Thrashing: examining students who fail to master in a timely manner

Joseph E. Beck and Yue Gong

Caution!

- First time I've presented this work
- Isn't appearing in ITS
- (Best analytical work I've done in awhile)
- Interested in your thoughts and extensions
  - Interrupt and disagree

Two main points of this talk

- Students are not doing as well with our tutors as we think they are

- Need to augment the typical mastery learning approach

Simplified notion of ITS mechanism

- Present problem to student
- Student learns a bit from solving the problem
- When student has mastered the topic, stop presenting it (mastery learning)
  - Alternately: keep presenting the topic until the student masters it

What should the ITS do?

- Present one item to the student
- He gets it wrong
- What should we do?

Sounds reasonable

- He gets a second item wrong
- Present another one
  - Hopefully this will help him learn

Still sounds reasonable…

- He gets a third item wrong
- Present him another item…

If it first you don’t succeed…

- He gets 20 items wrong
  - (sounds crazy, but have observed such cases in the data base)

- Present the 21st item with the hope that…

If it first you don't succeed…

- He gets 20 items wrong
  - (sounds crazy, but have observed such cases in the data base)

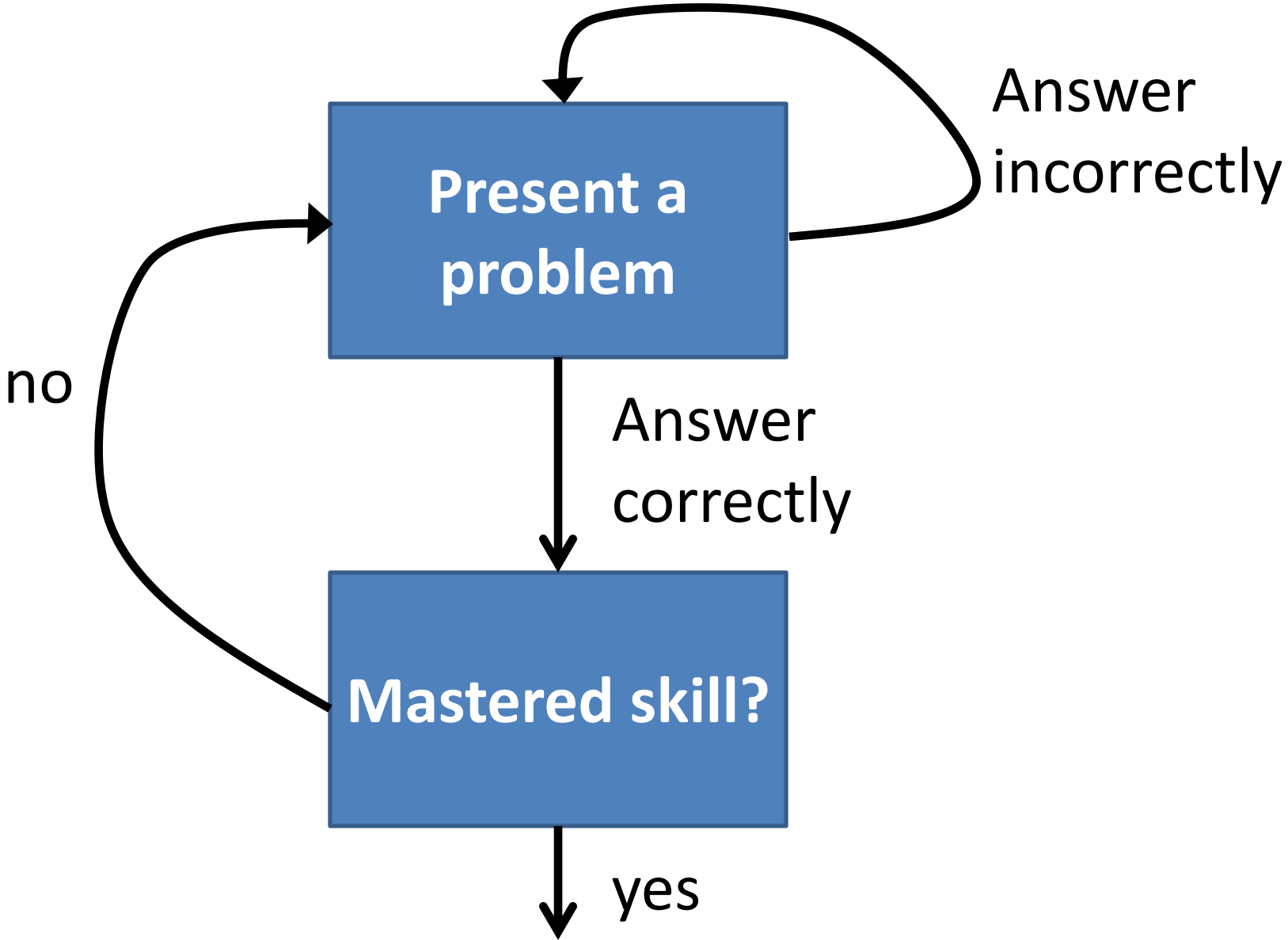
- Present the 21st item with the hope that…
- Wait! What?
  - Sounds even crazier
  - The definition of insanity is doing the same thing repeatedly and hoping for a different result

Two questions

- Why are our tutors behaving this way?

- How often does such learner behavior occur?
  - If extremely rare, might not be a big deal

Framework: mastery learning

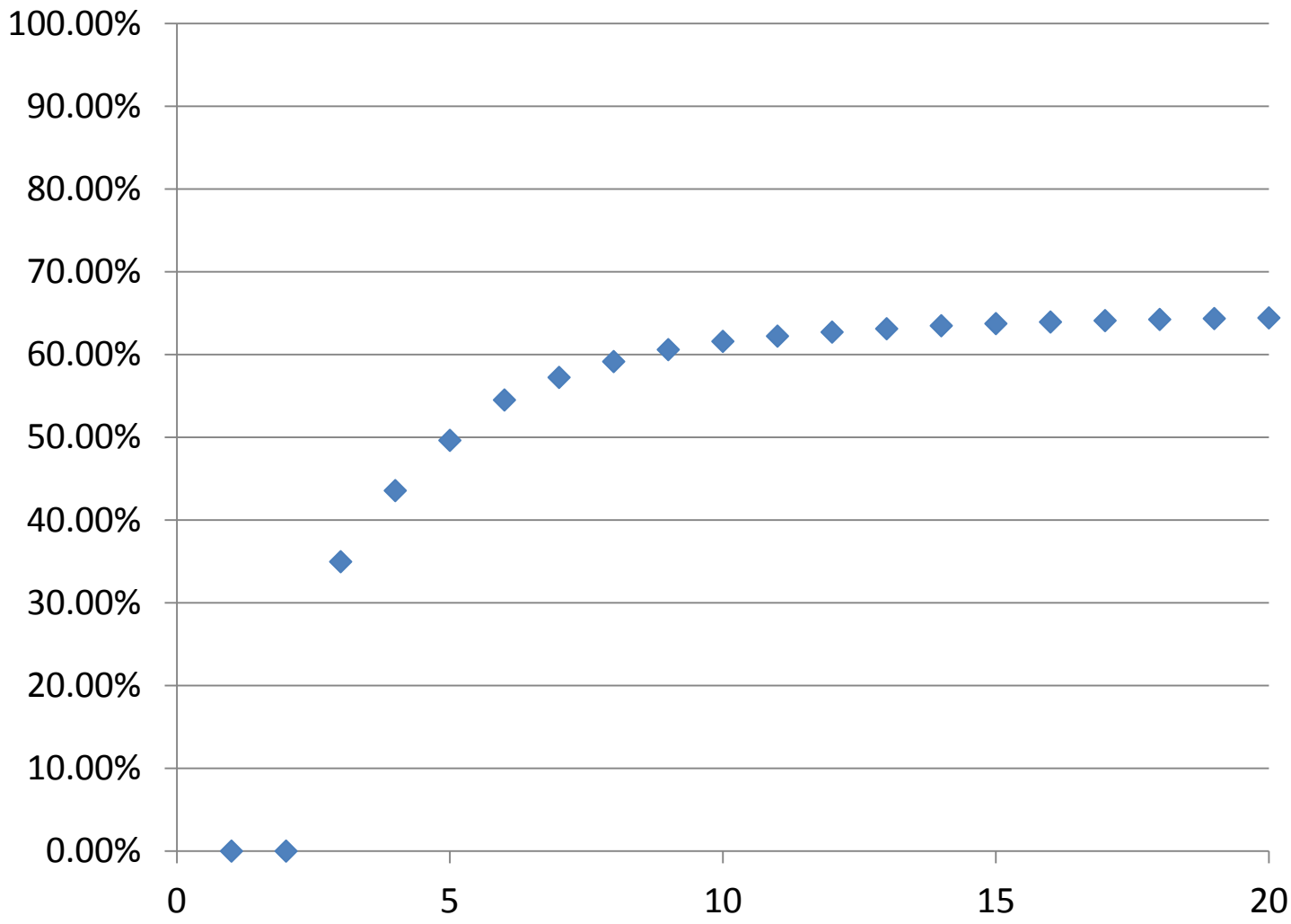


Framework

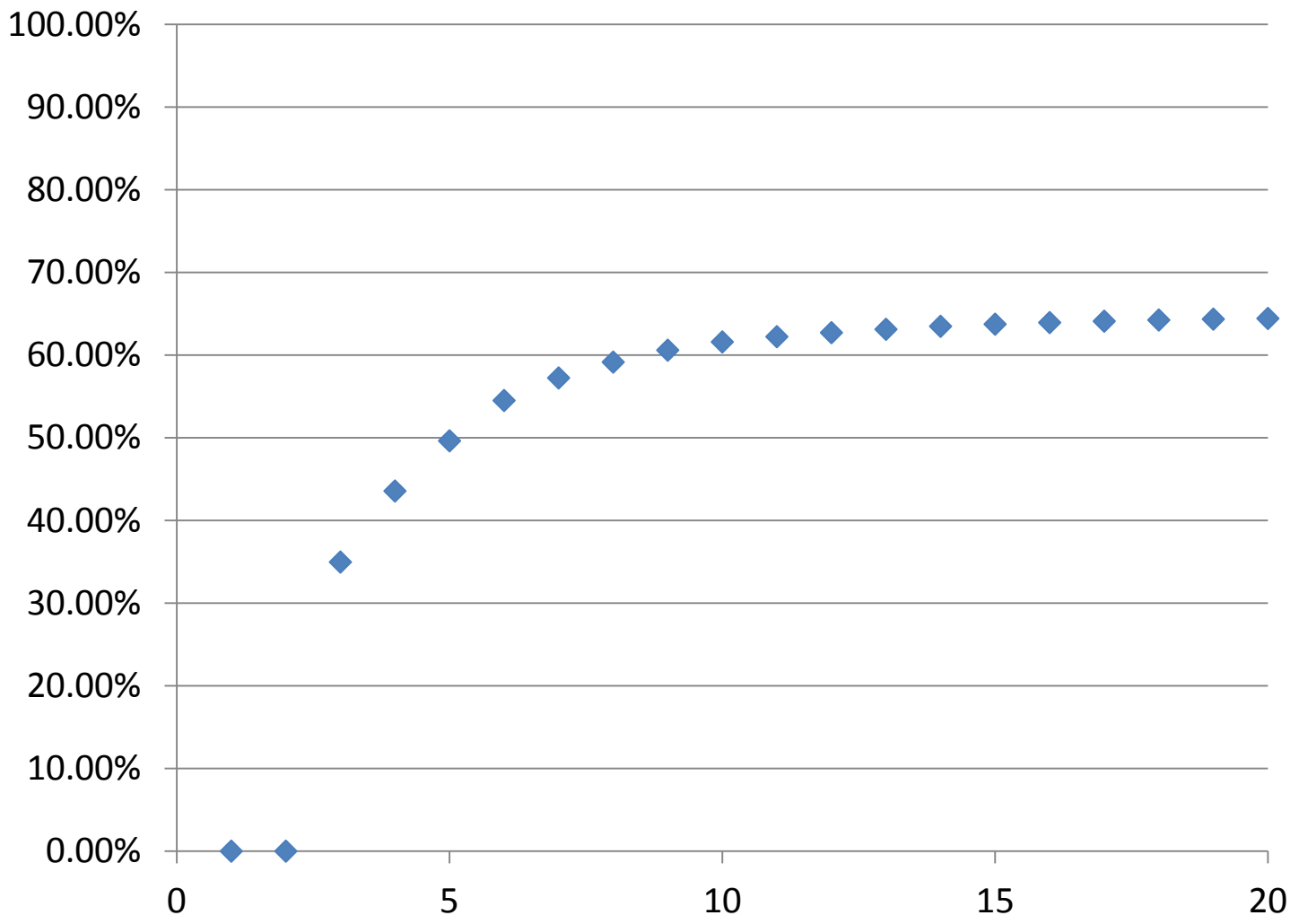
- Mastery: student gets 3 items in a row correct on a first attempt without using help
  - Could use other rules, probably not much difference

- Driving question: what percentage of students have mastered a skill after having X practice opportunities?

