Package PBO: Probability of Backtest Overfitting

Matthew R Barry, PhD, CAIA
Slipstream Advisors LLC

R/Finance - 2014 - Chicago

Motivation

Financial Charlatanism

"If the researcher tries a large enough number of strategy configurations, a backtest can always be fit to any desired performance for a fixed sample length. Thus, there is a minimum backtest length that should be required for a given number of trials."

Bailey et al., "Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance," Notices of the American Mathematical Society, May 2014. http://ssrn.com/abstract=2308659.

Overfitting Prevention Tools

- Compute the probability of backtest overfitting
- Compute the performance Degradation and probability of loss
- Apply the theory of stochastic dominance to rank gambles or lotteries without individual's utility function
- Apply model sequestration

Bailey and López de Prado, "How to Spot Backtest Overfitting"

Overfitting Prevention Tools

- Compute the probability of backtest overfitting **Package PBO**
- Compute the performance Degradation and probability of loss
- Apply the theory of stochastic dominance to rank gambles or lotteries without individual's utility function

- Apply model sequestration

Backtest Overfitting

- Let n^* indicate the strategy number that performed optimally in-sample by some performance measure R
- The performance of that strategy number out-of-sample is $\overline{R_{n^*}}$
- A backtest is overfit if the expectation of $\overline{R_{n^*}}$ is less than the median of all out-of-sample performance measures $\overline{R}$
- PBO ≡ Prob $[\overline{R_{n^*}} < Me[\overline{R}]]$

Performance Degradation

- Performance degradation indicates the negative relationship between performance in-sample and performance out-of-sample
- We fit a linear regression to the performance pairs (R_n^*, $\overline{R_n^*}$)
- Fitting to the out-of-sample response the slope will usually be negative
- The probability of loss we compute as the proportion of pairs with negative performance

Stochastic Dominance

- Determine whether the distribution of $\overline{R_n^*}$ across all cases stochastically dominates over the distribution of all $\overline{R}$; if not, this is more evidence of poor out-of-sample results
- F.O.D.: $Prob[\overline{R_n^*} \geq x] \geq Prob[\overline{R} \geq x]$ for all x, and for some x $Prob[\overline{R_n^*} \geq x] > Prob[\overline{R} \geq x]$
- S.O.D.: $\int_{-\infty}^{X} (Prob[\overline{R} \leq x] - Prob[\overline{R_n^*} \leq x])\, dx \geq 0$ for all x, and > 0 for some x

CSCV Algorithm

For each combination $c \in C_s$ the CSCV procedure applies as follows:

1. Form the training set $\mathbf{J}$ by joining the $S/2$ submatricies identified by c.
2. Form the testing set $\mathbf{J}$ as the complement of $\mathbf{J}$ in $\mathbf{M}$.
3. Form the vector R of performance statistics of order N across the training set $\mathbf{J}$.
4. Form the vector $\overline{R}$ of performance statistics of order N across the testing test $\overline{\mathbf{J}}$.
5. Identify the element $n^* = \arg\max_n \{R_n\}$.
6. Determine the relative rank $\overline{\omega_c}$ of $\overline{R_{n^*}}$ within $\overline{R}$.
7. Compute the logit $\lambda_c = \ln(\overline{\omega_c}/(1 - \overline{\omega_c}))$.

R Implementation

CSCV Selections

```
pbo.R:
. . .

pbo <- function(M,S=4,F=NA,threshold=0,inf_sub=6,allow_parallel=FALSE) {

  stopifnot(is.function(F))
  require(utils,quietly=TRUE)

  T <- nrow(M)                  # samples per study
  N <- ncol(M)                  # studies
  CS <- combn(S,S/2)            # combinations
  SN <- T / S                   # partition size
```

combn()

PBO Steps

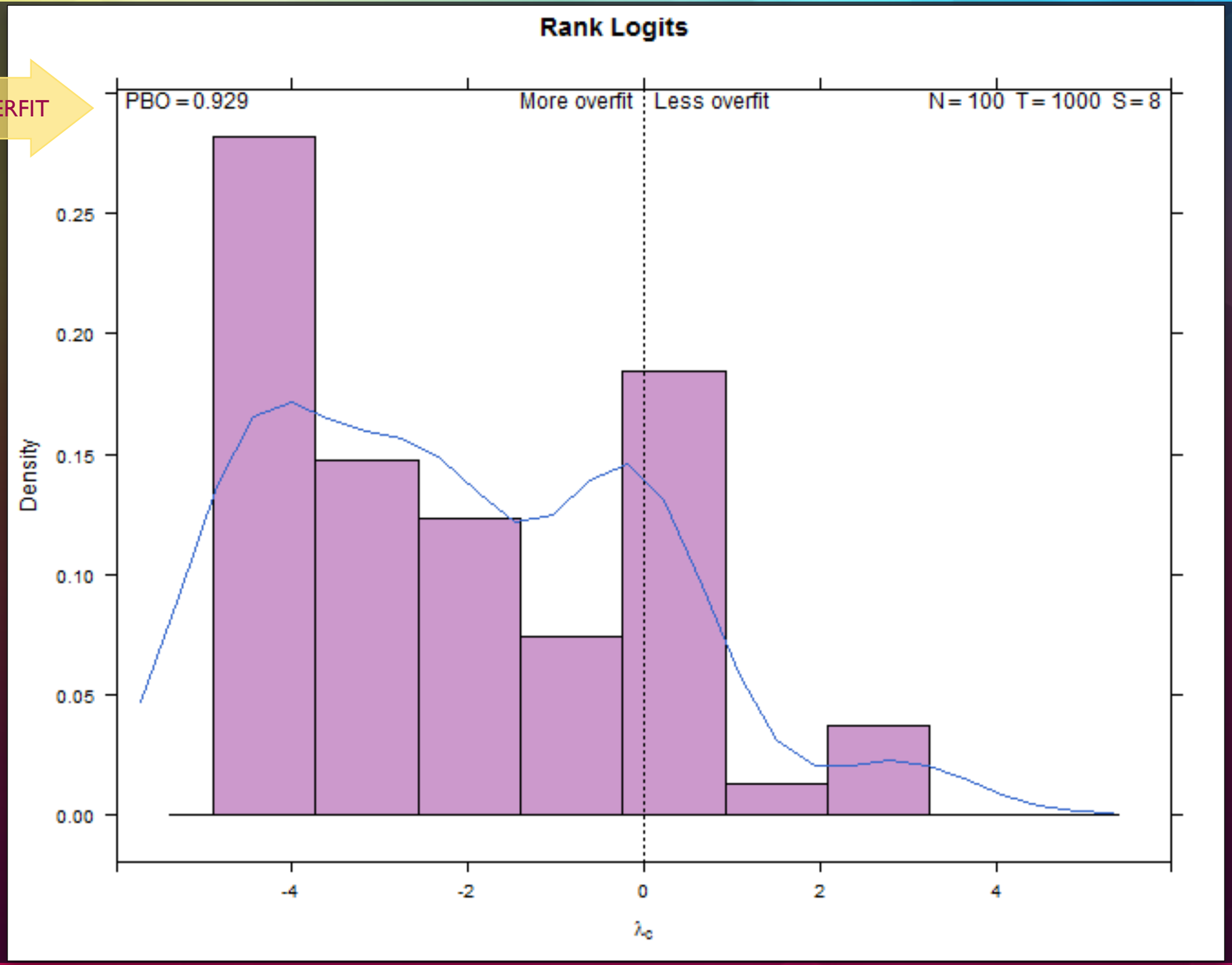
```
pbo.R:
. . .

 # training and test sets (in sample, out of sample)
 J <- M[is_indices,]
 J_bar <- M[os_indices,]

 # compute performance over the N strategies in each subset
 # could use for R any summary statistic e.g. SharpeRatio or Omega
 R <- mapply(F,J)
 R_bar <- mapply(F,J_bar)

 # compute n* by argmax over R vector
 n_star <- which.max(R)
 n_max_oos <- which.max(R_bar)

 # rank of n*th result from OOS performance; converted to (0,1) interval
 os_rank <- rank(R_bar)[n_star]
 omega_bar_c <- os_rank / length(R_bar)

 # logit
 lambda_c <- log(omega_bar_c / (1 - omega_bar_c))
```

