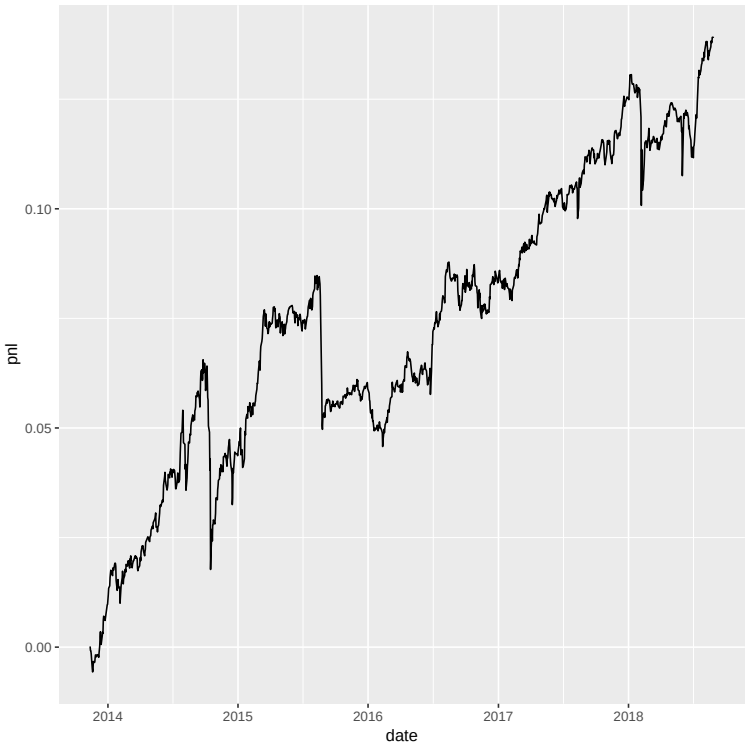
Contents

Document version

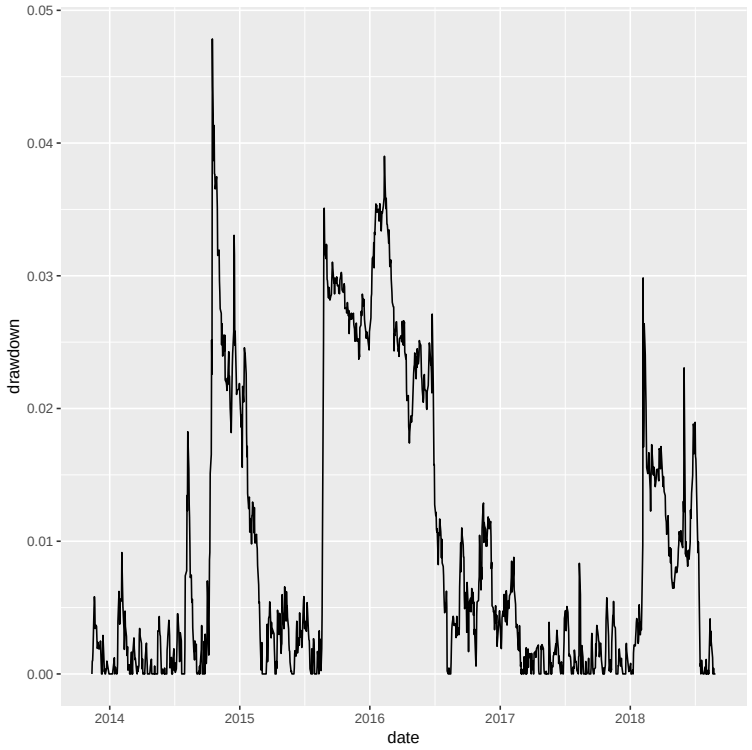
timestamp	2018-11-17 12:14:57
data and code checksum	fc5b947f526b9d5f68830174bd68967a
git id	f8df69836131f3c4c7930b375f191bcccd562a1f
user	nicknassuphis

1 DOF performance

Performance



Drawdown



- 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 Conclusions: Stop-loss vs return quality

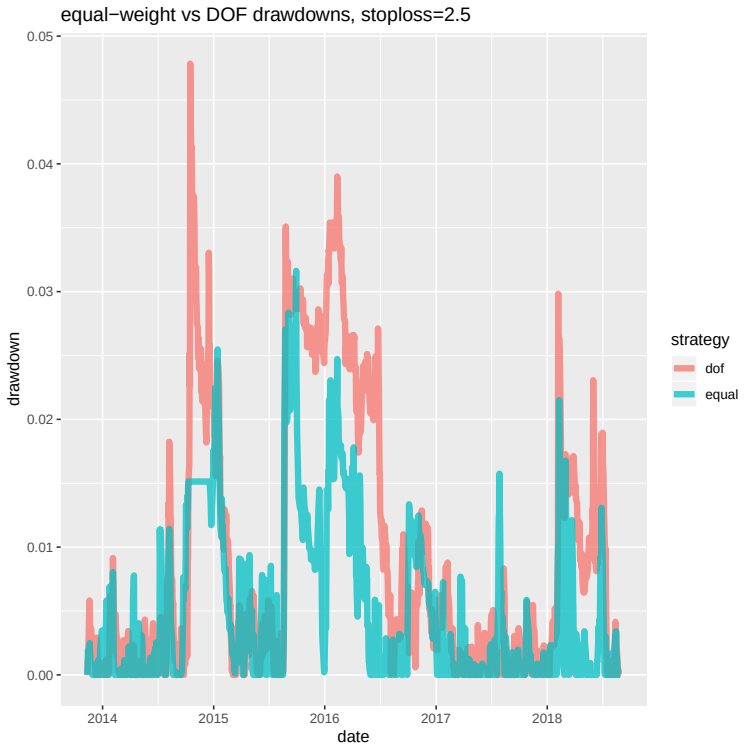
"Small positions with wide stops" are better than "Large positions with tight stops"

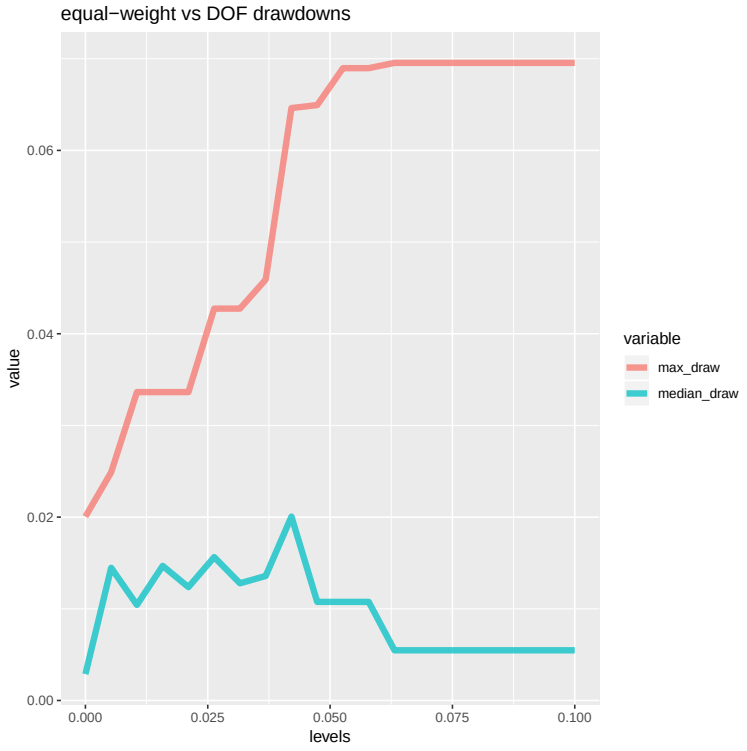
In general, we would expect stop-losses to reduce return quality. The intuition behind this expectation is that post-selloff rebounds are among the highest-sharpe periods in a risk-premium strategy's life. Stop-losses limit drawdowns but also deny us participation in rebounds. On balance, return quality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

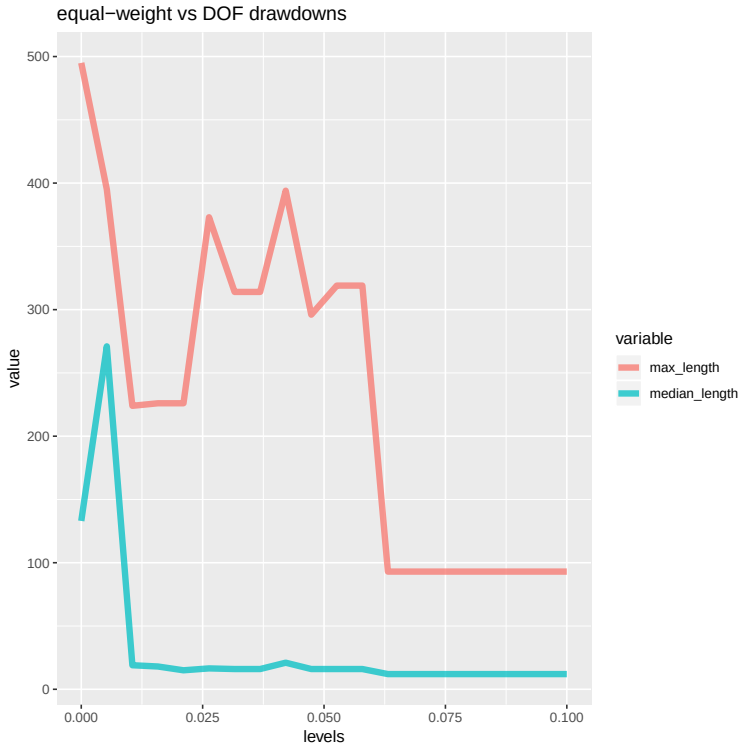
Return quality vs Stop-loss level				
	stop_loss	sharpe	calmar	proxy
1	0	0.59	34.27	equal weight
2	0.83	0.82	10.71	equal weight
3	1.67	0.89	10.5	equal weight
4	2.5	0.86	11.48	equal weight
5	3.33	0.86	10.84	equal weight
6	4.17	0.9	10.35	equal weight
7	5	0.96	10.51	equal weight
8	5.83	1.02	9.69	equal weight
9	6.67	1.13	6.35	equal weight
10	7.5	1.22	5.44	equal weight
11	8.33	1.31	4.48	equal weight
12	9.17	1.44	3.48	equal weight
13	10	1.53	3.15	equal weight
14	10.83	1.63	2.89	equal weight
15	11.67	1.7	2.56	equal weight
16	12.5	1.71	2.52	equal weight
17	13.33	1.78	2.31	equal weight
18	14.17	1.86	2.09	equal weight
19	15	1.91	1.96	equal weight
20	15.83	1.97	1.87	equal weight
21	16.67	1.92	1.84	equal weight
22	17.5	1.95	1.73	equal weight
23	18.33	2.01	1.65	equal weight
24	19.17	1.99	1.66	equal weight
25	20	1.99	1.62	equal weight

