Experiments in Learning to Rank

John Mount
Win Vector LLC
https://www.win-vector.com/
August 26, 2024

Outline

- The problem
    - Some history
- Arrogant Bayesian solution
    - Experiments/Results
- Simpler solutions
- Observations
- Conclusions/recommendations
- Next steps

The Situation

- Users (or user personas) have ***preferences*** (either implicit or explicit and unobserved/shared)
    - 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-personal data.
- We observe user ***behaviors***
    - Typical observation: we presented 5 alternatives and the user purchased one.
- Can we estimate persona preferences from persona behaviors?
    - We can use estimated preferences to plan (a lot more than just predicting future behaviors).

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)

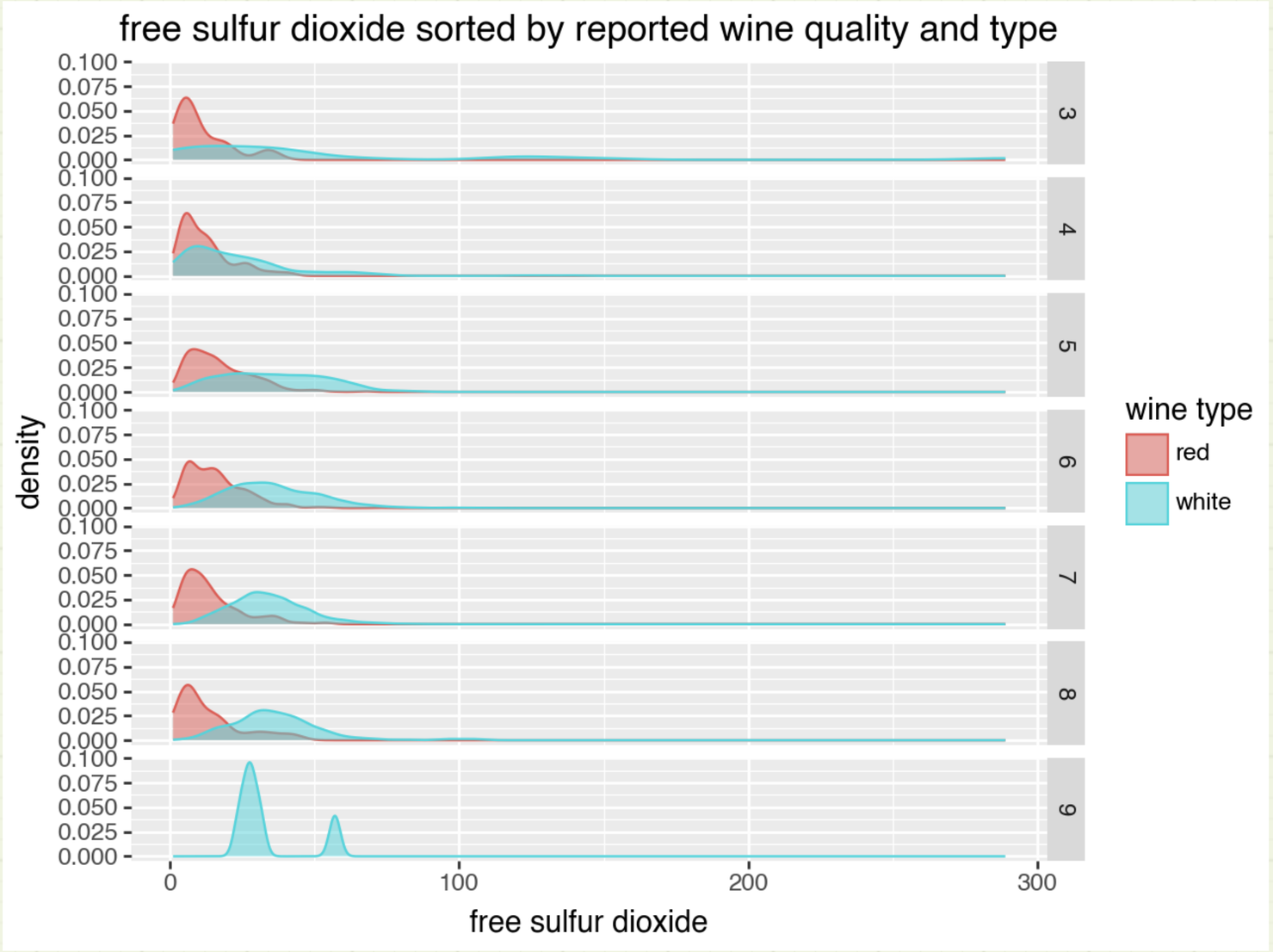
We **wish** they would tell us their numerical valuation of each wine!

The Wine Example

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 instead of domain perceptual).