S3 Class and Dispatch

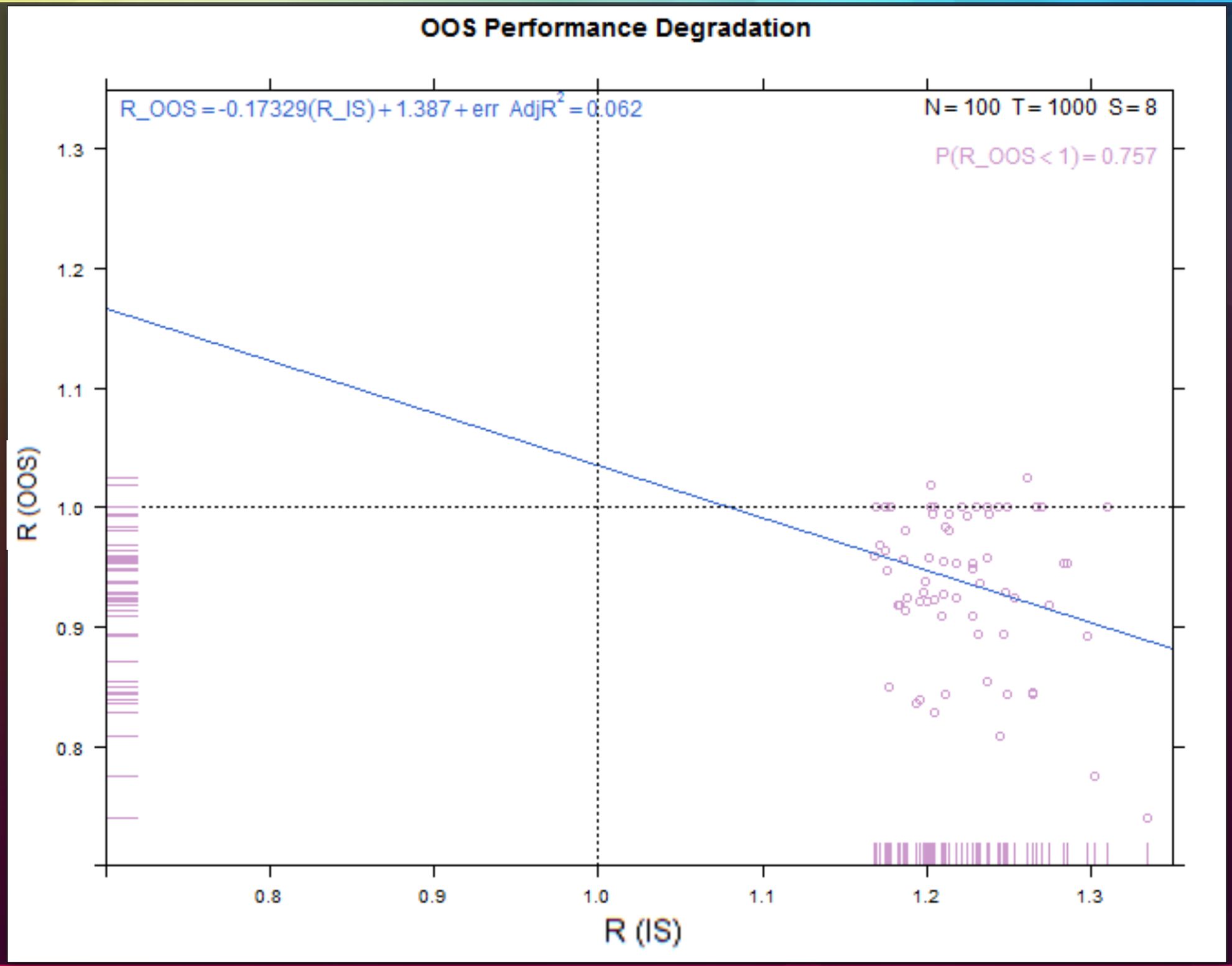
```
pbo.R:
. . .
rv = list(results=cs_results, combos=CS, lambda=lambda, phi=phi, rn_pairs=rn_pairs,
func=as.character(substitute(F)), slope=m, intercept=b, ar2=ar2, threshold=threshold,
below_threshold=p_oos_bt, test_config=test_config,inf_sub=inf_sub)

class(rv) <- "pbo"
rv
}
```

```
lattice.pbo.R:

# dotplot.pbo()
# histogram.pbo()
# xyplot.pbo()

> require(lattice)
> my_pbo = pbo(. . .)
> histogram(my_pbo)
> dotplot(my_pbo)
```