3 Weighted strategy backtests

3.1 equal weights







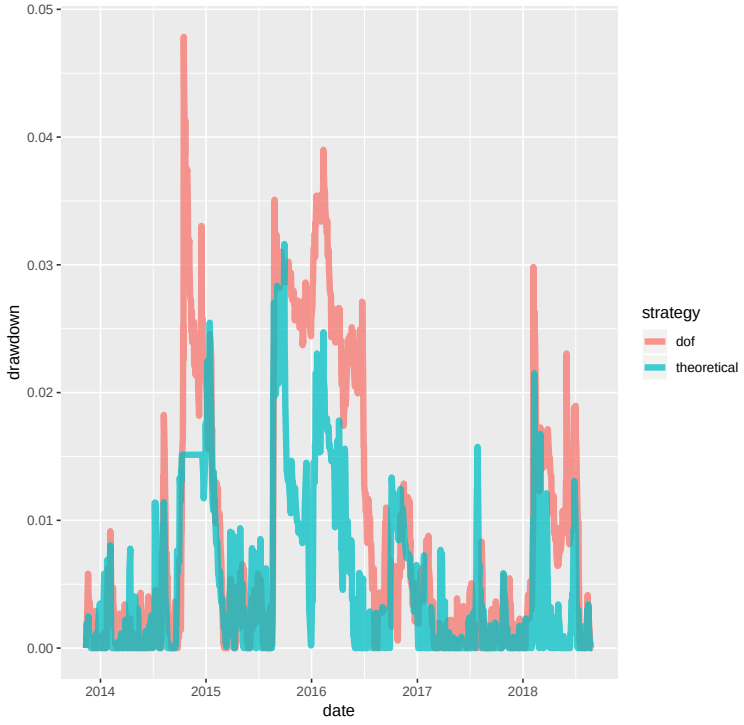
Tighter stops reduce the depth of drawdowns but increase their length. This is fairly typical for risk-premium collection strategies.

3.2 theoretical weights

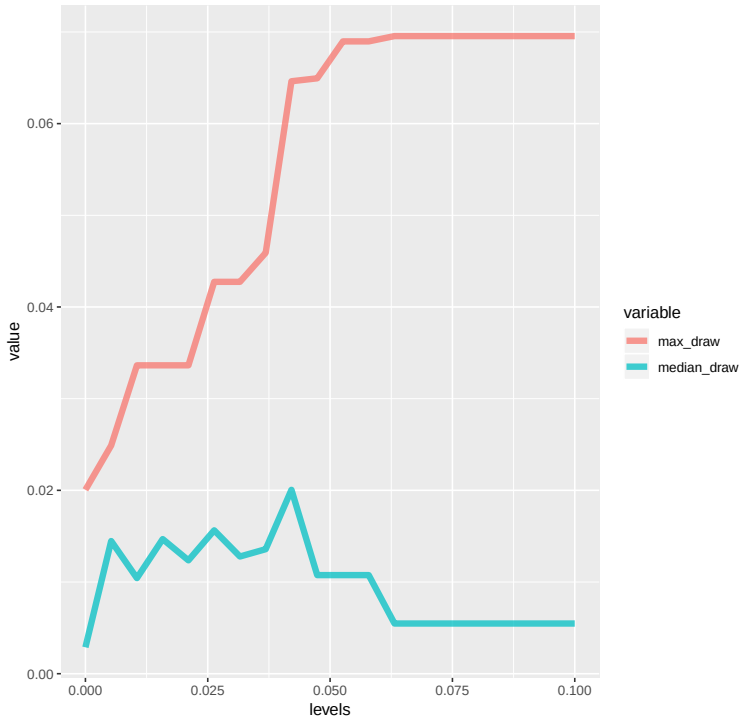
Portfolio composition	
Zero Cost Call Spread Collars	33 %
Zero Cost Put Spread Collars	0 %
Zero Cost Risk Reversals	0 %
Zero Cost Put Spread Ratios	33 %
Short Straddles	33 %

theoretical−weight all strategies, markets vs DOF

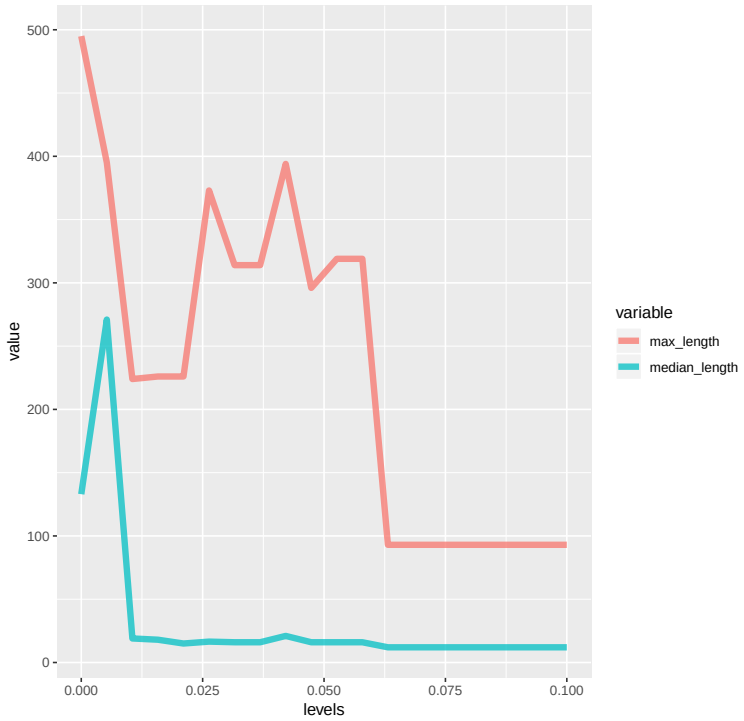
theoretical−weight vs DOF drawdowns, stoploss=2.5



