

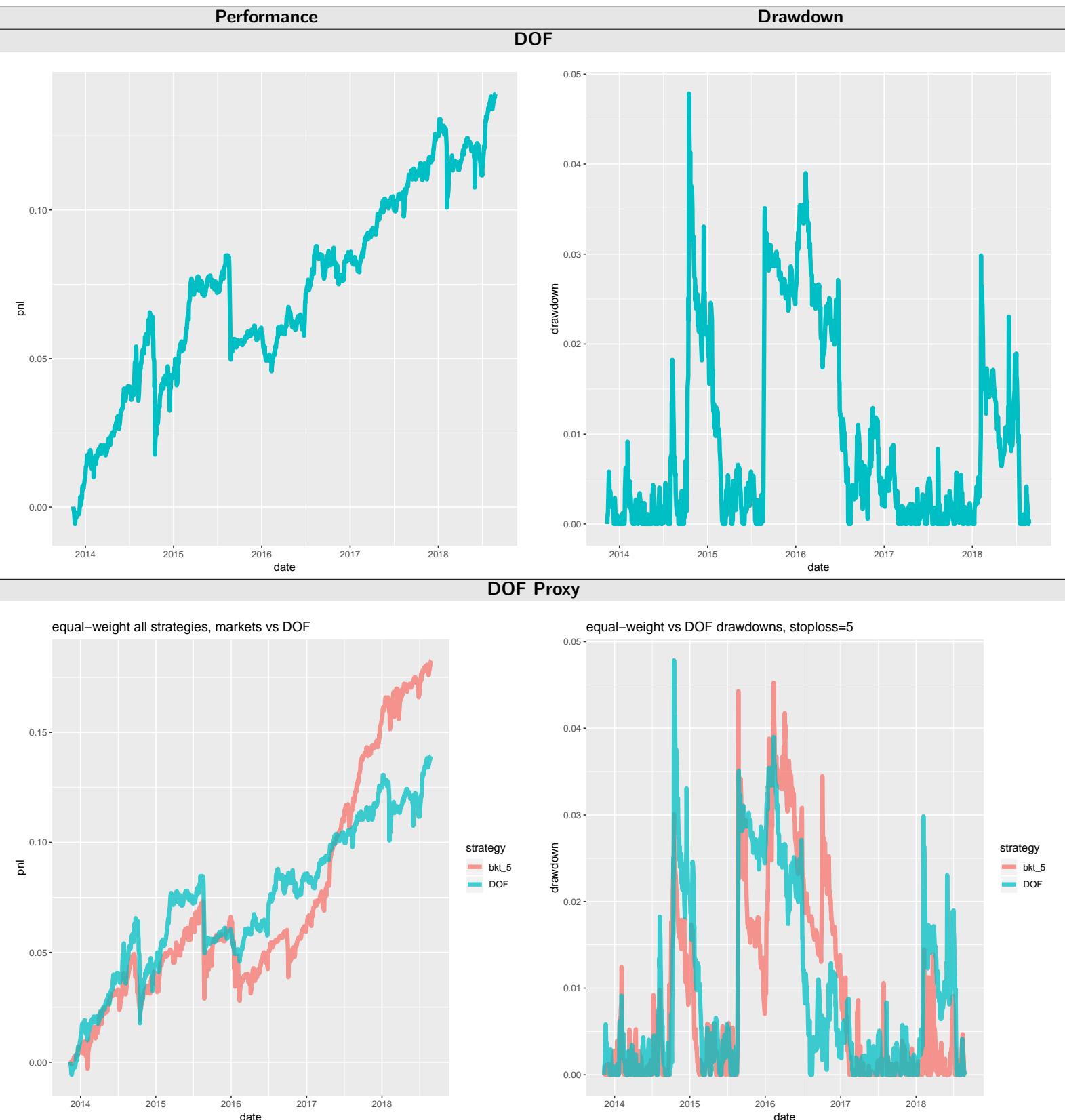
Contents

1	DOF performance	3
2	DOF Proxy: Return quality vs stop-loss level	4
3	The 90-100 PS closely matched DOF performance	6
4	return quality vs stop-loss level	7
5	Weighted strategy backtests	8
5.1	equal weights	8
5.2	theoretical weights	9
5.3	historical weights	10
5.4	delta 1 teenies	11
6	Drawdown mitigation: SPX “teenies”	12
7	All backtests	15
8	Strategies vs market backtests	16
8.1	3m maturity	16
8.2	6m (Jun, Dec) maturities	17
8.3	6m (Mar, Sep) maturities	18
9	Dataset	19
10	Backtest algorithm	20
10.1	backtest inputs	20
10.2	volatility surface input	20
10.3	schedule input	20
10.4	payoff input	20
10.5	backtest workflow	21
10.6	volatility interpolation	22
10.7	Unit tests	23

Document version

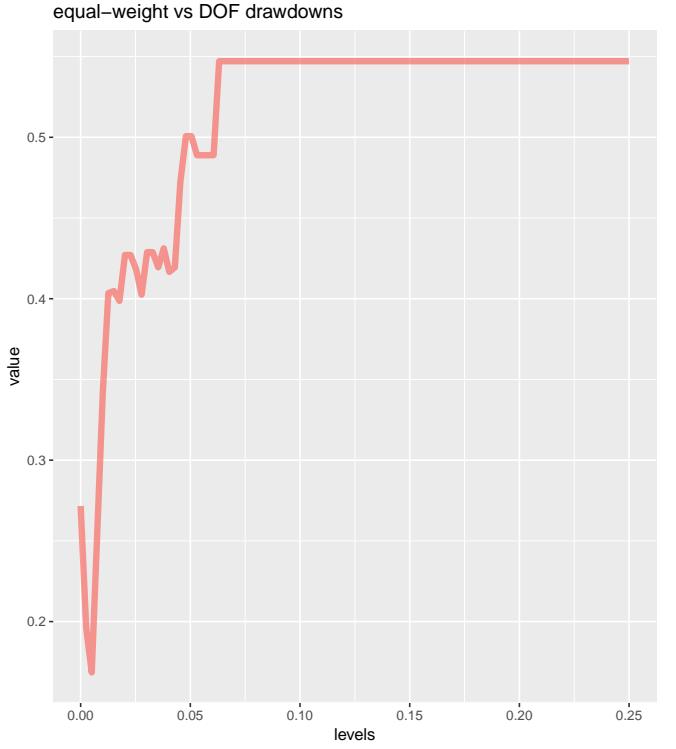
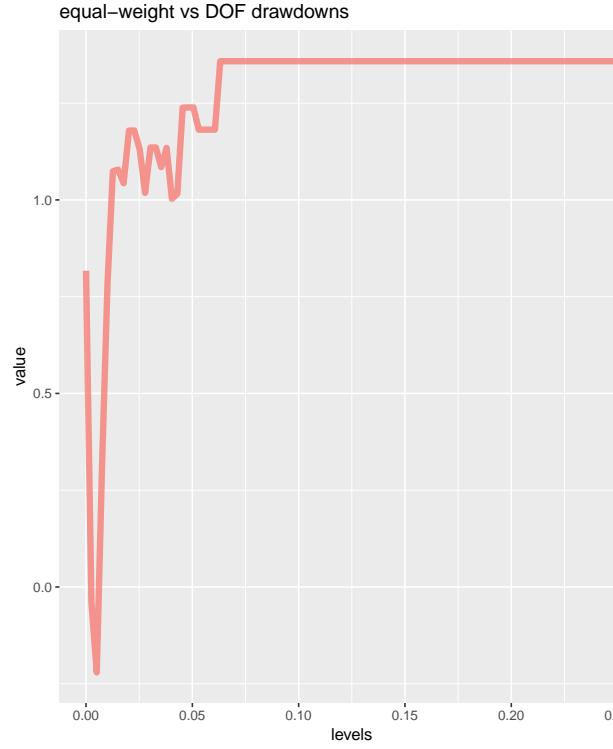
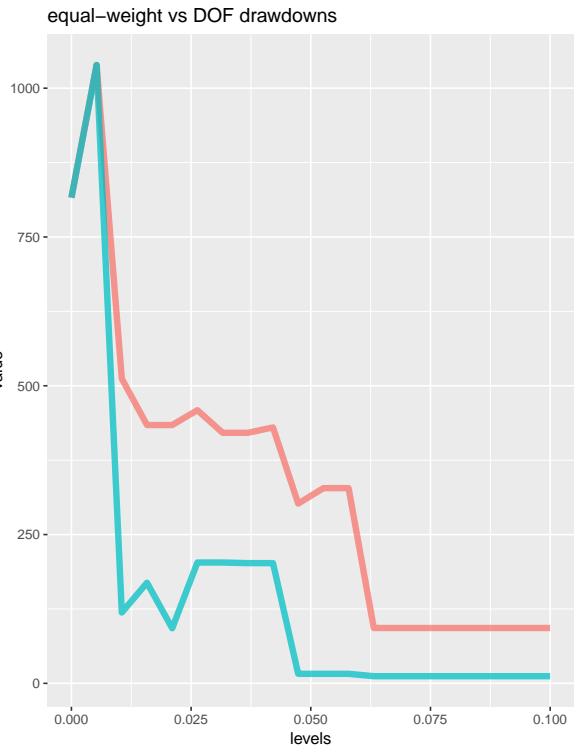
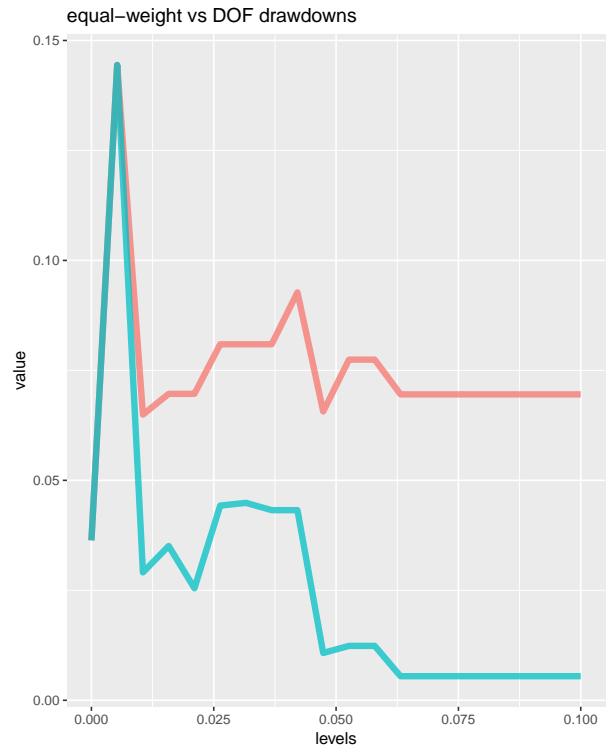
timestamp	2018-11-18 15:27:07
data and code checksum	fc5b947f526b9d5f68830174bd68967a
git id	79ebf26b1414a7ae75a438bfd77568bfb589a657
user	nicknassuphis