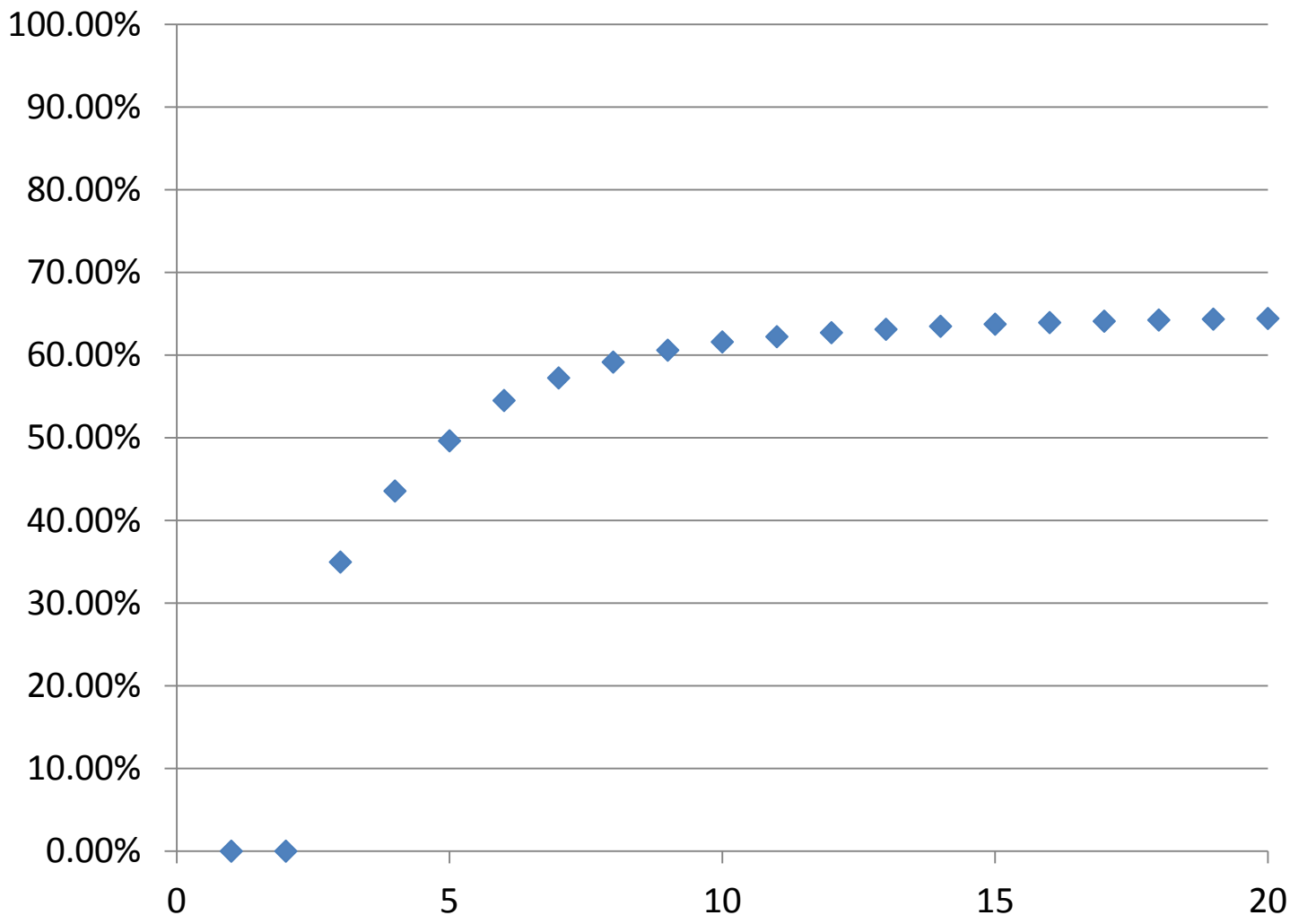
Data from ASSISTments



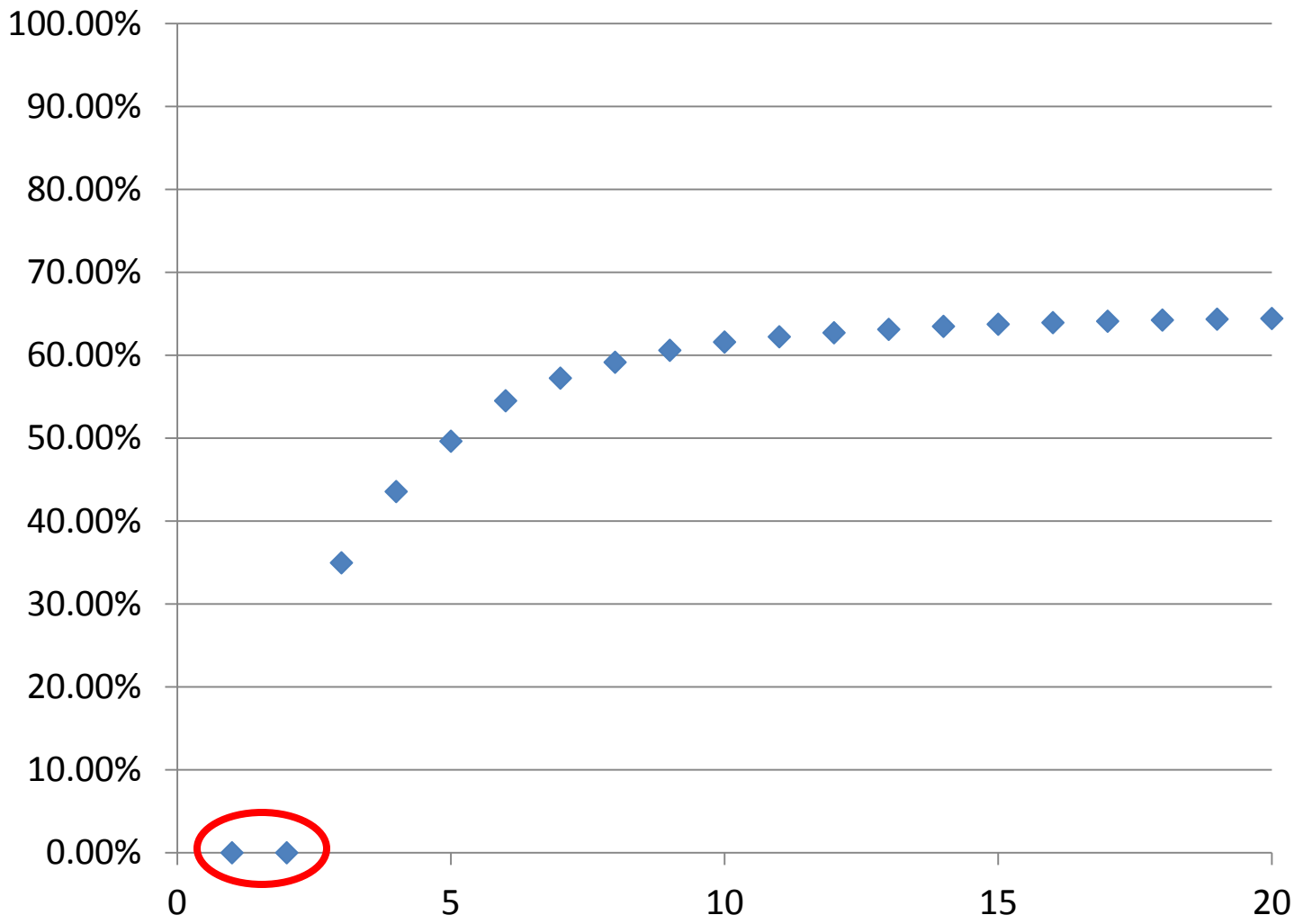
X-axis represents # problems seen



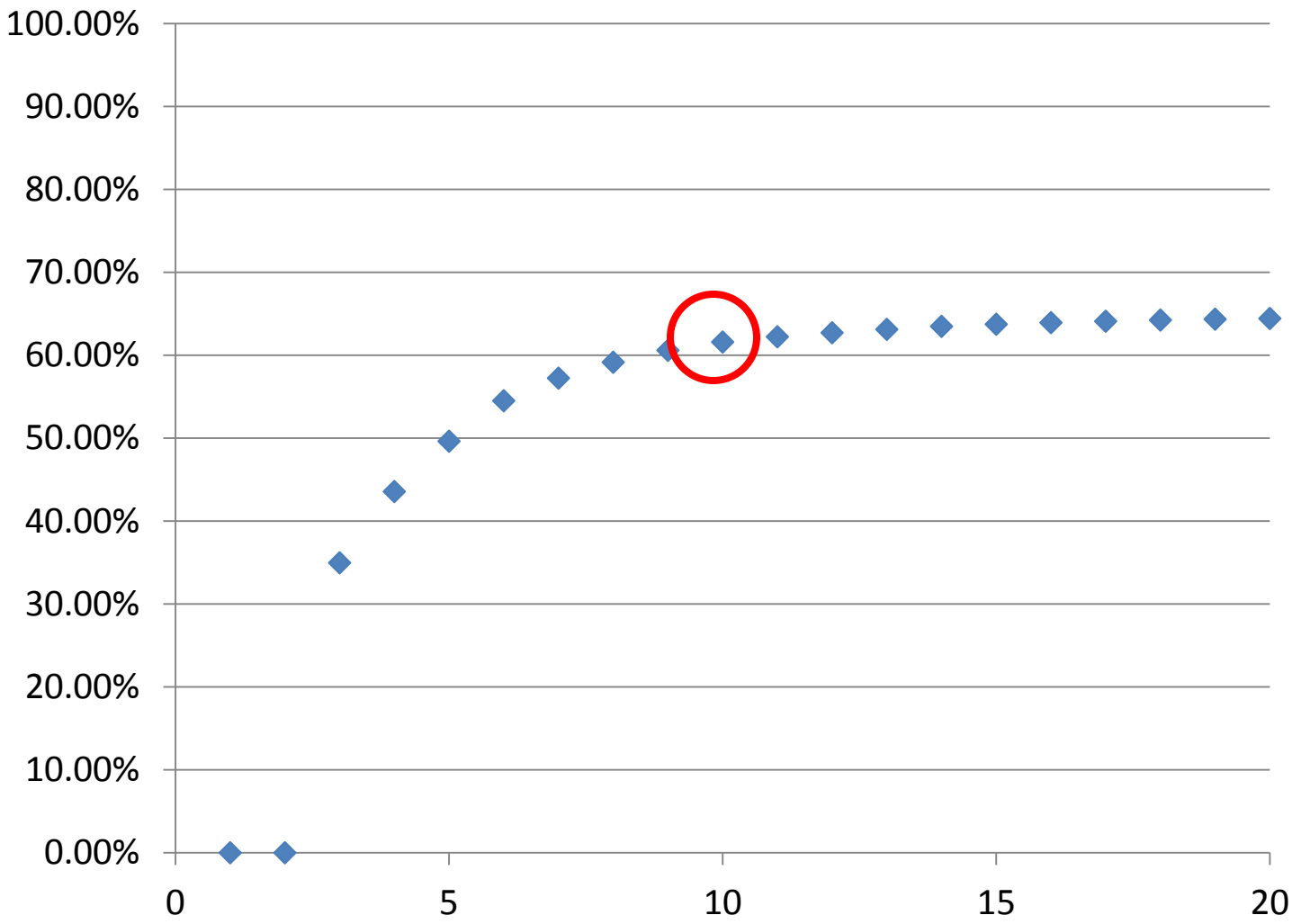
Y-axis represents % mastered so far



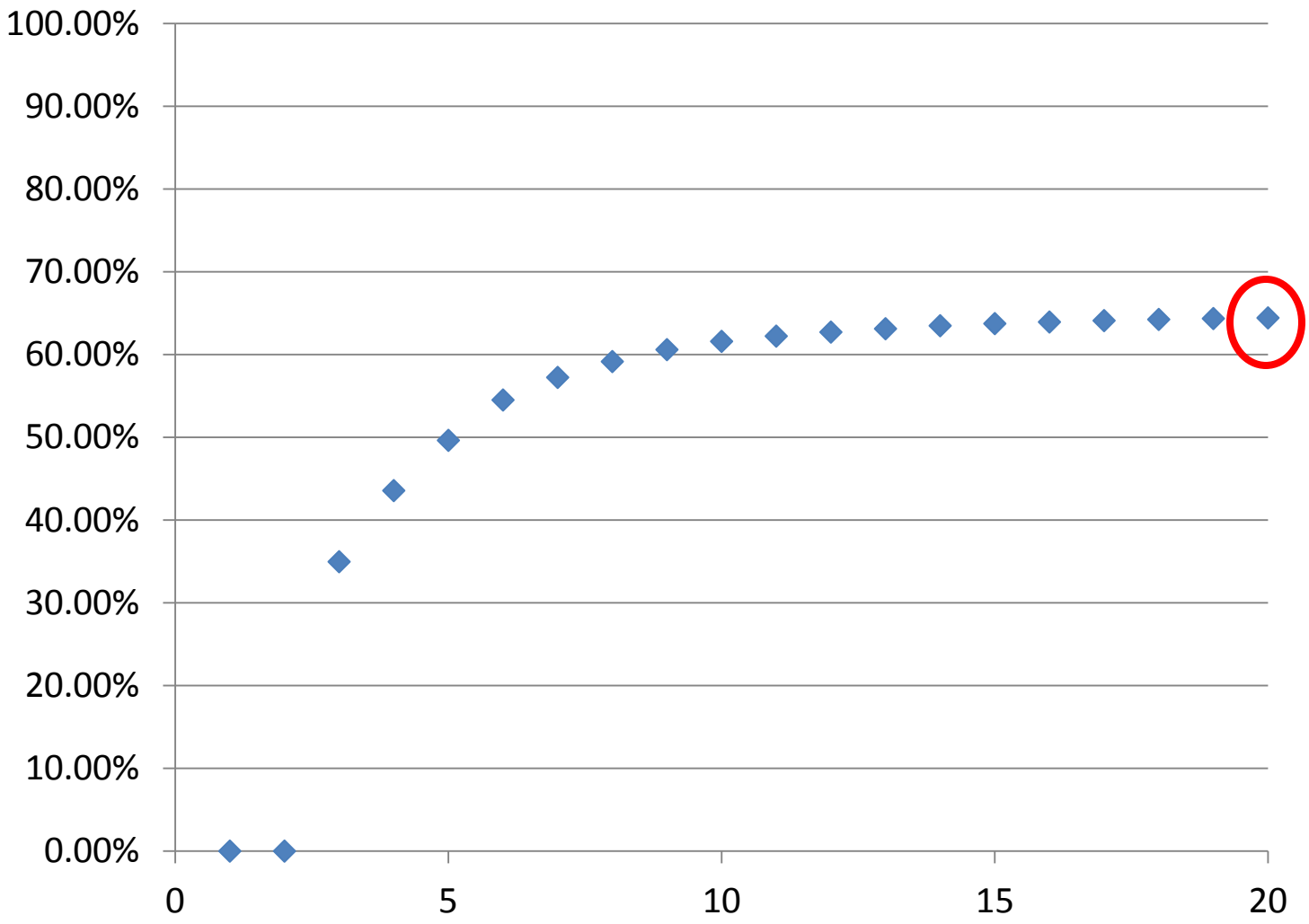
No mastery on first two problems (by definition)



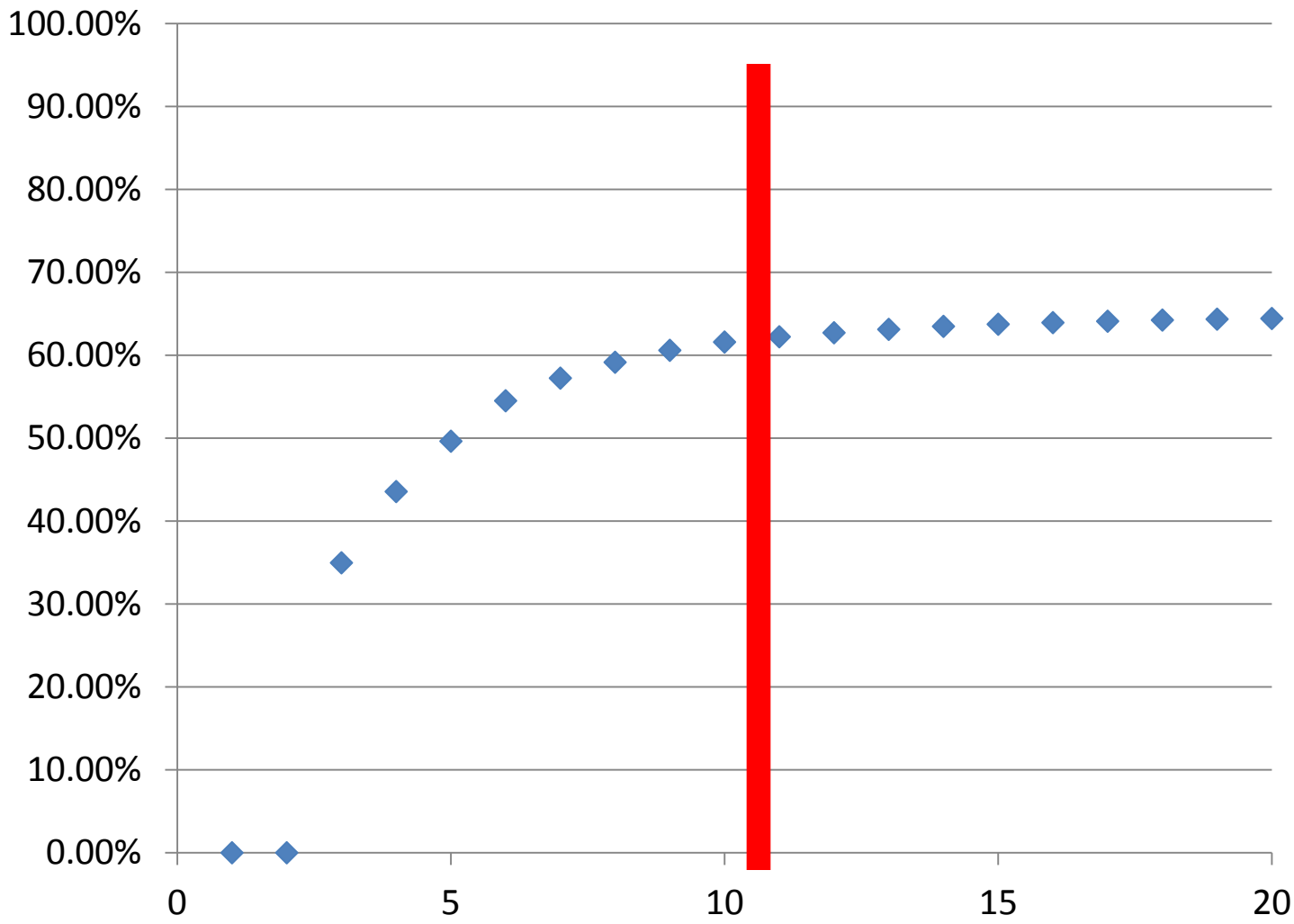
After 10 problems about 62% of students have mastered



