Probing student understanding

(Using Lisp, of course)

Christophe Rhodes

17 April 2018

Algorithms & Data Structures

(syllabus)

- big O notation, recurrence relations
- random access machine, higher-order functions
- canon:
  - lists, vectors, linear collections;
  - sorting and searching;
  - string-matching, edit distance;
  - trees, DAGs, graphs;
  - spanning trees, path-finding;
  - (etc.)

HE Context

Student constraints:

- one year of University
- no prior programming experience, nor maths entry requirement
- 120 students in cohort
- by and large, vocational (not academic) motivation

HE Context

Student constraints:

- one year of University
- no prior programming experience, nor maths entry requirement
- 120 students in cohort
- by and large, vocational (not academic) motivation

Institutional constraints:

- fail rate required to be low
  - if you do the work, you pass
- limited budget (my time and 1 TA)

HE Context

Student constraints:

- one year of University
- no prior programming experience, nor maths entry requirement
- 120 students in cohort
- by and large, vocational (not academic) motivation

Institutional constraints:

- fail rate required to be low
  - if you do the work, you pass
- limited budget (my time and 1 TA)

Personal constraint:

- don't want to award a pass to students who don't know anything

(Further HE Context)

(Further HE Context)

(Further HE Context)

Solution

- Force the students to do so much work they can't avoid learning something...

Solution

- Force the students to do so much work they can't avoid learning something...
- ... without making them hate me (or drop out) ...

Solution

- Force the students to do so much work they can't avoid learning something...
  - ... without making them hate me (or drop out) ...
  - ... ideally while allowing most of them to enjoy their time.

Algorithms and Data Structures

Shibboleth? Or necessary?

Algorithms and Data Structures

Shibboleth? Or necessary?

Theoretical

- complexity classes of algorithms
- recurrence relations, proofs by induction
- recursive formulations of linear problems
- loop invariants

Algorithms and Data Structures

Shibboleth? Or necessary?

Theoretical

- complexity classes of algorithms
- recurrence relations, proofs by induction
- recursive formulations of linear problems
- loop invariants

Vocational

- test-driven development
- implementing or using interfaces
- managing code using version control (with merges)
- practice and craft of programming

Learning Management Systems

Support from LMS (Virtual Learning Environment): Moodle

- multiple-choice questions (including multiple right answers)
- short-answer (numeric) questions

GIFT

Text-based format for multiple-choice questions:

```
$$19x^2+5x+6$$ can be described as being in {
  ~%33.3333%$$O(x^2)$$
  ~%33.3333%$$\Omega(x^2)$$
  ~%-100%$$o(x^2)$$ # little-o means "grows much slower than"
  ~%33.3333%$$\Theta(x^2)$$
  ~%-100%none of the other answers
}
```

- hand-editing
- emacs mode: `http://github.com/csrhodes/gift-mode`

Problem:

- need lots of questions (must not be able to learn questions, as opposed to learn material)

Autogenerate

Write a code interpretation problem:

- pretty print the code
- compute the right answer (by evaluating the code)
- compute likely wrong answers
  - think about what students know
  - think about how students guess
  - interpret modified code integrating student mistakes
  - give students feedback about what they don't understand

Generate hundreds of variants for each problem:

- different values
- different statement orders

Obligatory lisp slide

```
(defun make-break-continue-for-form ()
  (let* ((asc (= (random 2) 0))
         (comps (if asc '(< leq) '(> geq)))
         (start (* (maybe-sign) (random 10)))
         (diff (+ (random 10) (random 10) 1))
         (end (if asc (+ start diff) (- start diff))))
    `(progn
       ,(make-form 'setq 'x)
       (for (,start ,(elt comps (random 2)) i
                    ,(elt comps (random 2)) ,end)
            (progn
              (incf x 1)
              ,(if (= (random 2) 0) `(break) `(continue))
              (incf x 1)))
       (return x))))
```

Obligatory second lisp slide

```
(defun return-break-continue-for (n)
  (dotimes (i n)
    (let* ((form (make-break-continue-for-form))
           (answer (interpret-form form))
           (other (interpret-form (sublis '((break . continue)
                                            (continue . break)) form)))
           (neither (interpret-form (sublis '((break . progn)
                                              (continue . progn)) form))))
      (insert (format "::R.%s::" (make-name form)))
      (insert "What is the return value from the following block of pseudocode?
You may assume that the value for all variables before the start of this block
")
      (dolist (x (format-form form))
        (insert (format "  %s<br/>\n"
                        (replace-regexp-in-string " " " " x))))
      (insert (format "{#\n
  =%%100%%%s\n
  ~%%0%%%s#have you mixed up break and continue?\n
  =%%0%%%s#are both increments executed?\n
}\n\n" answer other neither)))))
```

