Round Turn Trade Simulation - R/Finance 2018 CFP Submission

Jasen Mackie & Brian G. Peterson

updated 31 January 2018

This is a proposal for a talk at R/Finance 2018 to present analysis of simulations based on stylized facts of observed round turn trades. The article below (available as an Rmd markdown file) contains all the code and descriptive text to explain and replicate the analysis we are proposing to cover in the presentation. For the conference, we propose to create an Rmd markdown presentation containing all code to replicate the analysis using the *blotter* package.

Back in August I wrote a post about a new function in the R blotter package called mcsim(). The objective for writing that post was to introduce the function, and how and why it may be useful when evaluating quantitative trading strategies. In this article I aim to achieve the same goal for a new function named txnsim(), with a similar application but with more analytical value.

With mcsim() an analyst is able to simulate a trading strategy's portfolio PL which will give a person some information on the statistical properties of the strategy overall. Simulating portfolio PL can have drawbacks; in particular that simulations on a portfolio level may lack transparency or may not line up with historical market regimes. In these instances sampling round turn trades may be more useful.

With txnsim() the analyst is able to construct a random strategy that preserves as many of the stylized facts (or style) of the observed strategy as possible, while demonstrating no skill. The round turn trades of the random replicate strategies, while outwardly resembling the original strategy in summary time series statistics, are the result of random combinations of observed features taking place at random times in the tested time period. This effectively creates simulated traders with the same style but without skill. For this reason txnsim() is most appropriate for discerning skill vs. luck or overfitting.

Stylized facts

If you consider the stylized facts of a series of transactions that are the output of a discretionary or systematic trading strategy, it should be clear that there is a lot of information available to work with. The stylized facts txnsim() uses for simulating round turns include;

- percent time in market (and percent time flat)
- ratio of long to short position taking (in duration terms)
- number of levels or layered trades observed, limited by max position

Using these stylized facts, txnsim() samples either with or without replacement between flat periods, short periods and long periods and then layers onto these periods the sampled quantities from the original strategy with their respective durations.

Round Turn Trades & tradeDef

In order to sample round turn trades, the analyst first needs to define what a round turn trade is for their purposes. In txnsim() there is a parameter named *tradeDef* which can take one of 3 arguments, 1. "flat.to.flat", 2. "flat.to.reduced", 3. "increased.to.reduced". The argument is subsequently passed to the blotter::perTradeStats() function from which we extract the original strategy's stylized facts. The simplest definition of a round turn trade would be flat.to.flat and would include all transactions between when a position is opened and when it is closed. This method is most suitable for a strategy that only puts on a single level per round turn. This definition would not be suitable for a strategy that is rarely flat and it

would be safe to assume that most quantitative strategies in production would be using a variation of position sizing and/or risk management. In the case of flat.to.reduced a trade's initial entry is always paired with a transaction which takes the position closer to zero, regardless of any transactions which may have increased the position along the way.

When Brian and I started building txnsim() it became clear we needed a third definition for round turns, namely increased.to.reduced. This was on Brian's TODO list for implementing in blotter in any event. For this definition every transaction that moves a position closer to zero will close the round turn. This round turn exit transaction will be paired with the one or more transactions which took the position further from zero, thereby locating the initiating transaction/s. This method is otherwise known as Average Cost First-in First-Out (ACFIFO), and is indeed how we built the method to account for PL inside perTradeStats().

Besides tradeDef the only other necessary arguments for txnsim() are:

- Portfolio - object stored in blotter
- n - number of replicates to sample
- replacement - boolean for sampling with or without replacement
- CI - Confidence Interval for distribution plot of summary statistics

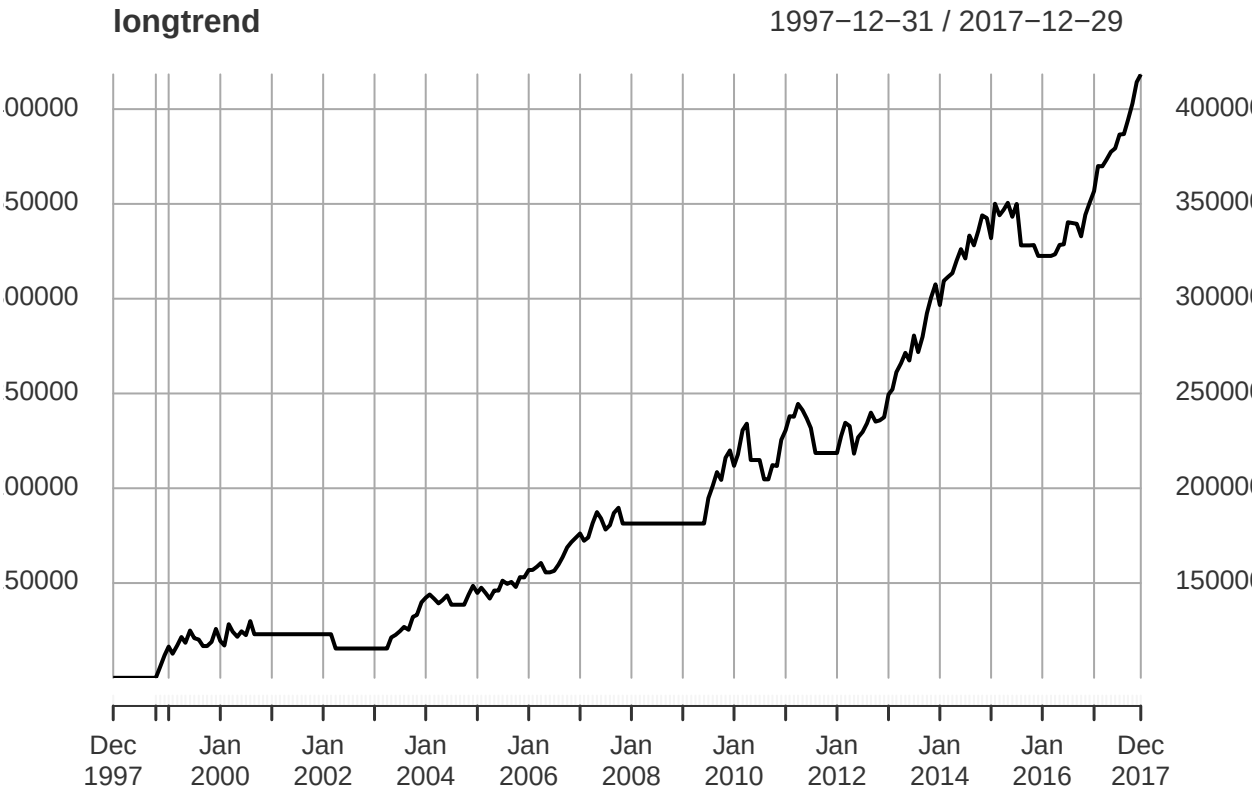
Output

To illustrate the output from txnsim() I will use 2 sample strategy backtests, the first being a slight variation of the 'longtrend' demo in the blotter package and the second a variation of the 'bbands' demo from the quantstrat package. In both instances I use a fixed end date (2017-12-31) for the purposes of replication. In addition, for bbands I vary the exit quantity of shares in order to generate a backtest with layers thereby illustrating how txnsim() honors these layers when building random replicates.

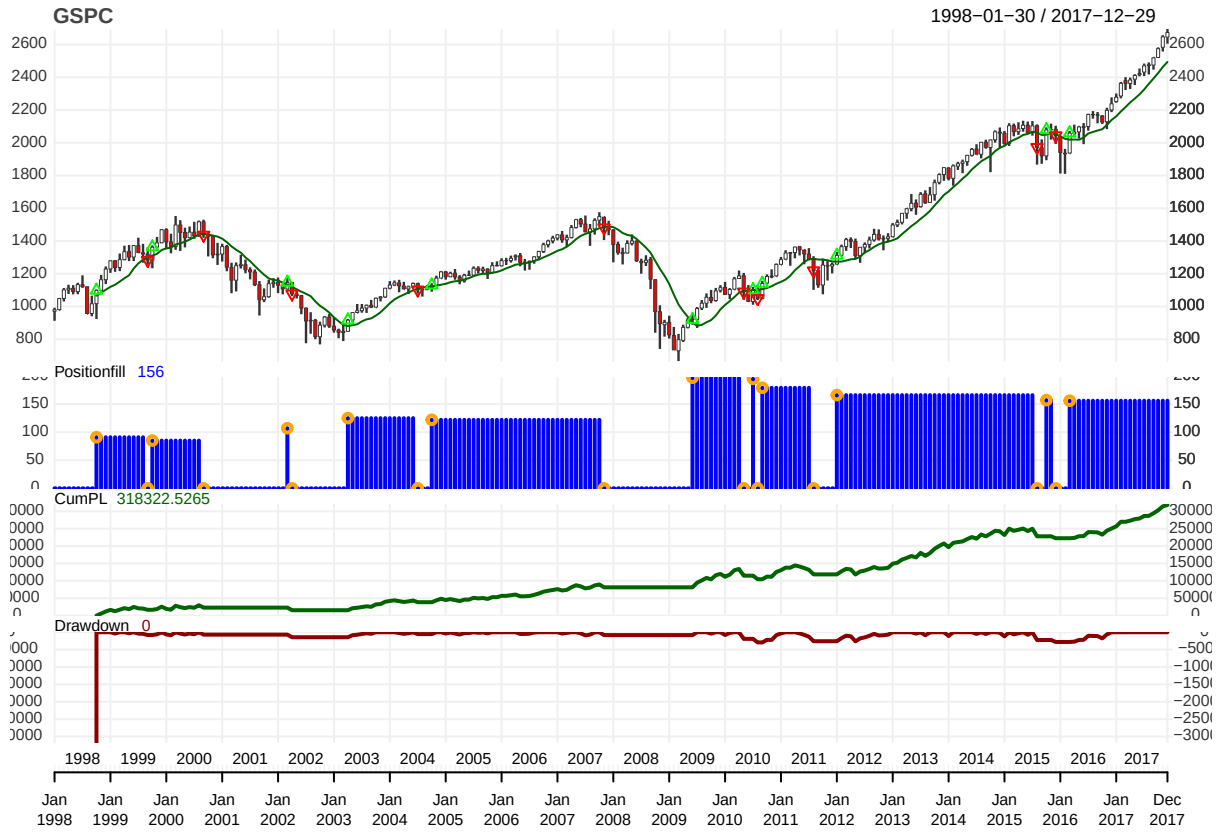
As with mcsim(), txnsim() uses S3 methods for plotting the replicate equity curves and summary statistic histograms. Before delving into those and the other methods and slots in txnsim(), a quick overview of the 'longtrend' strategy itself may be appropriate.

