4.13.23 RF notes & strategizing

Something odd to me this am, I implemented the RF for both the original and new samples and the RF in sklearn.RF.Regressor does super well like an R^2 of .91-.99 on test data.

It may be deeply overfit but that seems OK in this context [it's not really about having the most parsimonious model or a mechanistic interp, we just want really good predictions for new-but-similar combinations of S x N.]

In fact between the deep neural net & the random forest these are 2 inference approaches that give, visually at least, pretty good test sample recovery for the mean & hill 1.

(the KNN doesn't do well, @ least w/ this data config)

So here's a proposed next step that I think warrants some reflection...

Seems good to do a model comparison / test of recovery of actual hill 1 values.

Bad vtremes aren't the end of the world but I am curious to benchmark how well these methods can do.

Eventually I envision this as integrated into a function / automated pipeline where we just use a trained model to generate FS distributions.

↓ static

* I'm tempted to build / start to build this infrastructure now... but I think that's getting ahead.

So what I do see as a next step...

Report where we fit:

lm
DNN
RF

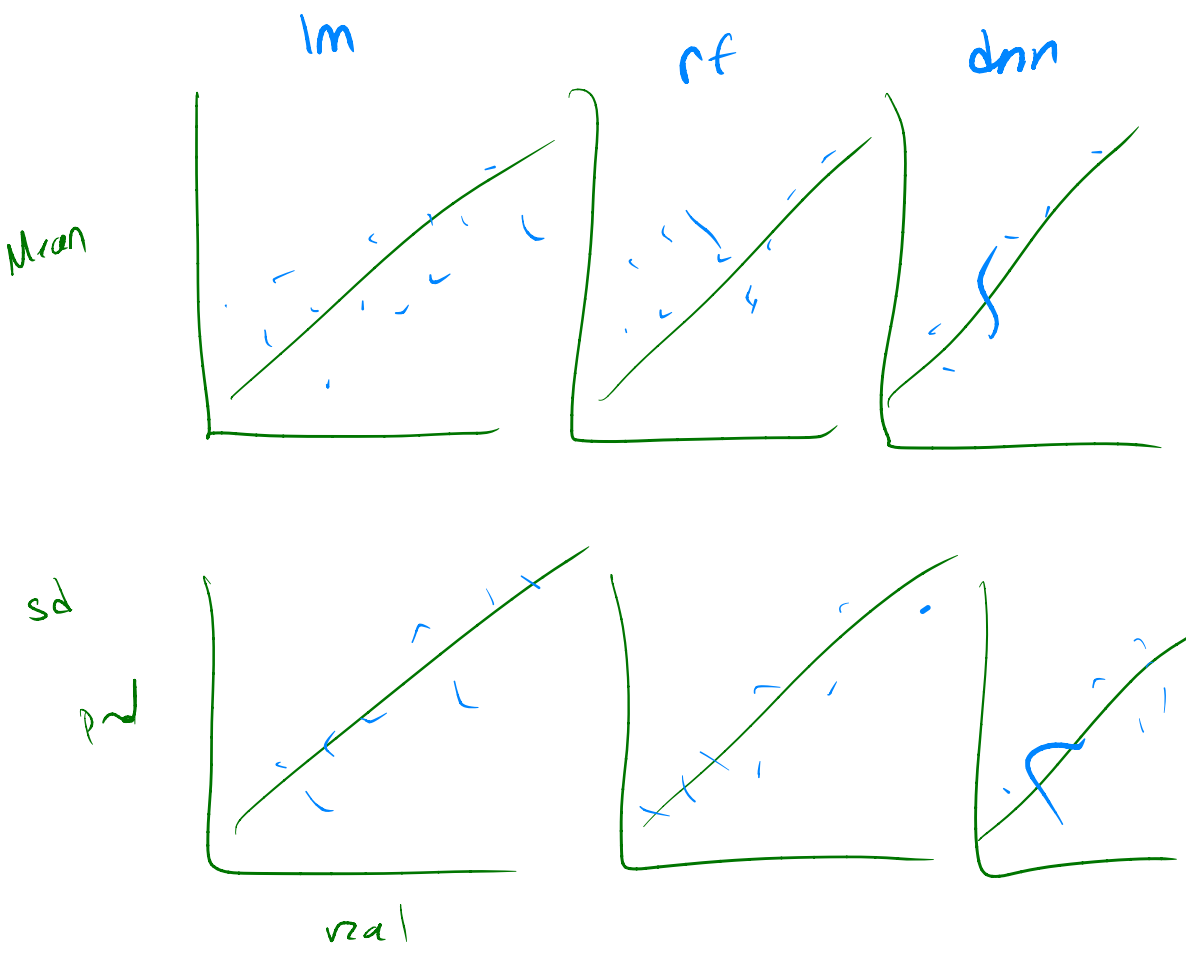
→ to mean & sd of hill 1

+ use this to generate "draws" of hill 1 across the test dataset.

↑ a step in here:

generate predictions of mean, sd of hill 1 for test sets and test recovery there. that's essentially where ggcs but... yeah.

Note currently I am not doing joint predictions. If I were using this for inference I think that would be a problem but in this context I think — probably — it's OK to let it slide... here I only care about utilitarian prediction accuracy, not inference about how $S/N \sim \mu/\sigma$.



mod	mean score	sd score
lm	...	...
rf	...	...
dnn	...	...

This would still be a step away from actual samples from a normal // actual but it's an important checkpoint on the way.