Questions

What is the return value of this block of code? You may assume that the value of all variables before the start of this block is 0.

```
x ← 8
for 4 ≤ i < 16 do
    x ← x + 1
    break
    x ← x + 1
end for
return x
```

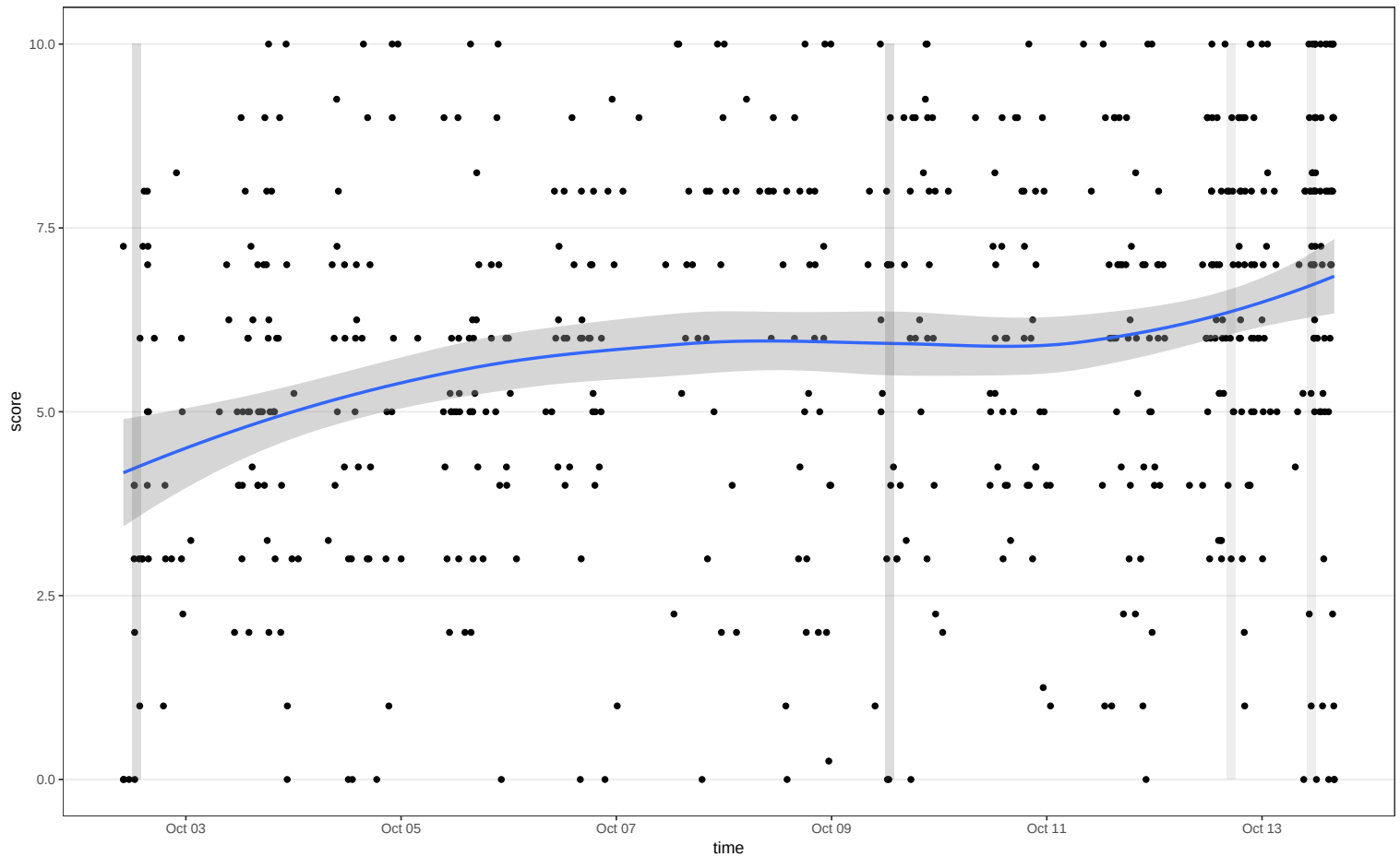
Questions

What code fragment should replace Z for function A to return the difference between a and b? You may assume that the initial arguments to the function A are positive integers and that b ≤ a.