After 20 problems about 64% of students have mastered



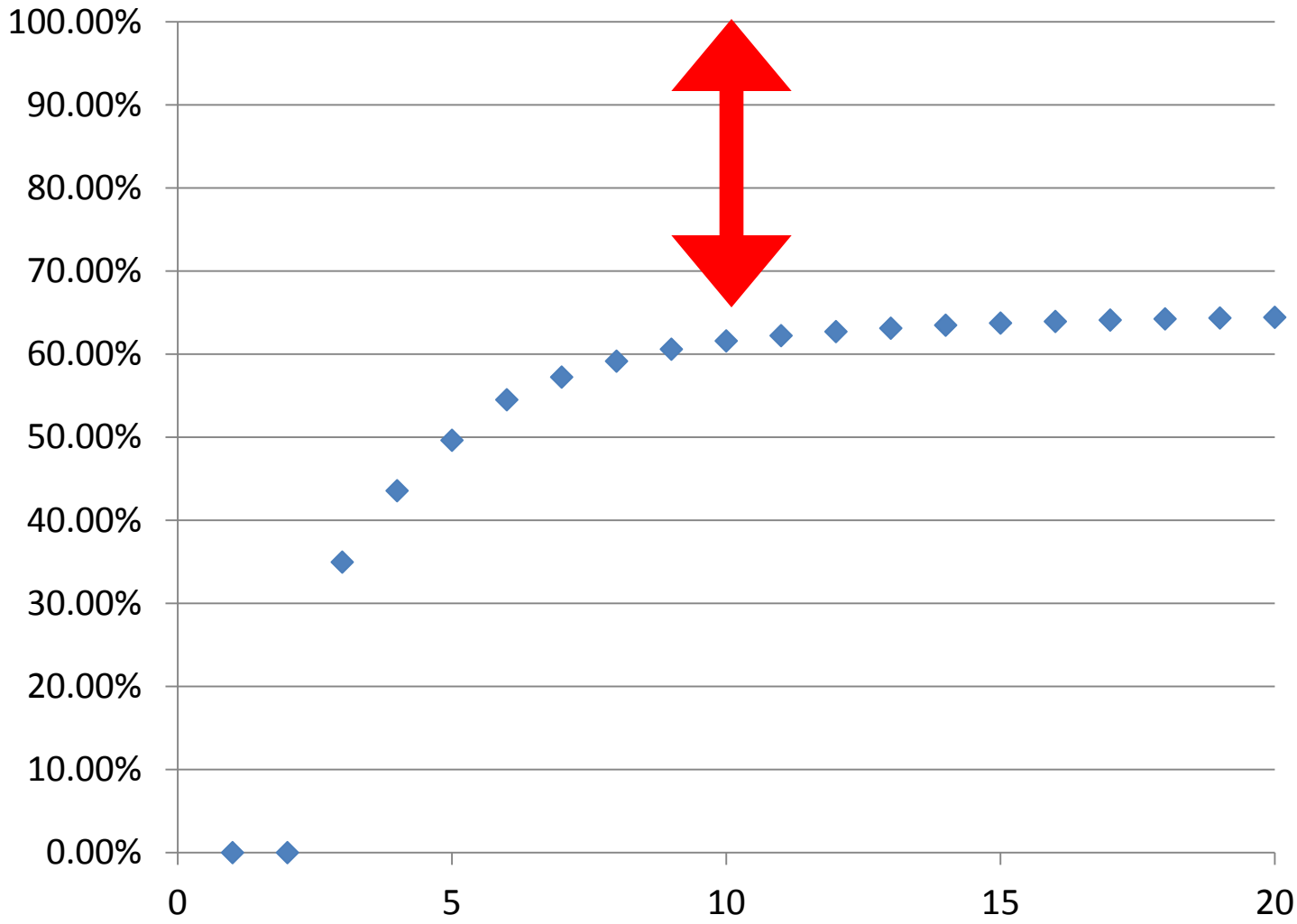
If haven’t mastered after 10 problems, probably won’t master



Implementation detail

- ASSISTments prevents students from making more than 10 attempts to solve a skill in a single day
- If student still has not mastered, locked out and told to talk with a parent or teacher
  - Added before I thought of this analysis
- Therefore, we don't analyze past 10 opportunities
  - Doesn't matter much (62% vs. 64%)

Corollary:
~38% of students will not master

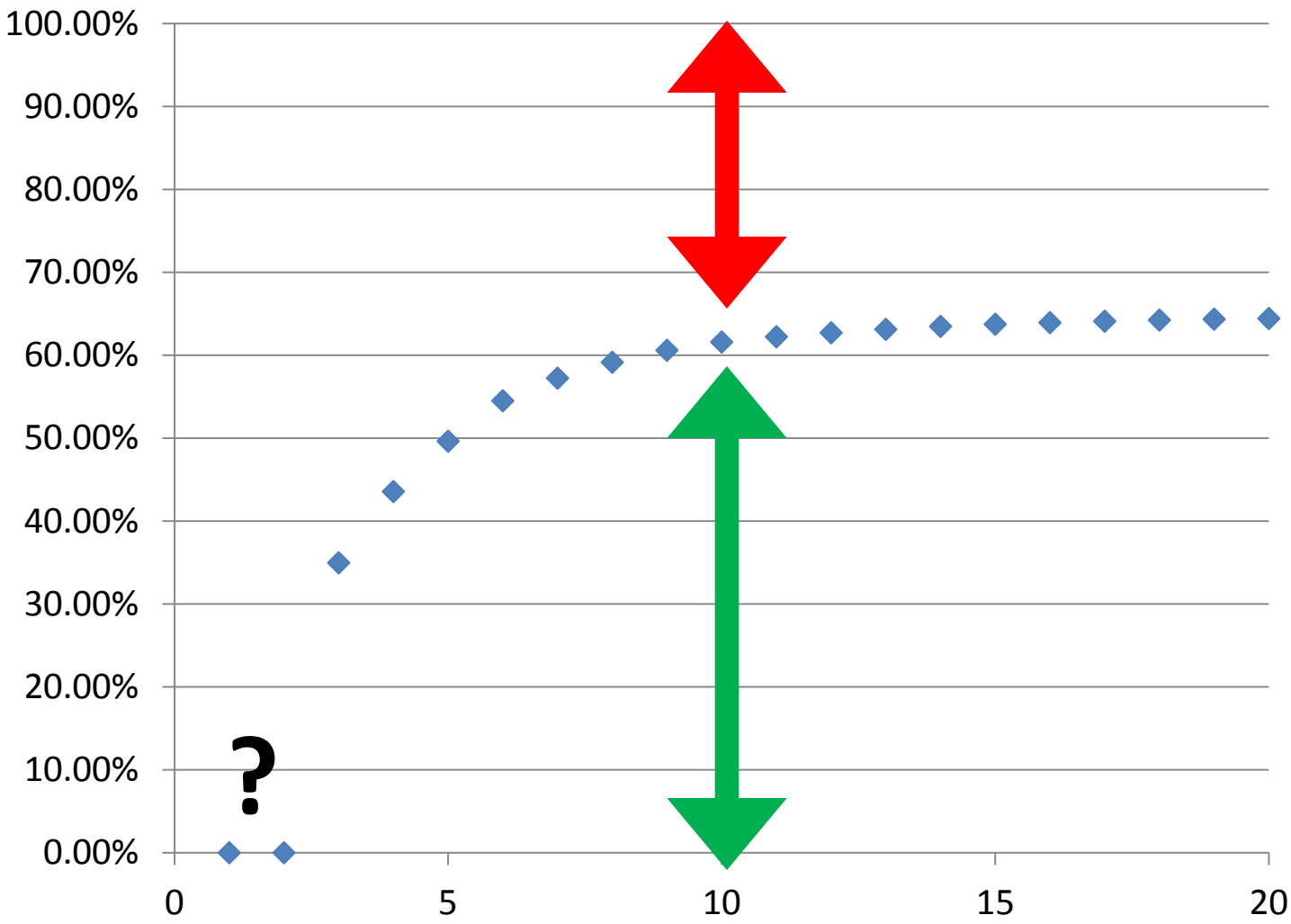


Definition: thrashing

- If student fails to master within 10 opportunities, we say that student is "thrashing"

- In OS, thrashing refers to repeated page faults when trying to run the next job in the queue
  - Computer spends a bunch of time getting nowhere

Goal: **predict** if student will **thrash** or **master** the topic



Approach: logistic regression

- Construct a separate model *for each number of problems seen*
  - i.e., build a model for when a student starts (has seen 0 problems), another model for having seen 1 problem, 2 problems, etc.
  - Allow independent variables to vary in strength (perhaps some more important initially)

- Dependent variable: will student thrash or master?

Independent variables

- # correct in a row
- Total number of correct responses

- Recent response time
- Recent guessing detector
- Recent rapid responses

- Recent bottom out hints
- Total bottom out hints

- Skill_identity (factor)

Results for predictive accuracy

# problems seen	R^2
0	.128
1	.280
2	.388
3	.373
4	.413
5	.452
6	.438
7	.648
8	.847

Before student starts this topic, have some idea

# problems seen	R^2
0	.128
1	.280
2	.388
3	.373
4	.413
5	.452
6	.438
7	.648
8	.847

After 1 problem, much stronger knowledge

# problems seen	R^2
0	.128
1	.280
2	.388
3	.373
4	.413
5	.452
6	.438
7	.648
8	.847

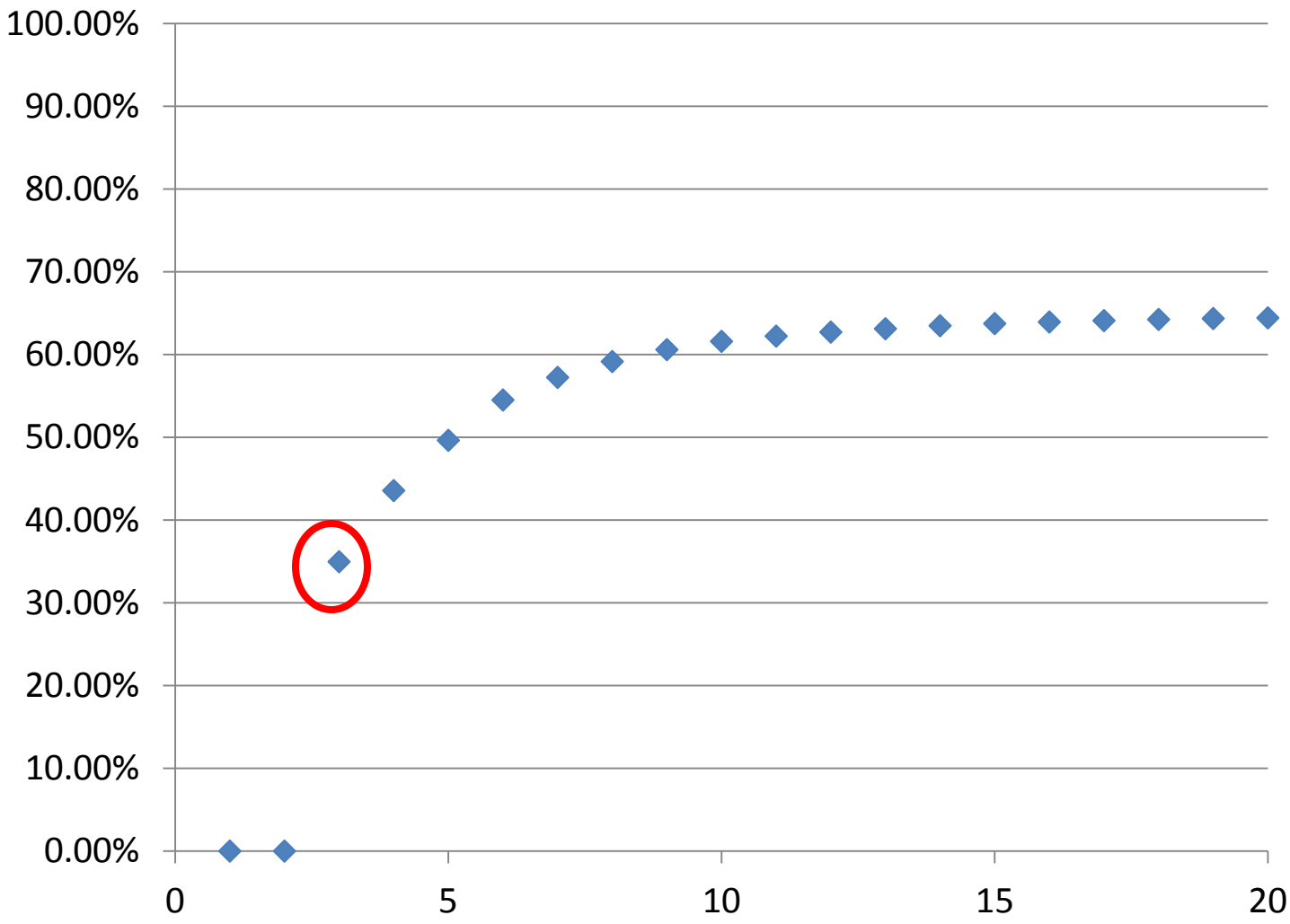
After 2 problems, account for 39% of the variability in the outcome

# problems seen	R^2
0	.128
1	.280
2	.388
3	.373
4	.413
5	.452
6	.438
7	.648
8	.847

For 3 problems, predictive accuracy drops – why?

# problems seen	R^2
0	.128
1	.280
2	.388
3	.373
4	.413
5	.452
6	.438
7	.648
8	.847

After 3 problems many of the students who will master have done so (!)



Thrashing

- Collect data that lets us fairly quickly determine which students are likely to master or thrash

- What is the source of power?

Initially: poor behavior on other topics

# of problems seen	Most useful features
0	Rapid responses (other topics)

Bottom-out hinting right away is bad

# of problems seen	Most useful features
0	Rapid responses (other topics)
1	Number of bottom out hints

Two problems is similar

# of problems seen	Most useful features
0	Rapid responses (other topics)
1	Number of bottom out hints
2	Prior successes, bottom out hints

Later, bottom-out hints stop mattering

# of problems seen	Most useful features
0	Rapid responses (other topics)
1	Number of bottom out hints
2	Prior successes, bottom out hints
3	Prior successes, skill_id
4	Prior successes, skill_id

Note: we do not have prior failures since # failures = # successes - # problems seen

Not just ASSISTments

- Replicated on cognitive tutor data set
- Overall trends similar
  - Substantial amount of thrashing
  - Predictable
    - Had to use slightly different feature set due to provided logs in data shop

Discussion: causality of thrashing

- Thrashing and gaming appear to be linked
  - Gaming detector features (rapid responses, bottom out hinting) predicts thrashing
  - i.e., students who will thrash game more
- Competing models
  - Used as an exam
  - If you don't know how to solve the problem, only choice is to trick the system

Model 1: Gaming causes thrashing

- Student is not taking learning seriously
  - Has rapid responses and bottom out hints
- Unlikely to get an item correct on first try
  - Let alone 3 times in a row to master
- → students who game are likely to thrash

Model 2: Thrashing causes gaming

- Student lack sufficient knowledge to solve the problem *or to learn how to solve it*

- Pays to think back to original work with LISP tutor
  - It is not automatic for students to acquire production rules!
  - Must have knowledge in *declarative memory* before it can be compiled as a result of practice

Model 2: Thrashing causes gaming

- Student lack sufficient knowledge to solve the problem *or to learn how to solve it*
- → unlikely to solve a problem without a bunch of help or many attempts

- What's his reward? Another problem
  - Definition of insanity…

- Becomes frustrated → gaming behaviors

Two competing models

- Gaming → Thrashing
  - Low engagement is root cause of bad behavior
- Thrashing → Gaming
  - Lack of knowledge is basis of bad behaviors
  - (personally, I suspect this one)

Hard to disambiguate!

