Package ‘lqtool’

May 1, 2017

Type Package

Title Factor Backtesting and Charting Functions

Version 0.1.0

Author Luos Quant

Description Package provides basic functions to perform simple backtesting and data checking. Univariate backtest charts R code to produce graphs for LBacktester More sample codes can be found in package gcookbook Main reference book: R Graphics Cookbook, by Winston Chang

Depends R (>= 3.3.0),
parallel(>= 3.3.0),
MASS (>= 7.3-45),
nlme,
ggplot2,
reshape2,
scales,
RColorBrewer,
corpcor,
tseries,
quadprog,
FRAPO,
Rglpk,
grid,
openxlsx,
wquantR,
aws.s3,
hashids,
BBmisc,
stringr,
randomForest

BugReports `https://wolferesearch.atlassian.net/projects/WQUAN/issues`

License Proprietary

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

R topics documented:

ADL *Accumulation Distribution Line*

Description

Accumulation Distribution Line

Usage

```
ADL(closeprice, highprice, lowprice, cashvol, clv, n = 20)
```

Arguments

`cashvol`	cash volume
`clv`	Close Location Value, if not provided will compute from price
`n`	

ADX *Average Directional Index*

Description

Average Directional Index

Usage

```
ADX(closeprice, lowprice, highprice, n = 14)
```

Arguments

`n`

AO *Aroon Oscillator*

Description

Aroon Oscillator

Usage

```
AO(highprice, lowprice, n = 25)
```

Arguments

`n`

appendForwardReturn *Computes forward returns and appends to the list of data matrices*

Description

Computes forward returns and appends to the list of data matrices

Usage

```
appendForwardReturn(data, name, period)
```

Arguments

data	List of matrices
name	Name of the forward return
period	period over which forward return is to be computed

Examples

```
price=matrix(c(100,101,50,51),nrow=2,byrow = TRUE)
div=matrix(c(10,11,3,3),nrow=2,byrow = TRUE)
appendForwardReturn(list(PRCCD=price,CUM_DIV=div),'FRTN1M',1)
```

ATR *Average True Range*

Description

Average True Range

Usage

```
ATR(closeprice, lowprice, highprice, n = 14)
```

Arguments

n

`awss3.def.s3.bucket.hadoop` *Returns the name of defaul hadoop bucket*

Description

Returns the name of defaul hadoop bucket

Usage

```
awss3.def.s3.bucket.hadoop()
```

Value

Name of default hadoop bucket

`awss3.def.s3.bucket.rdata` *Returns the name of the default Rdata bucket*

Description

Returns the name of the default Rdata bucket

Usage

```
awss3.def.s3.bucket.rdata()
```

Value

Name of default rdata bucket

`awss3.def.s3.delim` *Delimiter used in Hadoop Data Output*

Description

Delimiter used in Hadoop Data Output

Usage

```
awss3.def.s3.delim()
```

awss3.files *Lists files in a bucket and folder*

Description

Lists files in a bucket and folder

Usage

```
awss3.files(bucket, folder)
```

Arguments

bucket	top level bucket name
folder	sub folder

Examples

```
awss3.files('lquant-hadoop-out','edgar-2004')
```

awss3.getdf *Gets data frame from hadoop folder*

Description

Gets data frame from hadoop folder

Usage

```
awss3.getdf(file, folder)
```

Arguments

folder

Examples

```
txtdata<-awss3.getdf('edgar-2004/part-r-00013', 'edgar-2004')
```

`awss3.getobject` *Deserializes object to data frame*

Description

Deserializes object to data frame

Usage

```
awss3.getobject(obj)
```

Arguments

`obj`	S3 Object

Value

Data Frame

`awss3.ismeta` *Check if the file in the hadoop directory is a meta file*

Description

Check if the file in the hadoop directory is a meta file

Usage

```
awss3.ismeta(obj)
```

Arguments

`obj`

Value

TRUE if file is a meta file

`awss3.loadobject` *Loads object from S3 sub folder into a data frame*

Description

Loads object from S3 sub folder into a data frame

Usage

```
awss3.loadobject(folder)
```

Arguments

`folder`	Name of the folder to read data from

Examples

```
txtdata<-awss3.loadobject('edgar-2004')
```

`awss3.ls` *Lists down file in the hadoop bucket folder*

Description

Lists down file in the hadoop bucket folder

Usage

```
awss3.ls(folder)
```

Arguments

`folder`	Name of the folder

Examples

```
txtdata<-awss3.ls('edgar-2002')
```

`awss3.makedf` *Makes data frame from byte array [Internal Use Only]*

Description

Makes data frame from byte array [Internal Use Only]

Usage

```
awss3.makedf(x)
```

Arguments

`x`	byte array

`awss3.rdata.list` *Lists r data files in the RData bucket*

Description

Lists r data files in the RData bucket

Usage

```
awss3.rdata.list()
```

`awss3.rdata.load` *Loads object from RData file*

Description

Loads object from RData file

Usage

```
awss3.rdata.load(name)
```

Arguments

`name`	Name of the file

`awss3.rdata.save` *Saves a file onto to S3 RData bucket*

Description

Saves a file onto to S3 RData bucket

Usage

```
awss3.rdata.save(..., name)
```

Arguments

`name`	name of the file
`data`	object to store

backtest.Basic *basic backtest*

Description

Performs basic backtesting for a factor

Usage

```
backtest.Basic(factor, frtn, qnum = 5, classMatrix = NULL, tCost = 0,
  rebalanceFreq = "M", outputFreq = NULL, periodEnd = TRUE, skip = 0,
  startThresh = 0.6)
```

Arguments

`factor`	Factor Matrix
`frtn`	Next period return, can be different frequency as the factor
`qnum`	Number of Bins for Long/Short basket
`classMatrix`	Neutralize by categorical factor such as sector/country with the same size matrix as factor
`tCost`	Transaction cost for the long/short quantile backtest
`rebalanceFreq`	default is Monthly
`outputFreq`	Output frequency, can be 'A', 'Q', 'M', 'W', 'D' stands for annual, quarterly, monthly, weekly and daily. If outputFreq = NULL output same frequency as factor and rebalance frequency is the factor frequency
`periodEnd`	Boolean Flag, if to start at period end (week end, month end, quarter end, year end), if periodEnd =FALSE, means start at the first period of the factor
`skip`	Number of periods to skip after the period specified by the periodEnd
`startThresh`	Start backtest when the coverage is greater than this threshold

backtest.getCAGR *getCAGR*

Description

Computes Wealth Curve

Usage

```
backtest.getCAGR(rtn)
```

Arguments

`rtn`	Input the return time serious or the matrix each row is a return time series, can be different frequency

