00/2013-11-11 10:30:00"
> chart_Series(SPY[inter_val],
+       name=sym_bol)
```

Estimating Volatility From *OHLC* Data

Package `TTR` contains statistical estimators and technical indicators implemented in fast `C` code,

The function `volatility()` from package `TTR` estimates the volatility from *OHLC* data,

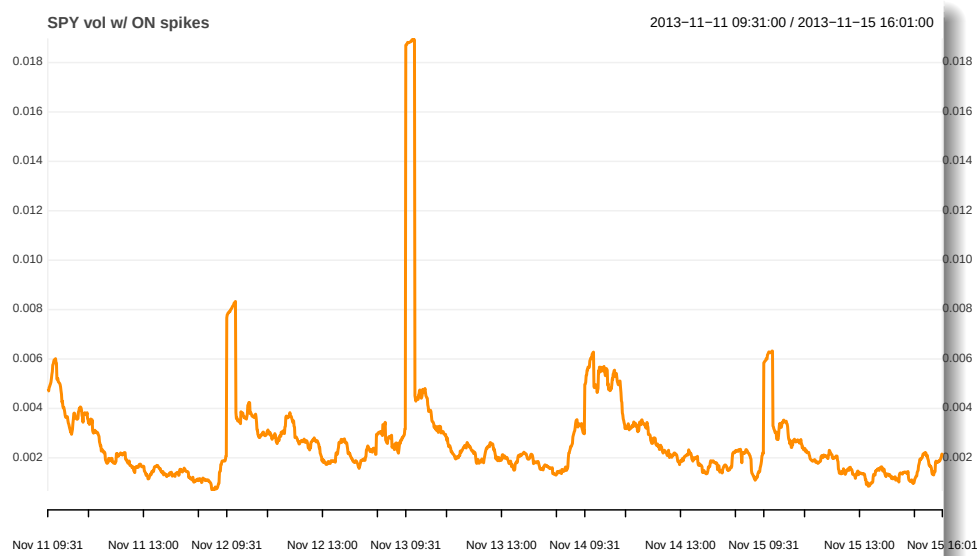
`volatility()` includes the *Garman-Klass* estimator:

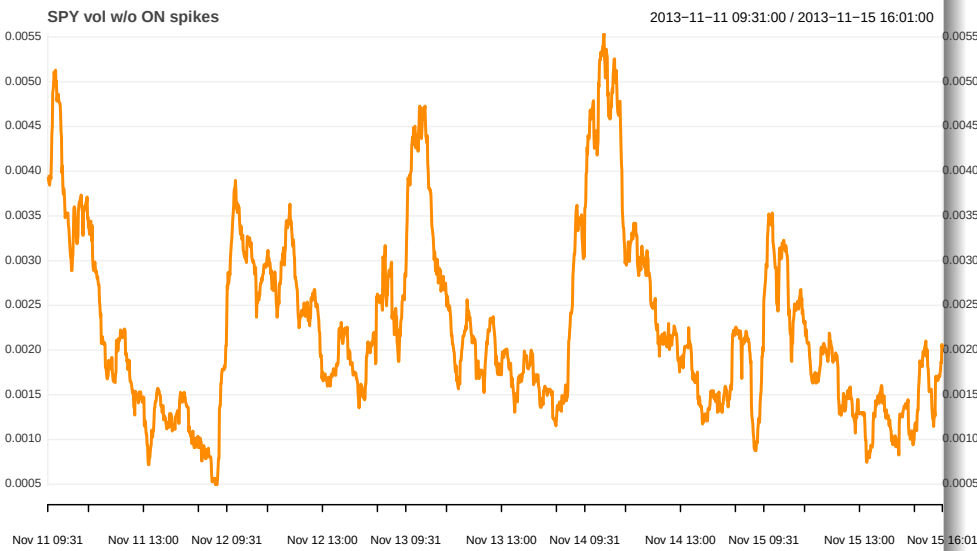
$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(0.5(H_i - L_i)^2 - (2\log 2 - 1)(C_i - O_i)^2)$$

and the *Rogers-Satchell* estimator:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}((H_i - O_i)(H_i - C_i) + (L_i - O_i)(L_i - C_i))$$

```
> library(quantmod)
> library(TTR)
> inter_val <- "2013-11-11/2013-11-15"
> var_iance <- volatility(OHLC=SPY,
+                 calc="yang.zhang", n=20)
> chart_Series(var_iance[inter_val],
+       name=paste(sym_bol, "vol w/ ON spikes"))
> var_iance <- volatility(OHLC=SPY,
+                 calc="rogers.satchell", n=20)
> chart_Series(var_iance[inter_val],
+       name=paste(sym_bol, "vol w/o ON spikes"))
```





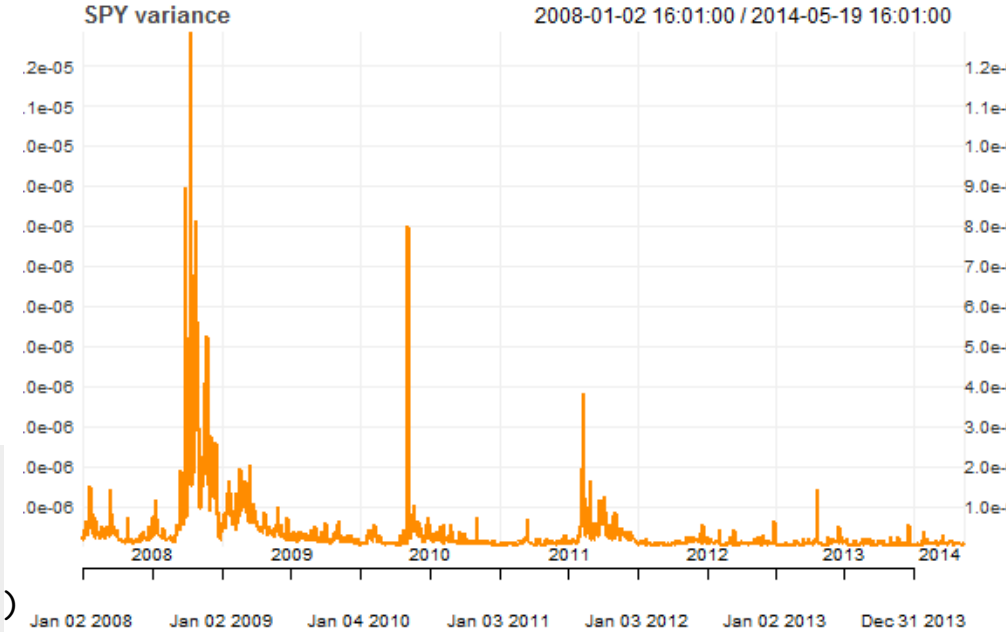
Daily Volume and Volatility From *OHLC* Data

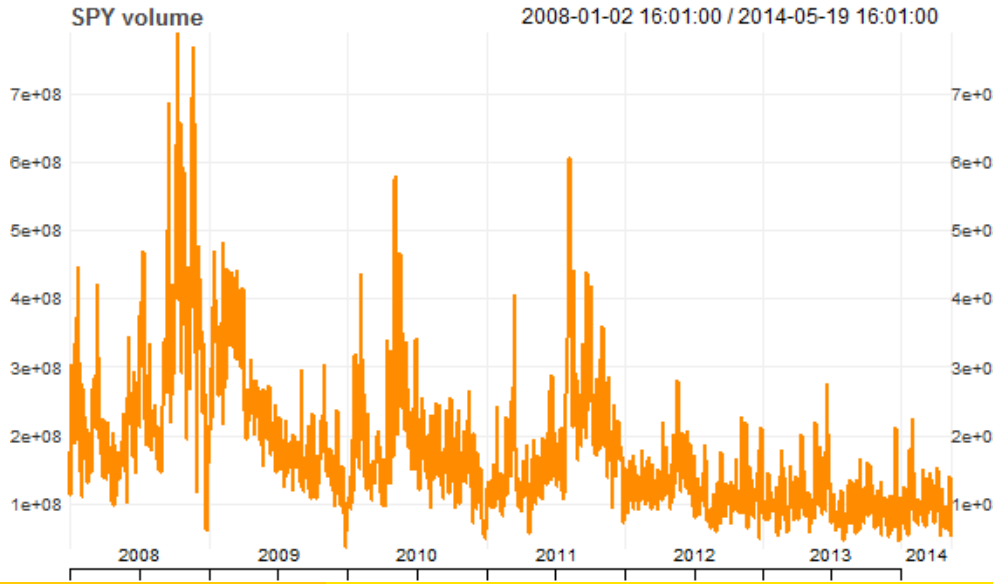
Trading volumes typically rise in sympathy with market price volatility,

The function `vol_ohlc()` from package `HighFreq` calculates volatility estimators over an `OHLC` time series,

The function `agg_ohlc()` calculates a statistical estimator over an `OHLC` time series,