Based on the strategy covered in Faber's paper A Quantitative Approach to Tactical Asset Allocation Faber looks to test a trend following system similar to that covered by Jeremy Siegel in "Stocks for the Long Run". Siegel tests a simple 200-day moving average strategy on the DJIA since 1885 and concludes that using the long-term moving average strategy an investor is able to outperform a buy-and-hold strategy on a risk-adjusted basis after transaction costs. Faber tests a similar strategy but using monthly data and the 10-month moving average of the S&P500 since 1901, and since 1973 for the "Global Tactical Asset Allocation (GTAA)" portfolio. Their use of monthly data is due to a restriction on the availability of data for their extension of the strategy to multi-asset class portfolios. The lower periodicity however has the added benefit of reducing transaction costs. Faber's results show the timing model outperforming a buy-and-hold strategy on the S&P500 as well as when applied to an equally weighted portfolio of 5 different asset classes on a risk-adjusted and absolute returns basis.

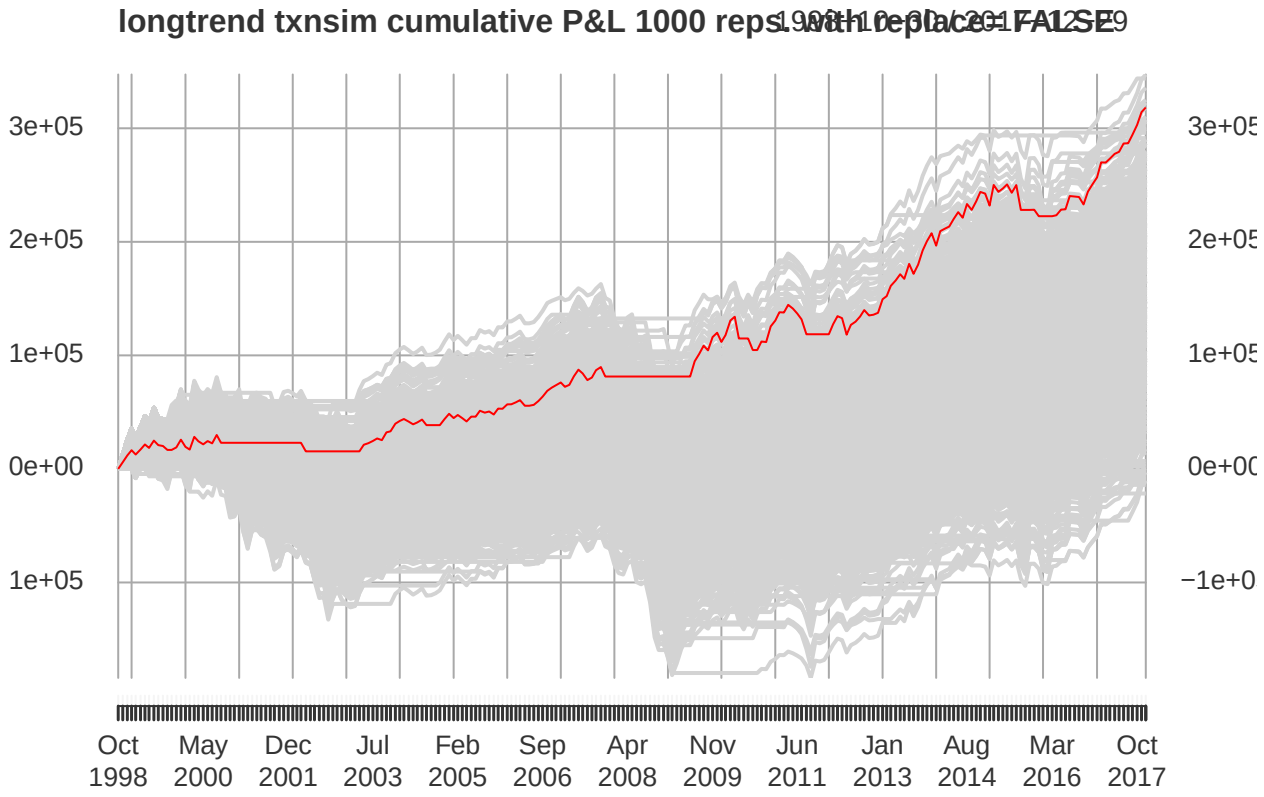
Taking a look at the 'longtrend' equity curve, its clear the strategy benefitted from being out the market during the protracted bear markets following the tech and housing bubbles. For this reason trend following systems add the most value when applied over entire busines cycles.

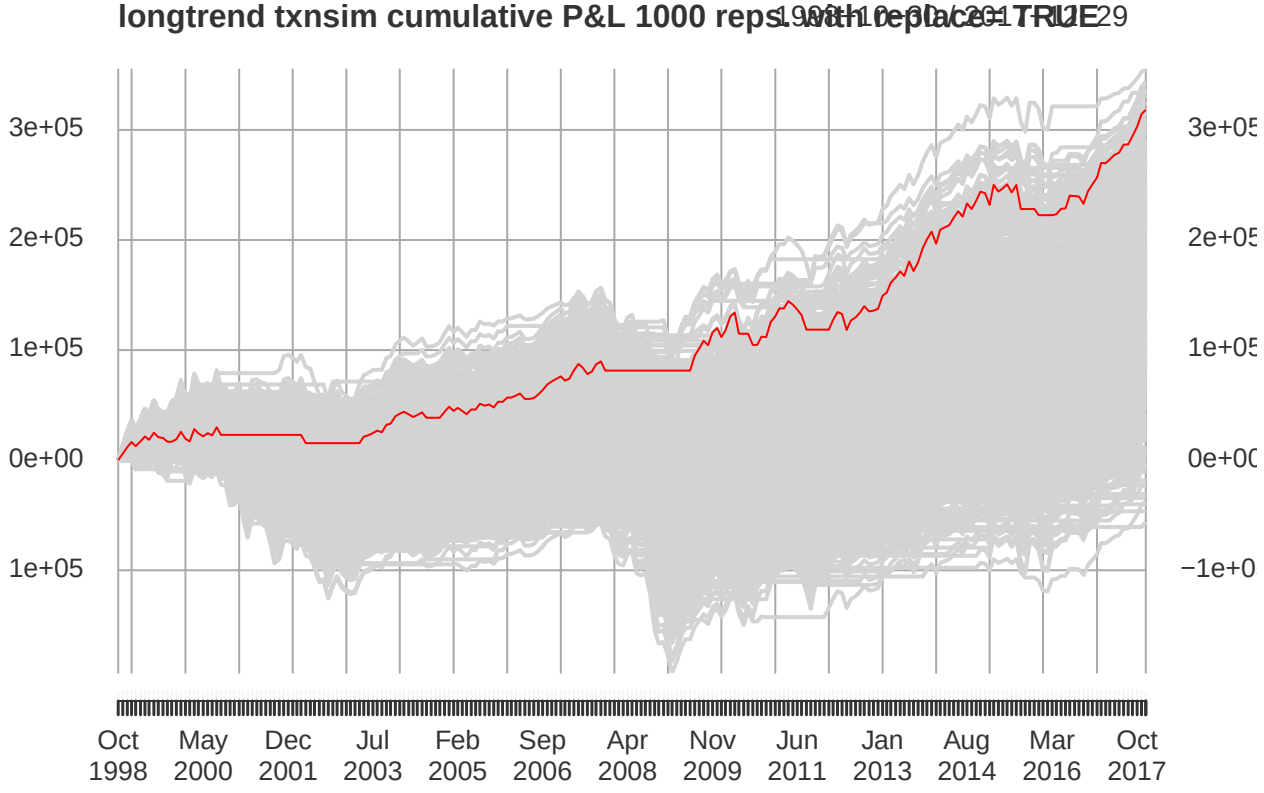


For a more holistic view of the strategy performance and time in the market we call *chart.Posn()*. The flat periods during the protracted bear markets should be more evident looking at the position fill window.



Using txnsim() we are able to measure the performance of any number of randomized versions of this 'longtrend' strategy (within reason of course, optimization is a TODO). Below is a visual comparison of the orignal strategy's equity curve and 1k random replicate equity curves with and without replacement.