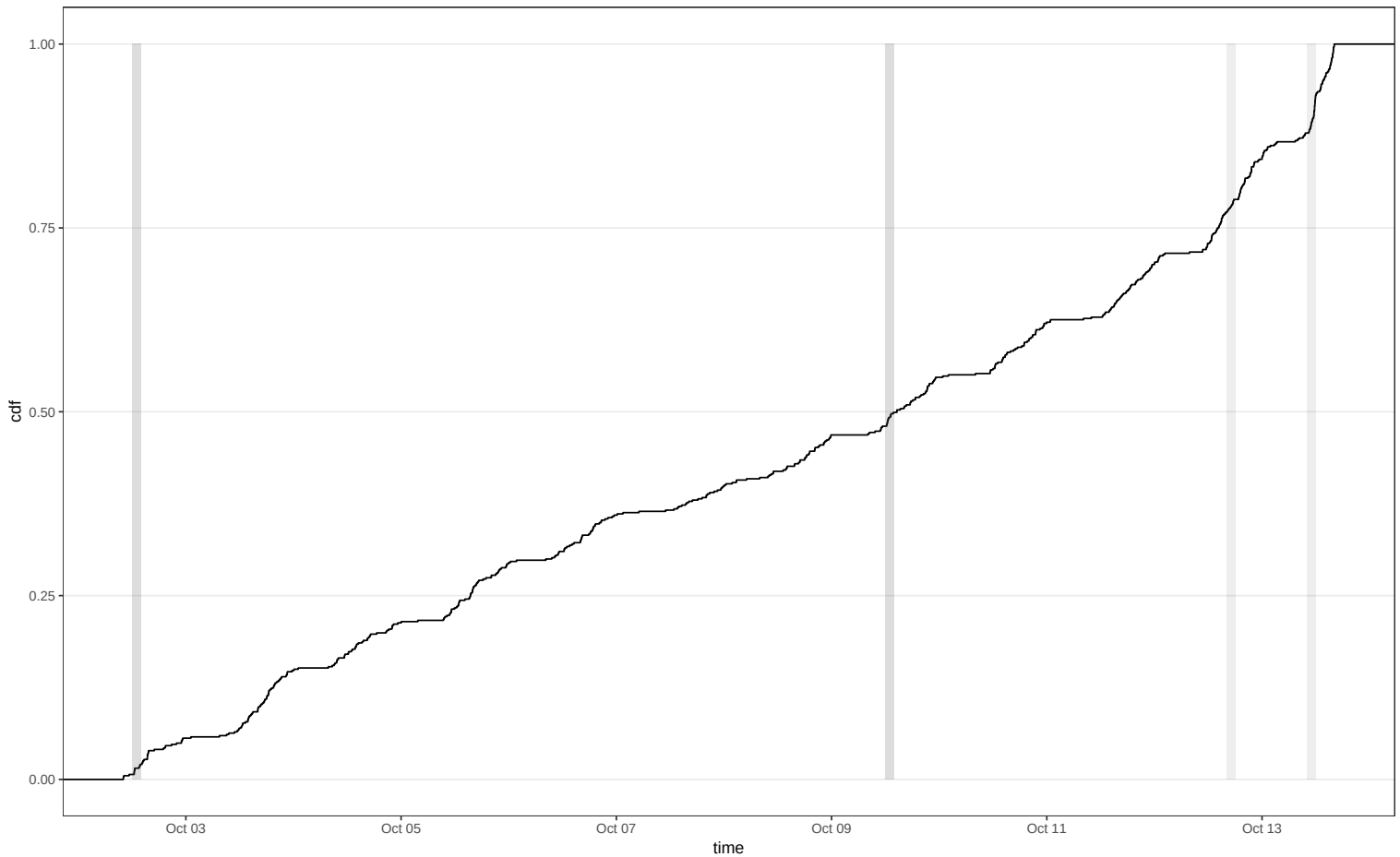
```
function A(a,b)
    if a = b then
        return 0
    else
        return Z + 1
    end if
end function
```

