Using *quantstrat*

to evaluate intraday trading strategies

this version: 2012-05-22

Contents

- Strategy creation
- R tools
- Luxor strategy: simple implementation
- Optimizing parameters
- Exit and risk management
- Walk forward analysis
- Appendix

Overview

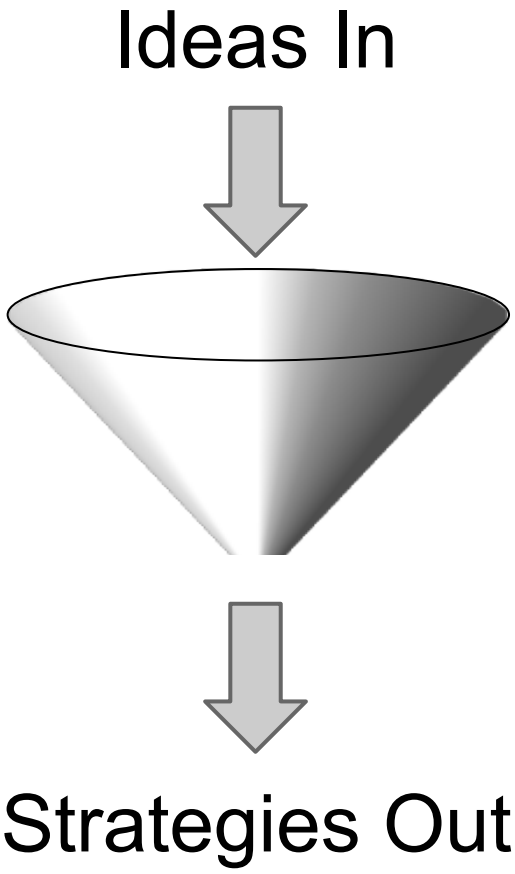
- Richard Feynman said that science was the process of first making a guess, then computing the consequences of that guess (building a model), and finally confirming or rejecting the guess with experiment.

- Why do we backtest?
- What do we hope to achieve?
- What tools do we use?

- create a scientific, hypothesis-driven approach to quantitative strategy development

Strategy Creation: Idea Funnel

- formulate your hypothesis
- test idea in research platform (R, historical data (e.g. Reuters))
- refine if promising
  - (or new ideas suggest themselves during testing)
- reject *quickly* if not promising
  - Vast majority (90%+) of ideas will not work out
- perform parameter robustness and walk forward optimization testing
- when you have a credible research model (your quantstrat code), implement for production rollout



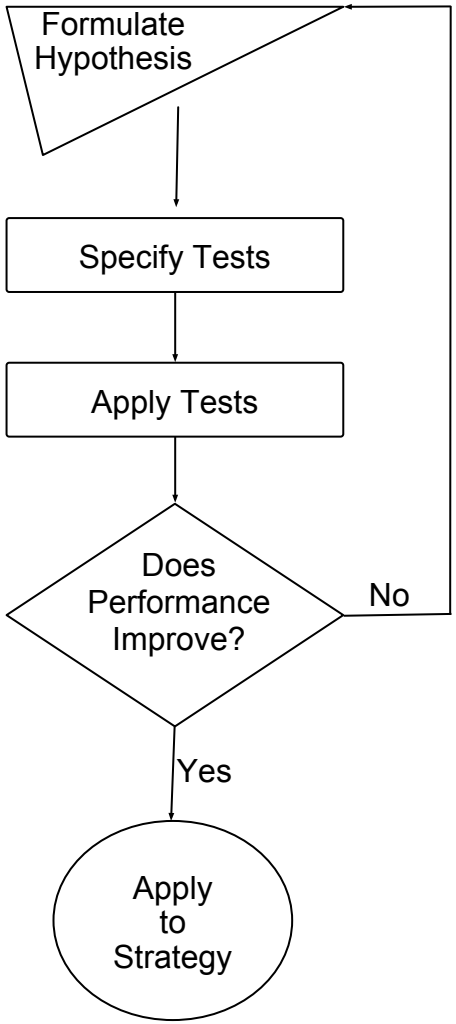
Strategy Creation : Product Testing

- product testing may be done both before and after production rollout
- identify hypotheses about the mechanism of action of the strategy. How does your strategy make money?
- generate lists of instruments that may be subject to the same method of action
- test target instruments
  - start with your 'standard' parameters or parameter ranges
  - run Parameter Robustness and
  - Walk Forward Optimization on promising assets
- roll out new instruments to production

Strategy Improvement Process

- no strategy is 'done' until you turn it off
- collect ideas on improving strategy
- formalize Hypotheses from ideas
- specify tests that will demonstrate improvement based on the hypothesis
- specify success criteria for measuring results
- code improvement hypothesis into the reference research implementation of the strategy
- perform standard optimizations on instruments currently traded on the strategy
- Evaluate Performance:

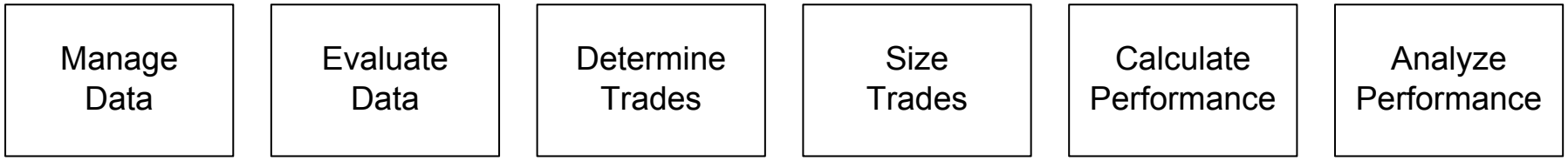
  if out of sample performance improves, code changes to put into production



R tools

How are we going to do all this?

Trade Simulation Tool Chain



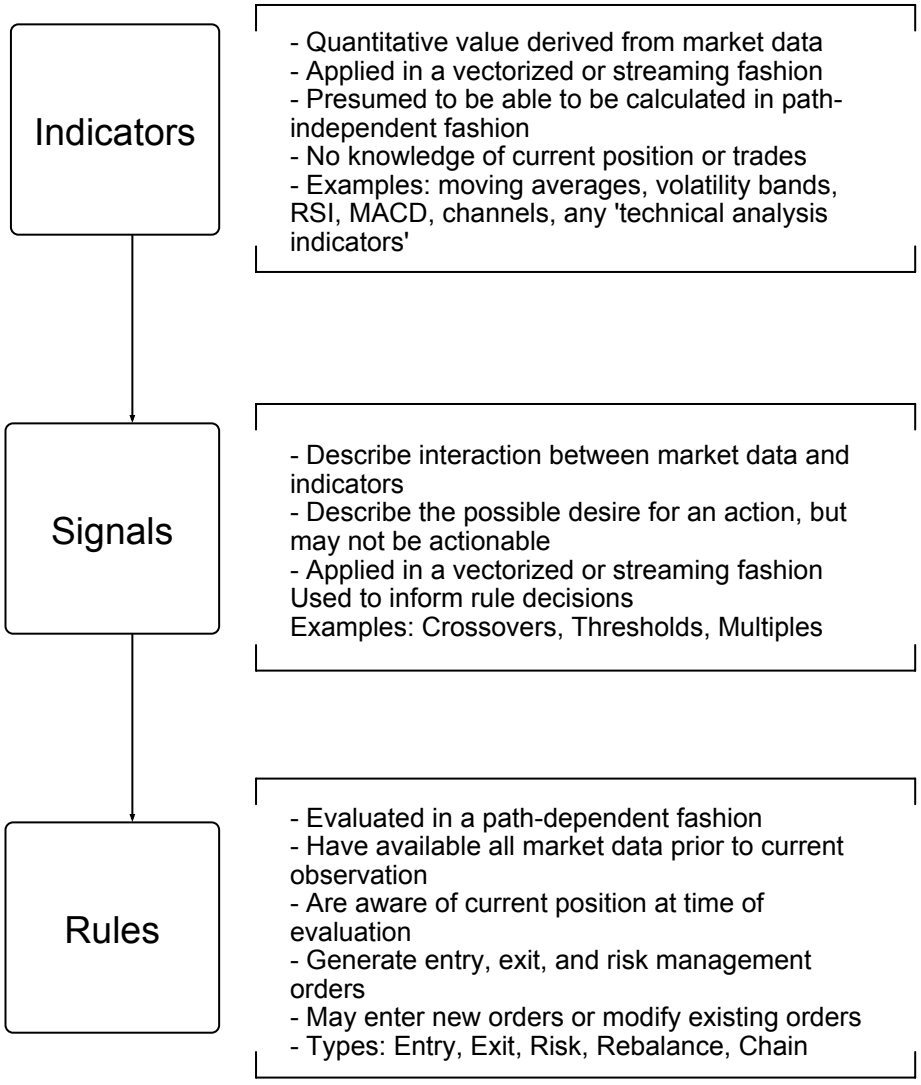
Types of Activities

Manage Data	Evaluate Data	Determine Trades	Size Trades	Calculate Performance	Analyze Performance
Connect to database	Calculate indicators	Estimate pre-trade pricing	Optimize portfolio	Specify contract specs	Calculate returns and risk
Download historical data	Transform prices	Forecast return	Budget risk	Capture trades	Compare to benchmarks
Clean and align data	Estimate volatility	Forecast risk	Calculate target position	Calculate positions	Provide attribution
Graph prices and indicators	Calculate trailing volume	Evaluate rules	Calculate trade size	Calculate P&L	Analyze risk
		Generate signals	Evaluate trading costs	Aggregate portfolio	

Example R Packages

quantmod indexing RTAQ xts ...	TTR signal-extraction urca ...	quantstrat quantmod	LSPM Portfolio-Analytics parma	blotter Financial - Instrument	Performance - Analytics

quantstrat



- Designed and used for 'real' quantitative strategies at all frequencies
- Many strategies may be constructed from all open source components
- Proprietary strategies add custom
  - Indicators
  - Signal Functions
  - Order Sizing Logic

Installing quantstrat package

- blotter, quantmod, FinancialInstrument, xts
  - cran
- quantstrat
  - not (yet) available on cran
  - http://r-forge.r-project.org/projects/blotter/
    - tar/zip:
      - http://r-forge.r-project.org/R/?group_id=316
    - svn
      - svn co svn://r-forge.r-project.org/svnroot/blotter
  - https://github.com/milktrader/blotter/wiki/Installation-Guide-for-Windows

Mailing list:

- r-sig-finance@r-project.org

Luxor strategy

Simple implementation

See: Trading Systems: A New Approach to System Development and Portfolio Optimisation

By Emilio Tomasini, Urban Jaekle - Harriman House London 2009 - ISBN(s): 9781905641796, 9780857191496

About the code

- Don't cut and paste code from these slides!
  - the most current code for these functions is in the package repository on R-Forge
  - the luxor demo will be maintained
    - find the scripts in the demo/ directory
  - *quantstrat* is still in development, and it will continue to change in small (and sometimes large) ways
  - backwards compatibility is important,
  - but we won't sacrifice features for compatibility

"Luxor" strategy

- Symbol data:
  - GBPUSD
  - 30 min OHLC

- Crossing of 2 SMA's:
  - SMA n=10 ("fast")
  - SMA n=30 ("slow")

- Entry logic:
  - Long: stop-limit at high of SMA crossing bar
  - Short: stop-limit at low of SMA crossing bar

FinancialInstrument

stores instrument metadata...