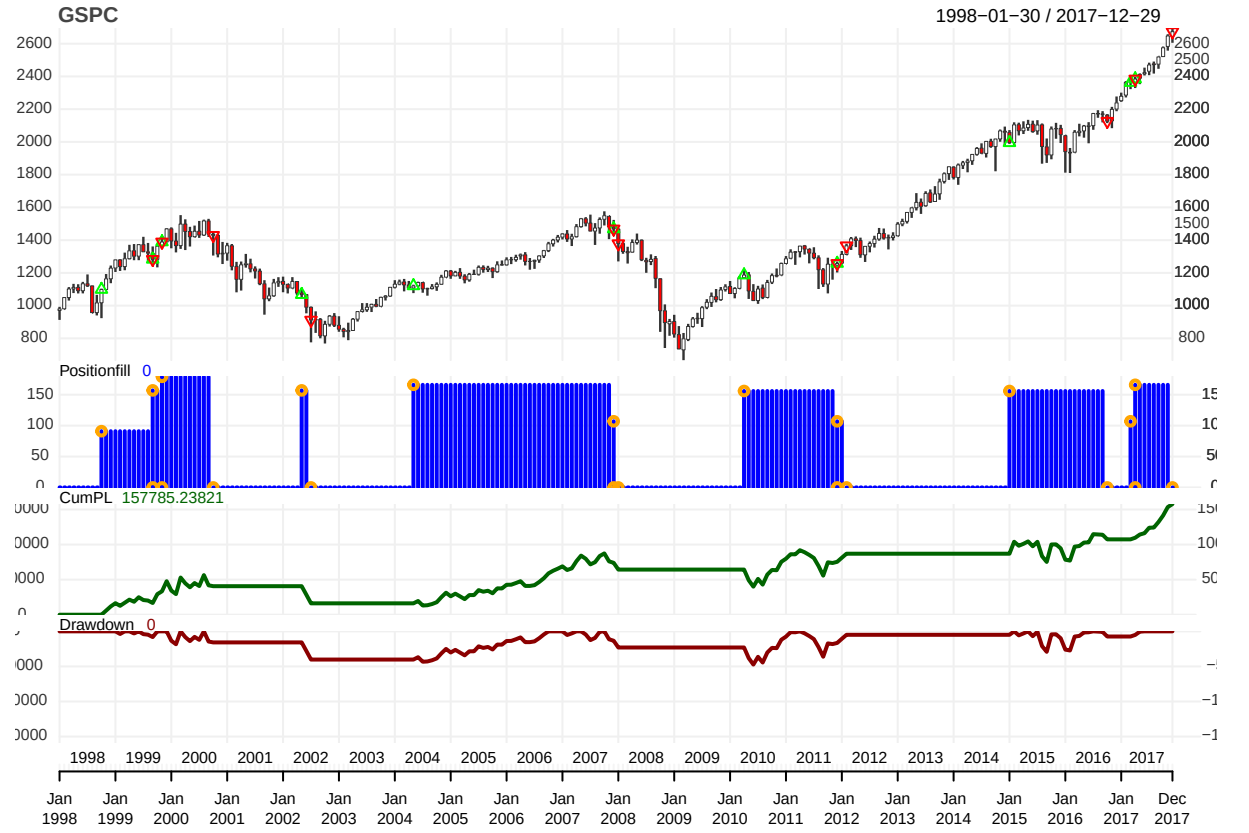




```
## Time difference of 4.476113 mins
```

Interestingly, the 'longtrend' strategy appears difficult to beat at random when constrained by the same characteristics as the original strategy. There are some "lucky" traders which manage to outperform 'longtrend' for a period until close to the end of the strategy when in early 2016 and until the end of the backtest period 'longtrend' is long and benefits from the recent bull market. In terms of totalPL 'longtrend' ranks 12th to eleven very lucky chimps. Of course many of the random traders would have been disadvantaged by taking positions during the deep drawdowns experienced following the tech and housing bubbles or being flat during the recent bull market, but their time in the market would resemble the same characteristics as 'longtrend' itself. We can take a look at the position chart of any one of the replicates to get a sense of how the respective random trader did. It should also illustrate how txnsim() honored the characteristics of the original strategy in terms of time in and out the market and quantities traded. Of course there is no layering in 'longtrend' nor are there any short trades, so we expect to see 1 level of long positions for a similar total duration as the original strategy.

```
chart.Posn("txnsim.wr.longtrend.1", Symbol = "GSPC")
```



From the analysis thus far we are able to deduce that our 'longtrend' strategy outperformed 989 random traders (out of 1,000) on a totalPL basis, following the same style. It would be difficult to conclude that the performance from 'longtrend' is the result of chance, since the strategy of long-term trend following has been used for decades for a reason. The benefits from being invested in risk-free assets during economic downturns cannot be understated. Indeed, the strategy's outpeformance increases over time as evidenced in the equity curve plot *(plot(lt.wr))* as many different market regimes are experienced. Could we have overfit the backtest? Referring to Faber's paper, he finds stability in his results for the GTAA portfolio when analyzing the range of monthly moving averages from 3m-12m. The analyst could easily perform a similar analysis using the *apply.paramset* function in quantstrat. Ignoring the many other potential objectives for assessing a strategy's feasibility for promotion to a production environment, 'longtrend' seems to pass initial scrutiny.

Ranks and p-values

One of the slots in the return object from txnsim() are the ranks of each replicate and the original strategy in terms of the summary statistics. The original strategy will be the first row in the dataframe. These ranks are used to determine the p-values of the statistics, in which p-values are calculated with reference to North et. al. (2002) who use Davison & Hinkley (1997) as their source.

```
head(lt.wr$ranks, 10)
```

```
##                         mean median stddev maxDD sharpe totalPL
## longtrend                  9    559    846    10      6      12
## txnsim.wr.longtrend.1    471    559    773    39    391     471
## txnsim.wr.longtrend.2    987    559    746   862    987     987
## txnsim.wr.longtrend.3    601    559    275   415    643     601
## txnsim.wr.longtrend.4    662    559    696   288    611     662
## txnsim.wr.longtrend.5    645    559    383   474    668     645
## txnsim.wr.longtrend.6    829    559    338   547    841     829
## txnsim.wr.longtrend.7    959    559    724   468    956     959
## txnsim.wr.longtrend.8    402    559    794   258    305     402
## txnsim.wr.longtrend.9    935    559    897   614    922     935
```

```
lt.wr$pvalues
```

```
##    mean  median  stddev   maxDD  sharpe totalPL
##  0.0090  0.5584  0.8452  0.0100  0.0060  0.0120
```

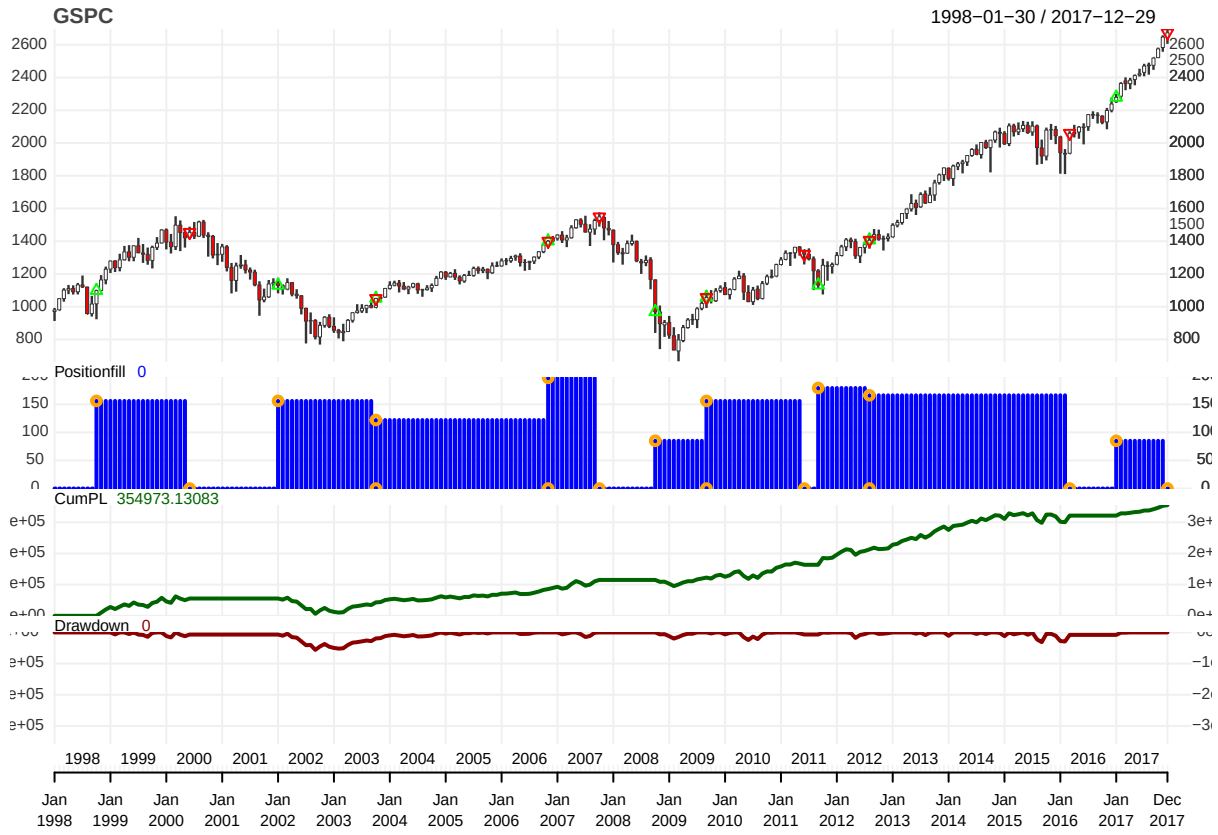
Since we have the statistic ranks of each replicate in a dataframe, it is possible to identify which replicates outperformed our original strategy based on either statistic. The below replicates outperformed 'longtrend' on a totalPL basis.

```
lt.wr$ranks[,6][which(lt.wr$ranks[,6] < 12)][order(lt.wr$ranks[,6][which(lt.wr$ranks[,6] < 12)])]
```

```
## txnsim.wr.longtrend.873 txnsim.wr.longtrend.956 txnsim.wr.longtrend.301
##                       1                       2                       3
## txnsim.wr.longtrend.102  txnsim.wr.longtrend.98  txnsim.wr.longtrend.17
##                       4                       5                       6
## txnsim.wr.longtrend.414 txnsim.wr.longtrend.836  txnsim.wr.longtrend.92
##                       7                       8                       9
##  txnsim.wr.longtrend.82 txnsim.wr.longtrend.211
##                      10                      11
```

One of the benefits mentioned previously for simulating round turn trades versus portfolio PL is the transparency. For a closer look at the winning random replicate we can analyse the trade durations and quantities using *chart.Posn()*.

```
win_rep <- names(lt.wr$ranks[,6][which(lt.wr$ranks[,6] == 1)])
chart.Posn(win_rep, Symbol = "GSPC")
```