theoretical−weight vs DOF drawdowns

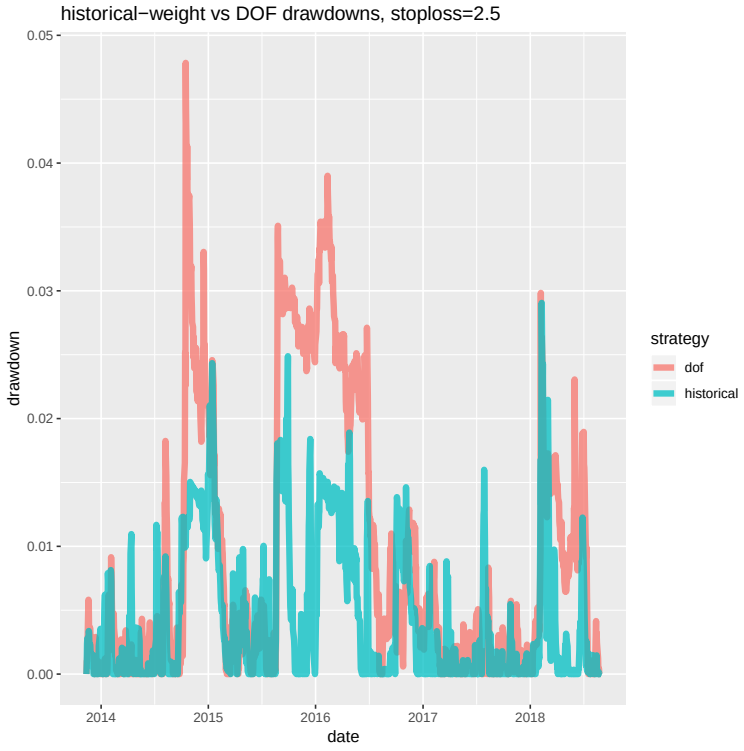


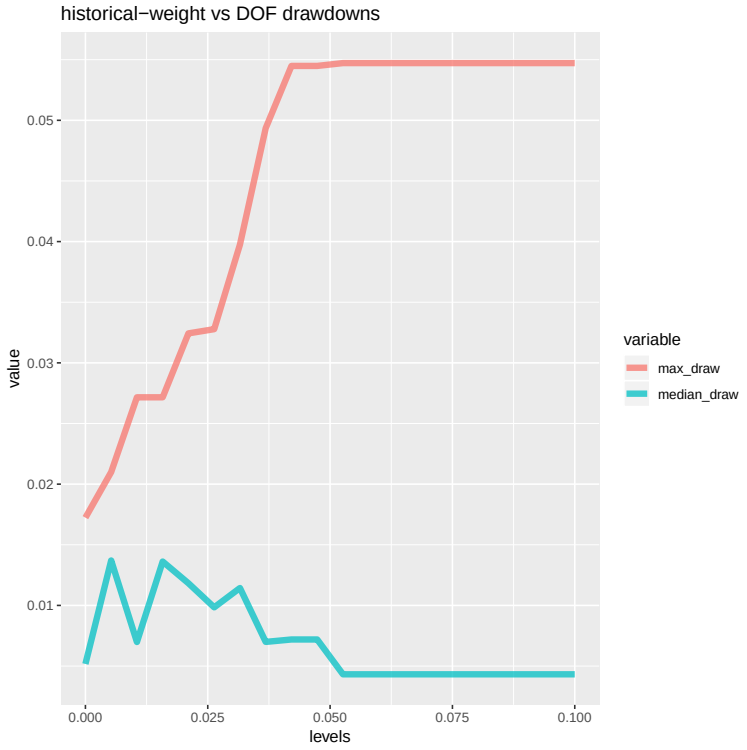
theoretical−weight vs DOF drawdowns

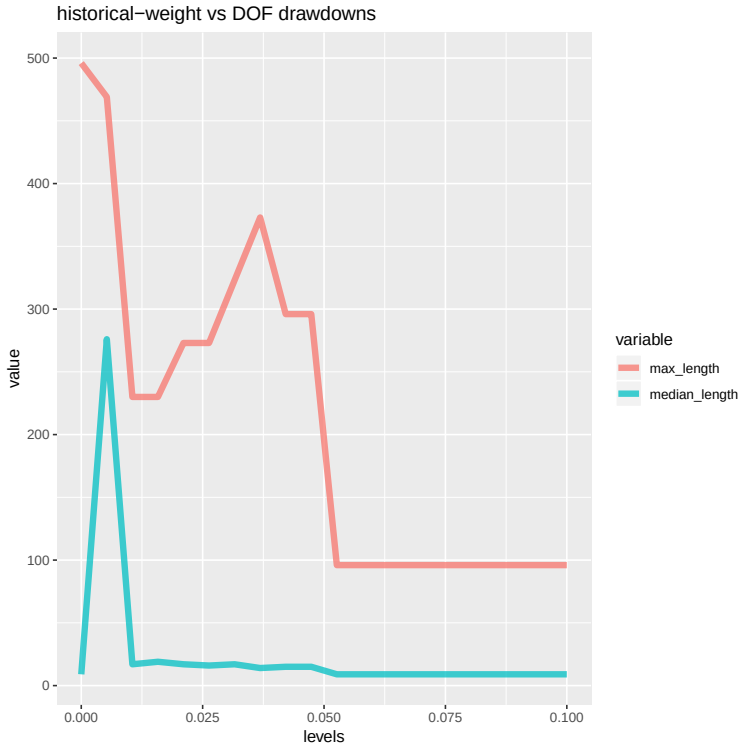


3.3 historical weights

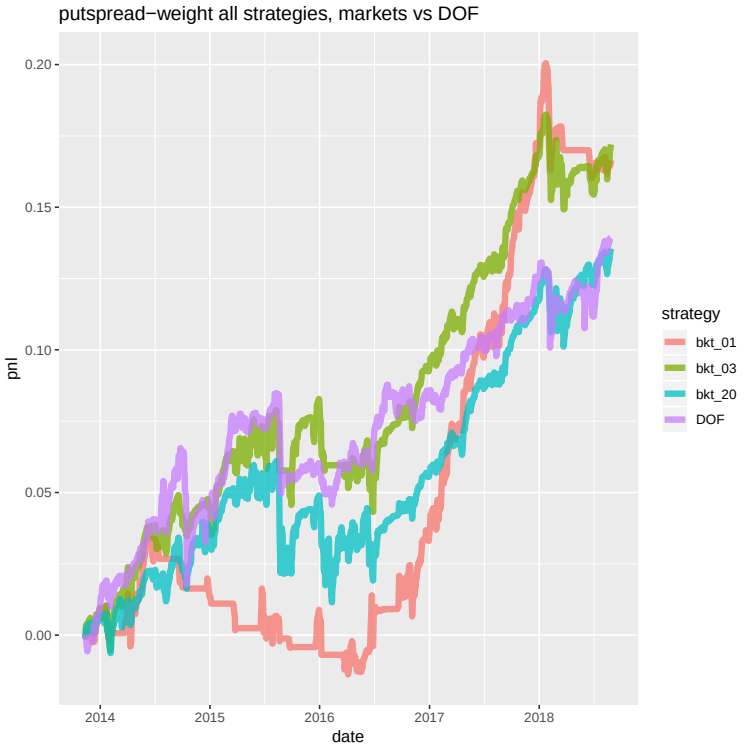
Portfolio composition	
Zero Cost Call Spread Collars	43 %
Zero Cost Put Spread Collars	19 %
Zero Cost Risk Reversals	7%
Zero Cost Put Spread Ratios	15 %
Short Straddles	16 %