`backtest.getCoverage` *getCoverage*

Description

Get Coverage of the factor

Usage

```
backtest.getCoverage(factor)
```

Arguments

`factor`	Factor Matrix usually returned by LQuant

`backtest.getDailyReturns` *getDailyReturns*

Description

Gets Daily Returns of a portfolio with weighting

Usage

```
backtest.getDailyReturns(weightList, identifier, dailyPrice, dailyCumDiv,
  endDate = NULL)
```

Arguments

`weightList`	A list of portfolio weights, each element in the list is the vector of the weights, the names of the vector is the identifier, and the names of the weightList is the rebalance dates
`identifier`	Matrix of the identifiers, colnames is the dates, can be any frequency
`dailyCumDiv`	daily cumulative dividend matrix, same dimension as dailyPrice
`endDate`	End date for the daily return backtesting
`dailyPrices`	daily PRCCD matrix, same number of rows as identifier

backtest.getHitRate *getHitRate*

Description

Computes Hit Rate for a factor

Usage

```
backtest.getHitRate(factor, frtn, cumulative = TRUE, period = 12)
```

Arguments

`factor`	Factor Matrix
`frtn`	Next period return, can be different frequency as the factor
`cumulative`	Boolean flag, if backtest for the cumulative hit rate
`period`	The number of period of hit rate the function output

backtest.getICDecay *getICDecay*

Description

Get Information Coefficient Decay function

Usage

```
backtest.getICDecay(factor, frtn, cumulative = TRUE, period = 12)
```

Arguments

`factor`	Factor Matrix
`frtn`	Next period return, can be different frequency as the factor
`cumulative`	Boolean flag, if backtest for the cumulative IC Decay
`period`	The number of period of IC decay the function output

`backtest.getIR` *getIR*

Description

Computes Information Ratio of monthly return series

Usage

```
backtest.getIR(rtn)
```

Arguments

`rtn`	Input the return time serious or the matrix each row is a return time series, can be different frequency

`backtest.getMaxDD` *getMaxDD*

Description

Computes Maximum Drawdown

Usage

```
backtest.getMaxDD(rtn)
```

Arguments

`rtn`	Input the return time serious or the matrix each row is a return time series

`backtest.getRankIC` *Get Rank IC Computes Rank Information Coefficient*

Description

Get Rank IC Computes Rank Information Coefficient

Usage

```
backtest.getRankIC(factor, frtn, classMatrix = NULL, method = "z_normal",
  ICType = "Spearman")
```

Arguments

`factor` Factor matrix as returned by LQuant Data Broker

`frtn` Forward Return matrix, can be different frequency as the factor

`classMatrix` Neutralize by categorical factor such as sector/country with the same size matrix as factor

`method` neutralize method, by default is the z normal

`ICType` By default use Spearman IC

Examples

```
factor<-wq.getdata(wq.newRequest()$runFor('i:006066.01')$from('2014-01-21')$to('2015-08-21')$at('1m')$a('CS
frtn<-wq.getdata(wq.newRequest()$runFor('i:006066.01')$from('2014-01-21')$to('2015-08-21')$at('1m')$a('RTN1
rankIC<-backtest.getRankIC(factor,lag(frtn,-1));
```

---

backtest.getReturns *getReturns*

Description

Computes Returns for a factor

Usage

```
backtest.getReturns(factor, frtn, qnum = 5, classMatrix = NULL,
  outputFreq = NULL, periodEnd = TRUE, skip = 0, cap = NULL)
```

Arguments

`factor` Factor Matrix usually returned by LQuant

`frtn` Next period return, can be different frequency as the factor

`qnum` Number of Baskets (default 5)

`classMatrix` Neutralize by categorical factor such as sector/country with the same size matrix as factor

`outputFreq` Output frequency, can be ‘A’, ‘Q’, ‘M’, ‘W’, ‘D’ stands for annual, quarterly, monthly, weekly and daily. If outputFreq = NULL output same frequency as factor and rebalance frequency is the factor frequency

`periodEnd` If to start at period end (week end, month end, quarter end, year end), if periodEnd =FALSE, means start at the first period of the factor

`skip` Number of periods to skip after the period specified by the periodEnd

`cap` Market cap weight, if cap = NULL, is equally weighted long/short quantile

`backtest.getSC` *getSC*

Description

Computes Serial Correlation of the Factor

Usage

```
backtest.getSC(factor)
```

Arguments

`factor`	Factor Matrix, usually returned by LQuant

`backtest.getStats` *getStats*

Description

Computes timeseries statistics

Usage

```
backtest.getStats(series)
```

Arguments

`series`	Input the time series to calculate the stats

`backtest.getTurnover` *getTurnover*

Description

Computes turnover of a factor based on the number of bins

Usage

```
backtest.getTurnover(factor, qnum = 5, classMatrix = NULL)
```

Arguments

`factor`	Factor Matrix
`qnum`	Number of Bins for Long/Short basket
`classMatrix`	Neutralize by categorical factor such as sector/country with the same size matrix as factor

`backtest.getVol` *getVol*

Description

Computes volatility of a return time series

Usage

```
backtest.getVol(rtn)
```

Arguments

`rtn`	Input the return time serious or the matrix each row is a return time series, can be different frequency

`backtest.getWealth` *getWealth*

Description

Computes wealth curve from monthly return timeseries

Usage

```
backtest.getWealth(rtn)
```

Arguments

`rtn`	Input the return time serious or the matrix each row is a return time series

`backtest.rtnFromWealth` *rtnFromWealth*

Description

Computes monthly returns from Wealth Curve

Usage

```
backtest.rtnFromWealth(wealth)
```

Arguments

`wealth`	Time series of wealth curve or the matrix each row is a wealth time series

`backtest.turnoverFromHolding` *turnover*

Description

Computes turnover from Holdings

Usage

```
backtest.turnoverFromHolding(holdings)
```

Arguments

`holidings`	Input holdings, in matrix format

`basic.backFill` *backFill*

Description

Performs backfill on the factor matrix.