- A(a, b + 1)
- A(a, b - 1)
- A(a - 1, b - 1)
- A(a - 1, b + 1)
- A(a + 1, b - 1)
- A(a + 1, b + 1)
- A(a - b, b)
- none of the other answers

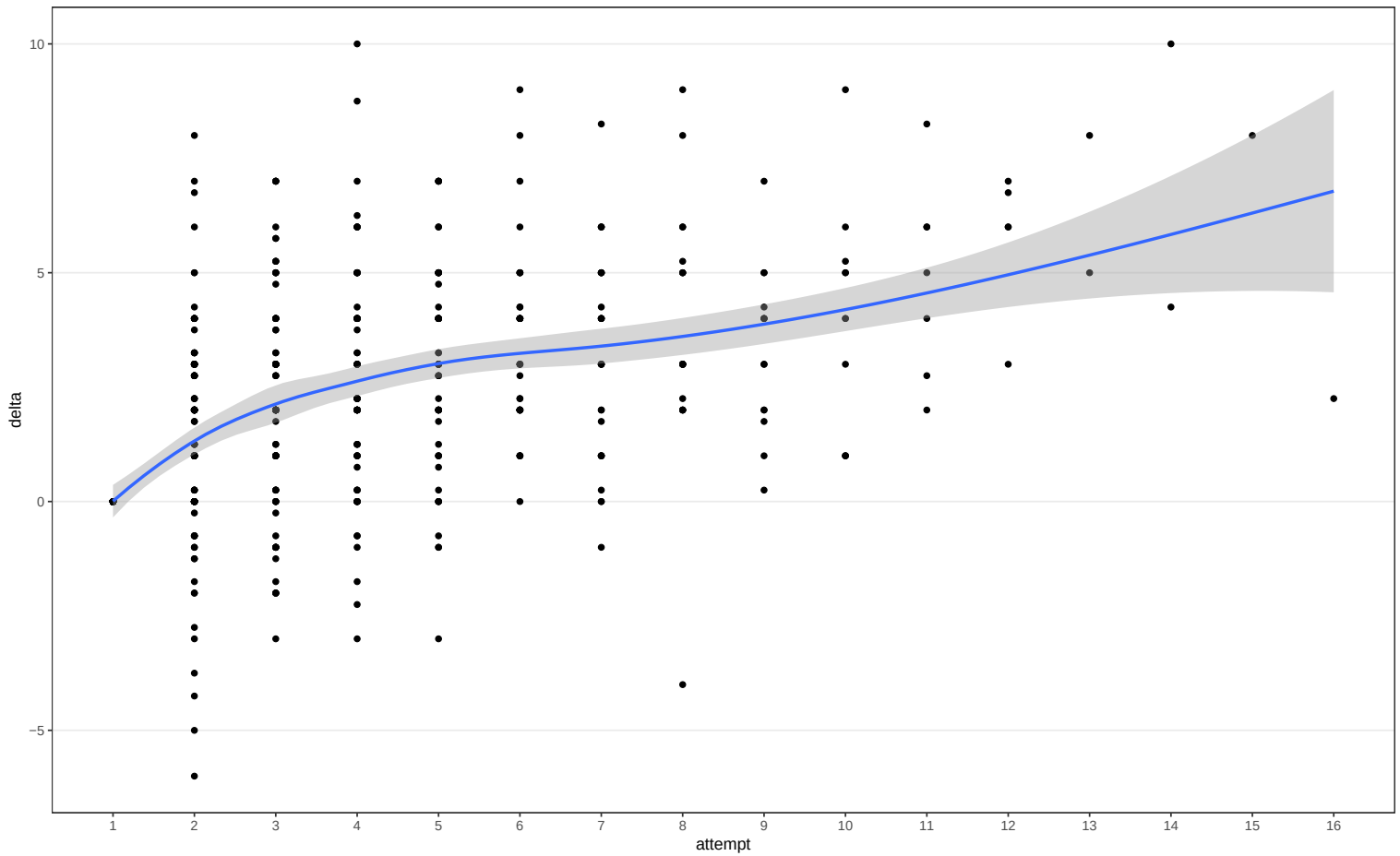
Quiz scores


10.0
7.5
5.0
2.5
0.0
score