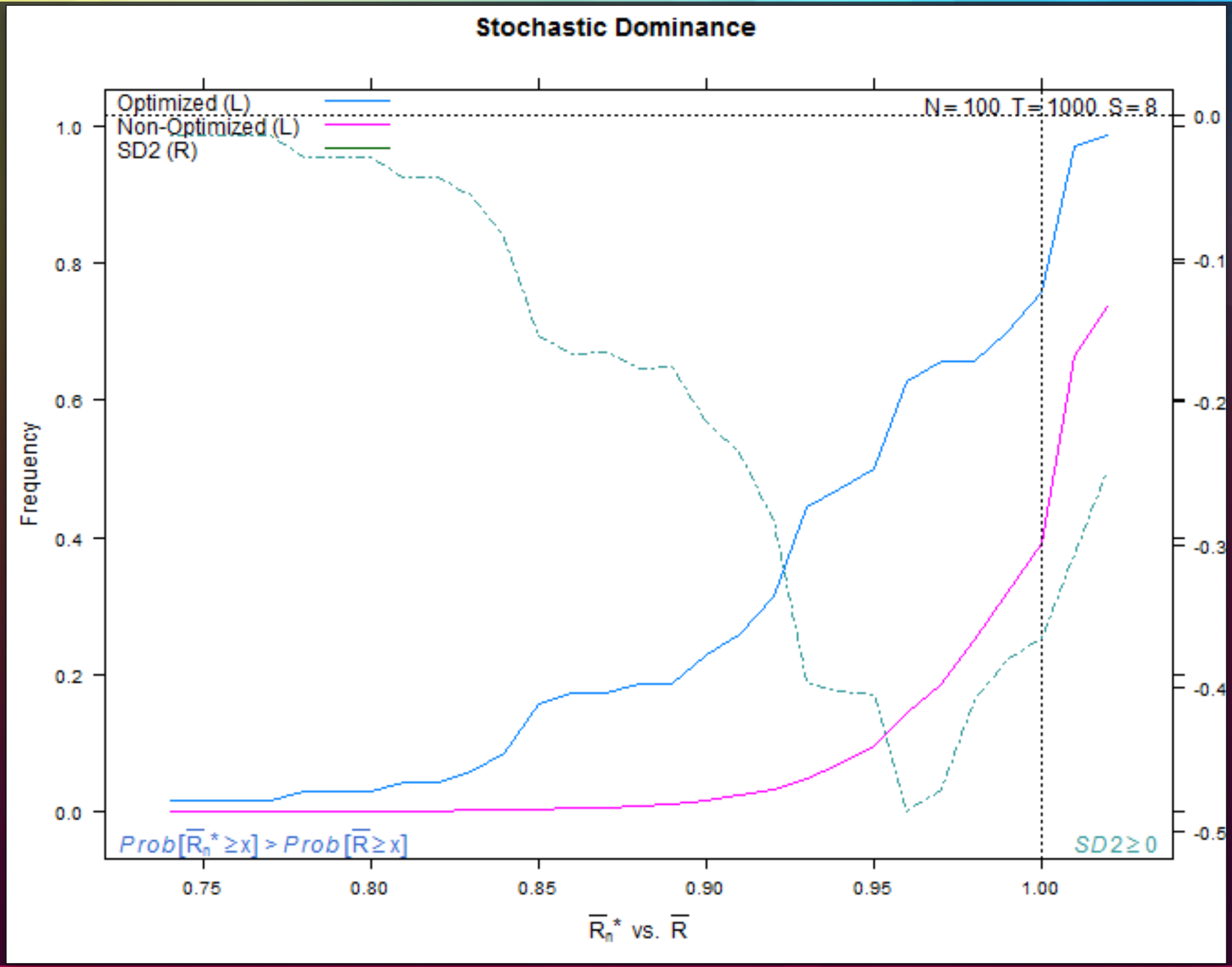
OVERFIT



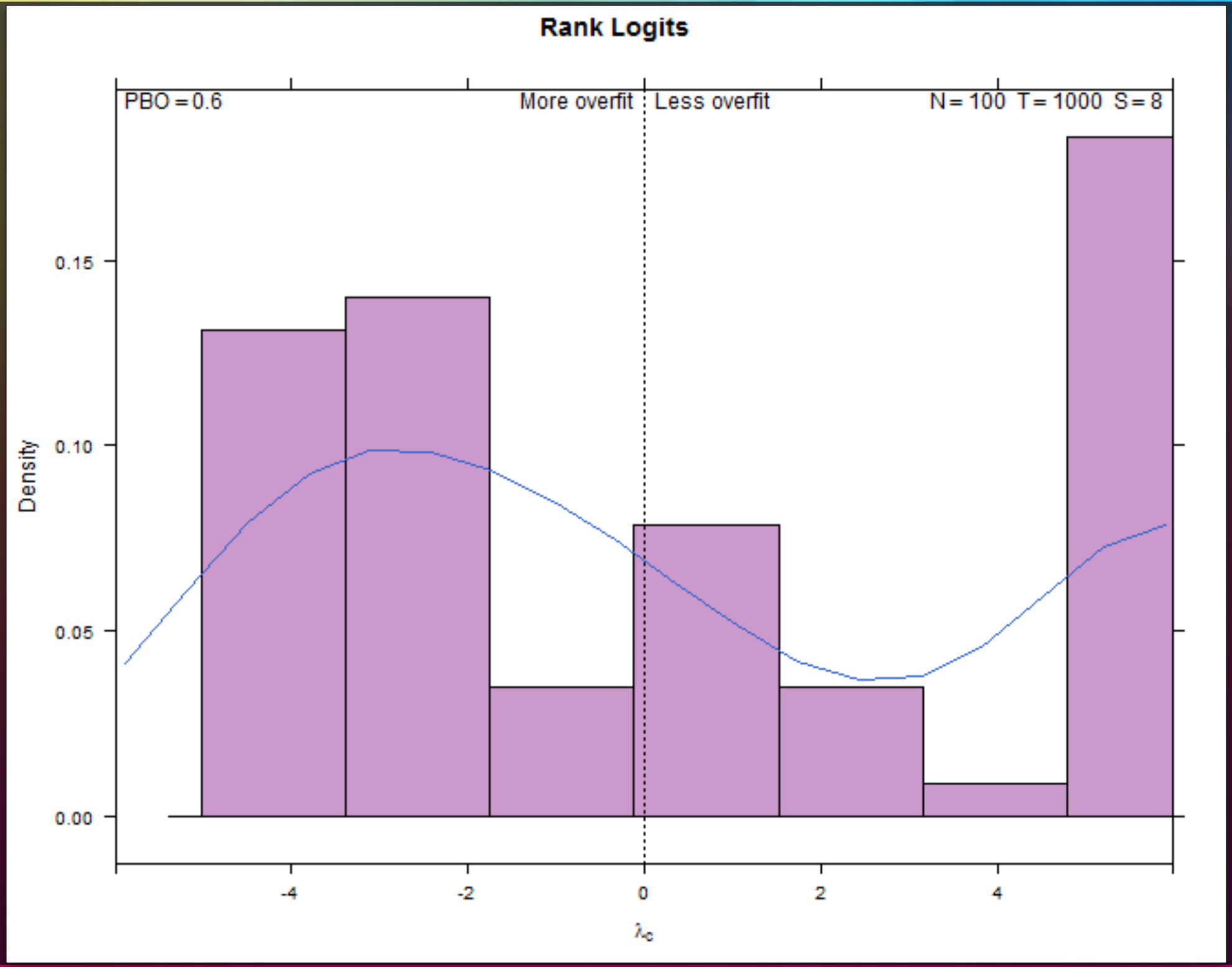
EXAMPLE OF STRONG BACKTEST