Prep Meeting 27

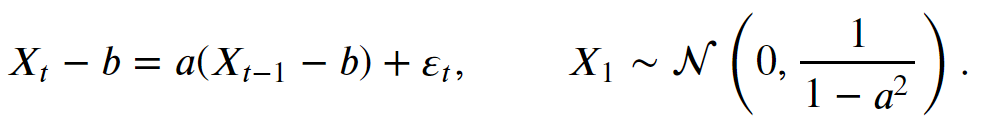
# SIOUX

Got a contract offer! Discuss this Wednesday, very excited.

# Using CV to Estimate AR(1) model with offset

## Problem Setting

Consider the AR(1) model with offset.

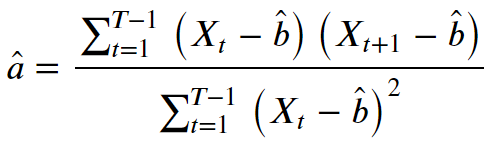


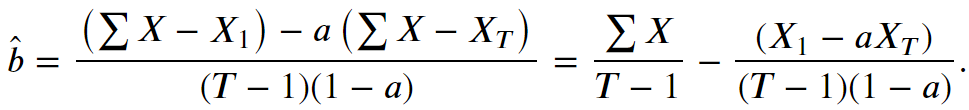
We want to use cross-validation to determine whether



## Cross-Validation Techniques

### Approach 0. Remove Nothing.

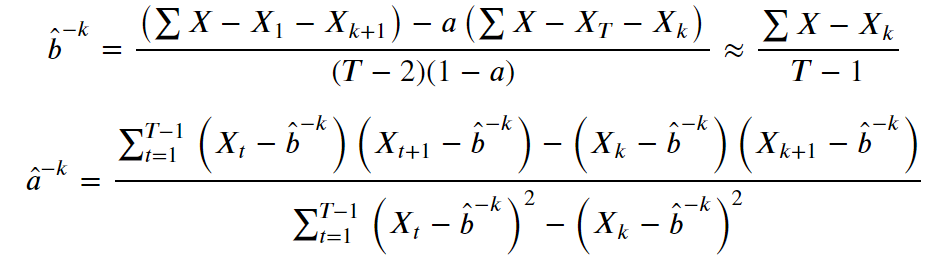




*No cross-validation, but just to show the MLE estimates*.

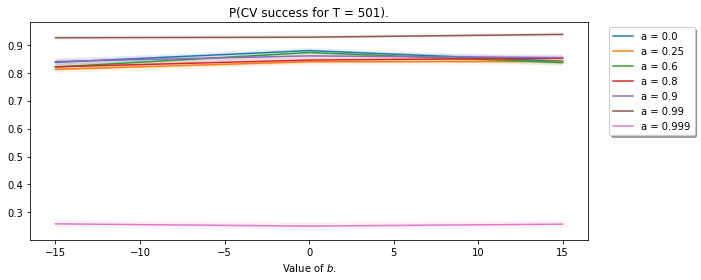
### Approach 1. Remove (Xk, Xk+1) pair.

**Exact / Approximate for b.**

****

Note that we require both one variable to estimate the other. A suitable approach could be to use the approximate value for *b* first, then estimate *a*, and then, using these two estimates, iteratively update *a* and *b* until we have reached convergence.

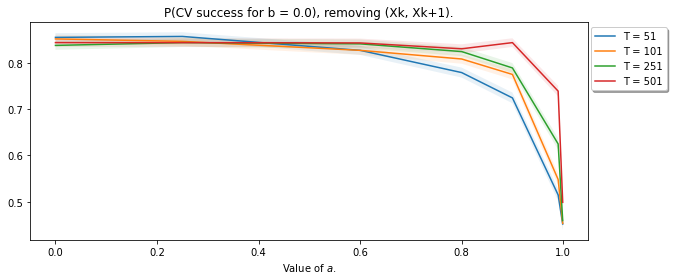
***Influence of b: Negligible.***

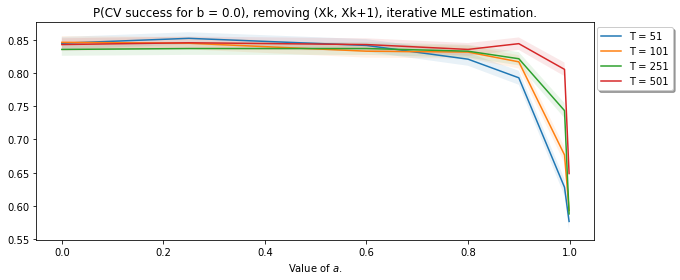


We see that the value of *b* has no influence whatsoever on the results. This makes sense, as both models use in fact the same estimation method for *b*. Note, however, that the estimation of *a* depends on *b*.