```
> # daily variance and volume
> daily_var <- apply.daily(x=SPY, FUN=agg_ohlc,
+                          agg_fun="vol_ohlc")
> colnames(daily_var) <- paste0(sym_bol, ".var")
> daily_volume <- apply.daily(x=Vo(SPY), FUN=sum
> chart_Series(daily_var,
+        name=paste(sym_bol, "variance"))
> chart_Series(daily_volume,
+        name=paste(sym_bol, "volume"))
```





Measuring Market Liquidity Using High Frequency Data

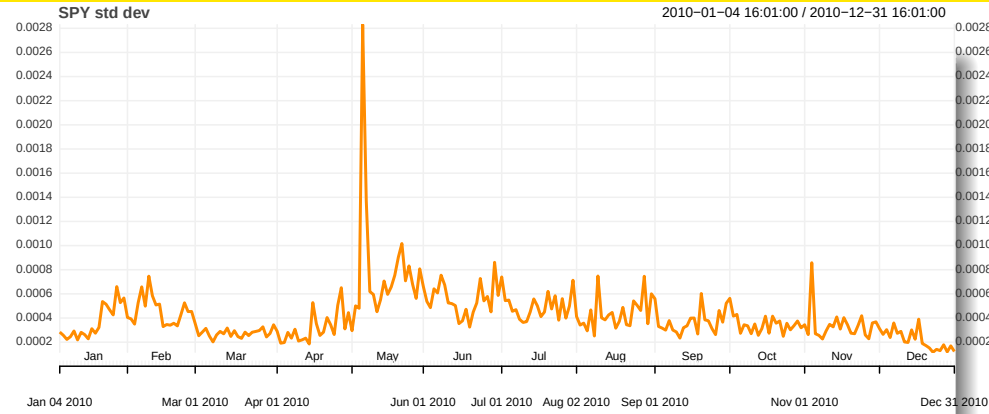
Market illiquidity is defined as the market price impact resulting from supply-demand imbalance,

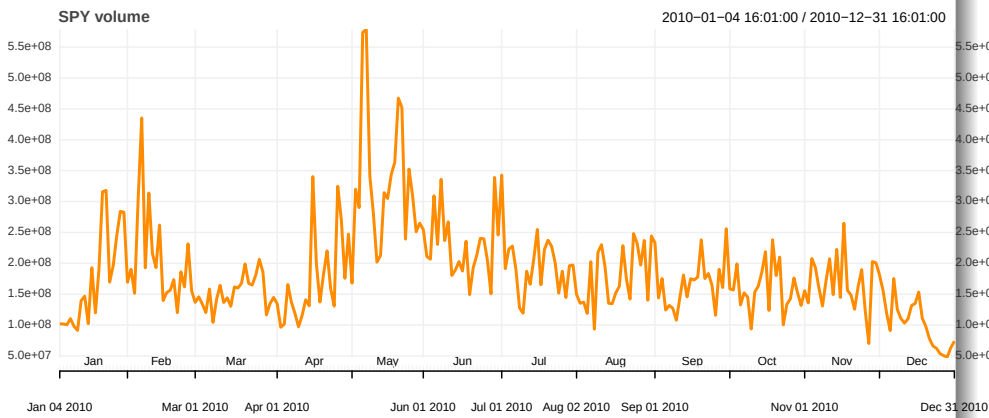
Market illiquidity is proportional to the price volatility divided by the square root of the trading volume:

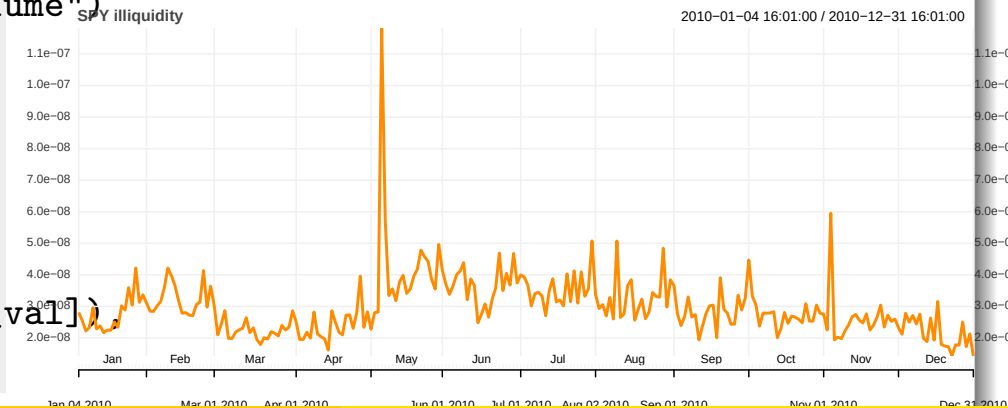
$$\mathcal{L}^{-1} \sim \frac{\sigma}{\sqrt{v}}$$

Recent research suggests that market crashes are conditioned by declining market liquidity: Donier et al., *Why Do Markets Crash? Bitcoin Data Offers Unprecedented Insights*