OVERFITTING



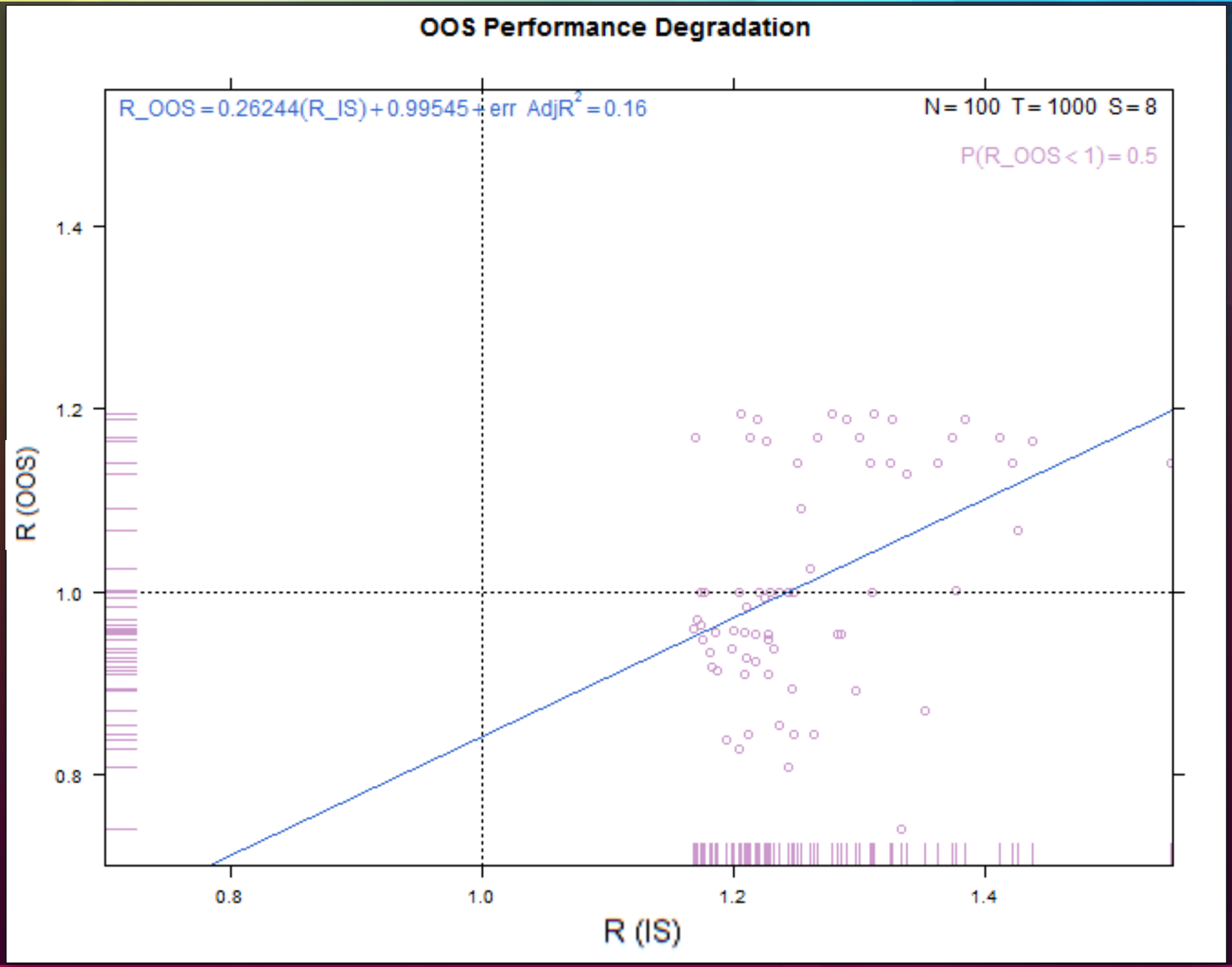
Example of Strong Backtest Overfitting - Omega Ratio



