Prototyping Preference Inference Using Stan

John Mount
Win Vector LLC
https://www.win-vector.com/
September 3, 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 .

Win-Vector LLC

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 list behaviors).

Our Thesis

- Factor overall system performance into the product of:
  - "**Model Structure**" (how well the explanatory variables can reproduce scores).
  - "**Presentation Un-Censoring Hygiene**" (how well we account for the censoring effects of presentation on the problem). We are treating this as a *required user supplied causal structure*.
- To improve: must know where we *actually* are losing quality.
  - *Want* to attribute quality loss to each component separately.
  - Can do this with Stan
    - *Only* try "clever implementations" after we know which assumed causal structure works for us.

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)

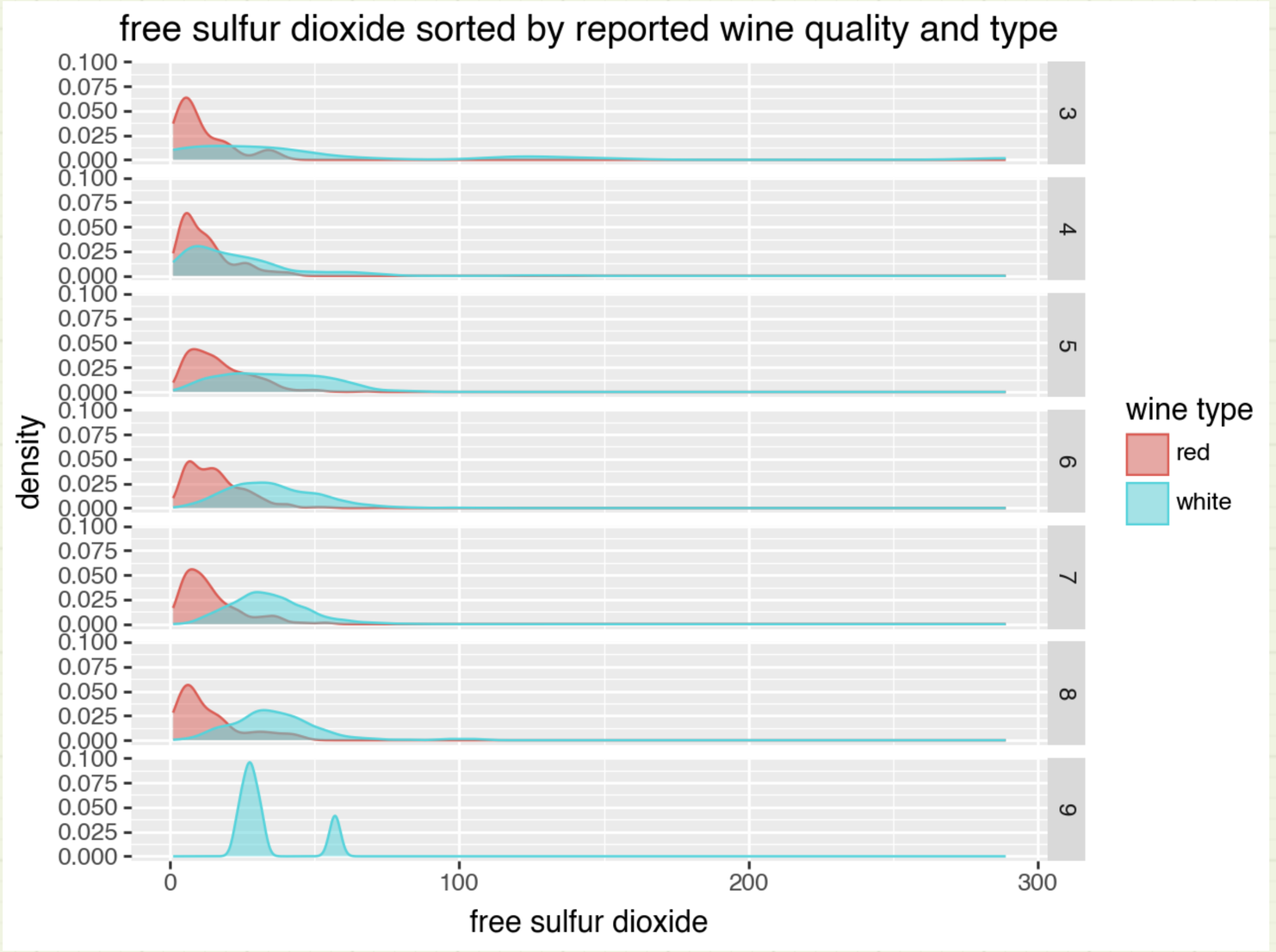
We **wish** they would tell us their numerical valuation of each wine!

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).