1 DOF performance



- We wish to investigate the cause of infrequent, relatively large drawdowns.
- In addition, we are looking for a simple, intuitive proxy that can be used to understand DOF performance.
- Moreover, we investigate 3 different mitigation strategies: sizing, stop-loss and a positive tail-risk overlay.

2 DOF Proxy: Return quality vs stop-loss level



equal-weight all strategies, markets vs DOF

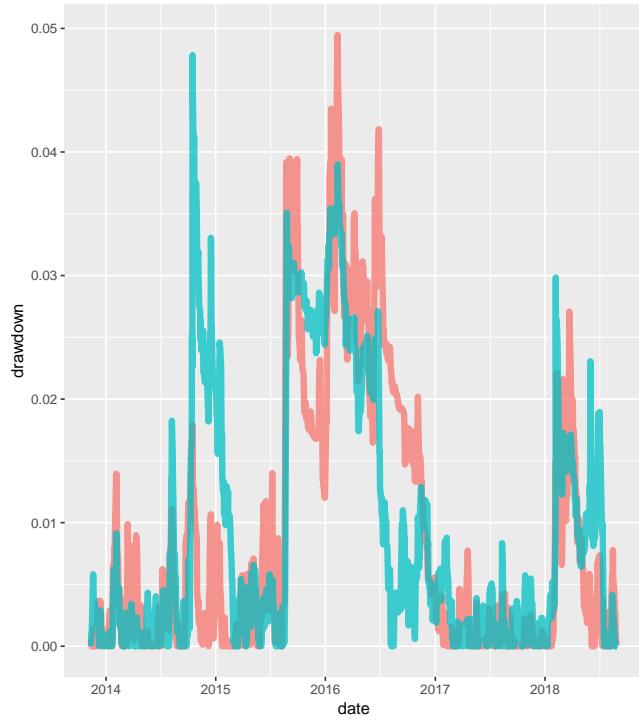


3 The 90-100 PS closely matched DOF performance

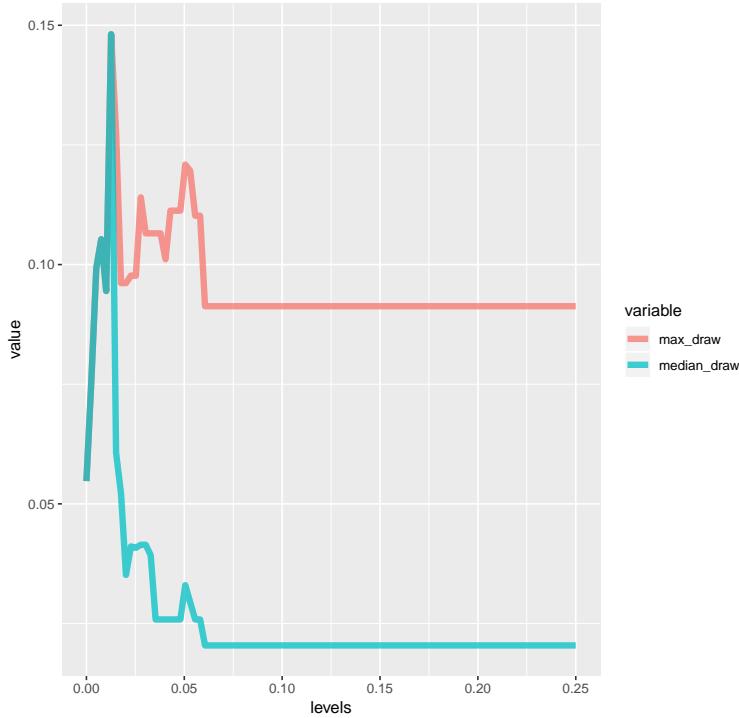
putspread-weight all strategies, markets vs DOF



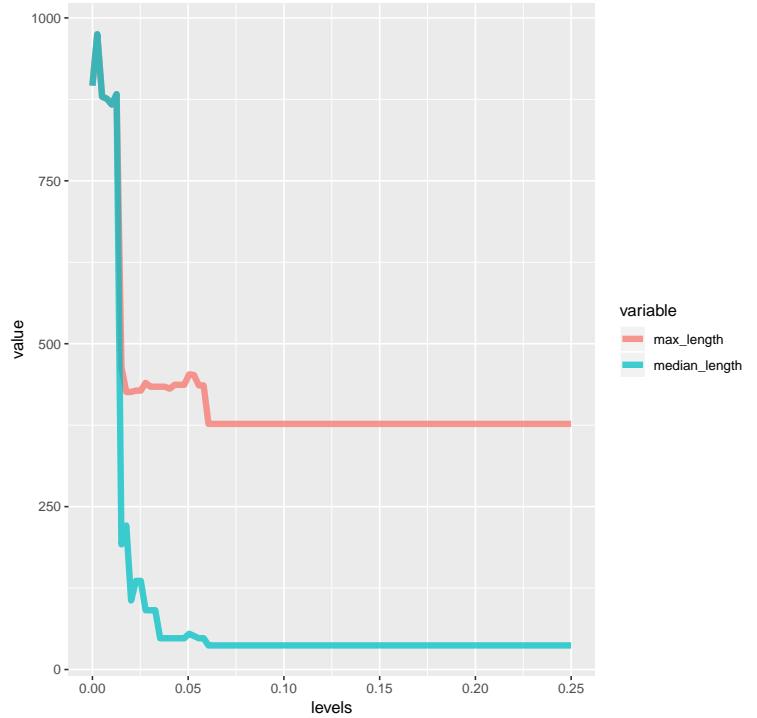
putspread-weight vs DOF drawdowns, stoploss=20