***As a function of a*: Decreases dramatically for *a* close to one*.***

We see that the probability of success remains constant for many values of T. Interestingly, the probability of success decreases to practically zero for *a* incredibly close to one.

****

****

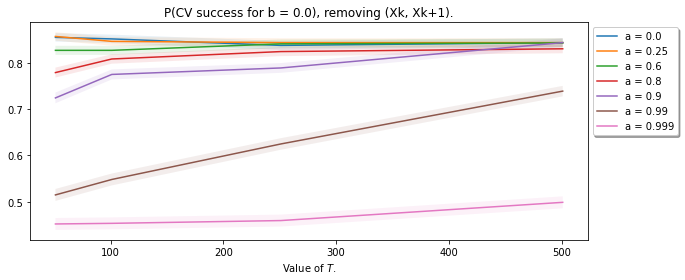
Furthermore, the estimation procedure for *a* and *b* seems a bit deviating for *a* close to one. As we need *b* to estimate *a*, and we need *a* to estimate *b*, we cannot compute this exactly. One method is to *iteratively* update *a* and *b* using our new estimates *a* and *b*, and hope they will converge. This seems to be working well for values of *a* not close to one, but gives problems for *a* close to one.

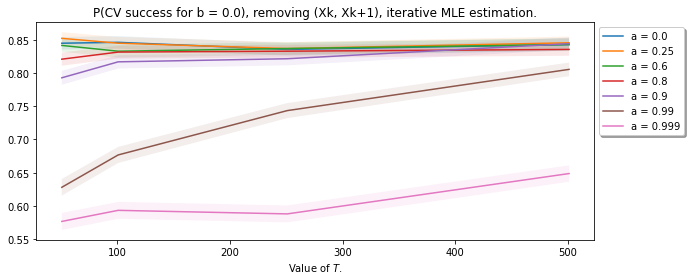
This gives rise to the question: are our one-step estimates for *a* and *b* incorrect for *a* close to one, so do we need to use this iterative procedure? I think so, as the initial estimate for *b* is a bit off.

The results differ quite a bit for values of *a* close to one depending on whether we re-estimate.

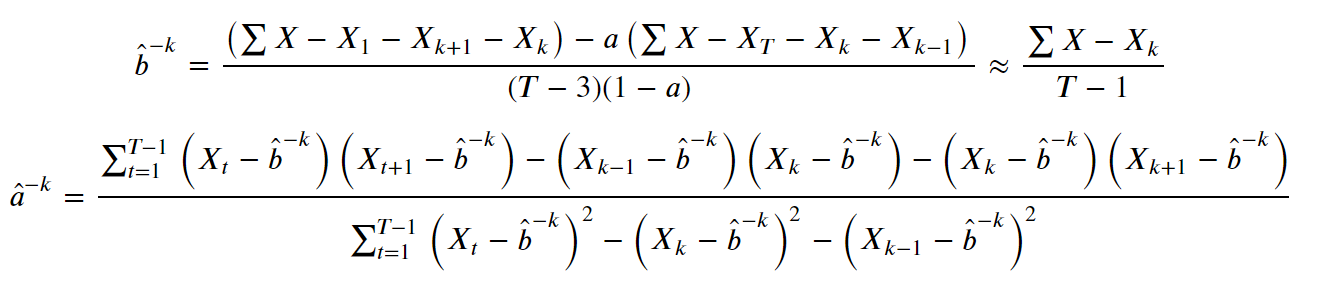
***As a function of T:* Remains constant for small *a*, grows for *a* close to one.**

We see that the probability of success remains constant for many values of *a*. Among interest is for the very large values of *a, a = 0.99* and *a = 0.999*. Interestingly, the probability of success increases as a function of *T* for *a = 0.99*. We will see that this is most likely due to the estimation of *a*.