```
> # daily volume
> daily_volu <- apply.daily(x=SPY,
+         FUN=function(da_ta) sum(Vo(da_ta)))
> colnames(daily_volu) <- paste0(sym_bol, ".volume")
> inter_val <- "2010"
> chart_Series(sqrt(daily_var[inter_val]),
+        name=paste(sym_bol, "std dev"))
> chart_Series(daily_volu[inter_val],
+        name=paste(sym_bol, "volume"))
> chart_Series(
+   sqrt(daily_var[inter_val]/daily_volu[inter_val]),
+   name=paste(sym_bol, "illiquidity"))
```







Estimating Hurst Exponent From *OHLC* Data

The Hurst exponent is a measure of long-term memory of a time series, and is related to its autocorrelation:

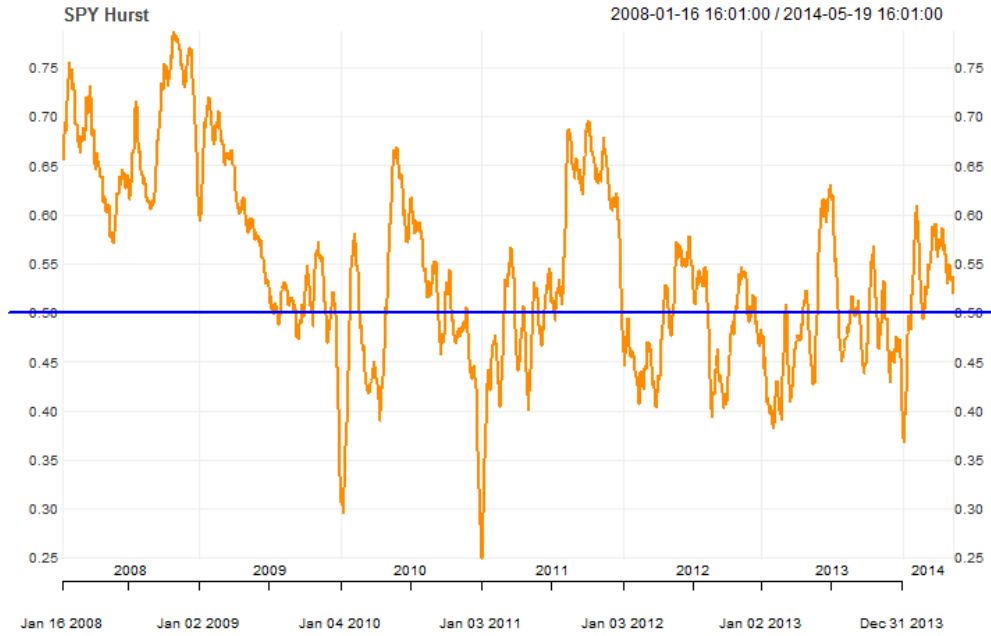
$$\mathbb{E}[\frac{(max(p) - min(p))}{\hat{\sigma}}] = t^H$$

$H = 0.5$ for Brownian motion (no autocorrelations),
$H > 0.5$ for positive autocorrelations,
$H < 0.5$ for negative autocorrelations,

The function `hurst_ohlc()` from package `HighFreq` calculates a Hurst-like indicator:

$$H = \frac{1}{n}\sum_{i=1}^{n} log(\frac{H_i - L_i}{abs(C_i - O_i)})$$

The function `agg_ohlc()` calculates a statistical estimator over an `OHLC` time series,

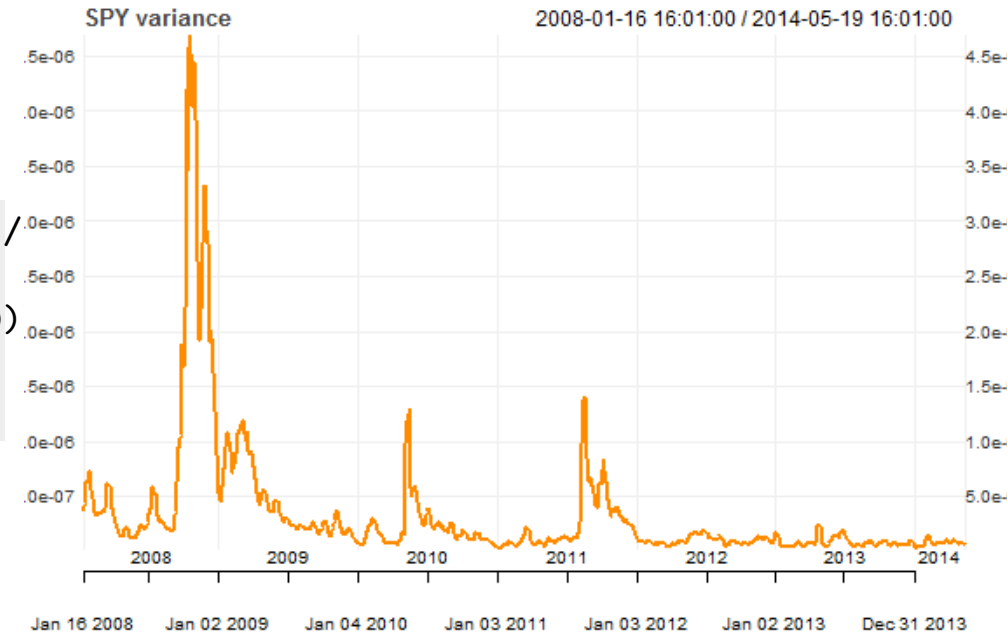


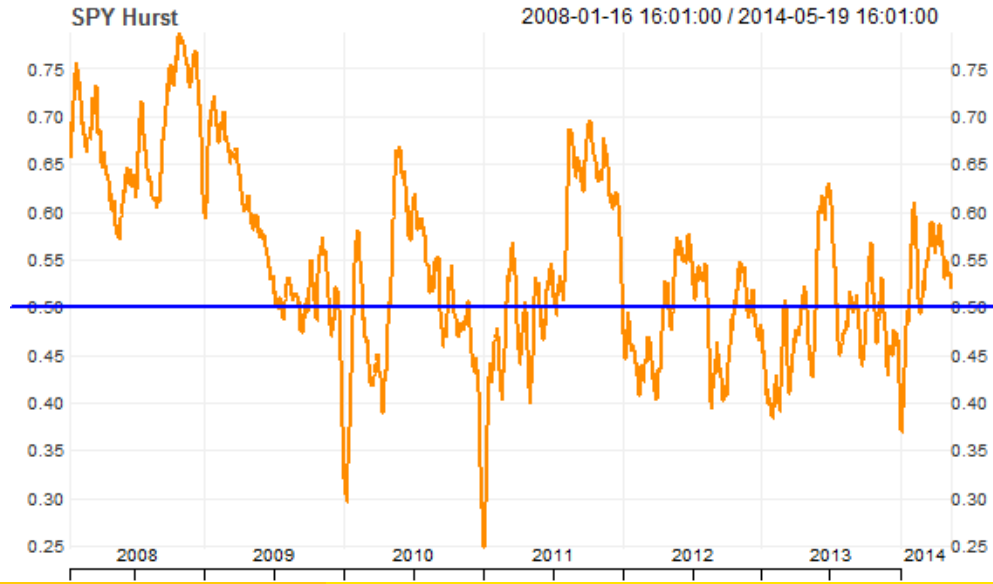