Define instruments, ie. currencies:

```
require('FinancialInstrument')
currency(c('GBP', 'USD'))
```

either

```
exchange_rate(primary_id='GBPUSD', tick_size=0.0001)
```

or

```
exchange_rate(currency='USD', counter_currency='GBP',
tick_size=0.0001)
```

in a production environment, you would most like use saveInstruments() and loadInstruments() to re-use your entire instrument environment

Retrieving symbol data

```
data.dir <- '~/R.symbols'   # for example
.from <- '2002-10-21'
.to <- '2008-07-4'

getSymbols.FI(
        Symbols='GBPUSD',
        dir=data.dir,
        from=.from, to=.to
)
```

or (more general)

```
setSymbolLookup.FI(base_dir=data.dir, Symbols='GBPUSD')
getSymbols('GBPUSD', from=.from, to=.to)
```

or (quantstrat/demo)

```
getSymbols.FI(
        Symbols='GBPUSD',
        dir=system.file('extdata',package='quantstrat'),
        from=.from, to=.to
)
```

xts GBPUSD data object

```
> getSymbols.FI(Symbols='GBPUSD', dir=system.file('extdata', package='quantstrat'), from='2002-10-21', to='2002-10-31')
loading  GBPUSD .....
Reading  2002.10.21.GBPUSD.rda ... done.
Reading  2002.10.22.GBPUSD.rda ... done.
<.....>
Reading  2002.10.30.GBPUSD.rda ... done.
Reading  2002.10.31.GBPUSD.rda ... done.
rbinding data ... done.
[1] "GBPUSD"
> GBPUSD = to.minutes30(GBPUSD)
> GBPUSD = align.time(GBPUSD, 1800)
> head(GBPUSD)
                     Open   High    Low     Close  Volume
2002-10-21 02:30:00 1.5501 1.5501 1.5481 1.5482       0
2002-10-21 03:00:00 1.5483 1.5483 1.5473 1.5474       0
2002-10-21 03:30:00 1.5473 1.5480 1.5470 1.5478       0
2002-10-21 04:00:00 1.5478 1.5481 1.5471 1.5479       0
2002-10-21 04:30:00 1.5480 1.5501 1.5479 1.5494       0
2002-10-21 05:00:00 1.5493 1.5497 1.5487 1.5492       0
```

- we don't need *getSymbols*(), all of this could be done with *save*() and *load*()
- adjust periodicity
  - change time scales
- adjust alignment (optional)

- your data may be different, but formatting it in accordance with *is.OHLC*() or *is.BBO*() from **quantmod** will make your life easier

Creating a new strategy object

```
strategy(
        name='luxor',
        store=TRUE
)
```

- .strategy environment
  - strategy object (store=TRUE)
  - orderbook object
- strategy components
  - indicators
  - signals
  - rules
  - paramsets
- creates mktdata object

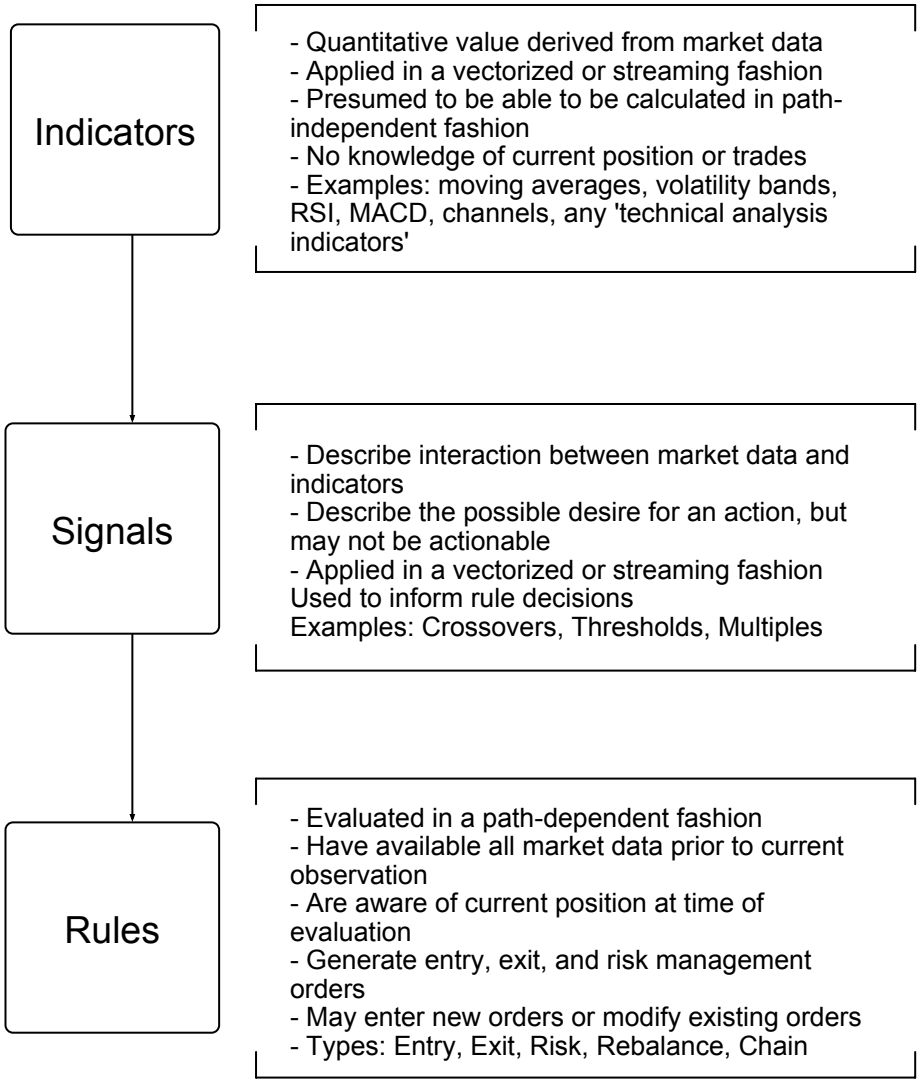
The mktdata object

- **xts** 'spreadsheet', strategy scratchpad
- initially: symbol data from portfolio

```
> head(GBPUSD)
                     Open   High    Low  Close Volume
2002-10-21 02:01:00 1.5501 1.5501 1.5501 1.5501      0
2002-10-21 02:02:00 1.5501 1.5501 1.5501 1.5501      0
2002-10-21 02:03:00 1.5501 1.5501 1.5501 1.5501      0
2002-10-21 02:04:00 1.5501 1.5501 1.5501 1.5501      0
2002-10-21 02:05:00 1.5500 1.5500 1.5500 1.5500      0
2002-10-21 02:06:00 1.5493 1.5497 1.5493 1.5497      0
```

- indicators will add columns
- signals will add columns
- user may add columns

quantstrat



- All of quantstrat is modular
- The 'name' property of most quantstrat functions names a function to call
- Takes advantage of delayed execution
- Your strategy specification can apply to multiple different instruments or timeframes

Indicators

Indicator: mathematical calculation based on security price and/or volume (Investopedia)

e.g. MA, MACD, RSI, custom indicators

TTR package provides many classical indicators, a.o. MA, MACD, RSI etc

Many proprietary strategies use custom indicators

quantstrat::add.indicator()

```
add.indicator(
        strategy = 'luxor',
        name = 'SMA',
        arguments = list(
                x = quote(Cl(mktdata)[,1]),
                n = 10
        ),
        label="nFast"
)

add.indicator(
        strategy = 'luxor',
        name = 'SMA',
        arguments = list(
                x = quote(Cl(mktdata)[,1]),
                n = 30
        ),
        label="nSlow"
)
```

- name of indicator, eg. TTR indicators
  - *SMA()*, *EMA()*
  - *RSI()*
  - *stoch()*
- data
  - use the mktdata object
  - apply *Cl()* function: first column named Close
  - *quote()*: evaluate later
- label
  - mktdata column name

Signals

Investopedia: A sign, usually based on technical indicators, that it is a good time to buy or sell a particular security

Quantstrat: Signals denote times at which the strategy *may* want to take action.

e.g. *sigCrossover(), sigThreshold(), sigFormula()*

quantstrat::add.signal()

```
add.signal(
        strategy='luxor',
        name='sigCrossover',
        arguments = list(
                columns=c("nFast", "nSlow"),
                relationship="gt"
        ),
        label='long'
)

add.signal(
        strategy='luxor',
        name='sigCrossover',
        arguments = list(
                columns=c("nFast", "nSlow"),
                relationship="lt"
        ),
        label='short'
)
```

- sample functions:
  - *sigCrossover*()
  - *sigThreshold*()
  - *sigComparison*()
  - *sigFormula*()
  - *sigPeak*()
- arguments:
  - columns
  - relationship, "gt", "lt", "eq", etc
- label
  - column name to place in mktdata

Rules

- rules are path dependent, have access to current market and portfolio state
- usually triggered by a signal
- rule types
  - risk
  - order
  - rebalance
  - exit
  - enter
  - chain
- rules are a point at which the strategy *may* place or cancel an order

quantstrat::add.rule()

```
add.rule(
        strategy,
        function to apply,
        list of arguments for function to apply
        type,
        label
)
```

- function: usually *ruleSignal*()
- rule type: 'enter', 'exit', 'risk' ...
- label: shows up in order book
- if not using *ruleSignal*, use a function with a similar argument signature to take advantage of dimension reduction

quantstrat::ruleSignal()

```
ruleSignal(

        sigcol='long',
        sigval=TRUE,

        orderside='long',

        ordertype='stoplimit',
        prefer='High',
        threshold=0.0005,

        orderqty=100000,

        replace=FALSE
)
```

- ordertypes:
  - 'limit'
  - 'stoplimit'
  - 'stoptrailing'
  - 'market'
    - transacts on next bar/obs.
    - default prefer='Close'
  - 'iceberg'
- prefer: price column to use from data object
- threshold: added to price

add.rule() - luxor entry rules

```
add.rule(strategy='luxor',
      name='ruleSignal',
      arguments=list(sigcol='long' , sigval=TRUE,
            orderside='long' ,
            ordertype='stoplimit', prefer='High', threshold=.threshold,
            orderqty=+.orderqty,
            replace=FALSE
      ),
      type='enter',
      label='EnterLONG'
)
add.rule(strategy='luxor',
      name='ruleSignal',
      arguments=list(sigcol='short', sigval=TRUE,
            orderside='short',
            ordertype='stoplimit', prefer='Low', threshold=-.threshold,
            orderqty=-.orderqty,
            replace=FALSE
      ),
      type='enter',
      label='EnterSHORT'
)
```

- stoplimit order using High+threshold (Low-threshold) as limit price
- we do not want to replace any pending exit orders

add.rule() - luxor exit rules

```
add.rule(strategy.st, name='ruleSignal',
      arguments=list(sigcol='long' , sigval=TRUE,
            orderside='short',
            ordertype='market',
            orderqty='all',
            TxnFees=.txn.fees,
            replace=TRUE
      ),
      type='exit',
      label='Exit2LONG'
)
add.rule(strategy.st, name='ruleSignal',
      arguments=list(sigcol='short', sigval=TRUE,
            orderside='long' ,
            ordertype='market',
            orderqty='all',
            TxnFees=.txn.fees,
            replace=TRUE
      ),
      type='exit',
      label='Exit2SHORT'
)
```

- 'market'/'all'
- TxnFees only on exit rules: 'end of trade marker' for *tradeStats()**
- replace any pending (open) entry orders!

* alternately, use method='trade' for *tradeStats()*

quantstrat::applyStrategy()

```
> applyStrategy(strategy.st, portfolio.st)
[1] "2002-10-22 02:00:00 GBPUSD 1e+05 @ 1.5447"
[1] "2002-10-22 17:30:00 GBPUSD -1e+05 @ 1.5435"
[1] "2002-10-22 17:30:00 GBPUSD -1e+05 @ 1.5447"
[1] "2002-10-23 03:00:00 GBPUSD 1e+05 @ 1.5486"
[1] "2002-10-23 03:00:00 GBPUSD 1e+05 @ 1.5492"
[1] "2002-10-23 17:30:00 GBPUSD -1e+05 @ 1.5468"
[1] "2002-10-24 03:00:00 GBPUSD -1e+05 @ 1.5459"
[1] "2002-10-24 11:30:00 GBPUSD 1e+05 @ 1.5484"
[1] "2002-10-24 12:00:00 GBPUSD 1e+05 @ 1.5493"
[1] "2002-10-25 04:00:00 GBPUSD -1e+05 @ 1.553"
[1] "2002-10-25 04:00:00 GBPUSD -1e+05 @ 1.553"
[1] "2002-10-25 12:00:00 GBPUSD 1e+05 @ 1.5513"
[1] "2002-10-27 23:30:00 GBPUSD -1e+05 @ 1.5508"
[1] "2002-10-28 10:30:00 GBPUSD 1e+05 @ 1.5554"
[1] "2002-10-28 10:30:00 GBPUSD 1e+05 @ 1.555"
[1] "2002-10-29 00:00:00 GBPUSD -1e+05 @ 1.5583"
[1] "2002-10-29 07:30:00 GBPUSD -1e+05 @ 1.5572"
[
```

- It Trades!
- lack of transactions at this point probably means you have a problem

The mktdata object:

