

# Prototyping Preference Inference Using Stan

John Mount

Win Vector LLC

<https://www.win-vector.com/>

September 1, 2024

# Win Vector LLC

- Win Vector LLC is a statistics, machine learning, and data science consultancy and training organization.
- Specialize in solution design, technology evaluation, and prototyping.
- Contact: [jmount@win-vector.com](mailto:jmount@win-vector.com) .

# Outline

- The problem
  - Some history
- Solutions
  - Experiments/Results
- Observations
- Conclusions/recommendations
- Next steps

# The Situation

- Users (or user personas) have intrinsic unobserved ***preferences*** or valuations
  - Personas are either representatives of a group of users or a label collecting together a group of users.
  - This lets us assume we can collect a lot of per-persona data.
- We observe user ***behaviors***
  - Typical observation: we presented 5 alternatives and the user purchased one.
  - Observation contaminated both by presentation position and alternative items in the presentation.
- Can we estimate persona preferences from persona behaviors?
  - Often called “learning to rank.”
  - We can use estimated preferences to plan (a lot more than just predicting future behaviors).

# Our Thesis

- Factor overall system performance into the product of:
  - “**model structure**” (how well the explanatory variables can reproduce scores).
  - “**presentation hygiene**” (how well we account for the censoring effects of presentation on the problem).
- To improve: must know where we *actually* are losing quality
  - So want to attribute quality loss to each separately.



# Example Problems

- User is shown 5 products online and clicks on one.
- User tastes 5 wines and buys one.
  - (repeat with 100 users thought to be in same persona)



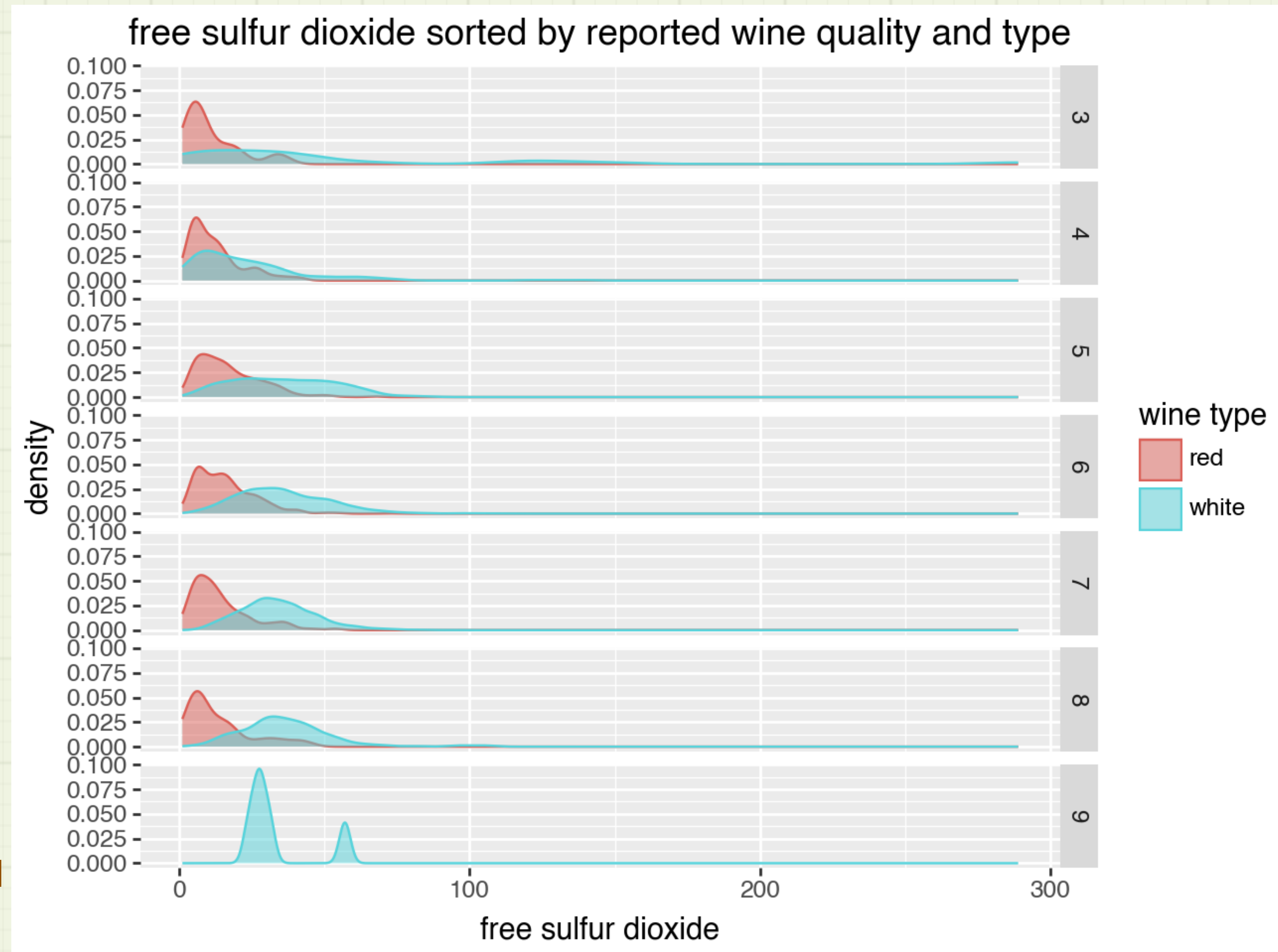
We **wish** they would tell us their numerical valuation of each wine!

# The Wine Example

Explanatory variables:

alcohol  
chlorides  
citric acid  
density  
fixed acidity  
**free sulfur dioxide** (shown as example)  
*is\_red* (we interact this with all other variables)  
pH  
residual sugar  
sulphates  
total sulfur dioxide  
volatile acidity

Not actually a good set of variables for the task  
(physical chemical variables instead of domain perceptual and provenance variables).