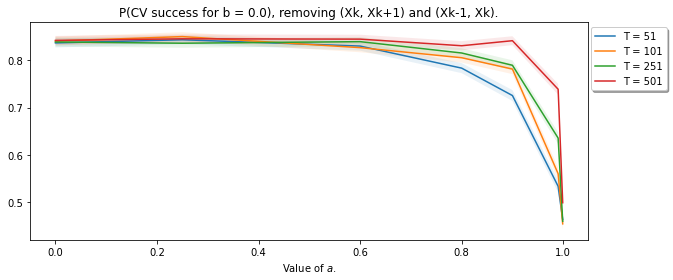
****

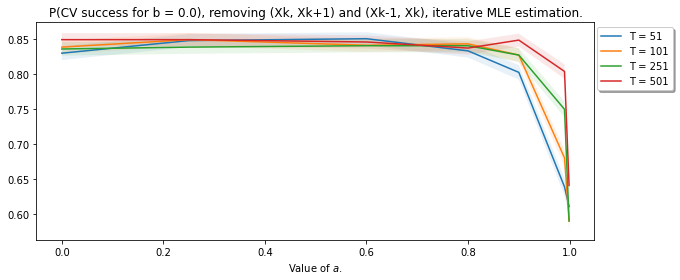
****

### Approach 2. Remove (Xk-1, Xk) and (Xk, Xk+1) pair.



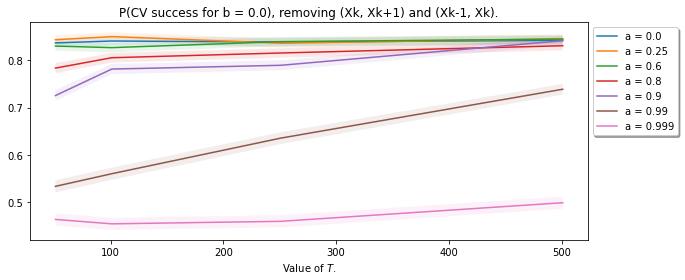
***As a function of a:* Decreases dramatically for *a* close to one, larger *T* is higher*.***

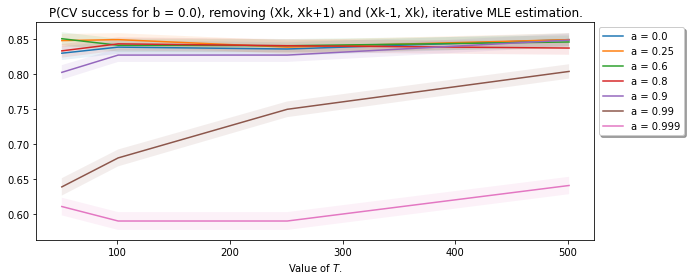
****

****

***As a function of T:* Remains constant for *a* far from one, increasing for *a* close to one*.***

We see that the probability of success remains high. However, as *T* grows, the probability decreases. This does not seem to matter for the value of *a*, interestingly. Furthermore, one might expect the probability of success to increase when we have a larger sample. However, the estimate for a will be better, and the difference between the LOOCV and the regular estimate becomes smaller and smaller.

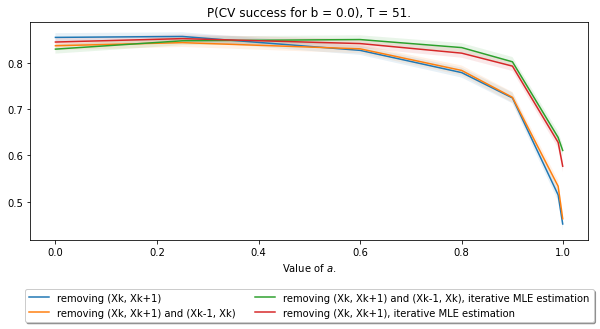




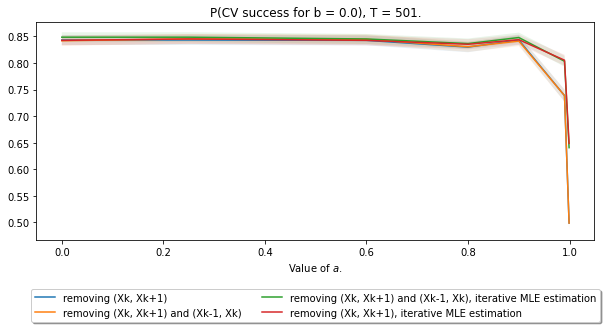
## Influence of method

**As a function of *a*, for different values of *T*: Difference for *a* close to one, re-estimating is better. Difference shrinks as *T* is larger.**

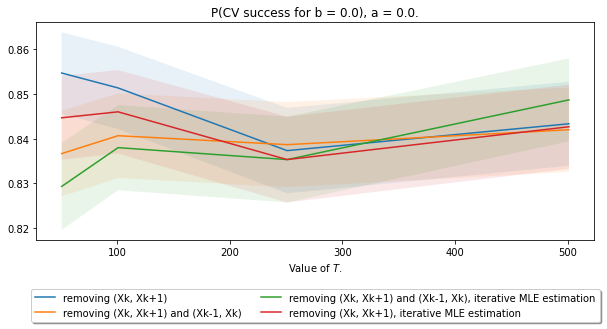
We see that the type of method slightly depends for large value of *a*. Here, the difference is between whether we iteratively estimate or not, not whether we exclude both pairs.