Usage

```
basic.backFill(x)
```

Arguments

`x`	Factor Matrix

`basic.calculateBeta` *CalculateBeta*

Description

Calculates Beta of a return series

Usage

```
basic.calculateBeta(returns, idx, mkt = NULL, classMatrix = NULL,
  outputFreq = "M", windowSize = 12, k = 5, universe = TRUE)
```

Arguments

`returns`	returns time series
`idx`	T/F matrix same size as returns, indicating if it's in the universe
`mkt`	market return, if =NULL use average return of the universe
`classMatrix`	Neutralize by categorical factor such as sector/country with the same size matrix as factor, if NULL means for the entire universe
`outputFreq`	output frequency
`windowSize`	window size to calculate beta
`k`	how many standard devation to remove for the outliers
`universe`	Boolean Flag, calculate beta for only the universe

`basic.frequencyConvert` *frequencyConvert*

Description

Convert the dates to the specified output frequency

Usage

```
basic.frequencyConvert(factorDates, outputFreq = "M", periodEnd = TRUE,
  skip = 0)
```

Arguments

`factorDates`	dates vector, 'yyyy-mm-dd' format
`outputFreq`	can be 'A' annual, 'Q' quarterly, 'M' monthly, 'W' weekly and 'D' daily
`periodEnd`	If to start at period end (week end, month end, quarter end, year end), if periodEnd =FALSE, means start at the first period of the factor
`skip`	Number of periods to skip after the period specified by the periodEnd

`basic.getFreq` *getFreq*

Description

Estimates Frequency of the time series data based on the dates. Output list of two elements: period, how many period per year for this frequency; freq, frequency of: 'A' annual, 'Q' quarterly, 'M' monthly, 'W' weekly and 'D' daily

Usage

```
basic.getFreq(inputDates)
```

Arguments

`inputDate`	date array, 'yyyy-mm-dd' format

basic.neutralizeFactor *Neutralize Factor*

Description

Neutralizes factor matrix

Usage

```
basic.neutralizeFactor(factor, method = c("mean", "median", "z_score",
  "z_normal", "percentile", "ranking"), classMatrix = NULL,
  winsorizeRatio = NULL)
```

Arguments

`factor`	Factor matrix to winsorize
`method`	Neutralize method
`classMatrix`	Neutralize by categorical factor such as sector/country with the same size matrix as factor, if NULL means for the entire universe

basic.quantileMatrix *quantileMatrix*

Description

Computes quantile matrix, return same size matrix as factor, with integers 1 to qnum indicating the quantile cross-sectionally each period

Usage

```
basic.quantileMatrix(factor, qnum = 5, classMatrix = NULL)
```

Arguments

`factor`	Factor Matrix
`qnum`	Number of Bins default 5
`classMatrix`	Neutralize by categorical factor such as sector/country with the same size matrix as factor, if NULL means for the entire universe

`basic.quantileMatrixZero` *quantileMatrixZero*

Description

Computes Quantile Matrix, return same size matrix as factor, with integers 1 to qnum indicating the quantile cross-sectionally each period, all the 0 value will be the first quantile

Usage

```
basic.quantileMatrixZero(factor, qnum = 5, classMatrix = NULL)
```

Arguments

`factor`	Factor Matrix
`qnum`	Number of Bins default 5
`classMatrix`	Neutralize by categorical factor such as sector/country with the same size matrix as factor, if NULL means for the entire universe

`basic.regressFactors` *Regress Factors*

Description

Performs multi-variate regression on Y vectors vs xList feature matrix, return the list of the residuals, coeffcients, and tstats when regress y against the factors in xList, for each point in time cross-sectionally

Usage

```
basic.regressFactors(y, xList, method = NULL, classMatrix = NULL,
  stepwise = FALSE, regMethod = "ols")
```

Arguments

`y`	Dependent variable observations, stored as matrix
`xList`	Independent Variable observations, stored as the list of matrices
`method`	Neutralization method
`classMatrix`	Neutralize by categorical factor such as sector/country with the same size matrix as factor, if NULL means for the entire universe
`stepwise`	Boolean Flag, if to use stepwise regression
`regMethod`	regMethod can be ’ols’, ’gls’, ’rlm’ or ’lad’, by default use ’ols’

`basic.regressFactorsFast` *regressFactorsFact*

Description

Fast regression

Usage

```
basic.regressFactorsFast(y, x, method = NULL, classMatrix = NULL)
```

Arguments

`y`	dependent variable observations
`x`	matrix of independent variables observations
`classMatrix`	

Value

Regression coefficients

`basic.removeOutliers` *removeOutliers*

Description

Removes outlier for a data frame

Usage

```
basic.removeOutliers(inputData, k = 3, setNA = TRUE, logScale = FALSE)
```

Arguments

`inputData`	vector or matrix
`k`	how many standard devation to remove
`setNA`	Boolean Flag, set NA for the outliers if TRUE, otherwise set the value at boundary
`logScale`	Boolean Flag, log scale the factor (such as market cap)

basic.timeSeriesNorm *timeSeriesNorm*

Description

Computes time series norm

Usage

```
basic.timeSeriesNorm(series, windowSize = 12, expending = TRUE)
```

Arguments

series	time series to normalize
windowSize	normalize window size

BollingerBands *Bollinger Bands*

Description

Bollinger Bands

Usage

```
BollingerBands(pricedata, n = 14)
```

Arguments

pricedata	typically input closeprice
n	window for the Bollinger Bands

Examples

```
BollingerBands(pricedata,28);
```

calculateGrowthRate *Growth Rate*

Description

Growth Rate

Usage

```
calculateGrowthRate(factorList)
```

Arguments

factorList	

calculateMA *Calculate Moving Average*

Description

Calculate Moving Average

Usage

```
calculateMA(factor_mat, windowSize = 60, minSize = windowSize * 0.75,
  numCores = 8)
```

Arguments

`numCores`

calculateTrend *Calculate Trend*

Description

Calculate Trend

Usage

```
calculateTrend(factor_mat, windowSize = 12)
```

Arguments

`windowSize`

CCI *Commodity Channel Index*

Description

Commodity Channel Index

Usage

```
CCI(closeprice, lowprice, highprice, n = 20)
```

Arguments

`n`

`CloseLocation` *Close Location Value*

Description

Close Location Value

Usage

```
CloseLocation(closeprice, highprice, lowprice)
```

Arguments

`lowprice`	

`CMF` *Chaikin Money Flow*

Description

Chaikin Money Flow

Usage

```
CMF(closeprice, highprice, lowprice, cashvol, clv, n = 20)
```

Arguments

`cashvol`	cash volume
`clv`	Close Location Value, if not provided will compute from price
`n`	

`CO` *Chaikin Oscillator*

Description

Chaikin Oscillator

Usage

```
CO(closeprice, highprice, lowprice, cashvol, clv, n = 20, n2 = 10, n3 = 3)
```

Arguments

`cashvol`	cash volume
`clv`	Close Location Value, if not provided will compute from price
`n`	long window
`n2`	short window
`n3`	average window

`concentration_ratio` *Concentration Ratio*

Description

Concentration Ratio

Usage

```
concentration_ratio(Port_Weights, Covariance_list)
```

Arguments

`Covariance_list`

`cvarOpt` *Title*

Description

Title

Usage

```
cvarOpt(rmat, alpha = 0.05, rmin = 0, wmin = 0, wmax = 1,
  weight.sum = 1)
```