- Low knowledge and gaming behaviors are very highly correlated
  - Student model does better by pretending gaming actions are incorrect than by treat as a null observation (under review)
- Low knowledge is correlated across skills
- In reality, *both explanations probably explain some student behavior*
  - *Low knowledge → thrash → bad behavior → thrash*

One possible experiment

- Intervene with some students to increase student knowledge directly

- Examine thrashing and gaming rates
  - Gaming→thrashing: thrashing should decrease
  - Thrashing→gaming: both should decrease
    - (see Baker's 2006 ITS paper)

Baker's ITS 2006 paper

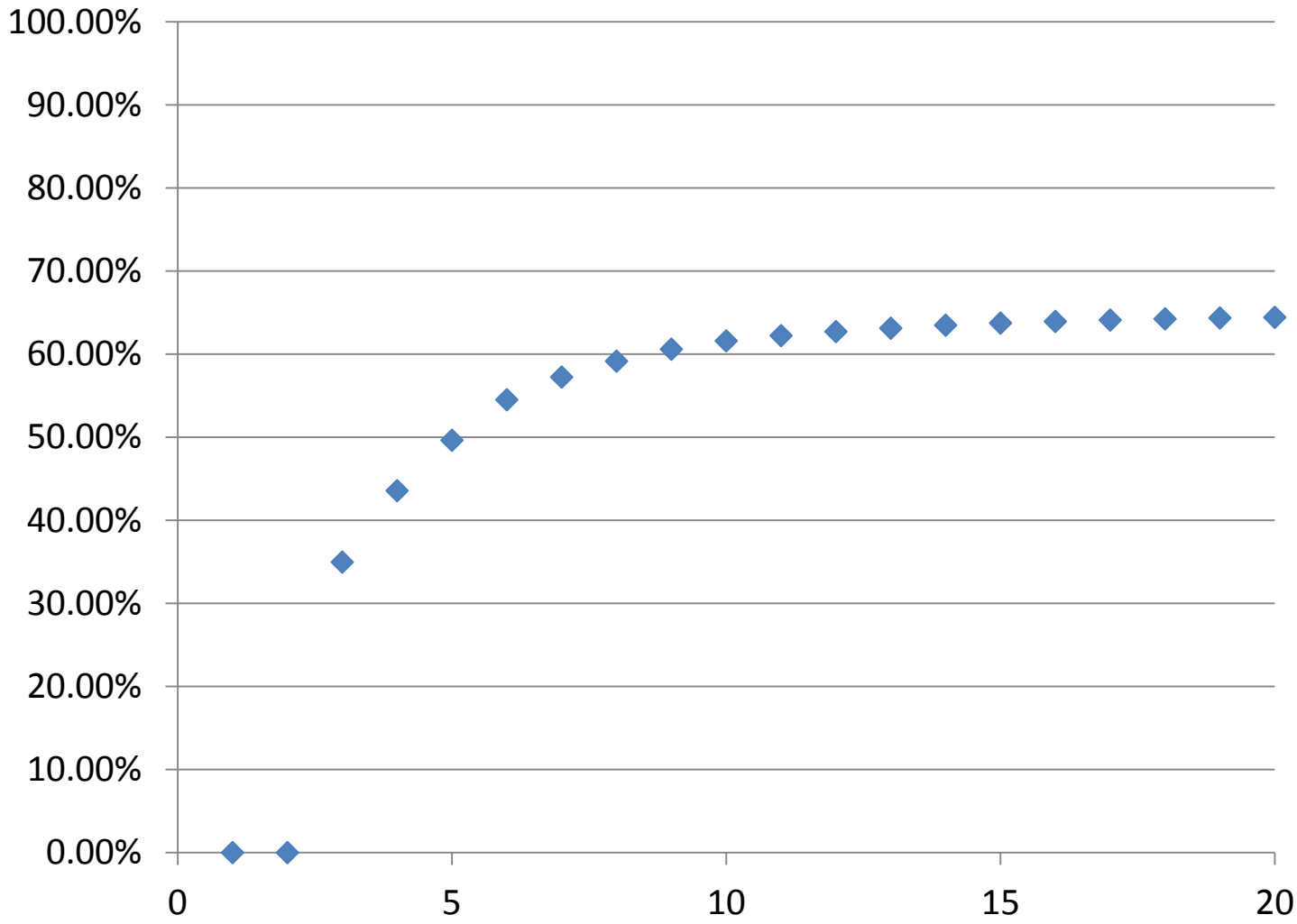
- Didith: "the Scooter the tutor paper"

- What he also did was add a pedagogical component
  - It taught students the skill they were struggling on

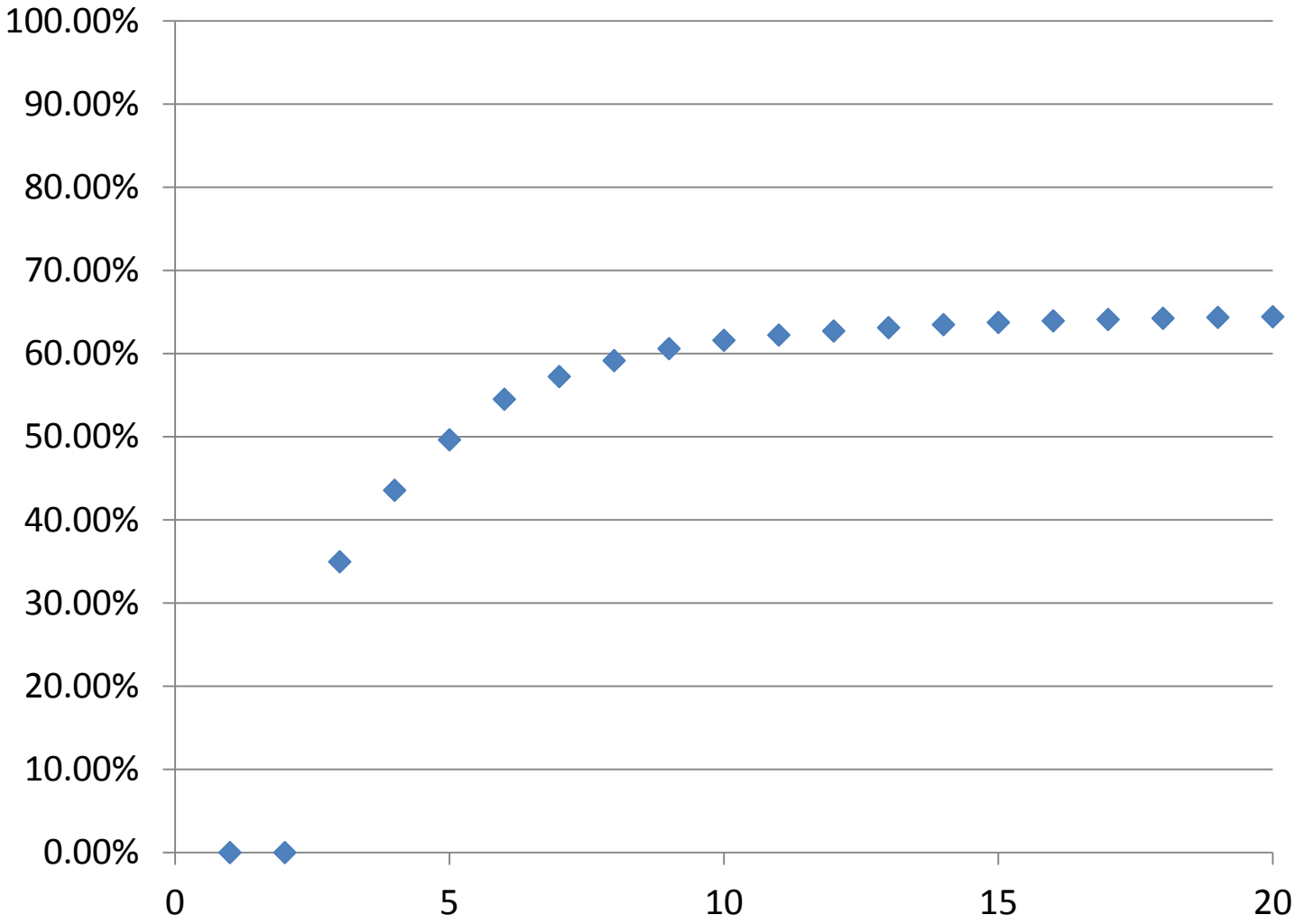
We're working on this issue

- Increasing student declarative / conceptual knowledge major issue in ITS
  - Discuss in WEBsistments talk
- Don't see how to resolve with EDM
  - Seems ideal case for intervention study
- (caring more and more about causality)

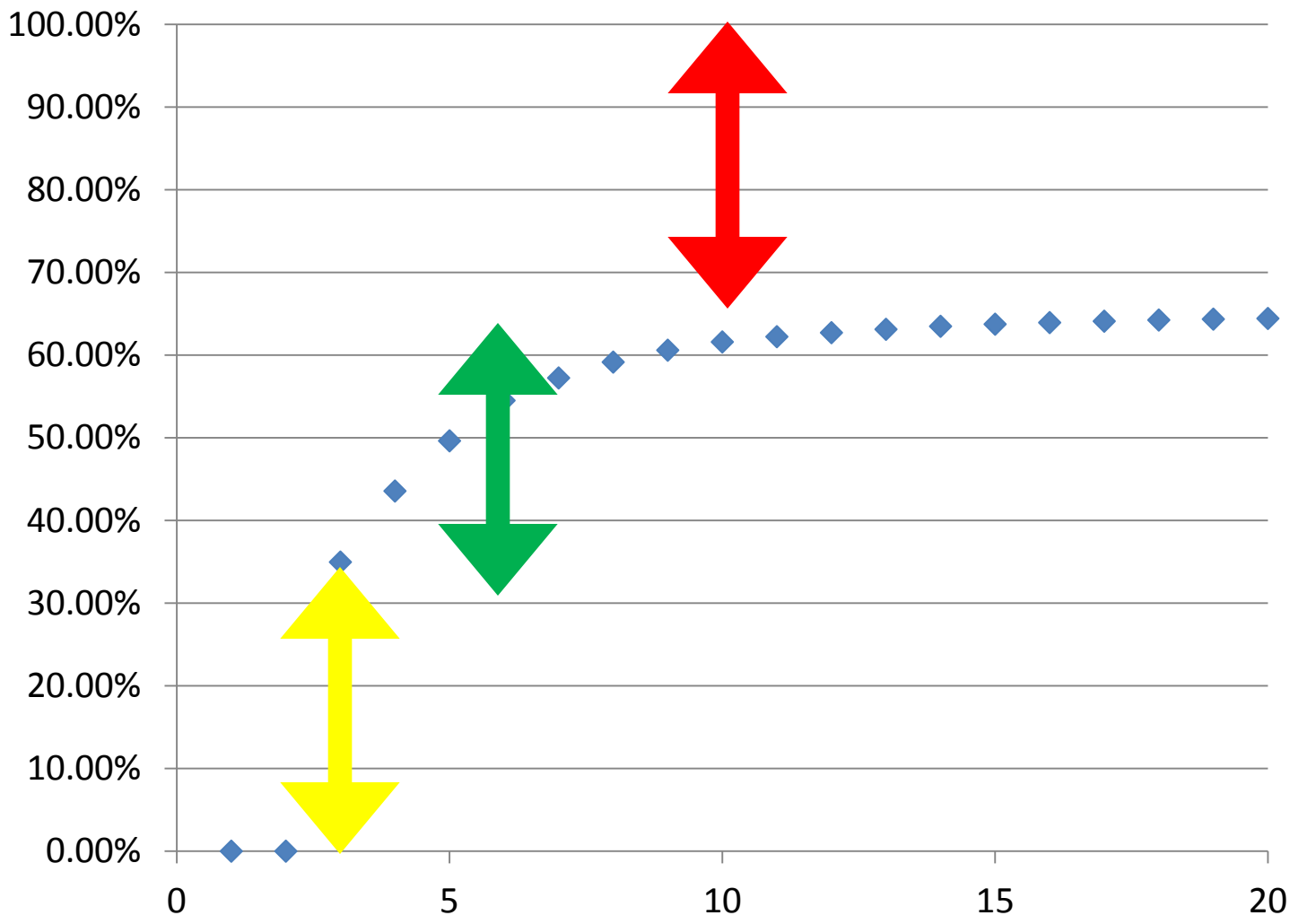
Discussion: use of thrashing graph for evaluating systems



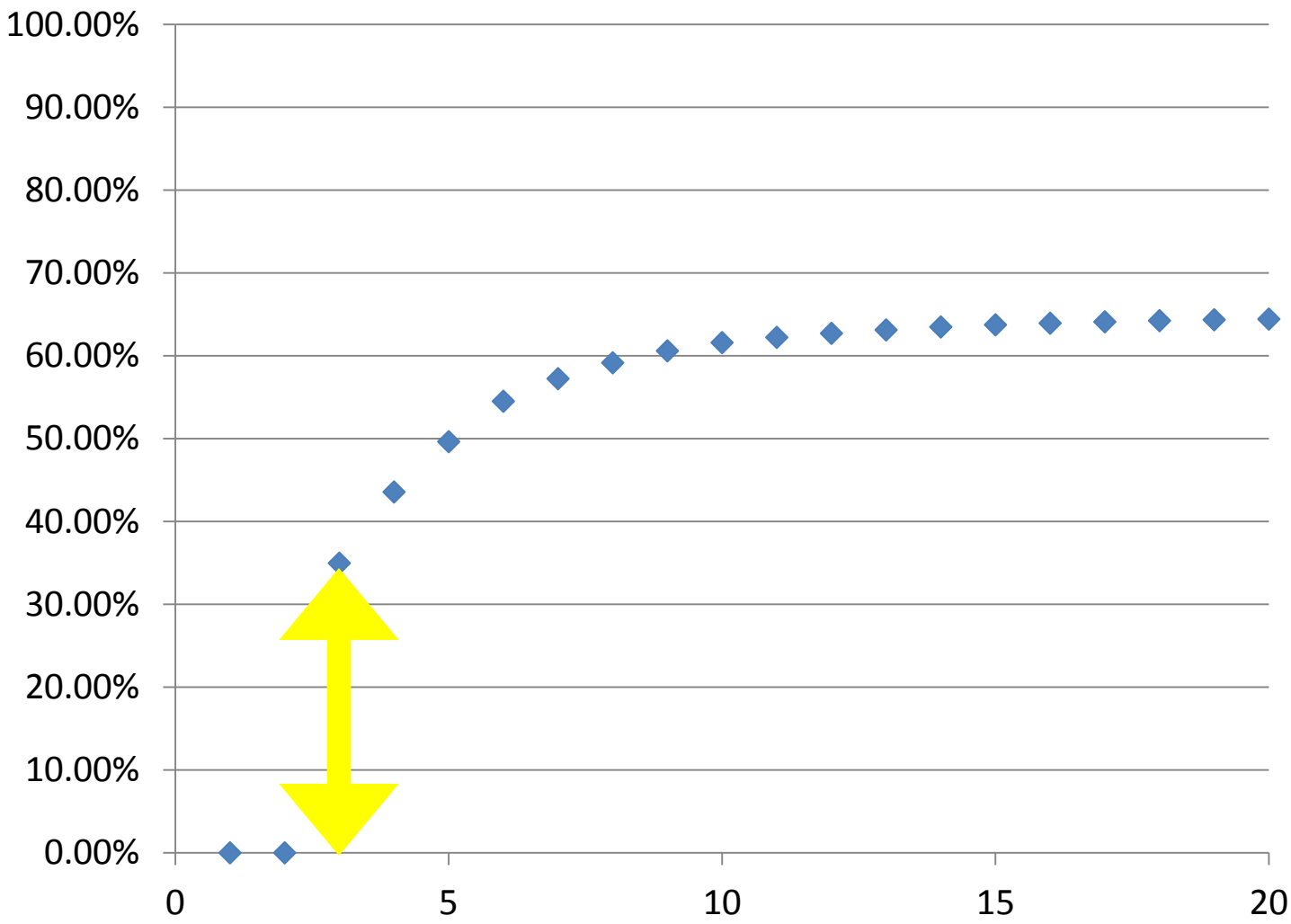
Discussion: use of thrashing graph for evaluating systems – thoughts?