```
> # daily Hurst exponents
> daily_hurst <- apply.daily(x=SPY,
+                            FUN=agg_ohlc,
+                            agg_fun="hurst_ohlc")
> colnames(daily_hurst) <-
+   paste(col_name(get(sym_bol)), ".Hurst")
> chart_Series(roll_sum(daily_hurst, 10)[-(1:10)]
+         name=paste(sym_bol, "Hurst"))
> abline(h=0.5, col="blue", lwd=2)
```

Daily Volatility and Hurst Exponent

The Hurst exponent is typically greater when market volatility is higher, and is usually above 0.5 in periods high volatility,

```
> chart_Series(roll_sum(daily_var, 10)[-(1:10)]/
+         name=paste(sym_bol, "variance"))
> chart_Series(roll_sum(daily_hurst, 10)[-(1:10)
+         name=paste(sym_bol, "Hurst"))
> abline(h=0.5, col="blue", lwd=2)
```



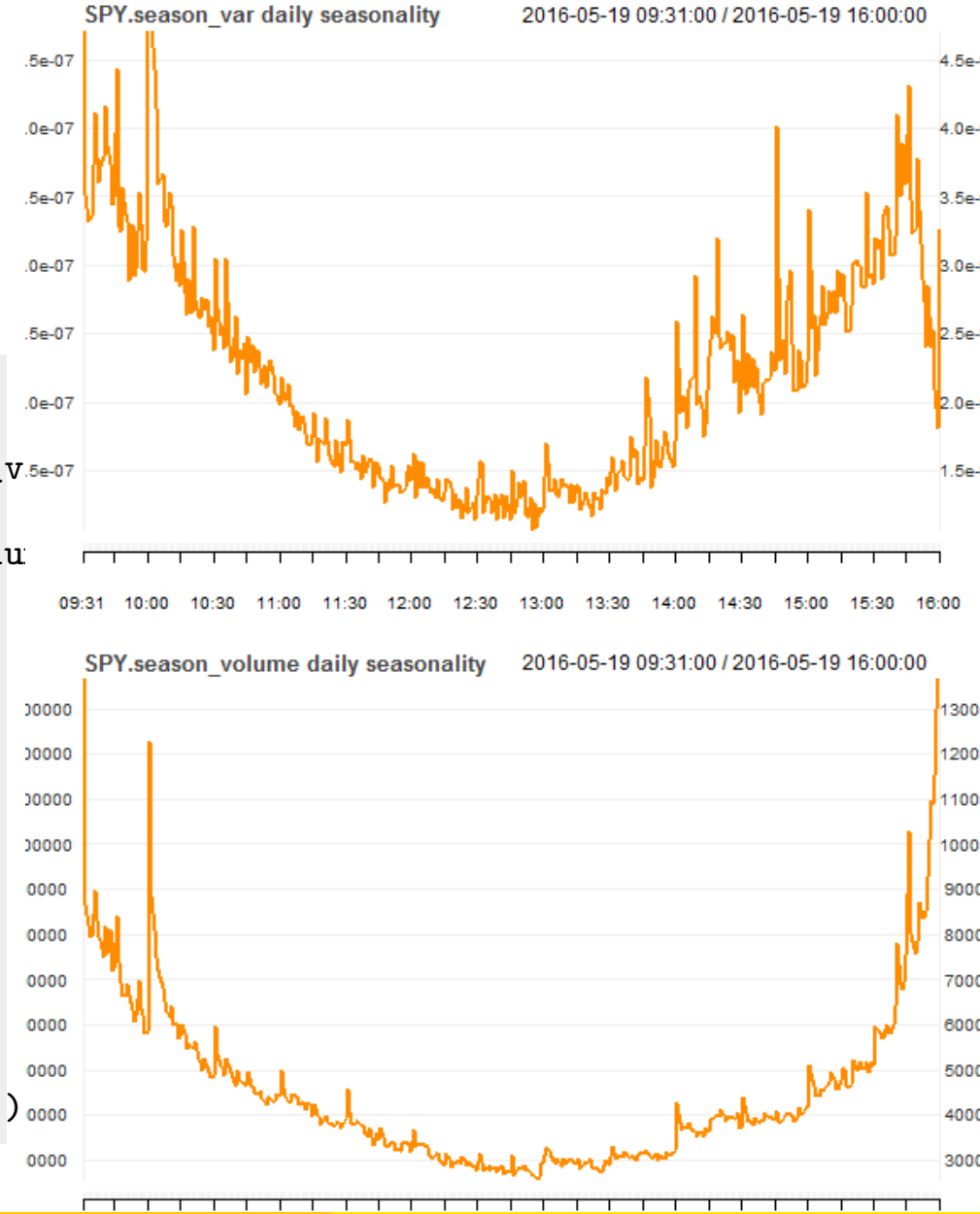


Intraday Seasonality of Volume and Volatility

Market price volatility and trading volumes are typically higher at the beginning and end of the trading sessions,

The function `season_ality()` calculates the seasonality aggregations of a statistical estimator over an `OHLC` time series,