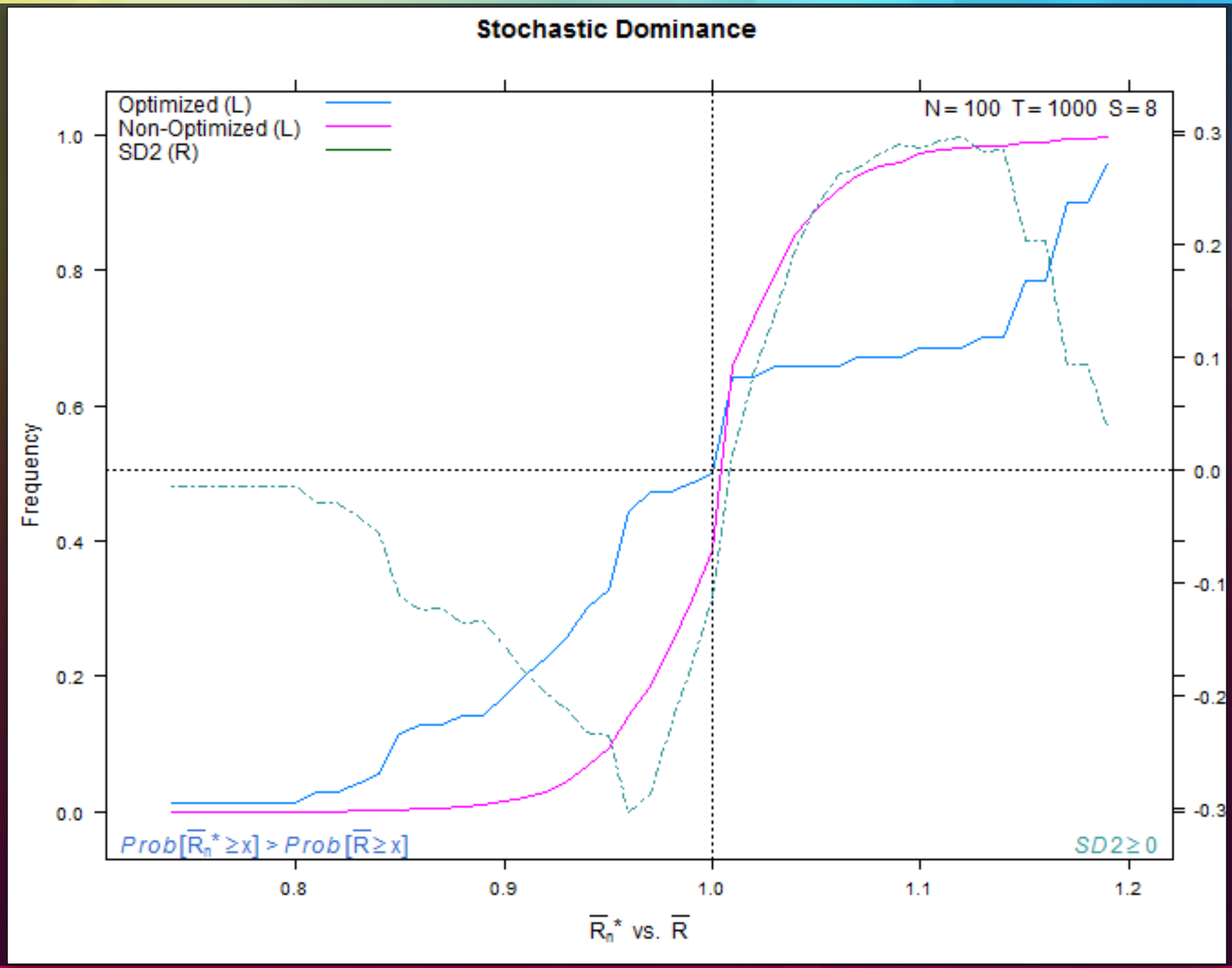
Example of Strong Backtest Overfitting



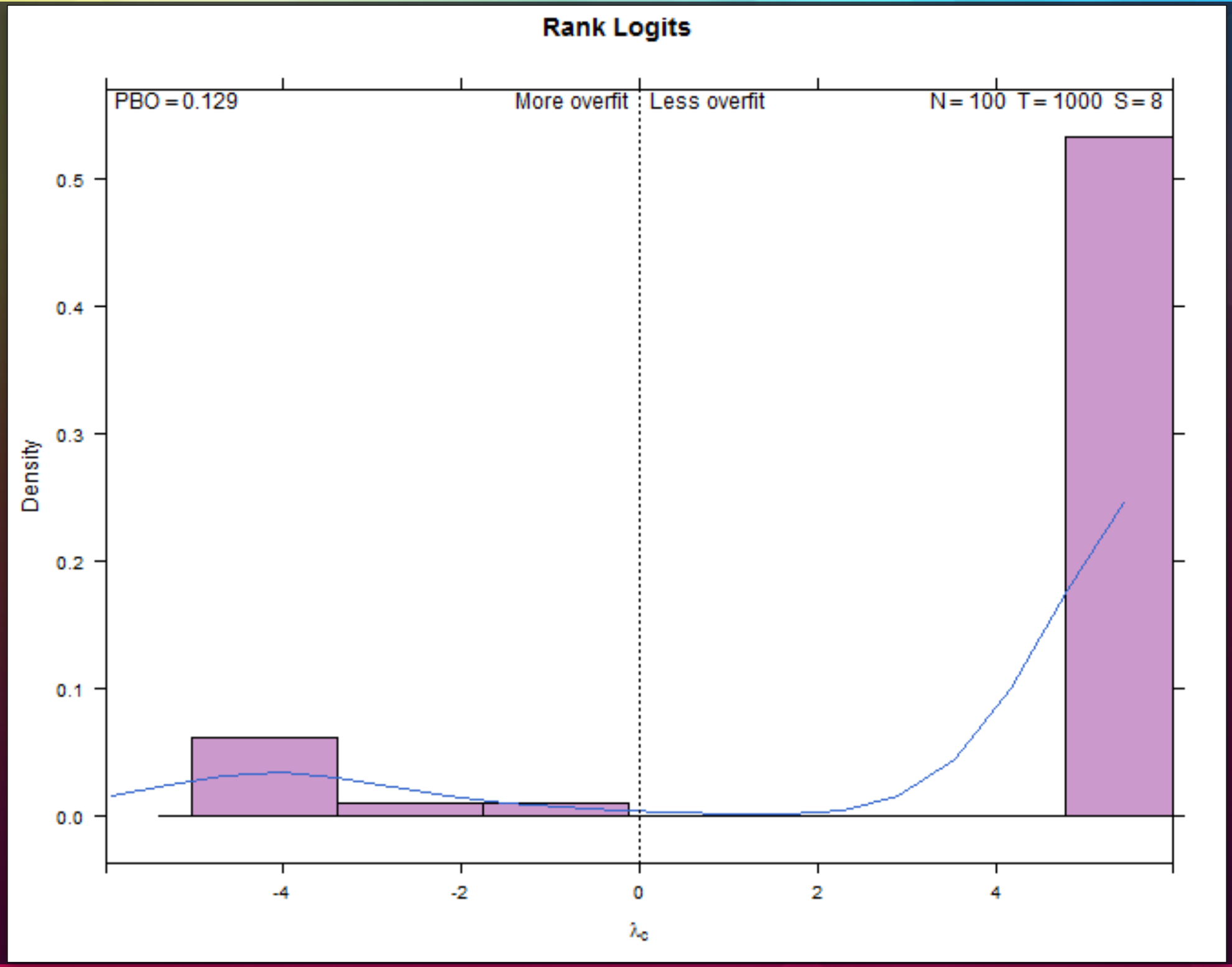
Example of Moderate Backtest Overfitting