Arguments

`weight.sum`

`diversification_ratio` *Diversification Ratio*

Description

Diversification Ratio

Usage

```
diversification_ratio(Port_Weights, Covariance_list)
```

Arguments

`Covariance_list`

DPO *Detrended Price Oscillator*

Description

Detrended Price Oscillator

Usage

```
DPO(pricedata, n = 20)
```

Arguments

`n`

EMA *EMA*

Description

EMA

Usage

```
EMA(values, n = 3, wilder = F)
```

Arguments

`wilder`

EMV *Ease of Movement*

Description

Ease of Movement

Usage

```
EMV(highprice, lowprice, cashvol, n = 14)
```

Arguments

`cashvol`	cash volume
`n`	

`EventStudyPlots` *EventStudyPlots*

Description

EventStudyPlots

Usage

```
EventStudyPlots(eventDts, returns, benchmark_index = NULL,
  benchmark_weights = NULL, period = 21, plot_title = "Event study",
  begingAtT0 = FALSE, method = "mean")
```

Arguments

`method`	

`FI` *Force Index*

Description

Force Index

Usage

```
FI(closeprice, cashvol, n = 14)
```

Arguments

`cashvol`	cash volume
`n`	

`forwardReturn` *Computes Forward Return using the price and dividend time series*

Description

Computes Forward Return using the price and dividend time series

Usage

```
forwardReturn(price, div, period)
```

Arguments

`price`	Matrix of adjusted price series
`div`	Cumulative adjusted dividend series
`period`	Period over which forward return is desired

Examples

```
price=matrix(c(100,101,50,51),nrow=2,byrow = TRUE)
div=matrix(c(10,11,3,3),nrow=2,byrow = TRUE)
forwardReturn(price,div,1)
```

`lqtool.leap.run` *Runs Leap Model*

Description

Runs Leap Model

Usage

```
lqtool.leap.run(directory, region, modeldir, lag_I, endDate, outdir,
  outdir2 = NA)
```

Arguments

`directory`	Source director of the factor data
`region`	Region (e.g., AXJ, JP)
`modeldir`	LEAP Model Parameters are stored
`lag_I`	Lag (e.g., 1)
`endDate`	Date for which to run LEAP model
`outdir`	Output directory where to store leap

`lqtool.output.identifiers` *List of identifiers that should be appended to the output file*

Description

List of identifiers that should be appended to the output file

Usage

```
lqtool.output.identifiers(includeCusip)
```

Arguments

`includeCusip`	

`lqtool.process.corefiles` *Returns list of core file prefixes*

Description

Returns list of core file prefixes

Usage

```
lqtool.process.corefiles()
```

`lqtool.process.countriesDone` *Check if all countries are done for a date*

Description

Check if all countries are done for a date

Usage

```
lqtool.process.countriesDone(directory, countries, endDate)
```

Arguments

`endDate`

`lqtool.process.createRegionFactorData` *Combines country level data and*

Description

Combines country level data and

Usage

```
lqtool.process.createRegionFactorData(directory, region, endDate)
```

Arguments

`endDate`

lqtool.process.download

Downloads Standard Factor Data

Description

Downloads Standard Factor Data

Usage

```
lqtool.process.download(directory, country, batchSize = 5, threads = 10)
```

Arguments

`directory`	where to download
`country`	which country
`batchSize`	Batch size for download default 5
`threads`	Number of thread

Examples

```
lqtool.process.download('/mnt/ebs1/data/d1','India',5,10)
```

lqtool.process.download.fn

Download Function

Description

Download Function

Usage

```
lqtool.process.download.fn(startDate1, endDate, freq, batchSize = 5,
  threads = 10, useNew = T)
```

Arguments

`useNew`

lqtool.process.excludefactors

List of factors to be excluded

Description

List of factors to be excluded

Usage

```
lqtool.process.excludefactors()
```

lqtool.process.filename *Returns file name for the argument File names are standardized*

Description

Returns file name for the argument File names are standardized

Usage

```
lqtool.process.filename(directory, prefix, country, endDate)
```

Arguments

`directory`	Base Directory
`prefix`	File Prefix
`country`	Country or Region
`endDate`	Date

lqtool.process.filterfactors *Filters out the list of factors*

Description

Filters out the list of factors

Usage

```
lqtool.process.filterfactors(data, exclude)
```

Arguments

`data`	LQuant Data Matrix
`exclude`	List of factors to be excluded

lqtool.process.hour *Return current hour based on the time zone*

Description

Return current hour based on the time zone

Usage

```
lqtool.process.hour(tz)
```

Arguments

`tz`	Time zone (e.g., America/New_York)

`lqtool.process.loadCountryFactors` *Loads*

Description

Loads

Usage

```
lqtool.process.loadCountryFactors(directory, country, endDate)
```

Arguments

`endDate`

`lqtool.process.mask` *Masks the data based on the binary flag*

Description

Masks the data based on the binary flag

Usage

```
lqtool.process.mask(data, mask)
```

Arguments

`data`	LQuant Data Matrix
`mask`	Mask (i.e., Boolean variable)

`lqtool.process.merge` *Merge a list of matrices into a single matrix*

Description

Merge a list of matrices into a single matrix

Usage

```
lqtool.process.merge(listData)
```

Arguments

`listData`

`lqtool.process.monthsBack` *Returns months back*

Description

Returns months back

Usage

```
lqtool.process.monthsBack(date, n)
```

Arguments

`n`

`lqtool.process.neutralizeRegionData` *Neutralizes Regional Factor Data*

Description

Neutralizes Regional Factor Data

Usage

```
lqtool.process.neutralizeRegionData(directory, region, endDate)
```

Arguments

`endDate`

`lqtool.process.regionDone` *Checks if the region download is complete*

Description

Checks if the region download is complete

Usage

```
lqtool.process.regionDone(directory, region, endDate)
```

Arguments

`endDate`

`lqtool.process.regionLoadAndNormalize` *Loads the regional data and writes the normalized z score to CSV Only last record of the file is outputted to CSV*

Description

Loads the regional data and writes the normalized z score to CSV Only last record of the file is outputted to CSV

Usage

```
lqtool.process.regionLoadAndNormalize(source, dest, region, date)
```

Arguments

`source`	Where raw factors are store
`dest`	Location where they should be written
`region`	Name of the region
`date`	Date of the file

`lqtool.process.regionNormalize` *Normalize and stores the data in output location*

Description

Normalize and stores the data in output location

Usage

```
lqtool.process.regionNormalize(factor_data, basic_factor, dest, region)
```