<https://archive.ics.uci.edu/dataset/186/wine+quality>

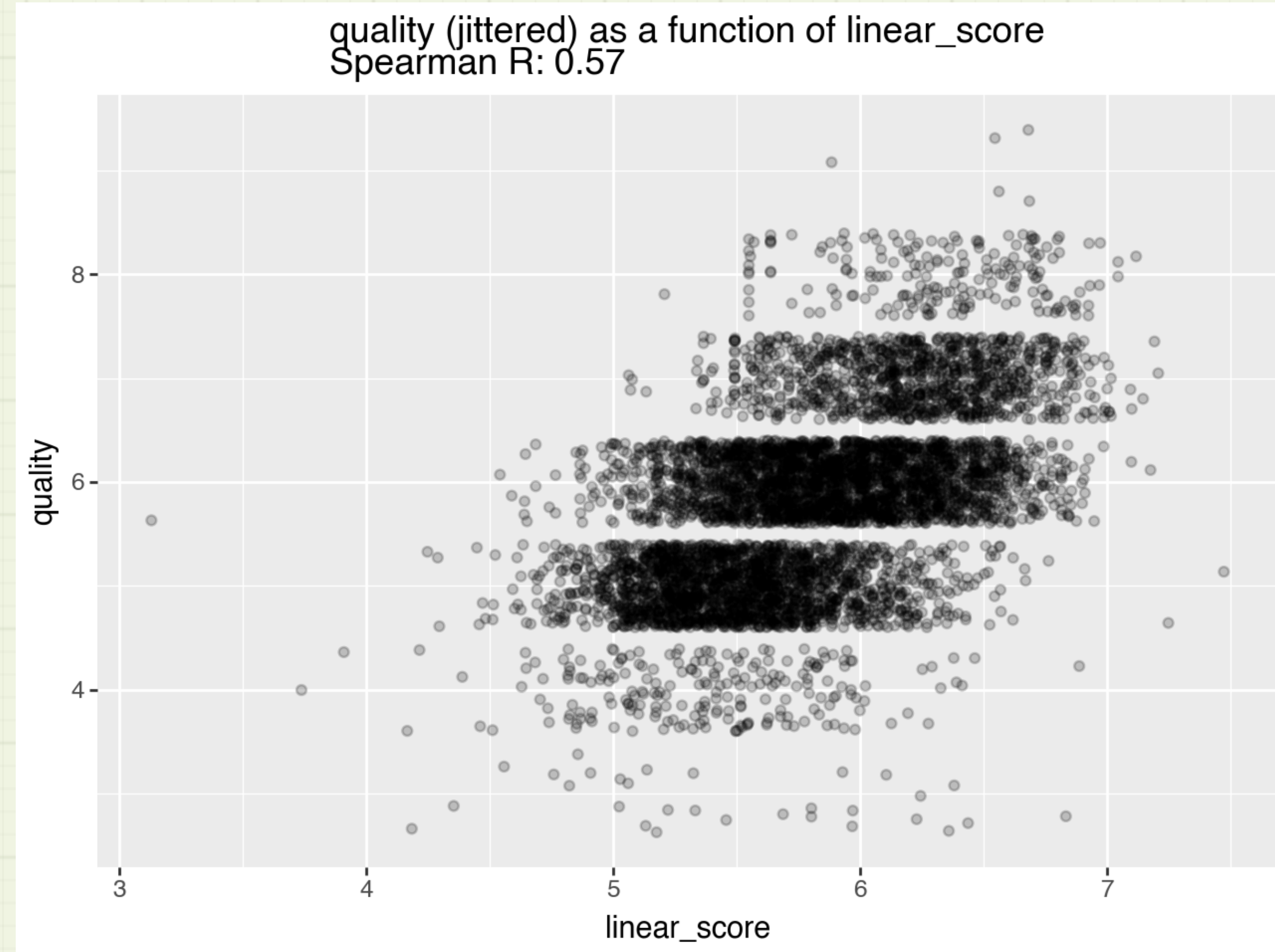


# The Data we Wish For

User tells us their darn scores!



With 100 tasting lists we observe heavily noisy and censored outcomes from about 500 wines out of our 6497.



(Above fit on all 6497 wines, without noise or list structure. Notice reported quality unfortunately isn't very well modeled as a linear function of *these* variables. Please remember 0.57 as "best possible" for this model structure.)



# Confounding Issues

- Presentation position may be a strong influence
  - Users may be biased to pick earlier presentation positions, forget earlier positions, or even not even try/look-at later positions!
  - Item quality may correlate with position (would require additional interactions to be introduced in the model).
  - What other items are in the list or panel has a huge effect.
- Data is noisy
  - User may make different choice when re-presented
- Data is low information or censored
  - Selecting 1 out of 5 positions is only 2.3 bits of information
- Presentation is not an ideal experiment (is influenced by business needs)
  - Ideal statistical procedure would be either:
    - Only present 2 alternatives and force a selection (Conjoint Analysis)
    - Bully exact scores out of the user (unlikely, they may not even be able to express numeric utilities)

# Learning to Rank History

- Tradition: each field develops solutions ignoring all other fields
  - BIG topic in search engines
    - Joachims, T. (2002), "Optimizing Search Engines using Clickthrough Data", *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*
    - Tie-Yan Liu (2009), "Learning to Rank for Information Retrieval", *Foundations and Trends in Information Retrieval*, **3** (3): 225–331
- Huge survey: [https://en.wikipedia.org/wiki/Learning\\_to\\_rank](https://en.wikipedia.org/wiki/Learning_to_rank)
  - Claims that logistic regression does not work
    - “Bill Cooper proposed **logistic regression** for the same purpose in 1992<sup>[19]</sup> and used it with his **Berkeley** research group to train a successful ranking function for **TREC**. Manning et al.<sup>[40]</sup> suggest that these early works achieved limited results in their time due to little available training data and poor machine learning techniques.”
- May over sell presentation hygiene.

# Our Goal

- Estimate the user (or user cohort) *intrinsic* preferences (independent of presentation position and other items).
- Eliminates some systems, such as `xgb.XGBRanker` (as it reproduces predictions over whole lists)
- From: [https://xgboost.readthedocs.io/en/stable/python/examples/learning\\_to\\_rank.html#sphx-glr-python-examples-learning-to-rank-py](https://xgboost.readthedocs.io/en/stable/python/examples/learning_to_rank.html#sphx-glr-python-examples-learning-to-rank-py)

```
X_test, clicks_test, y_test, qid_test = sort_ltr_samples(  
    test.X,  
    test.y,  
    test.qid,  
    test.click,  
    test.pos,  
)
```

```
...  
ranker.predict(X_test)
```

(notice the `qid` column, which means data is in presentation and we `xgb.XGBRanker` is compute probability of selection with respect to the alternatives, not utility.)



# The Models

model	model class	explanatory variables	modeling units	forces linear structure?	how position bias is modeled
<b>Stan sequential inspection</b>	list-wise	direct features	utility	yes	as early stopping (modeling probability of continuing)
<b>Stan list utility</b>	list-wise	direct features	utility	yes	as an additional estimated utility
<b>Stan pair utility</b>	pair-wise	direct features	utility	yes	as an additional estimated utility
<b>logistic differences, early stop</b>	pair-wise	per comparison pair difference in feature vectors	probability	yes	as additional variables and as early stopping (nothing after pick used)
<b>logistic differences</b>	pair-wise	per comparison pair difference in feature vectors	probability	yes	as additional variables
<b>logistic items (score model)</b>	item-wise	direct features	probability	<b>no</b>	ignored (could add position variables)

# Utility Formulation

- Suppose the user has a hidden valuation parameter  $\beta$  vector such that their valuation of a an item with features  $x_i$  is  $f_\beta(x_i)$ 
  - Often this is realized as  $f_\beta(x_i) = \beta \cdot x_i$ .
- Further assume for a list of items  $x_1, \dots, x_5$  the user values item  $i$  with a value of  $f_\beta(x_i) + e_i$  ( $e_i$  being a mean-zero noise term).
- Notice I have *not* chosen a probabilistic model!
  - Rank valuation is just an abstract number, utility, value, preference, or affinity.
- Variations:
  - Early stopping: They share only the  $i$  maximizing  $f_\beta(x_i) + e_i$  from  $i$  in  $1 \dots$  got bored in the tasting.
  - List-wise versus pair-wise: is the noise term  $e_j$  for the the picked  $j$  re-drawn on each pair comparison or not.