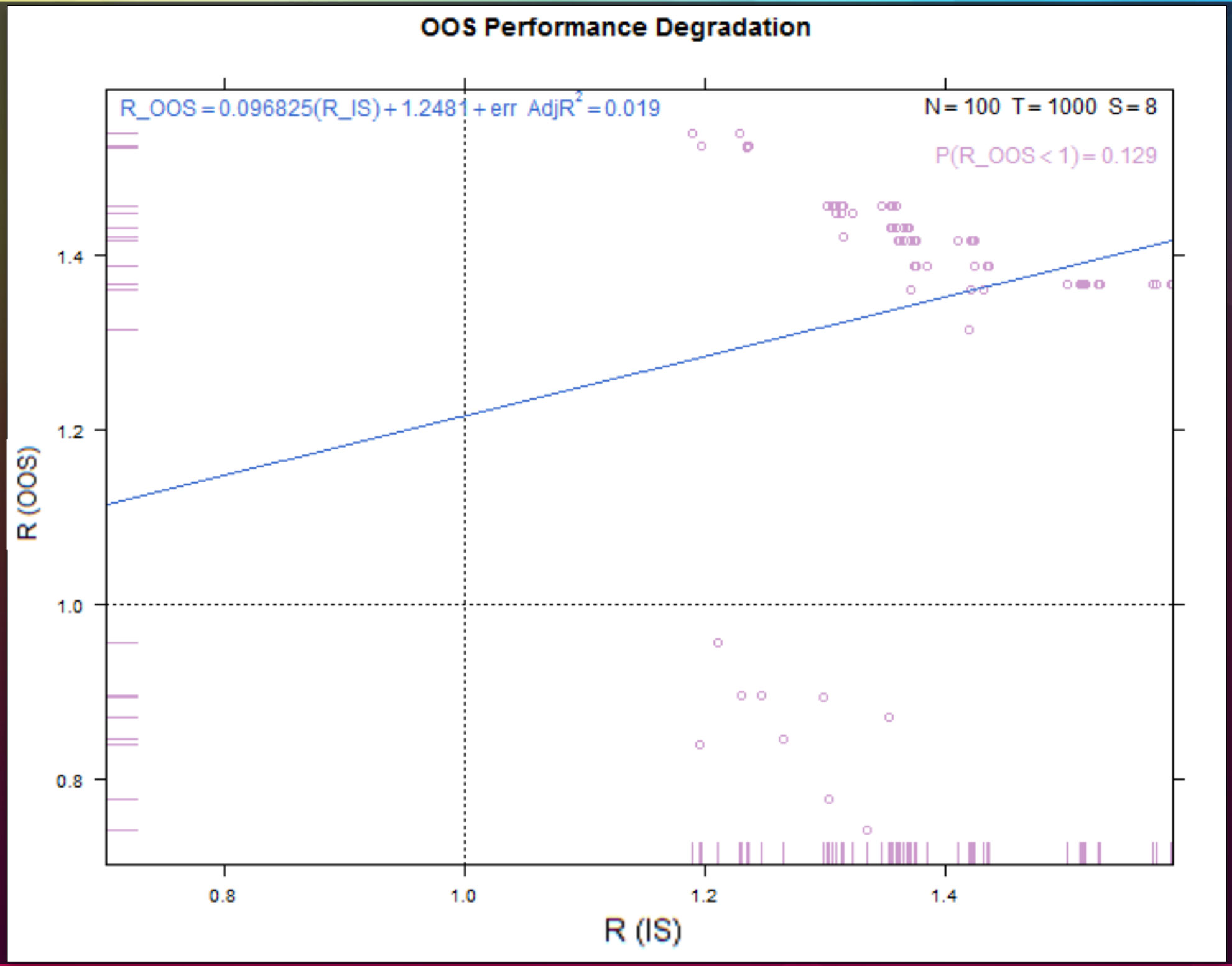
Example of Moderate Backtest Overfitting - Omega Ratio