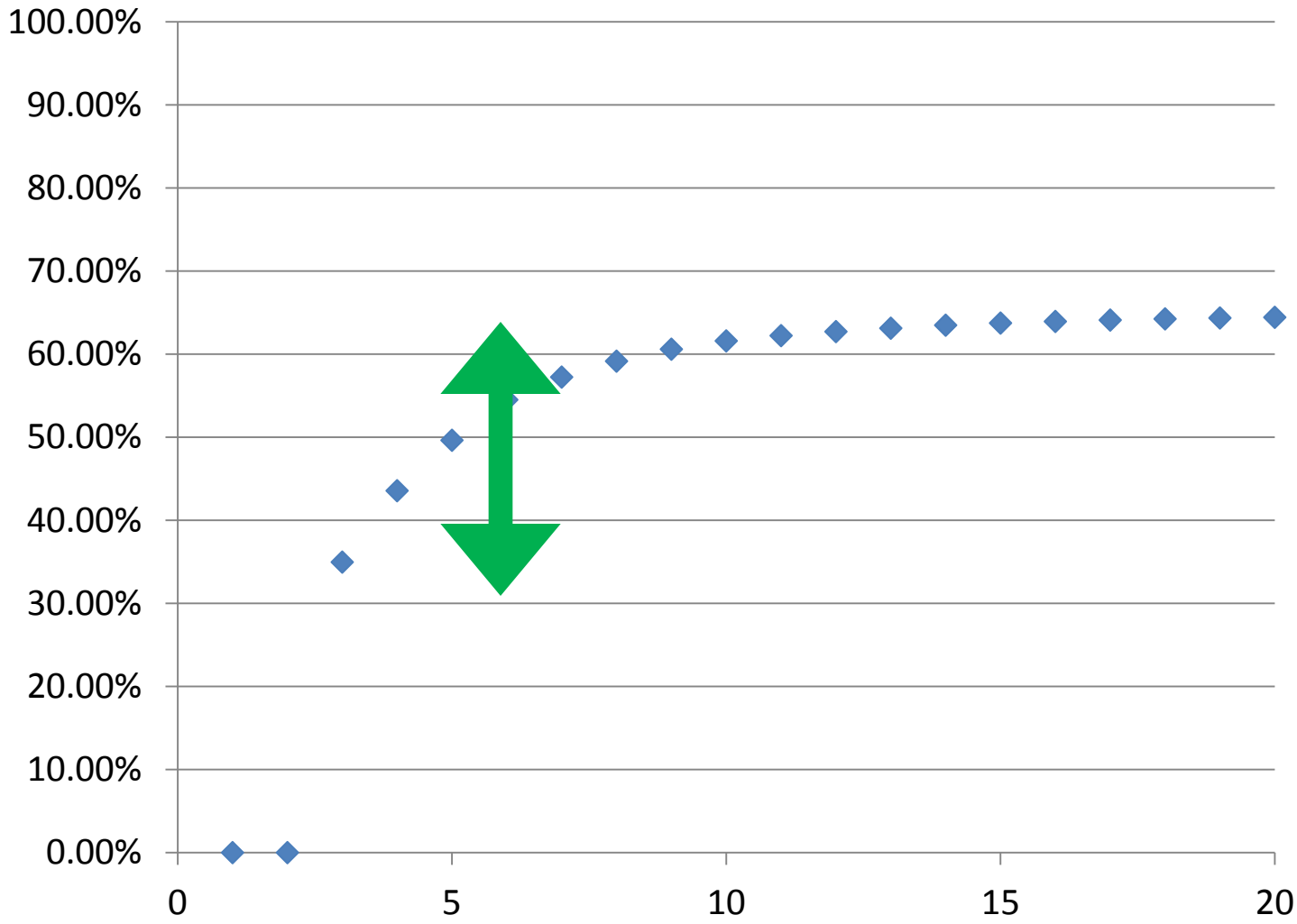
Three groups of students



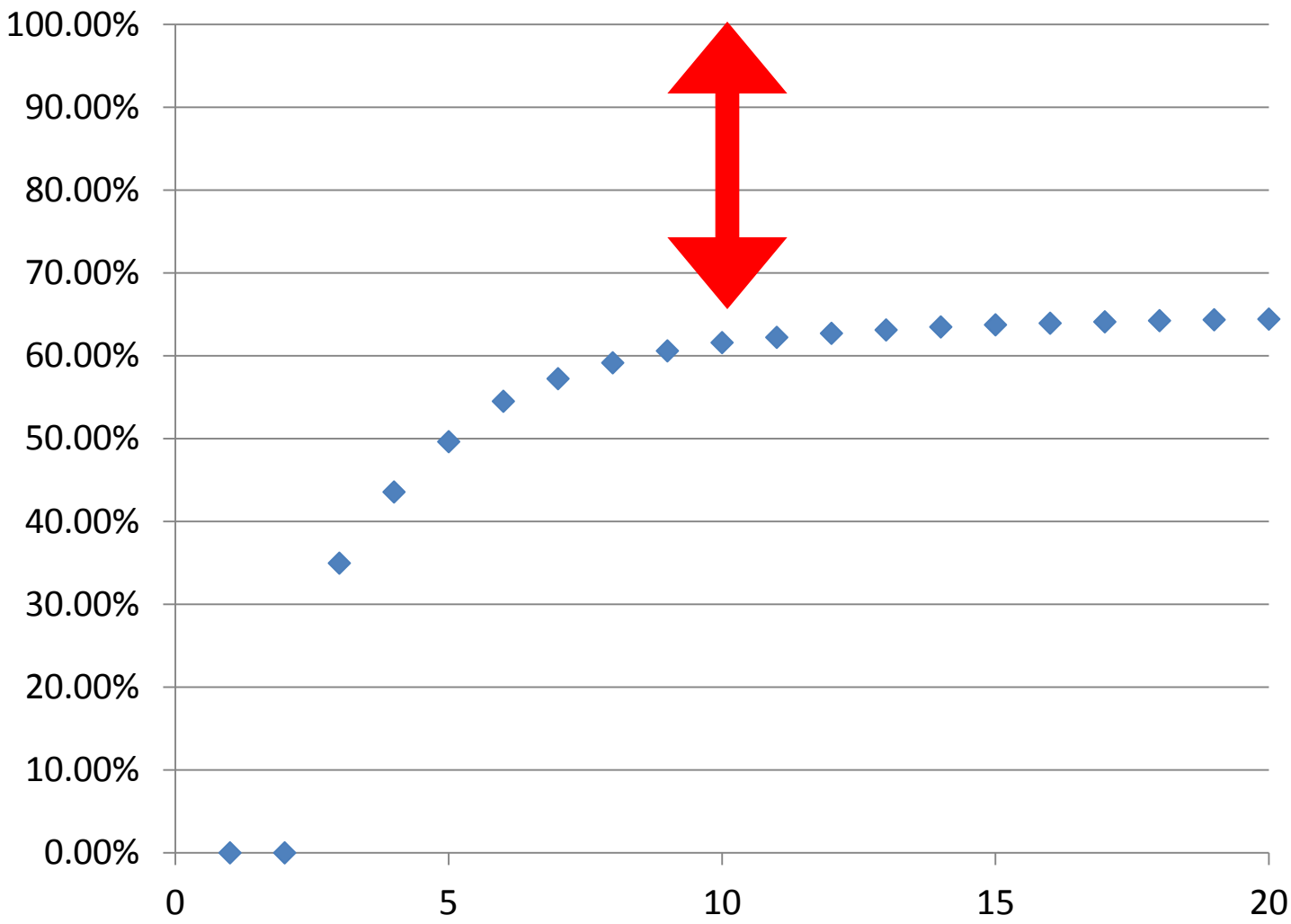
Some students already know the skill



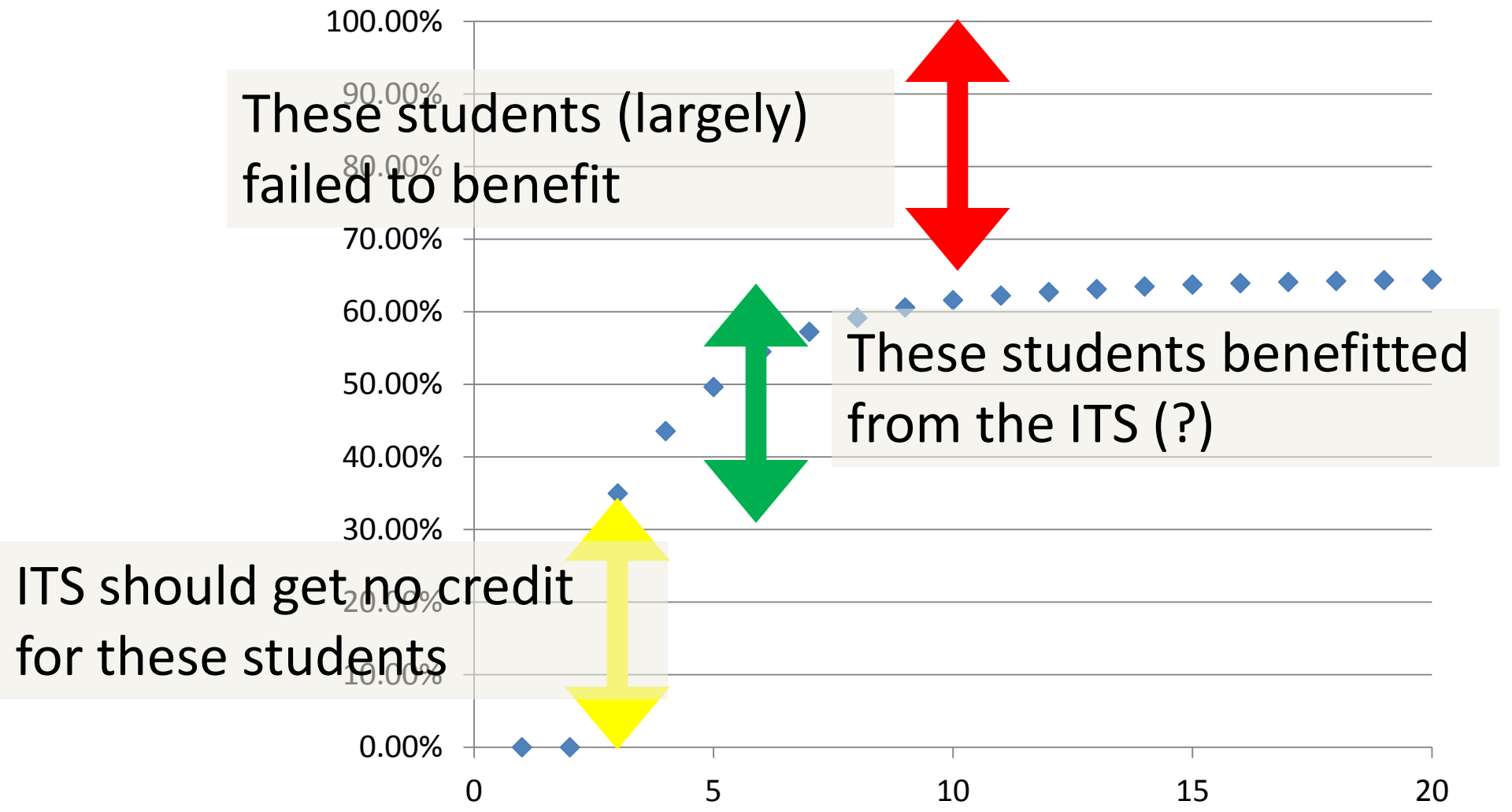
Some student **learn** the skill while using the ITS (**from** using it?)