# List-Wise Observations (models reality)

Outcome ( $y$ ):

Encodes as:

“picked wine 4”



Simulates a single pick from a single tasting of each wine.



Outcome (y):



Not what anyone wants (as it may be a weaker inference), but easiest for most modeling systems.

Simulates the (odd) situation of us guessing the user's preferred wine and then giving them 4 paired tastings of it against the 4 non-preferred wines. Notice wine 4 is tasted 4 times (all other wines tasted once).

# Pair-Wise Observations (usually not preferred)

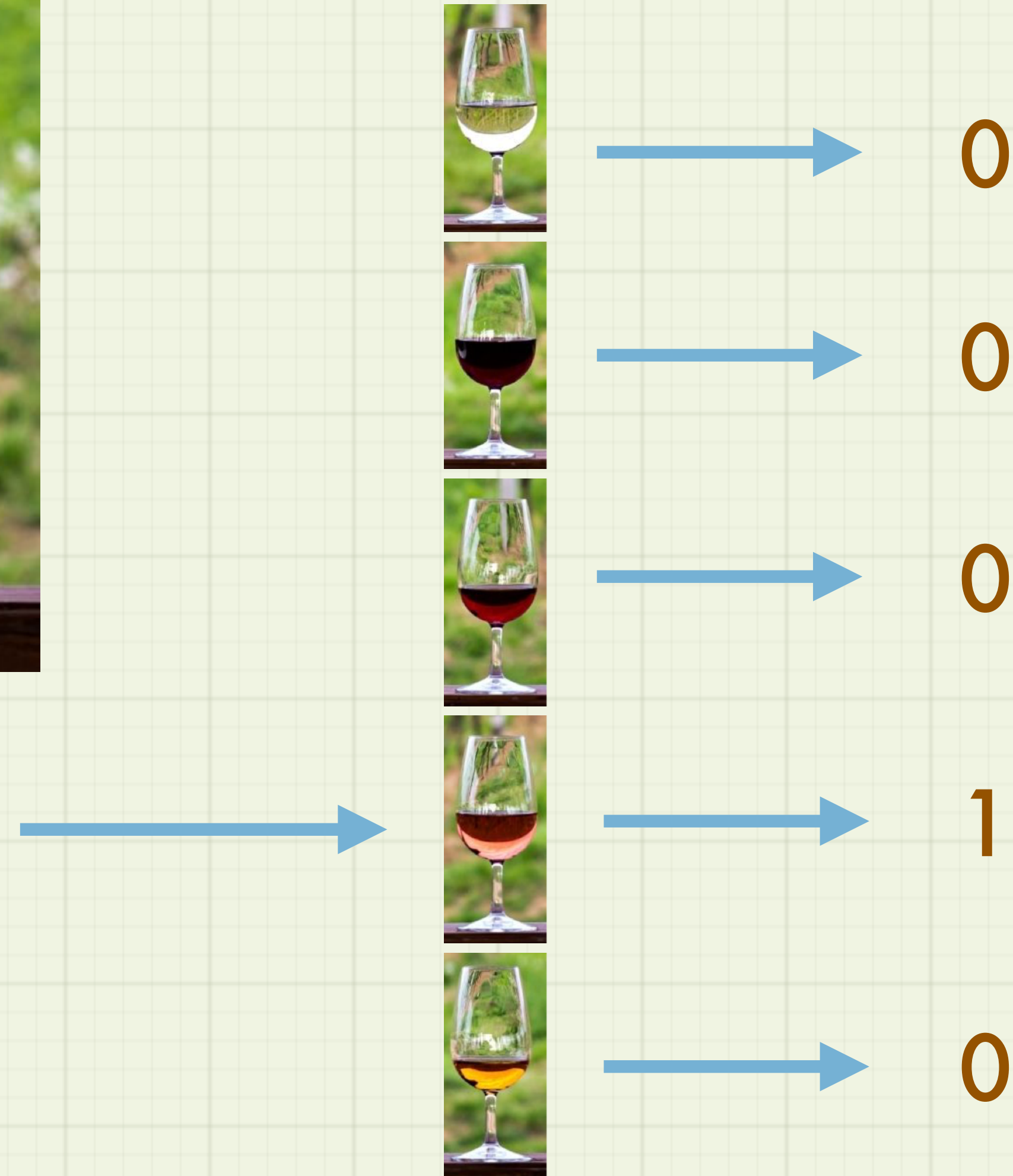




Outcome ( $y$ ):



# Item-Wise Observations (usually not preferred)



(wrongly) Treats each wine as being tasted alone.  
Ignores user picked no more that one wine.

# The Pick Data

Explanatory features  
are a combination of  
table lookup by  
item\_id and  
presentation facts (i.e.  
presentation position).

	item_id_0	pick_value_0	item_id_1	pick_value_1	item_id_2	pick_value_2	item_id_3	pick_value_3	item_id_4	pick_value_4
0	1569	0	1754	0	6425	1	2780	0	2646	0
1	4390	1	2031	0	2692	0	4416	0	1913	0
2	599	1	1808	0	64	0	59	0	1671	0
3	1392	0	2324	0	5815	0	1819	1	4567	0
4	2063	0	6283	0	3610	1	2085	0	5610	0
5	2010	1	1465	0	6388	0	25	0	420	0
6	5903	1	1374	0	312	0	926	0	5467	0
7	5194	1	3651	0	1494	0	1749	0	5865	0
8	5946	1	4527	0	5988	0	3021	0	4821	0
9	6469	1	6044	0	2787	0	5786	0	3709	0



# Feature Encoding

$x(item = 6425, position = 2) =$

6425	
fixed acidity	6.0
volatile acidity	0.34
citric acid	0.29
residual sugar	6.1
chlorides	0.046
free sulfur dioxide	29.0
total sulfur dioxide	134.0
density	0.99462
pH	3.48
sulphates	0.57
alcohol	10.7
is_red	False
posn_0	0
posn_1	0
posn_2	1
posn_3	0
posn_4	0

Feature table lookup

Encoding of presentation position. Could also encode demographics of participant Especially useful if we interact the demographic variables with the feature variables.