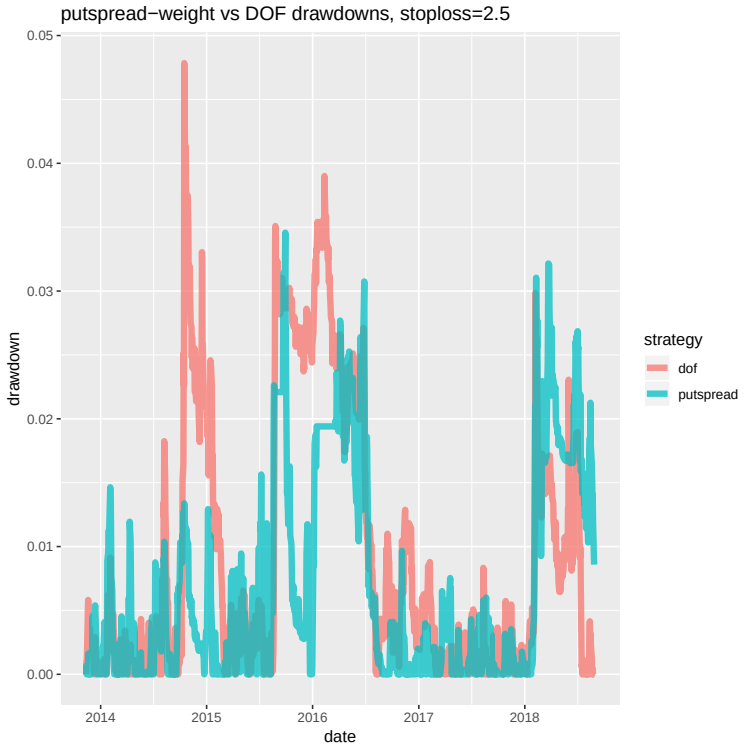


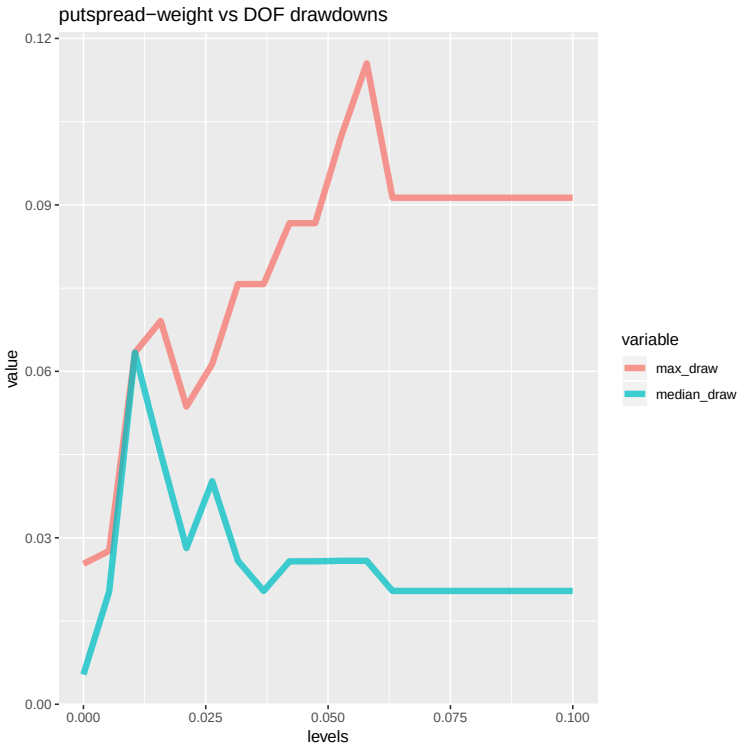


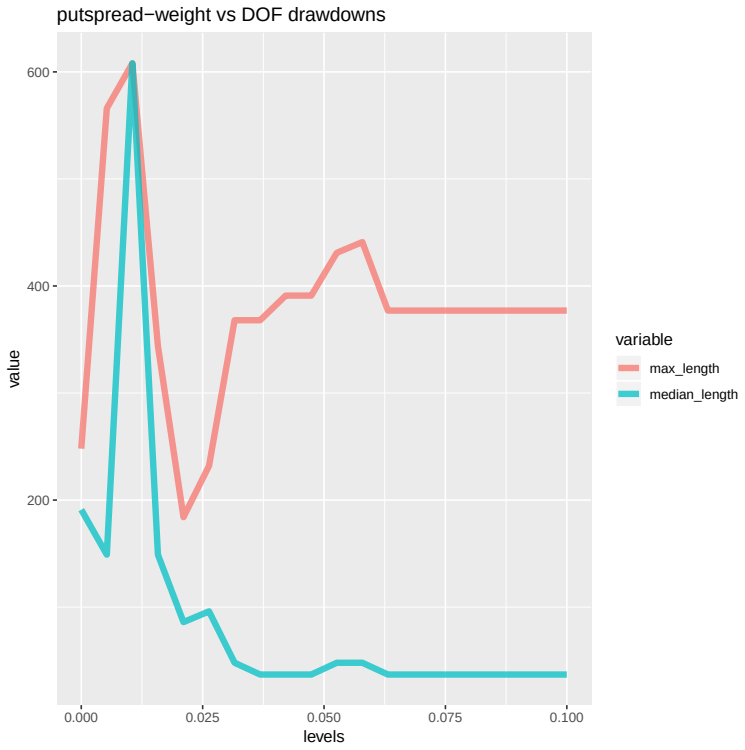


3.4 PS 90 100

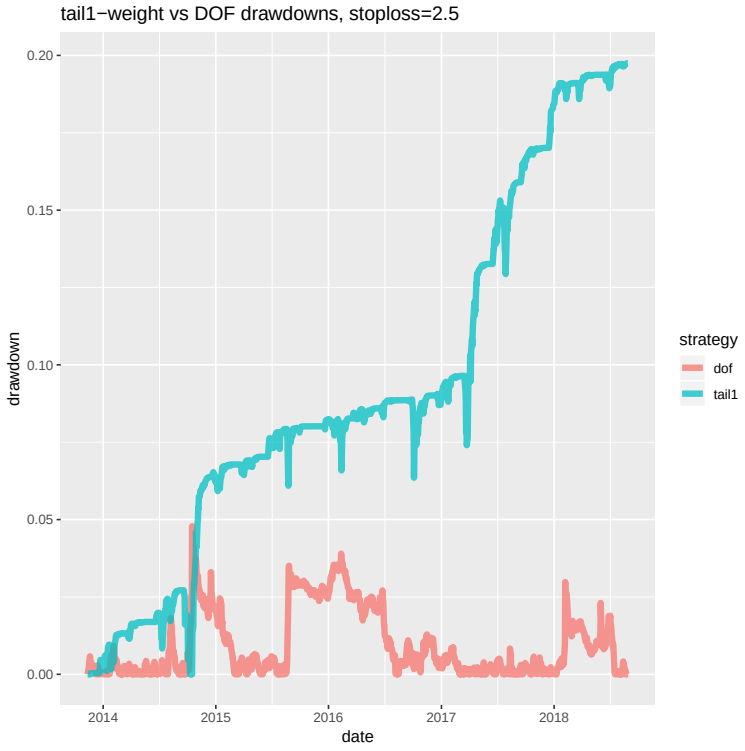

putspread−weight all strategies, markets vs DOF
pnl
date
strategy
bkt_01
bkt_03
bkt_20
DOF

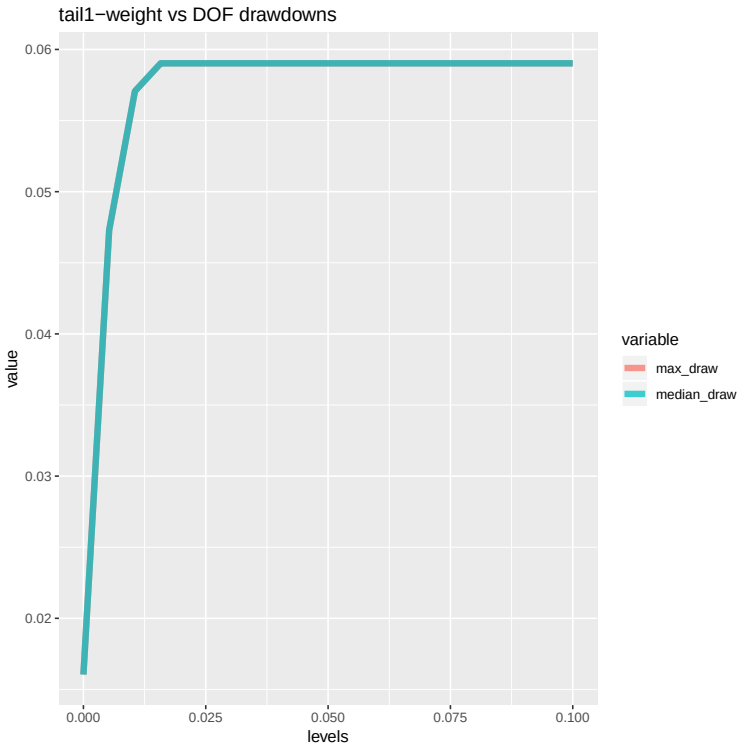

putspread−weight vs DOF drawdowns, stoploss=2.5
drawdown
date
strategy
dof
putspread