putspread-weight vs DOF drawdowns

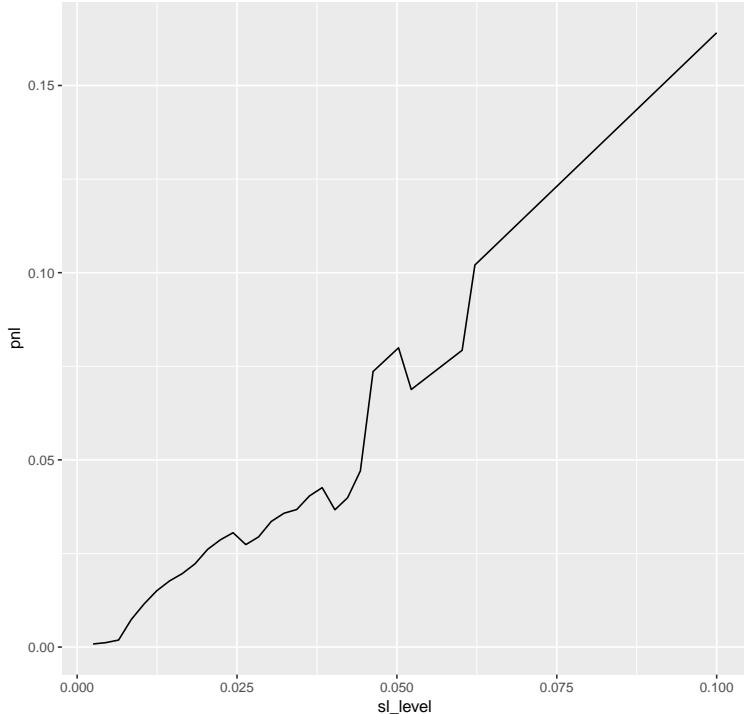


putspread-weight vs DOF drawdowns

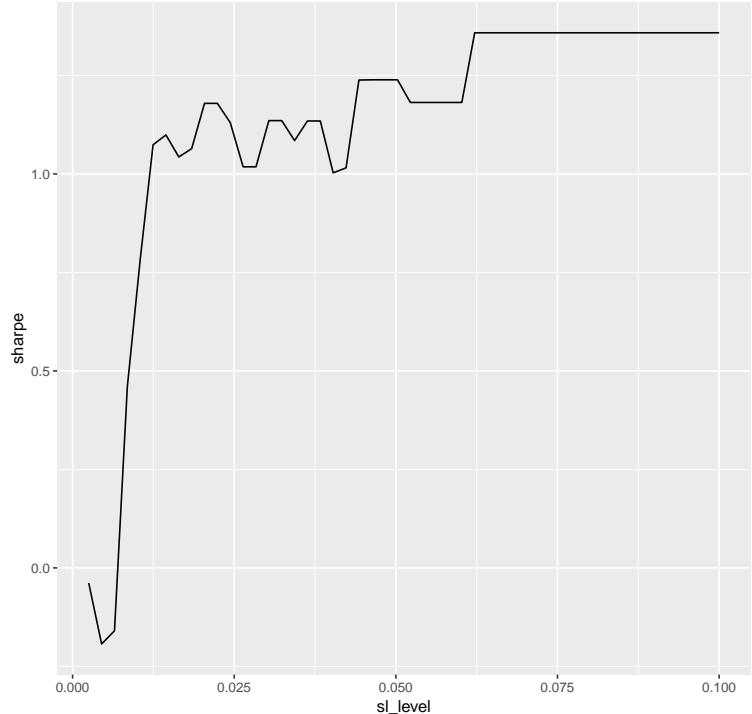


4 return quality vs stop-loss level

DOF Proxy stop-loss level vs P&L



DOF Proxy stop-loss level vs Sharpe

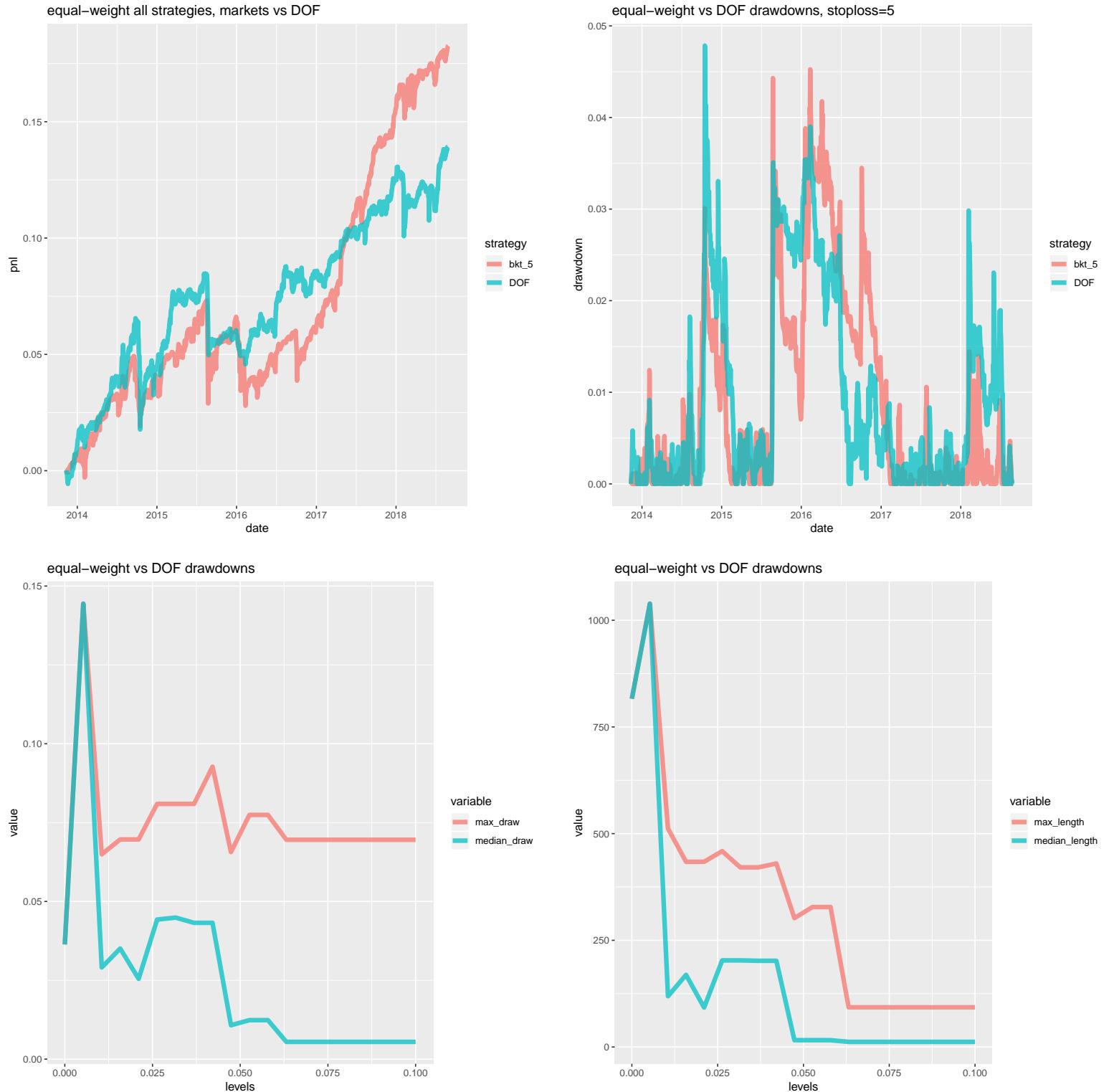


"Small positions with wide stops" are better than "Large positions with tight stops"

In general, we would expect stop-losses to reduce return quality. The intuition behind this expectation is that post-selloff rebounds are among the highest-sharpe periods in a risk-premium strategy's life. Stop-losses limit drawdowns but also deny us participation in rebounds. On balance, return quality reduction dominates. This means that we are better off reducing position sizes rather than tightening stops.

5 Weighted strategy backtests

5.1 equal weights



Tighter stops reduce the depth of drawdowns but increase their length. This is fairly typical for risk-premium collection strategies.

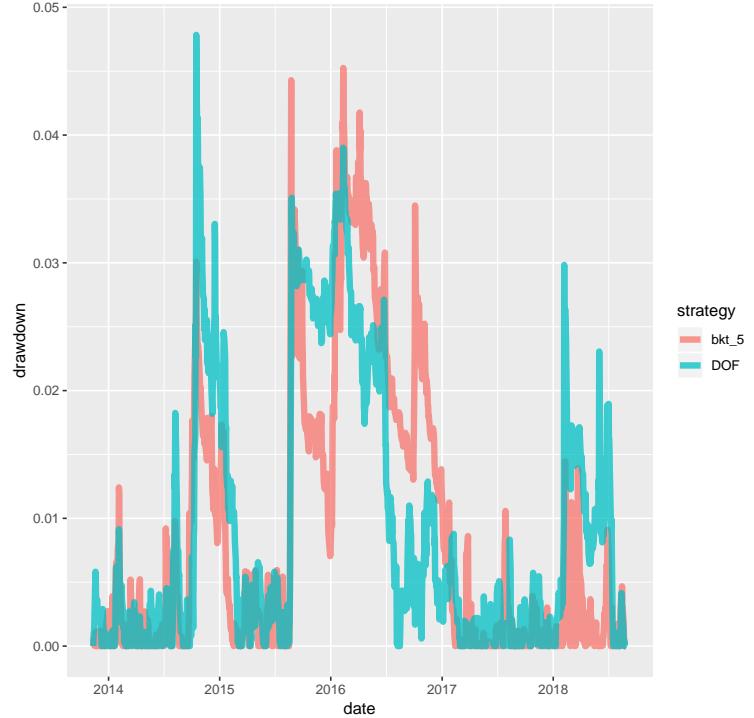
5.2 theoretical weights

Portfolio composition		
Zero Cost Call Spread Collars	33 %	
Zero Cost Put Spread Collars	0 %	
Zero Cost Risk Reversals	0 %	
Zero Cost Put Spread Ratios	33 %	
Short Straddles	33 %	

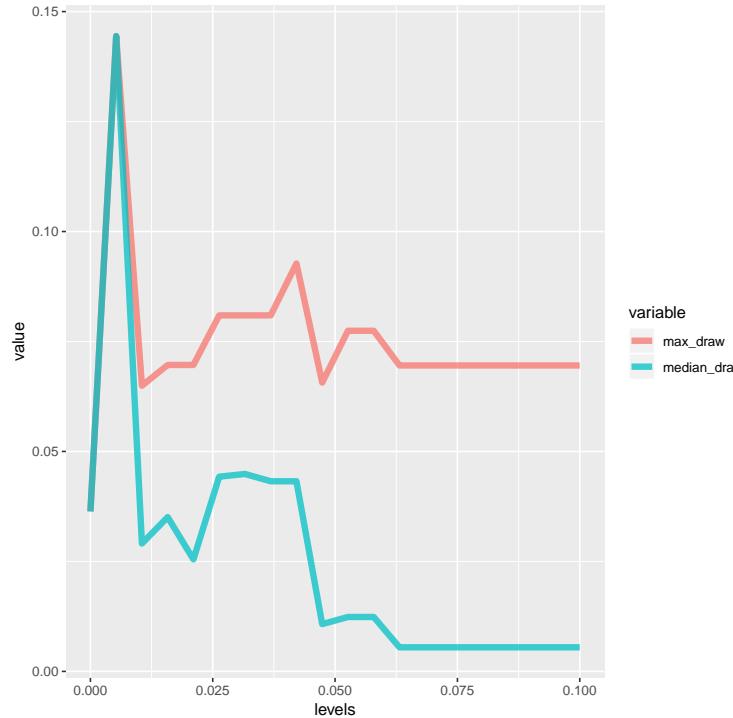
theoretical-weight all strategies, markets vs DOF



