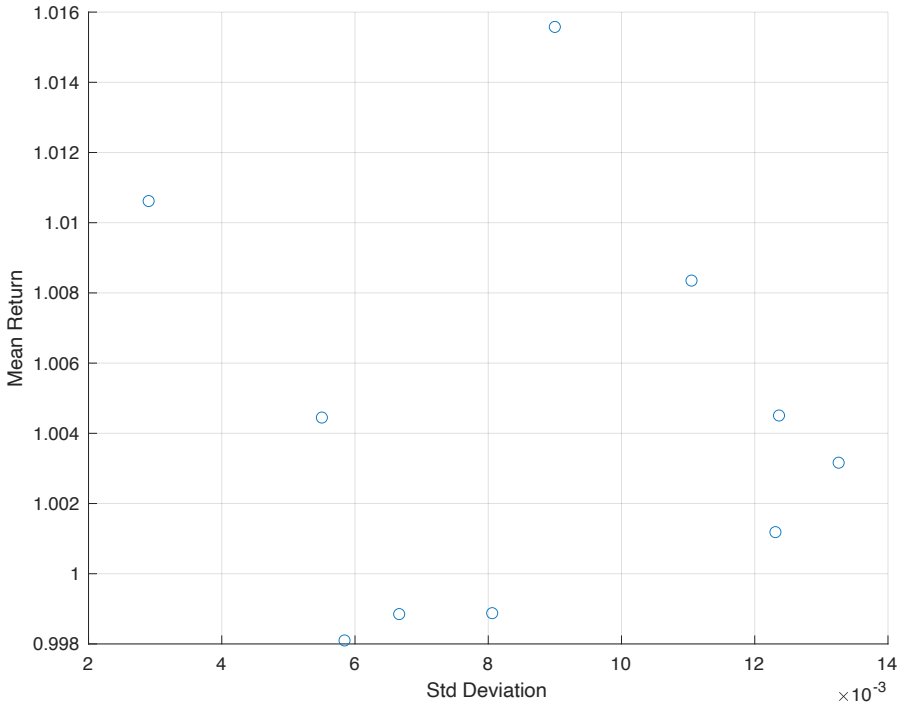
EDA

SPCA optimisation

Efficient Frontier (clearly not efficient)



Specs

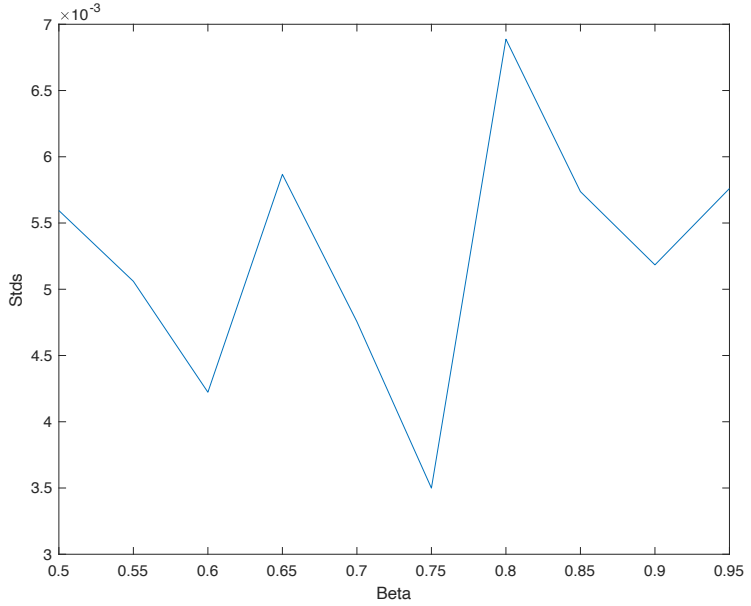
- nsims = 3
- No. of sparse PCs: k=2
- warmup = 100;
- stop = floor(0.5 * simObj.d + (lambda/20)*0.25*simObj.d);