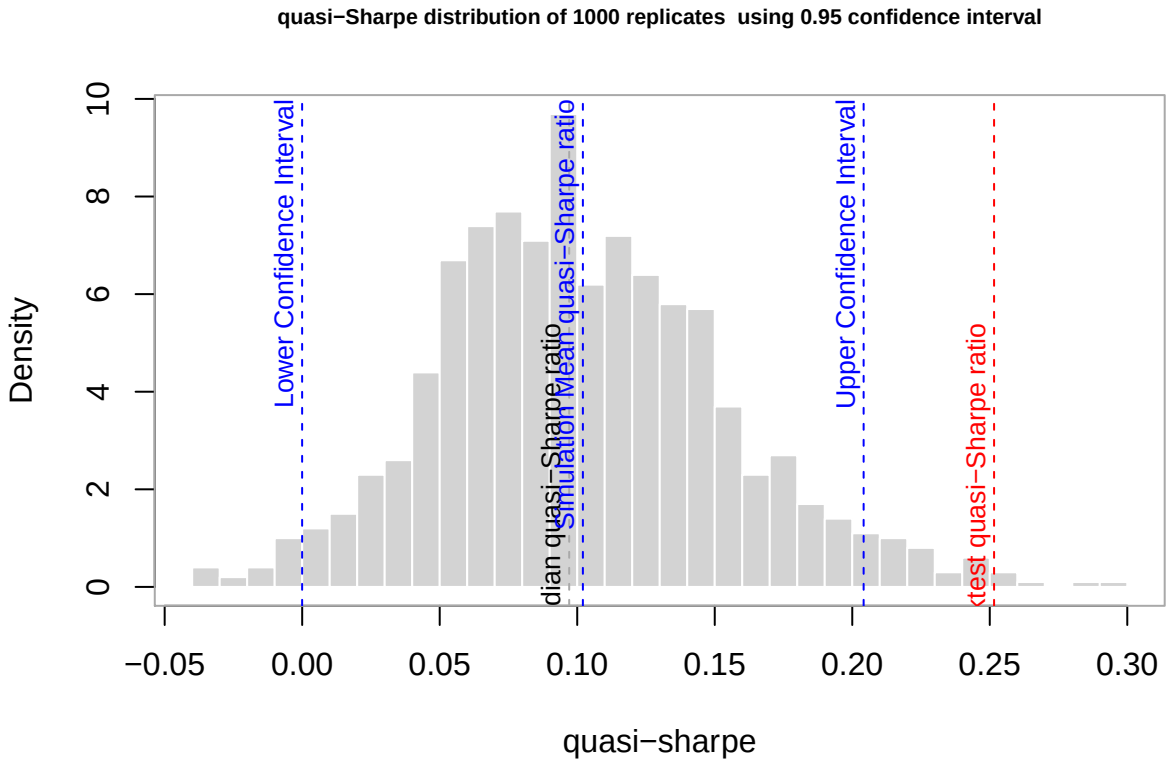


hist.txnsim()

With the hist.txnsim() method the analyst can generate a histogram from 5 different summary statistics: mean return, median return, max drawdown, standard deviation and sharpe ratio. The statistics are based on daily periodicities, and can be normalized or left as the default cash returns.

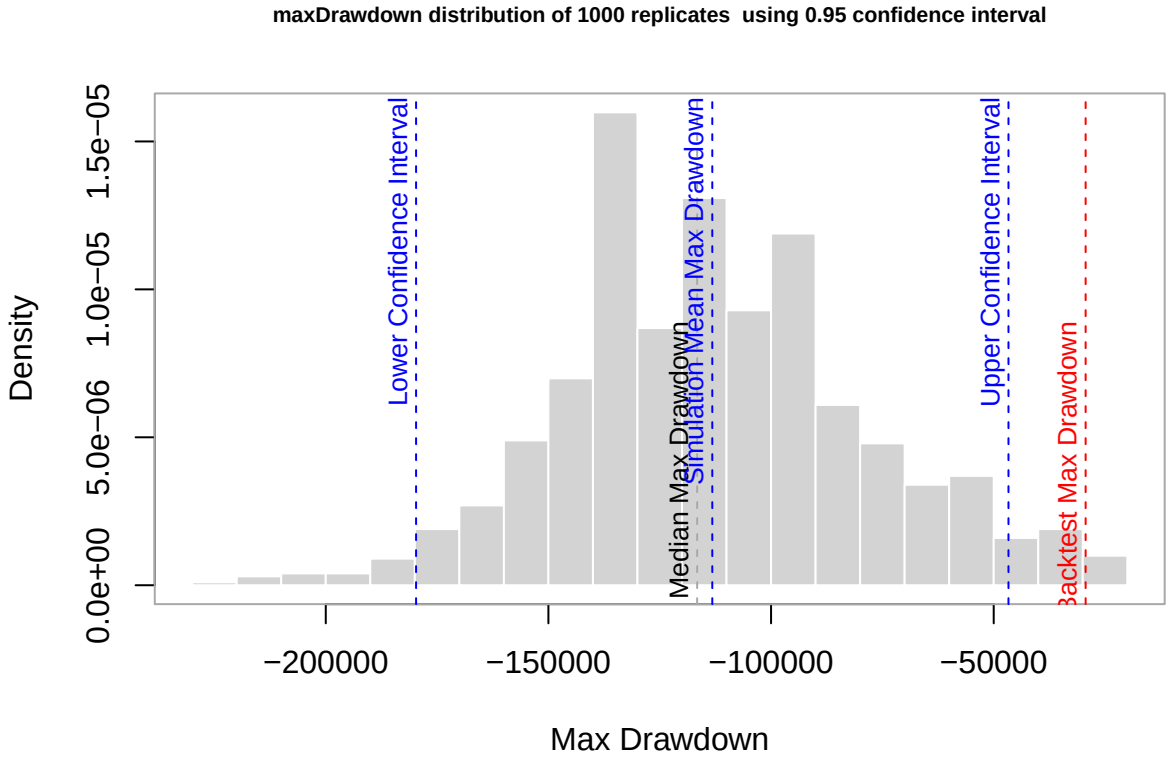
Looking at the risk adjusted return we see our 'longtrend' demo is close to the top of the distribution, well ahead of the upper confidence interval which was left as the default 95%.

```
hist(lt.wr, methods = "sharpe")
```



For max drawdown, since we missed the deep drawdowns in 2001-2002 and 2008, we expect to be close to the upper bound of that distribution too.

```
hist(lt.wr, methods = "maxDD")
```



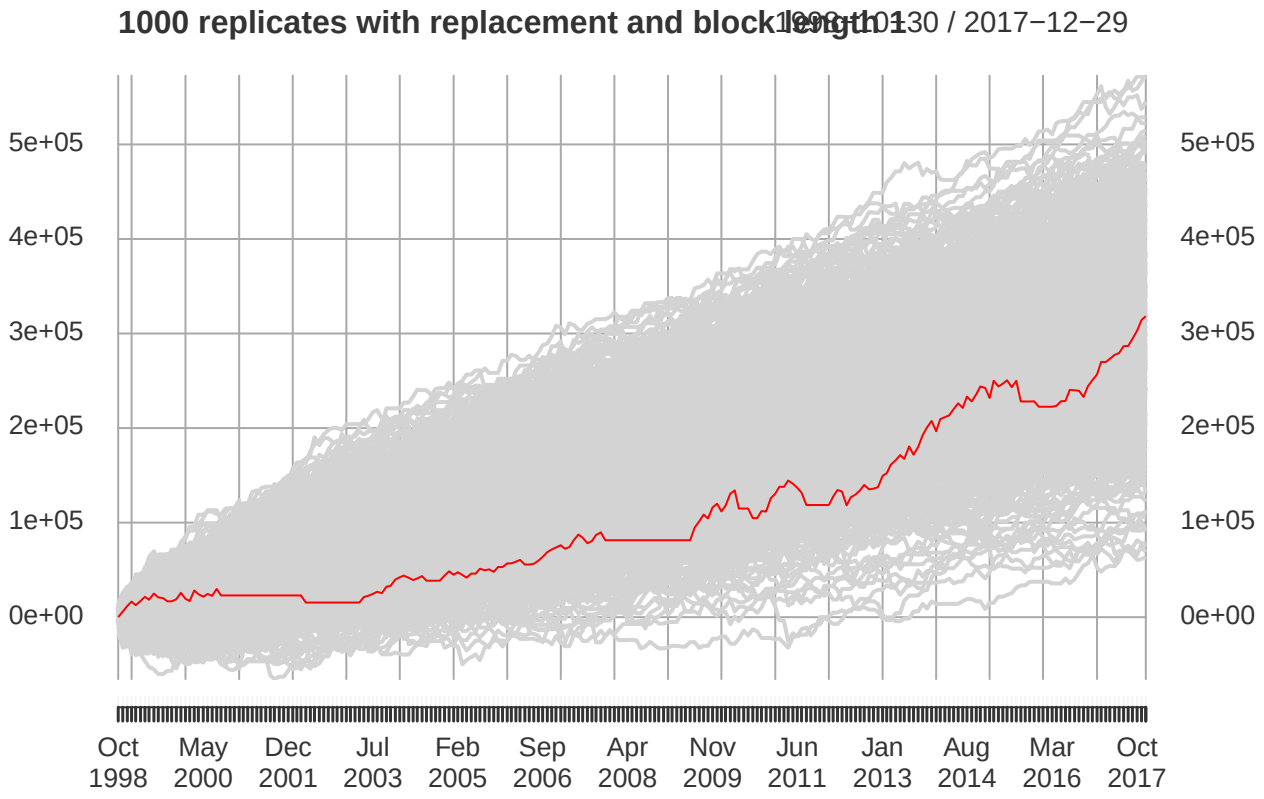
The 'longtrend' strategy may not be the most appropriate for determining luck versus skill or overfitting with

txnsim since random traders would be locked into their trades for lengthy durations thanks to the monthly periodicity of the signal process. Nevertheless using txnsim() we are able to dissect the performance of the strategy over its random counterparts and in so doing ascertain some level of confidence in the merits of this particular trend following strategy. Applying the analysis to a portfolio of asset classes may be an insightful undertaking.

