In the cogsci submission, I did qualitative comparisons to show that multiple explanations for procrastination are possible in Peipei’s NYU task

Now, I want to fit the models to get quantitative parameter estimates and also do a model comparison to check if really multiple models are equally feasible. Maybe I can also get best models for subsets that might match my model-agnostic data clusters.

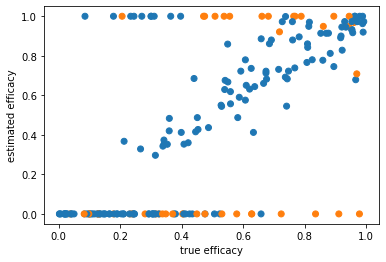
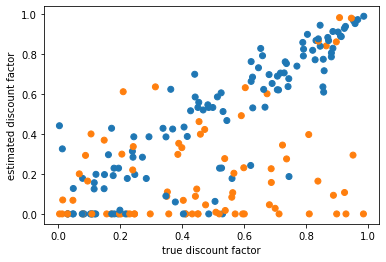
What methods to use for fitting?

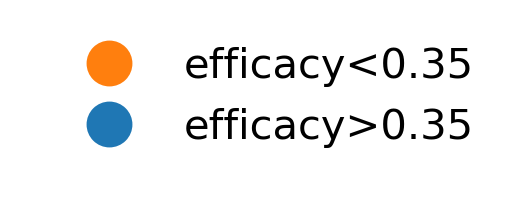
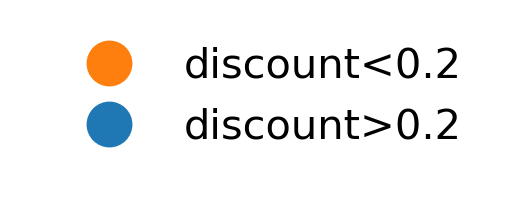
1. Likelihood-free methods – typically used when likelihood is intractable to calculate. Another use case is when data and model outputs are not compatible, so we can then project to a common space using summary statistics. Peter suggested this because we perhaps don’t want to make claims about when exactly students decided to work but rather about the broad patterns of working.
2. Likelihood-based methods – Starting from basic MLE to point Bayesian point estimates in MAP to estimating Posteriors exactly or approximately using MCMC or variational methods. Doing this hierarchically with random effects is also better so that we don’t overfit by fitting independently to each student (too many parameters to estimate!) but we group together data. Payam Piray’s CBM allows model selection and parameter estimation using multiple levels (it is Bayesian).   
   Since it is quite straightforward to estimate likelihood in my problem, I want to start with 2 and see where it goes.

Likelihood calculation: data is basically the string of states at each timestep. So we want P(s1, s2, …, sT | model, params). But our generative models output both states and actions, so we need to marginalize to get the probability of interest: Σ(a1, a2, …, aT) P(s1, a1, s2, a2, …, sT, aT | model, params) which can be simplified a lot due to the Markov structure.

Now, I attempt a basic MLE (starting with the basic model) and do a parameter recovery to see if this likelihood-based method could work. I generate data consisting of a trajectory over 16 weeks (so 16 data points) with some known parameters, and use scipy optimize to find the parameters that maximise the likelihood of this trajectory. How well does this perform?

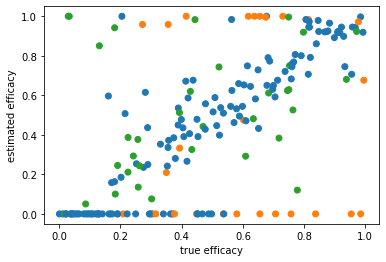
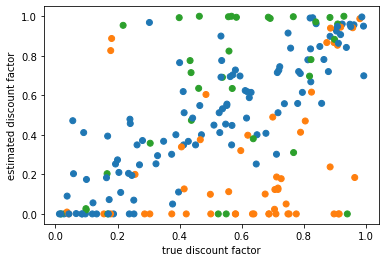
1. Basic model where all params are fixed except efficacy and discount factors

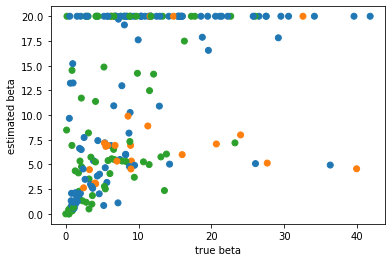
(reward\_shirk = 0.1, effort\_work = -0.3, beta = 7, reward\_work = 4.0, reward\_extra = 1.0)  


This is actually quite good in a reasonable range of params. When efficacy is too low, there is no point working 🡪 so hard to recover here and also when discount factor is v low, similar problem.