Arguments

`factor_data`	LQuant matrix for all factors
`basic_factor`	LQuant basic factor matrix
`dest`	Location where normalized score to store
`region`	Region name

`lqtool.process.zscore` *Computes normalized z score based on the partition matrix*

Description

Computes normalized z score based on the partition matrix

Usage

```
lqtool.process.zscore(data, partitionMatrix)
```

Arguments

`data`	LQuant matrix
`partitionMatrix`	Partition matrix

`lqtool.writeRegionFactor` *Generates and writes file to the output location*

Description

Generates and writes file to the output location

Usage

```
lqtool.writeRegionFactor(dest, c_date, region, IN, factor_data, basic_factor,
  includeCusip)
```

Arguments

`dest`	Destination directory
`c_date`	Date of the factor
`region`	Region Name
`IN`	Masking Flag
`factor_data`	LQuant data matrix
`basic_factor`	LQuant basic factor matrix
`includeCusip`	Flag indicating whether cusip should be included in the file

`ltool.addFootNote` *Adds Foot Note to View Port*

Description

Adds Foot Note to View Port

Usage

```
ltool.addFootNote()
```

`ltool.allnull` *Checks if All values in a row is null*

Description

Checks if All values in a row is null

Usage

```
ltool.allnull(data, row)
```

Arguments

`data`	data matrix
`row`	row to perform check on

Value

TRUE if all values in the row is null

Examples

```
ltool.allnull(matrix(c(NA,NA,1,NA),nrow = 2),2)
```

`ltool.as.data.frame` *Converts list of factors to data frame*

Description

Converts list of factors to data frame

Usage

```
ltool.as.data.frame(data)
```

Arguments

`data`

Examples

```
m<-matrix(c(1,2,3,4),nrow=2)
rownames(m)<-c('006066.01','001234.01')
colnames(m)<-c('2017-01-31','2017-02-28')
m2<-matrix(c(1,2,3,4),nrow=2)
rownames(m2)<-c('006066.01','001234.01')
colnames(m2)<-c('2017-01-31','2017-02-28')
ltool.as.data.frame(list(SCORE1=m,SCORE2=m2))
```

`ltool.codahale.profile.read` *Reads files from Codahale Directory*

Description

Reads files from Codahale Directory

Usage

```
ltool.codahale.profile.read(path, pattern)
```

Arguments

`path`	Path to Codahale director
`pattern`	Pattern of file to read

Value

Profile Matrix

`ltool.codahale.profile.trimlabel` *Gets label from Codahale Generated File*

Description

Gets label from Codahale Generated File

Usage

```
ltool.codahale.profile.trimlabel(l)
```

Arguments

`Name`	of the file

`ltool.codahale.profiler.plot` *Plots the profiler matrix*

Description

Plots the profiler matrix

Usage

```
ltool.codahale.profiler.plot(path, pattern = NULL, metric = "AVG", l1 = 1,
  l2 = NA)
```

Arguments

`path`	Codahale directory
`pattern`	Types of file
`metric`	Metric to plot [AVG,TOTAL,COUNT]
`l1`	[Starting Index, default 1]
`l2`	[Starting Index, default length(list)]

Value

GGPlot object

`ltool.codahale.profiler.summ` *Summarized Codahale profile matrix*

Description

Summarized Codahale profile matrix

Usage

```
ltool.codahale.profiler.summ(path, pattern, metric)
```

Arguments

`path`	Path to Codahale Directory
`pattern`	Pattern
`metric`	One of these [AVG,TOTAL,COUNT]

Value

Codahale profile matrix

`ltool.codahale.summary` *Metric Summary from CSV*

Description

Metric Summary from CSV

Usage

```
ltool.codahale.summary(d, metric)
```

Arguments

`metric`

`ltool.createReport` *Creates a multi page report based on the title and data queries <b> THE FUNCTION RELIES ON conn OBJECT to be defined in the session </b>*

Description

Creates a multi page report based on the title and data queries <b> THE FUNCTION RELIES ON conn OBJECT to be defined in the session </b>

Usage

```
ltool.createReport(title, f_query, qsum, q, cols, maxRowsPerPage)
```

Arguments

`title`	Text title of the report (Will be printed on each page)
`f_query`	Query Executor Function
`qsum`	SQL Query for getting the summary data
`q`	SQL Query for getting the char data
`cols`	Number of Horizontally stacked charts on a page
`maxRowsPerPage`	Number of Vertical charts on a page

Value

Prints to the current active device

`ltool.datacheck.createReport` *Creates a PDF file from 2 Factor data file*

Description

Creates a PDF file from 2 Factor data file

Usage

```
ltool.datacheck.createReport(file1, file2, pdfile)
```

Arguments

`pdfile`

`ltool.datacheck.detplot` *Makes the multi page detailed plot of the difference matrix*

Description

Makes the multi page detailed plot of the difference matrix

Usage

```
ltool.datacheck.detplot(title, diffmatrix, rowRange)
```

Arguments

`title`	Title to be printed on each page
`diffmatrix`	Difference Matrix
`rowRange`	Range of the matrix

Value

Prints on the open printing device

`ltool.datacheck.overview` *Draw overview chart*

Description

Draw overview chart

Usage

```
ltool.datacheck.overview(diff)
```

Arguments

`diff`	Diff list

ltool.datacheck.report

Create data check reports

Description

Create data check reports

Usage

```
ltool.datacheck.report(diffmatrix)
```

Arguments

diffmatrix	difference matrix

ltool.datacheck.summplot

Creates multi page summary plot

Description

Creates multi page summary plot

Usage

```
ltool.datacheck.summplot(title, diffmatrix, name, rowwise)
```

Arguments

title	Title to be printed on each page
diffmatrix	Difference Matrix
name	Name of the row or column
rowwise	Boolean Flag, when set to TRUE

Examples

```
m1<-matrix(c(0.1,0.2,0.3,0.4),nrow = 2)
colnames(m1)<-c('A','B')
rownames(m1)<-c('X','Y')
ltool.datacheck.summplot('Test',m1,'T1',TRUE)
```

ltool.diffmat *Computes Difference Matrix between 2 LQuant Matrices*

Description

Computes Difference Matrix between 2 LQuant Matrices

Usage

```
ltool.diffmat(dat1, dat2, epsilon)
```

Arguments

dat1	List of matrices from source 1 (IxNxM)
dat2	List of matrices from source 2 (IxNxM)
epsilon	Epsilon

Value

Returns matrix of (IXM)

Examples

```
ltool.diffmat(dat1,dat2,1e-5)
```

ltool.firstNonNullIndex *First Non Null Index Return the first row for which the data matrix has at least one non null value*