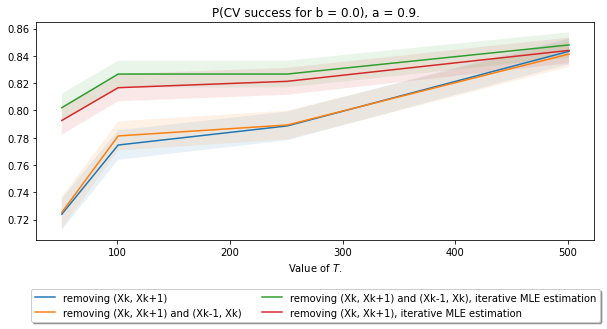
For smaller *a*, we see again that there is a difference between re-estimating for large *a*, but barely any difference between removing one or two pairs. Removing just one pair seems to be slightly “beneficial”, but the difference is almost negligible.

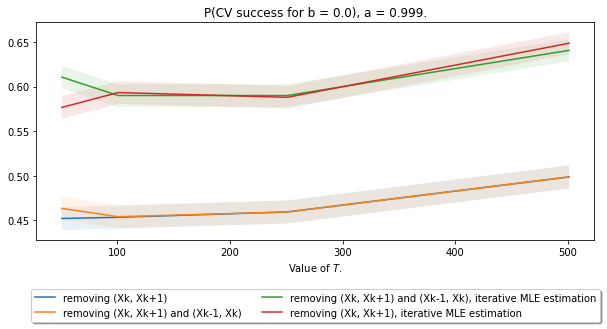


**As a function of *a*, for different values of *T*: Difference negligible for small *a* especially if *T* is large. If *a* is larger, differences grow much larger between estimating and re-estimating. Difference between one or two pairs small, but more noticeable for small *T*.**



We see that for small T, only removing the one sample results in a “higher probability”, but from larger values onwards, this difference seems almost negligible.

