Package 'HighFreq'

May 29, 2015

Type Package

Title High Frequency Data Management

Version 0.1

Date 2015-05-28

Author Jerzy Pawlowski

Maintainer Jerzy Pawlowski <algoquant@algoquants.ch>

Description Functions for chaining and joining time series, scrubbing bad data, managing time zones and alligning time indices, converting TAQ data to OHLC format, aggregating data to lower frequency, estimating volatility, skew, and higher moments.

License GPL (>= 2)

Depends xts, quantmod, TTR, caTools, lubridate

Repository GitHub

URL https://github.com/algoquant/HighFreq

R topics documented:

HighFreq-package *HighFreq R package*

Description

Package 'HighFreq' contains functions for managing high frequency TAQ and OHLC market data. The functions perform:

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

Package 'HighFreq' is inspired by the package 'highfrequency', and follows many of its conventions. For example, the high frequency data is assumed to be stored in separate directories for each symbol, containing multiple daily '*.RData' files, each file containing one day of TAQ data.

Package 'HighFreq' also follows many of the conventions of package 'quantmod', such as column names of OHLC objects.

The 'HighFreq' functions are designed for speed and reduced memory usage. Wherever possible they use vectorized functions from packages base, 'xts', 'quantmod', 'TTR', and 'caTools'.

Details

Package:	HighFreq
Type:	Package
Version:	0.1
Date:	2015-05-28
License:	GPL (>= 2)

~~ An overview of how to use the package, including the most important functions ~~

Author(s)

Jerzy Pawlowski Maintainer: Jerzy Pawlowski <algoquant@algoquants.ch>

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~ ~~ <pkg> ~~

Examples

```
~~ simple examples of the most important functions ~~
```

`calc_rets` *Calculate returns of either* `TAQ` *or* `OHLC` *data in* `xts` *format.*

Description

Calculate returns of either `TAQ` or `OHLC` data in `xts` format.

Usage

```
calc_rets(xts_data)
```

Arguments

`xts_data`	`xts` time series of `TAQ` or `OHLC` data.

Details

The function `calc_rets` calculates returns and binds them with volume data.

Value

`xts` time series of returns and volumes.

Examples

```
# create time index of one second intervals for a single day
in_dex <- seq(from=as.POSIXct("2015-02-09 09:30:00"), to=as.POSIXct("2015-02-09 16:00:00"), by="1 sec")
# create xts of random TAQ prices
x_ts <- xts(cumsum(rnorm(length(in_dex))), order.by=in_dex)
# create vector of random bid-offer prices
bid_offer <- abs(rnorm(length(in_dex)))/10
# create TAQ data using cbind
xts_data <- cbind(x_ts-bid_offer, x_ts+bid_offer)
# add Trade.Price
xts_data <- cbind(xts_data, x_ts+rnorm(length(in_dex))/10)
# add Volume
xts_data <- cbind(xts_data, sample(x=10*(2:18), size=length(in_dex), replace=TRUE))
colnames(xts_data) <- c("Bid.Price", "Ask.Price", "Trade.Price", "Volume")
xts_data <- calc_rets(xts_data)
```

do_call_rbind *Recursively 'rbind' a list of objects, such as xts time series.*

Description

Performs the same operation as `do.call(rbind, list_var)`, but using recursion, which is much faster and uses less memory. This is the same function as 'do.call.rbind' from package 'qmao'.

Usage

```
do_call_rbind(list_var)
```

Arguments

list_var	list of vectors, matrices, data frames, or time series.

Details

Calls lapply in a loop, each time binding neighboring elements and dividing the length of `list_var` by half. The result of performing `do_call_rbind(list_xts)` on a list of `xts` time series is identical to performing `do.call(rbind, list_xts)`. But `do.call(rbind, list_xts)` is very slow, and often causes an 'out of memory' error.

Value

single vector, matrix, data frame, or time series.

Examples

```
# create xts time series
x_ts <- xts(x=rnorm(1000), order.by=(Sys.time()-3600*(1:1000)))
# split time series into daily list
list_xts <- split(x_ts, "days")
# rbind the list back into a time series and compare with the original
identical(x_ts, do_call_rbind(list_xts))
```

extreme_values *Identify extreme values in a univariate xts time series.*

Description

Identifies extreme values as those that exceed a multiple of the running volatility.