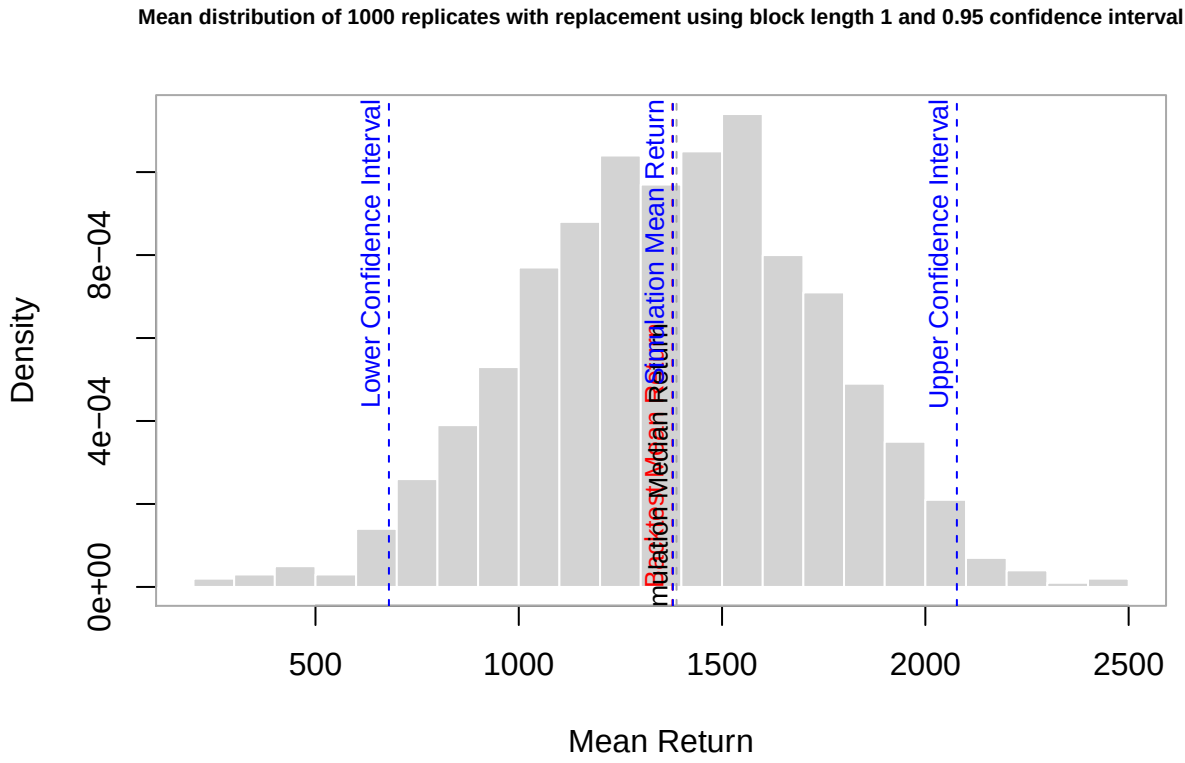
Another perspective using mcsim()

For an idea of the possible paths the daily equity curve could have taken, we also have the option of calling mcsim() without replacement and comparing the results of a portfolio PL simulation. A look at how the strategy compares in terms of maximum drawdown may add value to the analysis.

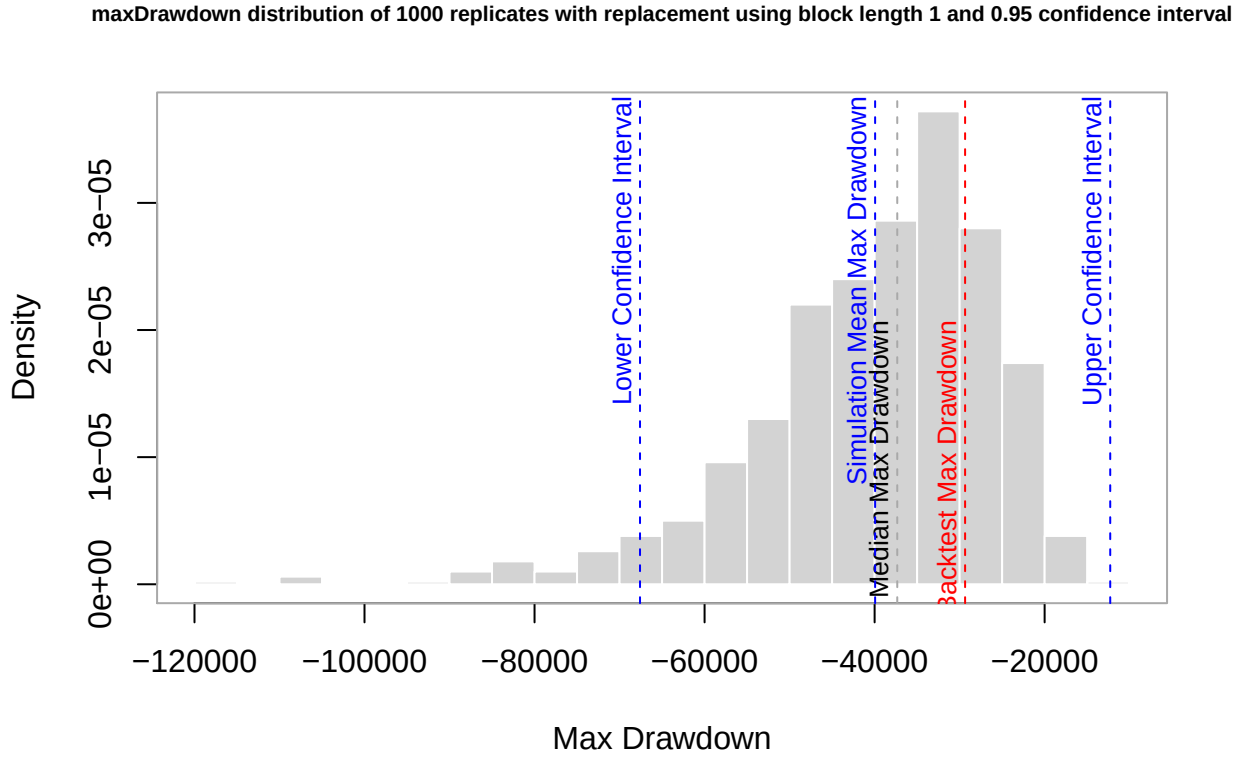
```
set.seed(333) #for the purposes of replicating my results
lt.mcsim.wr <- mcsim('longtrend', n = 1000)
plot(lt.mcsim.wr)
```




```
hist(lt.mcsim.wr, methods = "mean", normalize = FALSE)
```




```
hist(lt.mcsim.wr, methods = "maxDD", normalize = FALSE)
```




```
print(lt.mcsim.wr, normalize = FALSE)
```

```
##        Sample Mean Sample Median   Backtest   Lower CI   Upper CI
## mean      1379.065      1388.049   1378.020    680.639   2077.490
```

```
## median      57.552         0.000      0.000   -361.536    476.640
## stddev    5479.951      5471.796   5477.659   4762.202   6197.700
## maxDD   -39938.817    -37335.691 -29356.796 -67592.733 -12284.901
## sharpe       0.254         0.255      0.252      0.116      0.391
##        Std. Error
## mean      356.346
## median    213.825
## stddev    366.205
## maxDD   14109.400
## sharpe      0.070
```

Looking at the results of mcsim() on the 'longtrend' demo the return and drawdown characteristics of the original strategy are certainly reasonable, with both metrics closer to the average of the simulations compared with txnsim(). Of course maxixmum drawdown is slightly better than the average which should be expected for a trend following strategy.

print.txnsim()

Lastly for 'longtrend' we look at a summary of the backtest and replicate summary statistics using the print.txnsim() S3 method. It is a wrapper for the summary.txnsim() method, so calling print(lt.wr) will be sufficient for viewing a summary of the results.

```
print(lt.wr)
```