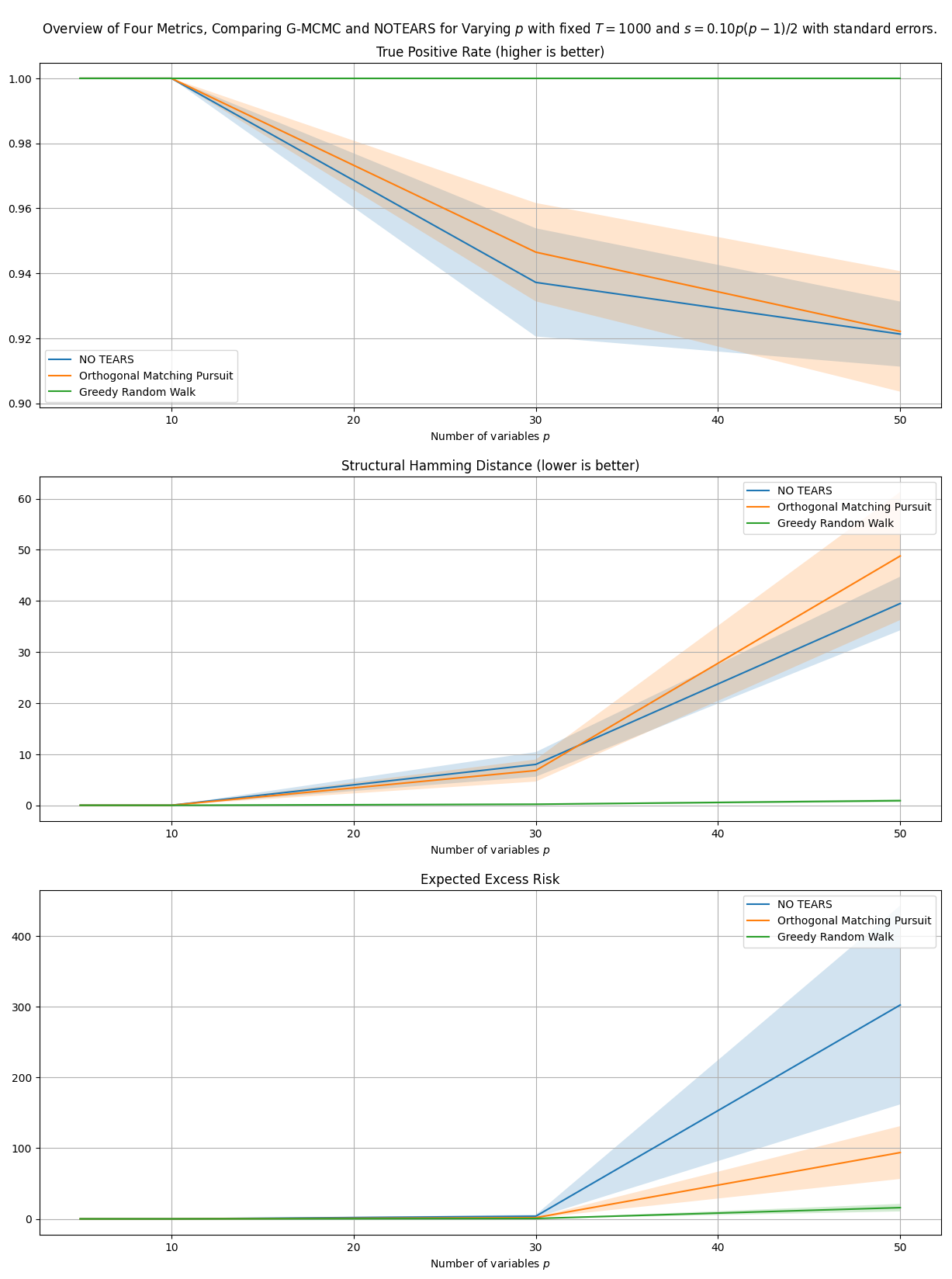




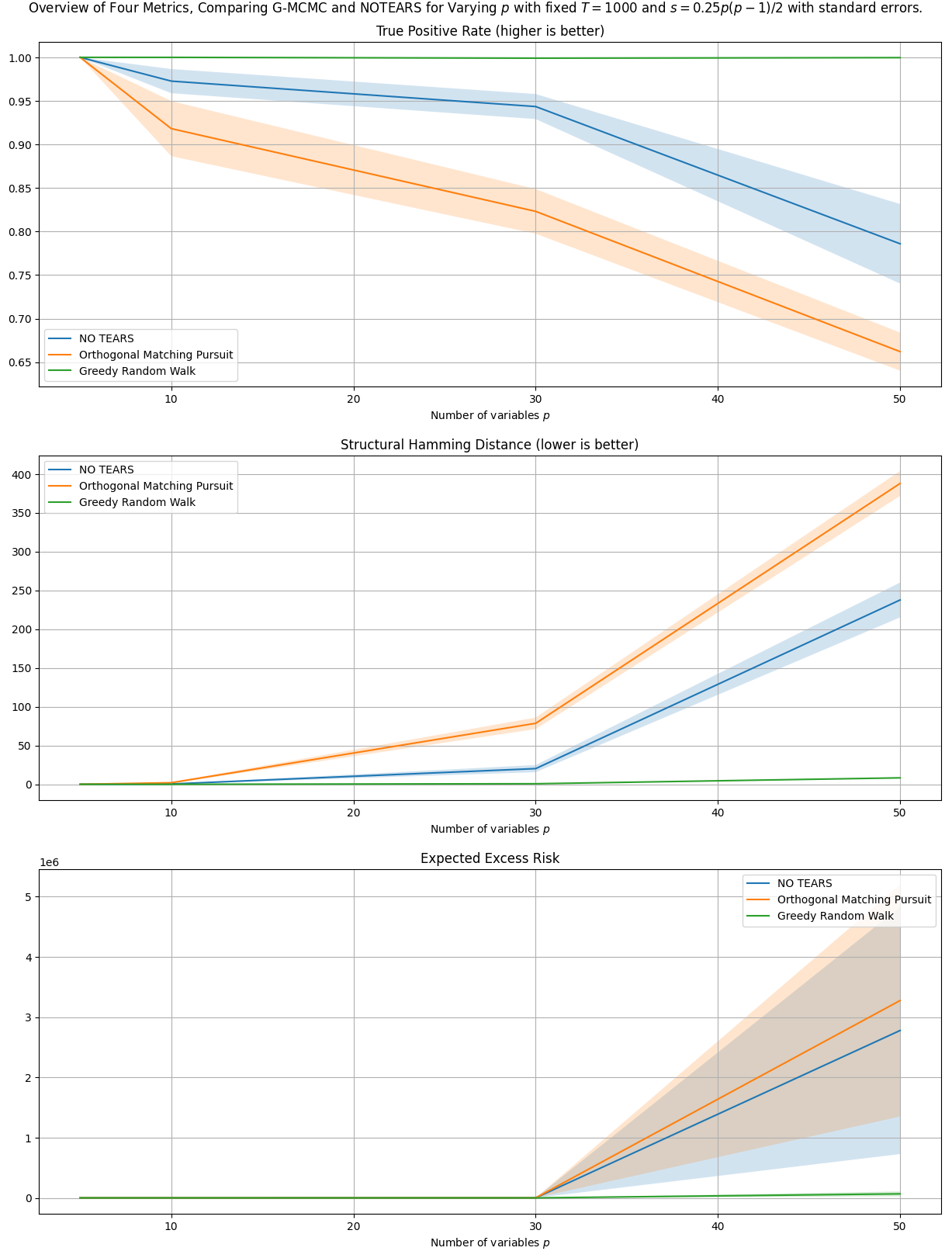
# Constant Sparsity Results

Tried sampling the three