# Stan List-Wise Position Utility Model

Pass one-hot encoded position indicators to the model as features. Models presentation position as a utility trade-off.

```
...
transformed parameters {
  ...
  expect_picked = x_picked * beta;           // modeled expected score of picked item
  v_picked = expect_picked + error_picked;    // reified actual score of picked item
  expect_passed_1 = x_passed_1 * beta;        // modeled expected score of passed item
  expect_passed_2 = x_passed_2 * beta;        // modeled expected score of passed item
  expect_passed_3 = x_passed_3 * beta;        // modeled expected score of passed item
  expect_passed_4 = x_passed_4 * beta;        // modeled expected score of passed item
}
model {
  // basic priors
  beta ~ normal(0, 10);
  error_picked ~ normal(0, 10);
  // log probability of observed ordering as a function of parameters
  // terms are independent conditioned on knowing value of v_picked!
  target += normal_lcdf( v_picked | expect_passed_1, 10);
  target += normal_lcdf( v_picked | expect_passed_2, 10);
  target += normal_lcdf( v_picked | expect_passed_3, 10);
  target += normal_lcdf( v_picked | expect_passed_4, 10);
}
```

Point-wise model, simply uses `expect_picked` instead of `v_picked` throughout.

# Difference Encoding

- Exploit linear structure to directly encode differences
  - Explanatory variables: encode comparison pairs.
    - For example: item 6425 in position 2 **picked** and item 1569 in position 2 **not picked** encoded as:

$$x_i = x(\text{item} = 6425, \text{position} = 2) - x(\text{item} = 1569, \text{position} = 0)$$

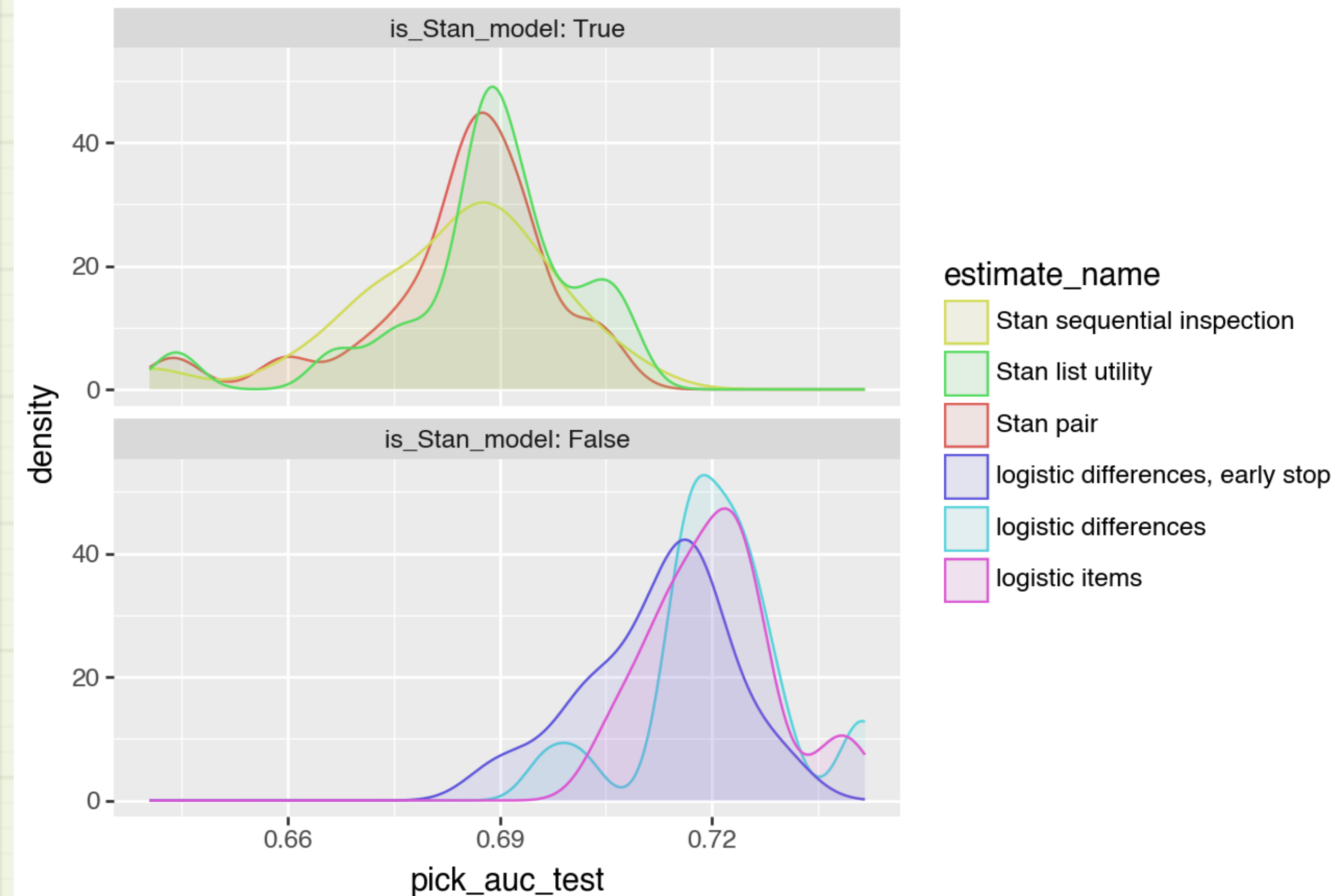
- The subtraction enforces that features have the same interpretation in picks and non-picks.
    - Outcomes encodes as “True”.
  - Outcomes/dependent variables must vary
    - So need to also encode a “False” outcome for  $-x_i$
- Encoding not natural or obvious.



