Functions_Organized

```
library(highfrequency)
# https://cran.r-project.org/web/packages/highfrequency/highfrequency.pdf
```

Covariance estimation

- rCholCov()
  - positive semi-definite covariance estimation using the CholCov algorithm. Details on page 70-72 of the R Documentation.
- rCov()
  - returns the Realized Covariation
  - Let $r_{t,i}$ be an intraday NXM return vector and $i = 1, ..., M$ the number of intraday returns. Then $rCov_t = \sum_{i=1}^{M} r_{t,i} r'_{t,i}$
- ReMeDI()
  - estimates the auto-covariance of market-microstructure noise. Details on page 74-75 of the R Documentation.
  - ReMeDIAsymptoticVariance(): estimates the asymptotic variance of the ReMeDI estimator.
- rHYCov()
  - calculates the Hayashi-Yoshia covariance estimator. Details on Hayashi, T. and Yoshida, N. (2005). On covariance estimation of non-synchronously observed diffusion processes. Bernoulli, 11, 359-379.
- rKernelCov()
  - realized covariance calculation using a kernel estimator. Available kernels: rectangular, bartlett, second-order, epanechnikov, cubiic, fifth, sixth, seventh, eighth, parzen, tukeyhanning, modified-tukeyhanning. Details on page 80 of the R Documentation.
- rMPV()
  - calculates the realized multipower variation, as defined in Andersen, T. G., Dobrev, D., and Schaumburg, E. (2012). Jump-robust volatility estimation using nearest neighbor truncation. Journal of Econometrics, 169, 75-93. Details on page 90 of the R Documentation.
- rMRC()
  - modulated realized covariance. Calculates the univariate or multivariate pre-averaged estimator. Details on page 91 of the R Documentation.
- rOWCov()
  - calculates the realized outlyingness weighted covariance, defined in Boudt, K., Croux, C., and Laurent, S. (2008). Outlyingness weighted covariation. Journal of Financial Econometrics, 9, 657–684. Details on page 95.
- rQPVar()

  - realized quad-power variation of intraday returns. Details on page 97.

- rRTSCov()
  - calculates the robust two time scale covariance matrix. Details on page 99-101.
- rSemiCov()
  - calculates the realized semicovariances. Details on page 102.
- rSV()
  - calculates the realized semivariances of highfrequency return series with two outcomes: downside realized semivariance and upside realized semivariance. Details on page 105.
- rThresholdCov()
  - calculates the threshold covariance matrix. Details on page 106.
- rTSCov()
  - calculates the two time scale covariance matrix. By the use of two time scales, this covariance estimate is robust to microstructure noise and non-synchronic trading.
- RV()
  - an estimator of realized variance.
- rAVGCov()
  - Calculates realized variance by averaging across partially overlapping grids.
- rBPCov()
  - Calculate the Realized BiPoweer Covariance defined by Barndorff-Nielsen and Shephard.

Other

- refreshTime()
  - implements the refresh time synchronization scheme proposed in: Harris, F., T. McInish, Shoesmith, G., and Wood, R. (1995). Cointegration, error correction, and price discovery on informationally linked security markets. Journal of Financial and Quantitative Analysis, 30, 563-581. It picks the so-called refresh times at which all assets have traded at least once since the last refresh time point. For example, the first refresh time corresponds to the first time at which all stocks have traded. The subsequent refresh time is defined as the first time when all stocks have traded again. This process is repeated until the end of one time series is reached.

Other Estimations

- rKurt()
  - calculates the realized kurtosis as defined in Amaya, D., Christoffersen, P., Jacobs, K., and Vasquez, A. (2015). Does realized skewness and kurtosis predict the cross-section of equity returns? Journal of Financial Economics, 118, 135-167.
- rMedRQ()
  - an estimator of integrated quarticity from applying the median operator on blocks of three returns. Defined in Andersen, T. G., Dobrev, D., and Schaumburg, E. (2012). Jump-robust volatility estimation using nearest neighbor truncation. Journal of Econometrics, 169, 75-93. Details on page 82 of the R Documentation.