1. Now allow beta to be free. (other params same as before)

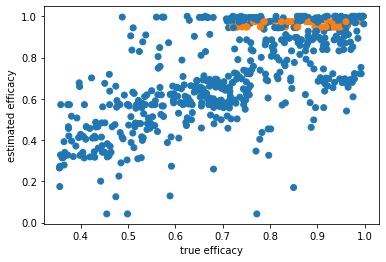
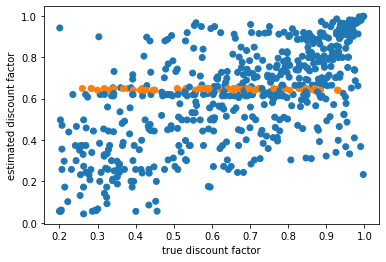


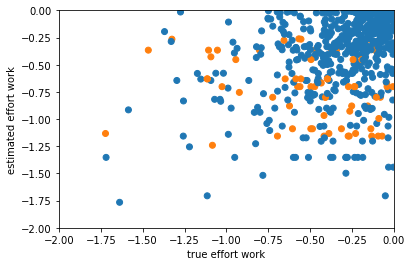
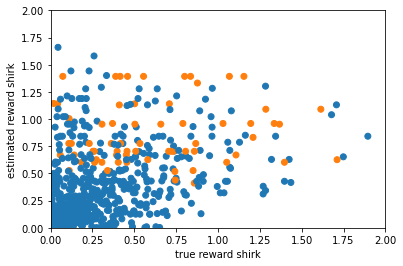


’bad’ regions parameter space: when beta is high (can’t differentiate beyond a point, they are all deterministic), discount factors and efficacys are too low. The recovery is still good.

1. Now free params: efficacy, discount factors, reward\_shirk, effort\_work  
     
   Things get a bit more challenging here. First of all more parameters to estimate from just 16 points. Second, multiple combinations of parameters can explain the same data : say true reward\_shirk is high and so there is more reason to put off work, this can also be due to higher discount factor or higher efforts – so exact params need not be recovered. Or if effort and shirk rewards are both proportionally increased, their tug-and-pull might preserve behavior similar to the true params. Finally, finding unrecoverable parameter spaces is not so easy because it depends on other parameters. So we cannot say that only effort above a certain magnitude is unrecoverable, because effort within the range can also be relatively too high (to be recovered)) if for eg reward\_shirk is too low, discounting is too steep or efficacy is low.   
   Reward and effort are sampled from exp(scale=0.5) and efficacy>0.35 and discount >0.2. But even after careful sampling we often get parameter estimates where efficacy = 0.0 (no work done at all) or reward\_shirk = 0 (always work). So recovery plots excluding this:

There seem to be special parameter combinations that many solutions converge to:  
0.648479, 0.973459, 0.672226, -1.08684 (here the effort is high enough, discounting steep enough and efficacy high enough that one should work only in the final timestep) – so all parameters that give this pattern go to this:

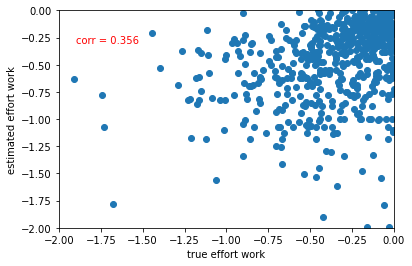
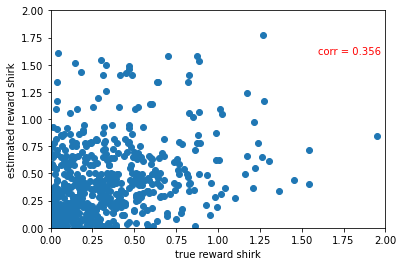
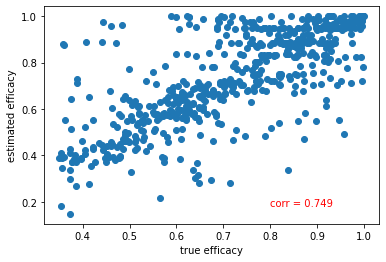
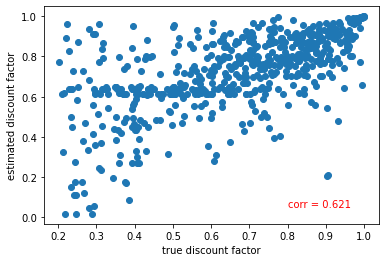