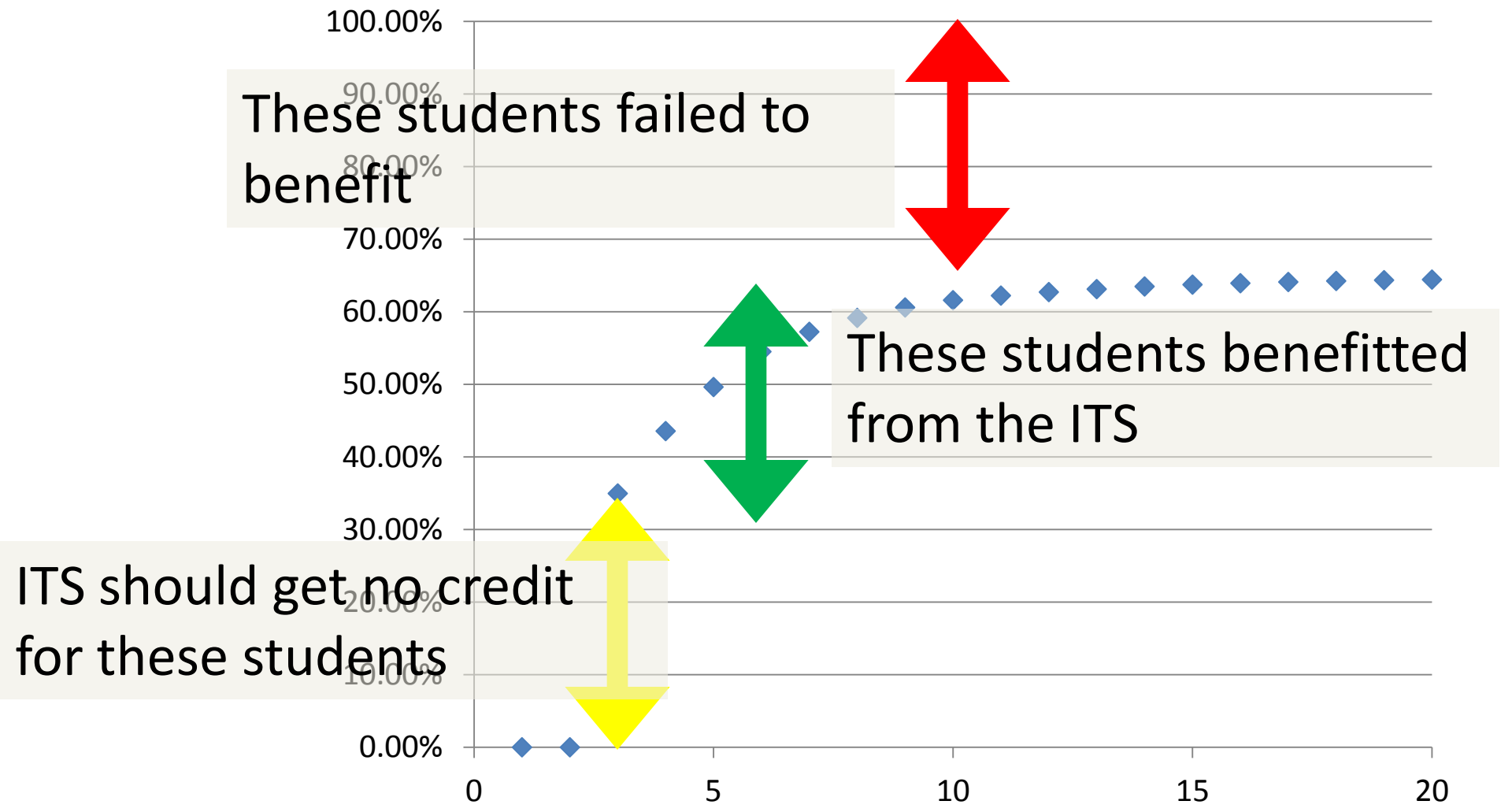
Some students **thrash**



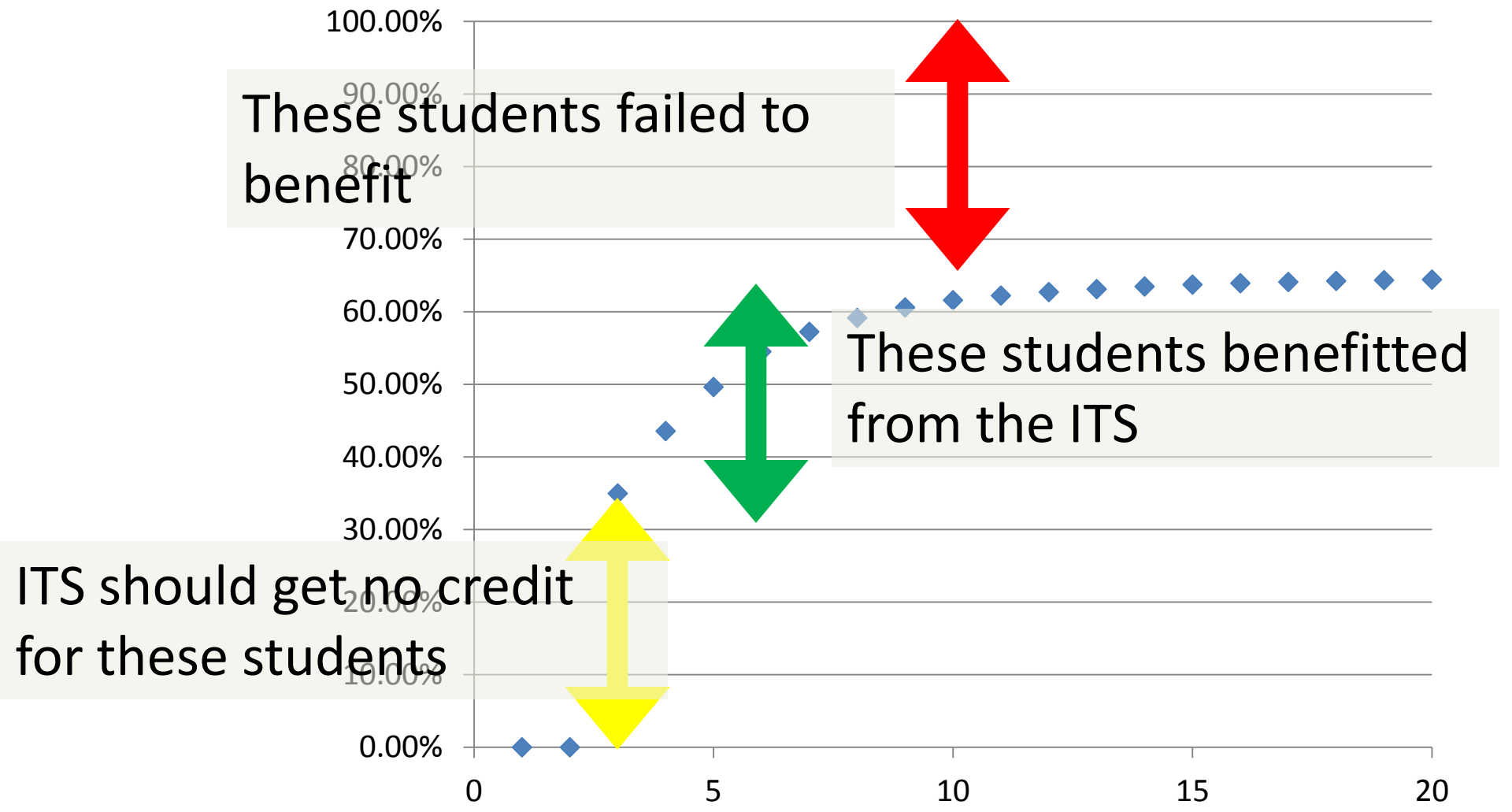
All three groups are interesting!



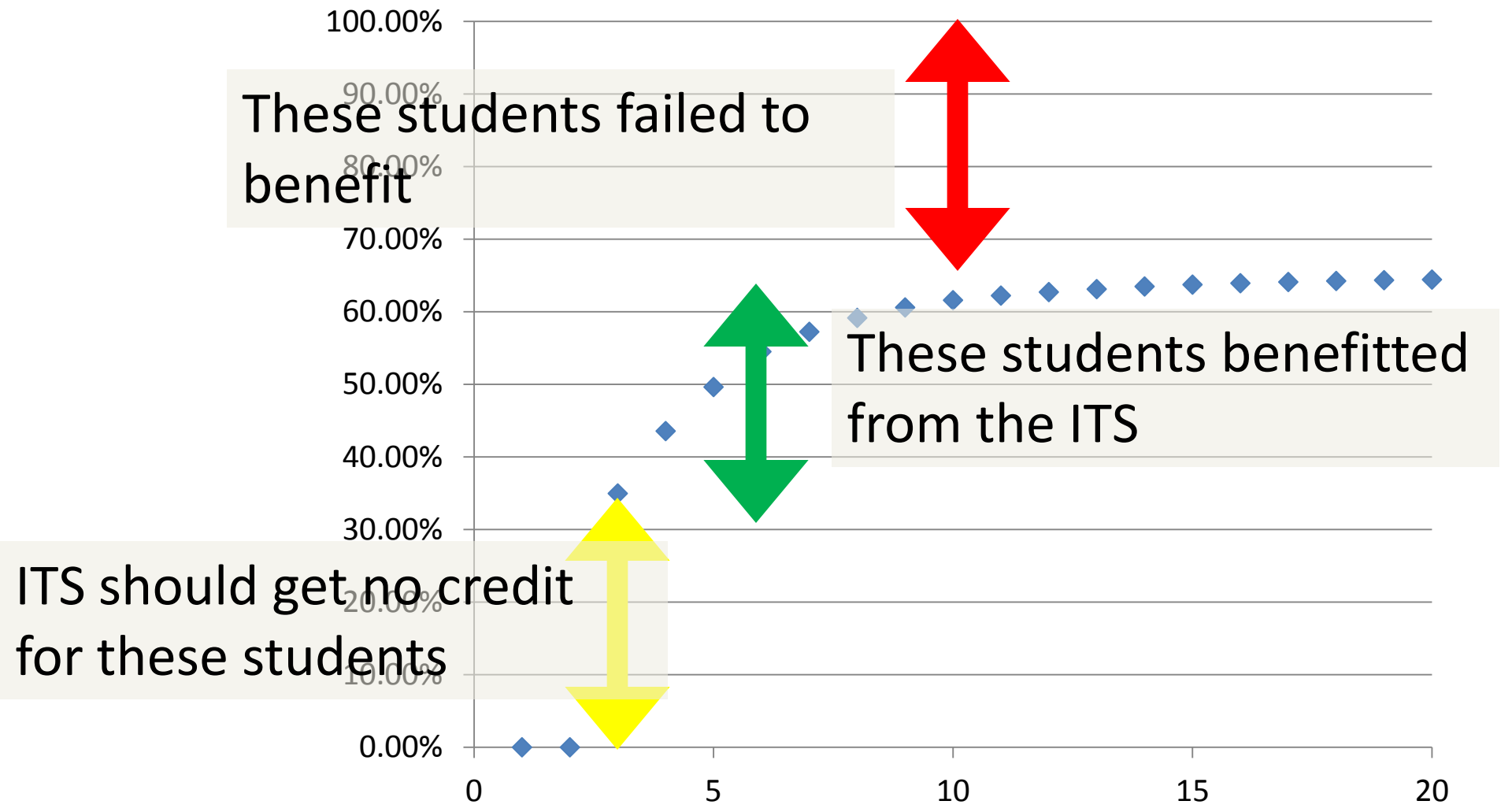
Argue: effectiveness of tutor α |learn group|



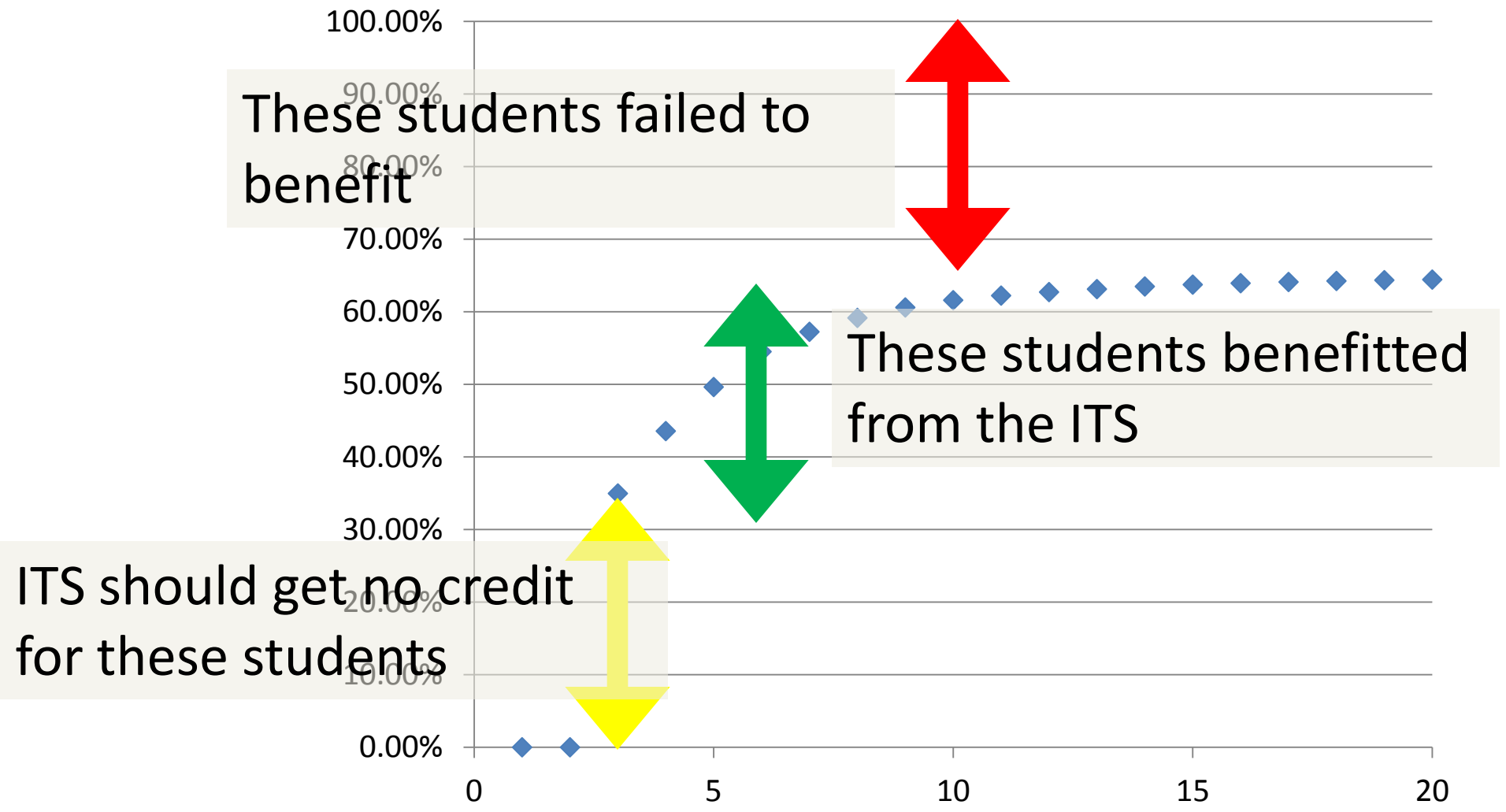
More precisely: effectiveness of tutor <= |learn group|



Argue: effectiveness of prior instruction α |already knew|

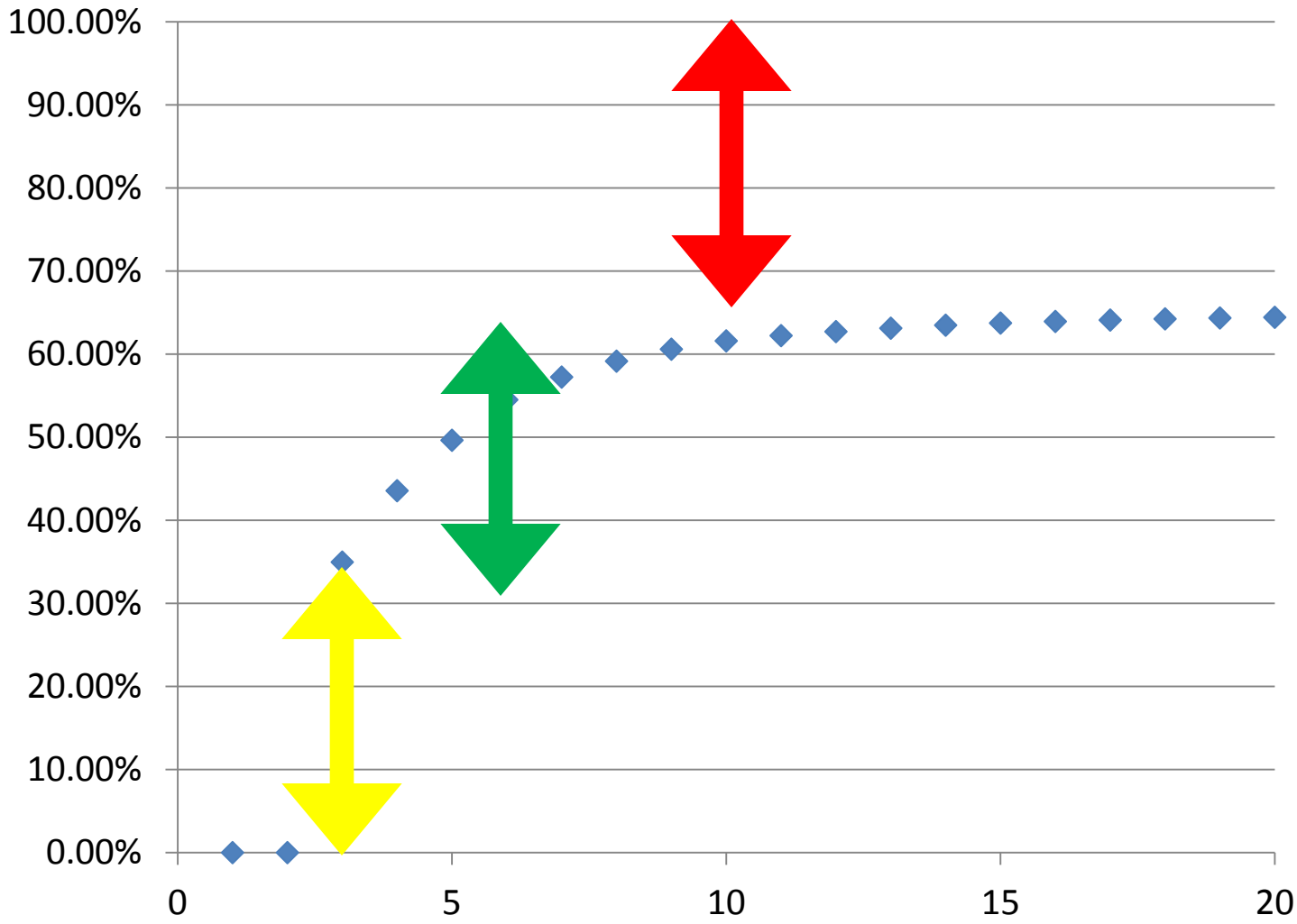


already knew is estimate of how well-timed ITS is in curriculum (sort of?)