> View(mktdata['2004-11'])

	row.names	GBPUSD.Open	GBPUSD.High	GBPUSD.Low	GBPUSD.Close	GBPUSD.Volume	GBPUSD.Close.SMA.10.nFast	GBPUSD.Close.SMA.30.nSlow	long	short
1	2004-11-01 00:00:00	1.835	1.8356	1.8348	1.8348	0	1.83625	1.833483	NA	NA
2	2004-11-01 00:30:00	1.8348	1.8356	1.8345	1.8347	0	1.83614	1.833607	NA	NA
3	2004-11-01 01:00:00	1.8347	1.8365	1.8347	1.8362	0	1.83615	1.833753	NA	NA
4	2004-11-01 01:30:00	1.8362	1.8367	1.8347	1.8355	0	1.83602	1.83384	NA	NA
5	2004-11-01 02:00:00	1.8355	1.836	1.8349	1.835	0	1.83584	1.83393	NA	NA
6	2004-11-01 02:30:00	1.8351	1.8356	1.8339	1.8349	0	1.83561	1.834057	NA	NA
7	2004-11-01 03:00:00	1.835	1.8355	1.834	1.8349	0	1.83535	1.83424	NA	NA
8	2004-11-01 03:30:00	1.8349	1.8352	1.8311	1.8318	0	1.83487	1.834297	NA	NA
9	2004-11-01 04:00:00	1.8317	1.8332	1.8309	1.8329	0	1.83457	1.834353	NA	NA
10	2004-11-01 04:30:00	1.8329	1.834	1.8324	1.8331	0	1.83438	1.83442	NA	1
11	2004-11-01 05:00:00	1.8331	1.8342	1.8329	1.8334	0	1.83424	1.834493	NA	NA
12	2004-11-01 05:30:00	1.8335	1.8336	1.8325	1.8329	0	1.83406	1.834597	NA	NA
13	2004-11-01 06:00:00	1.8329	1.8335	1.8316	1.8334	0	1.83378	1.834557	NA	NA
14	2004-11-01 06:30:00	1.8334	1.8338	1.8329	1.8334	0	1.83357	1.834583	NA	NA
15	2004-11-01 07:00:00	1.8335	1.8341	1.8319	1.8329	0	1.83336	1.83462	NA	NA
16	2004-11-01 07:30:00	1.8329	1.833	1.8316	1.832	0	1.83307	1.83464	NA	NA
17	2004-11-01 08:00:00	1.832	1.8329	1.8309	1.832	0	1.83278	1.83464	NA	NA
18	2004-11-01 08:30:00	1.832	1.8326	1.8299	1.8307	0	1.83267	1.834527	NA	NA
19	2004-11-01 09:00:00	1.8308	1.8335	1.8306	1.8333	0	1.83271	1.834503	NA	NA
20	2004-11-01 09:30:00	1.8332	1.8346	1.8316	1.8325	0	1.83265	1.83442	NA	NA
21	2004-11-01 10:00:00	1.8326	1.8332	1.8308	1.8329	0	1.8326	1.834363	NA	NA
22	2004-11-01 10:30:00	1.8329	1.8332	1.8303	1.8309	0	1.8324	1.8342	NA	NA
23	2004-11-01 11:00:00	1.831	1.8323	1.8295	1.8322	0	1.83228	1.83407	NA	NA
24	2004-11-01 11:30:00	1.8322	1.8341	1.8314	1.8338	0	1.83232	1.83397	NA	NA

quantstrat::getOrderBook()

> View(getOrderBook(portfolio.st)

$forex

$forex$GBPUSD)

	row.names	Order.Qty	Order.Price	Order.Type	Order.Side	Order.Threshold	Order.Status	Order.StatusTime	Prefer	Order.Set	Txn.Fees	Rule
1	2002-10-21 00:00:00	0	<NA>	init	long	0	closed	2002-10-21			0	
2	2002-10-21 23:30:00	1e+05	1.5447	stoplimit	long	5e-04	closed	2002-10-22 02:00:00	High	<NA>	0	EnterLONG
3	2002-10-22 17:00:00	all	1.5453	market	long	<NA>	closed	2002-10-22 17:30:00		<NA>	-30	Exit2SHORT
4	2002-10-22 17:00:00	-1e+05	1.5447	stoplimit	short	-5e-04	closed	2002-10-22 17:30:00	Low	<NA>	0	EnterSHORT
5	2002-10-23 02:30:00	all	1.5485	market	short	<NA>	closed	2002-10-23 03:00:00		<NA>	-30	Exit2LONG
6	2002-10-23 02:30:00	1e+05	1.5492	stoplimit	long	5e-04	closed	2002-10-23 03:00:00	High	<NA>	0	EnterLONG
7	2002-10-23 17:00:00	all	1.5468	market	long	<NA>	closed	2002-10-23 17:30:00		<NA>	-30	Exit2SHORT
8	2002-10-23 17:00:00	-1e+05	1.5463	stoplimit	short	-5e-04	replaced	2002-10-23 22:00:00	Low	<NA>	0	EnterSHORT
9	2002-10-23 22:00:00	1e+05	1.5491	stoplimit	long	5e-04	replaced	2002-10-24 01:30:00	High	<NA>	0	EnterLONG
10	2002-10-24 01:30:00	-1e+05	1.5459	stoplimit	short	-5e-04	closed	2002-10-24 03:00:00	Low	<NA>	0	EnterSHORT
11	2002-10-24 11:00:00	all	1.5476	market	short	<NA>	closed	2002-10-24 11:30:00		<NA>	-30	Exit2LONG
12	2002-10-24 11:00:00	1e+05	1.5493	stoplimit	long	5e-04	closed	2002-10-24 12:00:00	High	<NA>	0	EnterLONG
13	2002-10-25 03:30:00	all	1.5535	market	long	<NA>	closed	2002-10-25 04:00:00		<NA>	-30	Exit2SHORT
14	2002-10-25 03:30:00	-1e+05	1.553	stoplimit	short	-5e-04	closed	2002-10-25 04:00:00	Low	<NA>	0	EnterSHORT
15	2002-10-25 11:30:00	all	1.5528	market	short	<NA>	closed	2002-10-25 12:00:00		<NA>	-30	Exit2LONG
16	2002-10-25 11:30:00	1e+05	1.5533	stoplimit	long	5e-04	replaced	2002-10-25 12:00:00	High	<NA>	0	EnterLONG
17	2002-10-25 12:00:00	-1e+05	1.5508	stoplimit	short	-5e-04	closed	2002-10-27 23:30:00	Low	<NA>	0	EnterSHORT
18	2002-10-28 10:00:00	all	1.5538	market	short	<NA>	closed	2002-10-28 10:30:00		<NA>	-30	Exit2LONG
19	2002-10-28 10:00:00	1e+05	1.555	stoplimit	long	5e-04	closed	2002-10-28 10:30:00	High	<NA>	0	EnterLONG
20	2002-10-28 23:30:00	all	1.5582	market	long	<NA>	closed	2002-10-29 00:00:00		<NA>	-30	Exit2SHORT
21	2002-10-28 23:30:00	-1e+05	1.5572	stoplimit	short	-5e-04	replaced	2002-10-29 06:30:00	Low	<NA>	0	EnterSHORT
22	2002-10-29 06:30:00	1e+05	1.5594	stoplimit	long	5e-04	replaced	2002-10-29 07:00:00	High	<NA>	0	EnterLONG
23	2002-10-29 07:00:00	-1e+05	1.5572	stoplimit	short	-5e-04	closed	2002-10-29 07:30:00	Low	<NA>	0	EnterSHORT
24	2002-10-30 05:00:00	all	1.5563	market	short	<NA>	closed	2002-10-30 05:30:00		<NA>	-30	Exit2LONG
25	2002-10-30 05:00:00	1e+05	1.5578	stoplimit	long	5e-04	closed	2002-10-30 09:30:00	High	<NA>	0	EnterLONG
26	2002-10-30 11:00:00	all	1.5569	market	long	<NA>	closed	2002-10-30 11:30:00		<NA>	-30	Exit2SHORT
27	2002-10-30 11:00:00	-1e+05	1.5558	stoplimit	short	-5e-04	replaced	2002-10-30 12:00:00	Low	<NA>	0	EnterSHORT
28	2002-10-30 12:00:00	1e+05	1.5579	stoplimit	long	5e-04	closed	2002-10-30 13:00:00	High	<NA>	0	EnterLONG

blotter::updatePortf()

```
updatePortf(
        portfolio=portfolio.st,
        Symbols='GBPUSD',
        Dates=paste('::',as.Date(Sys.time()),sep='')
)
> paste('::',as.Date(Sys.time()),sep='')
[1] "::2013-04-03"
```

- *updatePortf*() 'marks the book' of the portfolio with the various accounting measures
- if you're working with high frequency data, it is common to mark the book on lower frequencies
- choosing a frequency to mark the book on does not change any trade data, which may still be higher frequency

blotter::chart.Pos() 2002-2008

blotter::chart.Posn()

blotter::tradeStats()

```
> View(t(tradeStats('forex')))
```

	row.names	GBPUSD
1	Portfolio	forex
2	Symbol	GBPUSD
3	Num.Txns	3793
4	Num.Trades	1896
5	Net.Trading.PL	30943.33
6	Avg.Trade.PL	16.43372
7	Med.Trade.PL	-146
8	Largest.Winner	4074
9	Largest.Loser	-2286
10	Gross.Profits	540586
11	Gross.Losses	-509427.7
12	Std.Dev.Trade.PL	783.0938
13	Percent.Positive	38.2384
14	Percent.Negative	61.7616
15	Profit.Factor	1.061163
16	Avg.Win.Trade	745.6359
17	Med.Win.Trade	514
18	Avg.Losing.Trade	-435.0364
19	Med.Losing.Trade	-356
20	Avg.Daily.PL	24.41876
21	Med.Daily.PL	-126
22	Std.Dev.Daily.PL	961.4933
23	Max.Drawdown	-17308
24	Profit.To.Max.Draw	1.787805
25	Avg.WinLoss.Ratio	1.713962
26	Med.WinLoss.Ratio	1.44382
27	Max.Equity	41754.33
28	Min.Equity	-3052
29	End.Equity	30943.33

- *tradeStats*() calculates statistics commonly seen in strategy reports from books and execution platforms
- 'trades' are calculated flat to flat for this report, representing a full round trip
- all statistics are explained in full in the documentation

blotter::perTradeStats()

	Start	End	Init.Pos	Max.Pos	Num.Txns	Max.Notional.Cost	Net.Trading.PL	MAE	MFE	Pct.Net.Trading.PL	Pct.MAE	Pct.MFE	tick.Net.Trading.PL	tick.MAE	tick.MFE
1	2002-10-21 23:00:00	2002-10-22 13:00:00	100000	100000	2	154690	-340.000	-340.000	180.00	-0.0022	-0.0022	0.0012	-34.0000	-34.0000	18.000
2	2002-10-22 14:00:00	2002-10-22 21:00:00	-100000	-100000	2	-154260	-385.650	-385.650	0.00	-0.0025	-0.0025	0.0000	-38.5650	-38.5650	0.000
3	2002-10-22 22:00:00	2002-10-23 03:00:00	100000	100000	2	154920	-387.300	-387.300	0.00	-0.0025	-0.0025	0.0000	-38.7300	-38.7300	0.000
4	2002-10-23 16:00:00	2002-10-23 20:00:00	100000	100000	2	154890	-190.000	-190.000	10.00	-0.0012	-0.0012	0.0001	-19.0000	-19.0000	1.000
5	2002-10-23 20:30:00	2002-10-24 06:00:00	-100000	-100000	2	-154650	-180.000	-180.000	160.00	-0.0012	-0.0012	0.0010	-18.0000	-18.0000	16.000
6	2002-10-24 07:00:00	2002-10-25 02:00:00	100000	100000	2	154920	340.000	0.000	620.00	0.0022	0.0000	0.0040	34.0000	0.0000	62.000
7	2002-10-25 03:00:00	2002-10-25 06:30:00	100000	100000	2	155560	-300.000	-300.000	0.00	-0.0019	-0.0019	0.0000	-30.0000	-30.0000	0.000
8	2002-10-25 06:30:00	2002-10-28 04:30:00	-100000	-100000	2	-155200	-180.000	-230.000	640.00	-0.0012	-0.0015	0.0041	-18.0000	-23.0000	64.000
9	2002-10-28 04:30:00	2002-10-29 02:00:00	100000	100000	2	155500	210.000	0.000	550.00	0.0014	0.0000	0.0035	21.0000	0.0000	55.000
10	2002-10-29 02:00:00	2002-10-29 04:00:00	-100000	-100000	2	-155600	-389.000	-389.000	90.00	-0.0025	-0.0025	0.0006	-38.9000	-38.9000	9.000
11	2002-10-30 01:00:00	2002-10-30 05:30:00	-100000	-100000	2	-155600	-90.000	-100.000	60.00	-0.0006	-0.0006	0.0004	-9.0000	-10.0000	6.000
12	2002-10-30 07:00:00	2002-10-30 13:00:00	100000	100000	2	155790	-190.000	-190.000	20.00	-0.0012	-0.0012	0.0001	-19.0000	-19.0000	2.000
13	2002-10-30 14:30:00	2002-10-31 22:30:00	100000	100000	2	155900	450.000	0.000	670.00	0.0029	0.0000	0.0043	45.0000	0.0000	67.000
14	2002-11-01 00:00:00	2002-11-01 01:30:00	-100000	-100000	2	-156240	-390.600	-390.600	0.00	-0.0025	-0.0025	0.0000	-39.0600	-39.0600	0.000
15	2002-11-03 21:00:00	2002-11-04 17:00:00	-100000	-100000	2	-156090	340.000	0.000	840.00	0.0022	0.0000	0.0054	34.0000	0.0000	84.000

- *perTradeStats*() collects statistics on flat-to-flat trades
- called by *charts.ME* for the raw data those charts draw from
- you could use these individual 'trades' as input to a Monte Carlo analysis of your strategy

quantstrat::save.strategy()

Save strategy, so we can load it later and add more components, or apply to different scenarios

- *`save.strategy(strategy.st)`*

jan@fuga:~/R/luxor$ ls -l luxor.RData

-rw-r--r-- 1 jan users 22450 Apr 7 22:53 ***luxor.RData***

- *`load.strategy(strategy.st)`*

quantstrat

Optimizing parameters

Parameter Optimization

- all strategies have parameters: What are the right ones?

