Collection of info on how all the things Knewton work

Internal/unpublished documentation

Adaptive behaviors

https://wiley.github.io/dev-knewton/reference/recommendations/

- **review** assumes you should interleave all the LOs of a topic until you demonstrate a need on one of them, and then you'll focus on that one for several consecutive practice questions and instruction.
- **Practice** only interleaves between the LOs at the beginning of each topic to introduce the students to them (our diagnose mode), and then you're pretty likely to stay focused on each LO until you master it or get stuck.
- This difference reflects learning science assumptions that initial learning is better when "focused" or "blocked," while consolidation is better when interleaved.
- The workflows are agnostic of specific completion criteria, except that review workflow has some secret three-correct-in-a-row target completion

How pretest mode works (to not always serve instruction after 2 corrects on an LO):

- pretest is almost always 2 questions, and the new guardrail would be "if you get both of those questions right, do not proceed to teach mode -- instead, return to pretesting another low-mastery LO or move to ASSESS_LO if there aren't low-mastery LOs."
- If you get one right and one wrong in pretest, we'd let you go to teach dependent on your proficiency -- so you might or might not, depending on your previous history on the LO
- We already have this guardrail in pretest mode in most other workflows -- the only real places where we don't, currently, are in the HiMarx workflow (used by Alta calculus and NWP practice) and the HiMarxInstructFirst workflow (used by Alta instruct-first and NWP pre-lecture).
- Our current proposal is to make the change impact all of those uses, since we think it will be a strictly positive change.

Progress

- progress going up on an incorrect answer is generally only possible in the first couple of questions on any given learning objective, and it only goes up a small amount: this is a way of giving the student some small amount of credit for effort
- Knewton's "progress" metric tracks a student's progress toward completing the current assignment. Since Knewton adaptive assignments implement mastery-based learning, "completing" the assignment doesn't mean the student answers a fixed number of questions or gets a raw percentage of the questions they've answered correct. Instead, assignment completion marks the point at which our proficiency model (alta-specific blog that would apply to NWP with branding changed: https://www.knewton.com/blog/mastery/how-does-knewtons-proficiency-model-estimate-student-knowledge-in-alta/) judges that the student has (1) done a minimum amount of work on each learning objective assigned, and (2) has

demonstrated through that work that they are now likely to succeed on further questions on that learning objective and are well-prepared to learn subsequent, postrequisite learning objectives. The progress metric scales from 0-100%, and you can think of it as the product of two terms:

w, the fraction of the minimum work requirement that the student has satisfied. The minimum work requirement per learning objective is one of the things determined by assignment intent. If the minimum work requirement for an assignment is 4 questions, the factor **w** would take the following values as the student completes questions: [0, 25%, 50%, 75%, 100%, 100%, 100%....]

p, the ratio of our proficiency model's estimate of the student's knowledge state to a value we've validated as a meaningful mastery threshold. A high value of p means: we are confident that the student is likely to get future questions on this learning objective correct and is prepared to move on to new material (old efficacy blog to say that "p is meaningful," probably hard to massage this into NWP-friendly format: https://marketing.knewton.com/wp-content/uploads/2018/09/Knewtons-alta-2017-Student-Data-Insights.pdf).

The progress metric surfaced in the product is **w*p.** Examples of how that plays out: if a student seems very good at the material (high **p** value) but has only answered a question or two (low **w** value), they'll see a middling and increasing progress value. When a student starts an assignment, they'll likely see a low but increasing progress value as they complete the minimum required work, increasing the value of **w**, even if they're getting some answers wrong. This is intended to be motivational, and it recognizes that the student is putting in effort toward learning. Note that the form of **w*p** makes it impossible to complete the assignment just by submitting a large number of wrong answers: once the student has passed **w** minimum questions, they must still perform above the **p** mastery threshold! As the student continues to answer questions right and wrong, the proficiency model's p estimate will fluctuate up and down, and the progress bar will move with it. And no matter how poorly the student starts an assignment, it's always possible to demonstrate mastery later: 100% progress can always be achieved through effort and improved accuracy.

Recommendations

- Selecting the next item to recommend as of 20210526 (this may change, esp. re: algo questions): sort by novelty first, but only after restricting the set of modules to the top ~50% best assessing questions - to avoid recommending assessment atoms that are not good when a concept is near exhaustion.
    - This may mean that algorithmic adaptive questions (pool size > 1) will end up repeated more frequently than expected; once you've seen all the best assessing items once, the algo item will be more novel for potentially forever. That's because novelty doesn't decrease more on many consecutive interactions.
    - The "best" assessing items is influenced by the seeded difficulties, if present.
    - A good cartoon picture is "we want an item such that the student with proficiency level X has a Y% chance of getting it right" (if Y --> 0 or Y --> 100, bad assessment!)

Justifications

Internal release notes

Algo questions

- In the CI, an algo question is represented by one Atom, and the only difference between a non-algo and algo question is that an algo question has a pool size larger than 1. That number should be how many times an instance of an algo question can be shown to a learner before it feels repetitive (so this depends on the question: if the instances are very similar, that number should likely be low, if the instances are different enough not to feel samey, the pool size can be higher). Basically, the pool size = how many times Knewton thinks it can recommend an instance of the algo question before it would be a repeat (like with non-algo questions). So, while the pool size depends on the questions themselves, 4 could be a reasonable pool size in many cases, for example.
- API calls: dev.knewton should be up-to-date about how to handle algo questions in calls, so I'd just follow it. Lmk if you have specific questions, though. The key thing is that when sending a graded event for an algo question, it should include the instance hash, which should be different for each instance of an algo question - that's pretty much it.
- For recs, the first bullet point above covers how an algo question would be handled, and for analytics, completing an instance of an algo question would count just like completing a regular question.

For more details, these pretty old docs seem to be still accurate (tho for the API calls, just follow dev.knewton):

https://docs.google.com/document/d/1HSzWbRV2pqtThJUFrUlFrvEYJr9LwCynbdKzcp6o_Bc/edit?usp=sharing

https://drive.google.com/file/d/0B_1gP_iu7Cb1d0FNZ1RZS2xUd3M/view?usp=share_link&resourcekey=0-2yIHK3w72NcCGImvFMYzsw