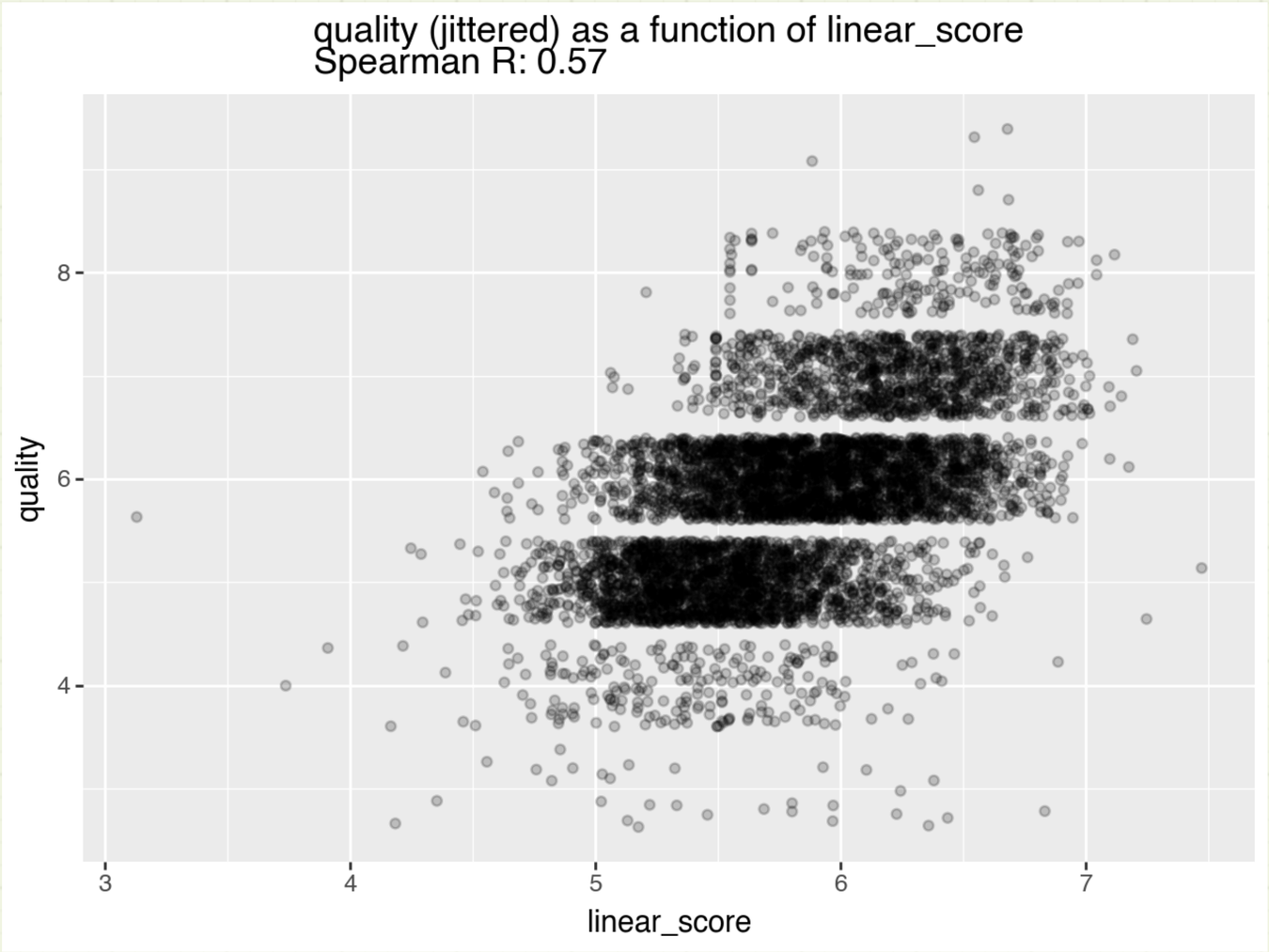


https://archive.ics.uci.edu/dataset/186/wine+quality

The Data we Wish For

User tells us their darn scores!

With 100 panels (lists or tastings) 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 a linear function of our 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 in earlier presentation positions.
- 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 (influenced by business needs)
    - Ideal statistical procedure would be to present 2 alternatives and force a seletion
        - Conjoint Analysis!
    - Item quality may correlate with position (would require an interaction to be introduced to the model).

My Intended Points

- The critical concern is having the right model structure
    - Must approximate the data generation process.
- The data presentation is statistical censoring
    - We can remove the censoring, but that doesn't change if we have the right or wrong model.
    - The censoring increases the required amount of data
- There are several "right ways" to undo the censoring
    - Gives us some useful trade-offs

Leaning to Rank History

- 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
    - Econometrics has its own methods
    - Claims that logistic regression does not work
    - *Tons* of ink spilled on “what is the right way to undo the pick censorship?” (delaying working on the actual problem)
    - …
- Issues such as “point-wise” (each comparison is an event) versus “list-wise” (each presented list is an event) dominate problem design.
- Huge emphasis on efficiency of calculation.
    - Trade-off may be different a quarter of a century later.
- Huge survey: https://en.wikipedia.org/wiki/Learning_to_rank
    - “Bill Cooper proposed logistic regression for the same purpose in 1992 [19] and used it with his Berkeley research group to train a successful ranking function for TREC. Manning et al. [40] suggest that these early works achieved limited results in their time due to little available training data and poor machine learning techniques.”

A Bayesian Solution

- Suppose the user has a hidden valuation parameter β 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, …, 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.
- Let's introduce probabilities by modeling the user as picking the i such that this expression is maximized.
    - This formulation differs from the standard logistic solution in that we are not assuming a link function and error-rate, but instead a value denominated noise process.

Bayesian Reasoning

- We apply Bayes' Theorem.

Define:

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

Then:

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

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

- The point is: each of the checks become independent (the probabilities can be written as a product) once we know e_i for the picked index i. So if explicitly draw e_i and β in our simulation, we can exploit the independence. We do not worry about the e_j, as they appear only once- so can't carry conditioning information between terms. We will call this formulation the list-wise model.

  - Or: I am going to type stuff into Stan, and not feel overly bad about it.

The Pick Data

	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

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

	encoding_0	item_id_0	encoding_1	item_id_1	encoding_2	item_id_2	encoding_3	item_id_3	encoding_4	item_id_4
0	2	6425	1	1754	0	1569	3	2780	4	2646
1	0	4390	1	2031	2	2692	3	4416	4	1913
2	0	599	1	1808	2	64	3	59	4	1671
3	3	1819	1	2324	2	5815	0	1392	4	4567
4	2	3610	1	6283	0	2063	3	2085	4	5610
5	0	2010	1	1465	2	6388	3	25	4	420
6	0	5903	1	1374	2	312	3	926	4	5467
7	0	5194	1	3651	2	1494	3	1749	4	5865
8	0	5946	1	4527	2	5988	3	3021	4	4821
9	0	6469	1	6044	2	2787	3	5786	4	3709

Feature Encoding

$$x(item = 6423, position = 2) =$$

	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.

List-Wise Observations

Outcome (y):

Encodes as:

[0, 0, 0, 1, 0]