methods, but now the number of edges is , where .

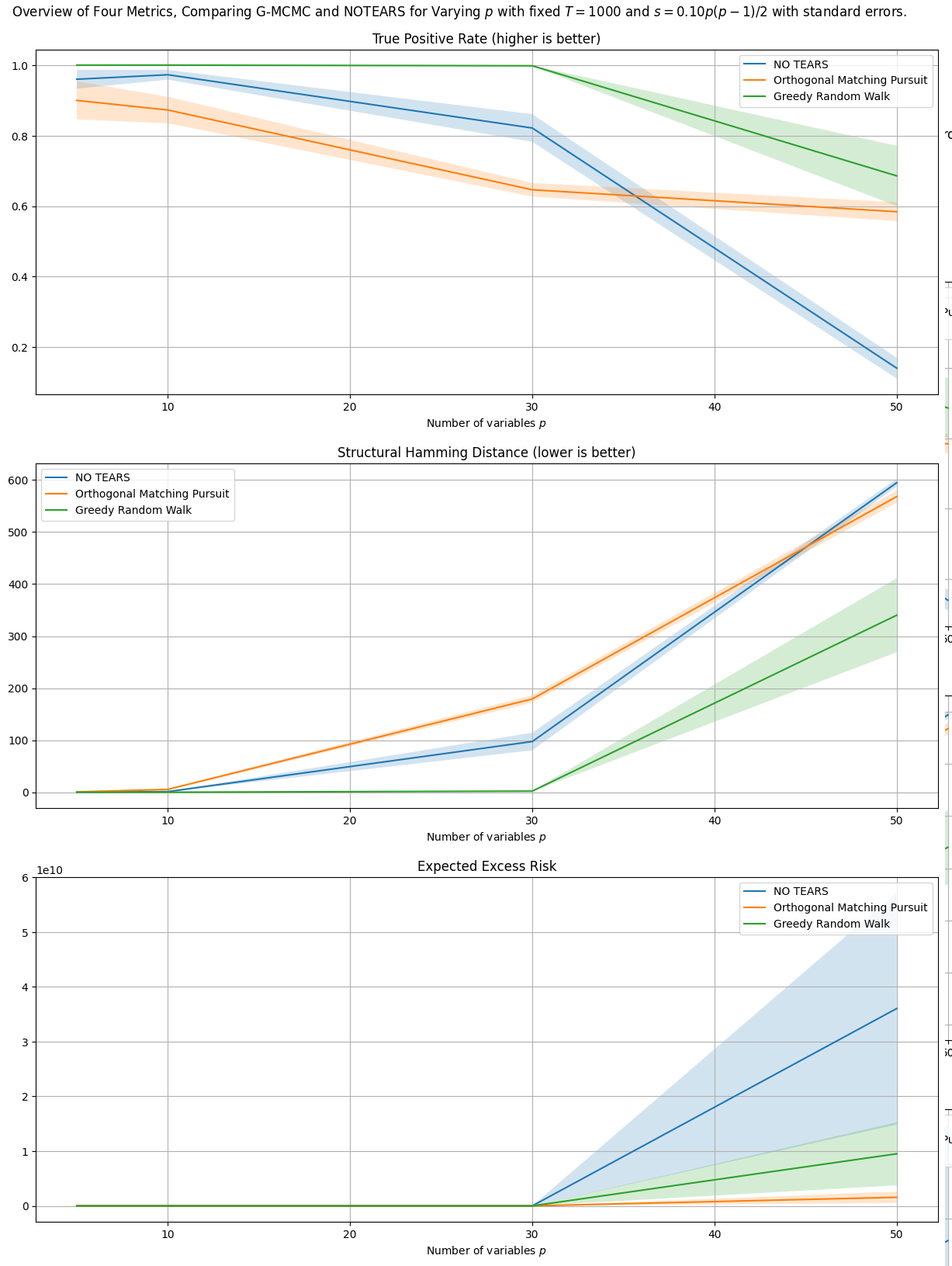
### *n = 1000, p in {5, 10, 30, 50, 75, 100}, c = 0.10*