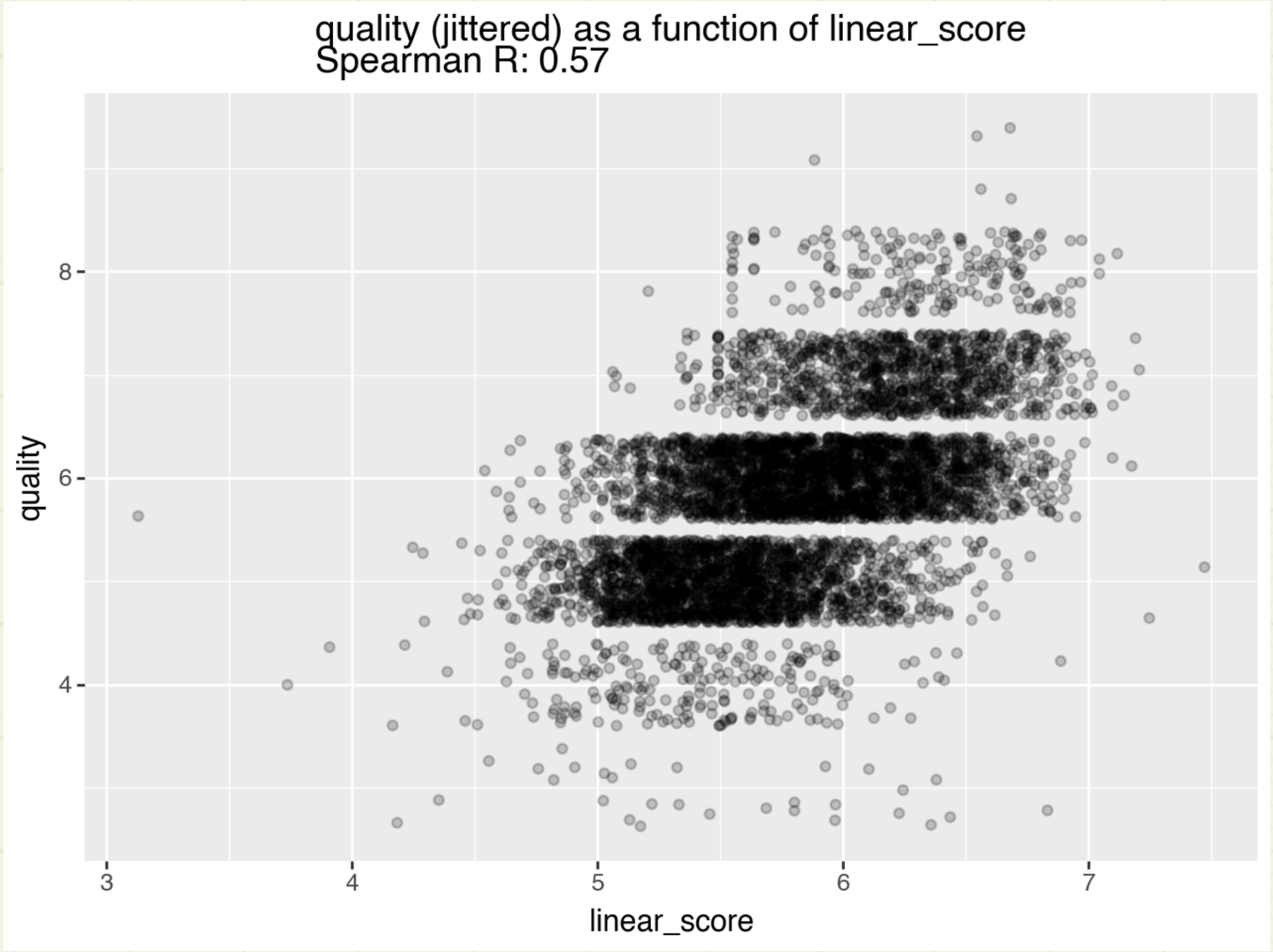


https://archive.ics.uci.edu/dataset/186/wine+quality

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.)

Win-Vector LLC

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 fidelity 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)

Leaning 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
  - Claims that logistic regression does not work
    - “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.”
- May over sell presentation hygine.