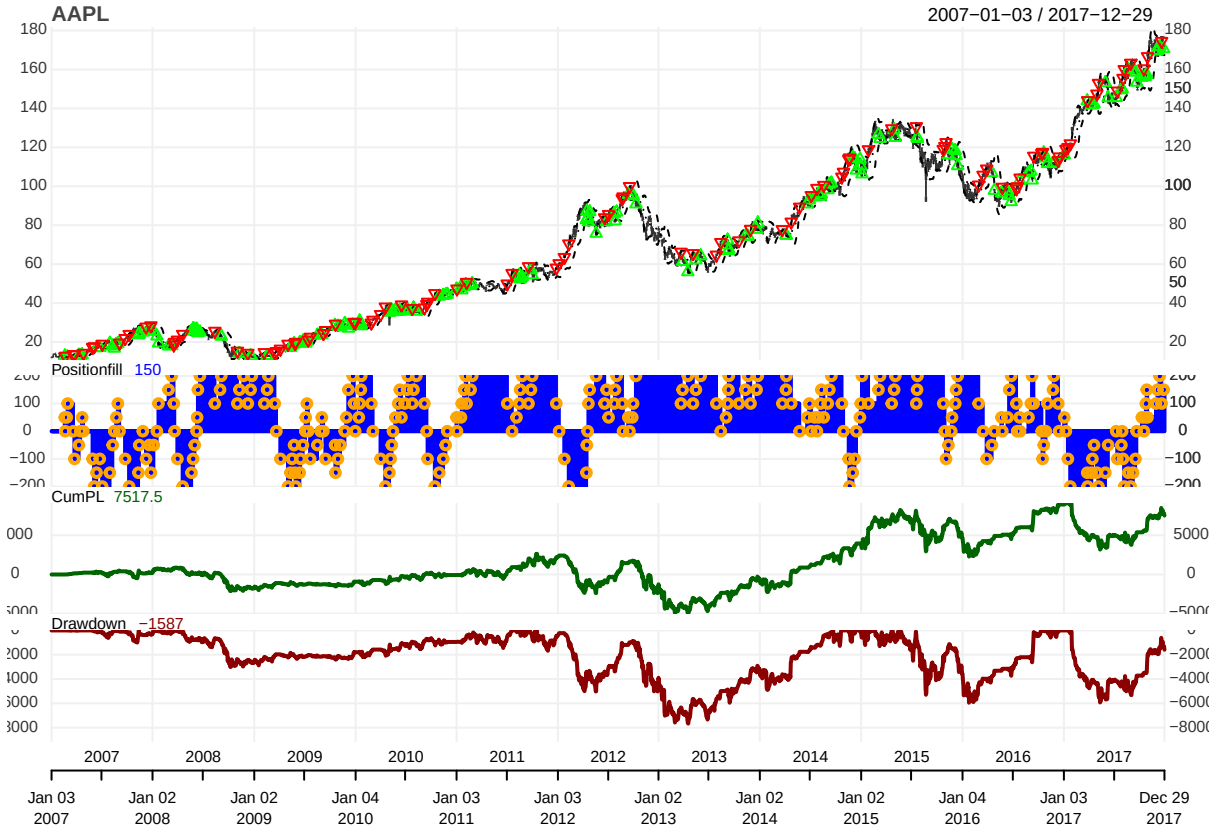
```
##           backtest Std. Error    Lower CI   Upper CI
## mean      1378.020    307.009      42.144   1245.599
## median       0.000    201.665    -341.058    449.452
## stddev    5477.659    869.183    4713.582   8120.715
## maxDD   -29356.796  33938.176 -179716.482 -46681.278
## sharpe       0.252      0.052       0.000      0.204
## totalPL 318322.527  73653.651   10158.215 298875.220
```

Layers and Long/Short strategies with 'bbands'

To highlight the ability of txnsim() to capture the stylized facts of more comprehensive strategies including Long/Short strategies with leveling we use a variation of the 'bbands' strategy. Since we apply an un-optimised position-sizing adjustment to illustrate leveling, we do not expect the strategy to outperform the majority of its random counterparts.

A call to *chart.Posn* highlights the additional traits, namely entering long **and** short positions **with leveling** in and out, when compared with 'longtrend'.

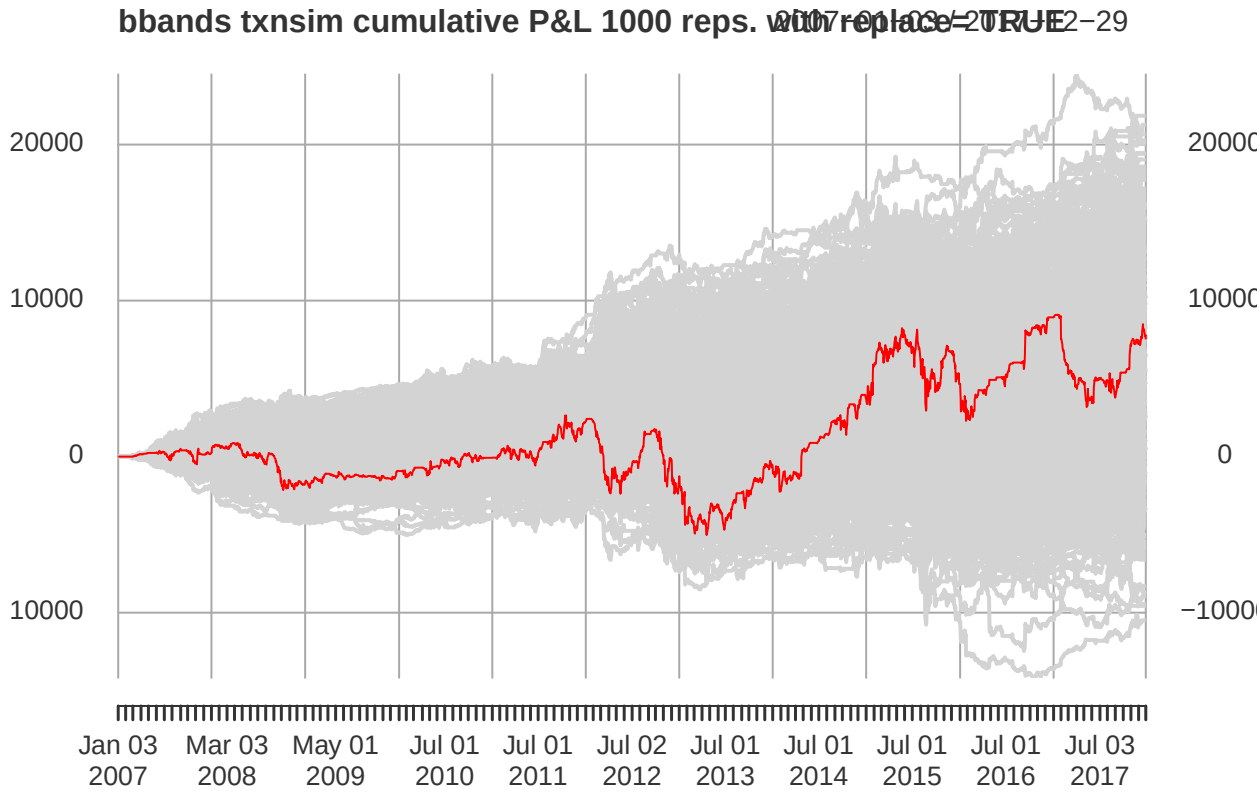
```
chart.Posn(Portfolio='bbands',Symbol="AAPL",
           TA="add_BBands(on=1,sd=SD,n=N)")
```



A thousand random traders

We run 1000 replicates from a slight variation on the 'bbands' quantstrat demo strategy to generate simulations for 1000 random traders using *txnsim*.

The *txnsim* function is currently (as of January 2018) very CPU and memory intensive, we have observed 1k replicates taking anywhere from less than 20 minutes on a large research PC to 15hrs on less capable hardware. In summary, don't try 1000 replicates using a laptop or smaller PC.

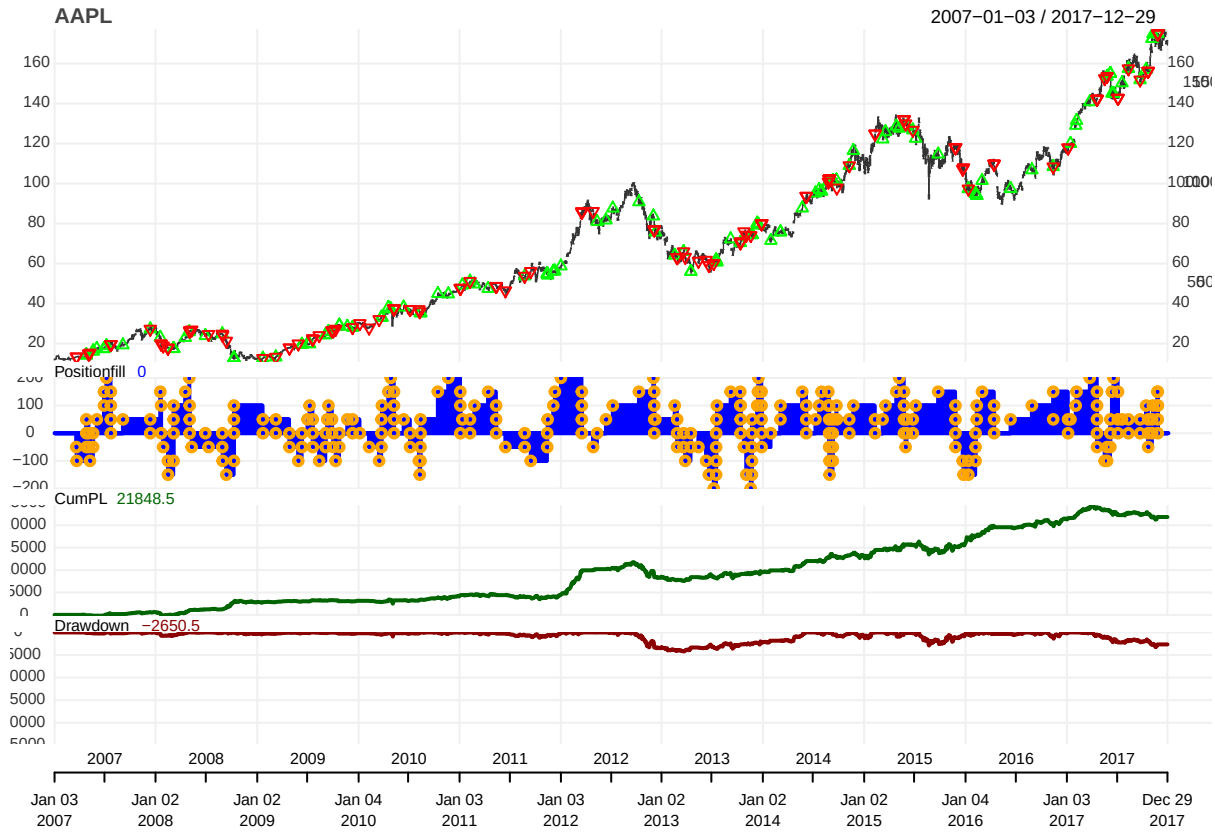



```
## Time difference of 18.83995 mins
```

The resulting equity curves confirm our suspicions that we have a lower probability of outperforming random replicates for this version of 'bbands'.

Taking a closer look at the performance and position taking of the "winning" random replicate, we get a sense of how the strategy attempts to mirror the original in terms of position sizing and duration of long versus short positions overall. It should also be evident how the replicate has honored the maximum long and short positions observed in the original strategy.

```
win_rep <- names(bb.wr$ranks[,6][which(bb.wr$ranks[,6] == 1)])
chart.Posn(win_rep, Symbol = "AAPL")
```



txnsim - the process

As alluded to in the *Round Turn Trades & tradeDef* section, there are 3 basic variations for sampling round turn trades in txnsim.