Especially reward\_shirk, effort\_work seem unrecoverable:

1. Am I actually finding the minimum? Start from more initial points, also set one of the initial points to the true params
2. Analyse loss landscape 🡪 variance, covariance from hessian, scatter plots (what are the tradeoffs, do some parts of the likelihoods make it unrecoverable)
3. What can make it better? More trajectories or experimental manipulations?

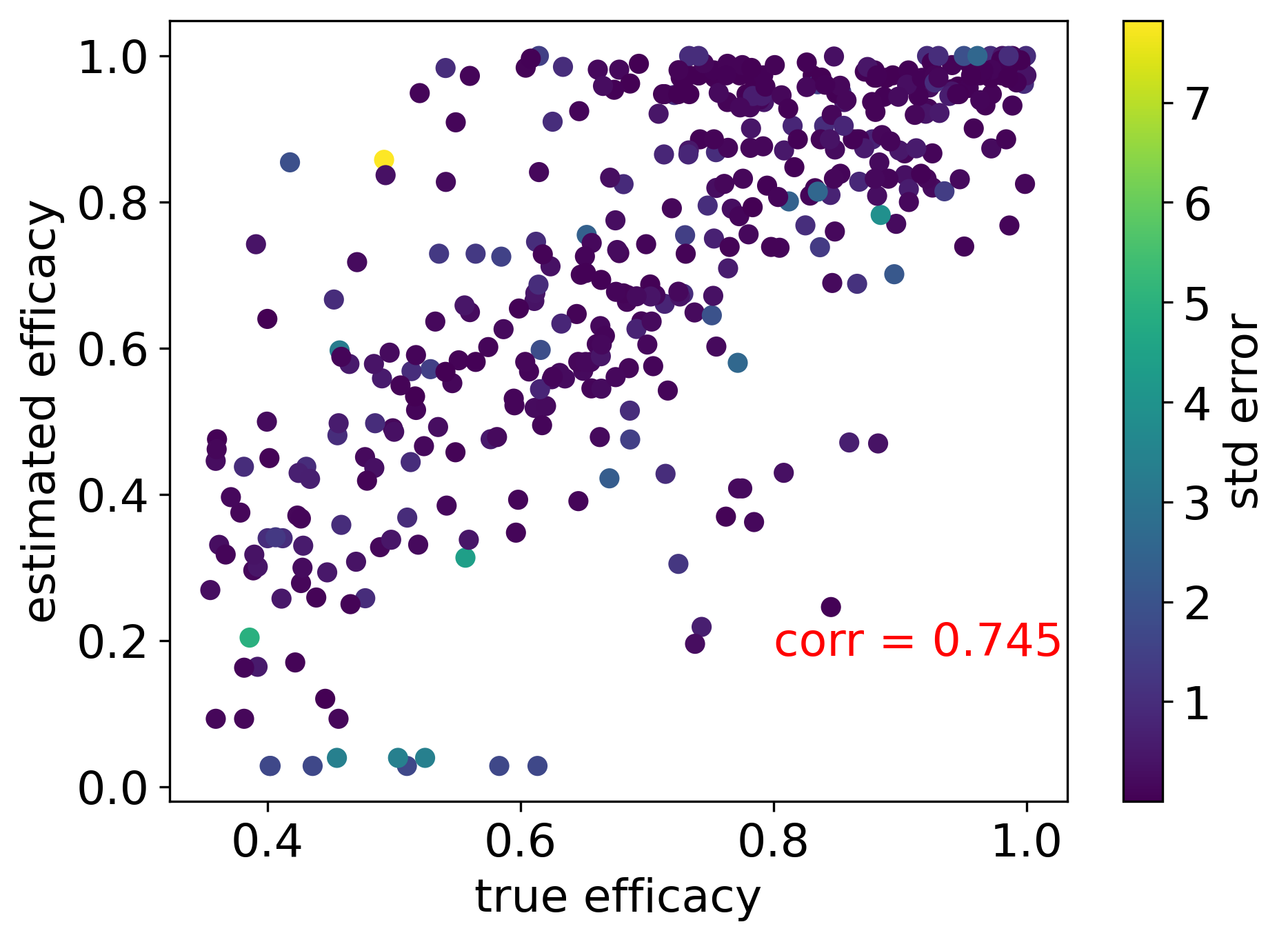
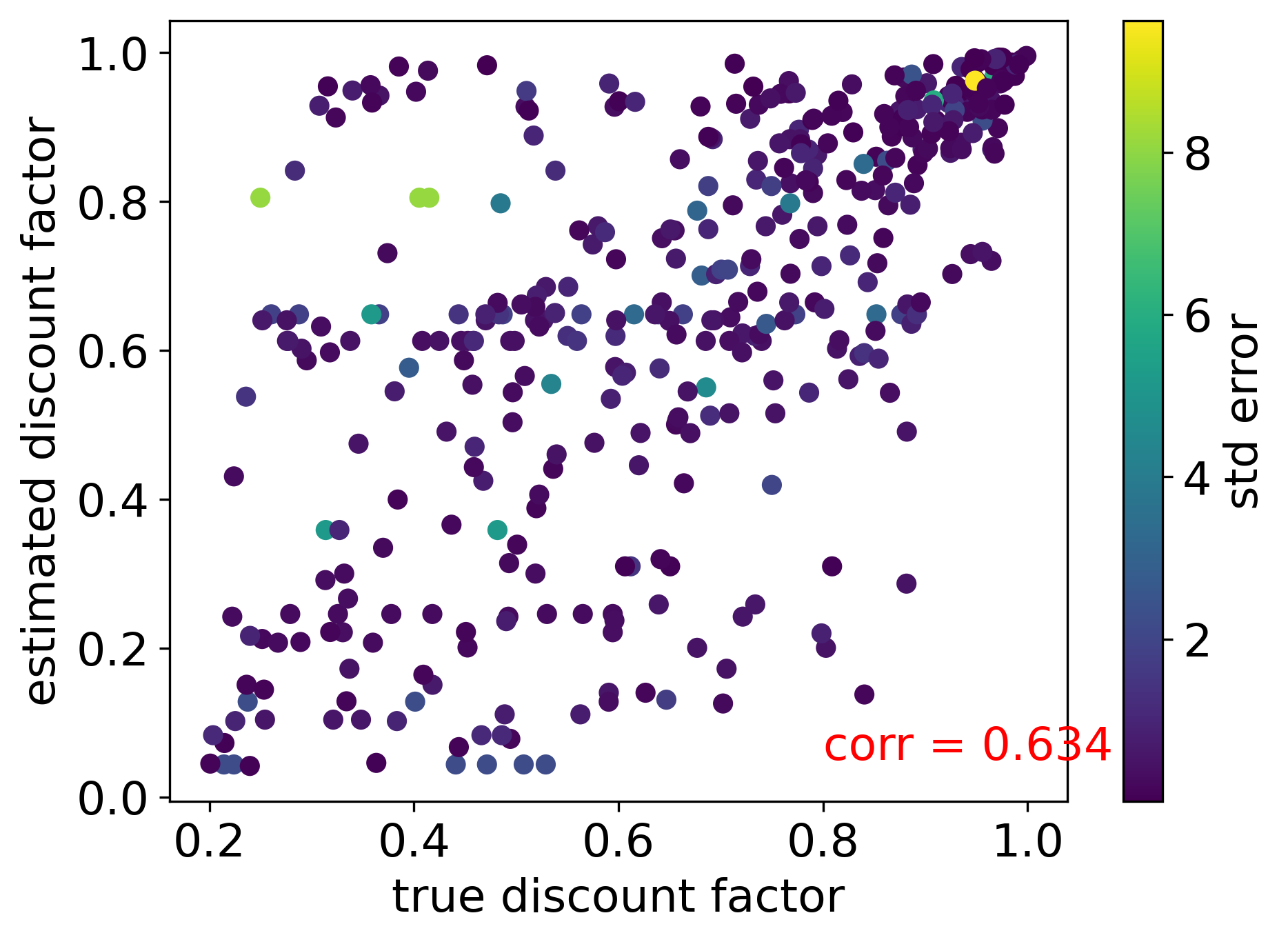
With more initial points (10 instead of 5 earlier)