Usage

```
extreme_values(time_series, vol_window = 51, vol_mult = 2)
```

Arguments

time_series	univariate xts time series.
vol_window	number of data points for estimating running volatility.
vol_mult	volatility multiplier.

Details

Calculates running volatility as a quantile of values over a sliding window. Extreme values are those that exceed the product of the volatility multiplier times the running volatility. Extreme values are the very tips of the tails when the distribution of values becomes very fat-tailed. The volatility multiplier vol_mult controls the threshold at which values are identified as extreme. Smaller volatility multiplier values will cause more values to be identified as extreme.

Value

logical vector.

Examples

```
# create xts time series
x_ts <- xts(x=rnorm(1000), order.by=(Sys.time()-3600*(1:1000)))
# scrub extreme values
x_ts <- x_ts[!extreme_values(x_ts, vol_mult=1)]
```

---

moment_ohlc *Calculate the moment of a* OHLC *time series.*

Description

Calculate the moment of a OHLC time series.

Usage

```
moment_ohlc(ohlc, mom_fun = "vol_ohlc", calc_method = "rogers.satchell",
  vo_lu = TRUE, ...)
```

Arguments

ohlc	OHLC time series of prices.
mom_fun	character string representing function for estimating the moment.
calc_method	character string representing method for estimating the moment.
vo_lu	logical should estimate be weighted by trade volume.

Details

Calculates a single number representing the estimate of the moment of a OHLC time series of prices. By default the estimate is trade volume weighted.

Value

numeric value with estimate of the moment.

Examples

```
# create time index of one minute intervals over several days
in_dex <- seq(from=as.POSIXct("2015-02-09 09:30:00"),
              to=as.POSIXct("2015-02-28 16:00:00"), by="1 min")
# create xts of random prices
x_ts <- xts(exp(cumsum(0.001*rnorm(length(in_dex)))), order.by=in_dex)
# add trade Volume data
x_ts <- merge(x_ts,
          volume=sample(x=10*(2:18),
            size=length(in_dex), replace=TRUE))
# aggregate to hours OHLC data
oh_lc <- to.period(x=x_ts, period="hours")
# calculate time series of daily skew estimates
sk_ew <- apply.daily(x=oh_lc, FUN=moment_ohlc, mom_fun="skew_ohlc")
```

`price_jumps` *Identify isolated price jumps in a univariate* `xts` *time series of prices, based on pairs of large neighboring returns of opposite sign.*

Description

Identify isolated price jumps in a univariate `xts` time series of prices, based on pairs of large neighboring returns of opposite sign.

Usage

```
price_jumps(time_series, vol_window = 51, vol_mult = 2)
```

Arguments

`time_series`	univariate `xts` time series of prices.
`vol_window`	number of data points for estimating running volatility.
`vol_mult`	volatility multiplier.

Details

Isolated price jumps are single prices that are very different from neighboring values. Price jumps create pairs of large neighboring returns of opposite sign. The function `price_jumps` first calculates simple returns from prices. Then it calculates the running volatility of returns as a quantile of returns over a sliding window. Jump prices are identified as those where neighboring returns both exceed a multiple of the running volatility, but the sum of those returns doesn't exceed it.

Value

`logical vector`.

Examples

```
# create xts time series
x_ts <- xts(x=rnorm(1000), order.by=(Sys.time()-3600*(1:1000)))
# scrub jump prices
x_ts <- x_ts[!price_jumps(x_ts, vol_mult=1.0)]
```

run_moment_ohlc *Calculate the running estimates over time of the moment of a* OHLC *time series.*

Description

Calculate the running estimates over time of the moment of a OHLC time series.

Usage

```
run_moment_ohlc(ohlc, mom_fun = "vol_ohlc", calc = "rogers.satchell",
  n = 20, N = 260, vo_lu = TRUE, ...)
```

Arguments

ohlc	OHLC time series of prices.
mom_fun	character string representing function for estimating the moment.
n	numeric number of periods for averaging of estimates.
N	numeric number of periods in a year (to annualize the estimates).
vo_lu	logical should estimate be weighted by trade volume.
calc_method	character string representing method for estimating the moment.