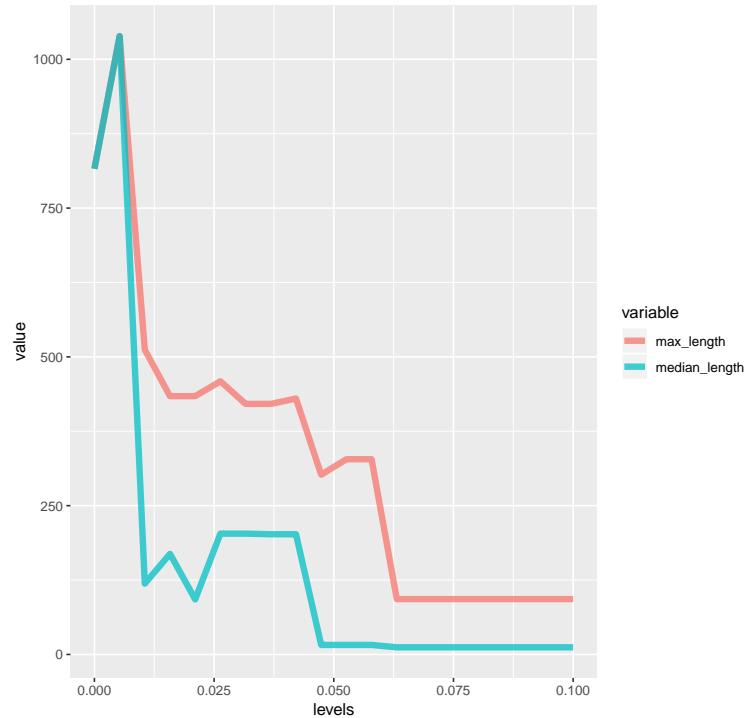
theoretical-weight vs DOF drawdowns, stoploss=5



theoretical-weight vs DOF drawdowns



theoretical-weight vs DOF drawdowns



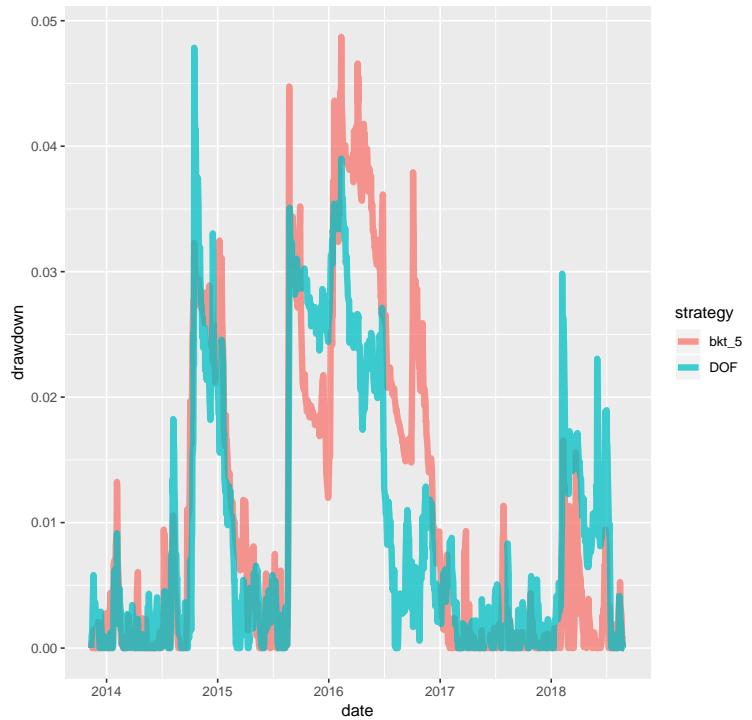
5.3 historical weights

Portfolio composition	
Zero Cost Call Spread Collars	43 %
Zero Cost Put Spread Collars	19 %
Zero Cost Risk Reversals	7%
Zero Cost Put Spread Ratios	15 %
Short Straddles	16 %

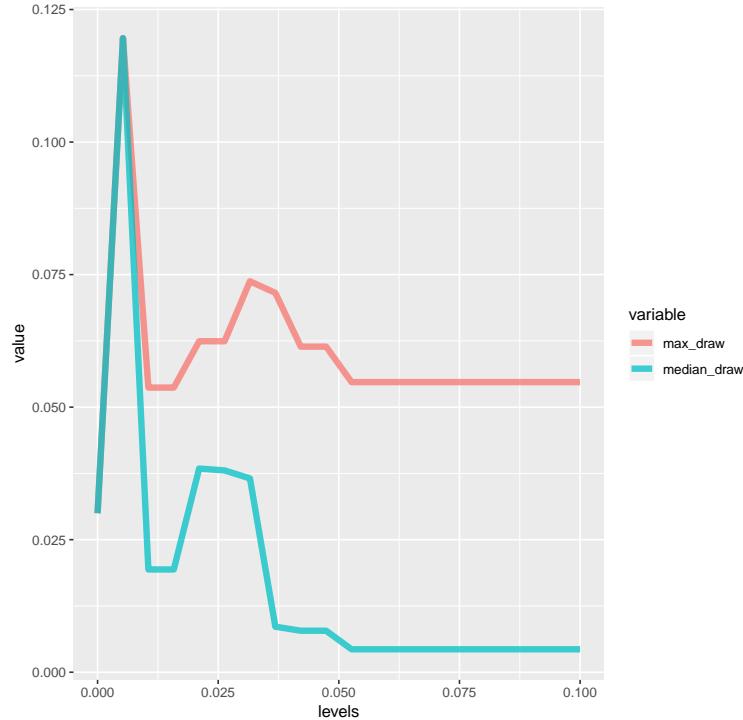
historical-weight all strategies, markets vs DOF



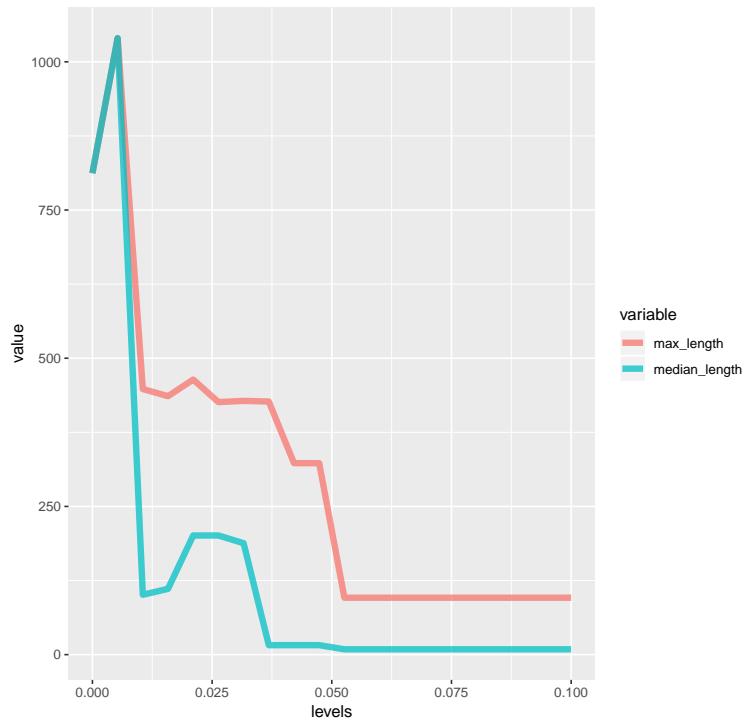
historical-weight vs DOF drawdowns, stoploss=5



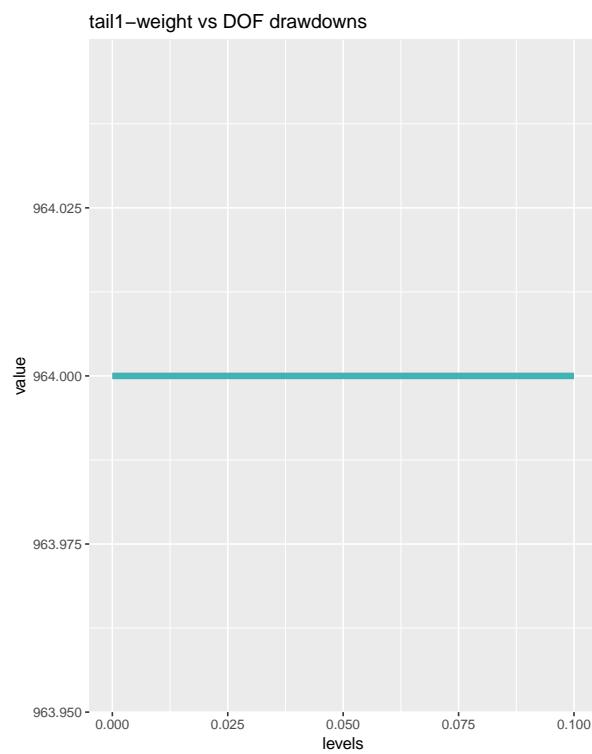
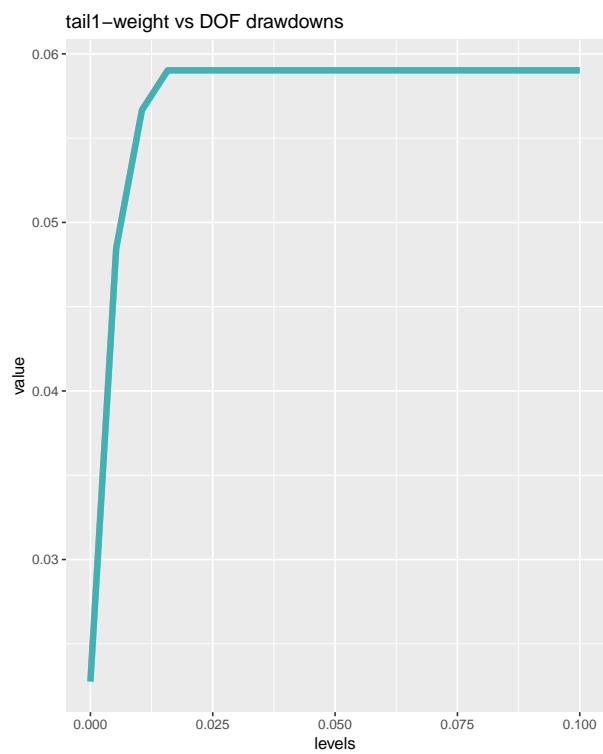
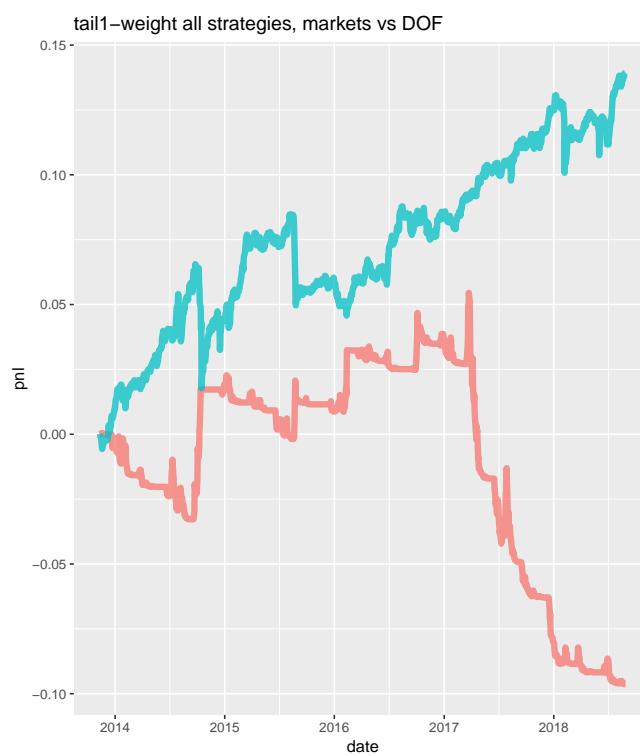
historical-weight vs DOF drawdowns