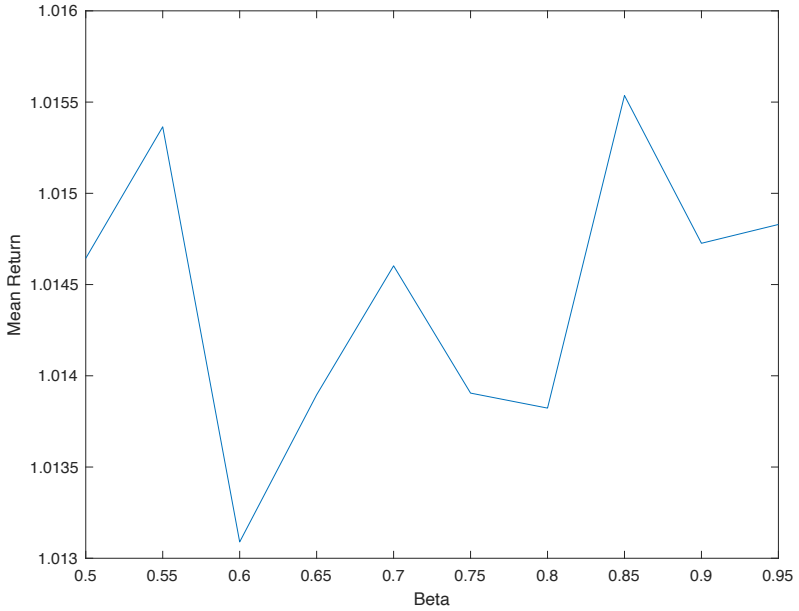
I suspect quadprog is not returning the correct solution.

cVar optimisation: Varying beta

	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Std	0.004338	0.0050674	0.004428	0.0041905	0.0036728	0.0038403	0.0045175	0.0039433	0.0042215	0.0035758
Mean Return	1.0066	1.0059	1.0067	1.0051	1.0078	1.0072	1.0062	1.0064	1.0081	1.0061

- (20 assets)
- Conjecture: higher beta – lower stds, lower mean return
- Doesn't seem to be the case