1. "flat.to.flat" && replace = FALSE

The simplest case would be sampling round turns defined as "flat.to.flat" without replacement. In this case we would simply be rearranging the vector of durations and quantities. The paths which the resulting equity curves can take will vary together with the final result, since we will be marking the simulated trades to market data timestamps based on randomly sampled durations.

Using the *replicates* slot returned in the output of txnsim, we can analyze the stylized facts used to build the *transactions* which are also returned as a slot in the txnsim object. Any timestamp variation between replicates and transactions will most probably stem from replicate timestamps falling on weekends or holidays, or due to missing market data such as in the case of a strategy with a monthly periodicity and corresponding monthly market data (such as the longtrend demo). In these cases the trasaction will be added at the most recent timestamp with market data.

Looking at the sum of long period durations from the original strategy as well as for the first 2 replicates should highlight how txnsim honors this stylized fact when building out the replicates using tradeDef=flat.to.flat and replace=FALSE for our 'longtrend' example.

```
##
##  4995 long period duration for original strategy
##
##  4995 long period duration for replicate 1
##
```

```
##  4995 long period duration for replicate 2
```

2. "flat.to.flat" && replace = TRUE

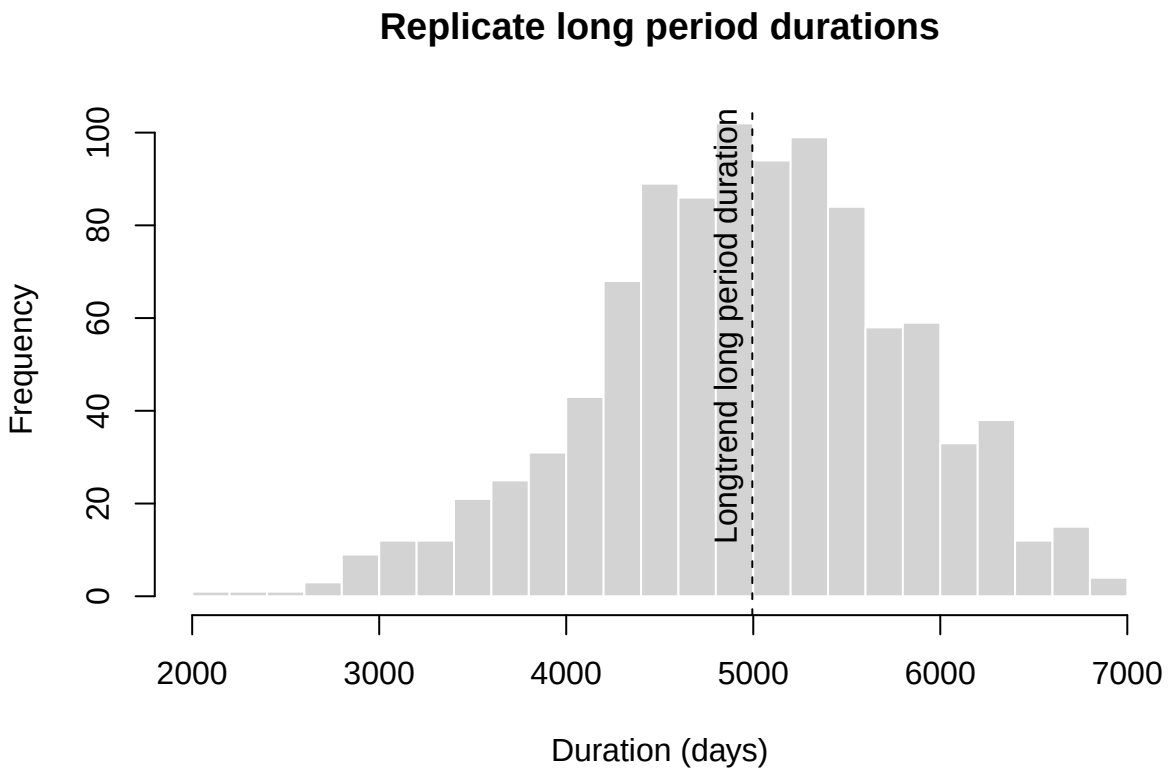
The next simplest case for simulating round turn trades is sampling "flat.to.flat" round turns with replacement. Since we will inevitably be sampling a particular duration multiple times, it is possible to end with a total duration greater than or less than the original strategy total duration. To manage this risk we, 1. add a "fudge factor" to the size of our sample and, 2. keep sampling until the sampled total duration is equal to or exceeds our target total duration (the total duration from the original strategy). Increasing the size of our sample (number of round turns from original strategy as well as flat periods) reduces the likelihood of needing to sample more than once.

After sampling the resultant durations we identify which n-th element in the vector takes us over our target duration, truncate any element beyond that and trim the duration in the n-th element to get a newly sampled dataframe of durations and their respective quantities which matches the total duration of the original strategy.

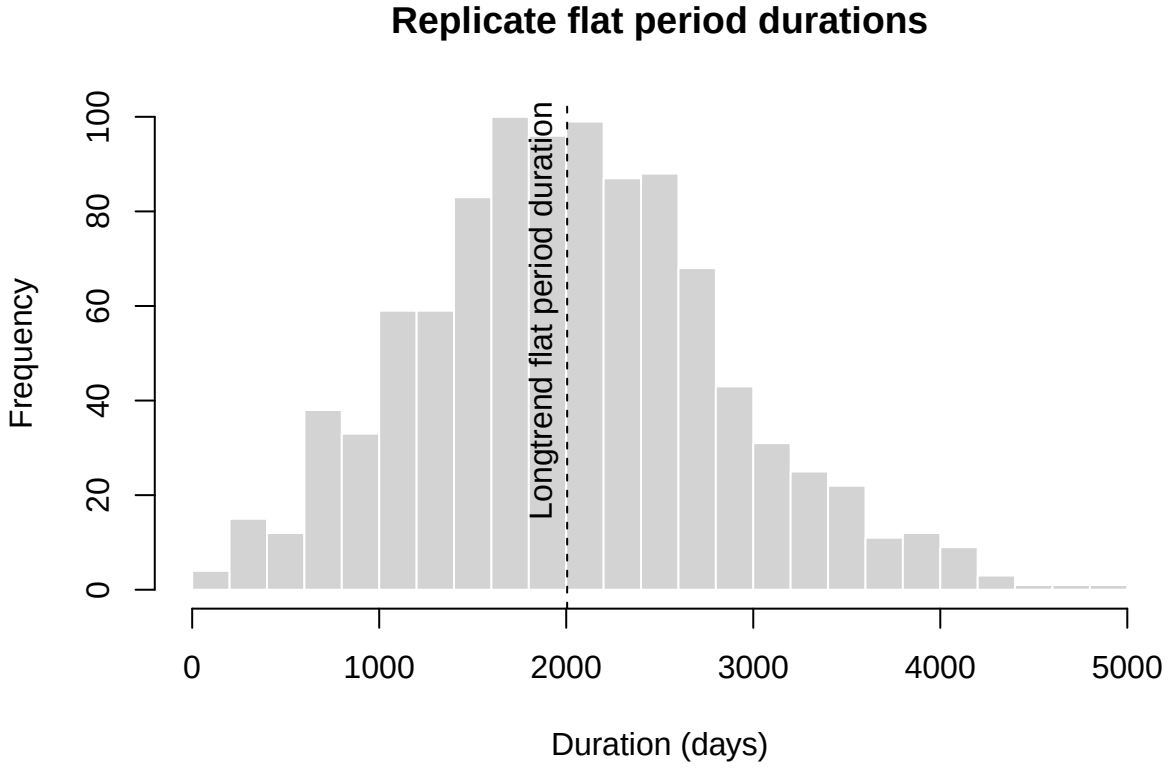
Since we have sampled with replacement, our replicates will have a range of long and flat period durations centered around the long and flat period durations from 'longtrend' itself. Before building a histogram to show these distributions, a quick look at the sum of long period durations and flat period durations from a few replicates is worthwhile.

```
##
##  4995 long period duration for original strategy
##  2005 flat period duration for original strategy
##  7000 total duration
##
##  3692 long period duration for replicate 1
##  3308 flat period duration for replicate 1
##  7000 total duration
##
##  4484 long period duration for replicate 5
##  2516 flat period duration for replicate 5
##  7000 total duration
##
##  5084 long period duration for replicate 10
##  1916 flat period duration for replicate 10
##  7000 total duration
```

The histogram below confirms the normal distribution around which our longtrend demo long period duration is centered.



By implication the flat period durations will display an equivalent distribution around the sum of our longtrend demo flat durations.



3. "increased.to.reduced" || "flat.to.reduced"

For any round turn trade methodology which is not measuring round turns as flat.to.flat, things get more complicated. Fortunately, the complication is the same for txnsim regardless of the methodology used to pair entry and exit trades.

The first major complication with any trade that levels into a position is that the sum of trade durations will be longer than the market data. The general pattern of the solution to this complication is that we sample as usual, to a duration eqivalent to the duration of the first layer of the strategy. In essence we are sampling assuming round turns are defined as "flat.to.flat". Any sampled durations beyond this first layer are overlapped onto the first layer. The number of layers is determined by the amount of times the *first layer* total duration is divisible into the total duration. In this way the total number of layers and their duration is directly related to the original strategy.

The next complication is max position. Now, a strategy may or may not utilize position limits. This is irrelevant. We have no idea which parameters are used within a strategy, only what is observable ex post. For this reason we store the maximum long and short positions observed as a stylized fact. To ensure we do not breach these observed max long and short positions during layering we keep track of the respective cumsum of each long and short levelled trade.