Description

First Non Null Index Return the first row for which the data matrix has at least one non null value

Usage

```
ltool.firstNonNullIndex(data)
```

Arguments

data	data matrix

Value

Row Index when the first non null value is encountered. If all values are null, -1 is returned

Examples

```
ltool.firstNonNullIndex(t(matrix(c(NA,NA,1,NA),nrow = 2)))
```

ltool.footNoteGP *Foot Note Graphic Parameter Hard Coded Font and Color that should be used in Reports Foot Note*

Description

Foot Note Graphic Parameter Hard Coded Font and Color that should be used in Reports Foot Note

Usage

```
ltool.footNoteGP()
```

Value

Returns GP object

Examples

```
ltool.footNoteGP()
```

ltool.heatmap *Creates heat map from a matrix*

Description

Creates heat map from a matrix

Usage

```
ltool.heatmap(diffmatrix, rowRange = NULL, colRange = NULL)
```

Arguments

diffmatrix	Numeric Matrix
rowRange	Row Range to be subsetted
colRange	Col Range to be subsetted

Value

GGPlot object

Examples

```
m<-matrix(c(0.0,0.1,0.2,0.3),nrow=2)
rownames(m)<-c('A','B')
colnames(m)<-c('A','B')
ltool.heatmap(m)
```

`ltool.id.charToInt` *Convert character string id to integer*

Description

Convert character string id to integer

Usage

```
ltool.id.charToInt(x)
```

Arguments

`x`	QES interal ID

Examples

```
ltool.id.charToInt('006066.01C')
```

`ltool.id.decrypt` *Decrpts and previously encrypted id.*

Description

Decrpts and previously encrypted id.

Usage

```
ltool.id.decrypt(ids)
```

Arguments

`ids`

Examples

```
ltool.id.decrypt(c(”04MGLED3K4” ”540ONQ2G0Z”))
```

`ltool.id.encrypt` *Encrypts an id string*

Description

Encrypts an id string

Usage

```
ltool.id.encrypt(ids)
```

Arguments

`ids`	list/vector of ids

`ltool.id.hasher` *Returns id hasher*

Description

Returns id hasher

Usage

```
ltool.id.hasher()
```

Examples

```
ltool.id.hasher()
```

`ltool.id.intToChar` *Convert int Id back to String id*

Description

Convert int Id back to String id

Usage

```
ltool.id.intToChar(x)
```

Arguments

`x`	integer id

Examples

```
ltool.id.intToChar(60660122)
```

ltool.plotbar *Plots a bar chart from a named vector or list*

Description

Plots a bar chart from a named vector or list

Usage

```
ltool.plotbar(data, x, y, title)
```

Arguments

data	vector or list
x	Name of the Data Item
y	Name of the Data Value
title	Title to Add to Chart

ltool.plotfactors *Plot Factors and return GGPlot object*

Description

Plot Factors and return GGPlot object

Usage

```
ltool.plotfactors(req)
```

Arguments

req	LQuant request object

Value

GGPlot object

`ltool.plotgrid` *Plots a data frame to a multi-page grid*

Description

Plots a data frame to a multi-page grid

Usage

```
ltool.plotgrid(title, n, plotfn, cols, maxRowsPerPage)
```

Arguments

`title`	Title text to be used. Will be printed on each report
`cols`	Number of horizontally stacked charts
`maxRowsPerPage`	Maximum number of vertically stacked charts
`summdata`	Data Frame, First row is considered to be the x-axis for all charts

Examples

```
d<-data.frame(x=c(1,2,3,4),y1=c(1,2,3,4),y2=c(1,4,9,16),y3=c(1,8,27,64))
ltool.plotgrid('This is a test',d,2,1)
```

`ltool.plotgridpage` *Creates charts from data frame. The charts are plotted on a multi-page report. Number of charts on a page can be controlled by paramters cols*

Description

Creates charts from data frame. The charts are plotted on a multi-page report. Number of charts on a page can be controlled by paramters cols

Usage

```
ltool.plotgridpage(title, n, plotfn, cols, offset, rows, margin)
```

Arguments

`title`	Text title to be printed on each page
`cols`	Number of horizontally stacked charts in a page
`offset`	First column to plot
`rows`	Number of rows
`margin`	Margin matrix
`summdata`	Data Frame whose column to be plotted

Examples

```
d<-data.frame(x=c(1,2,3,4),y1=c(1,2,3,4),y2=c(1,4,9,16),y3=c(1,8,27,64))
ltool.plotgridpage('This is a test',d,2,0,2,c(0.02,0.02,0.1,0.02))
```

`ltool.plotmargin` *Return default plot margin*

Description

Return default plot margin

Usage

```
ltool.plotmargin()
```

`ltool.printSummary` *Prints the Summary page of report*

Description

Prints the Summary page of report

Usage

```
ltool.printSummary(title, data)
```

Arguments

`title`	Text title to be printed on top
`data`	Data to be printed

Examples

```
ltool.printSummary('This is a test',data.frame(Item=c('A','B','C'),Value=c(1,10,100)))
```

`ltool.randomforest.getScoreRF` *Compute Random Forest Prediction*

Description

Compute Random Forest Prediction

Usage

```
ltool.randomforest.getScoreRF(factor_data, current_Date, classifier)
```

Arguments

`factor_data`	List of matrices containing features
`current_Date`	Date for which prediction to be done
`classifier`	Classified object returned by ltool.randomForest.learRF

Examples

```
trainingPeriod<-c('2014-12-31','2015-12-31')
model_RF<-ltool.randomforest.learnRF(FMRTN1M,factor_data,trainingPeriod)
testDate<-"2016-12-31"
score_RF<-ltool.randomforest.getScoreRF(factor_data,testDate,model_RF)
```

`ltool.randomforest.learnRF` *Runs Random Forest Algorithm*

Description

Runs Random Forest Algorithm

Usage

```
ltool.randomforest.learnRF(FMRTN1M, factor_data, trainingPeriod, thresh = 0.5,
  m_nodes = 10, binary = FALSE, minCoverage = 0.6)
```

Arguments

`FMRTN1M`	Forward Return as Label
`factor_data`	List of Data Matrices containing the features
`trainingPeriod`	Training Window Period as Dates
`minCoverage`	

Examples

```
trainingPeriod<-c('2014-12-31','2015-12-31')
model_RF<-ltool.randomforest.learnRF(FMRTN1M,factor_data,trainingPeriod)
```

`ltool.regression.getScoreLinear` *Forecasts dependent variable (see ltool.regression.getScoreLinear)*

Description

Forecasts dependent variable (see ltool.regression.getScoreLinear)