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

Stan (“list-wise”) (the model people want)

```
...
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 Observations

Outcome (y):

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 3 is tasted 4 times (all other wines tasted once).
Not what anyone wants, but easiest for most modeling systems.

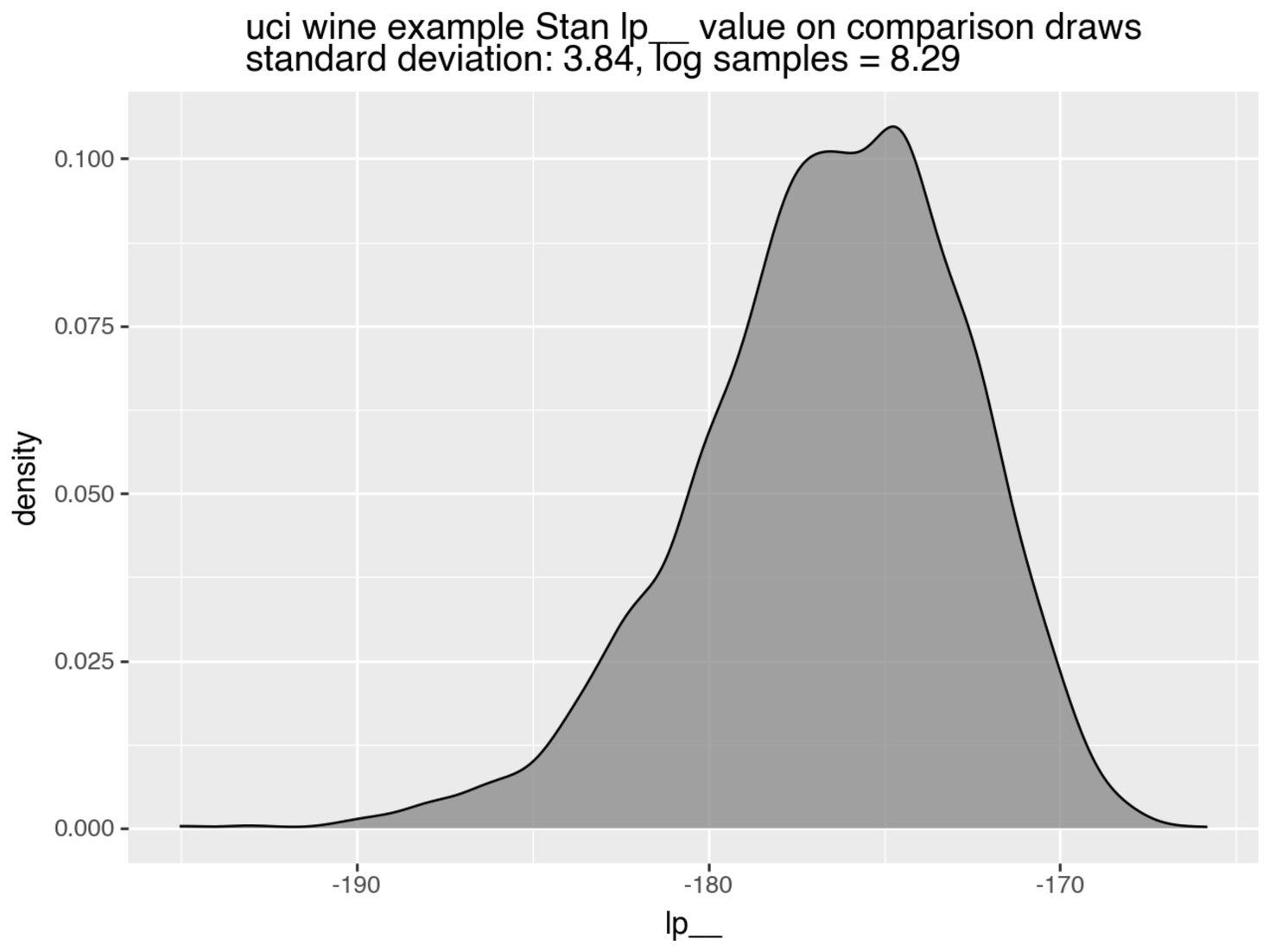
Encodes as (independent events):

$$score_3 > score_0$$
$$score_3 > score_1$$
$$score_3 > score_2$$
$$score_3 > score_4$$

Stan (“point-wise”) (the model people don’t want)

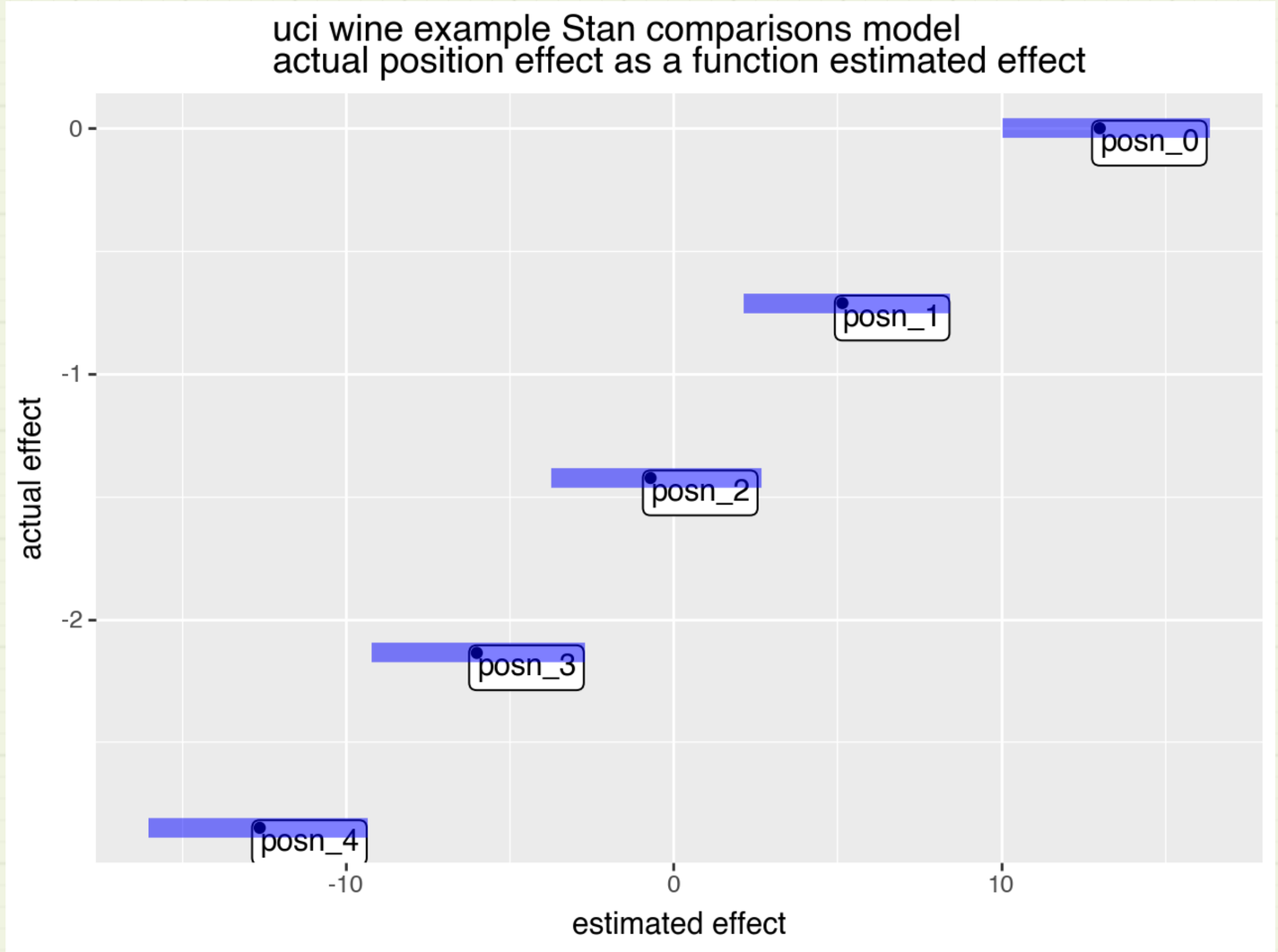
```
...
transformed parameters {
  ...
  expect_picked = x_picked * beta;          // modeled expected score of picked item
  expect_passed_1 = x_passed_1 * beta;      // modeled expected score of passed item
  expect_passed_2 = x_passed_2 * beta;      // modeled expected score of passed item
  expect_passed_3 = 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);
    // log probability of observed ordering as a function of parameters
  target += normal_lcdf( 0 | expect_passed_1 - expect_picked, sqrt(2) * 10);
  target += normal_lcdf( 0 | expect_passed_2 - expect_picked, sqrt(2) * 10);
  target += normal_lcdf( 0 | expect_passed_3 - expect_picked, sqrt(2) * 10);