### *n = 1000, p in {5, 10, 30, 50, 75, 100}, c = 0.25*



### *n = 1000, p in {5, 10, 30, 50, 75, 100}, c = 0.50*

**

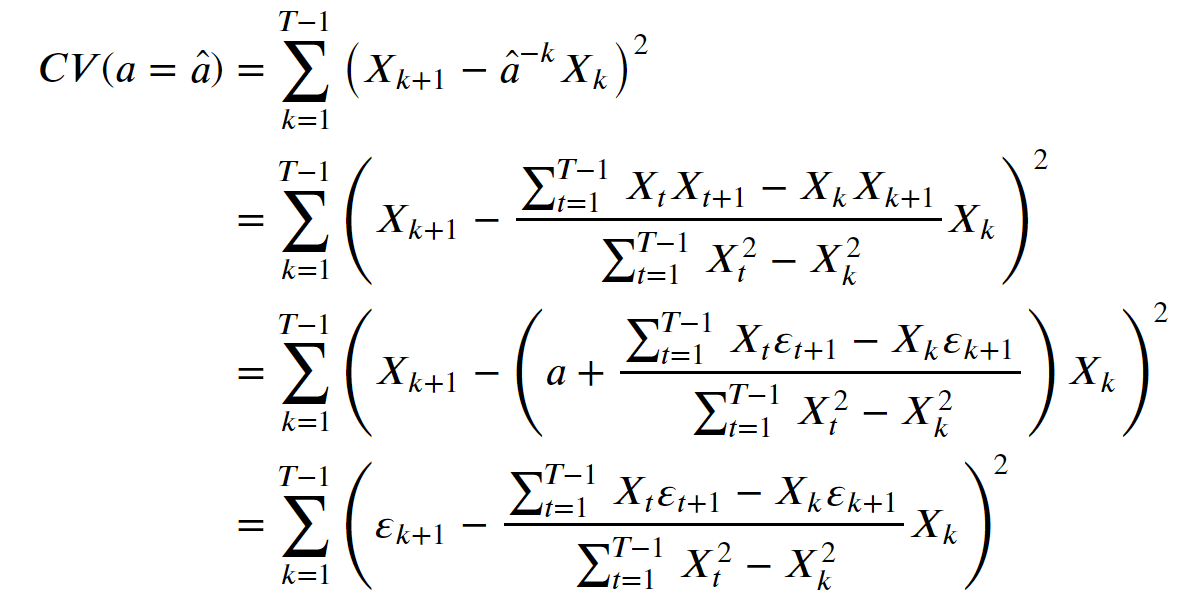
### *n = 1000, p in {5, 10, 30, 50, 75, 100}, c = 0.75*