Details

Calculates a time series of running estimates of the moment of a OHLC time series of prices. By default the estimates are trade volume weighted.

Value

numeric time series of estimates of the moment.

Examples

```
# create time index of one minute intervals over several days
in_dex <- seq(from=as.POSIXct("2015-02-09 09:30:00"),
              to=as.POSIXct("2015-02-28 16:00:00"), by="1 min")
# create xts of random prices
x_ts <- xts(exp(cumsum(0.001*rnorm(length(in_dex)))), order.by=in_dex)
# add trade Volume data
x_ts <- merge(x_ts,
          volume=sample(x=10*(2:18),
            size=length(in_dex), replace=TRUE))
# aggregate to hours OHLC data
oh_lc <- to.period(x=x_ts, period="hours")
# calculate time series of running volatility and skew estimates
vol_at <- run_moment_ohlc(ohlc=oh_lc)
sk_ew <- run_moment_ohlc(ohlc=oh_lc, mom_fun="skew_ohlc")
sk_ew <- sk_ew/(vol_at)^(1.5)
sk_ew[1, ] <- 0
sk_ew <- na.locf(sk_ew)
```

save_rets — *Load, scrub, aggregate, and rbind multiple days of* TAQ *data for a single symbol. Calculate returns and save them to a single '*.RData*' file.*

Description

Load, scrub, aggregate, and rbind multiple days of TAQ data for a single symbol. Calculate returns and save them to a single '*.RData' file.

Usage

```
save_rets(sym_bol, data_dir = "E:/mktdata/sec/",
  output_dir = "E:/output/data/", vol_window = 51, vol_mult = 2,
  period = "minutes", tzone = "America/New_York")
```

Arguments

sym_bol	character string representing symbol or ticker.
data_dir	character string representing directory containing input '*.RData' files.
output_dir	character string representing directory containing output '*.RData' files.
vol_window	number of data points for estimating running volatility.
vol_mult	volatility multiplier.
period	aggregation period.
tzone	timezone to convert.

Details

The function save_rets loads multiple days of TAQ data, then scrubs, aggregates, and rbinds them into a OHLC time series. It then calculates returns using function calc_rets, and stores them in a variable named 'symbol.rets', and saves them to a file called 'symbol.rets.RData'. The TAQ data files are assumed to be stored in separate directories for each 'symbol'. Each 'symbol' has its own directory (named 'symbol') in the 'data_dir' directory. Each 'symbol' directory contains multiple daily '*.RData' files, each file containing one day of TAQ data.

Value

time series of returns and volume in xts format.

Examples

```
## Not run:
save_rets("SPY")

## End(Not run)
```

save_rets_ohlc *Load* OHLC *time series data for a single symbol, calculate its returns, and save them to a single '*.RData*' file, without aggregation.*

Description

Load OHLC time series data for a single symbol, calculate its returns, and save them to a single '*.RData' file, without aggregation.

Usage

```
save_rets_ohlc(sym_bol, data_dir = "E:/output/data/",
  output_dir = "E:/output/data/")
```

Arguments

sym_bol	character string representing symbol or ticker.
data_dir	character string representing directory containing input '*.RData' files.
output_dir	character string representing directory containing output '*.RData' files.

Details

The function save_rets_ohlc loads OHLC time series data from a single file. It then calculates returns using function calc_rets, and stores them in a variable named 'symbol.rets', and saves them to a file called 'symbol.rets.RData'.

Value

time series of returns and volume in xts format.

Examples

```
## Not run:
save_rets_ohlc("SPY")

## End(Not run)
```

save_scrub_agg *Load, scrub, aggregate, and rbind multiple days of* TAQ *data for a single symbol, and save the* OHLC *time series to a single '*.RData*' file.*

Description

Load, scrub, aggregate, and rbind multiple days of TAQ data for a single symbol, and save the OHLC time series to a single '*.RData' file.

Usage

```
save_scrub_agg(sym_bol, data_dir = "E:/mktdata/sec/",
  output_dir = "E:/output/data/", vol_window = 51, vol_mult = 2,
  period = "minutes", tzone = "America/New_York")
```

Arguments

sym_bol	character string representing symbol or ticker.
data_dir	character string representing directory containing input '*.RData' files.
output_dir	character string representing directory containing output '*.RData' files.
vol_window	number of data points for estimating running volatility.
vol_mult	volatility multiplier.
period	aggregation period.
tzone	timezone to convert.