Usage

```
ltool.regression.getScoreLinear(factor_data, current_Date, coeffs)
```

Arguments

`factor_data`	List of matrices containing features
`current_Date`	Date for which forecast to be made
`coeffs`	Coefficient object returned by call to ltool.regression.linearCoeffs

Examples

```
trainingPeriod<-c('2014-12-31','2015-12-31')
coeffs<-ltool.regression.linearCoeffs(FMRTN1M,factor_data,trainingPeriod)
testDate<-"2016-12-31"
score_Linear<-ltool.regression.getScoreLinear(factor_data,testDate,coeffs)
```

`ltool.regression.linearCoeffs` *Runs regression on the matrices*

Description

Runs regression on the matrices

Usage

```
ltool.regression.linearCoeffs(FMRTN1M, factor_data, trainingPeriod,
  minCoverage = 0.6, regMethod = "ols", stepwise = TRUE)
```

Arguments

`FMRTN1M`	Forward return as the dependent variable
`factor_data`	List of matrices containing features
`trainingPeriod`	training period as list of 2 dates
`stepwise`	

Examples

```
trainingPeriod<-c('2014-12-31','2015-12-31')
coeffs<-ltool.regression.linearCoeffs(FMRTN1M,factor_data,trainingPeriod)
```

`ltool.sendEmail` *Sends emails*

Description

Sends emails

Usage

```
ltool.sendEmail(subject, message)
```

Arguments

`message`

`ltool.titleGP` *Title Graphic Parameter Hard Coded Font and Color for title of reports*

Description

Title Graphic Parameter Hard Coded Font and Color for title of reports

Usage

```
ltool.titleGP()
```

`ltool.to.df` *Converts matrix to data frame*

Description

Converts matrix to data frame

Usage

```
ltool.to.df(mat, factor)
```

Arguments

`mat`	Matrix containing securities and date
`factor`	Name of the factor

Examples

```
m<-matrix(c(1,2,3,4),nrow=2)
rownames(m)<-c('006066.01','001234.01')
colnames(m)<-c('2017-01-31','2017-02-28')
ltool.to.df(m,'SCORE')
```

`ltool.trim` *Trim Matrix The function returns a trimmed matrix. All preceeding null values are truncated.*

Description

Trim Matrix The function returns a trimmed matrix. All preceeding null values are truncated.

Usage

```
ltool.trim(data)
```

Arguments

`data`	data matrix to be trimmed

Value

trimmed matrix

Examples

```
ltool.trim(t(matrix(c(NA,NA,1,NA),nrow = 2)))
```

`MACD` *Moving Averages Convering Diverging*

Description

Moving Averages Convering Diverging

Usage

```
MACD(price, long = 26, short = 12, M = 9)
```

Arguments

`long`	long window
`short`	short window
`M`	moving average window

`MFI` *Money Flow Index*

Description

Money Flow Index

Usage

```
MFI(closeprice, highprice, lowprice, cashvol, n = 14)
```

Arguments

`n`

MI *Mass Index*

Description

Mass Index

Usage

```
MI(highprice, lowprice, long = 25, short = 9)
```

Arguments

`short`	

multiplot *Multiple plot function*

Description

If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE), then plot 1 will go in the upper left, 2 will go in the upper right, and 3 will go all the way across the bottom.

Usage

```
multiplot(..., plotlist = NULL, file, cols = 1, layout = NULL)
```

Arguments

`...`	ggplot objects
`plotlist`	list of ggplot objects
`file`	name of the file to draw the plot
`cols`	Number of columns in layout
`layout`	A matrix specifying the layout. If present, 'cols' is ignored.

NVI *Negative Volume Index*

Description

Negative Volume Index

Usage

```
NVI(closeprice, cashvol, n = 14)
```

Arguments

`cashvol`	cash volume
`n`	

OBV *On Balance Volume*

Description

On Balance Volume

Usage

```
OBV(closeprice, cashvol, n = 14)
```

Arguments

`cashvol`	cash volume
`n`	

PER *Alpha Risk Parity*

Description

Alpha Risk Parity

Usage

```
PER(Sigma, maxwgt = 1, par = NULL, alpha = NULL, percentage = TRUE, ...)
```

Arguments

`Sigma`	covariance matrix
`maxwgt`	maximum weight
`par`	initial portfolio weight
`alpha`	input alpha score
`percentage`	output percentage
`...`	

Value

portfolio weight for the alpha risk parity

`PGMV` *mean variance*

Description

mean variance

Usage

```
PGMV(Returns = NULL, Sigma = NULL, maxwgt = 1, minwgt = 0,
  alpha = NULL, lambda = 1, percentage = TRUE, ...)
```

Arguments

`...`

`plot.backtest.bar` *Bar chart input from LBacktester: CAGR, Vol, IR*

Description

Bar chart input from LBacktester: CAGR, Vol, IR

Usage

```
plot.backtest.bar(vector_quantile, yLabel, isPercent = TRUE, title = "")
```

Arguments

`title`

Examples

```
plot.backtest.bar(outBacktest$CAGR,"Average Annual Return (%)",isPercent=TRUE)
```

`plot.backtest.barline` *3. Bar charts with line overlay input from LBacktester: ICs, Coverage, SCs, turnover*

Description

3. Bar charts with line overlay input from LBacktester: ICs, Coverage, SCs, turnover

Usage

```
plot.backtest.barline(vector_ts, yLabel, isPercent = TRUE, stats = TRUE,
  period = 12, title = "", summaryReturns = NULL)
```

Arguments

`summaryReturns`

Examples

```
plot.backtest.barline(outBacktest$ICs,"Rank IC (%)",isPercent = TRUE,stats=TRUE)
```

`plot.backtest.barsimple` *1.b Simple bar chart input from LBacktester: ICDecay, hitRate*

Description

1.b Simple bar chart input from LBacktester: ICDecay, hitRate

Usage

```
plot.backtest.barsimple(vector_monthly, yLabel, isPercent = TRUE,
  title = "")
```

Arguments

`title`

`plot.backtest.Basic` *Basic Backtest*

Description

Basic Backtest

Usage

```
plot.backtest.Basic(list_BacktestBasic, mat_factor, universeName,
  factorName = "", baskets, factorCode = "")
```

Arguments

`factorCode`

plot.backtest.density *5.a Plot raw factor score density input from LBacktester: Factor score matrix*

Description

5.a Plot raw factor score density input from LBacktester: Factor score matrix

Usage

```
plot.backtest.density(matrix_ts, yLabel, title = "")
```

Arguments

`title`

Examples

```
plot.backtest.density(1/rawData$pe,"Factor score")
```

plot.backtest.density3D *5.a Plot raw factor score density input from LBacktester: Factor score matrix*