```
> # intraday seasonality of volume
> season_volume <- season_ality(Vo(SPY))
> season_volume <- season_volume[-(nrow(season_v
> colnames(season_volume) <-
+   paste0(col_name(get(sym_bol)), ".season_volu
> plot_theme <- chart_theme()
> plot_theme$format.labels <- "%H:%M"
> ch_ob <- chart_Series(x=season_volume,
+   name=paste(colnames(season_volume),
+   "intraday seasonality"), theme=plot_theme,
+   plot=FALSE)
> y_lim <- ch_ob$get_ylim()
> y_lim[[2]] <- structure(c(y_lim[[2]][1],
+         y_lim[[2]][2]/2), fixed=TRUE)
> ch_ob$set_ylim(y_lim)
> plot(ch_ob)
> # intraday seasonality of volatility
> season_var <- season_ality(vol_ohlc(ohlc=SPY))
```

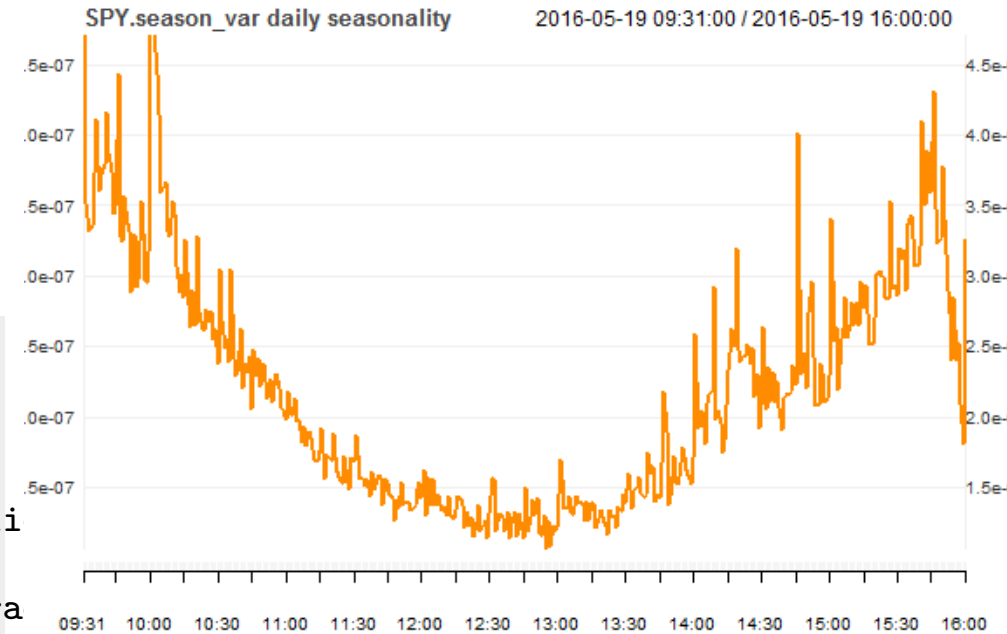


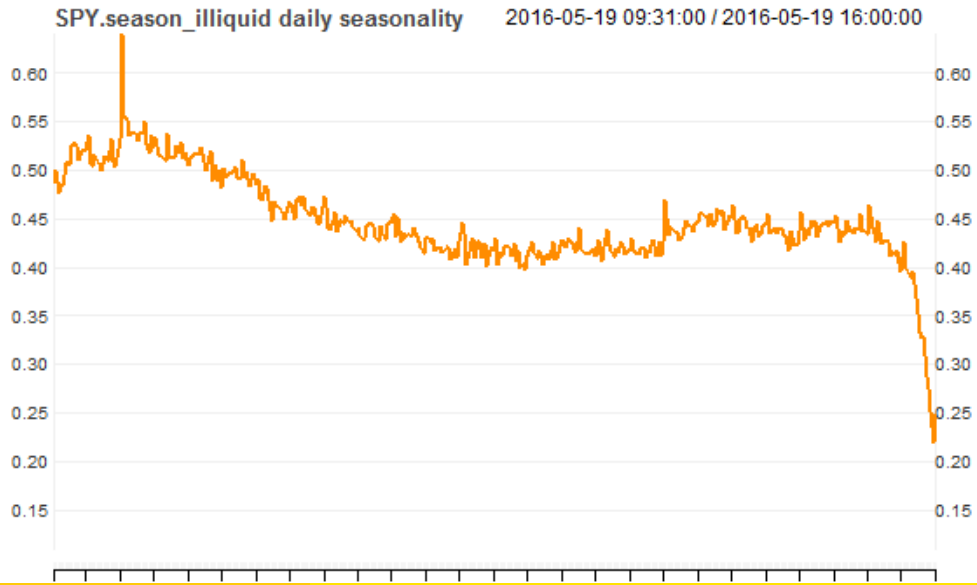
Intraday Seasonality of Liquidity and Volatility

Market illiquidity rises in sympathy with market price volatility,

Market liquidity is typically the highest at the end of the trading session, and the lowest at the beginning,

```
> # intraday seasonality of liquidity
> season_illiquid <- season_ality(
+   sqrt(vol_ohlc(ohlc=SPY)/Vo(SPY)))
> colnames(season_illiquid) <-
+   paste0(col_name(get(sym_bol)), ".season_illi
> chart_Series(x=season_illiquid,
+   name=paste(colnames(season_illiquid), "intra
> # intraday seasonality of volatility
> season_var <- season_ality(vol_ohlc(ohlc=SPY))
```

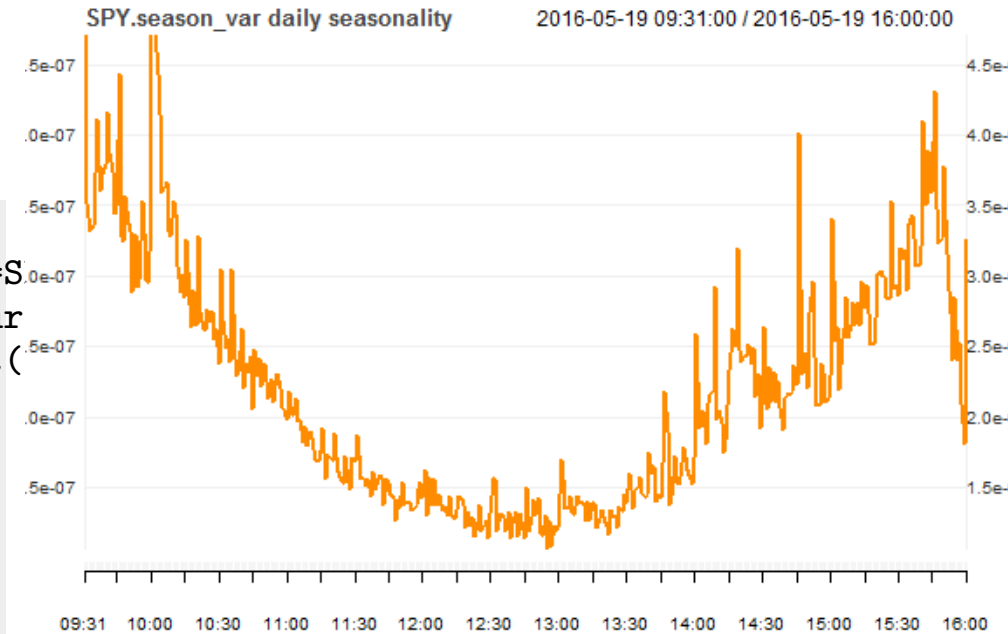


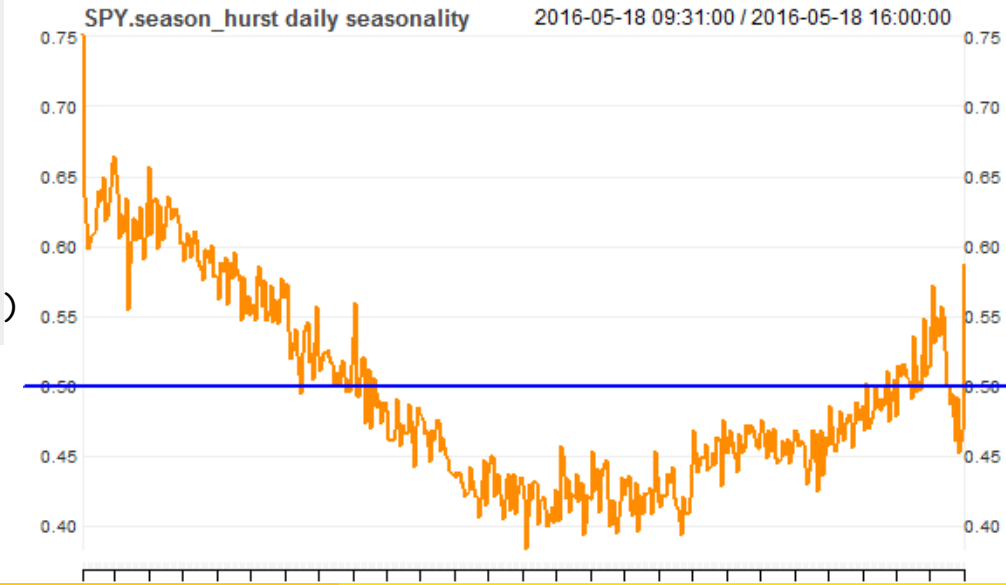


Intraday Seasonality of Hurst Exponent and Volatility

The Hurst exponent typically moves higher with higher market price volatility, and is above 0.5 with high volatility,