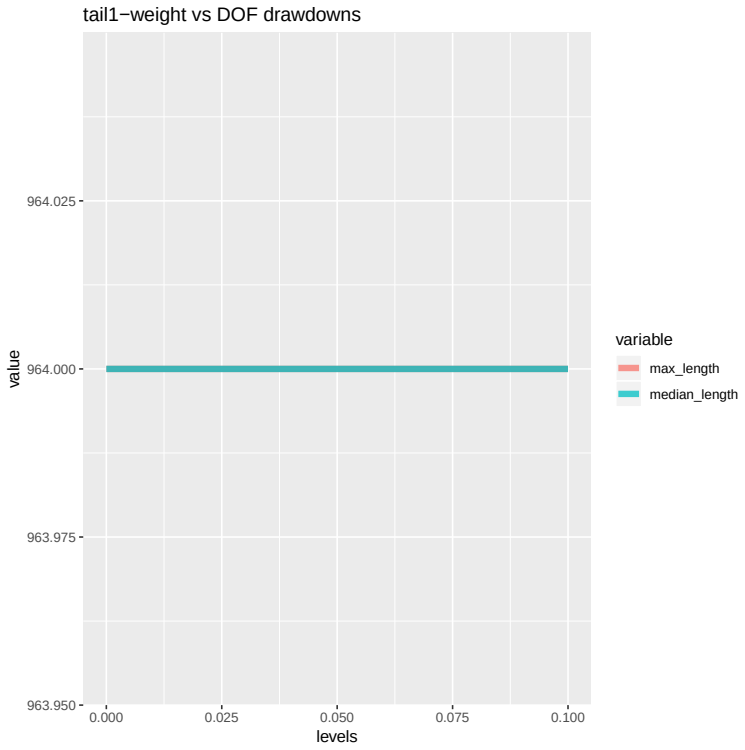

putspread−weight vs DOF drawdowns
value
levels
variable
max_draw
median_draw


putspread−weight vs DOF drawdowns
value
levels
variable
max_length
median_length

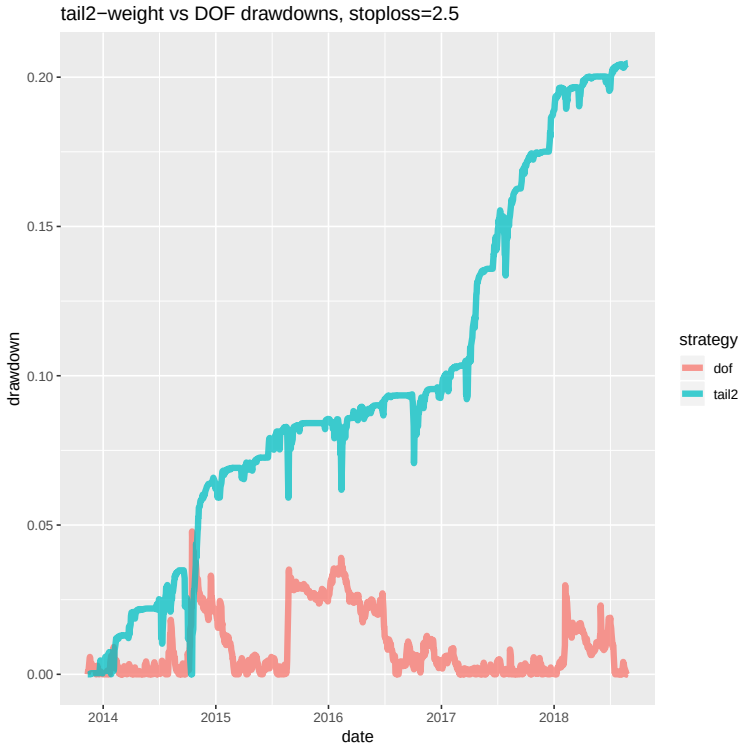
3.5 delta 1 teenies

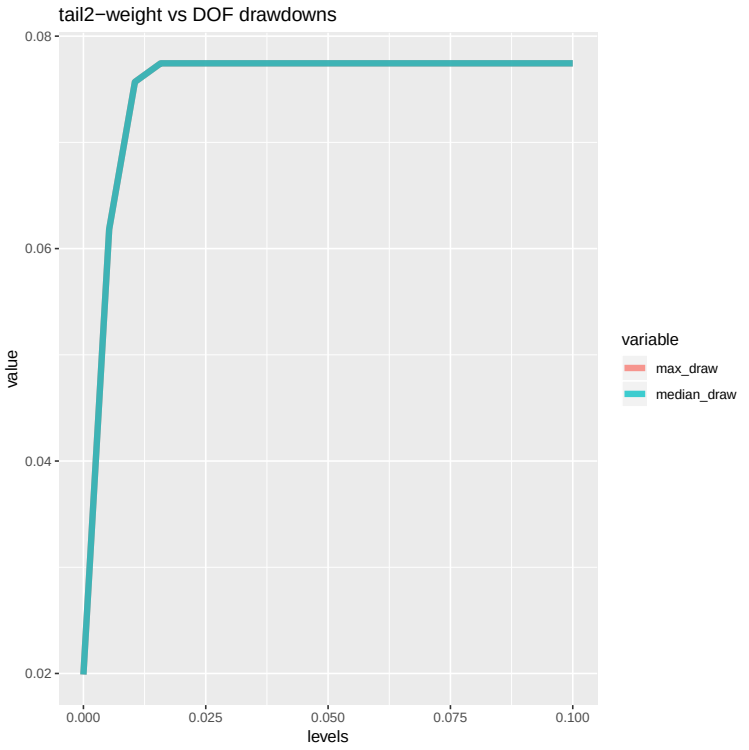


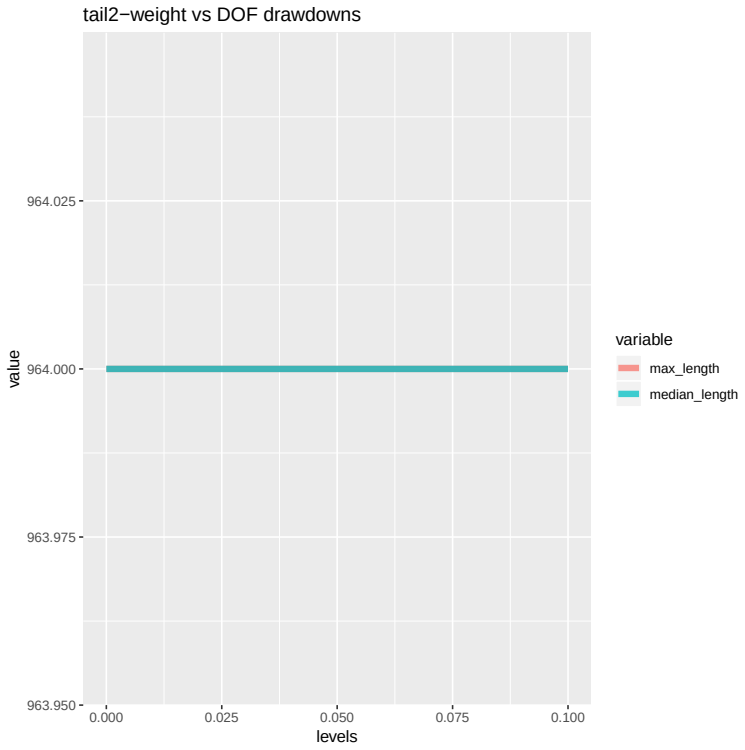




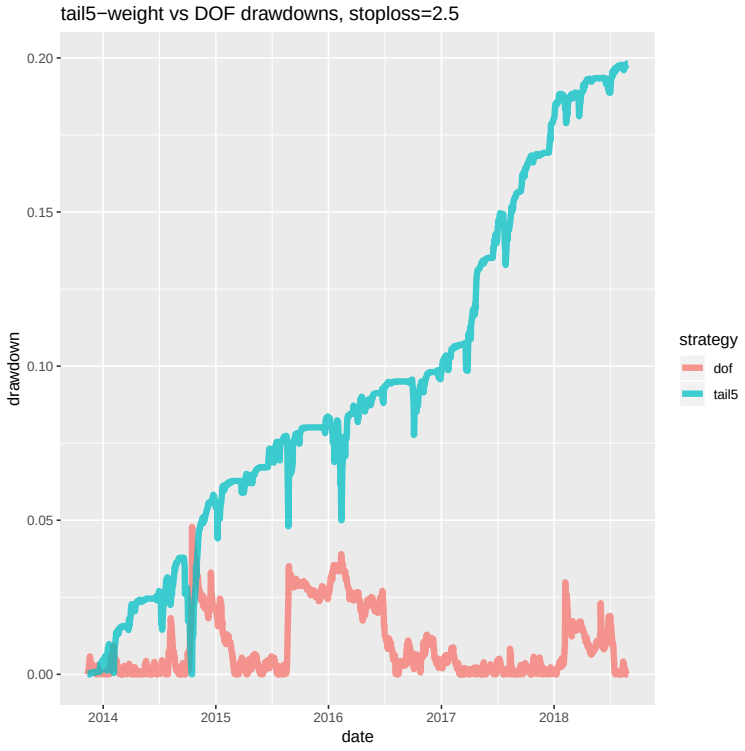
3.6 delta 2 teenies

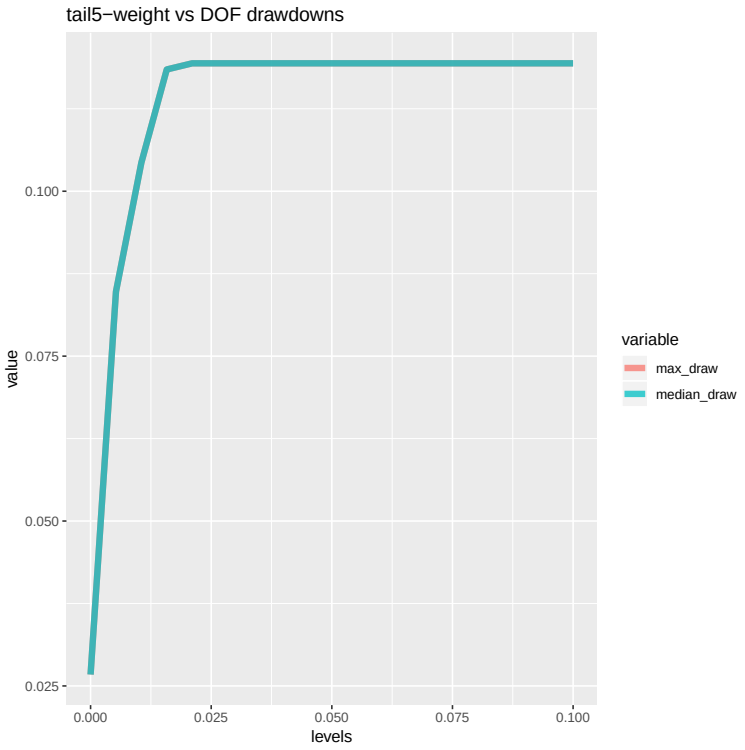


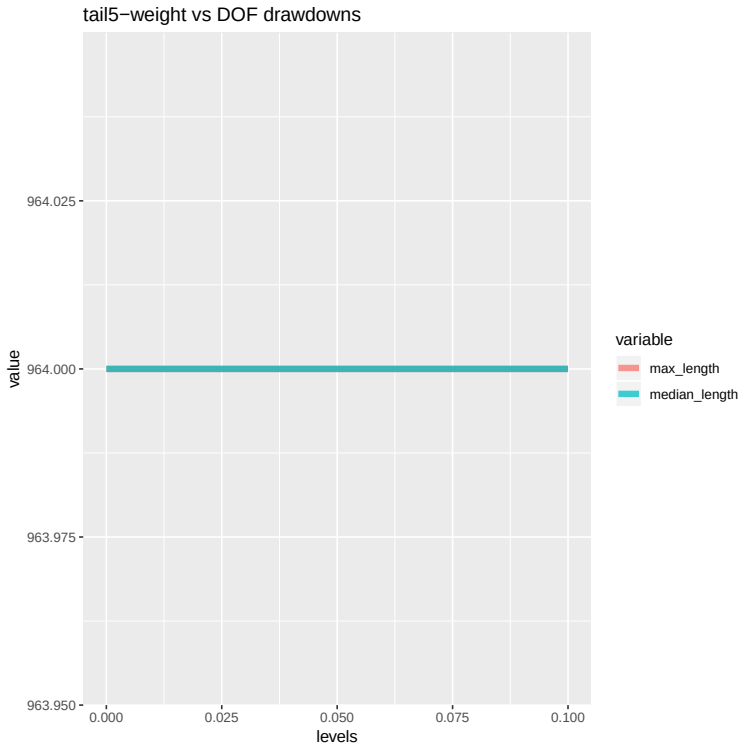




3.7 delta 5 teenies

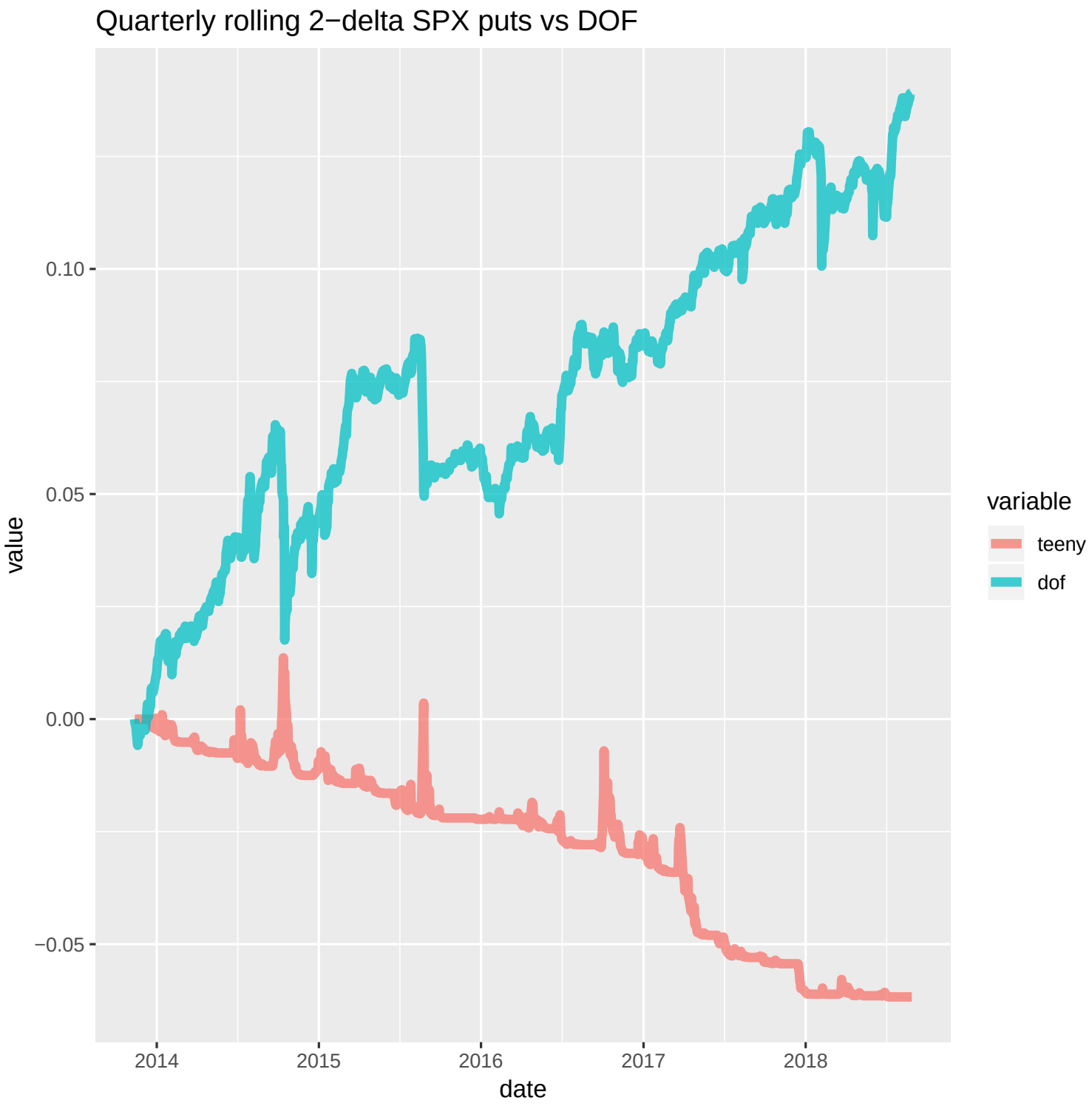






4 Drawdown mitigation: SPX "teenies"

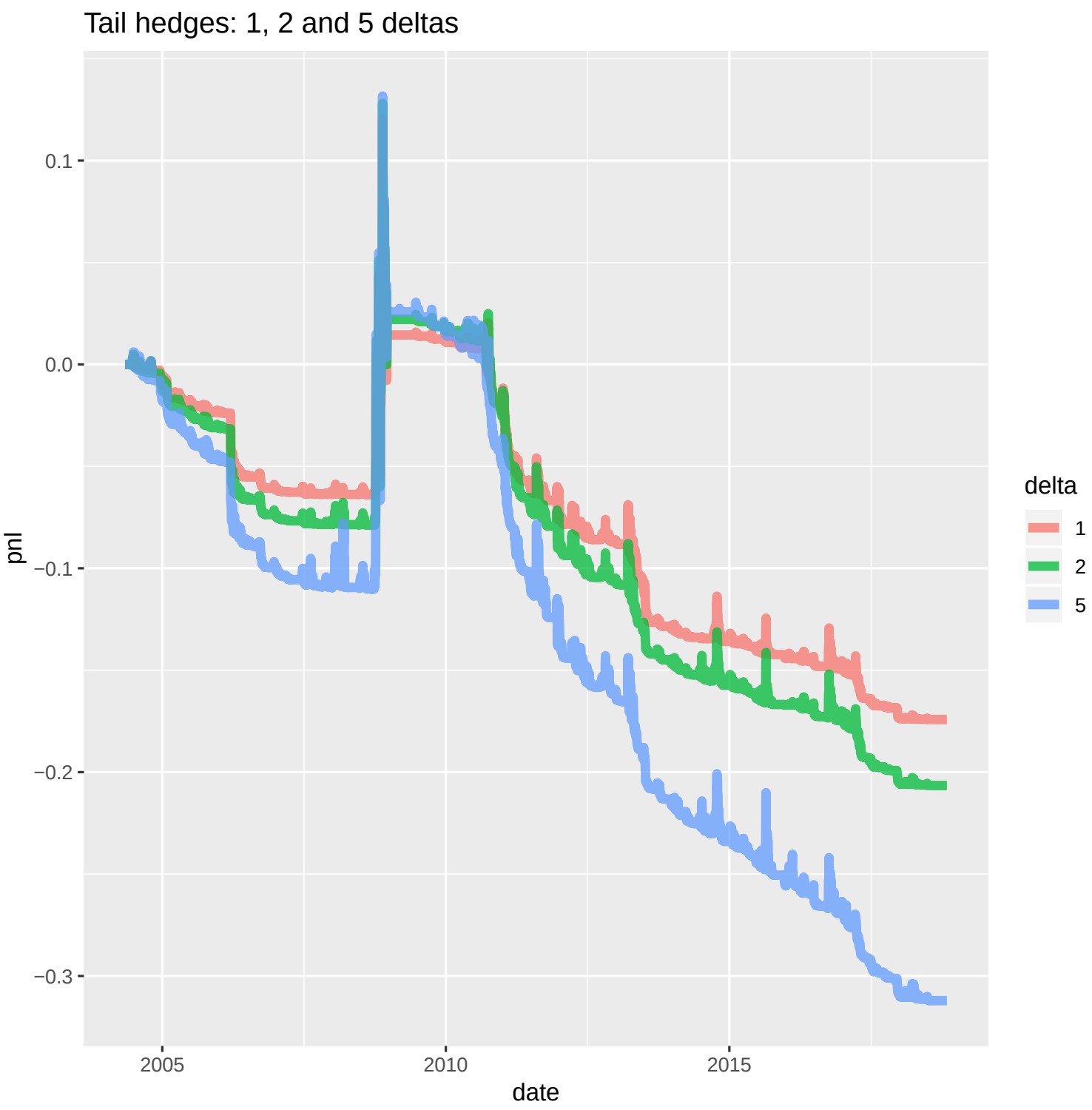