- larger numbers of free parameters vastly increase the amount of data you need, lowering your degrees of freedom and increasing your chance of overfitting

- even your initial parameter choices are an optimization, you've chosen values you believe may be optimal

- parameter optimization just adds process and tools to your investment decision

Parameter sets

Optimizing strategy parameters

- Parameter sets are part of strategy
  - paramset.label
    - 'SMA'
  - distributions
    - FastSMA range from 1 to 30
    - SlowSMA range from 20 to 80
  - constraints
    - FastSMA < SlowSMA
- *apply.paramset*()
  - calls *applyStrategy*()
  - random selection or all combos

Optimizing Luxor SMA values

```
.FastSMA = (1:30)
.SlowSMA = (20:80)

load.strategy(strategy.st)

add.distribution(strategy.st,
      paramset.label = 'SMA',
      component.type = 'indicator',
      component.label = 'nFast',
      variable = list(n = .FastSMA),
      label = 'nFAST')

add.distribution(strategy.st,
      paramset.label = 'SMA',
      component.type = 'indicator',
      component.label = 'nSlow',
      variable = list(n = .SlowSMA),
      label = 'nSLOW')
```

```
add.constraint(s,
      paramset.label = 'SMA',
      distribution.label.1 = 'nFAST',
      distribution.label.2 = 'nSLOW',
      operator = '<',
      label = 'SMA')

save.strategy(strategy.st)

### want to delete this paramset?

delete.paramset(strategy.st, 'SMA')
```

foreach package

- hybrid of the standard ‘for’ loop and ‘lapply’ function
- facilitates parallelization
- returns a value
  - .combine function
  - default a list
- some parallel backends:
  - *registerDoSEQ*(): default backend (sequential)
  - doMC/doParallel: multiple CPUs or cores
  - doRedis:cross-platform backend
    - Redis server
    - Redis workers

apply.paramset()

```
require(doMC)
registerDoMC(cores=8)

load.strategy(strategy.st)

results <- apply.paramset(strategy.st,
        paramset.label='SMA',
        portfolio.st=portfolio.st,
        nsamples = 80,
        audit = NULL,
        verbose=TRUE)

print(results$tradeStats)
```

- creates param.combos from distributions and constraints
- runs *applyStrategy()* on portfolio for each param. combo
- nsamples: draws random selection
- audit: file name to store all portfolios and orderbooks

SMA paramset *tradeStats()*

View(t(results$tradeStats))

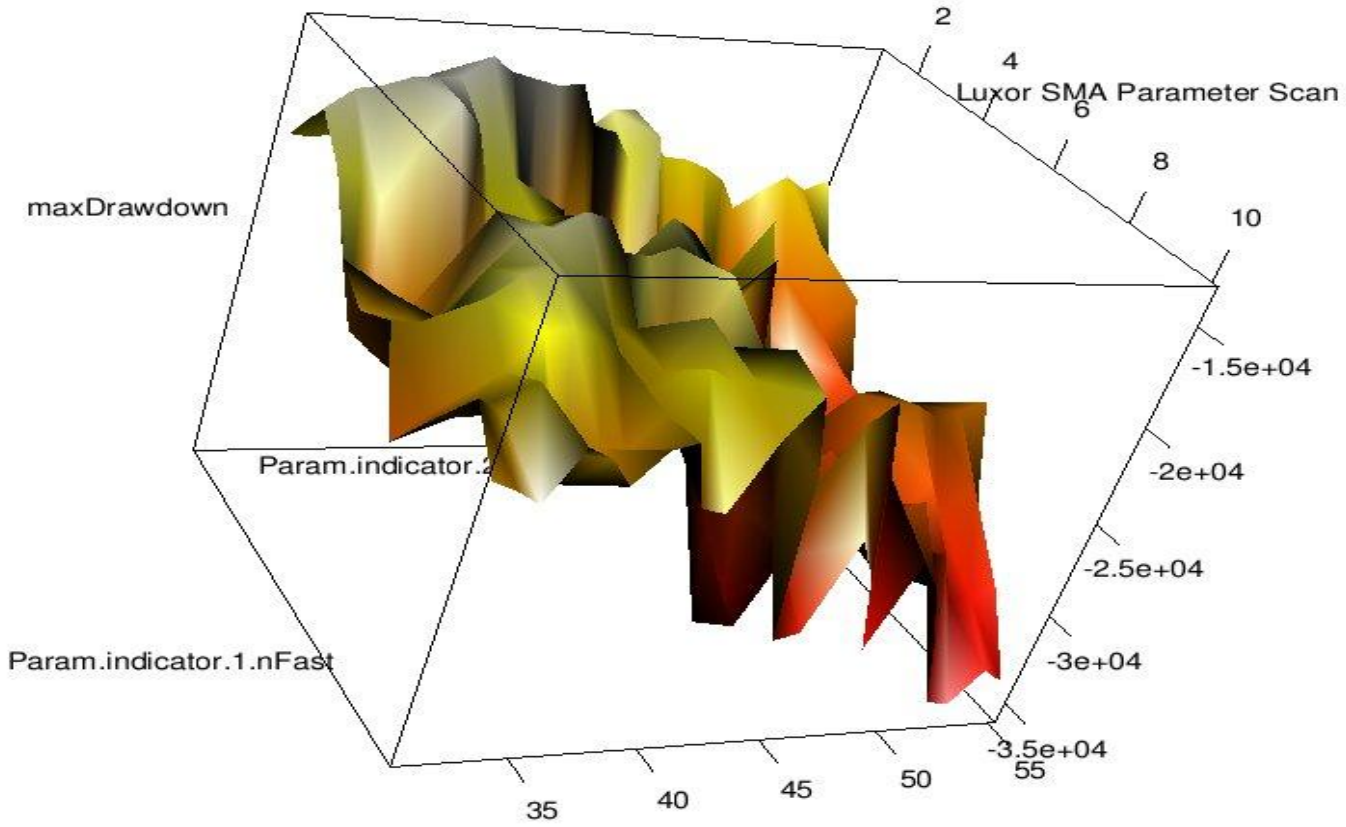
	row.names	1	50	99	148	197	246	295	344	393	442	491	540
1	StopLossLONG	0.0005	0.0010	0.0015	0.0020	0.0025	0.0030	0.0035	0.0040	0.0045	0.0050	0.0055	0.0060
2	StopLossSHORT	0.0005	0.0010	0.0015	0.0020	0.0025	0.0030	0.0035	0.0040	0.0045	0.0050	0.0055	0.0060
3	Portfolio	forex.1	forex.50	forex.99	forex.148	forex.197	forex.246	forex.295	forex.344	forex.393	forex.442	forex.491	forex.540
4	Symbol	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD
5	Num.Txns	3188	3188	3188	3187	3187	3187	3187	3187	3187	3187	3187	3187
6	Num.Trades	1594	1594	1594	1593	1593	1593	1593	1593	1593	1593	1593	1593
7	Net.Trading.PL	-89410.726	-44186.223	1139.573	34354.403	55205.610	54165.462	61243.988	60672.648	59776.311	52253.139	53657.860	51535.035
8	Avg.Trade.PL	-56.092049	-27.720341	0.714914	21.702701	34.791971	34.139022	38.582541	38.223884	37.661212	32.938568	33.820377	32.487781
9	Med.Trade.PL	-104.8628	-190.7764	-272.8800	-327.5400	-266.0000	-226.0000	-206.0000	-196.0000	-196.0000	-196.0000	-196.0000	-196.0000
10	Largest.Winner	4994	4994	5124	5674	5674	5674	5674	5674	5674	5674	5674	5674
11	Largest.Loser	-1498.045	-1498.045	-1498.045	-1498.045	-1498.045	-1464.864	-1412.136	-1412.136	-1412.136	-1751.000	-1751.000	-1751.000
12	Gross.Profits	208486	286906	363053	420426	462955	472321	486143	491917	495512	495512	499629	499629
13	Gross.Losses	-297896.7	-331092.2	-361913.4	-385853.6	-407531.4	-417937.5	-424681.0	-431026.4	-435517.7	-443040.9	-445753.1	-447876.0
14	Std.Dev.Trade.PL	541.6761	606.2002	683.5233	748.7764	790.9457	803.3402	822.3976	832.7906	840.4954	847.0377	854.0916	856.6080
15	Percent.Positive	13.48808	20.38896	26.72522	31.19900	34.08663	35.34212	36.03264	36.34652	36.47207	36.47207	36.59761	36.59761
16	Percent.Negative	86.51192	79.61104	73.27478	68.80100	65.91337	64.65788	63.96736	63.65348	63.52793	63.52793	63.40239	63.40239
17	Profit.Factor	0.6998600	0.8665441	1.0031487	1.0895998	1.1359984	1.1301234	1.1447251	1.1412690	1.1377540	1.1184341	1.1208648	1.1155522
18	Avg.Win.Trade	969.7023	882.7877	852.2371	845.9276	852.5875	838.9361	846.9390	849.5976	852.8606	852.8606	856.9966	856.9966
19	Med.Win.Trade	704.0	634.0	575.5	564.0	564.0	554.0	564.0	564.0	564.0	564.0	572.0	572.0
20	Avg.Losing.Trade	-216.0237	-260.9080	-309.8574	-352.0562	-388.1251	-405.7646	-416.7625	-425.0753	-430.3534	-437.7874	-441.3397	-443.4415
21	Med.Losing.Trade	-108.2270	-200.6220	-285.0375	-369.9800	-445.8125	-476.0000	-406.0000	-386.0000	-386.0000	-386.0000	-384.5000	-379.5000
22	Avg.Daily.PL	-87.486033	-43.108510	1.083244	31.893360	50.111763	48.905991	54.439316	53.742849	52.811894	46.230078	47.425933	45.557249
23	Med.Daily.PL	-180.1375	-198.8600	-271.9075	-296.0000	-216.0000	-156.5000	-146.0000	-136.0000	-136.0000	-136.0000	-136.0000	-136.0000
24	Std.Dev.Daily.PL	644.2032	723.4102	809.1650	886.5554	919.8179	935.4111	947.4436	959.2027	968.8617	981.2656	984.0928	986.3786
25	Max.Drawdown	-91149.20	-48222.27	-29710.20	-16165.66	-14881.30	-16481.47	-17290.09	-18675.61	-19372.61	-20726.00	-21634.22	-20242.64
26	Profit.To.Max.Draw	-0.98092711	-0.91630329	0.03835629	2.12514690	3.70973008	3.28644549	3.54214388	3.24876374	3.08560944	2.52113959	2.48023085	2.54586533
27	Avg.WinLoss.Ratio	4.488869	3.383521	2.750417	2.402820	2.196682	2.067544	2.032186	1.998699	1.981768	1.948116	1.941807	1.932603
28	Med.WinLoss.Ratio	6.504846	3.160172	2.019033	1.524407	1.265106	1.163866	1.389163	1.461140	1.461140	1.461140	1.487646	1.507246
29	Max.Equity	1738.479	2451.558	27939.164	47163.523	64703.829	65291.513	70979.005	68556.282	67168.639	59807.182	61065.593	59563.060
30	Min.Equity	-89410.726	-45770.710	-1771.033	-1774.940	-2087.950	-1903.540	-1685.910	-1746.000	-1746.000	-1746.000	-1746.000	-1746.000
31	End.Equity	-89410.726	-44186.223	1139.573	34354.403	55205.610	54165.462	61243.988	60672.648	59776.311	52253.139	53657.860	51535.035

quantstrat::tradeGraphs()

```
### show trade graphs from stats
tradeGraphs (
        stats = stats,
        free.params = c(
                "Param.indicator.1.nFast",
                "Param.indicator.2.nSlow"
        ),
        statistics = c(
                "Net.Trading.PL",
                "maxDrawdown",
                "Avg.Trade.PL",
                 "Num.Trades",
                "Profit.Factor"
        ),
        title = 'Luxor SMA Parameter Scan'
)
```