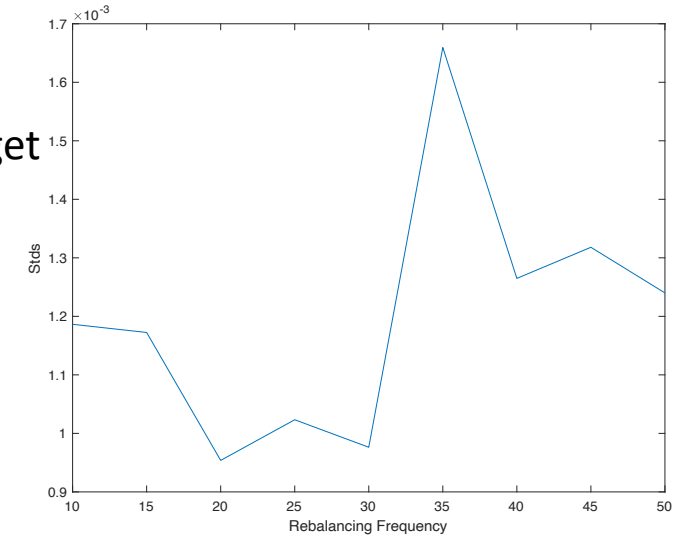


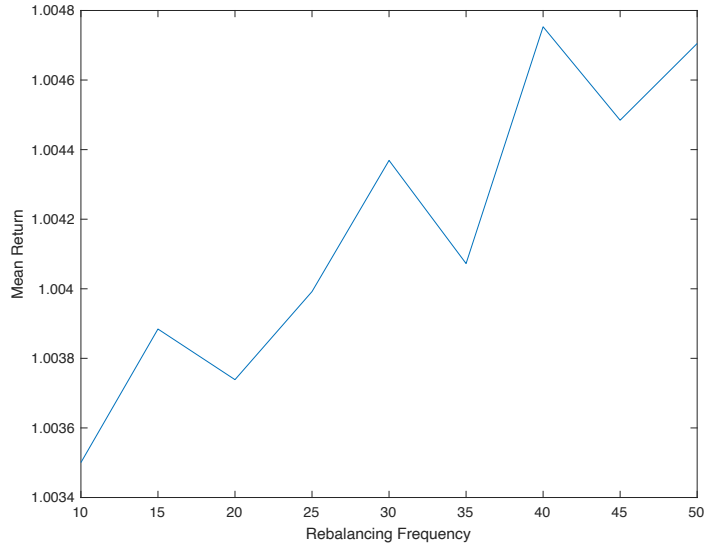


cVar optimisation: Changing rebalancing frequency

	10	15	20	25	30	35	40	45	50
Std	0.0011864	0.0011724	0.00095384	0.0010233	0.0009763	0.0016594	0.0012648	0.001318	0.00124
Mean Return	1.0035	1.0039	1.0037	1.004	1.0044	1.0041	1.0048	1.0045	1.0047

- Idea
  - Longer period between rebalancing - get killed by transaction costs – lower return
  - Longer period between rebalancing - higher chance of unexpected losses in between – higher std
- Specs
  - eta = 0.0002
  - Nsims = 30
  - T = 250, d = 50