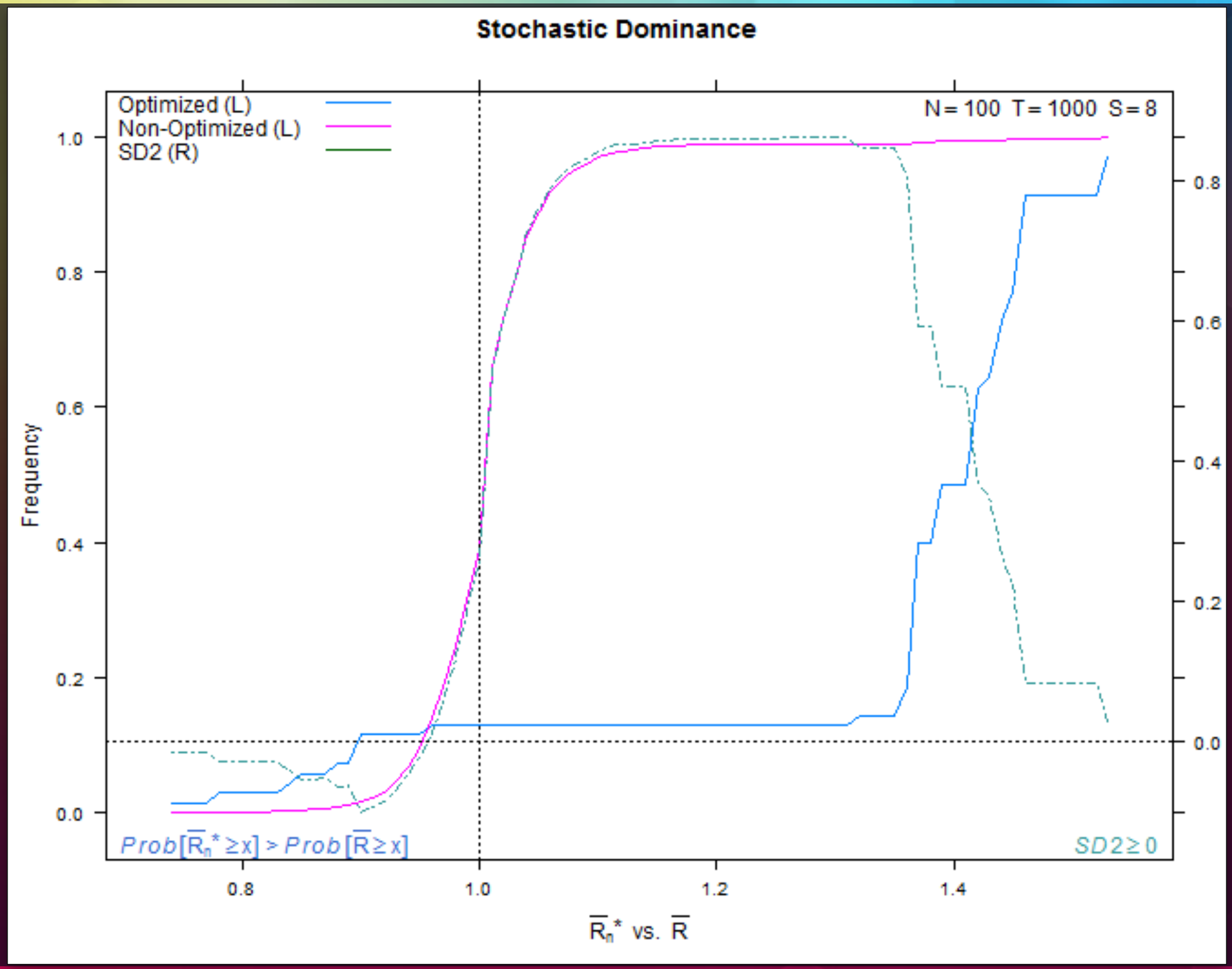
Example of Moderate Backtest Overfitting