- creates a 3-D rendering of the parameter surface
- ideally, the strategy will be 'parameter-robust'
  - relatively smooth
  - no huge peaks based on small changes
- *tradeGraphs*() can generate multiple charts with a single call
- requires '**rgl**' package

quantstrat::tradeGraphs()



quantstrat

Exit and risk management

Order Sets

- set consisting of one or more OCO orders
- OCO - one cancels other
  - if one gets filled, all others are cancelled
- identified by orderset name tag
- eg. a Stop-Loss order + a Take-Profit order

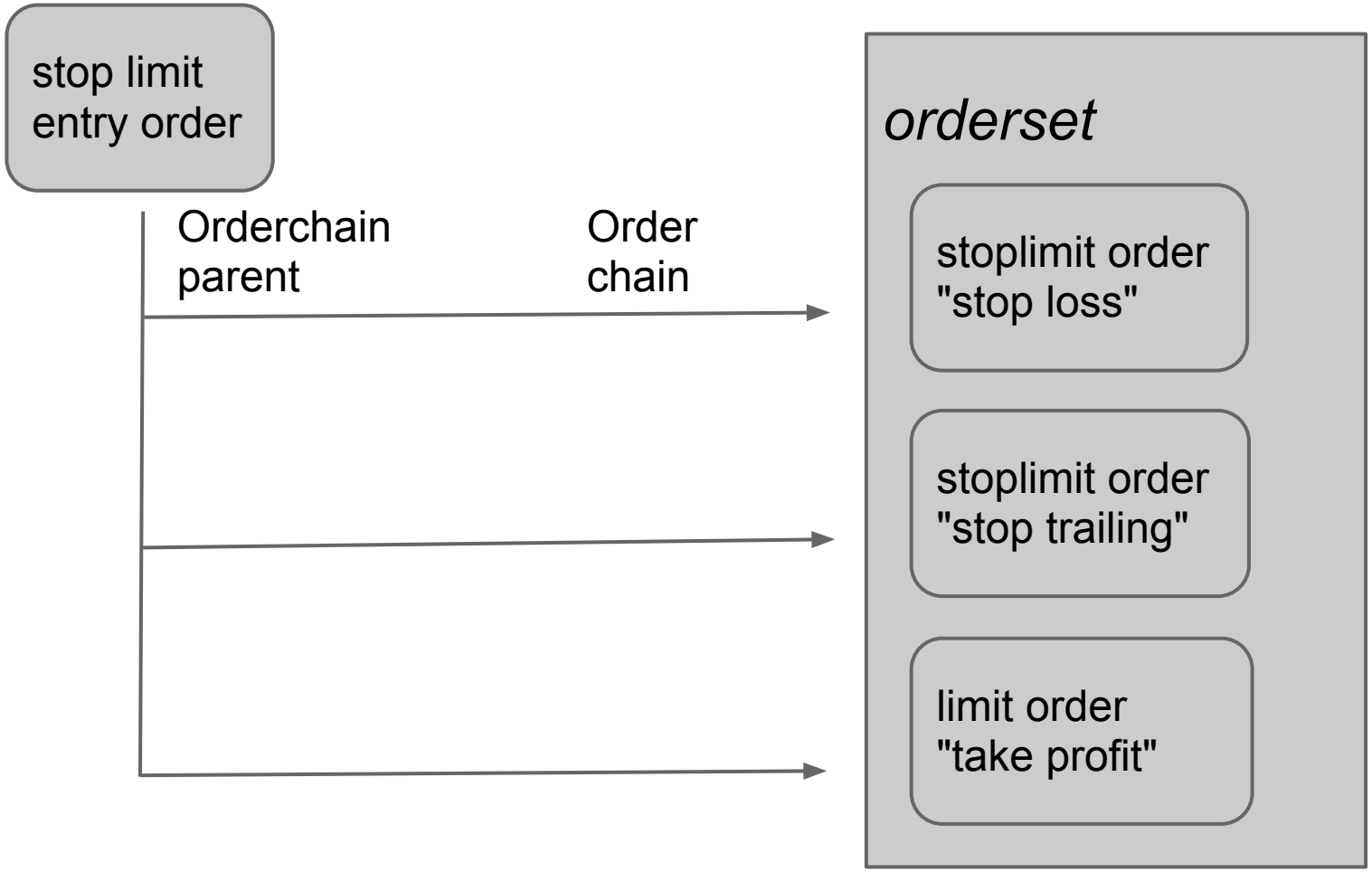
ruleSignal(..., orderset='ocolong', ...)

- commonly combined with order chains

Order Chains

- rule type='chain'
- fill of one order creates new order(s)
- parent order identified by "label"
- child orders use "parent" parameter to refer to parent
- commonly combined with order sets

Ordersets & Orderchains



Rule for Stop-loss

```
add.rule(strategy.st, name = 'ruleSignal',
    arguments=list(sigcol='long' , sigval=TRUE,
        replace=FALSE, orderside='long',
        ordertype='stoplimit', tmult=TRUE, threshold=quote(.stoploss),
        TxnFees=.txnfees, orderqty='all',
        orderset='ocolong'
    ),
    type='chain', parent='EnterLONG',
    label='StopLossLONG',
    enabled=FALSE
)
add.rule(strategy.st, name = 'ruleSignal',
    arguments=list(sigcol='short' , sigval=TRUE,
        replace=FALSE, orderside='short',
        ordertype='stoplimit', tmult=TRUE, threshold=quote(.stoploss),
        TxnFees=.txnfees, orderqty='all',
        orderset='ocoshort'
    ),
    type='chain', parent='EnterSHORT',
    label='StopLossSHORT',
    enabled=FALSE
)
```

- chain order
  - type
  - parent
- threshold
  - tmult
  - threshold
  - *quote()*
- enabled
- replace
  - must be FALSE (ordersets)

Order Sizing

Order Sizing, like everything else in quantstrat, is modular. You can write your own order sizing functions. Included are:

- *ruleSignal*() includes an osFUN argument
- *osNoOp*()
  - no operation, the default
- *osMaxPos*()
  - implements simple levels based maximums
  - works with *addPosLimit*() and *getPosLimit*()

addPosLimit() & osMaxPos()

```
addPosLimit(
        portfolio=portfolio.st,
        symbol='GBPUSD',
        timestamp=initDate,
        maxpos=.orderqty
)
add.rule(strategy.st, name = 'ruleSignal',
     arguments=list(sigcol='long' , sigval=TRUE,
          replace=FALSE,
          orderside='long' ,
          ordertype='stoplimit',
          prefer='High',
          threshold=.threshold,
          TxnFees=0,
          orderqty=+.orderqty,
          osFUN=osMaxPos,
          orderset='ocolong'
     ),
     type='enter',
     timespan = .timespan,
     label='EnterLONG'
)
```

- separates maximum positions from the rules
- makes decisions based on current position, not open orders
- needed with ordersets
- other order sizing functions are possible, *osMaxPos()* is an example

Trade optimization approaches

- paramsets
  - a "brute force" approach
  - covers a large part of the total parameter space

- MAE/MFE - "graphs"
- quantiles - "statistic"
  - use the data from a backtest or production trade to evaluate where stop loss or profit taking orders could be added

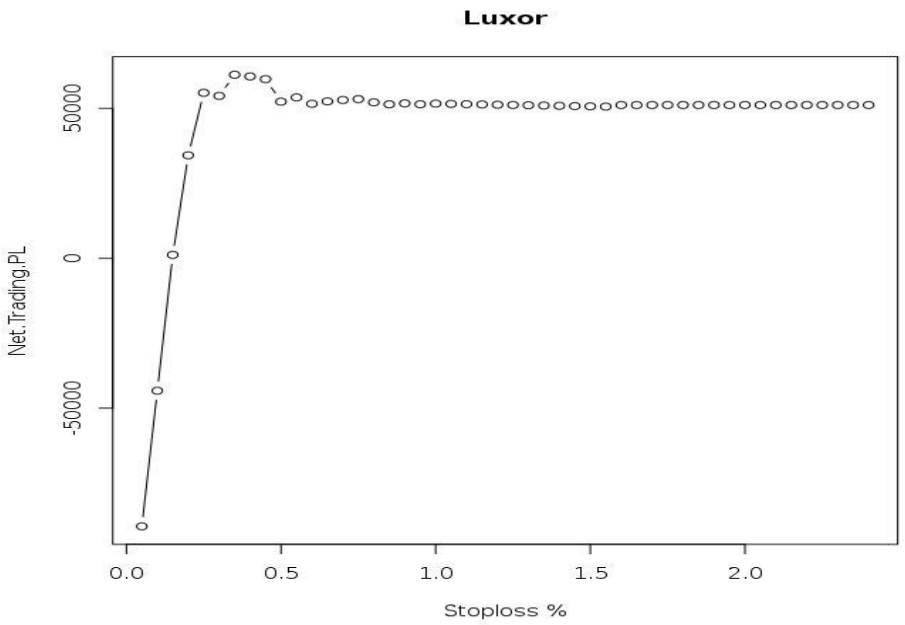
- walk forward
  - utilize an objective function to change parameters through time

Stop-loss using paramsets

```
.StopLoss = seq(0.05, 2.4, length.out=48)/100
```

```
add.distribution(strategy.st,
        paramset.label = 'StopLoss',
        component.type = 'chain',
        component.label = 'StopLossLONG',
        variable = list(threshold = .StopLoss),
        label = 'StopLossLONG'
)

add.distribution(strategy.st,
        paramset.label = 'StopLoss',
        component.type = 'chain',
        component.label = 'StopLossSHORT',
        variable = list(threshold = .StopLoss),
        label = 'StopLossSHORT'
)

add.constraint(strategy.st,
        paramset.label = 'StopLoss',
        distribution.label.1 = 'StopLossLONG',
        distribution.label.2 = 'StopLossSHORT',
        operator = '==',
        label = 'StopLoss'
)
```

```
require(foreach)


require(doMC)
registerDoMC(cores=8)


results <- apply.paramset(strategy.st, paramset.label='StopLoss',
portfolio.st=portfolio.st, account.st=account.st, nsamples=80,
verbose=TRUE)


plot(100*results$tradeStats$StopLossLONG, stats$Net.Trading.PL, type='b',
xlab='Stoploss %', ylab='Net.Trading.PL', main='Luxor')
```



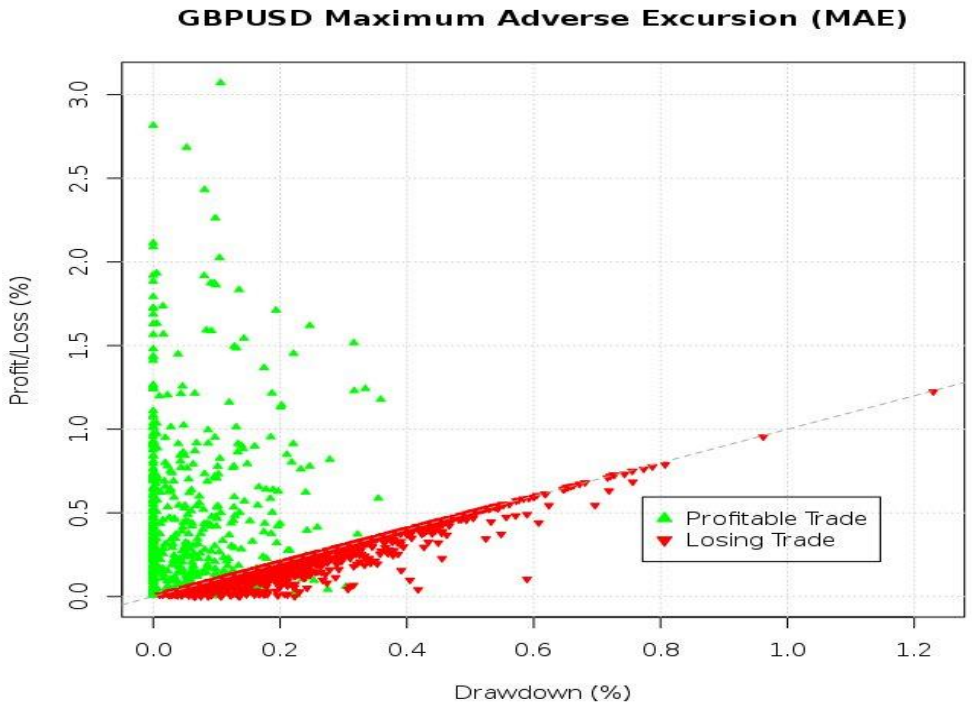
Stop-loss paramset *tradeStats()*

View(t(results$tradeStats))