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

```
    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 is 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
Stan sequential inspection	list-wise	direct features	utility	yes	as early stopping (modeling probability of continuing)
Stan list utility	list-wise	direct features	utility	yes	as an additional estimated utility
Stan pair utility	pair-wise	direct features	utility	yes	as an additional estimated utility
logistic differences, early stop	pair-wise	per comparison pair difference in feature vectors	probability	yes	as additional variables and as early stopping (nothing after pick used)
logistic differences	pair-wise	per comparison pair difference in feature vectors	probability	yes	as additional variables
logistic items (score model)	item-wise	direct features	probability	**no**	ignored (could add position variables)

Utility Formulation

- 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.
- Variations:
    - Early stopping: They share only the i maximizing $f_\beta(x_i) + e_i$ from i in 1…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.

Pair-Wise Observations (usually not preferred)

Outcome (y):

$$score_{4a} > score_1$$

$$score_{4b} > score_2$$

$$score_{4c} > score_3$$

$$score_{4e} > score_5$$

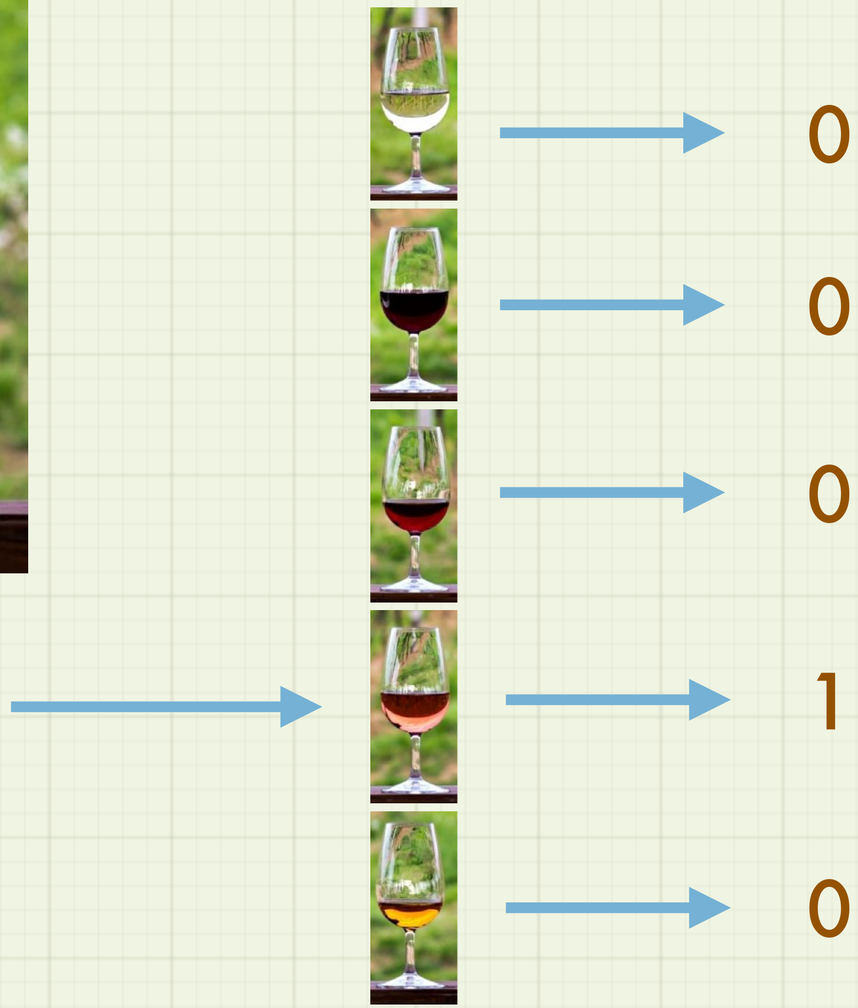
Not what anyone wants (as it may be a weaker inference), but easiest for most modeling systems.

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).

Wine 4 breaks into 4 *similar* wines, *not one* (due to noise term). May obscure scores of other wines as they can't help each other in guessing wine 4's noise term.

Item-Wise Observations (usually not preferred)

Outcome (y):



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

The Pick Data

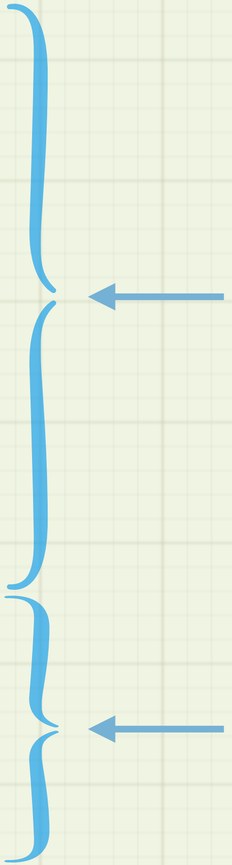
Explanatory features are a combination of table lookup by item_id and presentation facts (i.e. presentation position).