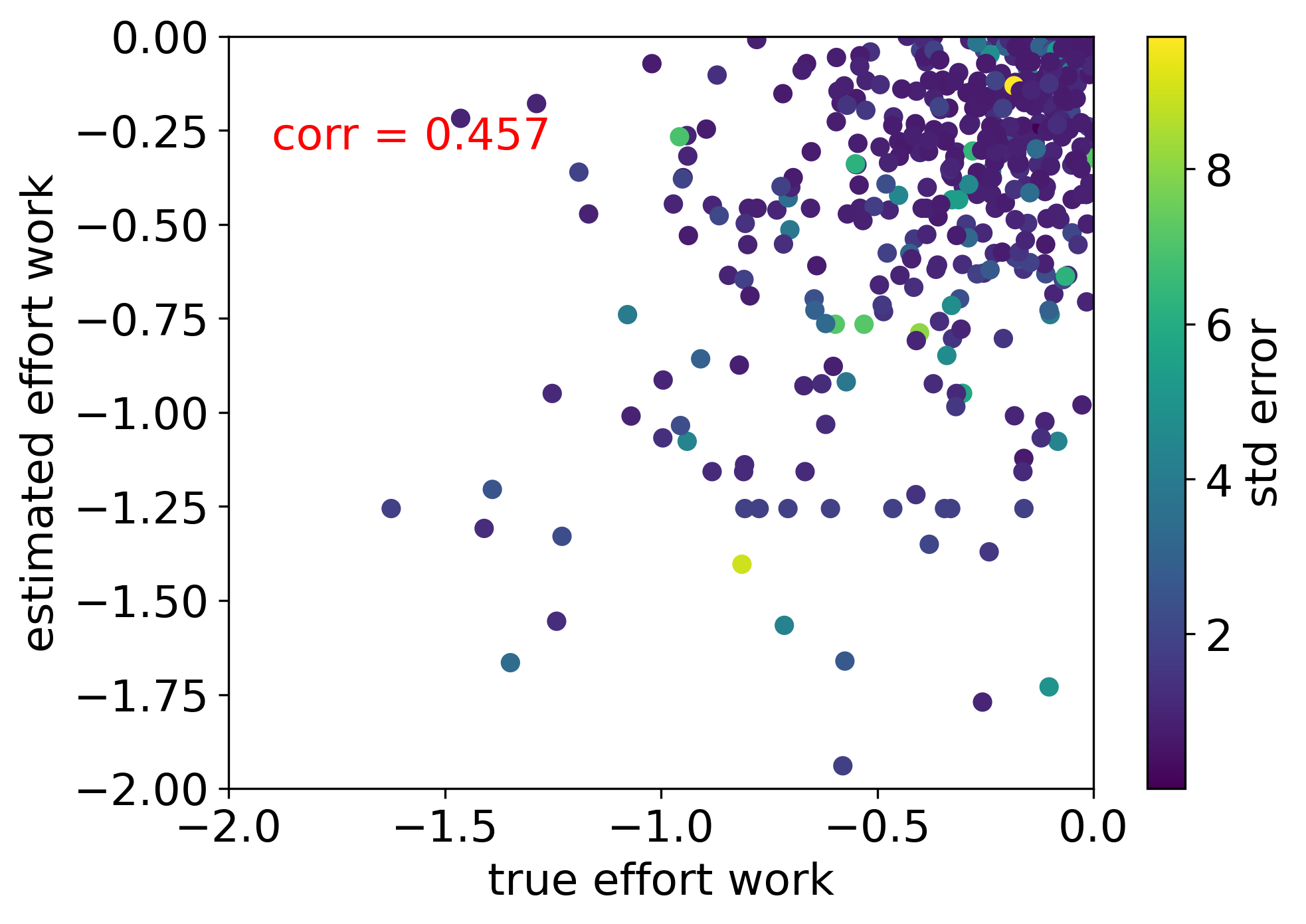
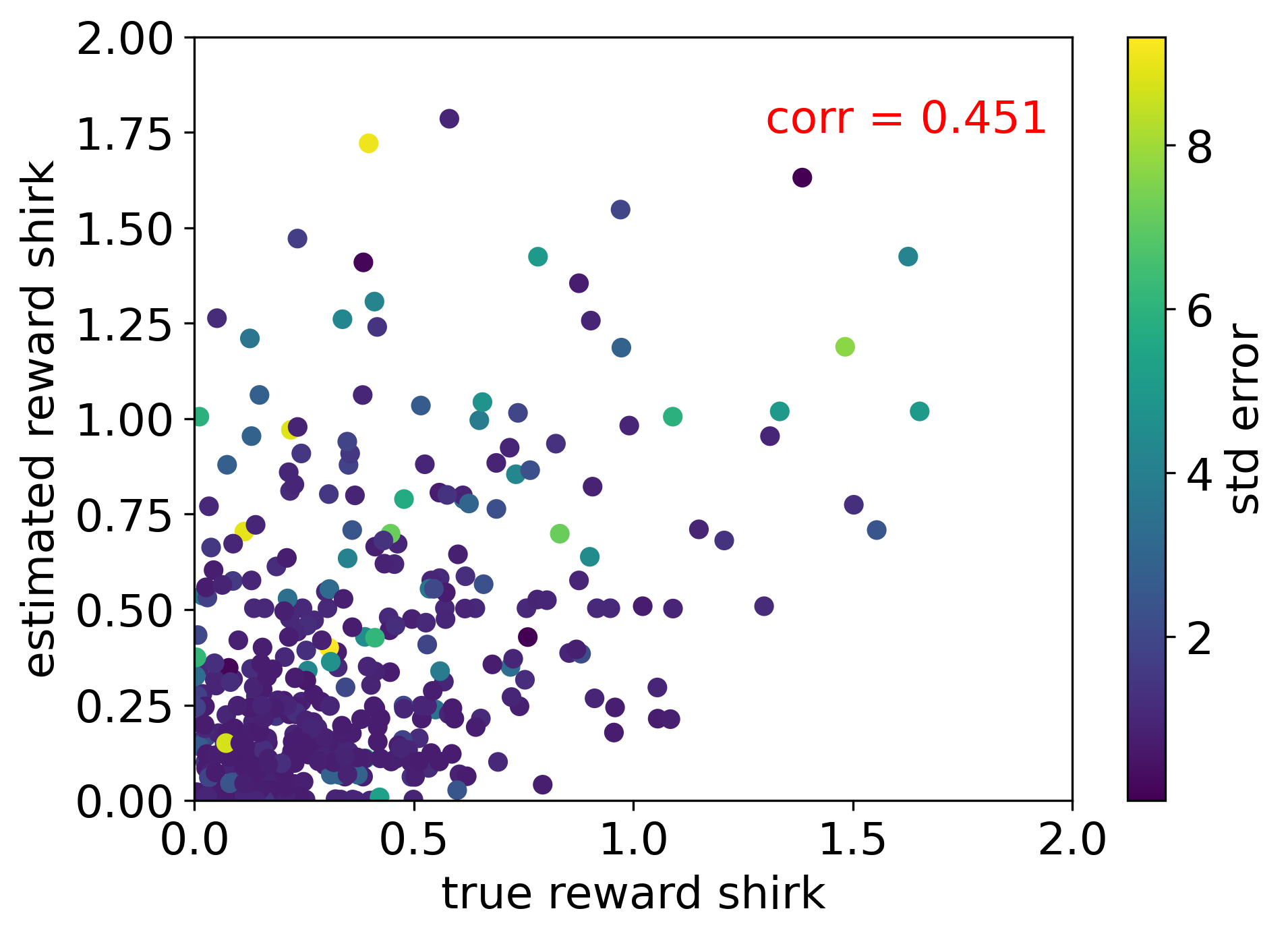


What about when we include the true params as one of the initial points?

(as usual, I remove points where the estimates == 0 🡪 about 250 out of 700 and those with v high variance > 10 🡪 about 40 out out of 450 )

Basically the same as without initial param = true param, so the minimisation routine is finding a better parameter





Maybe one issue is that when I sample reward and effort, because I do it in an exponential manner, there are a lot of low values that get selected 🡪 these lead to bad fits (no need to work here). So maybe better to have a sampling distribution that peaks at 1 or so. Even from the recovery plots, it looks like those above 0.5 get recovered better.