  target += normal_lcdf( 0 | expect_passed_4 - expect_picked, sqrt(2) * 10);
}
```

Safety Inspection



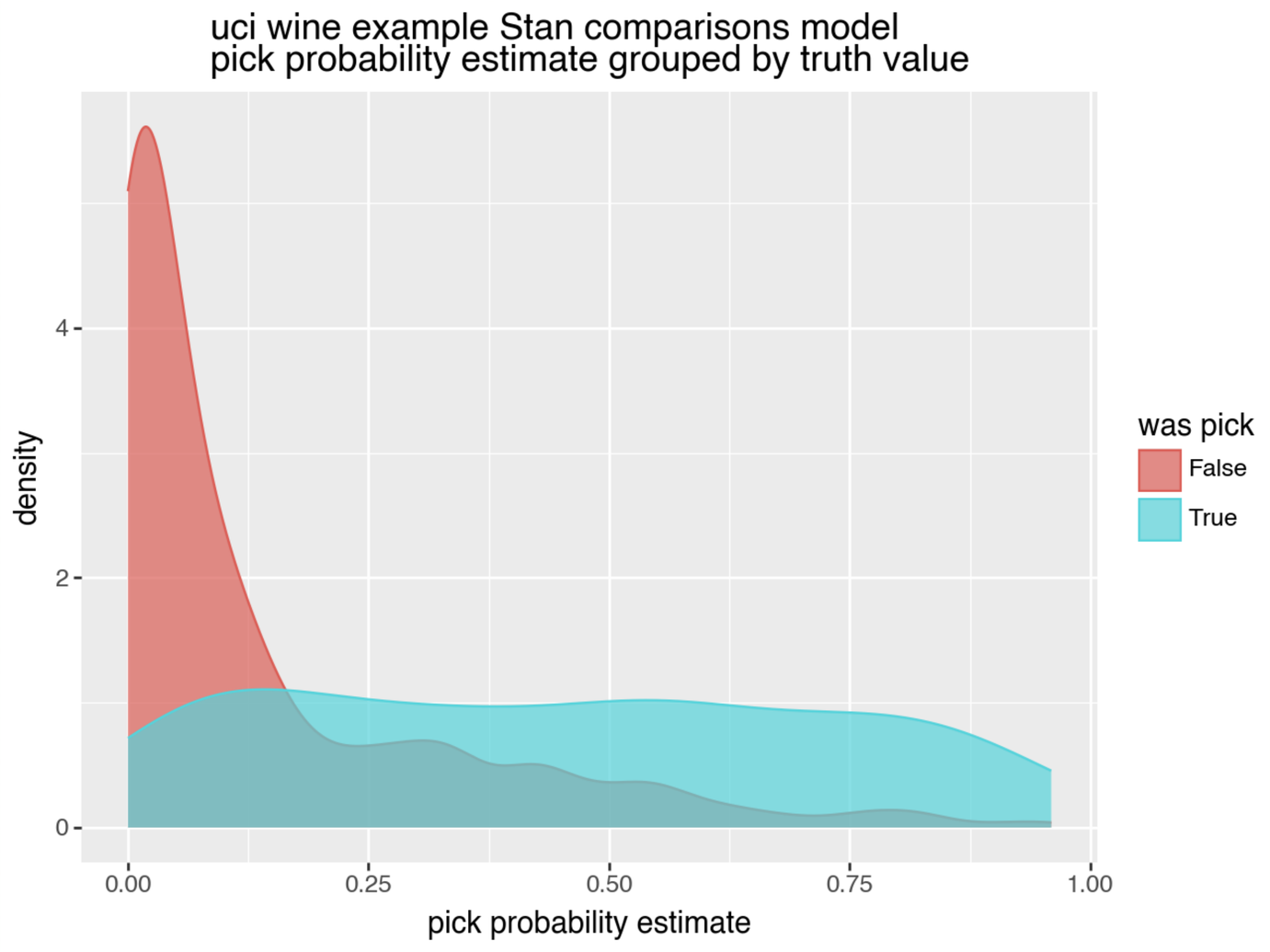
At least during development, you want to check lp__ is unimodal with a standard deviation not too much wider than +- log(n-samples) = log(4000) ~ 8.29. Though I have seen good runs that violate this.

Can Peel off Estimated Position Effect



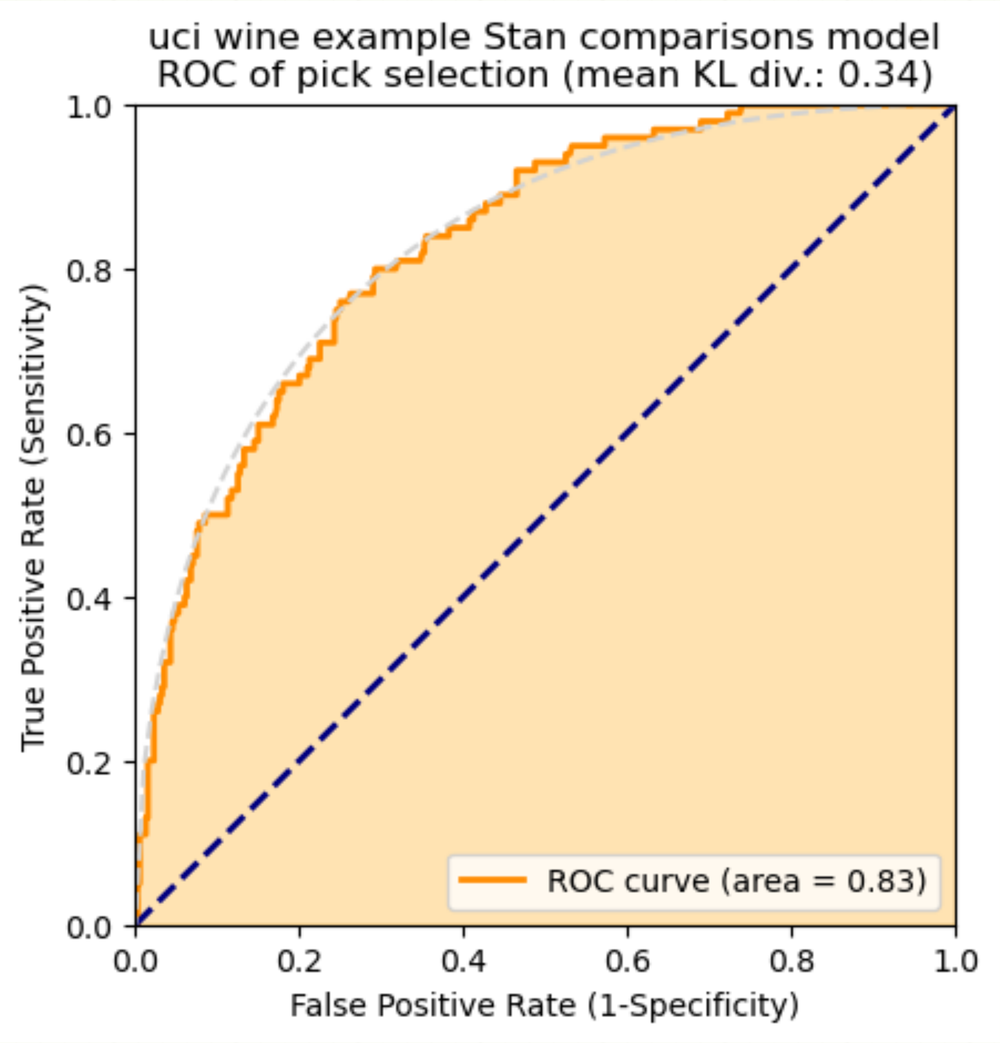
The model is peeling off position as a confounding or nuisance effect in the observed picks (thus affecting recovered scores).

Model Reproduces Observed Picks (out of sample)



Can estimate probabilities by simulating user choices over recovered scores and an (assumed or recovered) noise scale.

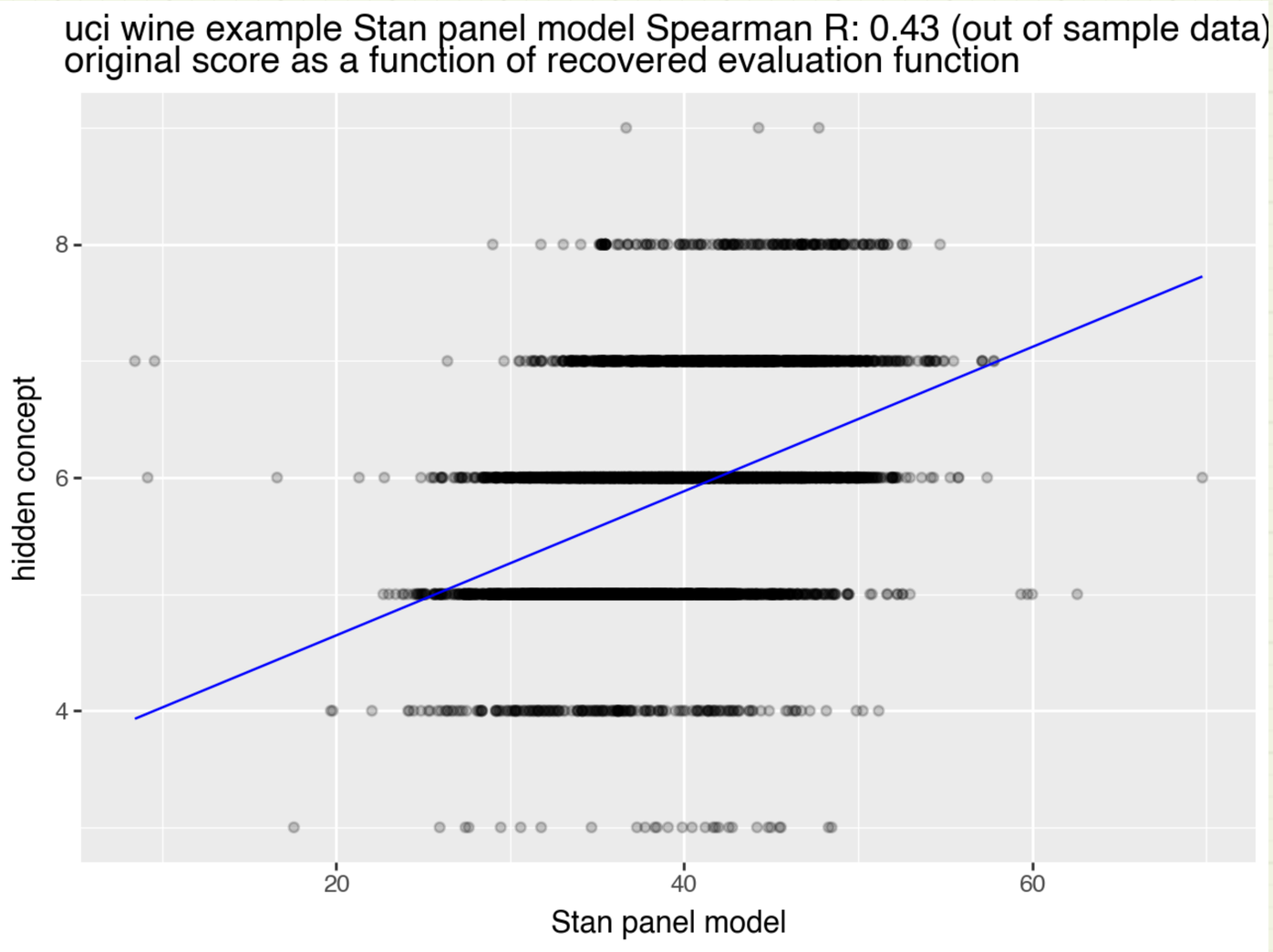
Quality of Reproduced Picks (out of sample)



Sanity check: can the model reproduce the observations? We don't intrinsically care about this as we already have the observations. What we want is for the model to give us estimated per-item value or utility scores.

Can also use a more detailed quality metric such as KL divergence.

Quality of Reproduced Preference Score



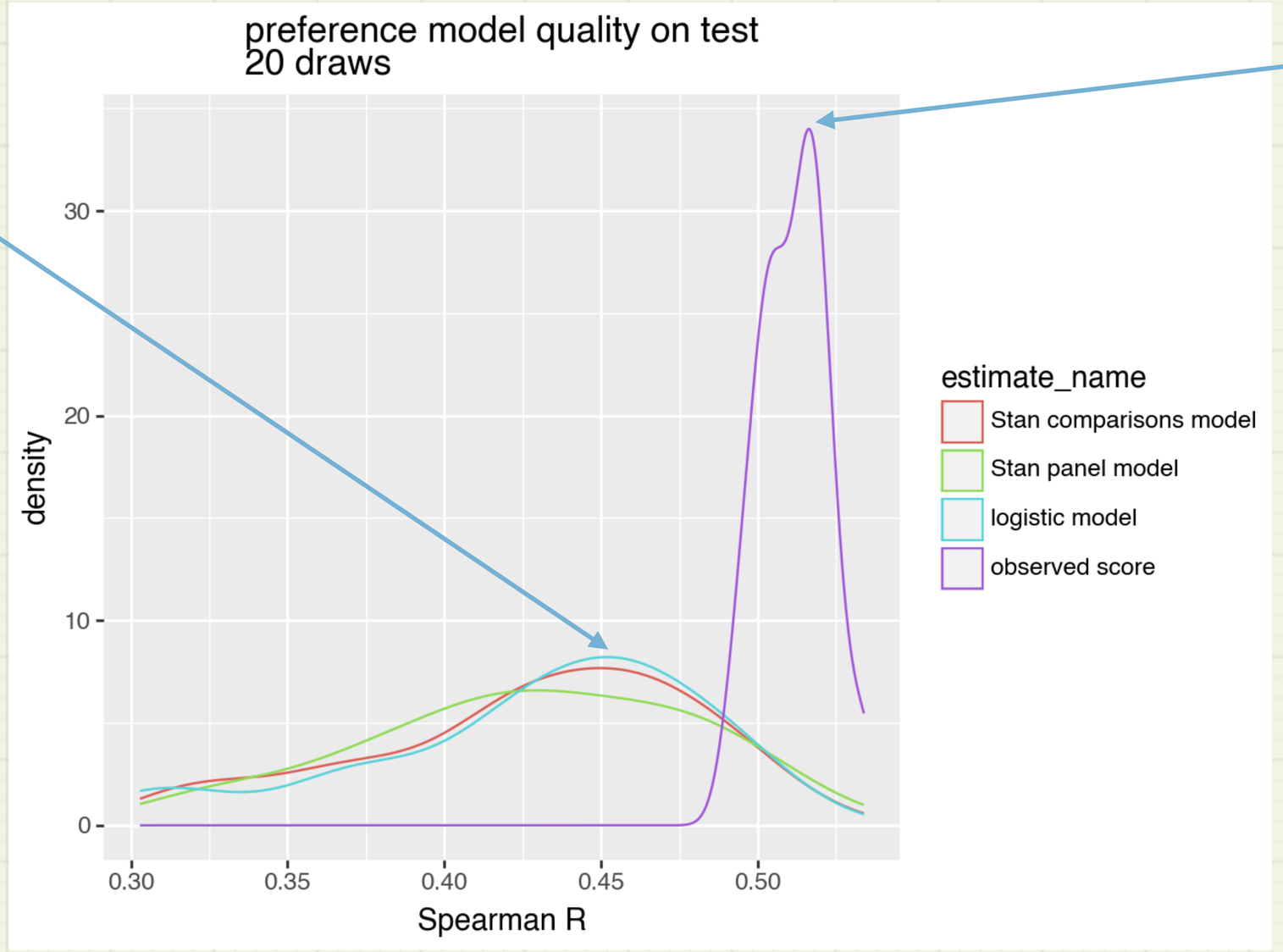