historical-weight vs DOF drawdowns



5.4 delta 1 teenies



6 Drawdown mitigation: SPX “teenies”

Downside mitigation can be achieved using “teenies”, i.e. very low delta puts (50 to 300 bps). These options almost never pay out anything at settlement. However, their value tends to spike during market-wide bouts of liquidation. We can take advantage of this by holding some amount of these and selling them back whenever their value spikes by a pre-determined amount. The combined portfolio should have smaller returns but much reduced drawdowns. The key to this risk-mitigation strategy is that we use a “stop-profit” on the tail-protection portfolio rather than a “stop-loss” on the risk-premium portfolio. The reason this works better than stop-losses is that volatility risk premia are very strongly mean-reverting. It is better to buy tail-risk when the market is calm and then sell volatility on the pop rather than stop-out the risk premium collection portfolio on the pop and buy the volatility highs. (tail risk becomes vol risk on a strong selloff, the teenies become 5-10 deltas pretty quickly)

Quarterly rolling 2-delta SPX puts vs DOF

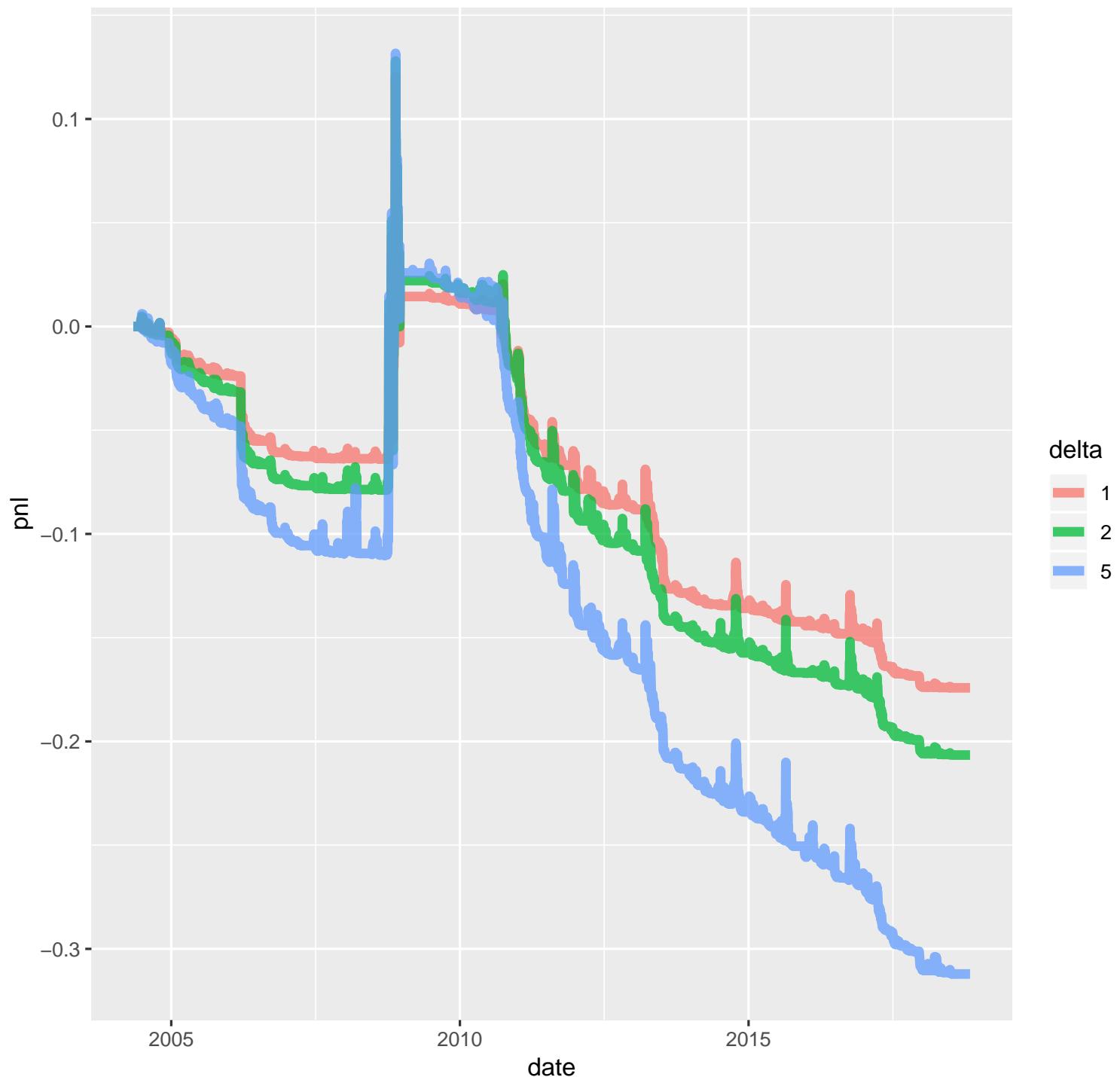


We examine the performance of a tail-risk hedge during the largest draw-down episodes

largest DOF drawdown episodes, optimum teeny performance								
episode	max_draw	date_draw	date_start	date_end	teeny1	teeny2	teeny5	
1	63	4.78	2014-10-16	2014-09-25	2015-03-04	1.66	1.89	2.27
2	84	3.9	2016-02-11	2015-08-11	2016-08-05	1.74	2.44	3.73
3	152	2.98	2018-02-06	2018-01-09	2018-07-17	0.01	0.02	0.05
4	54	1.83	2014-08-08	2014-07-30	2014-09-02	0.01	0.11	0.15
5	88	1.29	2016-11-15	2016-08-15	2017-02-28	1.89	2.08	2.38
6	17	0.91	2014-02-04	2014-01-21	2014-02-25	0.17	0.23	0.4
7	118	0.83	2017-08-11	2017-08-08	2017-08-15	0	0.01	0.02
8	73	0.66	2015-05-07	2015-04-13	2015-05-27	0.11	0.14	0.22
9	1	0.58	2013-11-19	2013-11-12	2013-12-06	0	0	0
10	76	0.58	2015-06-30	2015-06-02	2015-07-17	0	0.01	0.03

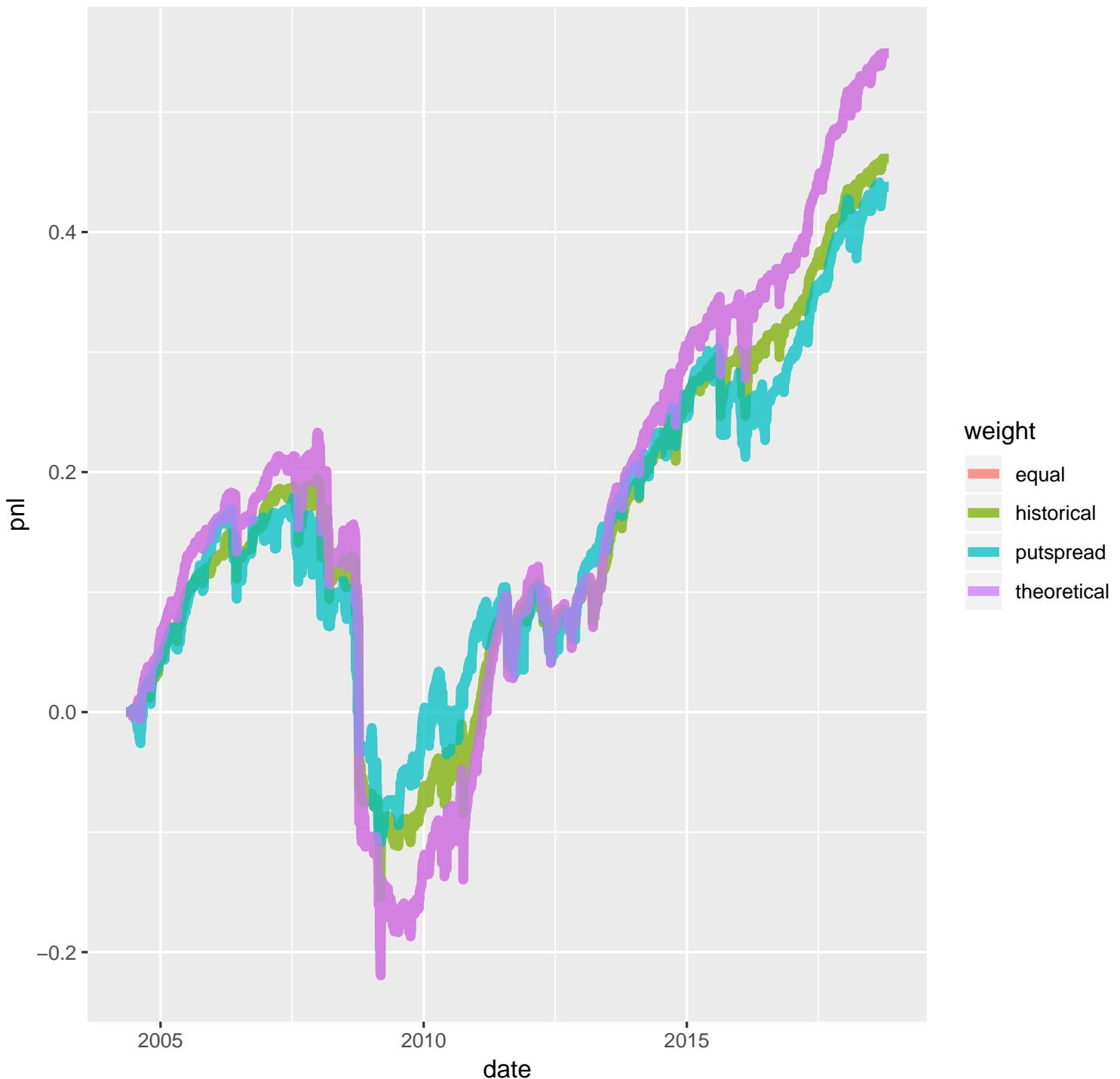
The key metric for choosing a tail hedge is the ratio of premium to potential upside. We expect to rarely be able to recoup any of the premium, since tail options almost never end up paying out anything. From this viewpoint, lower delta options are the best choice. For the examples examined in this report, 1-delta SPX puts have the best ratio of premium to "price pop" during a selloff.