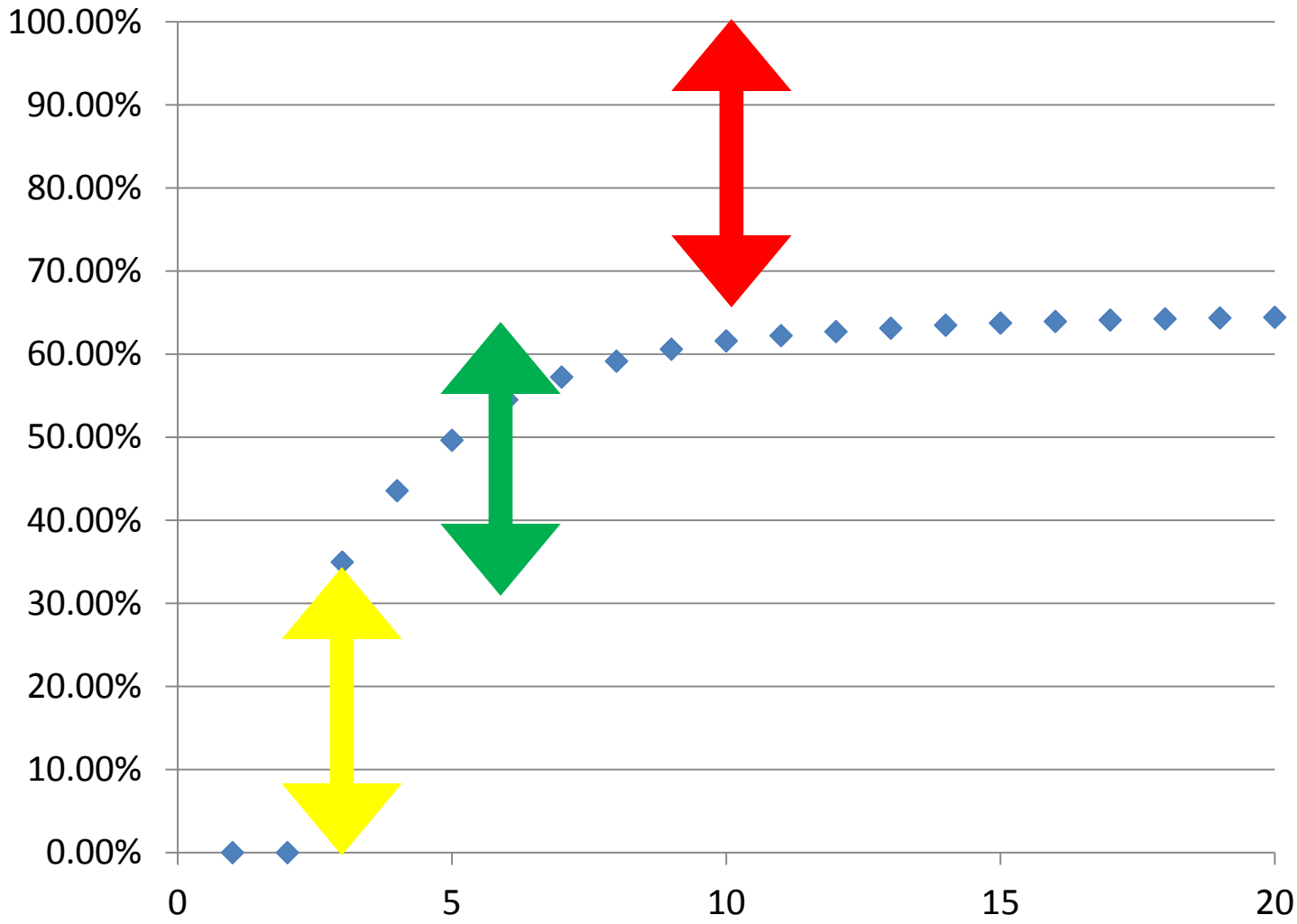


Conjecture: gaming/disengagement/xyz

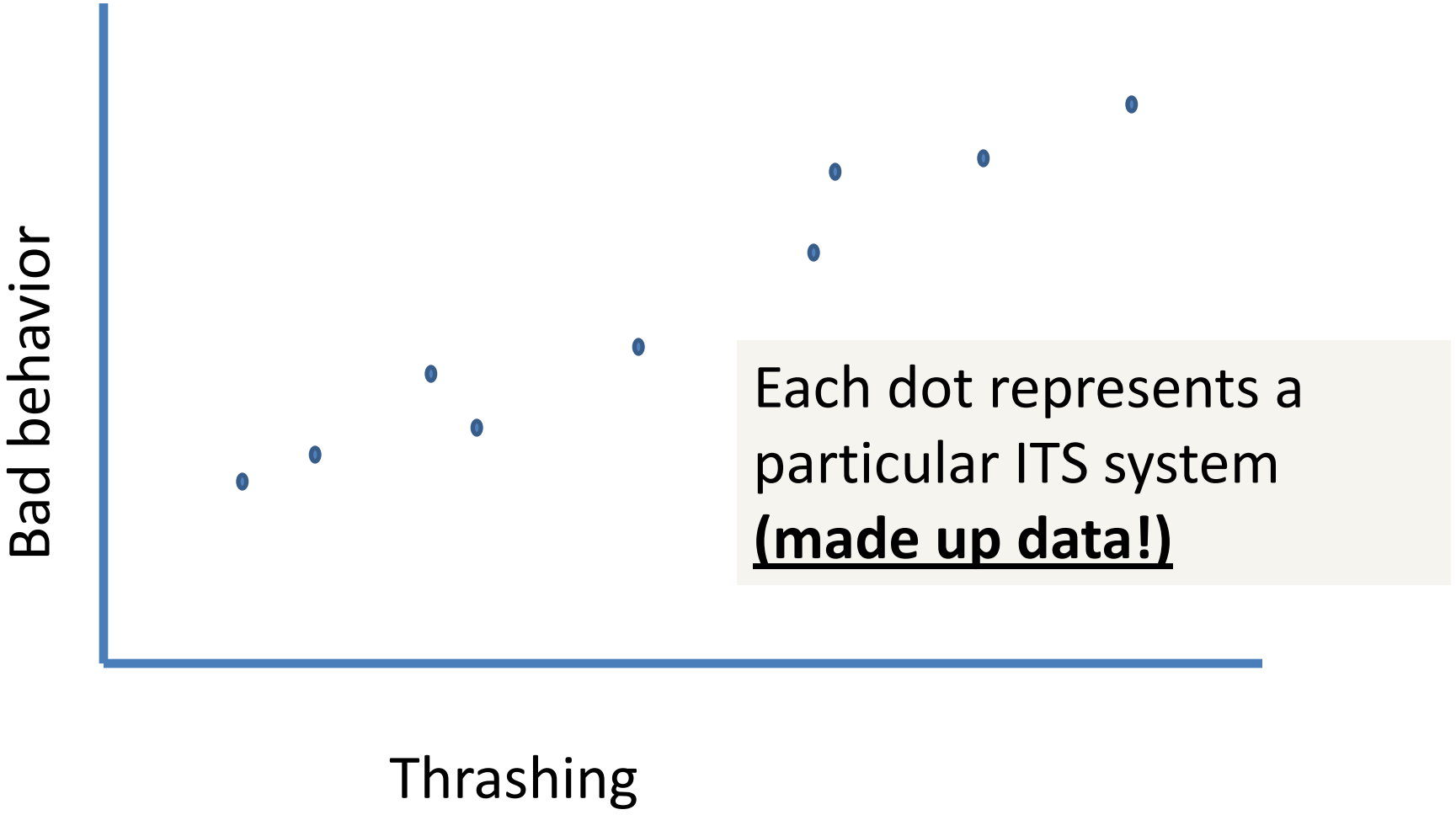
α|thrashing|



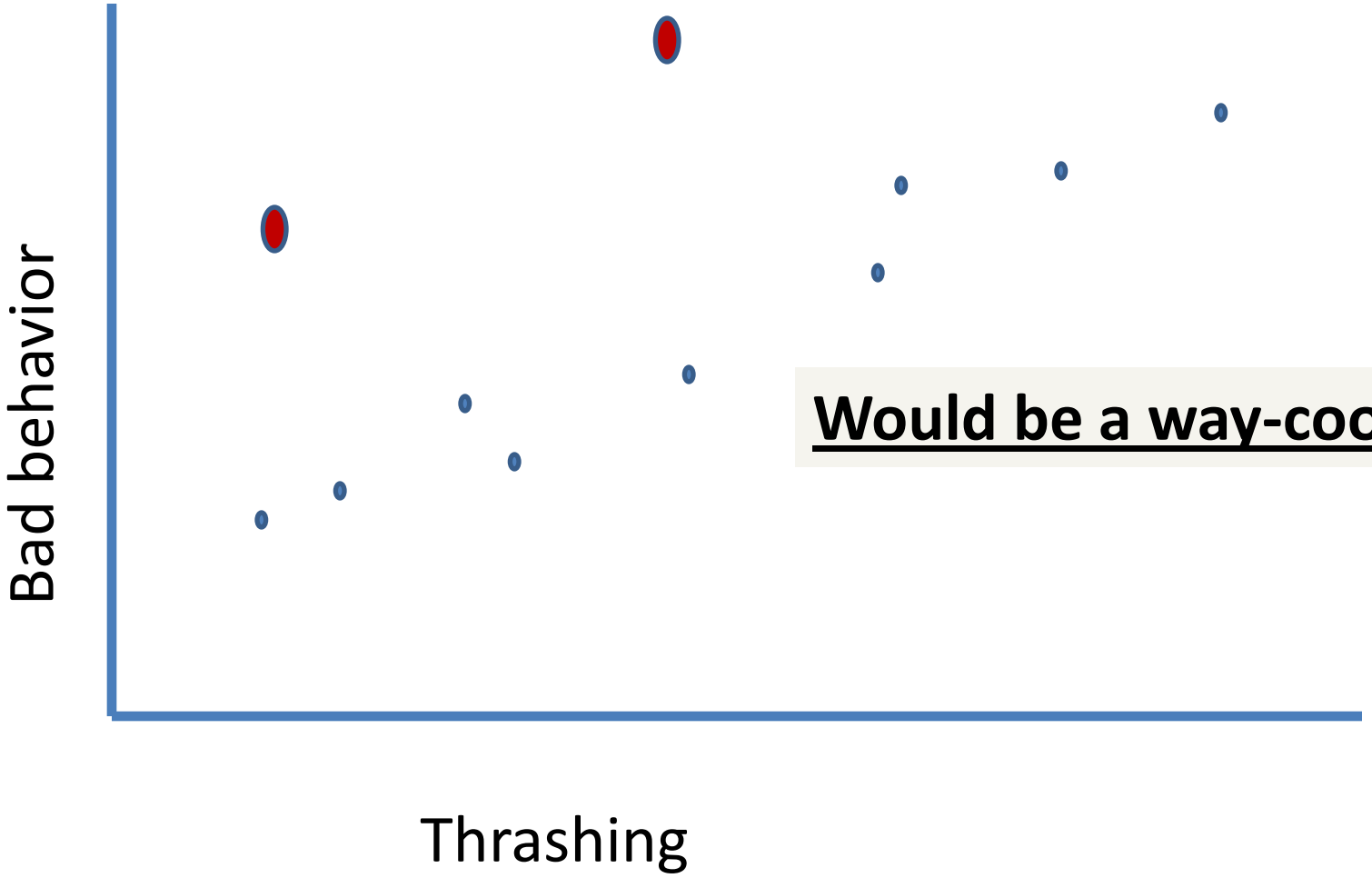
If **thrashing** is root cause of many bad behaviors, reducing thrashing will help



Use observational data (learnlab)! Plot thrashing vs. bad behavior rate

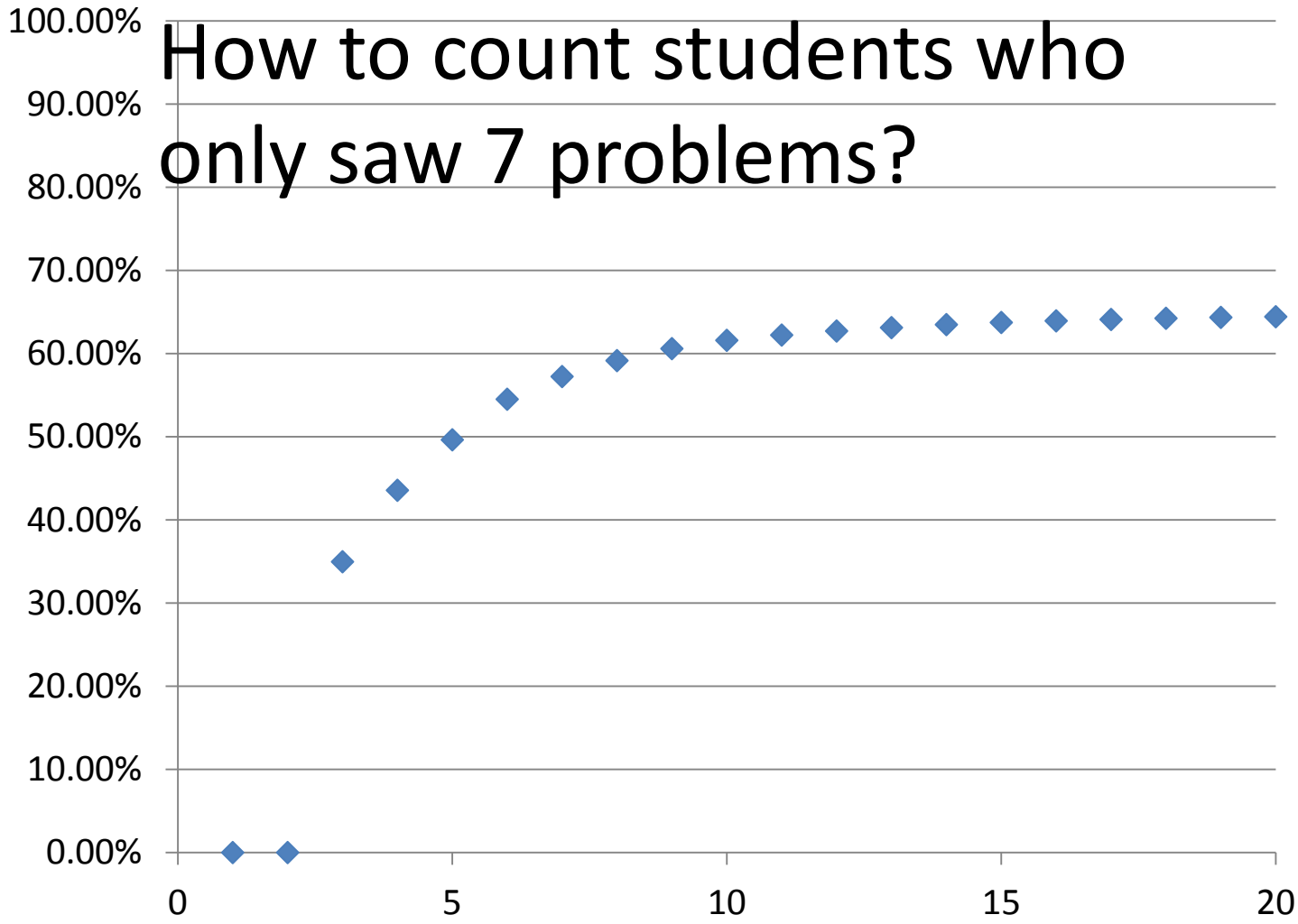


In real life, **detectors vary**



Would be a way-cool paper!

Important caveat of graph!