	row.names	1	50	99	148	197	246	295	344	393	442	491	540
1	StopLossLONG	0.0005	0.0010	0.0015	0.0020	0.0025	0.0030	0.0035	0.0040	0.0045	0.0050	0.0055	0.0060
2	StopLossSHORT	0.0005	0.0010	0.0015	0.0020	0.0025	0.0030	0.0035	0.0040	0.0045	0.0050	0.0055	0.0060
3	Portfolio	forex.1	forex.50	forex.99	forex.148	forex.197	forex.246	forex.295	forex.344	forex.393	forex.442	forex.491	forex.540
4	Symbol	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD	GBPUSD
5	Num.Txns	3188	3188	3188	3187	3187	3187	3187	3187	3187	3187	3187	3187
6	Num.Trades	1594	1594	1594	1593	1593	1593	1593	1593	1593	1593	1593	1593
7	Net.Trading.PL	-89410.726	-44186.223	1139.573	34354.403	55205.610	54165.462	61243.988	60672.648	59776.311	52253.139	53657.860	51535.035
8	Avg.Trade.PL	-56.092049	-27.720341	0.714914	21.702701	34.791971	34.139022	38.582541	38.223884	37.661212	32.938568	33.820377	32.487781
9	Med.Trade.PL	-104.8628	-190.7764	-272.8800	-327.5400	-266.0000	-226.0000	-206.0000	-196.0000	-196.0000	-196.0000	-196.0000	-196.0000
10	Largest.Winner	4994	4994	5124	5674	5674	5674	5674	5674	5674	5674	5674	5674
11	Largest.Loser	-1498.045	-1498.045	-1498.045	-1498.045	-1498.045	-1464.864	-1412.136	-1412.136	-1412.136	-1751.000	-1751.000	-1751.000
12	Gross.Profits	208486	286906	363053	420426	462955	472321	486143	491917	495512	495512	499629	499629
13	Gross.Losses	-297896.7	-331092.2	-361913.4	-385853.6	-407531.4	-417937.5	-424681.0	-431026.4	-435517.7	-443040.9	-445753.1	-447876.0
14	Std.Dev.Trade.PL	541.6761	606.2002	683.5233	748.7764	790.9457	803.3402	822.3976	832.7906	840.4954	847.0377	854.0916	856.6080
15	Percent.Positive	13.48808	20.38896	26.72522	31.19900	34.08663	35.34212	36.03264	36.34652	36.47207	36.47207	36.59761	36.59761
16	Percent.Negative	86.51192	79.61104	73.27478	68.80100	65.91337	64.65788	63.96736	63.65348	63.52793	63.52793	63.40239	63.40239
17	Profit.Factor	0.6998600	0.8665441	1.0031487	1.0895998	1.1359984	1.1301234	1.1447251	1.1412690	1.1377540	1.1184341	1.1208648	1.1155522
18	Avg.Win.Trade	969.7023	882.7877	852.2371	845.9276	852.5875	838.9361	846.9390	849.5976	852.8606	852.8606	856.9966	856.9966
19	Med.Win.Trade	704.0	634.0	575.5	564.0	564.0	554.0	564.0	564.0	564.0	564.0	572.0	572.0
20	Avg.Losing.Trade	-216.0237	-260.9080	-309.8574	-352.0562	-388.1251	-405.7646	-416.7625	-425.0753	-430.3534	-437.7874	-441.3397	-443.4415
21	Med.Losing.Trade	-108.2270	-200.6220	-285.0375	-369.9800	-445.8125	-476.0000	-406.0000	-386.0000	-386.0000	-386.0000	-384.5000	-379.5000
22	Avg.Daily.PL	-87.486033	-43.108510	1.083244	31.893360	50.111763	48.905991	54.439316	53.742849	52.811894	46.230078	47.425933	45.557249
23	Med.Daily.PL	-180.1375	-198.8600	-271.9075	-296.0000	-216.0000	-156.5000	-146.0000	-136.0000	-136.0000	-136.0000	-136.0000	-136.0000
24	Std.Dev.Daily.PL	644.2032	723.4102	809.1650	886.5554	919.8179	935.4111	947.4436	959.2027	968.8617	981.2656	984.0928	986.3786
25	Max.Drawdown	-91149.20	-48222.27	-29710.20	-16165.66	-14881.30	-16481.47	-17290.09	-18675.61	-19372.61	-20726.00	-21634.22	-20242.64
26	Profit.To.Max.Draw	-0.98092711	-0.91630329	0.03835629	2.12514690	3.70973008	3.28644549	3.54214388	3.24876374	3.08560944	2.52113959	2.48023085	2.54586533
27	Avg.WinLoss.Ratio	4.488869	3.383521	2.750417	2.402820	2.196682	2.067544	2.032186	1.998699	1.981768	1.948116	1.941807	1.932603
28	Med.WinLoss.Ratio	6.504846	3.160172	2.019033	1.524407	1.265106	1.163866	1.389163	1.461140	1.461140	1.461140	1.487646	1.507246
29	Max.Equity	1738.479	2451.558	27939.164	47163.523	64703.829	65291.513	70979.005	68556.282	67168.639	59807.182	61065.593	59563.060
30	Min.Equity	-89410.726	-45770.710	-1771.033	-1774.940	-2087.950	-1903.540	-1685.910	-1746.000	-1746.000	-1746.000	-1746.000	-1746.000
31	End.Equity	-89410.726	-44186.223	1139.573	34354.403	55205.610	54165.462	61243.988	60672.648	59776.311	52253.139	53657.860	51535.035

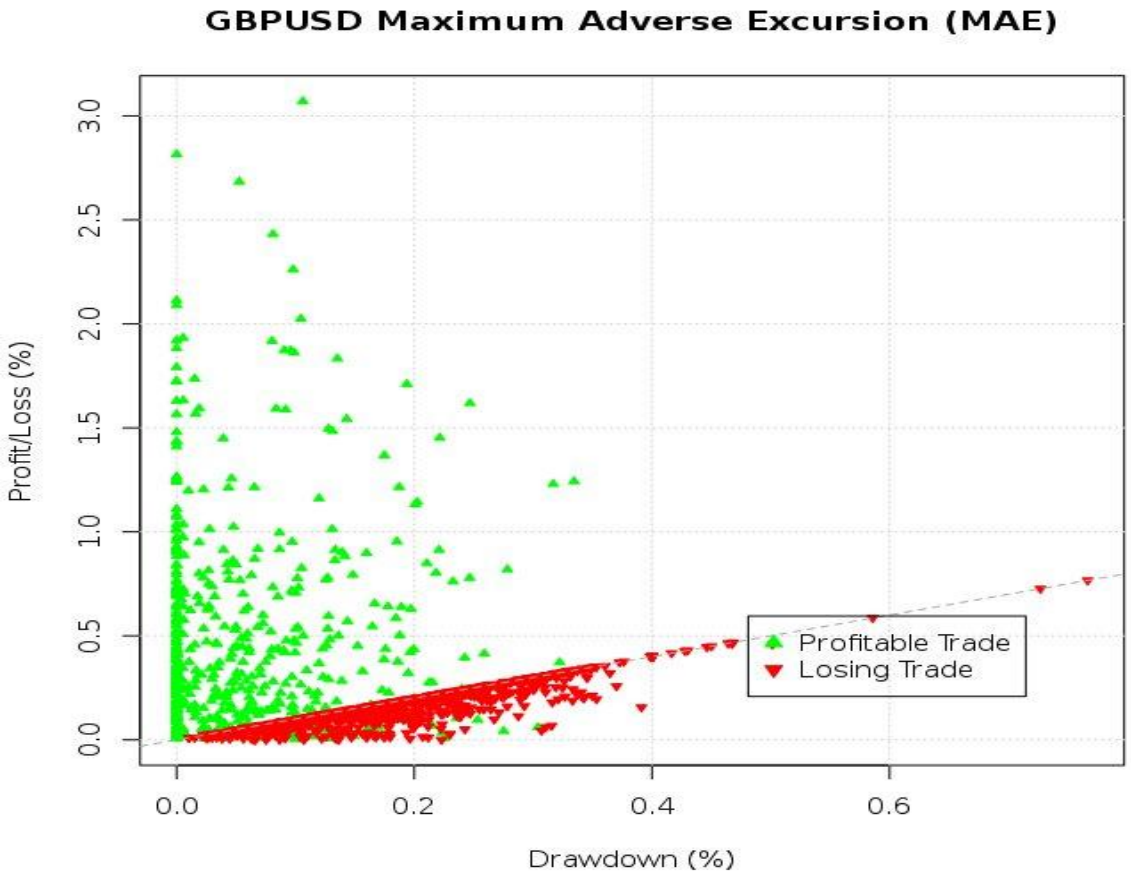
MAE: Max Adverse Excursion

```
chart.ME(
    Portfolio='forex',
    Symbol='GBPUSD',
    type='MAE',
    scale='percent'
)
```



- trades on diagonal axis closed on their lows
- trades on vertical axis experienced no drawdown
- *Chart.ME* type
  - MAE
  - MFE
- scale
  - cash
  - percent
  - tick

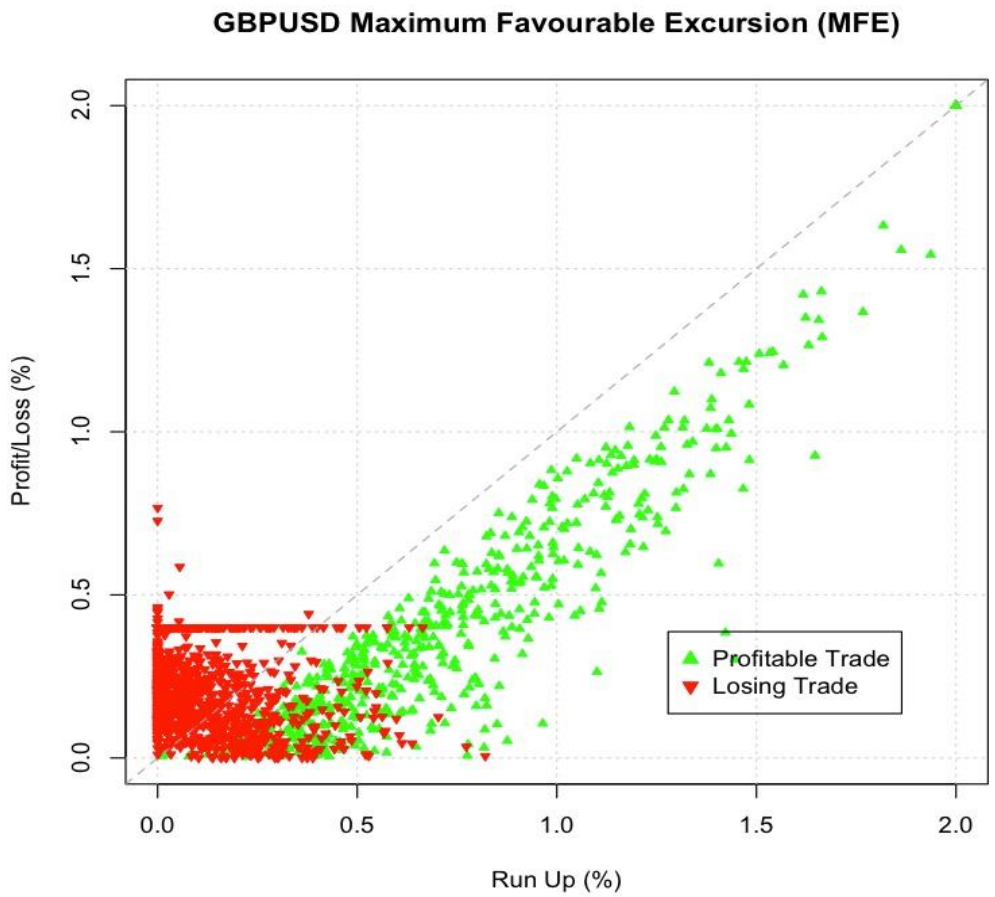
MAE using 0.4% stop-loss



	row.names	GBPUSD
1	Portfolio	forex
2	Symbol	GBPUSD
3	Num.Txns	3357
4	Num.Trades	1678
5	Net.Trading.PL	79167.54
6	Avg.Trade.PL	47.30962
7	Med.Trade.PL	-196
8	Largest.Winner	4190.74
9	Largest.Loser	-1412.136
10	Gross.Profits	512579.2
11	Gross.Losses	-433193.7
12	Std.Dev.Trade.PL	823.4333
13	Percent.Positive	35.39928
14	Percent.Negative	64.60072
15	Profit.Factor	1.183256
16	Avg.Win.Trade	862.9279
17	Med.Win.Trade	584
18	Avg.Losing.Trade	-399.6251
19	Med.Losing.Trade	-366
20	Avg.Daily.PL	68.31802
21	Med.Daily.PL	-126
22	Std.Dev.Daily.PL	969.0271
23	Max.Drawdown	-15088.95
24	Profit.To.Max.Draw	5.246723
25	Avg.WinLoss.Ratio	2.159343
26	Med.WinLoss.Ratio	1.595628
27	Max.Equity	86145.67
28	Min.Equity	-2135.2
29	End.Equity	79167.54