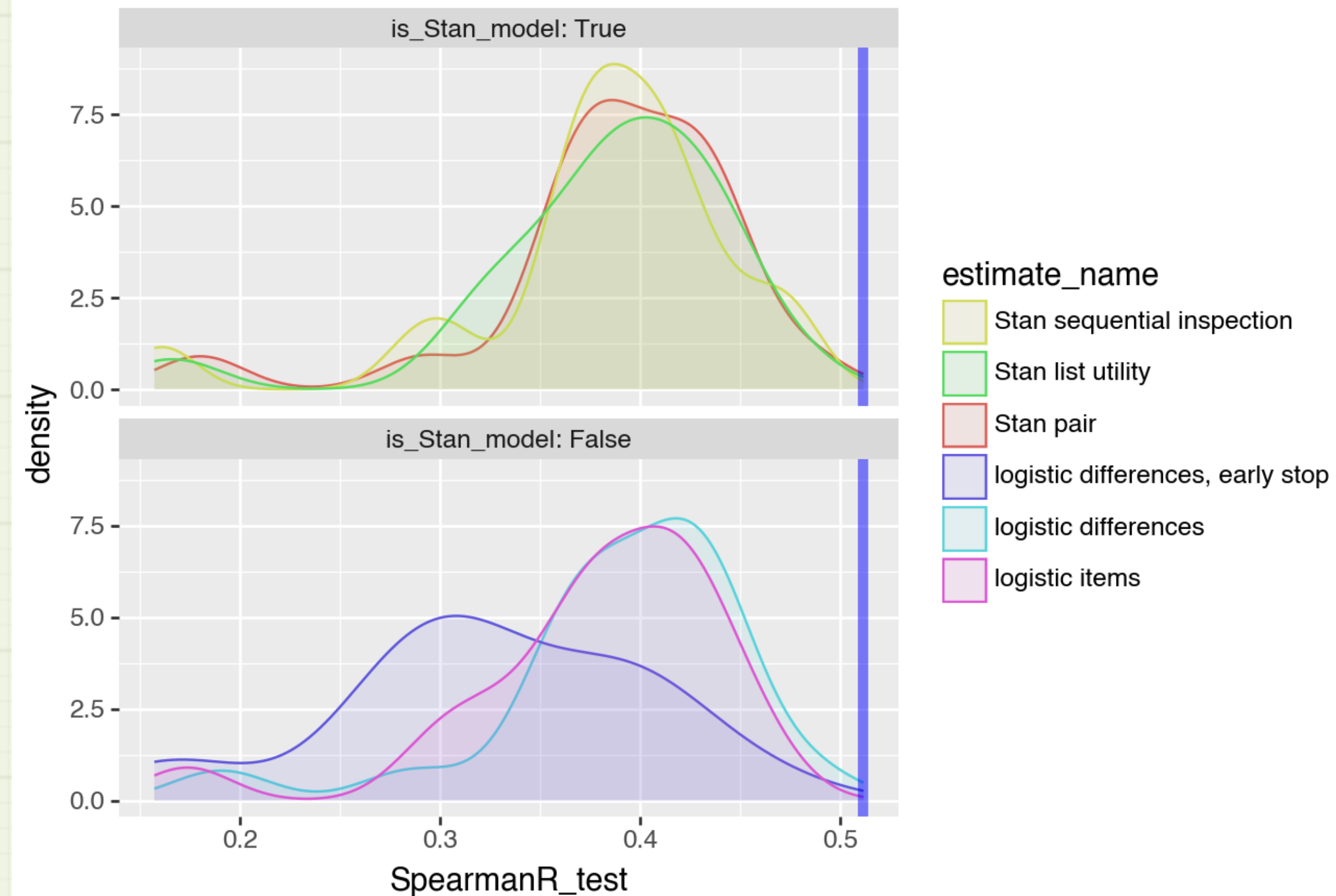
# Comparison of Methods

(100 lists, redrawn 20 times)

observable model quality on test picks  
size 100 observation lists, 20 re-simulations



intrinsic model quality on test items  
size 100 observation lists, 20 re-simulations

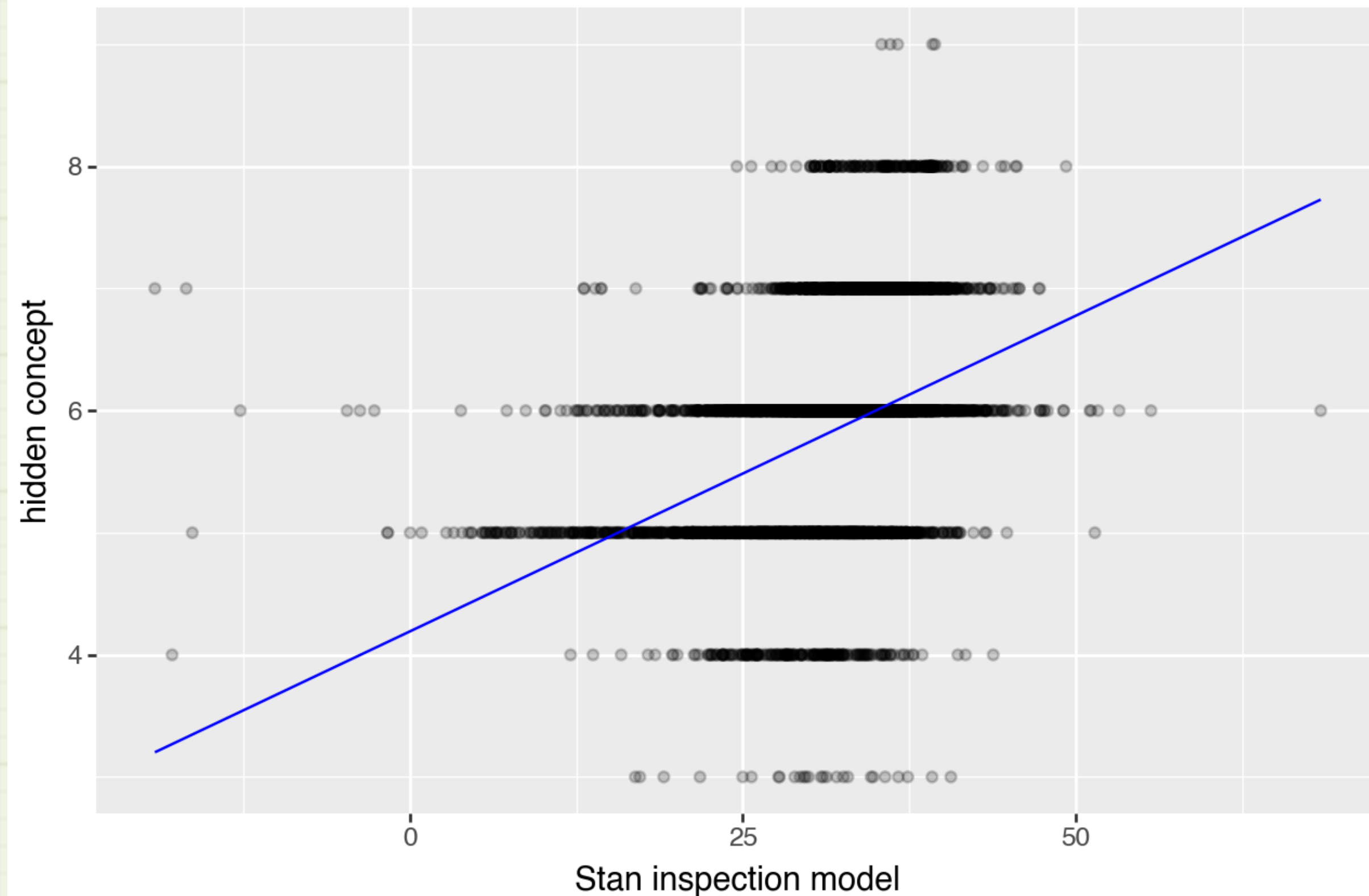


For most models, performance differences go away if we have more data!