Oct 03
Oct 05
Oct 07
Oct 09
Oct 11
Oct 13
time

Quiz attempts



Quiz improvements

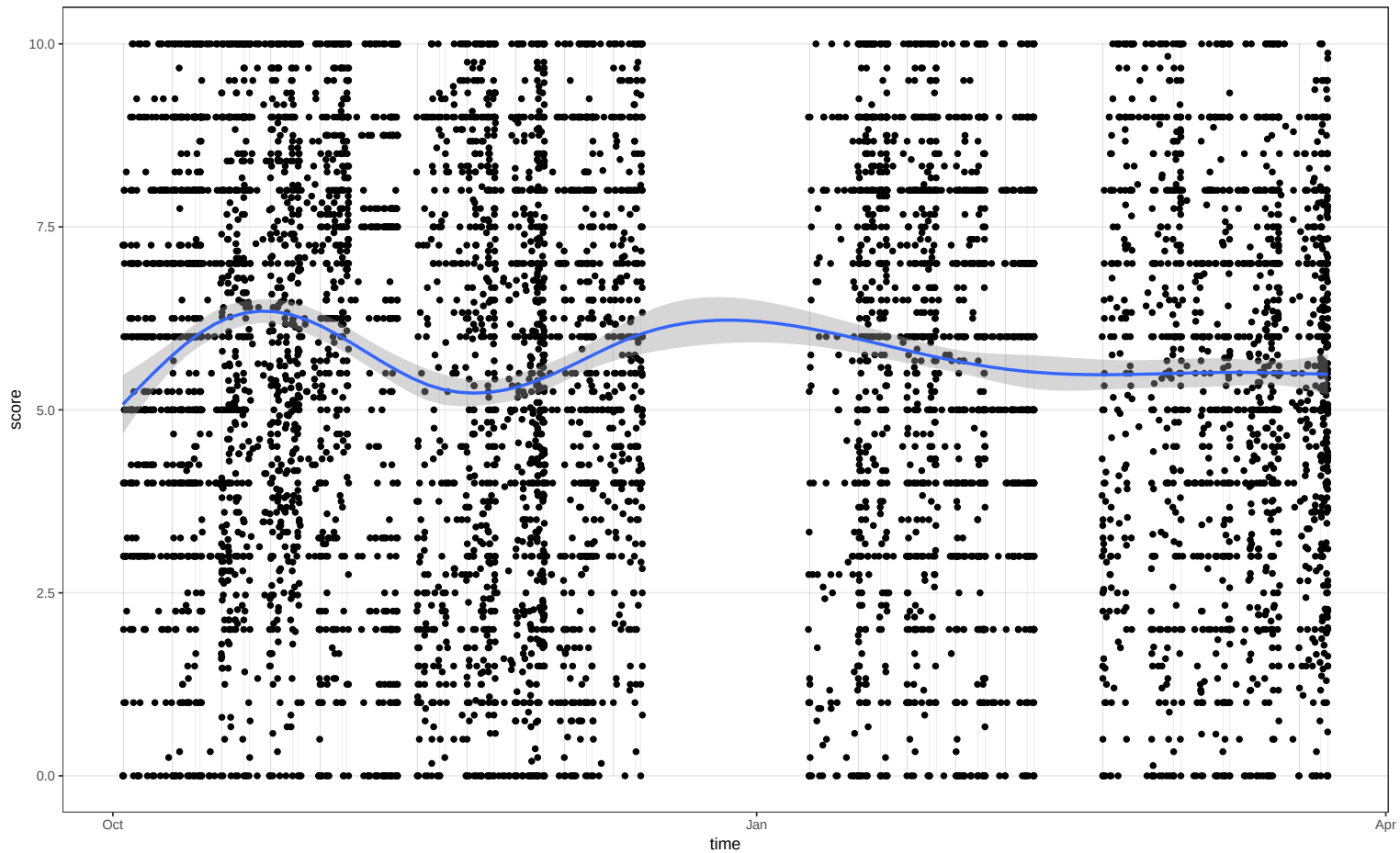


Student activity

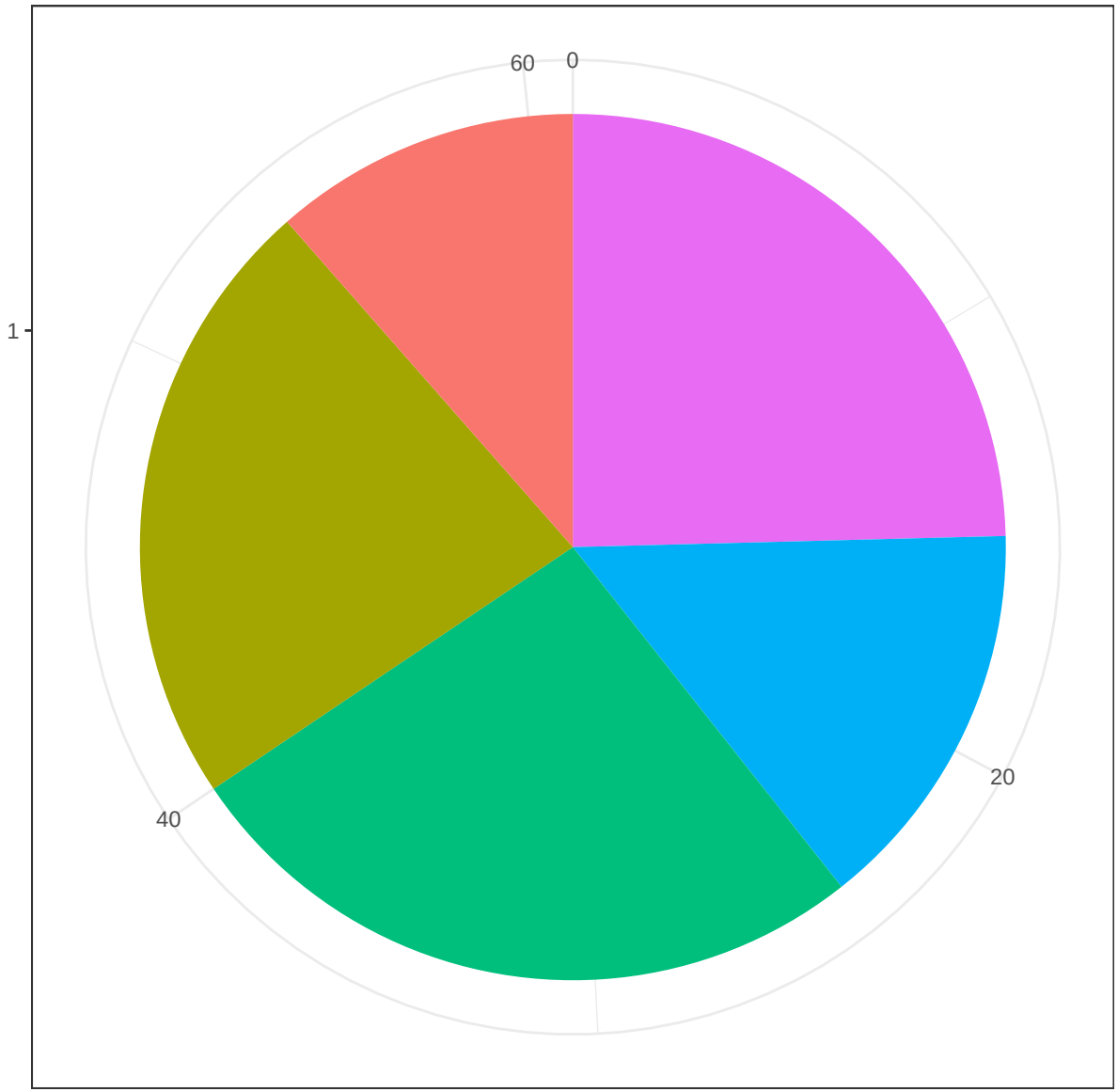
16 quizzes through the year:

- each quiz has 10 questions
  - attempting to probe different atoms of understanding
- around 7000 quiz attempts
- students rarely see the same question twice
- each student has received ~600 pieces of feedback from quizzes