MFE: Max Favorable Excursion

```
chart.ME(Portfolio='luxor', Symbol='GBPUSD', type='MFE', scale='percent')
```

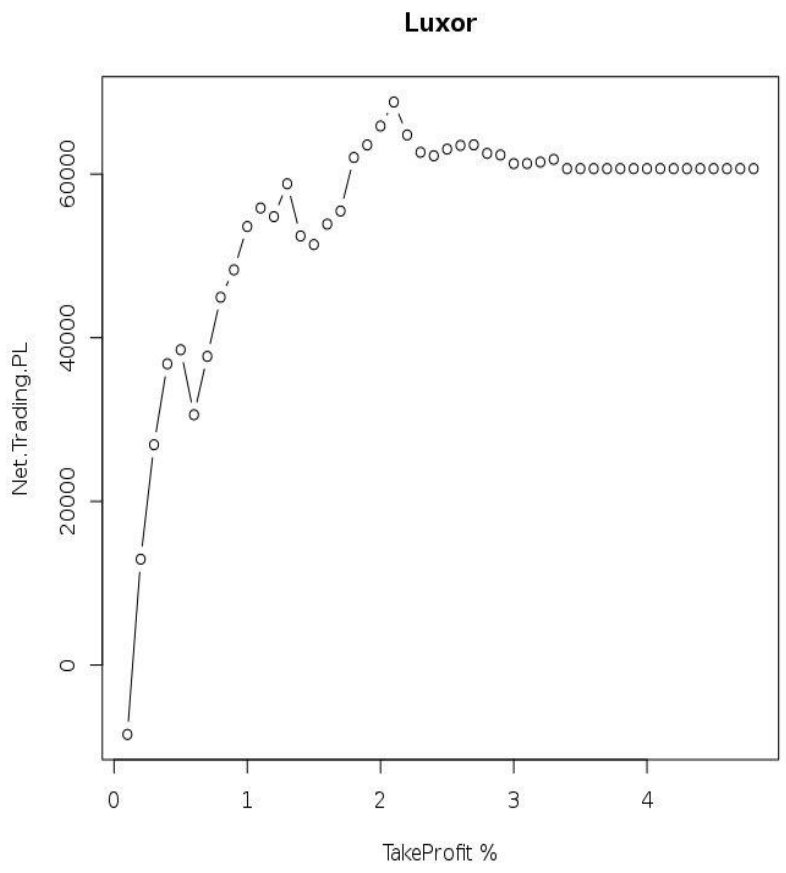


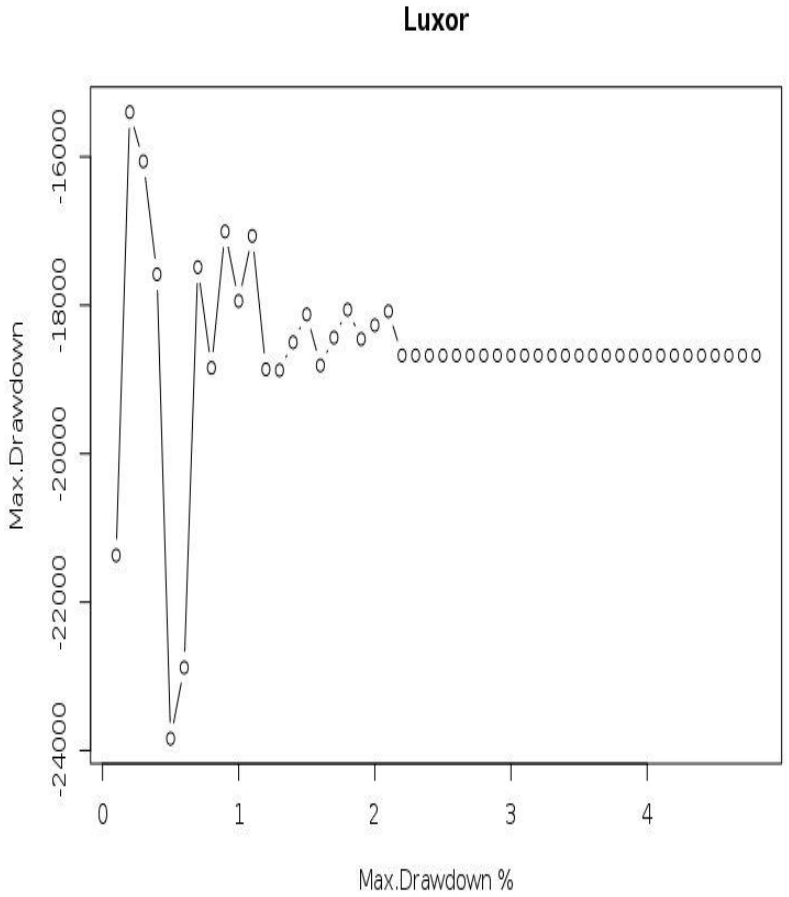
- trades on the diagonal axis closed on their highs
- trades on the vertical axis experienced no positive P&L
- the horizontal line demonstrates the influence of a stop in this data set

Take-Profit using paramsets

plot(100*stats$TakeProfitLONG, stats$Net.Trading.PL, type='b', xlab='TakeProfit %', ylab='Net.Trading.PL', main='Luxor')

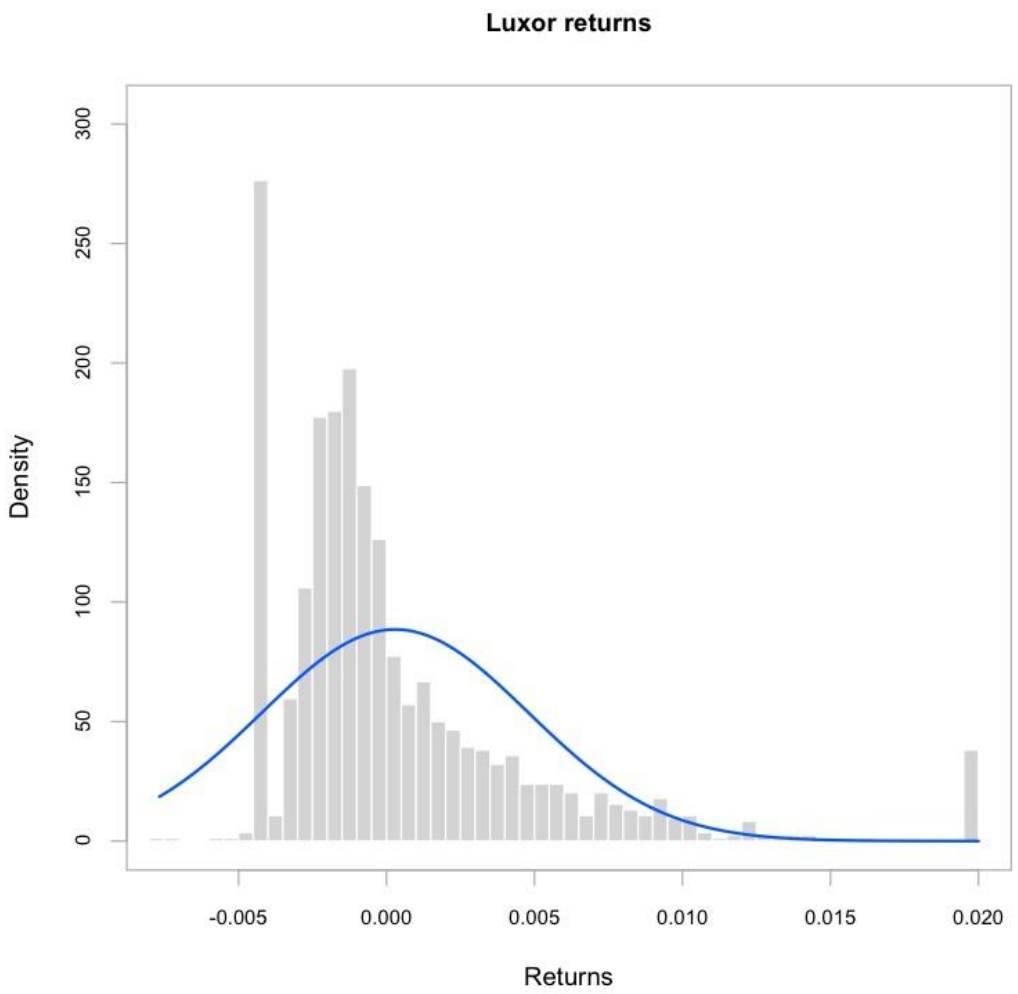
plot(100*stats$TakeProfitLONG, stats$Max.DrawdownL, type='b', xlab=Max.Drawdown %', ylab='Max.Drawdown', main='Luxor')





PerformanceAnalytics::chart.Histogram

```
> ps <- perTradeStats('forex', Symbol='GBPUSD')
> chart.Histogram(ps$Pct.Net.Trading.PL, methods=c('add.normal'), main='Luxor returns')
```



Notice peaks:

- Stop Loss
- Take Profit

Other obs:

- Negative skew
- Excess positive kurtosis

blotter::tradeQuantiles()

```
> tradeQuantiles('forex', Symbol='GBPUSD', scale='percent', probs=c(.5,.9,.99))

                forex.GBPUSD
posPctPL 50%        0.003148
posPctPL 90%        0.010359
posPctPL 99%        0.019000
negPctPL 50%       -0.002500
negPctPL 90%       -0.002500
negPctPL 99%       -0.002500
posPctMFE 50%       0.006655
posPctMFE 90%       0.014280
posPctMFE 99%       0.019000
posPctMAE 50%      -0.000297
posPctMAE 90%      -0.001337
posPctMAE 99%      -0.002050
negPctMFE 50%       0.000623
negPctMFE 90%       0.003038
negPctMFE 99%       0.005597
negPctMAE 50%      -0.002500
negPctMAE 90%      -0.002500
negPctMAE 99%      -0.002500
%MAE~max(cum%PL)   -0.001438
```

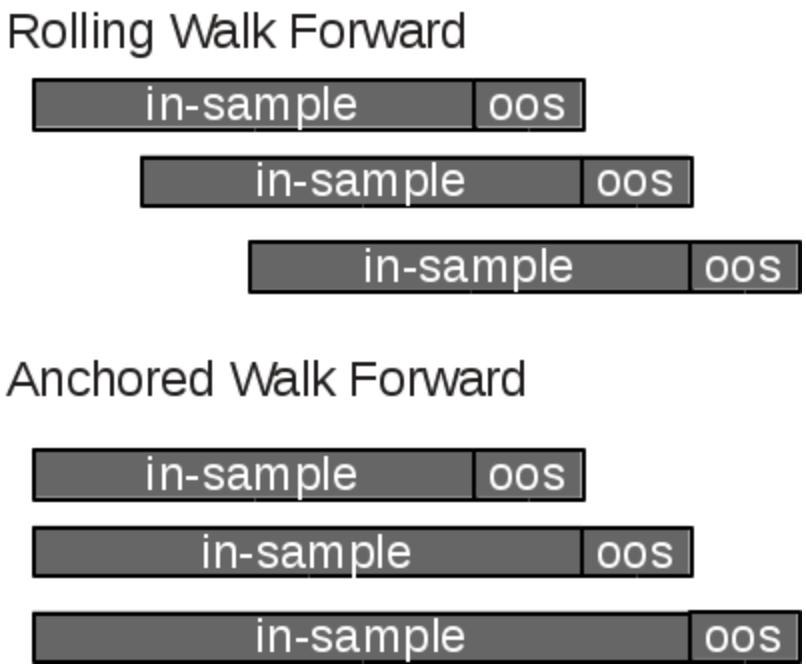
- *tradeQuantiles*() calculates and returns statistics for quantiles of
  - positive and negative
    - P&L
    - MAE
    - MFE
- %MAE~max line captures the peak of the distribution (with perfect hindsight)
- other options include quantiles as ticks, or cash P&L

quantstrat

Walk Forward Analysis

Walk Forward Analysis

- walk forward analysis periodically reparameterizes the strategy
- in production, this could be done daily, weekly, monthly, or quarterly
- rolling window analysis is the most common, assumes evolving parameter space
- anchored analysis assumes that the residual information from earlier periods helps you make decisions now



walk.forward()

```
ess <- function(account.st, portfolio.st)
{
   require(robustbase, quietly=TRUE)
   require(PerformanceAnalytics, quietly=TRUE)
   portfolios.st <- ls(pos=.blotter, pattern=paste('portfolio', portfolio.st, '[0-9]*',sep='.'))
   pr <- PortfReturns(Account = account.st, Portfolios=portfolios.st)
   return(ES(R=pr, clean='boudt'))
}
my.obj.func <- function(x)
{
   return(which(max(x$GBPUSD) == x$GBPUSD))
}
walk.forward(
      strategy.st, paramset.label='WFA',
      portfolio.st=portfolio.st, account.st=account.st,
      period='months', k.training=3, k.testing=1,
      obj.func=my.obj.func, obj.args=list(x=quote(result$apply.paramset)),
      user.func=ess, user.args=list(''account.st'=account.st, 'portfolio.st'=portfolio.st),
      audit.prefix='wfa.ples',
      anchored=FALSE
)
chart.Posn(portfolio.st)
View(t(tradeStats(portfolio.st)))
```