### *n = 1000, p in {5, 10, 30, 50, 75, 100}, c = 1.00*

# 

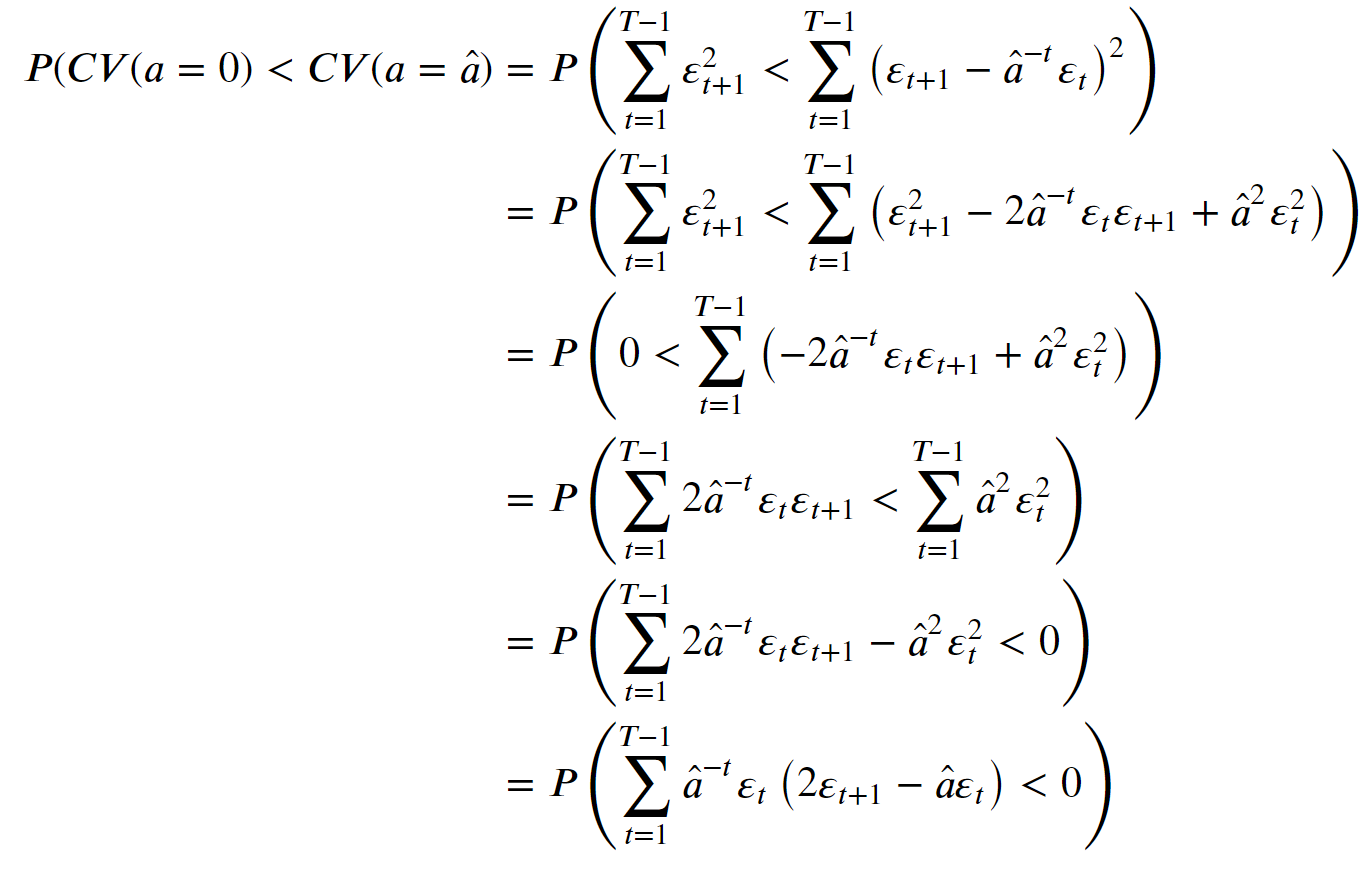
# Deriving 0.82 probability

Difficult. Tried the easiest setting, where *a = 0* and there is no offset. Then, the true squared error would be Chi squared distributed with *T – 1* degrees of freedom. However, the cross-validated squared error is not exactly this, as we have this estimation for *a*, which is close to zero but not quite.



For 𝑎=0 , the bottom of fraction is chi squared distributed with 𝑇−2 degrees of freedom.

Unfortunately, the top of the fraction is a product of two iid standard normal random variables, which is equal to 𝜋 times a Modified Bessel Function of the Second Kind distribution.





**First Moment**

First part has approximately mean zero. Makes sense as it is a sum of a product of three “almost independent” “almost normal” random variables with all zero mean.

Second part should have approximately mean one, as the variance of epsilon is 1, and the variance of a is 1 / T – 1. Summing T – 1 of these variables yields a mean of about 1.

All in all, we can expect a mean of about – 1.

**Second Moment**

Splitting into two parts might not be very helpful, as we cannot split them.

However, now, this sum becomes difficult to dissect. Individually, we can verify whether it is smaller than zero, but this total sum is more difficult. Some may be negative, some positive. Furthermore, the “distribution” of a is very difficult to analytically state.

**Perhaps some limiting behavior, e.g. CLT?**

It seems that the distribution of hat{a} is approximately normal with mean 0 and variance 1 / sqrt(T). One can argue that for T large enough, all things are “almost” independent.

**Chebychev’s Inequality**

Assume we have the mean and variance, then we can at least give a bound, albeit not super tight.

# Greedy Equivalent Search

Did some research into Greedy Equivalent Search. Watched some lectures, quite interesting. Basically, GES greedily finds the “best” edge addition / removal / reversal within the *markov equivalence class*. The *markov equivalence class* is basically all “graphs” which give rise to the same joint distribution. This is done until a local maximum has been reached.

After this, we start with the backwards phase, where we iteratively remove edges until we reach a local optimum again. In the end, we do not have a DAG necessarily, but we have a markov equivalence class of directed acyclic graphs. To compare, NO TEARS orients the edges in the correct direction to give it the most optimistic judgement.

# Other Structure Learning Benchmarks

Sachs data is all over the place.

Many other datasets exist, but are they good in doing what we want? We are not interested in cause effect, but rather in prediction.