	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

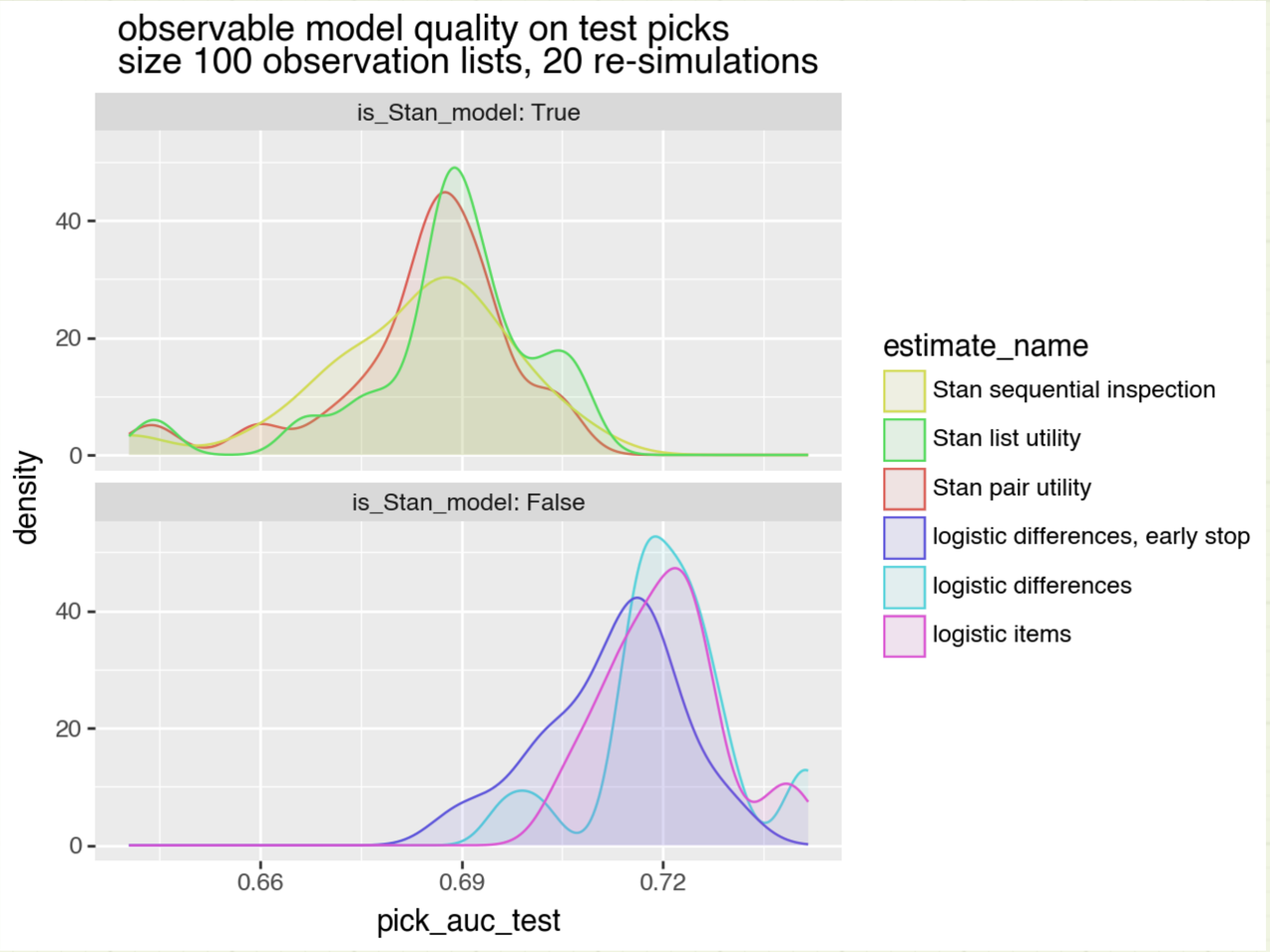
- Exploit linear structure to directly encode differences
  - Explanatory variables: encode comparison pairs.
    - For example: item 6425 in position 2 **picked** and item 1569 in position 2 **not picked** encoded as:

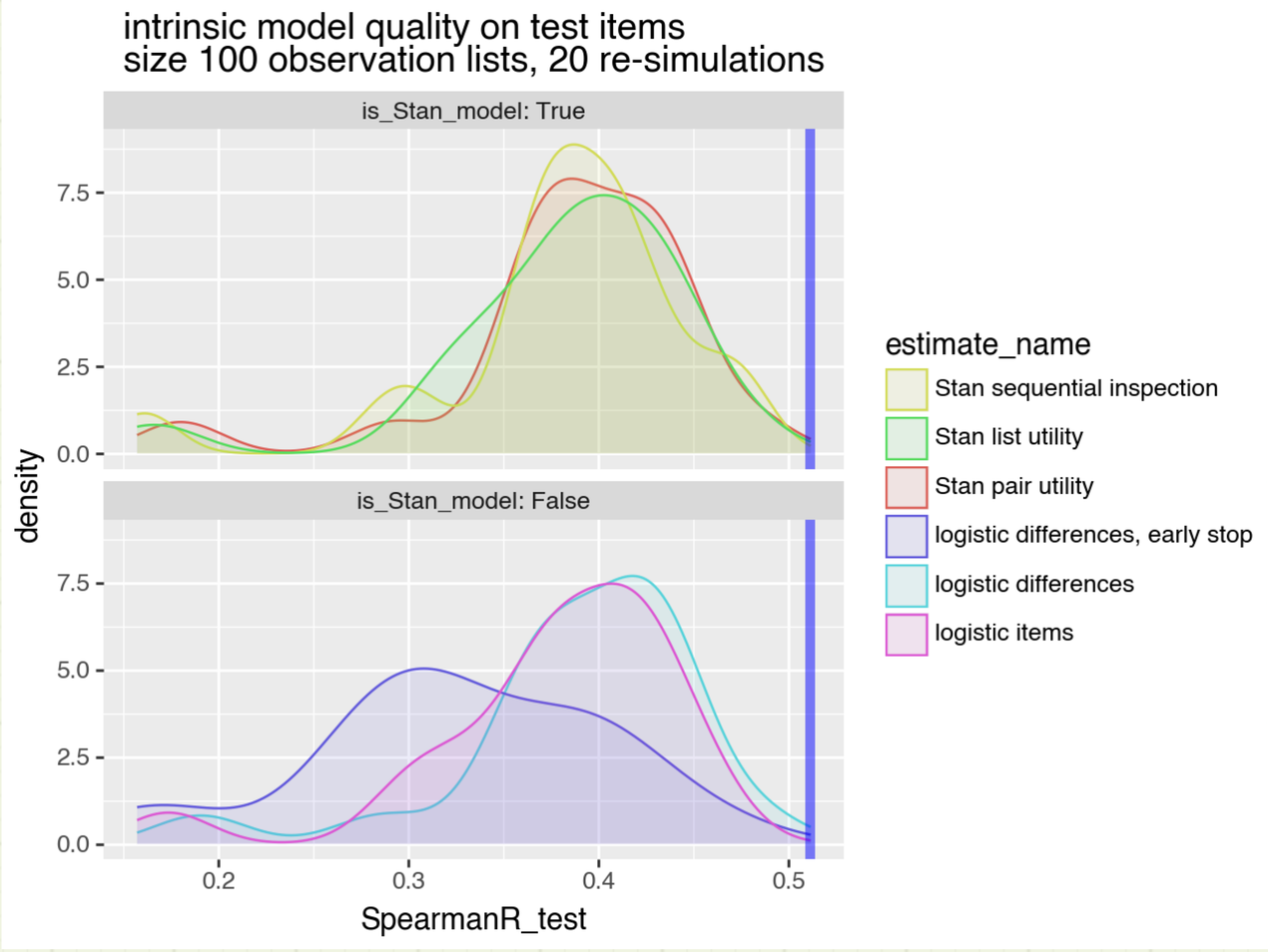
$$x_i = x(item = 6425, position = 2) - x(item = 1569, position = 0)$$

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

Comparison of Methods

(100 lists, redrawn 20 times)





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

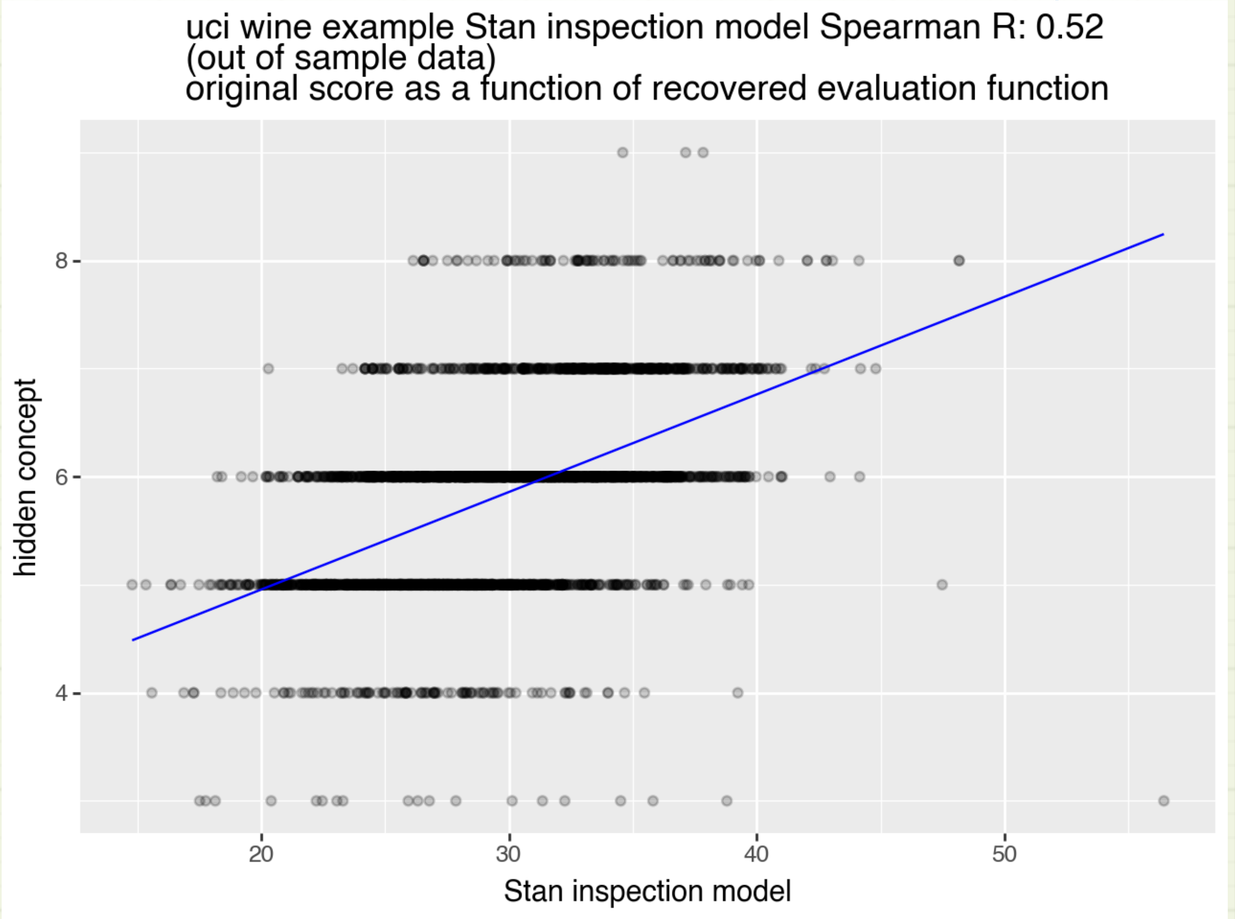
Win-Vector LLC

The Effect of More Data

Close to perfect (0.57)!

100 training lists

1000 training lists



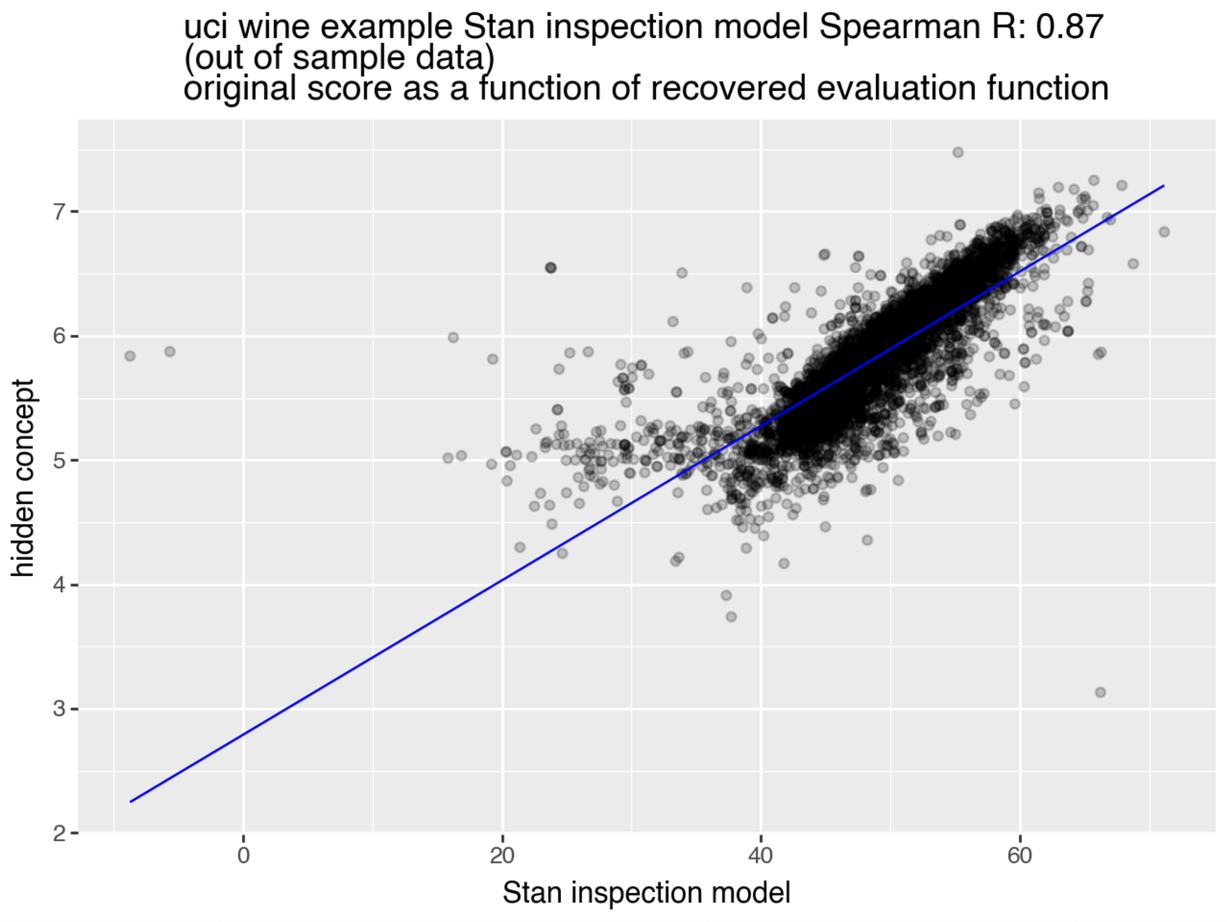
Note: next slide shows we are not hiding behind a noisy evaluation.

When We Have Correct Model Structure