Downside mitigation can be achieved using "teenies", i.e. very low delta puts (50 to 300 bps). These options almost never pay out anything at settlement. However, their value tends to spike during market-wide bouts of liquidation. We can take advantage of this by holding some amount of these and selling them back whever 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)



We examine the performanse 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

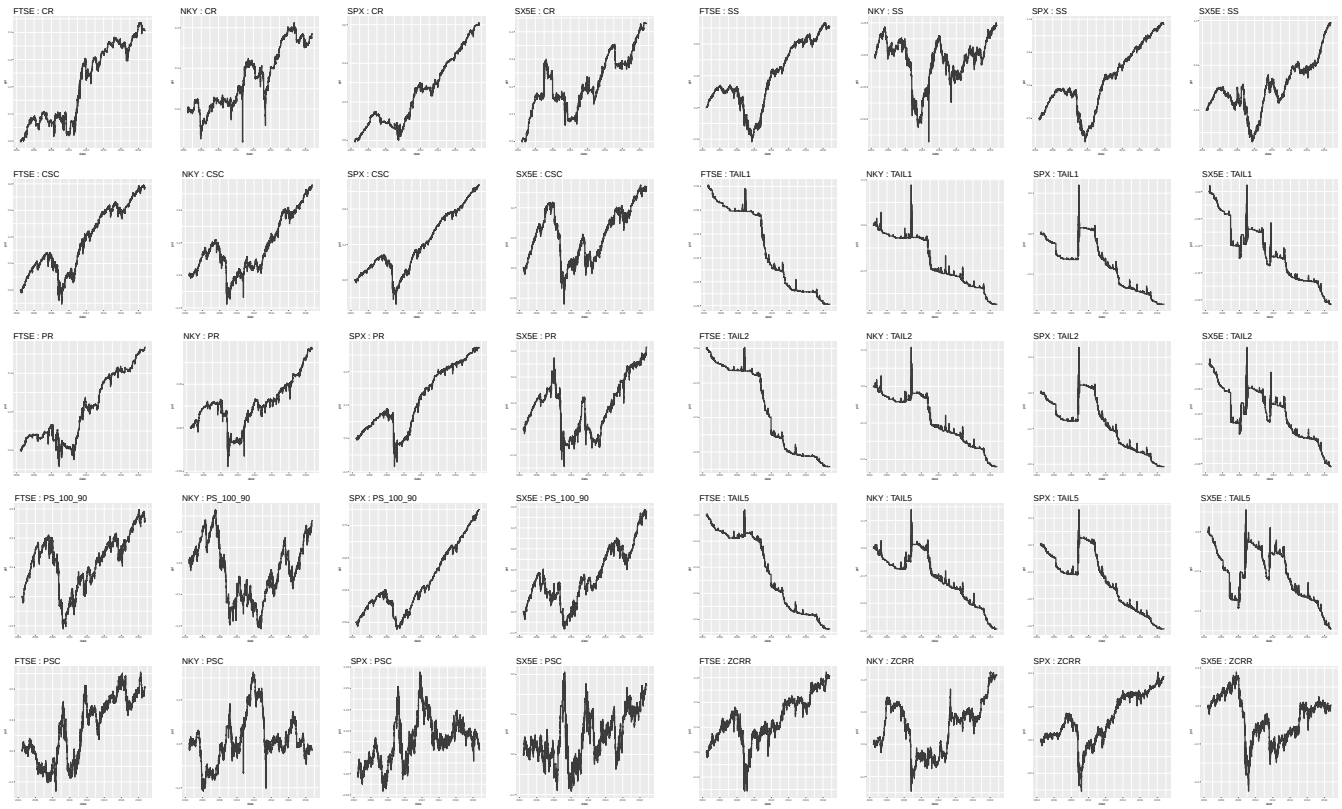
The key metric for choosing a tail hedge is the ratio of premium to potential upside. We expect to rarely be able to recoup any of the premium, since tail options almost never end up paying out anything. From this viewpoint, lower delta options are the best choice. For the examples examined in thise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