```
> # intraday seasonality of Hurst exponent
> season_hurst <- season_ality(hurst_ohlc(ohlc=S
> season_hurst <- season_hurst[-(nrow(season_hur
> colnames(season_hurst) <- paste0(col_name(get(
> plot_theme <- chart_theme()
> plot_theme$format.labels <- "%H:%M"
> ch_ob <- chart_Series(x=season_hurst,
+   name=paste(colnames(season_hurst),
+   "intraday seasonality"), theme=plot_theme,
+   plot=FALSE)
> y_lim <- ch_ob$get_ylim()
> y_lim[[2]] <- structure(c(y_lim[[2]][1],
+         y_lim[[2]][2]), fixed=TRUE)
> ch_ob$set_ylim(y_lim)
> plot(ch_ob)
> abline(h=0.5, col="blue", lwd=2)
> # intraday seasonality of volatility
> season_var <- season_ality(vol_ohlc(ohlc=SPY))
```

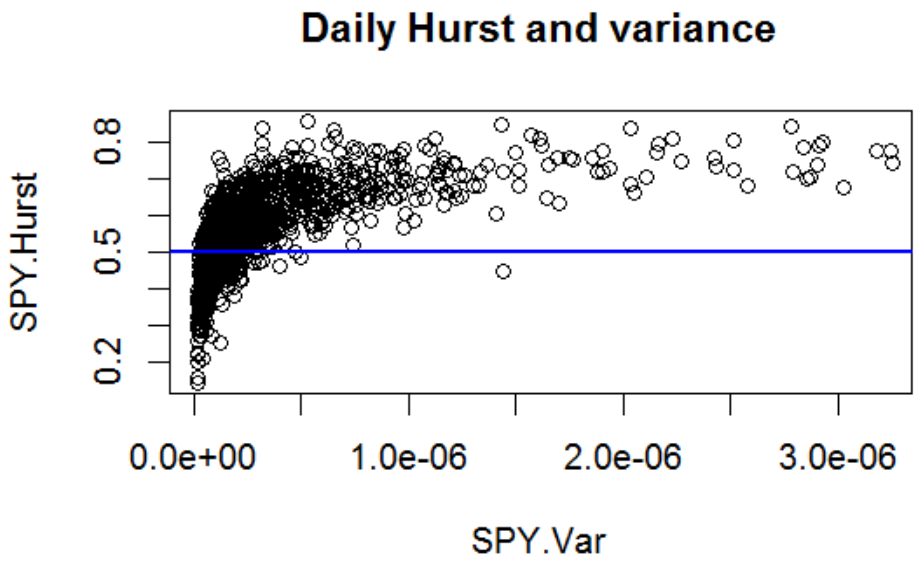


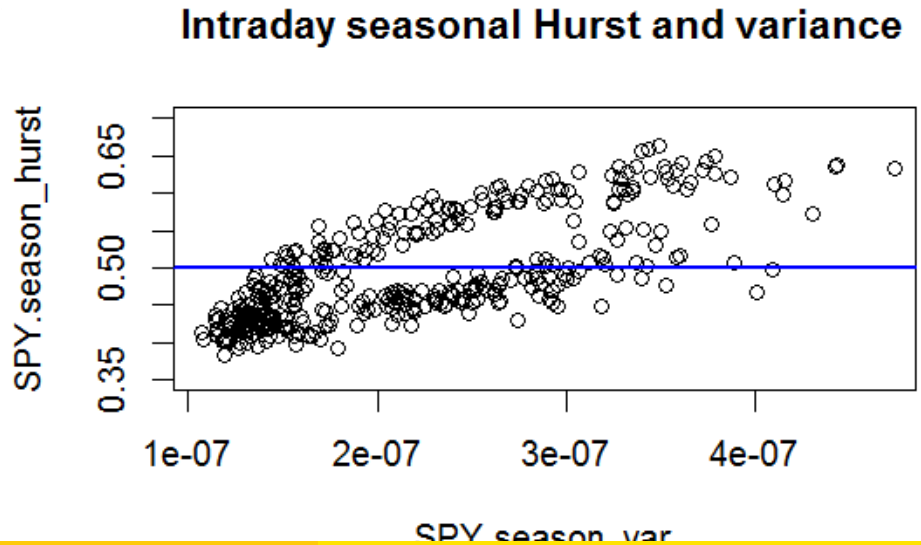


Dependence of Hurst Exponent on Volatility

The Hurst exponent typically moves higher with higher market price volatility, and is above 0.5 with high volatility,