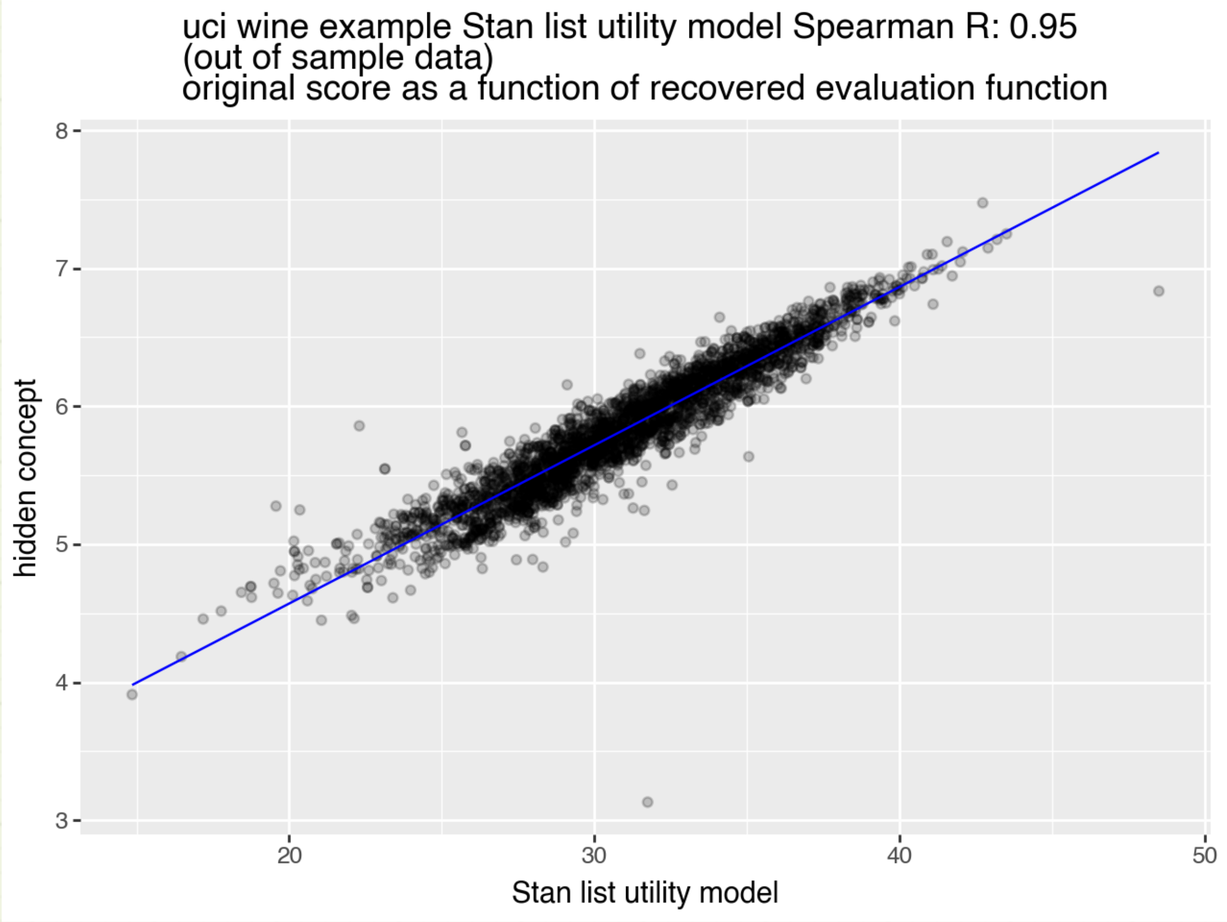
Replace training data with a linear model of user score (an easier problem).
Now (modulo noise) the concept is in our modeling space.

Getting better!

100 training lists

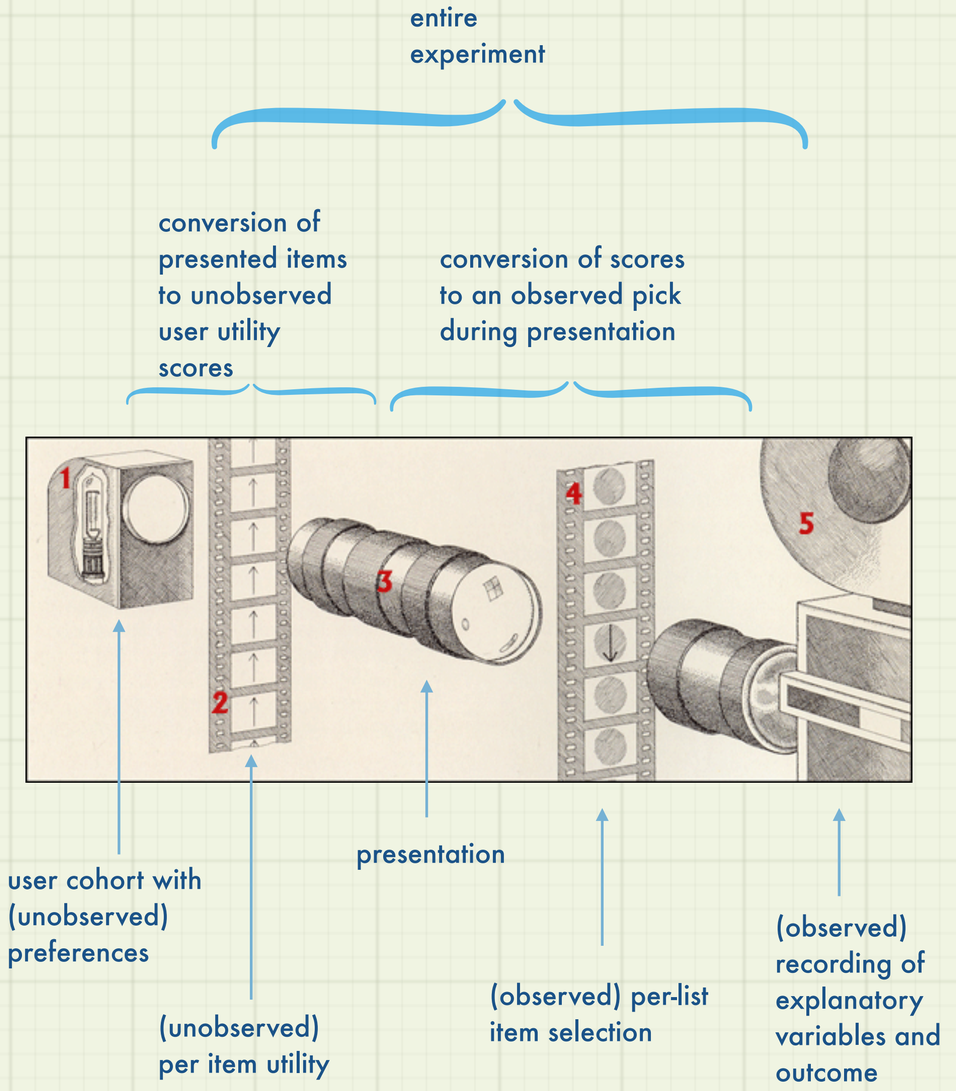


1000 training lists



Factoring the System

- Decompose observation process into two steps
  - System = Utility ○ Pick
    - Utility is modeled as a function of explanatory variables
    - Pick is modeled as a *specified* causal model structure over utility values.
  - (reproducibility of system) ~ (quality of utility model) * (effect of pick process)
- Desired quality of utility model (unfortunately) not directly observed
- Try to back it out:
  - Utility ~ System ○ inverse(Pick)
  - (quality of utility estimate) ~ (reproducibility of system) * (quality of pick process inversion)
  - Try to *independently* estimate quality of utility estimate (how well we model user formation of preference or utility) independently of how pick process inversion (how well we model un-censorship of user picks).