Tail hedges: 1, 2 and 5 deltas



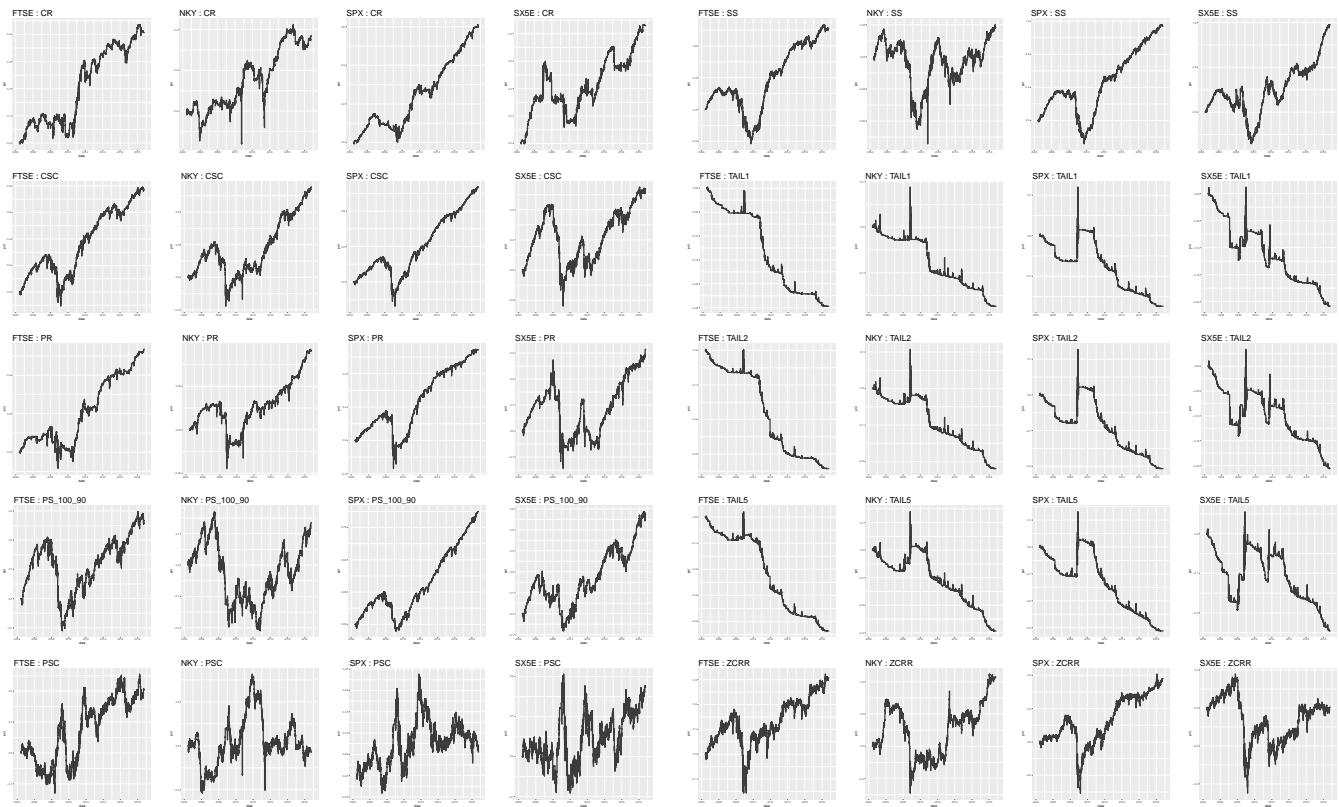
7 All backtests

All backtests

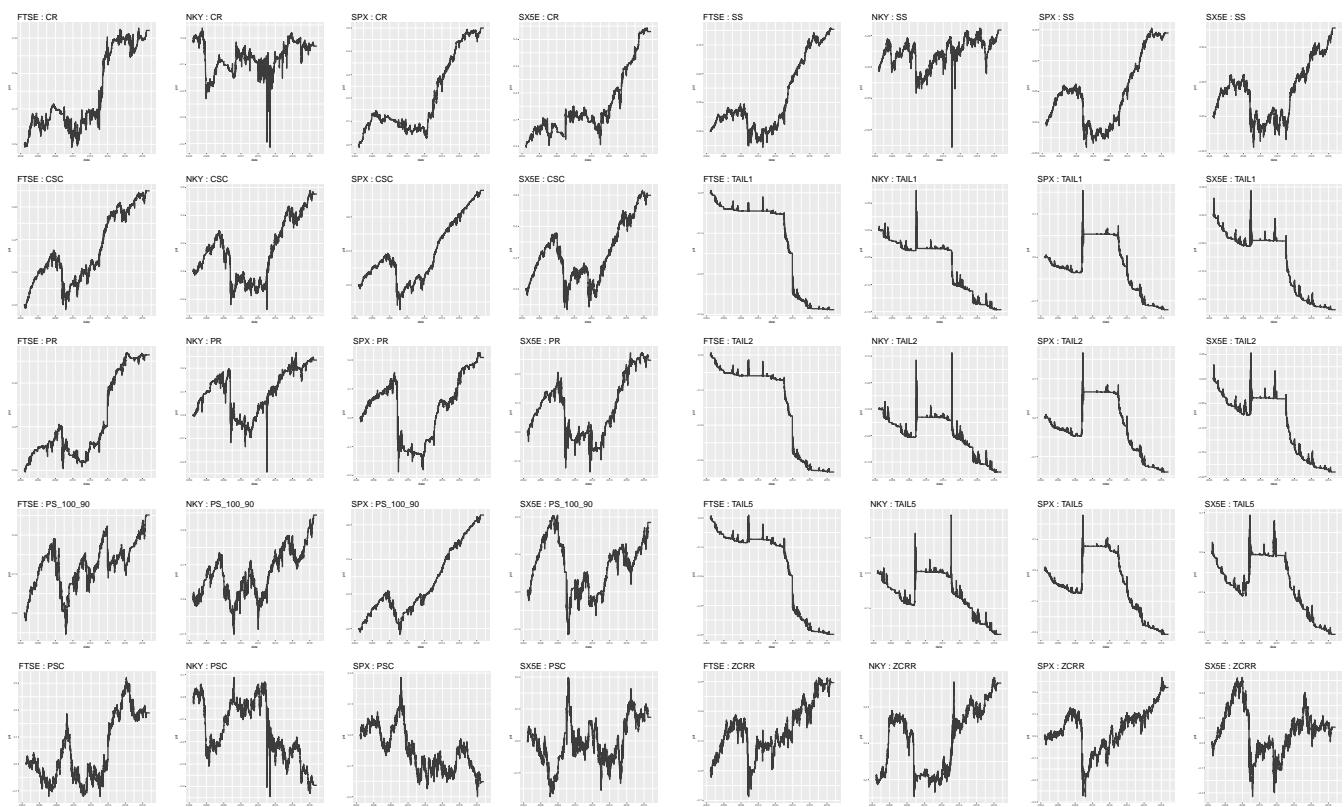


8 Strategies vs market backtests

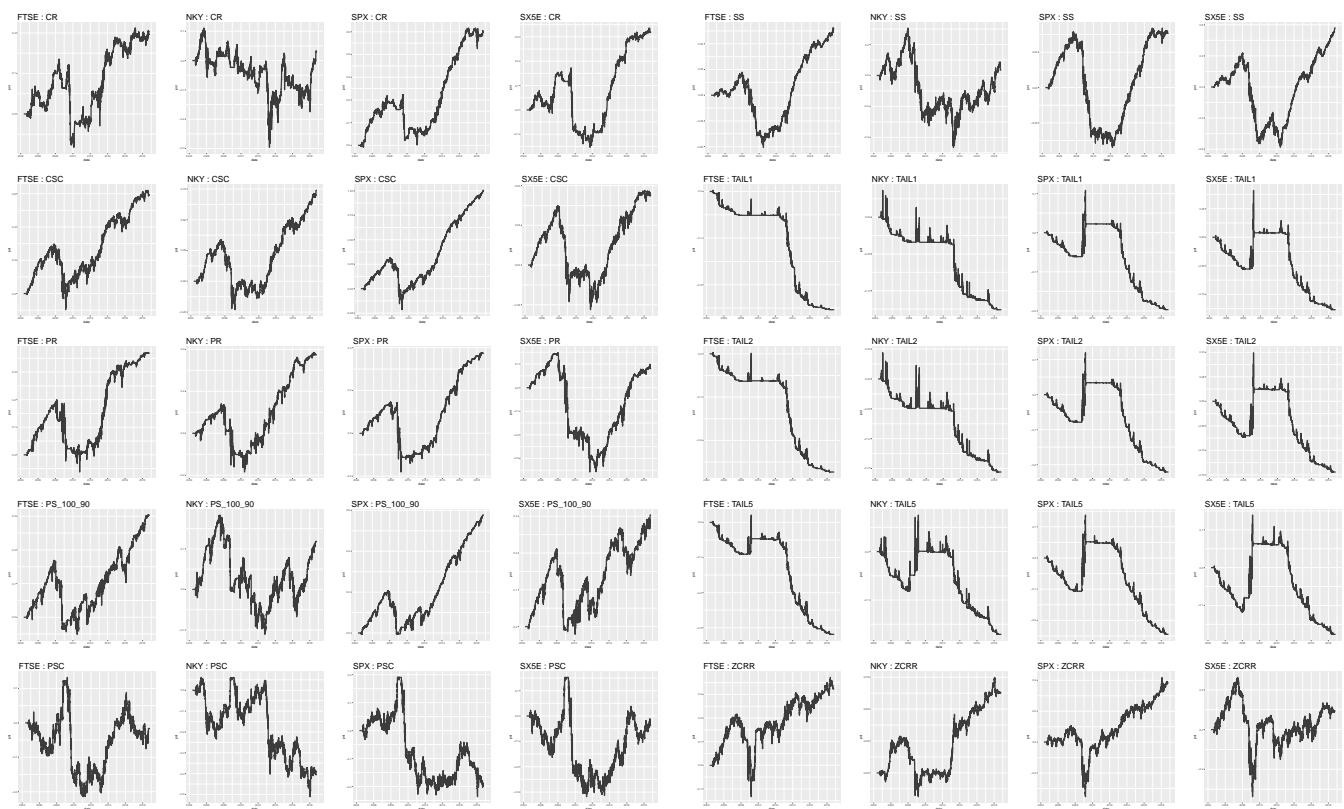
8.1 3m maturity



8.2 6m (Jun, Dec) maturities

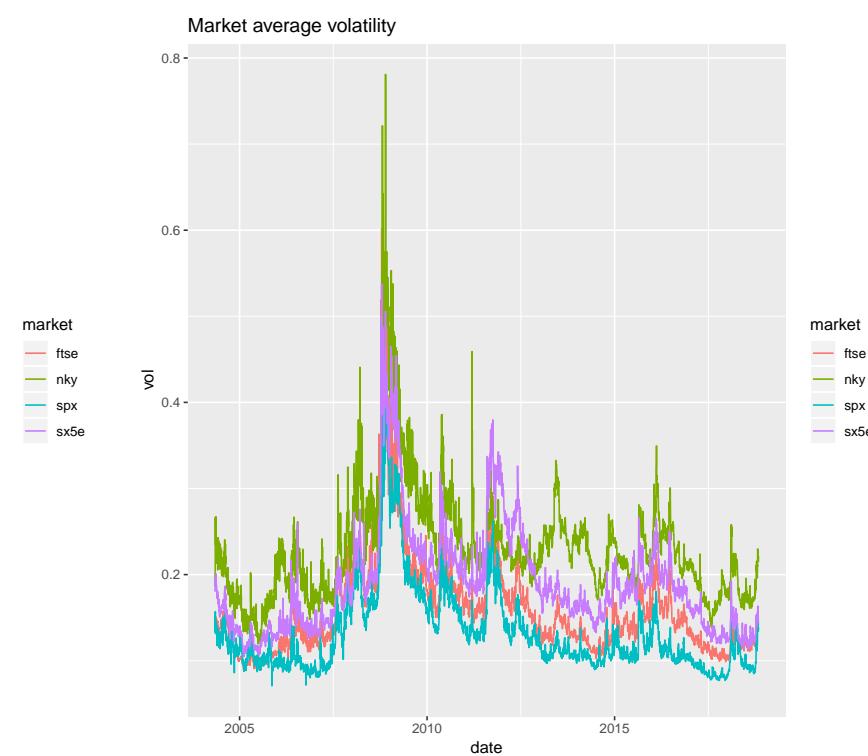
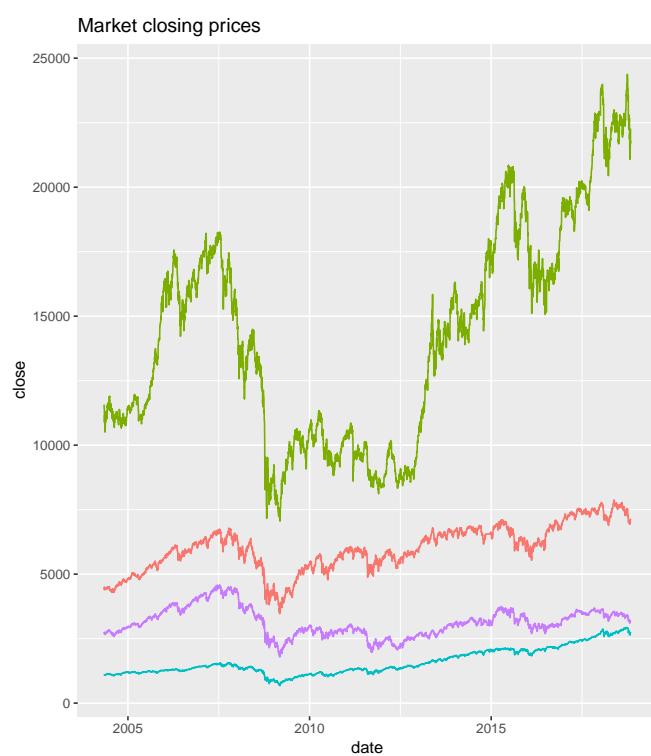


8.3 6m (Mar, Sep) maturities



9 Dataset

We use historical volatility surfaces from JPM's OptionMetrics database. Average volatility, closing prices by market shown below.



10 Backtest algorithm

10.1 backtest inputs

A backtest requires the following inputs:

1. volatility surface
2. schedule
3. payoff

10.2 volatility surface input

The volatility surface input is an R `data.table` containing volatility marks for all markets, strikes and maturities. The data is stored in "long format", so every volatility mark occupies a single row. The schema is as follows:

volatility surface	
Date	reval date
Strike	option strike
Days	option maturity
ImpliedVol	implied volatility
ClosePrice	closing price underlying
market	market name

10.3 schedule input

The schedule input is an R `data.table` containing one row for each date and market we want bactested p&l for. In addition to these data items, the shudule contains the market's close, the average volatility and the current roll period number, start date, end date and days left. Options are assumed to expire on the last day of their respective roll period.

shedule	
date	trading date
market	market
close	closing price
vol	average vol across all strikes and maturities
roll	roll period
start	roll period starting date
end	roll period end date
maturity	days until end of current roll period

10.4 payoff input