What we actually wanted: recovered estimates of the underlying preference or utility scores. This may not look great, but remember we established 0.57 as best possible Spearman R for a linear model with this set of features. So Spearman 0.43 isn't that bad.

Our formulation works only on order data. So it is shift invariant (no reason to match absolute scores), and estimated scale (or differences) comes from the modeled signal to noise ratio.

Comparison of Methods

(100 panels, redrawn 20 times)

Various "try to infer from picks" methods.
What we can do in practice.
Has been implemented from just observed picks.



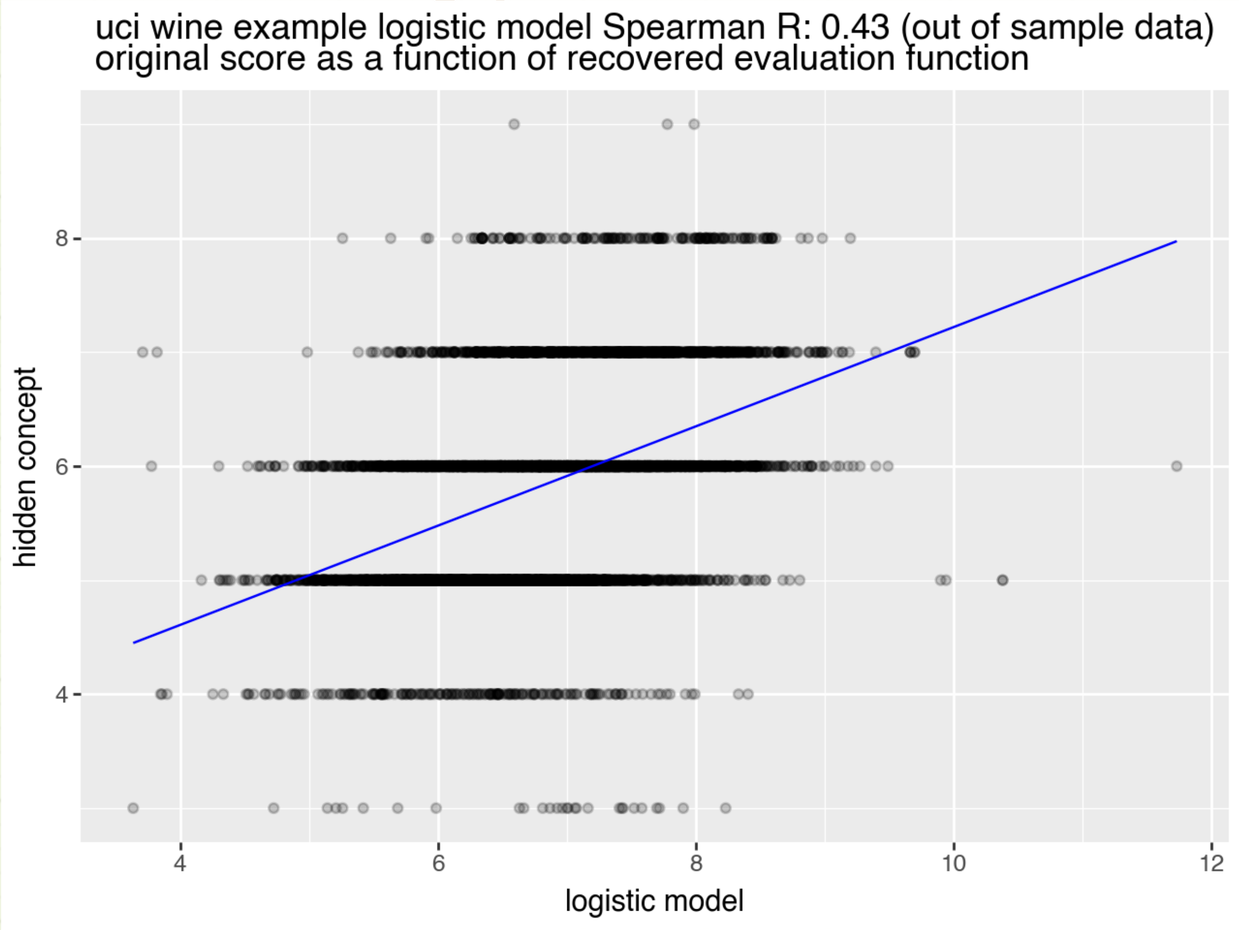
The "we wish users would tell us their scores" situation.
An upper bound on what is possible with the given model structure.
Can't be implemented from pick data.

Gap goes away if we have more panels!

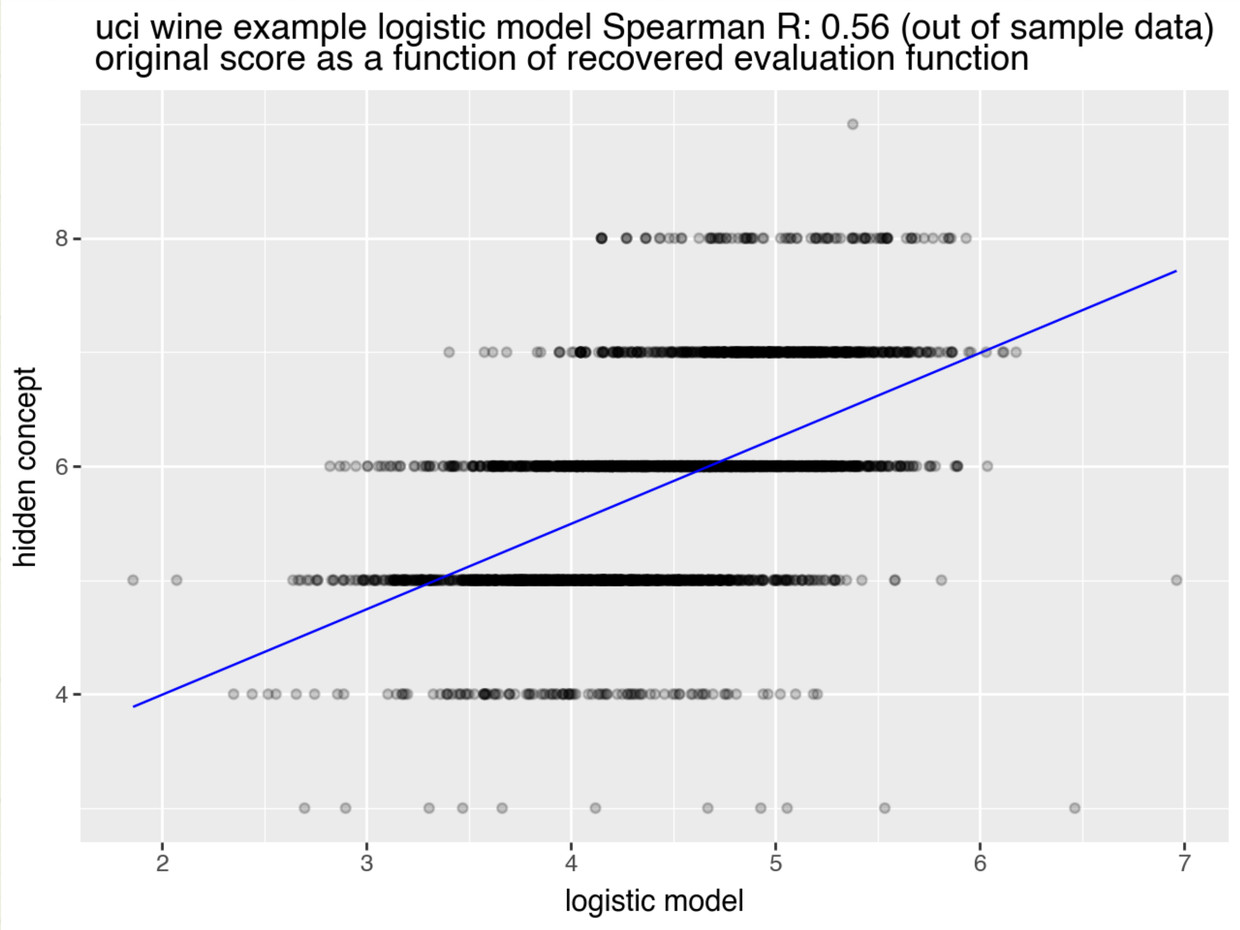
The Effect of More Data

Nearly perfect (0.57)!

100 training panels



1000 training panels



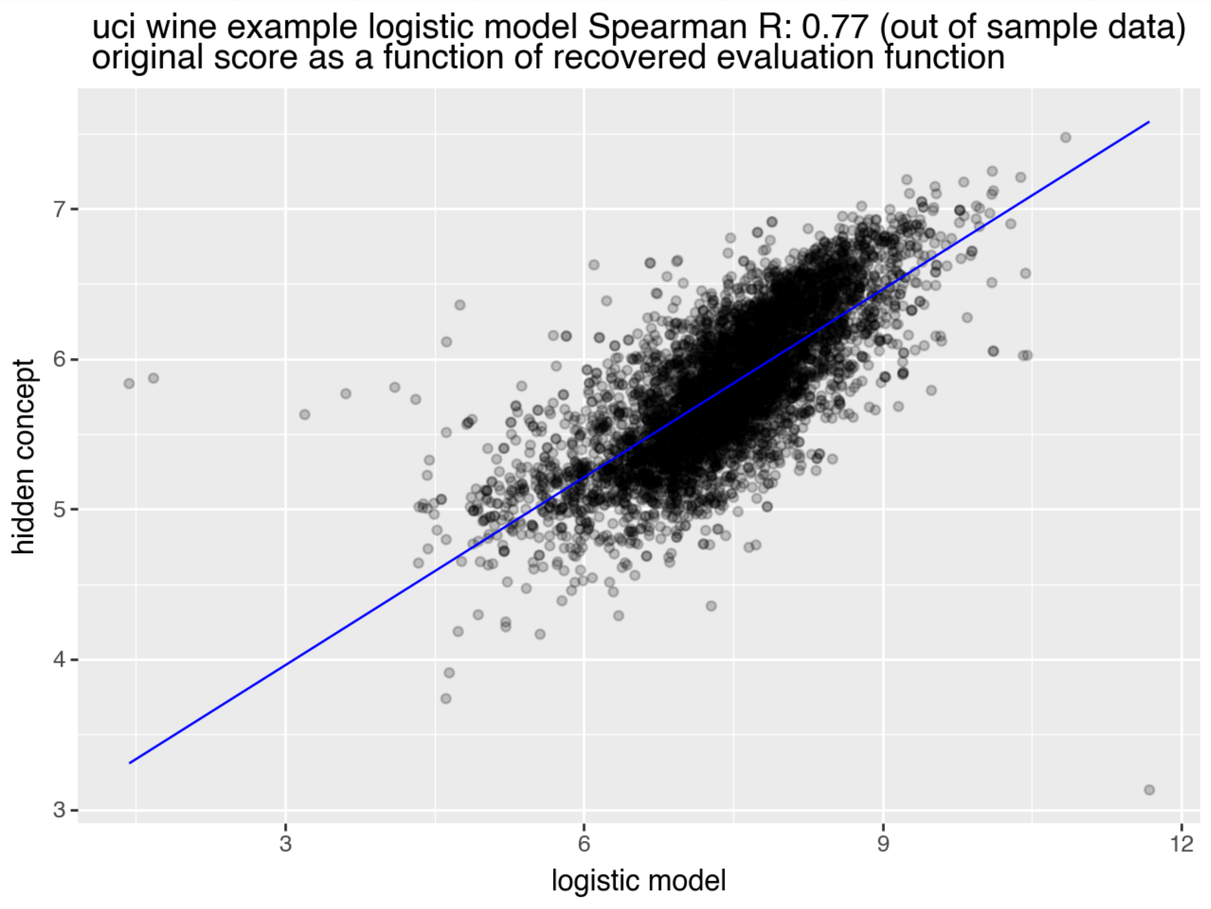
Note: next slide shows we are not hiding behind a noisy evaluation.

When We Have Correct Model Structure

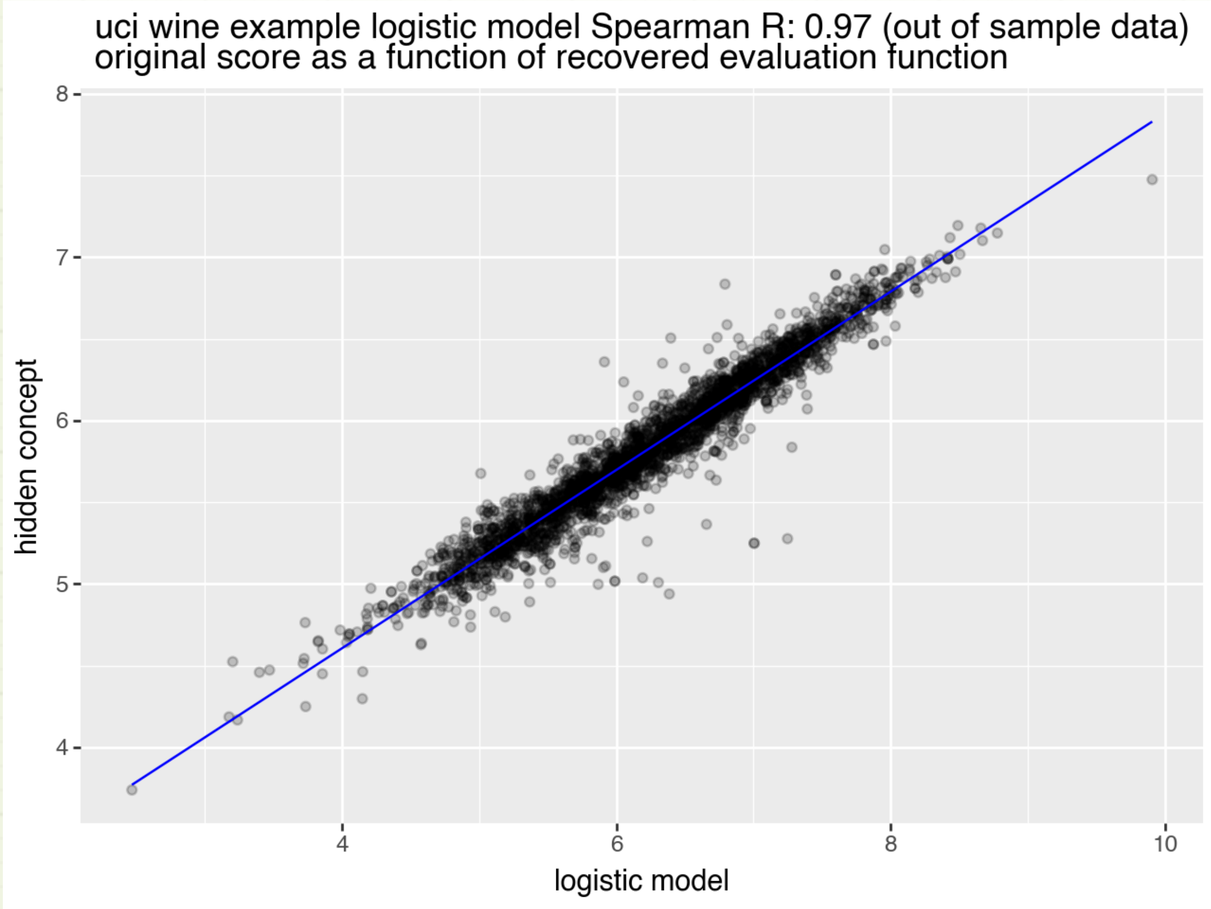
Replace training data with a linear model of user score (an easier problem).
Now (modulo noise) the concept is in our modeling space.

Nearly perfect (1.0)!

100 training panels



1000 training panels



(a sad) Issue