```
> # intraday seasonality of Hurst exponent
> ran_ge <- range(daily_var)
> plot(x=as.vector(daily_var), y=as.vector(daily
+      xlab=colnames(daily_var), ylab=colnames(d
+      main="Daily Hurst and variance",
+      xlim=c(ran_ge[1], ran_ge[2]/4))
> abline(h=0.5, col="blue", lwd=2)
> ran_ge <- range(season_var)
> plot(x=as.vector(season_var), y=as.vector(seas
+      xlab=colnames(season_var), ylab=colnames(
+      main="Intraday seasonal Hurst and varianc
+      xlim=c(ran_ge[1], ran_ge[2]/2), ylim=c(0.
> abline(h=0.5, col="blue", lwd=2)
```





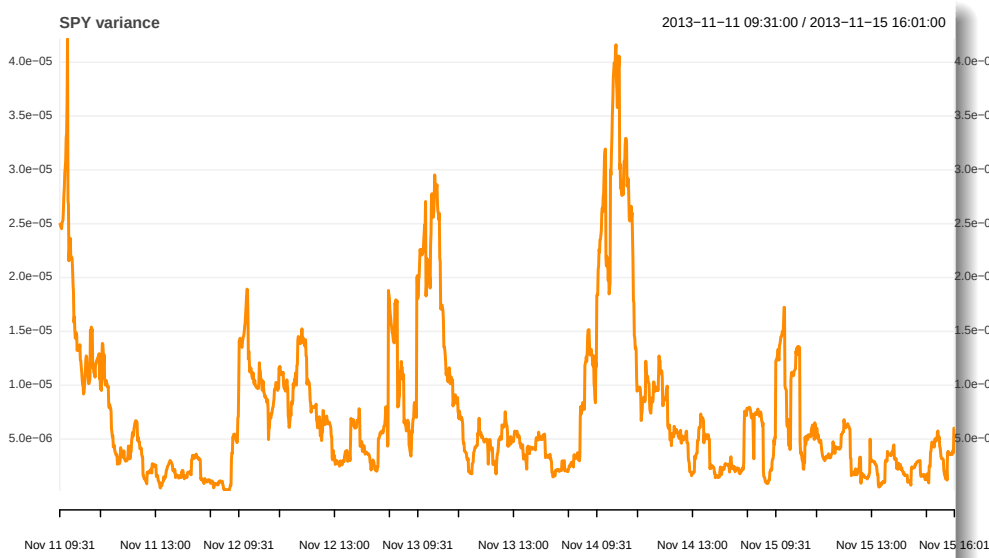
Estimating Skew From *OHLC* Data

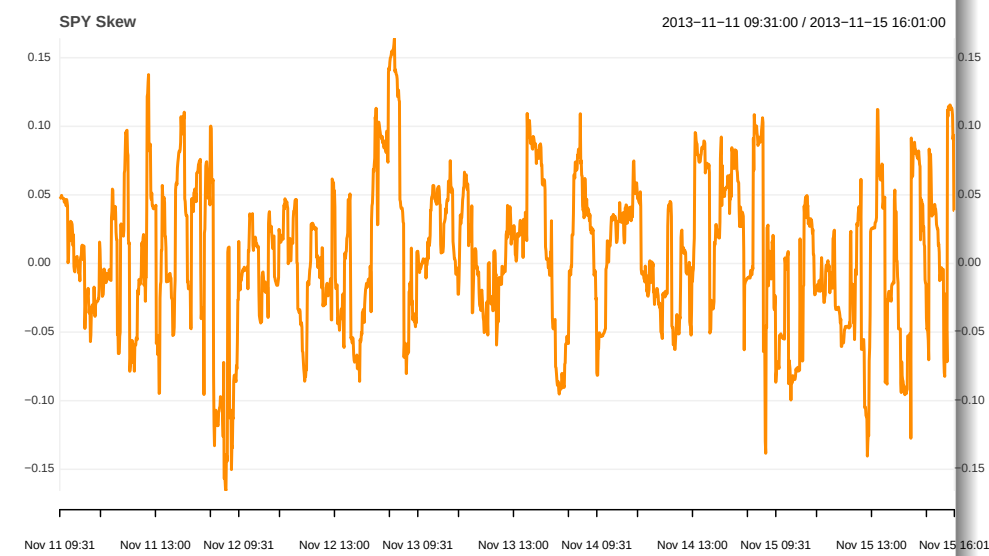
The function `skew_ohlc()` from package `HighFreq` calculates a skew-like indicator:

$$s^3 = \frac{1}{n}\sum_{i=1}^{n}((H_i - O_i)(H_i - C_i)(H_i - 0.5(O_i + C_i)) + (L_i - O_i)(L_i - C_i)(L_i - 0.5(O_i + C_i))$$

The function `roll_agg_ohlc()` aggregates rolling, volume weighted moment estimators,

```
> library(HighFreq)  # load package "HighFreq"
> # rolling variance
> var_iance <-
+   roll_agg_ohlc(ohlc=SPY, agg_fun="vol_ohlc")
> # rolling skew
> sk_ew <-
+   roll_agg_ohlc(ohlc=SPY, agg_fun="skew_ohlc")
> sk_ew <- sk_ew/(var_iance)^(1.5)
> sk_ew[1, ] <- 0
> sk_ew <- na.locf(sk_ew)
> inter_val <- "2013-11-11/2013-11-15"
> chart_Series(var_iance[inter_val],
+       name=paste(sym_bol, "variance"))
> chart_Series(sk_ew[inter_val],
+       name=paste(sym_bol, "Skew"),
+       ylim=c(-1, 1))
```





Daily Volatility and Skew From *OHLC* Data

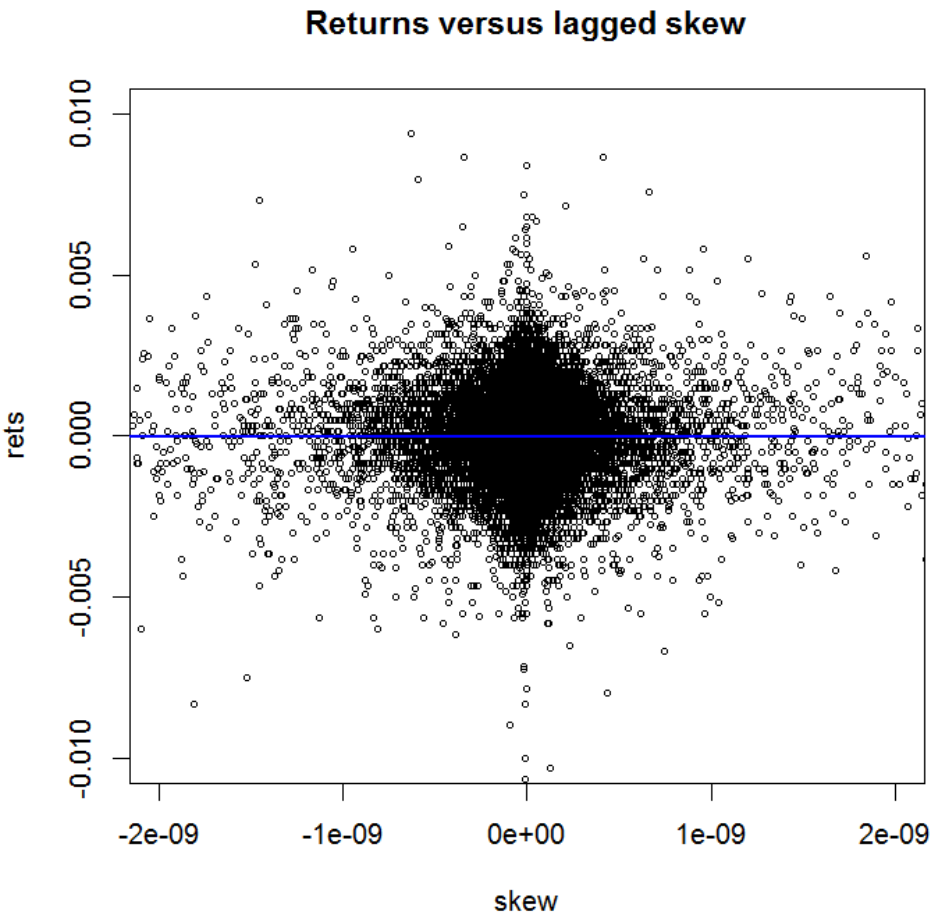
The function `agg_ohlc()` calculates a statistical estimator over an OHLC time series,