- rMedRV()
  - Defined in Andersen, T. G., Dobrev, D., and Schaumburg, E. (2012). Jump-robust volatility estimation using nearest neighbor truncation. Journal of Econometrics, 169, 75-93.
- rMinRQ()
  - an estimator of integrated quarticity from applying the minimum operator on blocks of two returns. Defined in Andersen, T. G., Dobrev, D., and Schaumburg, E. (2012). Jump-robust volatility estimation using nearest neighbor truncation. Journal of Econometrics, 169, 75-93. Details on page 85 of the R Documentation.
- rMinRV()
  - Defined in Andersen, T. G., Dobrev, D., and Schaumburg, E. (2012). Jump-robust volatility estimation using nearest neighbor truncation. Journal of Econometrics, 169, 75-93.
- rQuar()
  - calculates the realized quarticity. Details on page 98.
- rSkew()
  - calculates the realized skewness. Details on page 104.
- rTPQuar()
  - calculates the realized tri-power quarticity. Details on page 108.
- spotDrift()
  - funnction used to estiamte the spot drift of intraday stock prices/returns. Details on page 118-119.
- spotVol()
  - estimates a wide variety of spot volatility estimators. Details on page 121-126.
- AJjumpTest()
  - Ait-Sahalia and Jacod test for the presence of jumps in the price series.
- BNSjumpTest()
  - Barndorff-Nielsen and Shephard tests for the presence of jumps in the price series. Null hypothesis: there are no jumps. * driftBursts()
  - This function will return the result of testing drift burst hypothesis and also shows the test statistics.
- getCriticalValues()
  - get critical values for drift burst hypothesis
- getLiquidityMeasures()
  - Compute Liquidity Measures
- intradayJumpTest()
  - This can be used to test for jumps in intraday price paths.
- IVinference()
  - This function returns the SE, value and confidence band of Integrated variance estimator.
- JOjumpTest()
  - Test for jumps in the price series by using Jiang and Oomen test.
- makePsd()
  - this function can return the positive semidefinite projection of a symmetric matrix using the eigenvalue method.
- rankJumpTest()
  - Calculate the rank jump test of Li et al.

Data Cleaning Functions

- rmLargeSpread()
  - deletes entires for which the spread is more than "maxi" times the median spread on that day.
- rmNegativeSpread()
  - deletes entries for which the spread is negative.
- rmOutlierQuotes()
  - deletes entries for which the mid-quote is outlying with respect to surrounding entries. Details on page 89 of the R Documentation.
- rmTradeOutliersUsingQuotes()
  - deletes entries with prices that are above the ask plus the bid-ask spread. Similar for prices below the bid minus the bid-ask spread.
- selectExchange()
  - filter raw trade data to only contain specified exchanges
- tradesCleanup()
  - this function is a wrapper function for cleaning the trade data. It must contain the following columns: DT2, exchange code, SYMBOL, PRICE, SIZE ,BID
- quotesCleanup()
  - this function is a wrapper function for cleaning the quotes data. It must contain the following columns: DT2, SYMBOL, EX, BID, BIDSIZ, OFR, OFRSIZ, PRICE
- tradesCleanupUsingQuotes()
- tradesCondition()
  - deletes entries with abnormal trades condition
- aggregatePrice()
  - Aggregate a times series but keep first and last observations.
- aggregateQuotes()
  - Aggregate a quote data in a xts format
- aggregateTrades()
  - Aggregate a trade data in a xts format
- aggregateTS()
  - Aggregate a time series, it did pretty much the same thing as the aggregatePrice.
- autoSelectExchangeQuotes() *Only return the data from the stock exchange with the highest volume in quote data
- autoSelectExchangeTrades()
  - Only return the data from the stock exchange with the highest trading volume in trade data
- businessTimeAggregation()
  - Aggregation function based on business time.
- exchangeHoursOnly()
  - This function is used for extracting data from an xts object for the exchange hours only.
- makeOHLCV()

  - this function is a kind of aggregation function that can make the high frequency data become OHLCV data by the time range that we set.

- makeRMFormat()
  - this function is used for spliting data to a format which can be used for realized measure.
- matchTradesQuotes()
  - this function can match trade data and quote data and combine them.
- mergeQuotesSameTimestamp()
- mergeTradesSameTimestamp()
  - these functions aggregate the quote/trade data that have the same timestamp.

Building Models

- getTradeDirection()
  - Using Lee and Ready algorithm to determine the inferred trade direction.
- HARmodel()
  - This function returns the estimates for the HAR model for realized volatility.
- HEAVYmodel()
  - This functions calculate HEAVY model which is introduced by Shepard and Sheppard.