the payoff is an R `data.table` containing one row for each of the payoffs we intend to roll on each roll date. For each payoff, the table contains a valuation model and a roll function. The valuation model is used to compute option values on each day in the roll period. The roll function is used to compute option strikes on roll dates. The valuation model is calld for each day in the shudule. The roll function is called only on roll period start days

payoff	
model	function(spot,strike,maturity,rate,volatility)
strike	function(spot,volatility)

10.5 backtest workflow

- start with schedule
- (not vectorized) compute table of strikes for each roll date (date is equal to start of roll period) by calling roll function. Since roll dates are months apart, the roll function is called relatively infrequently and does not need to be optimized.
- LEFT JOIN option strikes on roll period. strikes are constant over the whole roll period. We use the `data.table X[Y]` idiom to achieve this, it is very fast.
- (vectorized) compute maturity, strike pillars surrounding contract economics. We use the `findInterval` function for this, which is very fast.
- LEFT JOIN surface volatilities on surrounding pillars. `data.table`'s `X[Y]` idiom is used here too.
- (vectorized) perform bilinear interpolation. This is done on the whole `data.table` in one go since the bilinear interpolation formula is vectorizable.
- (vectorized) call valuation model, store eval. The model needs to support vectorized operation, there is a single call with vectors of spot, vol, maturity, strike and rate as inputs.

The calculations outlined are vectorizable. This means that the backtest is quite fast. We can backtest a single option on a single market over a 20-year period in about 0.5 seconds.