All quizzes

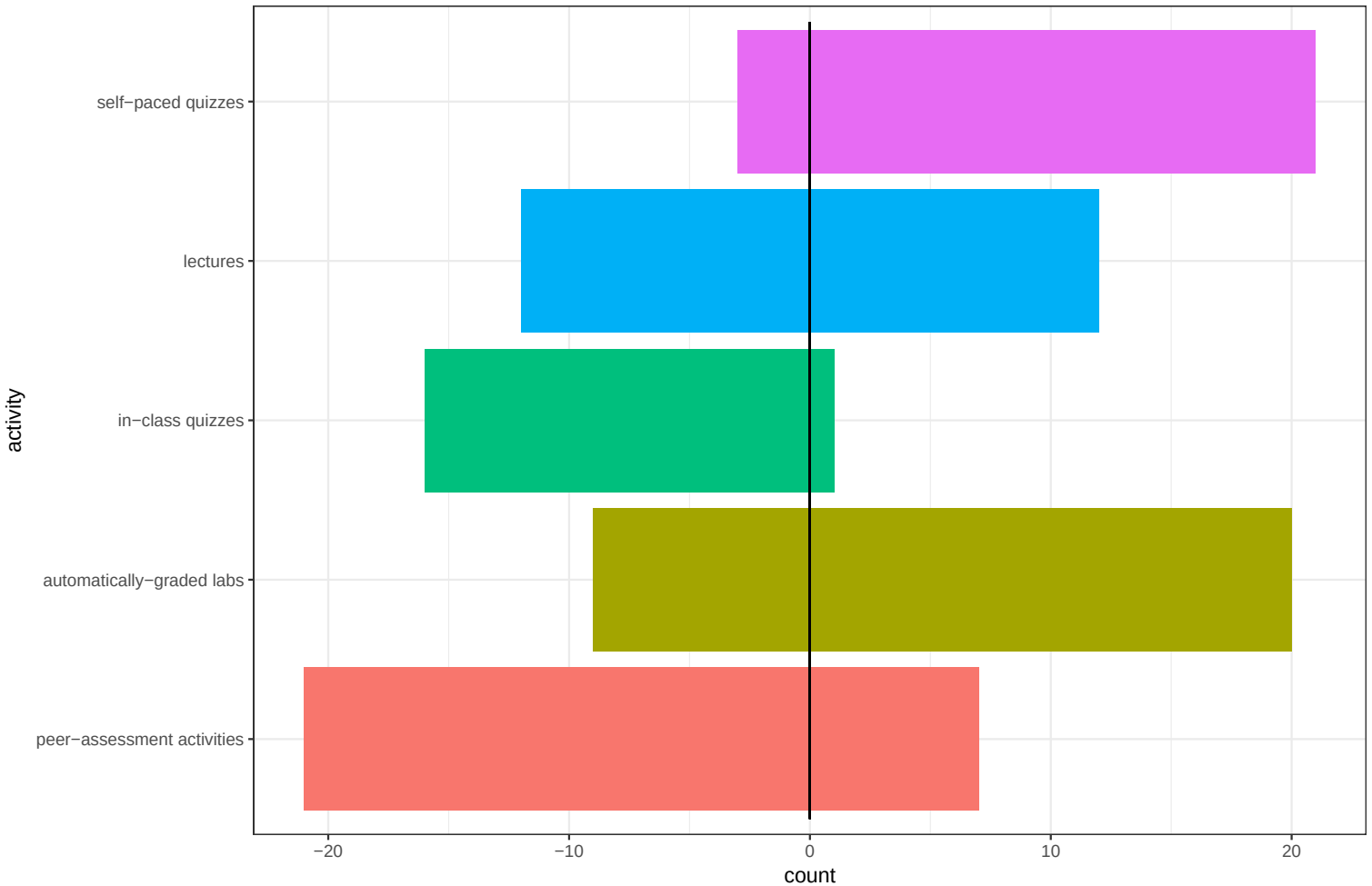


Student feedback



enioved

Student feedback



Teaching and learning

Learning:

- students have displayed increased competence in many areas
  - ”learning gain”
  - proof of pudding: exam, in 4 weeks’ time
  - no control group

Teaching and learning

Learning:

- students have displayed increased competence in many areas
  - ”learning gain”
  - proof of pudding: exam, in 4 weeks’ time
  - no control group

Teaching:

- hard to convince colleagues to adopt this/similar approach
  - increased productivity, but need to learn esoteric tools
  - (sound familiar?)

The future of education?

No, surely not.

The future of education?

No, surely not.
... but:

- framework for students to help each other;
- lessen need for TAs to hand-hold;
- allow students to go (broadly) at their own pace;

The future of education?

No, surely not.
... but:

- framework for students to help each other;
- lessen need for TAs to hand-hold;
- allow students to go (broadly) at their own pace;
- free up expensive instructor time to deliver targeted interventions

The future of education?

No, surely not.
... but:

- framework for students to help each other;
- lessen need for TAs to hand-hold;
- allow students to go (broadly) at their own pace;
- free up expensive instructor time to deliver targeted interventions

Automating myself out of a job?

Thank you!

Questions?