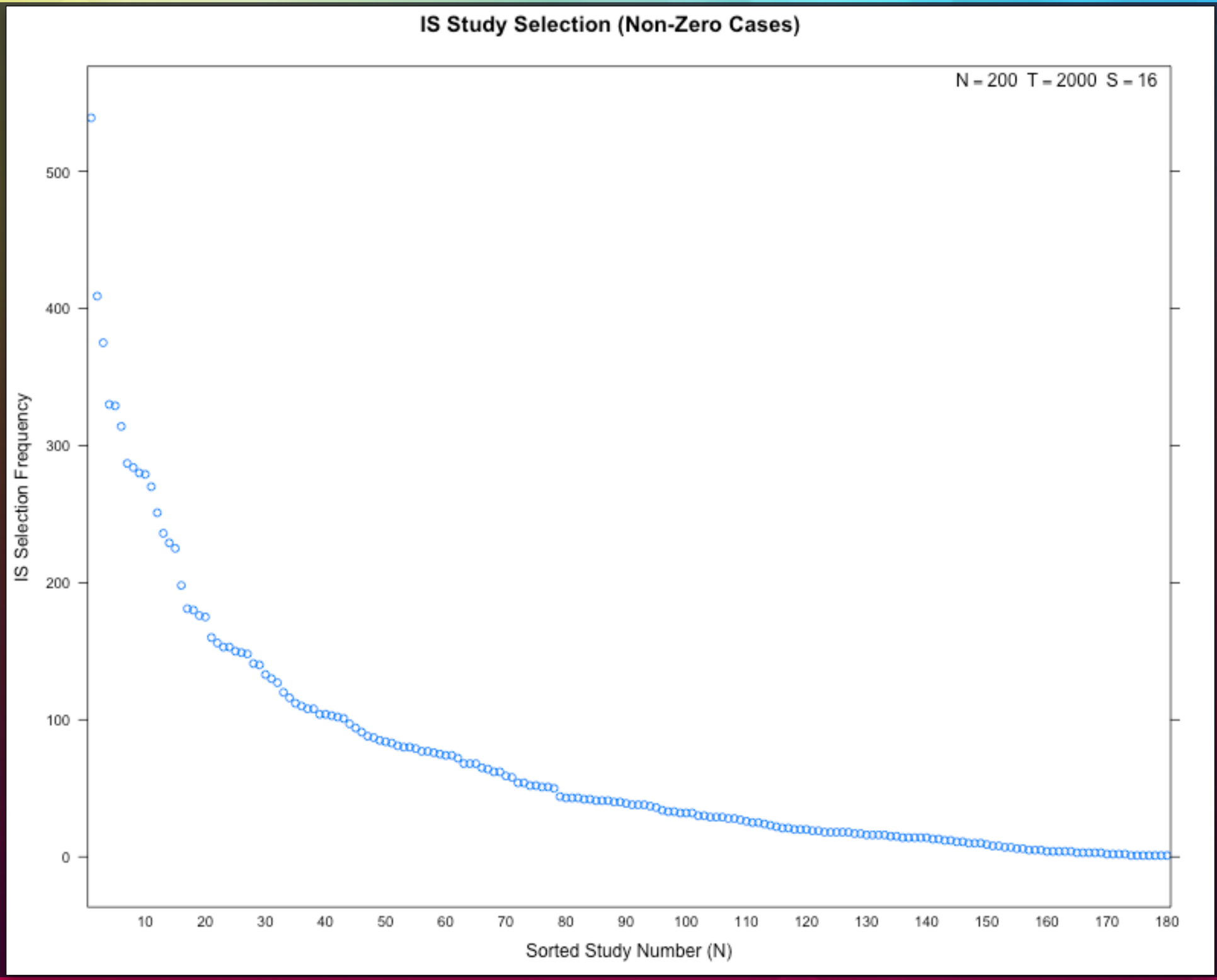
Example of Small Backtest Overfitting



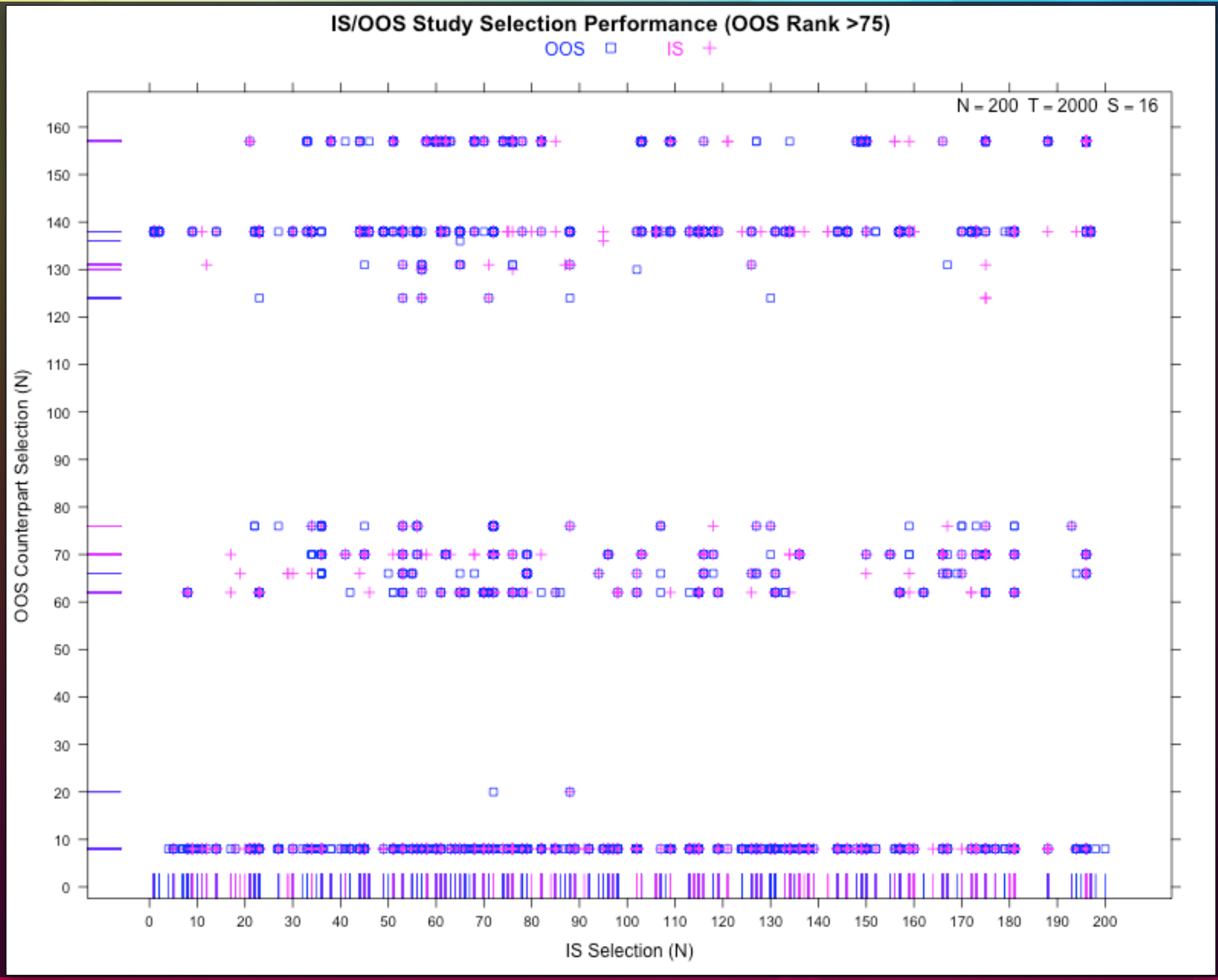
Example of Small Backtest Overfitting - Omega Ratio Measure



Example of Small Backtest Overfitting



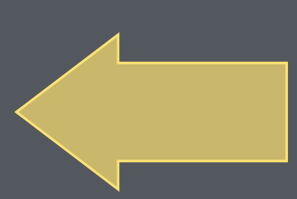
Study Selection Frequency



Study Selection Pairs

Parallel Computation Support

```
pbo.R:
. . .

cs_results <- NULL
if ( allow_parallel ) {
    require(foreach,quietly=TRUE)
    cs_results <- foreach ( cs=1:ncol(CS),
                .combine=rbind,
                .multicombine=TRUE) %dopar%
      cs_compute(cs)
}
```



Parallel Benchmark

```
> require(rbenchmark) # 8-core MacBook Pro 2.6 GHz Intel Core i7, 8 GB RAM

> benchmark(
pbo(M,S,F=Omega,threshold=1,allow_parallel=FALSE),
pbo(M,S,F=Omega,threshold=1,allow_parallel=TRUE),
replications=1)

                                                      test replications  elapsed relative

1 pbo(M, S, F = Omega, threshold = 1, allow_parallel = FALSE)            1 2091.530     6.85

2 pbo(M, S, F = Omega, threshold = 1, allow_parallel = TRUE)             1  305.342     1.00

  user.self sys.self user.child sys.child

1  1963.484  125.885          0         0

2    19.184    2.793          0         0

>
```

Package Availability

```
> require(devtools)

> install_github('pbo',username='mrbcuda')
```

References

- Bailey and López de Prado, "How to Spot Backtest Overfitting," http://www.financial-math.org
- Bailey et al., "The Probability of backtest Overfitting," Feb 2014. http://ssrn.com/abstract=2326253
- Bailey et al., "Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance," Notices of the American Mathematical Society, May 2014. http://ssrn.com/abstract=2308659.

Session Info

```
> sessionInfo()

R version 3.0.3 (2014-03-06)
Platform: x86_64-apple-darwin10.8.0 (64-bit)

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] parallel  grid      stats     graphics  grDevices utils     datasets  methods
base

other attached packages:
 [1] rbenchmark_1.0.0           doParallel_1.0.8           iterators_1.0.7
 [4] foreach_1.4.2              latticeExtra_0.6-26        RColorBrewer_1.0-5
 [7] lattice_0.20-29            PerformanceAnalytics_1.1.0 xts_0.9-7
[10] zoo_1.7-11

loaded via a namespace (and not attached):
[1] codetools_0.2-8 compiler_3.0.3  tools_3.0.3
```