Details

The function save_scrub_agg loads multiple days of TAQ data, then scrubs, aggregates, and rbinds them into a OHLC time series, and finally saves it to a single '*.RData' file. The OHLC time series is stored in a variable named 'symbol', and then it's saved to a file named 'symbol.RData' in the 'output_dir' directory. The TAQ data files are assumed to be stored in separate directories for each 'symbol'. Each 'symbol' has its own directory (named 'symbol') in the 'data_dir' directory. Each 'symbol' directory contains multiple daily '*.RData' files, each file containing one day of TAQ data.

Value

OHLC time series in xts format.

Examples

```
## Not run:
save_scrub_agg("SPY")

## End(Not run)
```

---

save_TAQ *Load and scrub multiple days of* TAQ *data for a single symbol, and save it to multiple '*.RData*' files.*

Description

Load and scrub multiple days of TAQ data for a single symbol, and save it to multiple '*.RData' files.

Usage

```
save_TAQ(sym_bol, data_dir = "E:/mktdata/sec/",
  output_dir = "E:/output/data/", vol_window = 51, vol_mult = 2,
  tzone = "America/New_York")
```

Arguments

sym_bol	character string representing symbol or ticker.
data_dir	character string representing directory containing input '*.RData' files.
output_dir	character string representing directory containing output '*.RData' files.
vol_window	number of data points for estimating running volatility.
vol_mult	volatility multiplier.
tzone	timezone to convert.

Details

The function save_TAQ loads multiple days of TAQ data, scrubs it, and saves it to '*.RData' files. It uses the same file names for output as the input file names. The TAQ data files are assumed to be stored in separate directories for each 'symbol'. Each 'symbol' has its own directory (named 'symbol') in the 'data_dir' directory. Each 'symbol' directory contains multiple daily '*.RData' files, each file containing one day of TAQ data.

Value

TAQ time series in xts format.

Examples

```
## Not run:
save_TAQ("SPY")

## End(Not run)
```

---

scrub_agg *Scrub a single day of* TAQ *data, aggregate it, and convert to* OHLC *format.*

Description

Scrub a single day of TAQ data, aggregate it, and convert to OHLC format.

Usage

```
scrub_agg(taq_data, vol_window = 51, vol_mult = 2, period = "minutes",
  tzone = "America/New_York")
```

Arguments

taq_data	TAQ xts time series.
vol_window	number of data points for estimating running volatility.
vol_mult	volatility multiplier.
period	aggregation period.
tzone	timezone to convert.

Details

The function scrub_agg performs:

- index timezone conversion,
- data subset to trading hours,
- removal of duplicate time stamps,
- scrubbing of quotes with suspect bid-offer spreads,
- scrubbing of quotes with suspect price jumps,
- cbinding of mid prices with volume data,
- aggregation to OHLC using function to.period from package xts,

Valid 'period' character strings include: "minutes", "3 min", "5 min", "10 min", "15 min", "30 min", and "hours". The time index of the output time series is rounded up to the next integer multiple of 'period'.

Value

OHLC time series in xts format.

Examples

```
# create time index of one second intervals for a single day
in_dex <- seq(from=as.POSIXct("2015-02-09 09:30:00"), to=as.POSIXct("2015-02-09 16:00:00"), by="1 sec")
# create xts of random TAQ prices
x_ts <- xts(cumsum(rnorm(length(in_dex))), order.by=in_dex)
# create vector of random bid-offer prices
bid_offer <- abs(rnorm(length(in_dex)))/10
# create TAQ data using cbind
taq_data <- cbind(x_ts-bid_offer, x_ts+bid_offer)
# add Trade.Price
taq_data <- cbind(taq_data, x_ts+rnorm(length(in_dex))/10)
# add Volume
taq_data <- cbind(taq_data, sample(x=10*(2:18), size=length(in_dex), replace=TRUE))
colnames(taq_data) <- c("Bid.Price", "Ask.Price", "Trade.Price", "Volume")
# aggregate to ten minutes OHLC data
ohlc_data <- scrub_agg(taq_data, period="10 min")
chartSeries(ohlc_data, name=sym_bol, theme=chartTheme("white"))
```

---

scrub_TAQ *Scrub a single day of* TAQ *data in* xts *format, without aggregation.*

Description

Scrub a single day of TAQ data in xts format, without aggregation.