  - Simulate pick process (a chosen causal modeling assumption) separately by observing it over a *chosen* (notional) utility function (as we did with our “linearized preferences” example).
    - Gives us one estimate of “ideal” pick inversion process quality (unfortunately not specialized to our actual data).



Observations

- Model structure likely more important than the presentation hygiene.
    - Only possible to determine if you can measure.
- Stan is great for prototyping solution methods and experimenting with how much model structure and presentation hygiene 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:

$$x_i = x(item = 6425, position = 2) - x(item = 1569, position = 0)$$
$$y_i = True$$

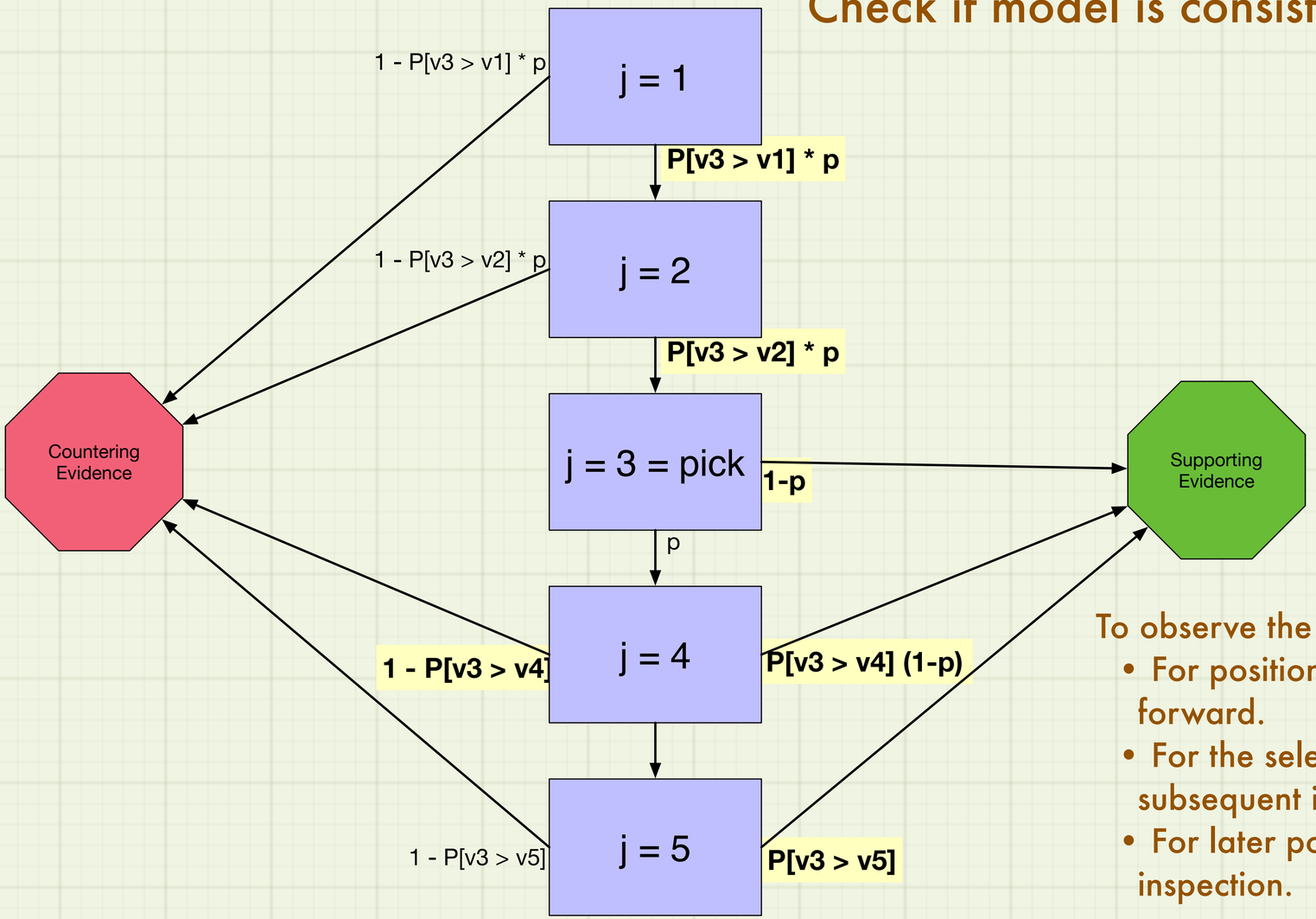
  - The subtraction enforces that features have the same interpretation in picks and non-picks. This is needed to have a model we can apply to items outside of lists and not knowing if they are picked or not. It also halves the number of model parameters, presumably making inference more statistically 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:

$$x_i = -\left(x(item = 6425, position = 2) - x(item = 1569, position = 0)\right)$$
$$y_i = False$$

  - 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).

(Approximate) 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.

Bayesian Reasoning (the ugly slide we skip)

- We apply Bayes' Theorem.

Define:

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

Then:

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

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

The trick in working with Stan:
make things conditionally independent.

Instead of using math such as this, just try copying code from a simulation implementation of an assumed generative process.