Pick Quality (observable, a mere proxy) Doesn't Track Recovered Preference Quality (unobservable, and the goal)

	example_name	estimate_name	SpearmanR_all	SpearmanR_test	pick_auc	mean pick KL divergence	training panels	test panels	data_size	test_size
0	uci wine example	Stan panel model	0.429198	0.426176	0.822612	0.336419	100	100	6497	6013
1	uci wine example	logistic model	0.434607	0.431649	0.830313	0.417672	100	100	6497	6013
2	uci wine example	Stan comparisons model	0.436790	0.433734	0.829363	0.341213	100	100	6497	6013
3	uci wine example	observed score	0.551482	0.549029	0.829688	0.425096	100	100	6497	6013

	example_name	estimate_name	SpearmanR_all	SpearmanR_test	pick_auc	mean pick KL divergence	training panels	test panels	data_size	test_size
0	uci wine example	Stan panel model	0.554212	0.565128	0.854821	0.304847	1000	1000	6497	3000
1	uci wine example	logistic model	0.552739	0.564960	0.849689	0.415712	1000	1000	6497	3000
2	uci wine example	Stan comparisons model	0.552703	0.564901	0.854053	0.306262	1000	1000	6497	3000
3	uci wine example	observed score	0.570074	0.579847	0.851171	0.424418	1000	1000	6497	3000

Not a big issue if your goal is to predict picks on new data.

Issues with Error

- May be able to post-hoc estimate degree of attenuation due to rank induced data censorship by running synthetic linearized data through the system (as we did earlier).