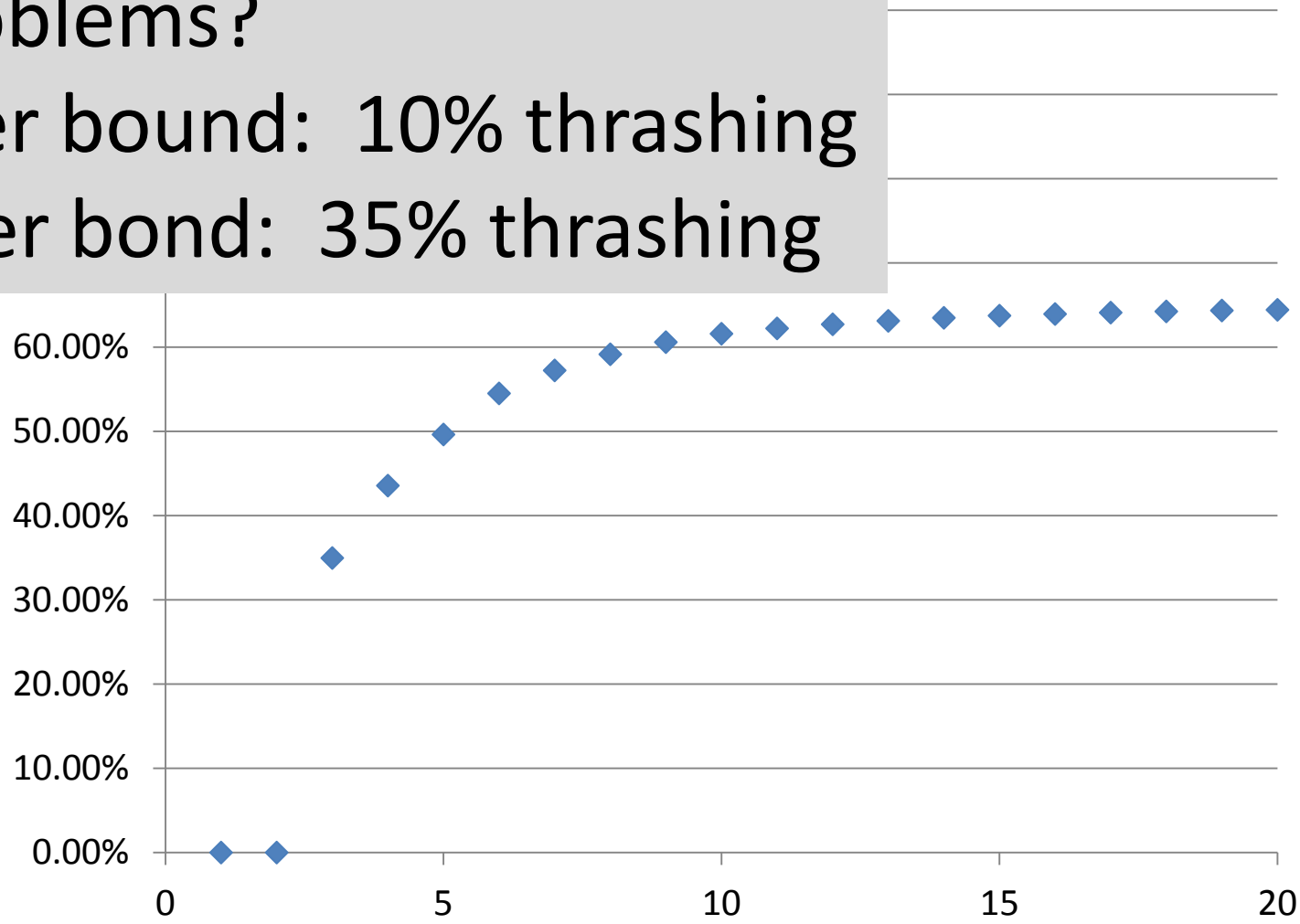


Important caveat of graph!

How to count students who only saw 7 problems?

lower bound: 10% thrashing

upper bond: 35% thrashing



Discussion: thrashing is *actionable*

- KDD cup focused on predicting next student action
  - To be fair, so did I (UM conference, 2003)
  - Is a common approach to validating a student model
- In hindsight: **who cares?**
- Let's play a game

Pretend you have an oracle

- Will tell you whether the student's next response will be correct or incorrect
  - Essentially the KDD cup challenge
- What would you do with it?
  - ???

Pretend you have an oracle

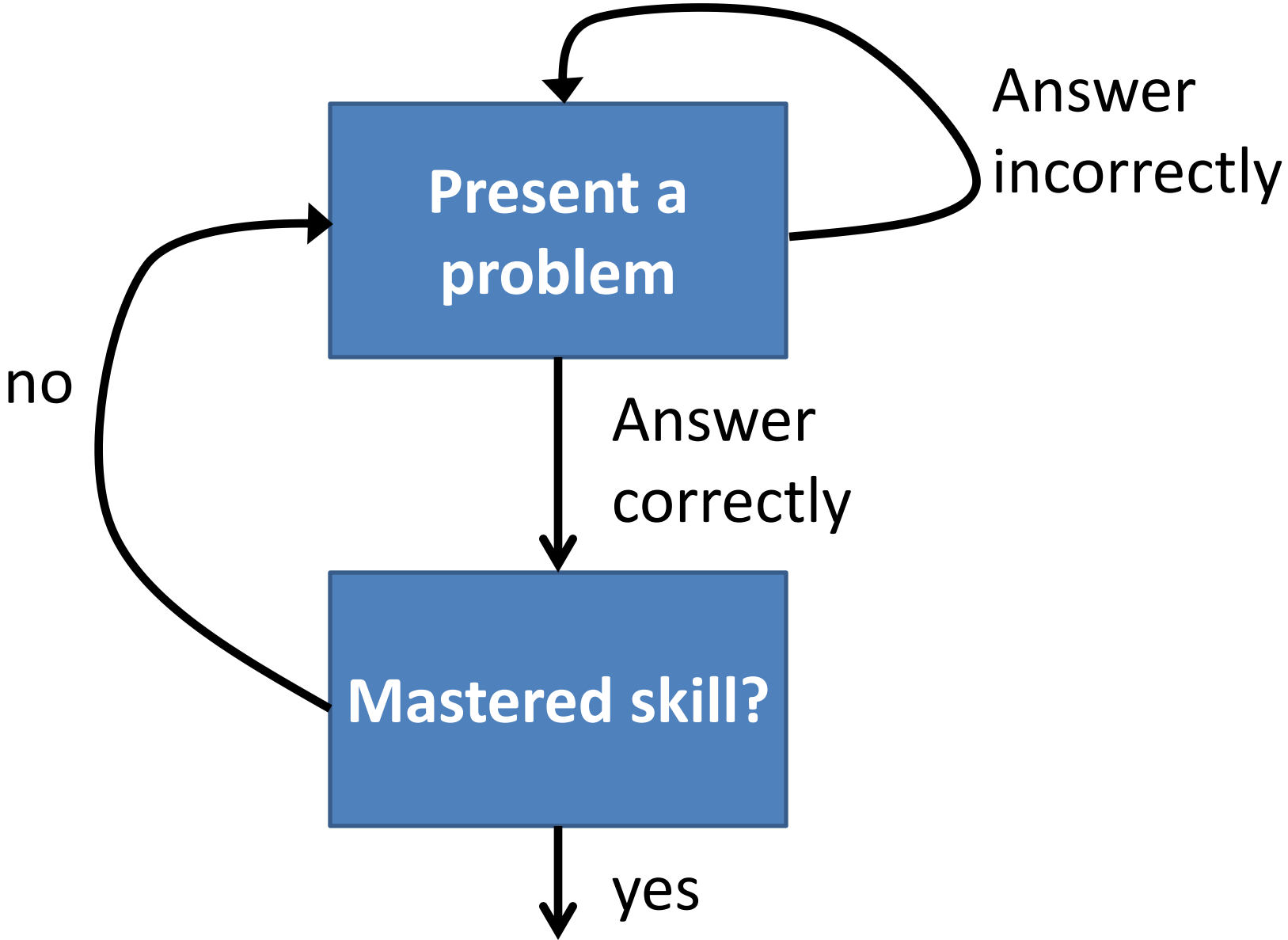
- Will tell you whether the student's next response will be correct or incorrect
  - Essentially the KDD cup challenge
- What would you do with it?
  - ???
- **Feels like a worthless question to answer**

Let’s play a different game

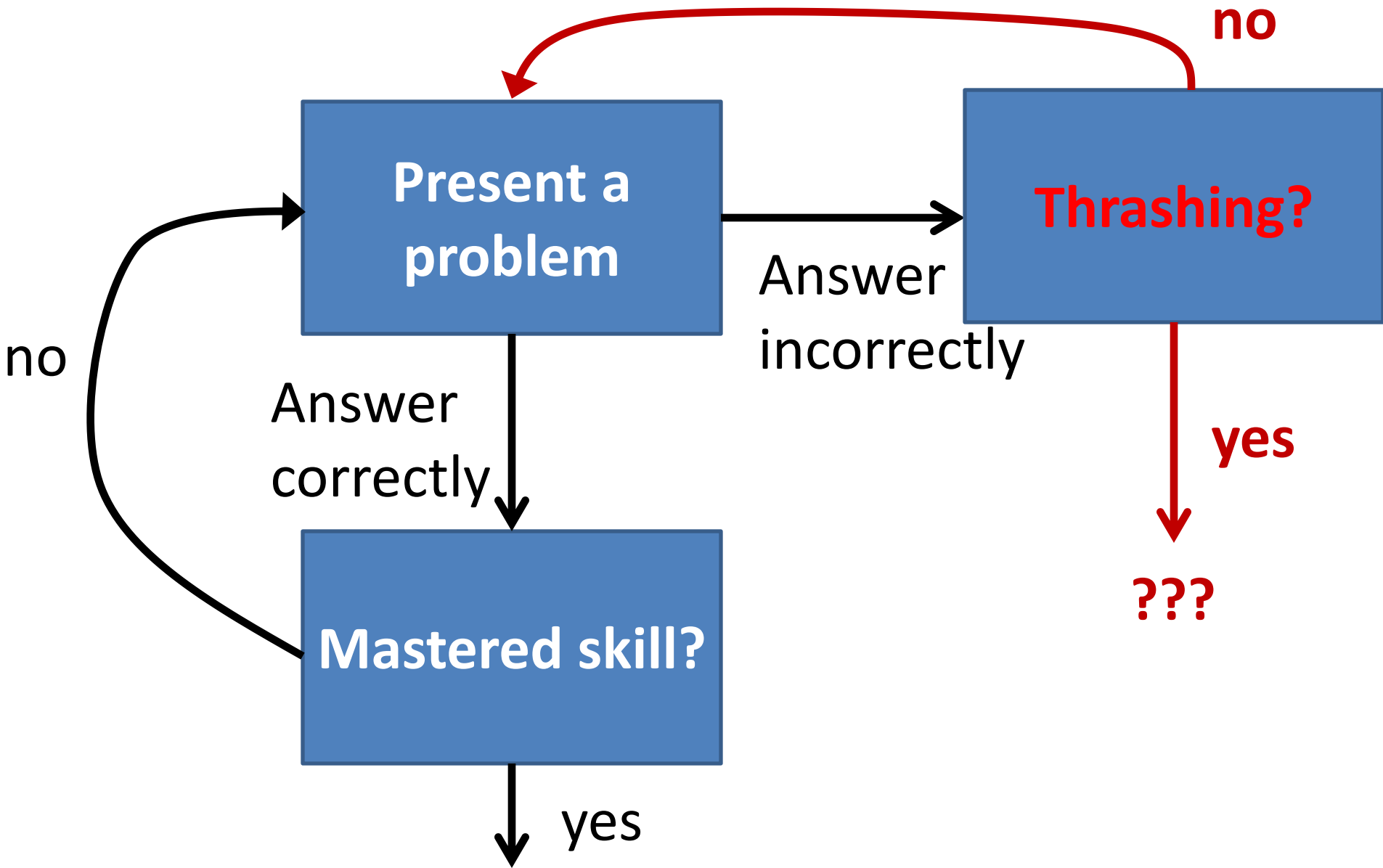
- Pretend you know whether the student will thrash on this skill

- What could you do with it?

Recap: mastery learning



Extending the framework



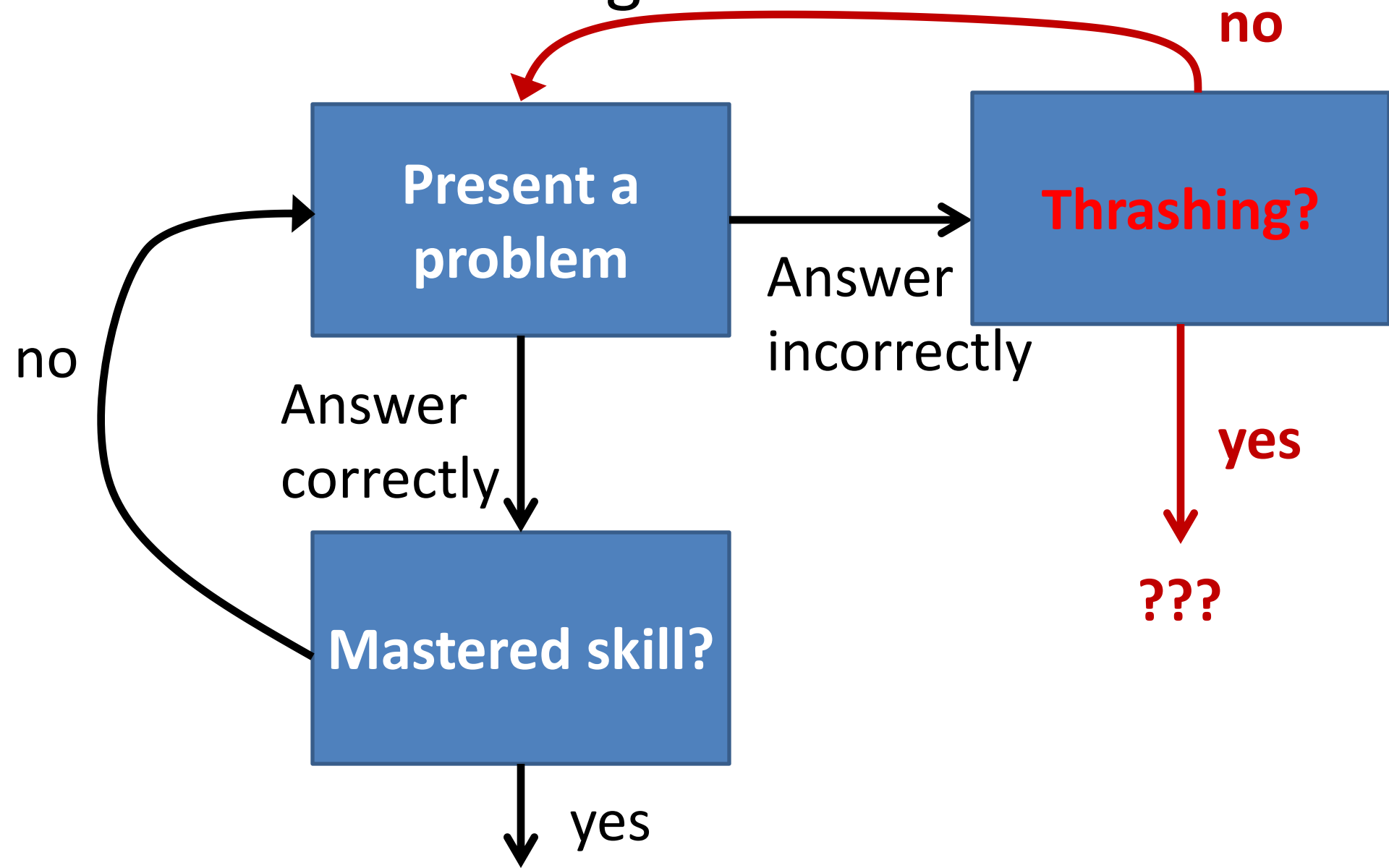
What if student is predicted to thrash?

- The student is (probably) going to fail to master this skill
  - No point in giving another problem
  - The definition of insanity…
- So what to do?

Have the student do *anything* else

- Have some method for teaching the topic
- Work on a different assigned topic
- Work on a prerequisite skill
- Stop using the tutor

Purists would say this **is** the mastery learning framework



Future work

- Decide if thrashing detector is accurate enough for inclusion in tutor
  - Depends on what action is in event of trashing
- Better validate thrashing on other systems
  - What is range of systems that exhibit it
- Disentangle causality with gaming / bad behavior and thrashing

Conclusions

- Like graph representation for evaluating ITS
- Thrashing is an important concept we have ignored (dirty secret of computer tutors)
- Thrashing is *actionable* student modeling
  - Has clear implications for tutor behavior
- The definition of insanity is…

- Do students care less about template questions (ones where just the numbers change?) – another relatively easy paper
- Why don’t we compute interval between bottom out hint and submitting answer?
- Why aren’t we comparing thrashing at home vs. in class?

Paradoxical result with help

- Two students see a word
  - A asks for help and gets it right
  - B does not ask for help and gets it right
- Who is more likely to get it right next time?
  - B

Possible research directions

- Construct the graph representation for some system with which you're familiar
- Build a thrashing detector for another tutor
  - PSLC Data Shop
  - What are the features for building your detector
- Extension: do it on a system where they have published gaming, off-task, disengage, boredom rates (any affective construct)
  - Need to do this for a couple of systems

Bigger projects

- Build a thrashing detector that uses time
- PhD dissertation level
  - Build a thrashing detector
  - Come up with some interventions
  - See how much you improve the system with your detector