Usage

```
scrub_TAQ(taq_data, vol_window = 51, vol_mult = 2,
  tzone = "America/New_York")
```

Arguments

taq_data	TAQ xts time series.
vol_window	number of data points for estimating running volatility.
vol_mult	volatility multiplier.
tzone	timezone to convert.

Details

The function scrub_TAQ performs the same scrubbing operations as scrub_agg, except it doesn't aggregate, and returns the TAQ data in xts format.

Value

TAQ xts time series.

Examples

```
# create time index of one second intervals for a single day
in_dex <- seq(from=as.POSIXct("2015-02-09 09:30:00"),
              to=as.POSIXct("2015-02-09 16:00:00"), by="1 sec")
# create xts of random TAQ prices
x_ts <- xts(cumsum(rnorm(length(in_dex))), order.by=in_dex)
# create vector of random bid-offer prices
bid_offer <- abs(rnorm(length(in_dex)))/10
# create TAQ data using cbind
taq_data <- cbind(x_ts-bid_offer, x_ts+bid_offer)
# add Trade.Price
taq_data <- cbind(taq_data, x_ts+rnorm(length(in_dex))/10)
# add Volume
taq_data <- cbind(taq_data, sample(x=10*(2:18), size=length(in_dex), replace=TRUE))
colnames(taq_data) <- c("Bid.Price", "Ask.Price", "Trade.Price", "Volume")
taq_data <- scrub_TAQ(taq_data)
taq_data <- scrub_TAQ(taq_data, vol_window=11, vol_mult=1)
```

---

skew_ohlc *Calculate time series of skew estimates from a* OHLC *time series.*

Description

Calculate time series of skew estimates from a OHLC time series.

Usage

```
skew_ohlc(log_ohlc, calc_method = "rogers.satchell")
```

Arguments

log_ohlc	OHLC time series of log prices.
calc_method	character string representing method for estimating skew.

Details

Calculates skew estimates from OHLC values at each point in time (row). The methods include Garman-Klass and Rogers-Satchell.

Value

time series of skew estimates.

Examples

```
# create time index of one second intervals for a single day
in_dex <- seq(from=as.POSIXct("2015-02-09 09:30:00"),
              to=as.POSIXct("2015-02-09 16:00:00"), by="1 sec")
# create xts of random prices
x_ts <- xts(cumsum(rnorm(length(in_dex))), order.by=in_dex)
# add Volume data
x_ts <- merge(x_ts,
          volume=sample(x=10*(2:18),
            size=length(in_dex), replace=TRUE))
# aggregate to minutes OHLC data
oh_lc <- to.period(x=x_ts, period="minutes")
# calculate skew estimates
sk_ew <- skew_ohlc(oh_lc)
```

---

vol_ohlc *Calculate time series of volatility estimates from a* OHLC *time series.*

Description

Calculate time series of volatility estimates from a OHLC time series.

Usage

```
vol_ohlc(log_ohlc, calc_method = "rogers.satchell")
```

Arguments

log_ohlc	OHLC time series of log prices.
calc_method	character string representing method for estimating volatility.

Details

Calculates volatility estimates from OHLC values at each point in time (row). The methods include Garman-Klass and Rogers-Satchell.

Value

time series of volatility estimates.

Examples

```
# create time index of one second intervals for a single day
in_dex <- seq(from=as.POSIXct("2015-02-09 09:30:00"),
              to=as.POSIXct("2015-02-09 16:00:00"), by="1 sec")
# create xts of random prices
x_ts <- xts(cumsum(rnorm(length(in_dex))), order.by=in_dex)
# add Volume data
x_ts <- merge(x_ts,
          volume=sample(x=10*(2:18),
            size=length(in_dex), replace=TRUE))
# aggregate to minutes OHLC data
oh_lc <- to.period(x=x_ts, period="minutes")
# calculate volatility estimates
vol_at <- vol_ohlc(oh_lc)
```

Index