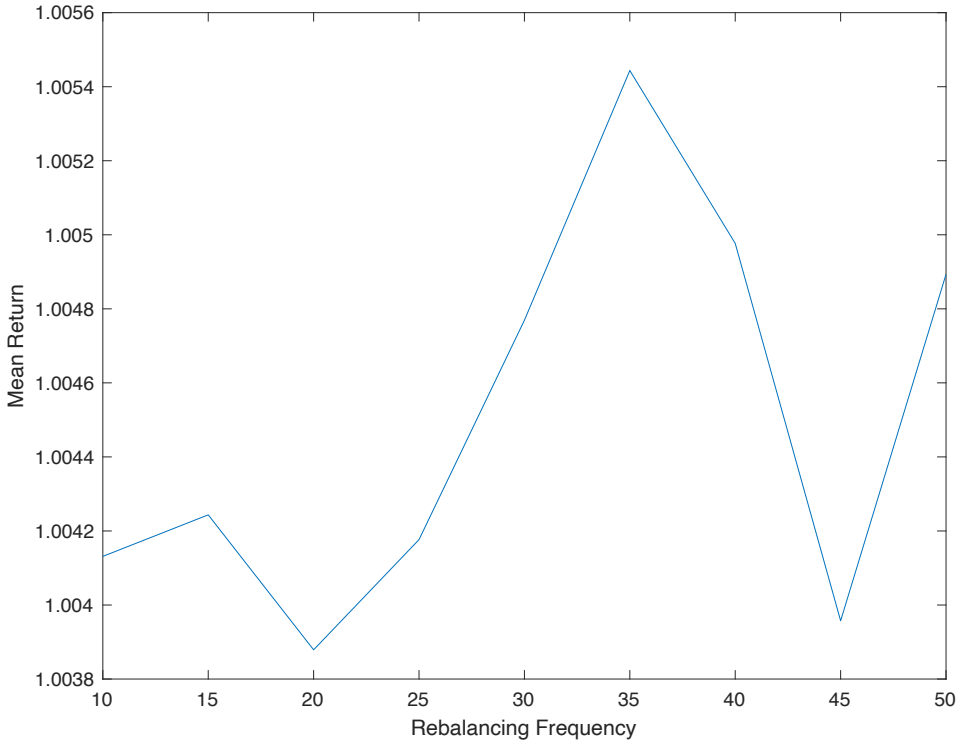


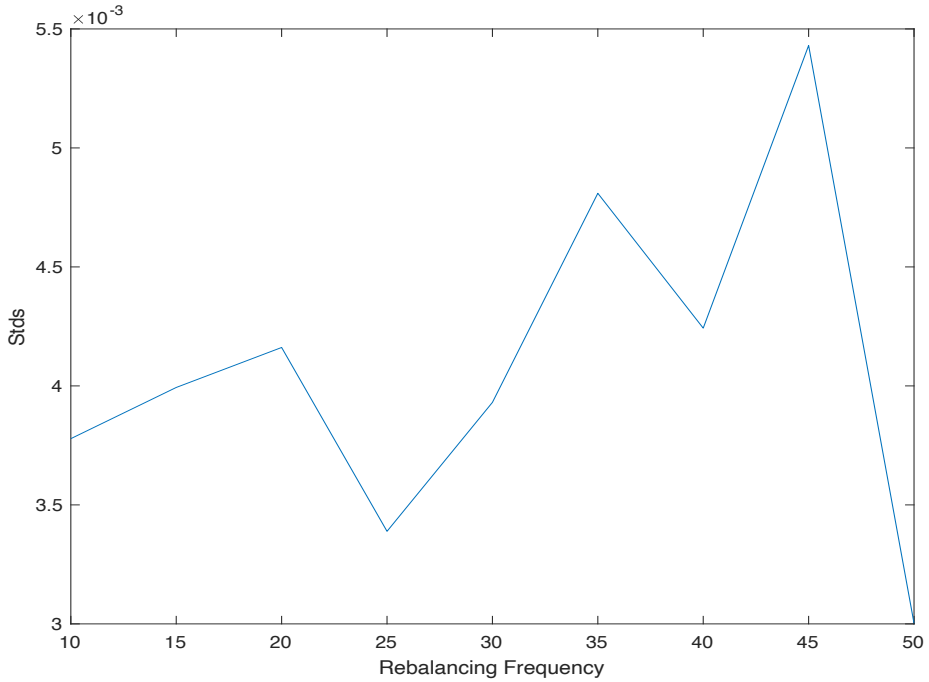


Mean return: as expected

Stds: looks strange. ??

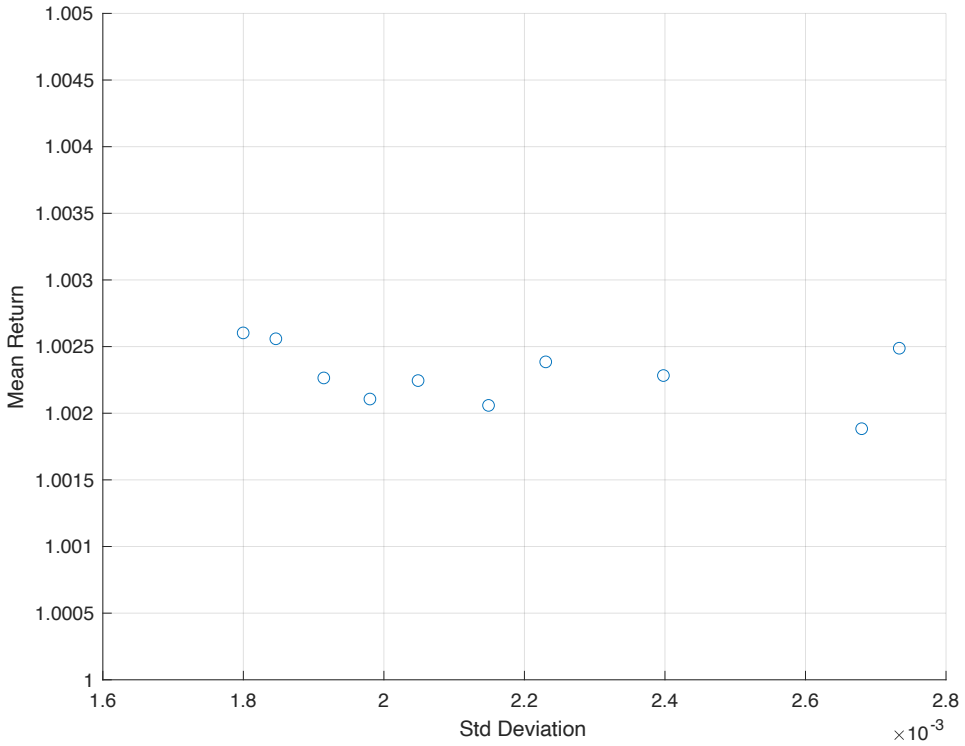
Try again: cVaR rebalancing frequency but increase to nsims = 50





Now makes sense. Longer period between rebalancing → higher mean return but higher stds

Efficient Frontier for cVaR optimisation



rebalancing_periods = floor(0.2 * simObj.T - (lambda-0.5)*0.15*simObj.T/4.5)

Starting to look like a proper efficient frontier?