# The Effect of More Data

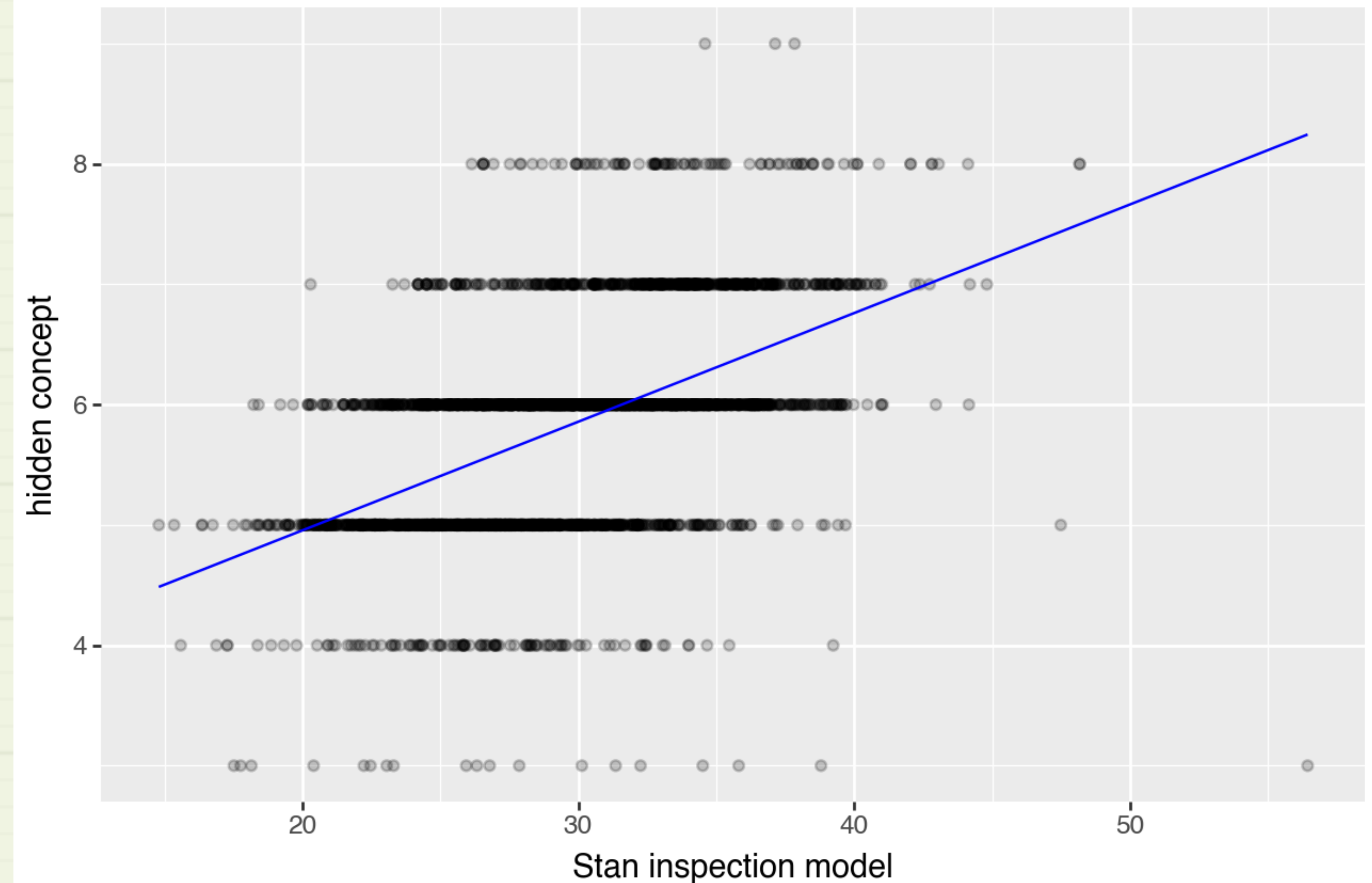
## 100 training lists

uci wine example Stan inspection model Spearman R: 0.43  
(out of sample data)  
original score as a function of recovered evaluation function

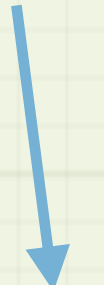


## 1000 training lists

uci wine example Stan inspection model Spearman R: 0.52  
(out of sample data)  
original score as a function of recovered evaluation function



Close to perfect (0.57)!



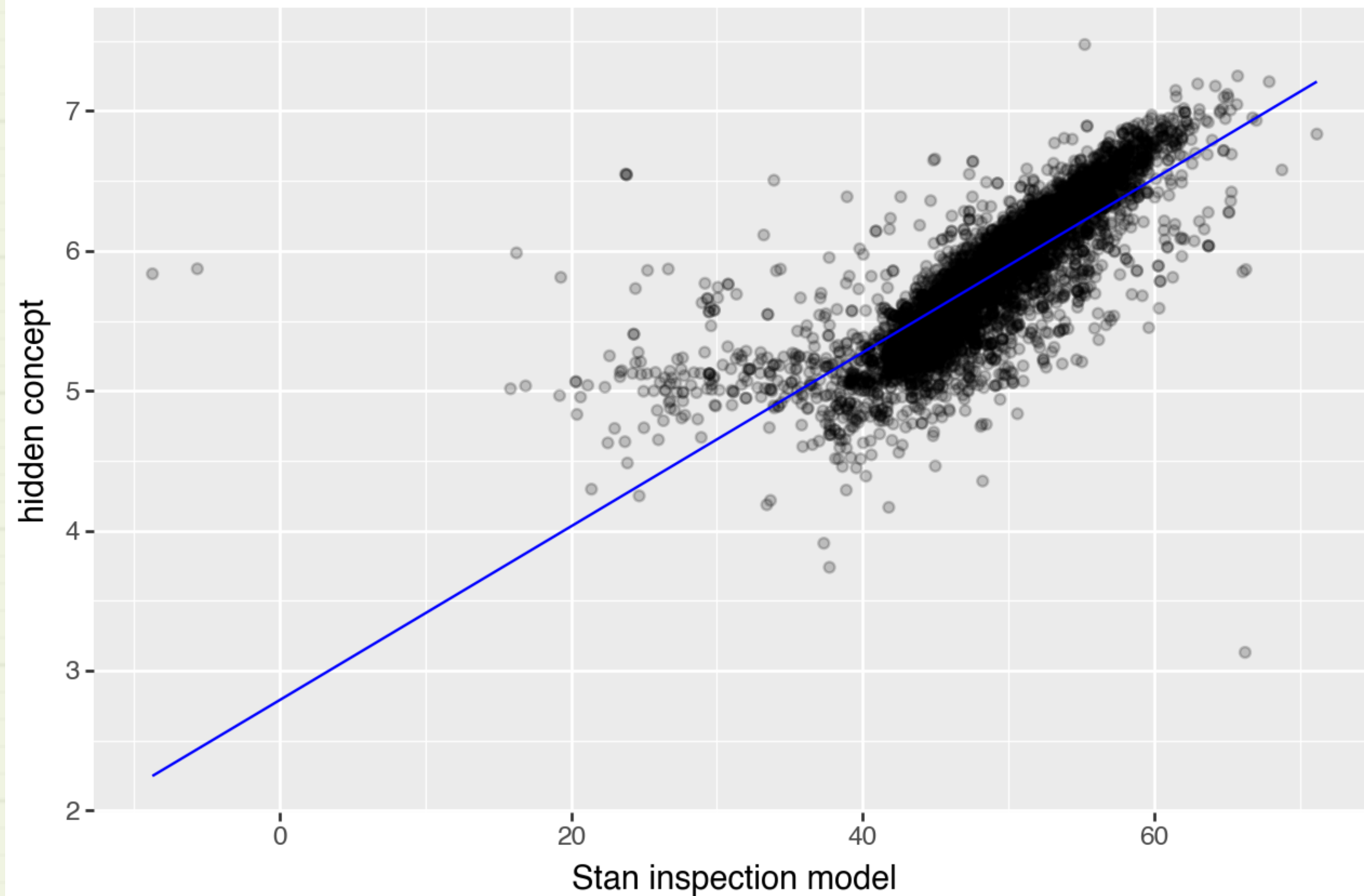
Note: next slide shows we are not hiding behind a noisy evaluation.