```
> # daily variance and skew
> daily_var <- apply.daily(x=SPY, FUN=agg_ohlc,
+                  agg_fun="vol_ohlc")
> colnames(daily_var) <- paste0(sym_bol, ".var")
> daily_skew <- apply.daily(x=SPY, FUN=agg_ohlc,
+                  agg_fun="skew_ohlc")
> daily_skew <- daily_skew/(daily_var)^(1.5)
> colnames(daily_skew) <- paste0(sym_bol, ".skew")
> inter_val <- "2013-06-01/"
> chart_Series(daily_var[inter_val],
+        name=paste(sym_bol, "variance"))
> chart_Series(daily_skew[inter_val],
+        name=paste(sym_bol, "skew"))
```

Regression of Skews Versus Returns

A regression of lagged skews versus returns appears to be statistically significant, especially in periods of high volatility during the financial crisis of 2008-09,

```
> re_turns <- calc_rets(x_ts=SPY)
> sk_ew <- skew_ohlc(ohlc=SPY[, -5])
> colnames(sk_ew) <- paste0(sym_bol, ".skew")
> lag_skew <- lag(sk_ew)
> lag_skew[1, ] <- 0
> da_ta <- cbind(re_turns[, 1], lag_skew)
> for_mula <- as.formula(paste(colnames(da_ta)[1
+     paste(paste(colnames(da_ta)[-1],
+       collapse=" + ")), sep="~"))
> l_m <- lm(for_mula, data=da_ta["2010"])
> summary(l_m)$coef
> summary(lm(for_mula, data=da_ta["2011"]))$coef
> summary(lm(for_mula, data=da_ta["2012"]))$coef
```




```
> plot(for_mula, data=da_ta["2010"],
+      main="Returns versus lagged skew",
+      xlim=c(-2e-09, 2e-09), ylim=c(-0.01, 0.01
+      cex=0.6, xlab="skew", ylab="rets")
> abline(l_m, col="blue", lwd=2)
```

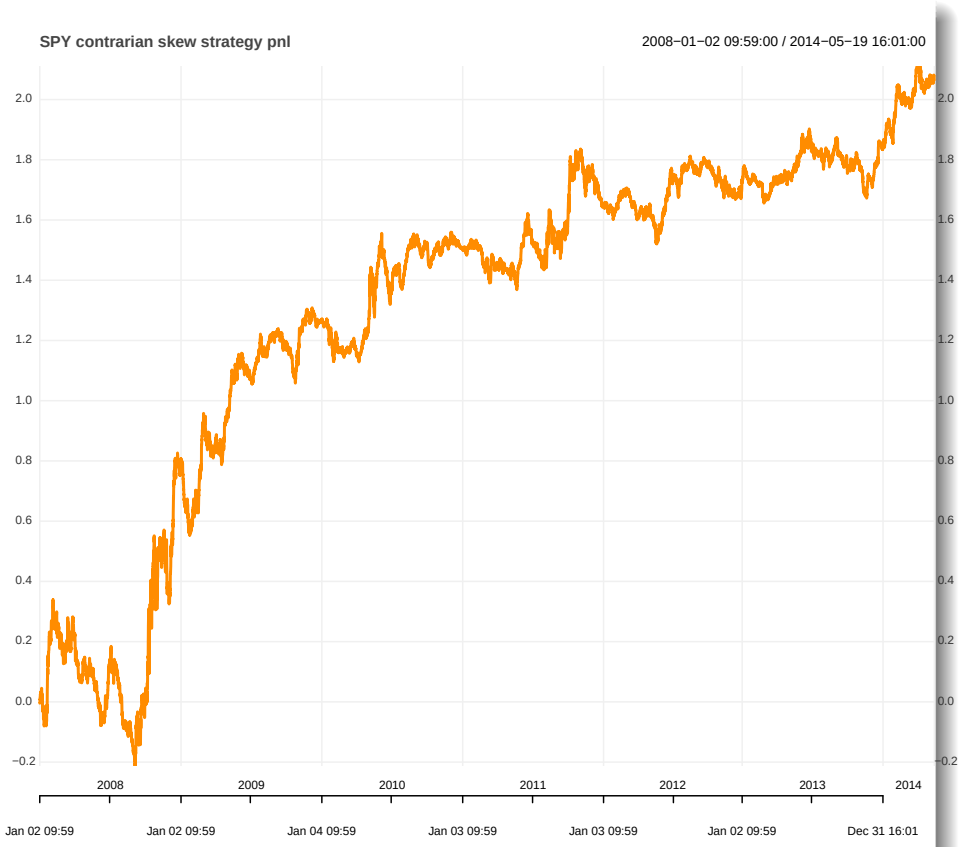
Contrarian Strategy Using Skew Oscillator

The contrarian skew trading strategy involves long or short positions in a single unit of stock, that is opposite to the sign of the skew,

Skew is calculated over one-minute bars, and trades are executed in the following period,

The contrarian strategy shows good hypothetical performance before transaction costs, and since it's a liquidity providing strategy, should have very low transaction costs,

The contrarian strategy is hyperactive, trading almost 46% of the time in each period,




```
> # lag the skew to get positions
> position_s <- -sign(lag_skew)
> position_s[1, ] <- 0
> # cumulative PnL
> cumu_pnl <- cumsum(position_s*re_turns[, 1])
> # calculate frequency of trades
> 50*sum(abs(sign(sk_ew)-sign(lag_skew)))/nrow(s
> # calculate transaction costs
> bid_offer <- 0.001  # 10 bps for liquid ETFs
> bid_offer*sum(abs(sign(sk_ew)-sign(lag_skew)))
```

```
> chart_Series(
+   cumu_pnl[endpoints(cumu_pnl, on="hours"), ],
+   name=paste(sym_bol, "contrarian skew strateg
```