10.6 volatility interpolation

The `interpolate_vol` function takes a `reval` date, a strike and a maturity as inputs and performs volatility interpolation on a single-market volatility surface. This calculation needs to be performed for every day, every market and every strike in the strategy. This means that the number of interpolations can run into the hundreds of thousands. It is therefore really important to be as efficient as possible. This means we need to vectorize operations as much as we can. The calculation as shown works well for a single market. To do interpolation over multiple markets (for example from a multi-market schedule), we will use the `data.table` **keyby** feature to split the schedule by market and perform the interpolation on every subset separately, inside the `data.table`'s `J` expression.

```
interpolate_vol<-function(date ,strike ,days ,vsurf)
{
  # vector of strike pillars in vol surface
  strikes<-sort(unique(vsurf$Strike))

  # vector of maturity pillars in vol surface
  maturities<-sort(unique(vsurf$Days))

  # volatility surface pillars surrounding contract
  # strike and expiry days
  option_dets<-data.table(
    date=date ,
    lo_strike=strikes[findInterval(strike ,strikes)] ,
    hi_strike=strikes[findInterval(strike ,strikes)+1] ,
    lo_mat=maturities[findInterval(days ,maturities)] ,
    hi_mat=maturities[findInterval(days ,maturities)+1]
  )

  # left join lower left corner's volatility
  vol_ll<-merge(
    x=option_dets ,
    y=vsurf ,
    by.x=c("date","lo_strike","lo_mat") ,
    by.y=c("Date","Strike","Days")
  )$ImpliedVol

  # left join higher left corner's volatility
  vol_lh<-merge(
    x=option_dets ,
    y=vsurf ,
    by.x=c("date","lo_strike","hi_mat") ,
    by.y=c("Date","Strike","Days")
  )$ImpliedVol

  # left join right lower corner's volatility
  vol_hl<-merge(
    x=option_dets ,
    y=vsurf ,
    by.x=c("date","hi_strike","lo_mat") ,
    by.y=c("Date","Strike","Days")
  )$ImpliedVol

  # left join right higher corner's volatility
  vol_hh<-merge(
    x=option_dets ,
    y=vsurf ,
    by.x=c("date","hi_strike","hi_mat") ,
    by.y=c("Date","Strike","Days")
  )$ImpliedVol

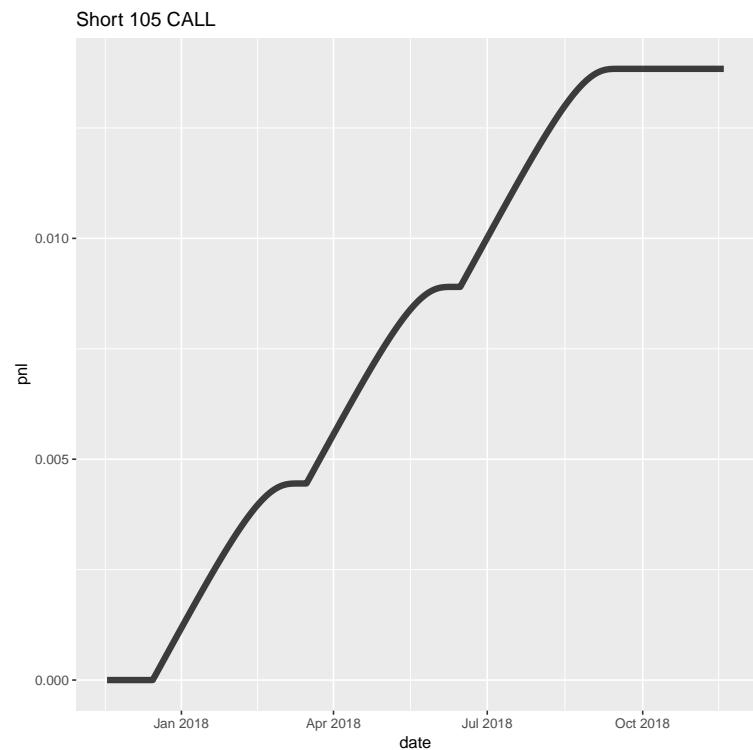
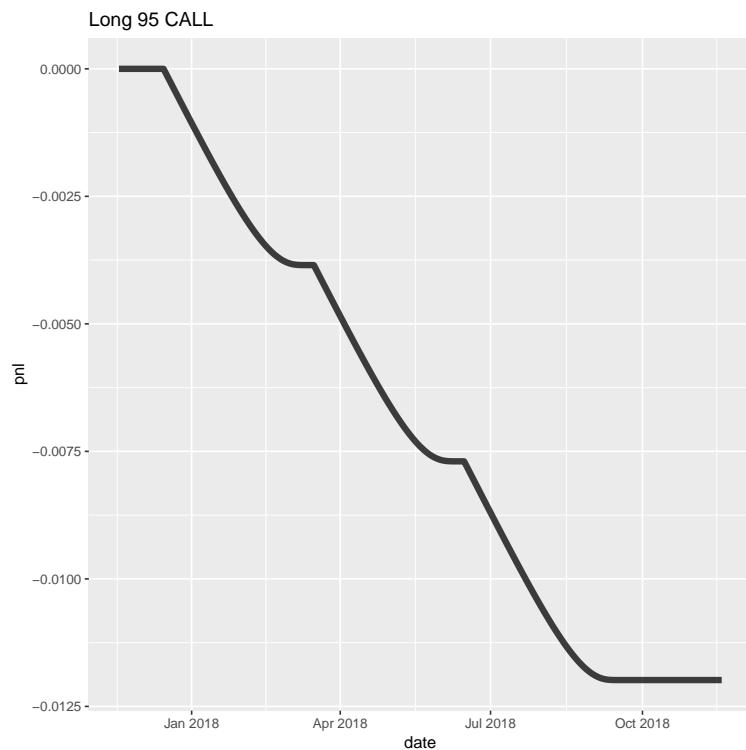
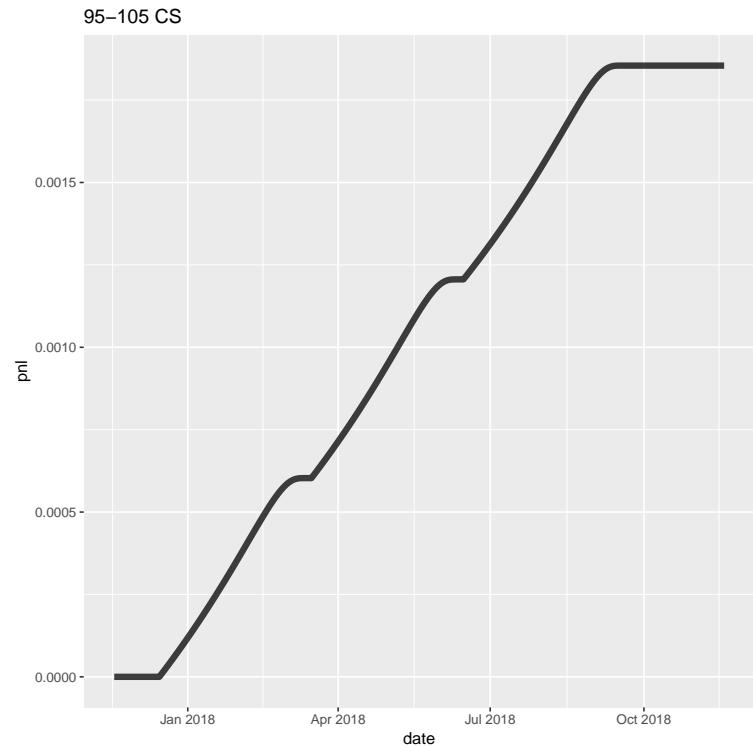
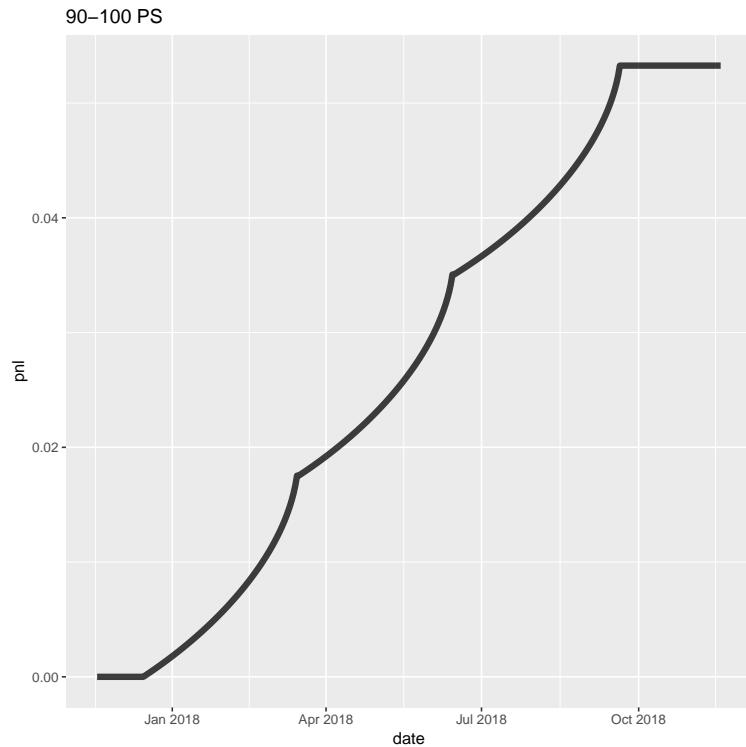
  # coordinates of contract strike, expiry inside surrounding pillar points
  t_strike<-(strike-option_dets$lo_strike)/(option_dets$hi_strike-option_dets$lo_strike)
  t_mat<-(days-option_dets$lo_mat)/(option_dets$hi_mat-option_dets$lo_mat)

  # bilinear interpolation formula,
  # result is vector of volatilities
  rowSums(cbind(
    vol_ll*(1-t_strike)*(1-t_mat) ,
    vol_hl*(t_strike)*(1-t_mat) ,
    vol_lh*(1-t_strike)*(t_mat) ,
    vol_hh*(t_strike)*(t_mat)
  ))
}
```

10.7 Unit tests

We perform a number of backtests for selected strategies on synthetic data to ensure the backtest engine is working as expected.

Synthetic market, constant price at 100, constant vol at 10%



Synthetic market, constant price at 100, constant vol at 10%, 25bp SL, TP