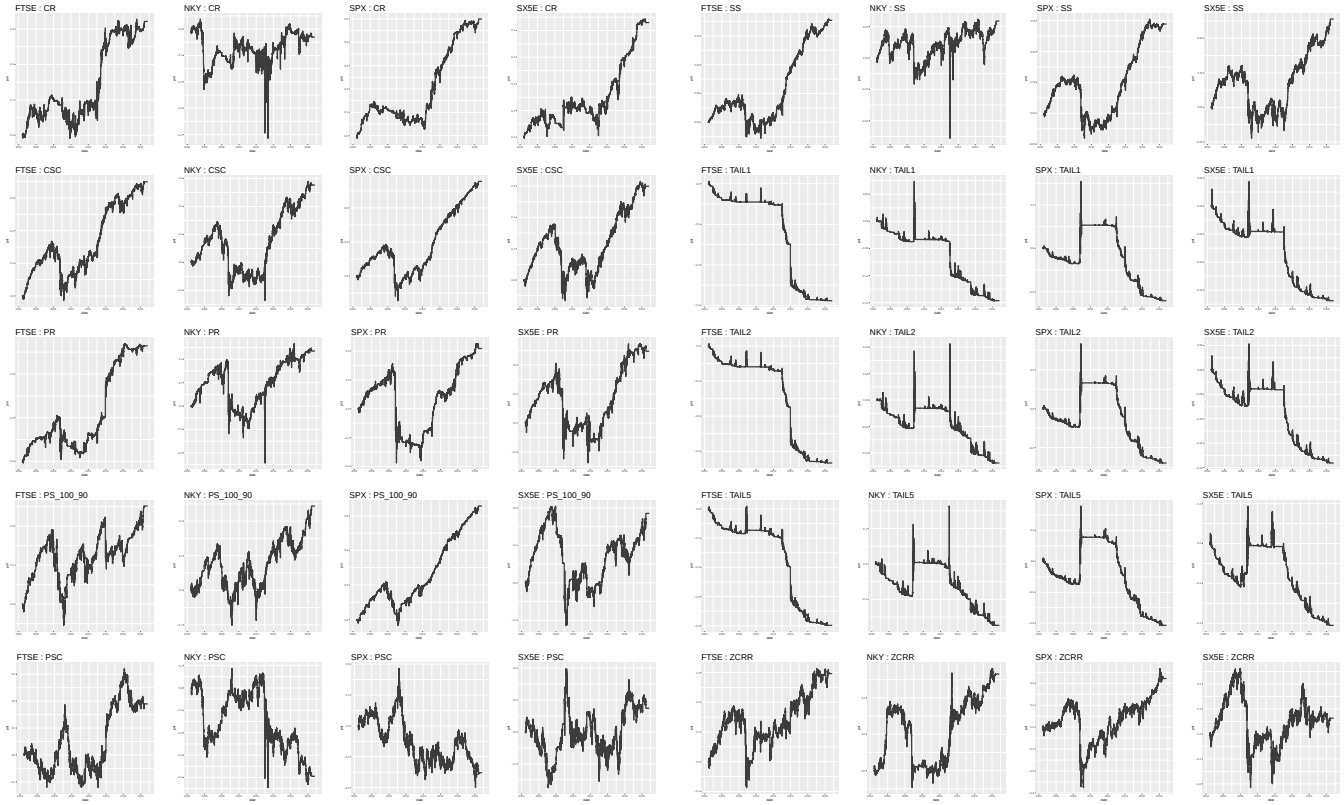
5 All backtests

6 Strategies vs market backtests

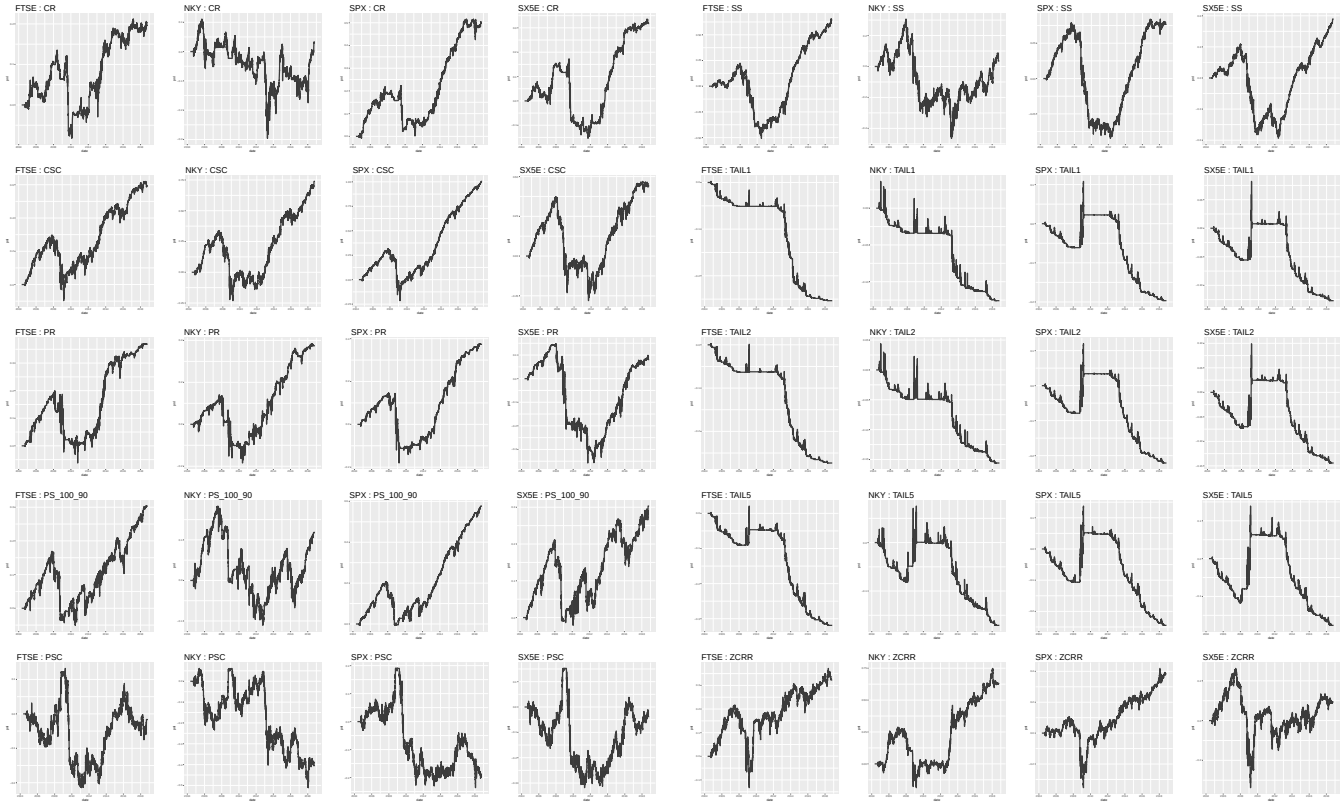
6.1 3m maturity



6.2 6m (Jun, Dec) maturities

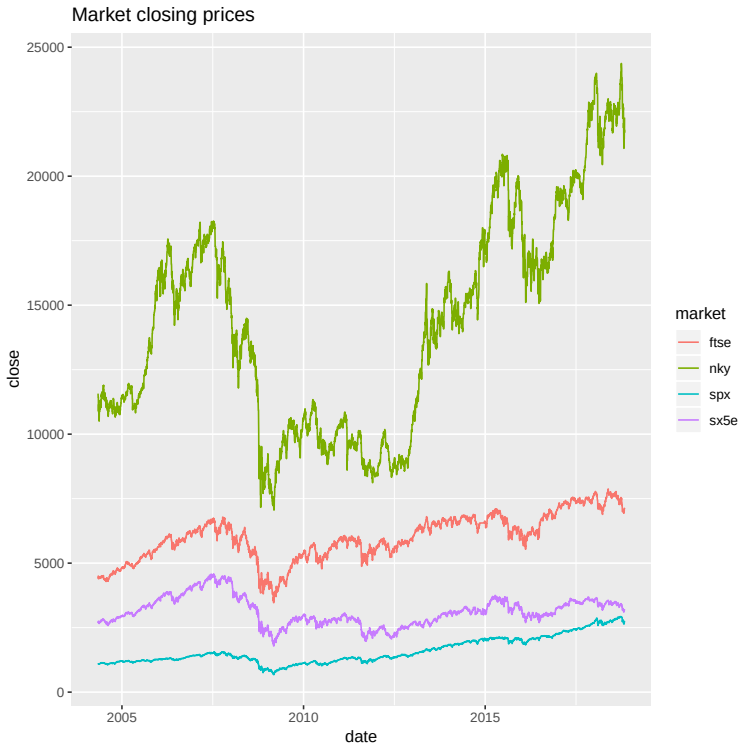


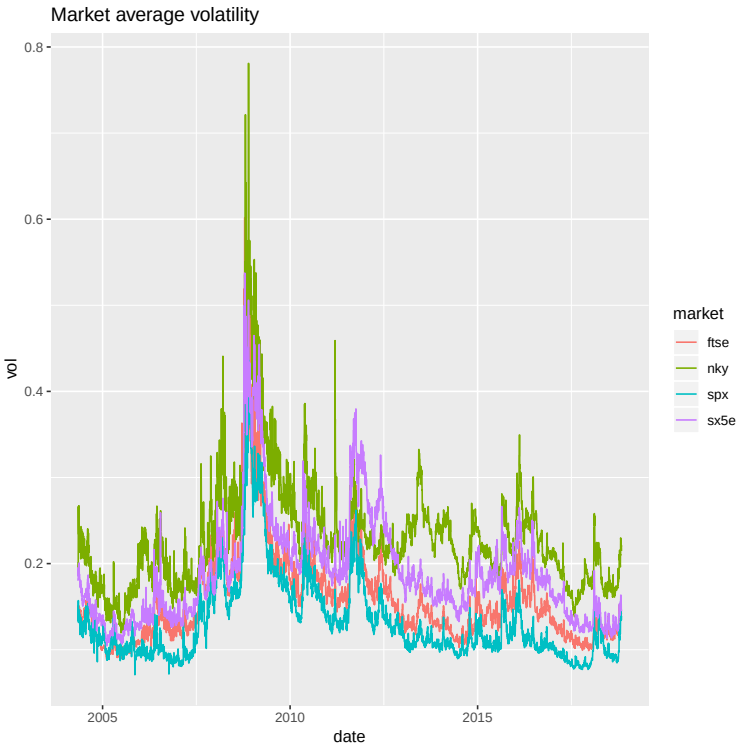
6.3 6m (Mar, Sep) maturities



7 Dataset

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





8 Backtest algorithm

8.1 backtest inputs

A backtest requires the following inputs:

1. volatility surface
2. schedule
3. payoff

8.2 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

8.3 shedule 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 shedule 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

8.4 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 shedule. The roll function is called only on roll period start days

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

8.5 backtest workflow

- start with shedule
- (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 appart, 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) compuite 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 biliear interpolation. This is done on the whole `data.table` in one go since the biliear interpolation formula is vectorizeable.
- (vectorized) call valuation model, store reval. 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 vectorizeable. 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.

8.6 volatility interpolation

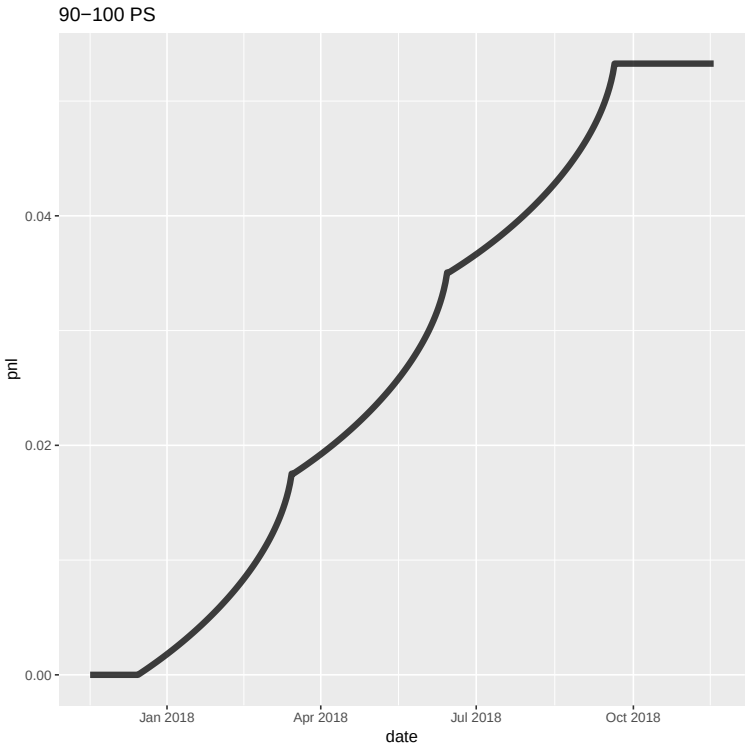
The `interpolate_vol` function takes a reval date, a strike and a maturity as inputs and performs volatility intepolation on a single-market volatility surface. This calculation needs to be performed for every day, every market and every strike in the strategy. This means that the number of interpolations 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 intepolation over multiple markets (for example from a multi-market shedule), 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.

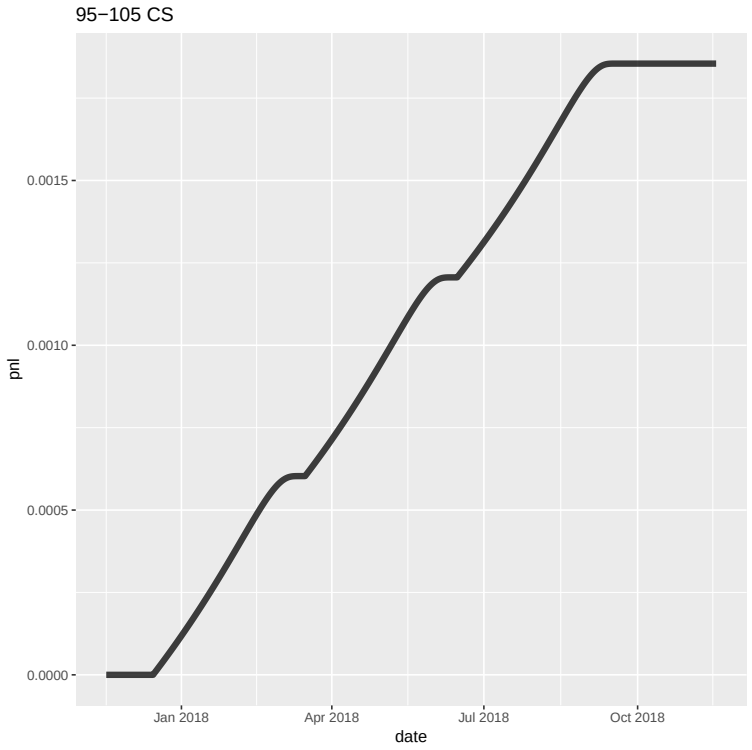
```
interpolate_vol<-function(date,strike,days,vsurf)
{

  # 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)
  ))

}
```

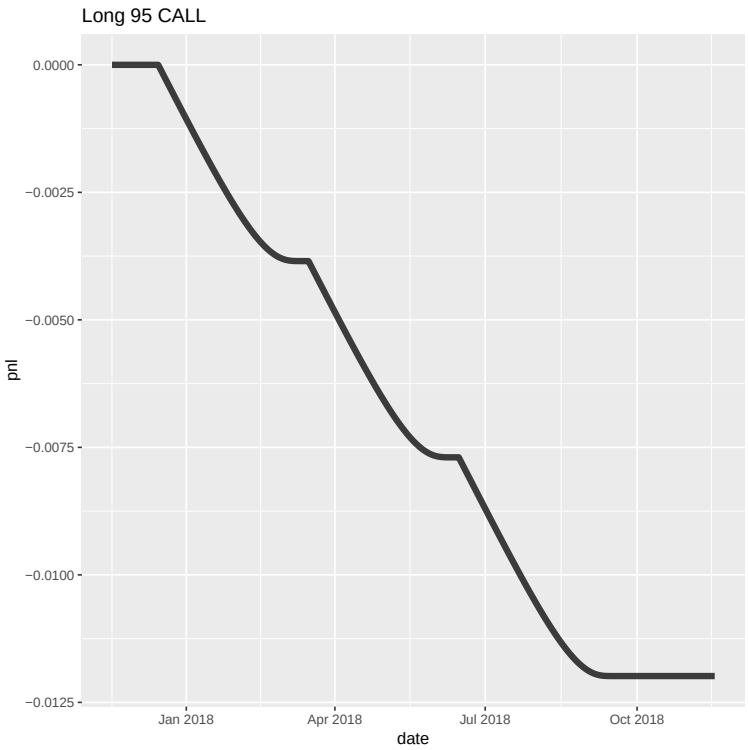
8.7 Unit tests

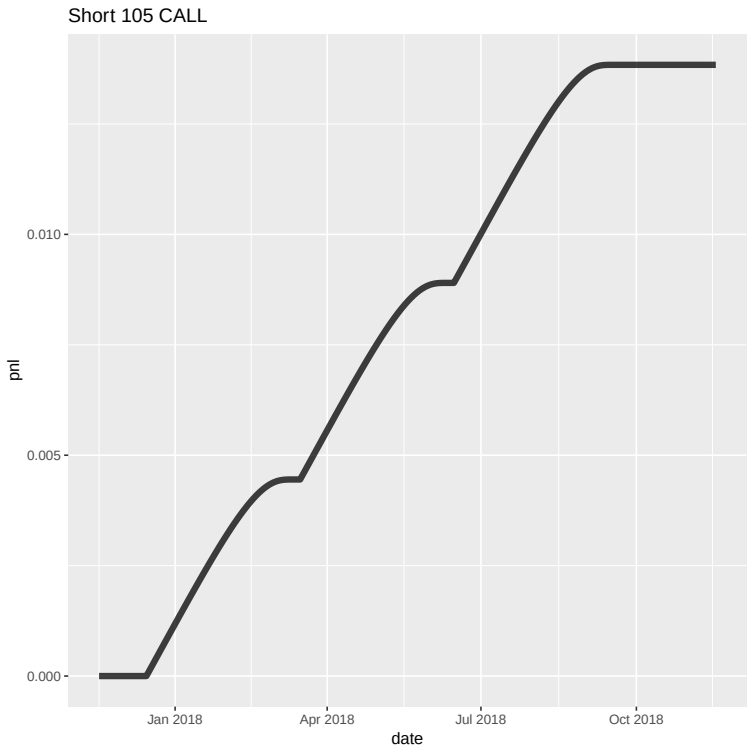
We perform a number of backtests for selected strategies on synthetic data to ensure the backtest engine is working as expected.

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









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