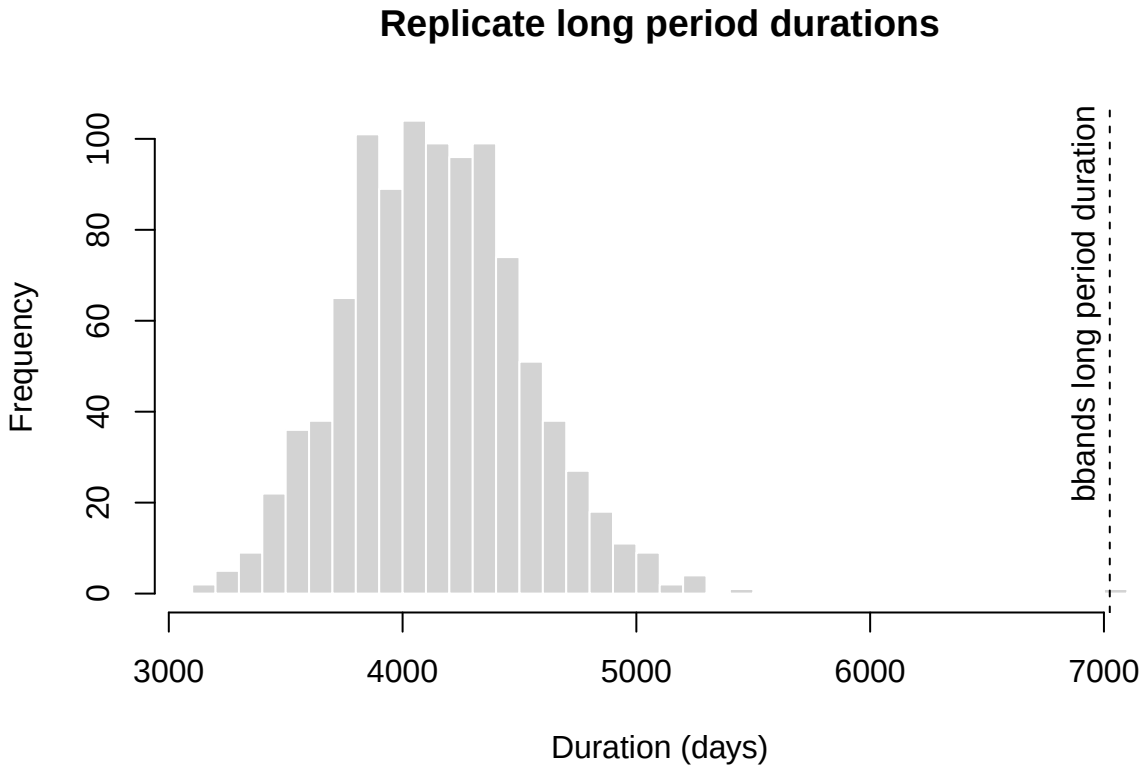
The procedure for building the first layer in a replicate strategy using tradeDef="increased.to.reduced" is slightly different to the procedure for building replicates when tradeDef="flat.to.flat". For "flat.to.flat" round turns, it is perfectly suitable to sample care free between flat, long and short periods with their respective quantities. For any trade definition other than flat.to.flat, however, we need to be cogniscant of flat periods when layering to ensure we do not layer into an otherwise sampled flat period. For this reason we match the sum duration of flat periods in the original strategy for every replicate. To complete the first layer with long and short periods, we sample these separately and truncate the respectively sampled long and short duration which takes us over our target duration. When determining a target long and shorttotal duration to sample to, we use the ratio of long periods to short periods from the original strategy to distinguish between the direction of non-flat periods.

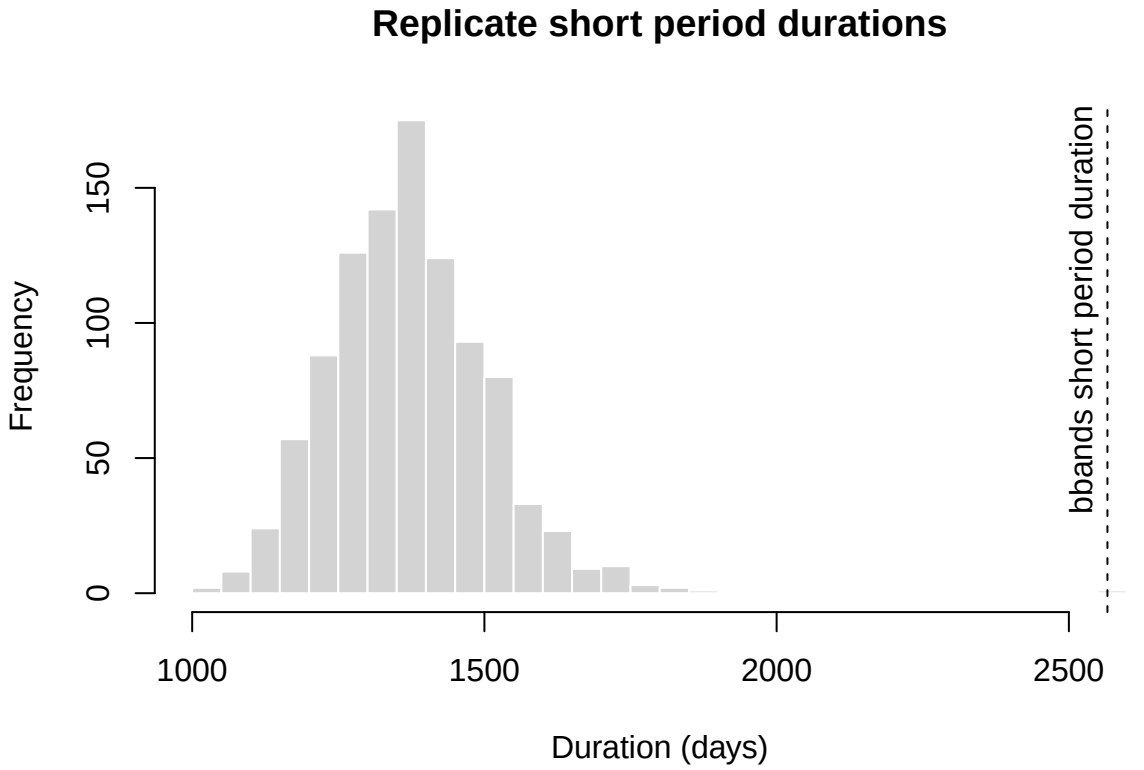
At this point it would be worth taking a look at a few *first layer* replicate flat, long and short durations before looking at their total distributions. We expect the first layer distributions to be tightly centered around the original strategy. This is because we secured the flat periods per the original strategy and used the long:short ratio stylized fact *(lsratio)* to target sample first layer long and short durations (the sum of which may not exceed the calendar duration of the original strategy). The variation for long and short durations will stem from the fact that we truncate completely the duration which takes us over our target for long and short duration sampling.

```
##
##  2393 first layer long period duration for original strategy
##  584 first layer flat period duration for original strategy
##  987 first layer short period duration for original strategy
##  3964 total duration of first layer
##
##  2355 first layer long period duration for replicate 1
##  584 first layer flat period duration for replicate 1
##  1024 first layer short period duration for replicate 1
##  3963 total duration of first layer
##
##  2358 first layer long period duration for replicate 2
##  584 first layer flat period duration for replicate 2
##  1012 first layer short period duration for replicate 2
##  3954 total duration of first layer
```

The replicate flat periods are each 584 days and identical to the original strategy. Due to truncation, the first layer long and short periods will vary but to a much lesser extent than with tradeDef="flat.to.flat" with replacement. Of course, when comparing the total duration of each strategy including their respective layers, the distribution will fan out somewhat. Let's look at the long and short period distributions of each replicate

for all layers combined.





The above histograms highlight how the observed durations from our bbands demo are quite far removed from the distribution of the random replicates themselves, which derive directly from the bbands demo. Our layering procedure inside txnsim may need to capture more granularly the structure of the layering in the

observed strategy. At present we essentially sample randomly between 2 points to determine the start time of a layered trade. Future work on txnsim will most likely focus on this perceived shortcoming, to better capture the style of the original strategy where tradeDef="increased.to.reduced".

Future Work

This research area is full of pitfalls and opportunities. Using the simulation to try to replicate stylized facts is a series of tradeoffs. Each stylized fact adds more realism to the simulated random traders, but also runs the risk of overfitting to the behavior of the original observed series of trades. For example, as of January 2018, the trade layering simulations systematically under-represent the amount of time the simulation takes a larger position, in comparison to the observed trades. We feel that this particular problem is solvable, but other simulation difficulties are sure to arise as other stylized facts are considered for inclusion in later versions.

There are other simulation methodologies still to be implemented, including Combinatorially Symmetric Cross-Validation (CSCV per Bailey and Lopez de Prado), various drawdown analysis simulation metrics, simulations using simulated or resampled market data, application of the *txnsim* stylized facts to other market data than the original observed data, and more.

Further, though some methods for doing so are already implemented in *quantstrat*, there is more work to be done in evaluating whether a series of observed transactions are likely to be overfit.

In round turn trade simulation, there may be utility in allowing the analyst to choose which stylized facts to attempt to replicate or use in the simulation. There may also be value in providing the stylized fact functions as exposed user functions that could be called on an observed set of transactions without doing any simulation, just for descriptive purposes.

Conclusion

Round turn trade Monte Carlo simulates random traders who behave in a similar manner to an observed series of real or backtest transactions. We feel that round turn trade simulation offers insights significantly beyond what is available from:

- equity curve Monte Carlo (implemented in *blotter* in *mcsim*),
- from simple resampling (e.g. from *pbo* or *boot*),
- or from the use of simulated input data (which typically fails to recover many important stylized facts of real market data).

Round turn trade Monte Carlo as implemented in *txnsim* directly analyzes what types of trades and P&L were plausible with a similar trade cadence to the observed series. It acts on the same real market data as the observed trades, efficiently searching the feasible space of possible trades given the stylized facts. It is, in our opinion, a significant contribution for any analyst seeking to evaluate the question of "skill vs. luck" of the observed trades, or for more broadly understanding what is theoretically possible with a certain trading cadence and style.