- paramset
- period
- objective function (runs on master)
- optional user function (runs on slaves)
- audit.prefix
- anchored

walk.forward() graphs

chart.forward.testing('wfa.Net.Trading.PL.results.RData')
RData')

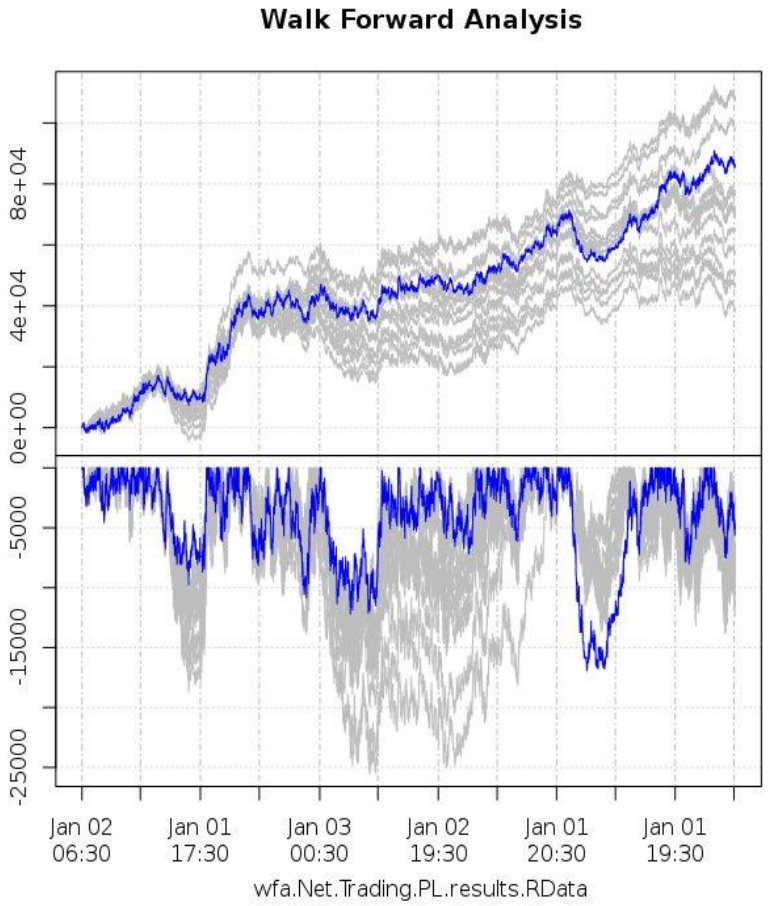
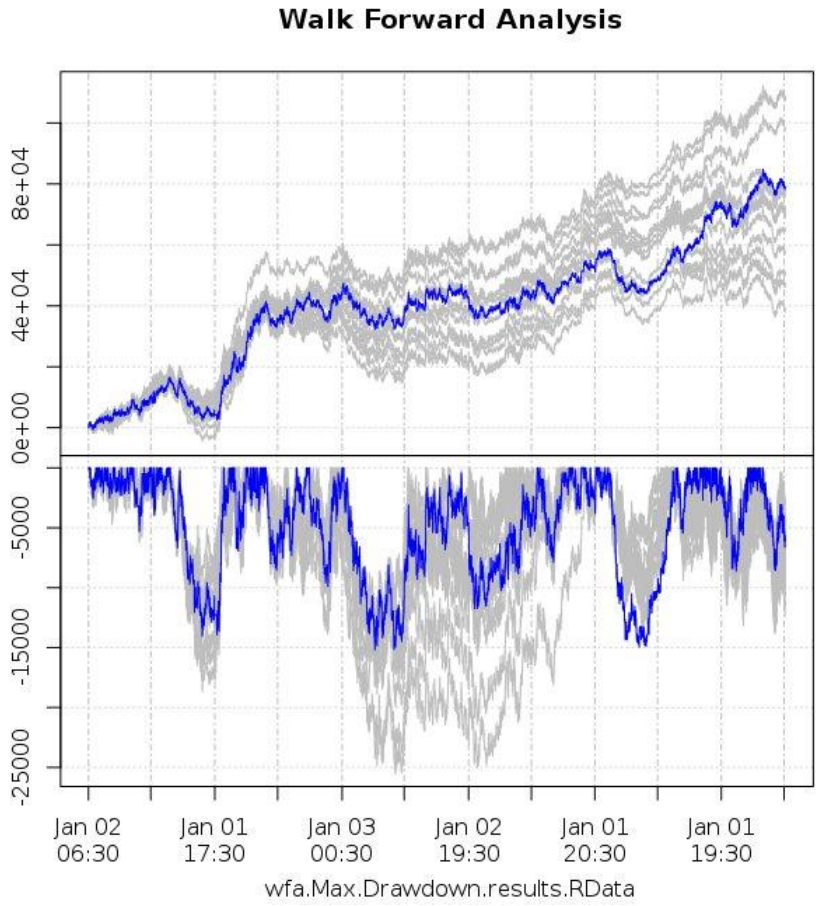
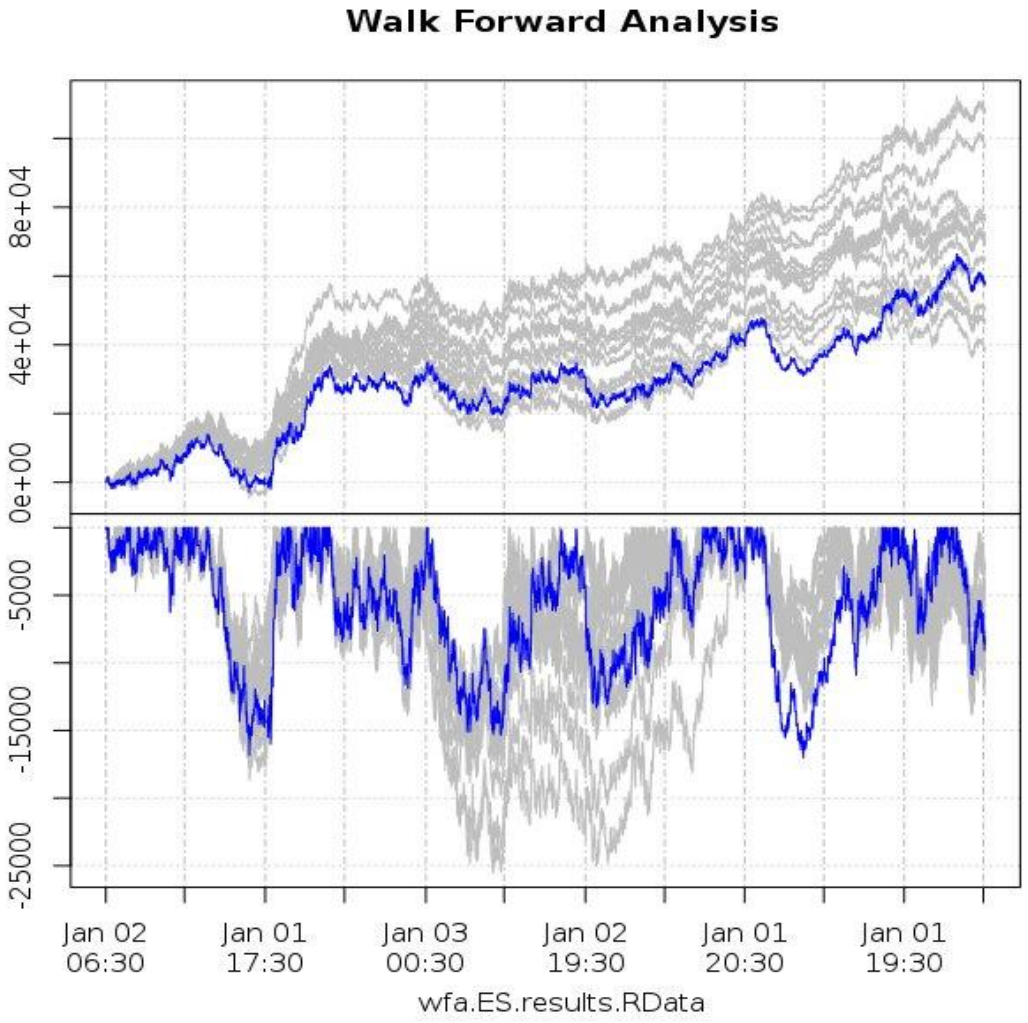


chart.forward.testing('wfa.Max.Drawdown.results.



walk.forward() using ES



- choice of your objective function is very important to walk forward analysis
- in this case, the objective function chooses rather poor out-of-sample parameters
- chasing highest in-sample End.Eq doesn't work...

Appendix

Other Things we likely don't have time for

Speed: making it faster

- quantstrat's speed is a nearly linear function of the number of rule evaluations
- this means that you should take care not to add unnecessary rules
- you should also consider pre-processing your signals via 'logical and' type operations
- it is feasible to get speeds of 1 core minute per symbol per day on high frequency data

Analyzing Returns

- quantstrat (and blotter) work in "price and trade space"
- sometimes you need to do analysis in "returns space"
- function *PortfReturns*() generates returns from portfolio equity
- note that this is sensitive to starting equity
- see `demo('rsi')` and `demo('faber')` for examples
- makes all the returns based analysis of packages like **PerformanceAnalytics** or **PortfolioAnalytics** available

Rebalancing

- how do you allocate capital among different instruments in a strategy?
- periodically apply a rebalancing rule
- quantstrat includes special rule type 'rebalance' and a special version of *applyStrategy*() : *applyStrategy.rebalancing*()
- example rebalance rules *rulePctEquity* and *ruleWeight* are provided
- see `demo('faber_rebal')` for example
- allocation across strategies probably requires **PortfolioAnalytics**, not just rebalancing rules

Further Work

stuff we're working on or would like to

- compiled applyRules
  - move the path-dependent state machine loop to faster compiled code

- strategy initialization and wrapup
  - cuts down on 'boilerplate' code
  - would reduce code duplication in e.g. paramsets

- Monte Carlo or Burns-style random trades analysis

- generalized objective for walk forward analysis

- Encyclopedia of Trading Strategies
  - get involved! write demos we can include in the package

Conclusion

- pursue a scientific approach to backtesting:
  - guess
  - build a model
  - test your hypothesis through experiment

- overfitting remains a danger

- try to make as many of your assumptions as possible *explicit* rather than *implicit*

- a backtest that makes money gives you a *possibility* of making money in production with your model,

- a losing backtest is almost a guarantee of *larger* production losses

Summary

- Strategy creation
- R tools
- Luxor strategy: simple implementation
- Optimizing parameters
- Exit and risk management
- Walk forward analysis
- Appendix

Thank You for Your Attention

Bios

Jan Humme

- independent financial trader
- computer scientist and software developer
- quantstrat package co-author
- jan@opentrades.nl

Brian G. Peterson

- Partner and Director of Quantitative Trading at DV Trading in Chicago
- author or co-author of over 10 R packages for Finance
- regularly writes and speaks on portfolio construction and risk
- brian@braverock.com

Resources

- R-Forge
  - https://r-forge.r-project.org/projects/blotter/
- R-SIG-Finance mailing list
  - https://stat.ethz.ch/mailman/listinfo/r-sig-finance
- Blogs
  - http://blog.fosstrading.com/
  - http://timelyportfolio.blogspot.com/
  - http://quantistat.blogspot.com/
- github, StackOverflow, etc.
- email us

Link to Slides

Latest slides available on Google Drive here:

https://docs.google.com/presentation/d/1fGzDc-LFfCQJKHHzaonspuX1_TTm1EB5hlvCEDsz7zw/pub?