- Model error is at least 4 components:
    - Wrong structure
        - Feature and model engineering, may not always be able to stay with a linear structure.
    - Censorship
        - Seeing picks instead of scores. The topic of this note.
    - Noise
    - Small sample size

The Extra Solution

- A logistic regression solution was included in the earlier summary
    - Maybe logistic regression isn’t so bad with a good encoding.

Our Logistic Encoding

- Trick 1: encode as a difference
  - Item 6425 in position 2 picked and item 1569 in position 2 not picked encoded as:

$$x_i = x(item = 6423, position = 2) - x(item = 1569, position = 0)$$
$$y_i = True$$

  - 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 panels 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).
  - Problem: creates a data set with only “True” outcomes (can’t run fitter!).
- Trick 2: also encode reversal
  - Add in extra data rows of the form:

$$x_i = -\left(x(item = 6423, position = 2) - x(item = 1569, position = 0)\right)$$
$$y_i = False$$

  - 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).
- As we saw, this works.
  - May not be the “industry standard” logistic encoding.

Observations

- All 3 methods perform about as well as each other
    - With correct or incorrect model structure
- The path to a high quality fit is
    - Correct model structure
    - More data
    - And *apparently not* over-worrying on ranking technique.
        - Some non-linear structure can expressed in Stan.
        - Logistic regression has to be linear (after feature engineering).
    - Though some clever ideas can still be incorporated (such as training no examples past the winner).

Conclusions

- Problem structure likely more important than the issues of pick list presentation.
- Stan may supply bounds on quality for a class of solution methods.
- Logistic regression may take some steps to justify, but works well and is fast.

Tools Used

- Python and Stan
  - 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 score: ability to reproduce picks.

Next Steps

- See if the 3 solutions (Stan list-wise, Stan point-wise, logistic regression) behave similarly on data from chosen domains (where we may not know ground truth).
    - Code works from picks alone
        - hidden preferences used only to generate picks and check inference quality
        - can do different length panels by encoding a “never picked” extra indicator variable
- Implement “signal attenuation due to pick data” estimator tool.

Thank you