Description

5.a Plot raw factor score density input from LBacktester: Factor score matrix

Usage

```
plot.backtest.density3D(matrix_ts, yLabel, title = "Factor density",
  round_decimal = 1, expand1 = 0.3, theta1 = 140, phi1 = 40,
  trim_outliers = c(0.1, 0.9))
```

Arguments

`trim_outliers`

Examples

```
plot.backtest.density(1/rawData$pe,"Factor score")
```

plot.backtest.distribution

5.a Plot raw factor score density input from LBacktester: Factor score matrix

Description

5.a Plot raw factor score density input from LBacktester: Factor score matrix

Usage

```
plot.backtest.distribution(matrix_ts, yLabel, title = "", baskets)
```

Arguments

baskets

Examples

```
plot.backtest.density(1/rawData$pe,"Factor score")
```

plot.backtest.seasonality

4. Seasonality chart (only if backtesting frequency=monthly) input from LBacktester: ICs, LSreturns

Description

4. Seasonality chart (only if backtesting frequency=monthly) input from LBacktester: ICs, LSreturns

Usage

```
plot.backtest.seasonality(vector_ts, yLabel, isPercent = TRUE, title = "")
```

Arguments

title

Examples

```
plot.backtest.seasonality(outBacktest$ICs,"Rank IC (%)",isPercent = TRUE)
```

plot.backtest.wealth *2.a Line charts input from LBacktester: wealth*

Description

2.a Line charts input from LBacktester: wealth

Usage

```
plot.backtest.wealth(matrix_quantile_wealth, yLabel, title = "")
```

Arguments

`title`

Examples

```
plot.backtest.wealth(outBacktest$wealth,"Cumulative Performance")
```

PMD *max diversificaiton*

Description

max diversificaiton

Usage

```
PMD(Returns = NULL, Sigma = NULL, maxwgt = 1, minwgt = 0,
  percentage = TRUE, ...)
```

Arguments

`...`

PMO *DecisionPoint Price Momentum Oscillator*

Description

DecisionPoint Price Momentum Oscillator

Usage

```
PMO(pricedata, slow = 26, fast = 12, M = 9)
```

Arguments

`slow`	slow window (long)
`fast`	fast window (short)
`M`	moving average window

PMTD *Tail dependence*

Description

Tail dependence

Usage

```
PMTD(Returns, Sigma = NULL, maxwgt = 1, alpha = NULL, method = "EmpTC",
  k = NULL, percentage = TRUE, ...)
```

Arguments

`Returns`	return matrix
`Sigma`	covariance matrix (optional)
`maxwgt`	maximum weight
`alpha`	input alpha score
`method`	default will use 'EmpTC'
`...`	

port_CoVar *Portfolio Covariance Computes covariance matrix when provided with the time series of returns of securities*

Description

Portfolio Covariance Computes covariance matrix when provided with the time series of returns of securities

Usage

```
port_CoVar(mat_returns, isrolling = TRUE, window_size = 12)
```

Arguments

`mat_returns`	Return matrix (N: Securities, M: Dates)
`window_size`	Size of the window to compute the covariance statistics

PPO *percentage price oscillator*

Description

percentage price oscillator

Usage

```
PPO(price, fast = 12, slow = 26, M = 9)
```

Arguments

fast	fast window (short)
slow	slow window (long)
M	moving average window

PVO *Percentage Volume Oscillator*

Description

Percentage Volume Oscillator

Usage

```
PVO(vol, fast = 12, slow = 26, M = 9)
```

Arguments

vol	volume
fast	fast window (short)
slow	slow window (long)
M	moving average window

RSI *Relative Strength Index*

Description

Relative Strength Index

Usage

```
RSI(pricedata, n = 14)
```

Arguments

pricedata	typically input closeprice
n	window for the Bollinger Bands

SO *Stochastic Oscillator*

Description

Stochastic Oscillator

Usage

```
SO(closeprice, lowprice, highprice, n = 39)
```

Arguments

`n`

StochRSI *Stochastic RSI*

Description

Stochastic RSI

Usage

```
StochRSI(closeprice, highprice, lowprice, n = 14)
```

Arguments

`n`

TRIX *TRIX*

Description

TRIX

Usage

```
TRIX(price, n = 15)
```

Arguments

`n`

`TSI` *True Strength Index*

Description

True Strength Index

Usage

```
TSI(pricedata, slow = 25, fast = 13)
```

Arguments

`slow`	slow window (long)
`fast`	fast window (short)

`UO` *Ultimate Oscillator*

Description

Ultimate Oscillator

Usage

```
UO(closeprice, lowprice, highprice, n1 = 28, n2 = 14, n3 = 7)
```

Arguments

`n1`	long window
`n2`	short window
`n3`	moving average window

`VI` *Vortex Indicator*

Description

Vortex Indicator

Usage

```
VI(closeprice, lowprice, highprice, n = 20)
```

Arguments

`n`	

VPT *Volume Price Trend*

Description

Volume Price Trend

Usage

```
VPT(closeprice, cashvol, n = 14)
```

Arguments

`n`

WillimsR *Williams percentage R*

Description

Williams percentage R

Usage

```
WillimsR(closeprice, highprice, lowprice, n = 14)
```

Arguments

`closeprice`

`n`

winsorize *Winsorize*

Description

Windorizes a factor matrix

Usage

```
winsorize(factor, winsorizeRatio = 0.01)
```

Arguments

`factor`	Factor matrix to winsorize
`winsorizeRatio`	Threshold for winsorization default 0.01

`wLag` *wLag*

Description

Computes Lag for the factor

Lag Function

Usage

```
wLag(x, k = 1)

wLag(x, k = 1)
```

Arguments

`x` can be a vector or a matrix, with dates as names or colnames

`k` Number of periods to lag it by, default = 1, if k<0, indicating future data

`k`

`wRank` *wRank*

Description

Computes Rank of a numerical array

Rank Function for unsorted array

Usage

```
wRank(x, ties.method = "average", na.last = "keep")

wRank(x, ties.method = "average", na.last = "keep")
```

Arguments

`x` vector of number including NAs

`ties.method` A character string specifying how ties are treated, see ‘Details’; can be abbreviated

`na.last`

`write.summary.stats.temp` *Summary Stats*

Description

Summary Stats

Usage

```
write.summary.stats.temp(list_BacktestBasic, universeName, factorName, baskets)
```

Arguments

`baskets`

`write.summary.statsALL` *Summary*

Description

Summary

Usage

```
write.summary.statsALL(list_BacktestBasic, universeName, factorNames, baskets)
```

Arguments

`baskets`

Index