# When We Have Correct Model Structure

Replace training data with a linear model of user score (an easier problem).  
Now (modulo noise) the concept is in our modeling space.

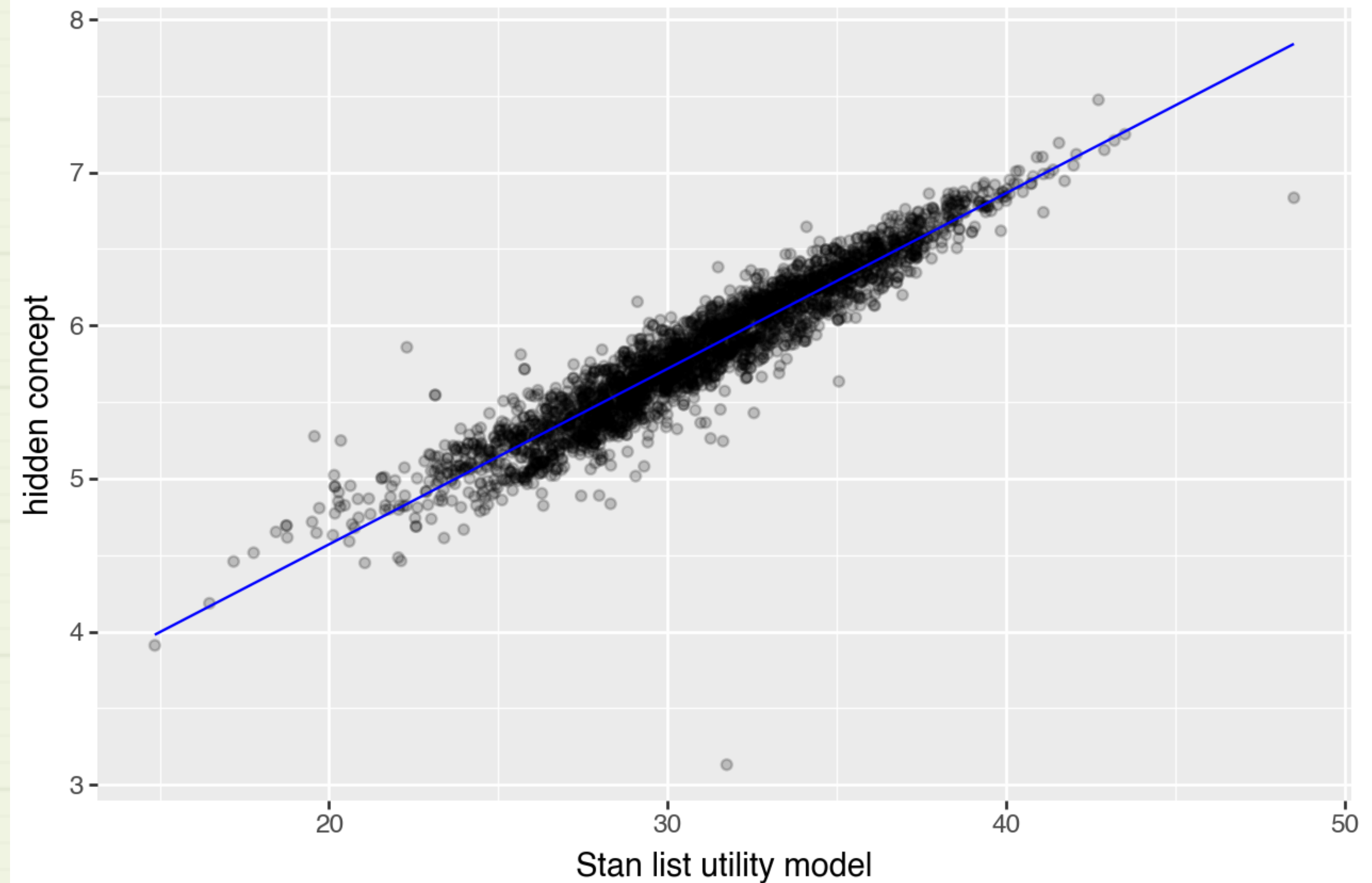
## 100 training lists

uci wine example Stan inspection model Spearman R: 0.87  
(out of sample data)  
original score as a function of recovered evaluation function



## 1000 training lists

uci wine example Stan list utility model Spearman R: 0.95  
(out of sample data)  
original score as a function of recovered evaluation function



Getting better!





# Idea

- Decompose model quality into issues related to model structure and un-censoring the presentation separate.
  - $(\text{Quality of System}) \sim (\text{Quality Presentation Hygiene}) * (\text{Quality of Model Structure})$
  - Write this as a function of training set size!
- Can directly measure of Quality of System (how well it reproduces user picks).
- Try to estimate Presentation Hygiene (in general)
  - Collect your data to build the actual explanatory variable lists.
  - Pick a random (or more appropriate, if possible) linear function over explanatory variables as your pretend utility measure.
  - Generate synthetic picks from above as “synthetic ground truth.”
  - Evaluate Quality Presentation Hygiene

(If quality is routinely high, stop trying to improve this step)
- Maybe  $(\text{Quality of Model Structure}) \sim (\text{Quality of System}) / (\text{Presentation Hygiene})$ 
  - Made (Quality of Model Structure) somewhat observable (so we can try and improve it).

# Observations

- Model structure likely more important than the issues of pick list presentation.
- Stan is great for prototyping solution methods and experimenting with how much model structure you wish to capture.
- Logistic regression may take some steps to justify, but often works well and is fast.

# Tools Used

- Stan, Python, and R
  - All code and data shared here:  
<https://github.com/WinVector/Examples/tree/main/rank>
- Example code can work from just selection data
  - Though it would have no base score to compare to.
  - Can compare to proxy goal: ability to reproduce picks.



# Thank you

# Appendix

# Logistic Difference Model

- Trick 1: encode as a difference

- Item 6425 in position 2 picked and item 1569 in position 2 not picked encoded as:

$$x_i = x(\text{item} = 6425, \text{position} = 2) - x(\text{item} = 1569, \text{position} = 0)$$

$$y_i = \text{True}$$

- The subtraction enforces that features have the same interpretation in picks and non-picks. This is needed to have a model we can apply to items outside of lists and not knowing if they are picked or not. It also halves the number of model parameters, presumably making inference more statistically efficient (requiring less data). This is also why we are not using multinomial logistic regression, or multi class classification.
- Problem: creates a data set with only “True” outcomes (can’t run fitter!).
- Trick 2: also encode reversal (in addition to seeing the winner winning, we saw losers lose)
- Add in extra data rows of the form:

$$x_i = -(x(\text{item} = 6425, \text{position} = 2) - x(\text{item} = 1569, \text{position} = 0))$$

$$y_i = \text{False}$$

- Now we have both True and False outcomes, and can try a logistic regression.
- Collect all the above rows as a logistic regression training set (one row for each pair with a different outcome per list).



# Bayesian Reasoning (the ugly slide we skip)

- We apply Bayes' Theorem.

Define:

$$Z_i = 1/P[\text{select}(i) | x_1, \dots, x_k]$$
$$\text{select}(i) = \bigwedge_{j, j \neq i} (\beta \cdot x_i + e_i > \beta \cdot x_j + e_j)$$

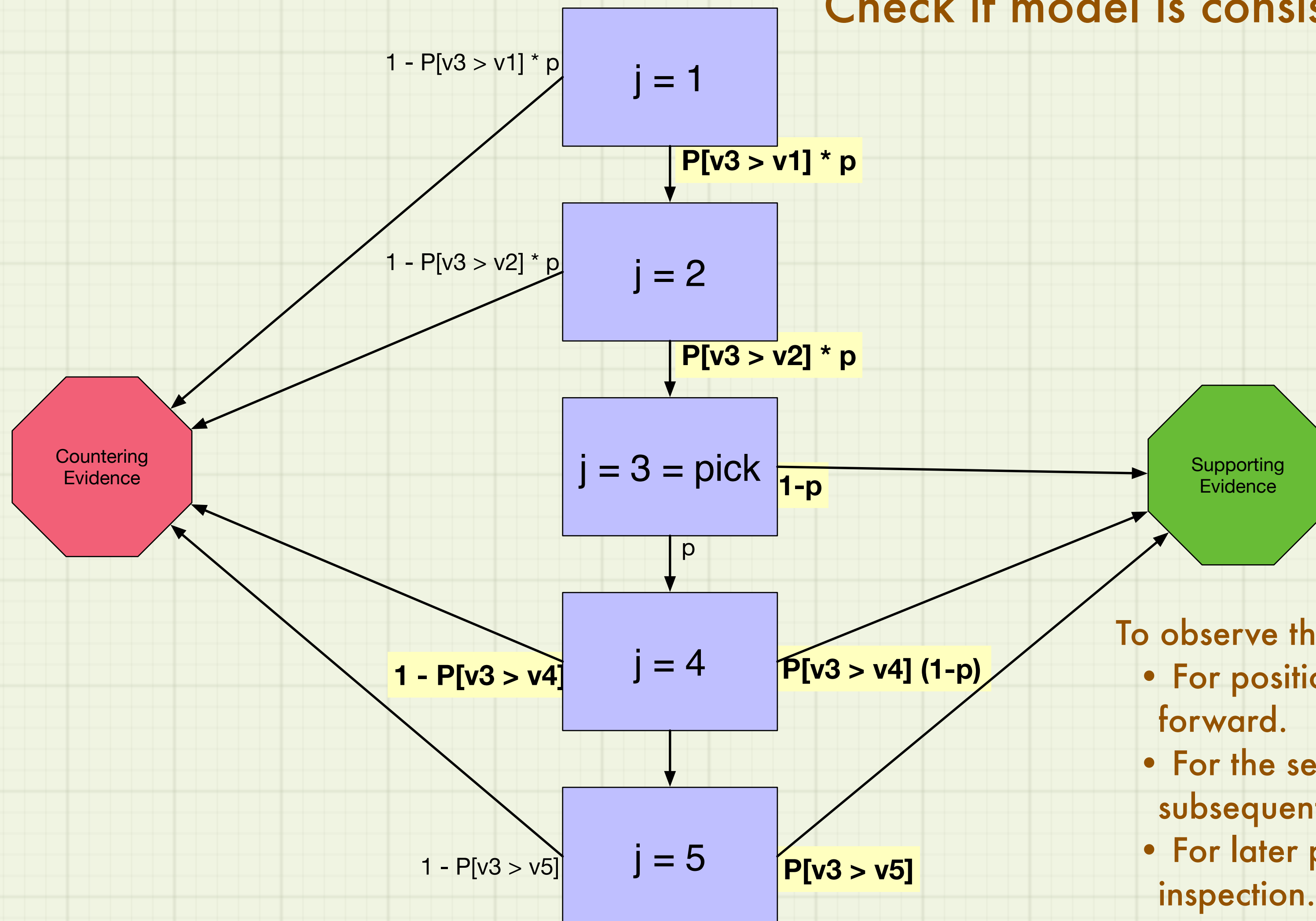
Then:

$$\begin{aligned} P[\beta | \text{select}(i), e_1, \dots, e_k] &= P[\beta | e_1, \dots, e_k] P[\text{select}(i) | \beta, e_1, \dots, e_k] / P[\text{select}(i) | e_1, \dots, e_k] && // \text{Bayes' theorem} \\ &= Z_i P[\beta] P[\bigwedge_{j, j \neq i} (\beta \cdot x_i + e_i > \beta \cdot x_j + e_j) | \beta, \bigwedge_j e_j] && // \text{expanding definitions, remove unused conditions} \\ &= Z_i P[\beta] \prod_{j, j \neq i} P[\beta \cdot x_i + e_i > \beta \cdot x_j + e_j | \beta, \bigwedge_j e_j] && // \text{conditional independence} \\ &= Z_i P[\beta] \prod_{j, j \neq i} P[\beta \cdot x_i + e_i > \beta \cdot x_j + e_j | \beta, e_i, e_j] && // \text{remove unused conditions} \end{aligned}$$

- ( $Z_i$  can be ignored in finding a maximum likelihood  $\beta$ , as it does not depend on  $\beta$ .)

# Sequential Inspection Model

Check if model is consistent with observation of wine 3 winning.



To observe the "saw position 3 picked" we must:

- For positions prior to pick: not see a larger value *and* also continue forward.
- For the selected position either stop continuing forward, or pass subsequent inspections
- For later positions: not see a larger value or also stop forward inspection.

Does not use position indicator columns.