University of Glasgow

Scaling Assessment with Adaptive Comparative Judgement

Sarah Honeychurch
Niall Barr
Jeremy Singer
Steve Draper

Adaptive Comparative Judgement

A method of ranking artefacts by making ***comparative*** judgements, rather than ***absolute*** ones.

- Intuitively plausible
- Removes pretence of expert, objective standards

A radically different approach to grading

- Produces a fully ranked set of scripts
- Allows for separate consideration about where to insert grade boundaries
  - Marking to a curve
  - Marking to rigid standards (e.g. ILOs)

A radically different approach to grading

- Uses a single, implicit criterion rather than a complex, explicit set of ILOs
- Can be used both for questions that do have a single correct answer as well as those that don't

Distinctive Benefits of Pollitt's ACJ Approach

- Method "scales"
- Compelling naturalness
- Can be used with sets of markers
- Can be used for peer review
- Can easily mark cross-media (& multi-media)
- Can easily be used for/with unusual, subjective, and implicit marking criteria
- Can be used by matching against exemplars
- http://www.psy.gla.ac.uk/~steve/apr/apr.html#usp

Adaptive Comparative Judgement

- The software has been built, tested, and used; and by more than one person / organisation. (Also done for conference talk refereeing at UofG.)
- A major experiment has been done and published, using professional markers; supporting the key claims (Pollitt, 2012).
- This paper additionally reports an important qualitative datum: that the markers were highly sceptical (did the experiment for the money, at standard professional rates for marking) but came to see it as better as well as faster than their traditional way of doing marking).

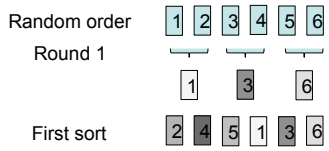
Our ACJ Implementation: the software

- A simple IMS LTI application that can be linked from Moodle, FutureLearn or any other LTI host.
- Submissions can be text, source code, PDFs, images or YouTube URLs.
- Submissions can be added by staff for a review only exercise, or by each student.
- Like *Moodle Workshop* and *Aropä*, it has separate submission and review phases

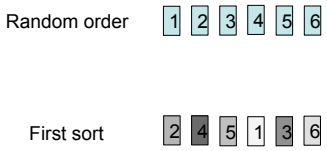
Our ACJ Implementation: the algorithm

- Sorting done in 'rounds'
- New pairing allocated at start of each round
- Three different phases, each with a different 'scoring' method as sort improves
- A simulation (using random errors in comparison) was used to refine the algorithm

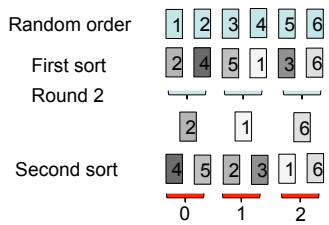
Our ACJ Implementation: the process



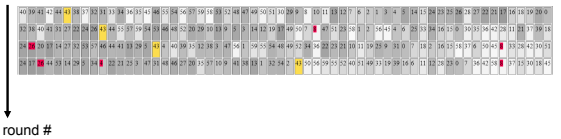
Our ACJ Implementation: the process



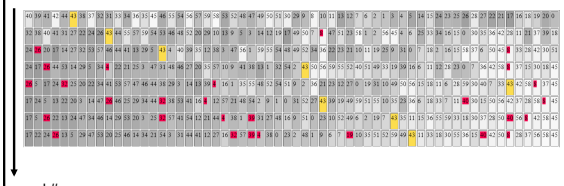
Our ACJ Implementation: the process



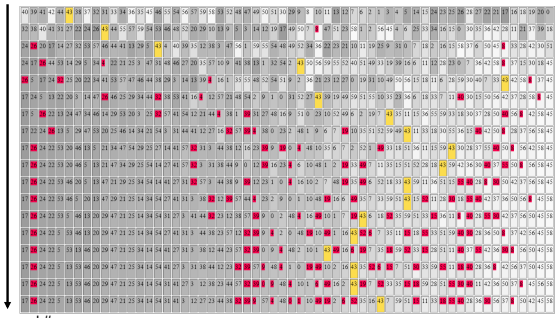
Phase 1: Random Initial Order, Neighbour Comparison, Quartile Bins



hase 2: Using Earlier Judgments to Select New Comparisons

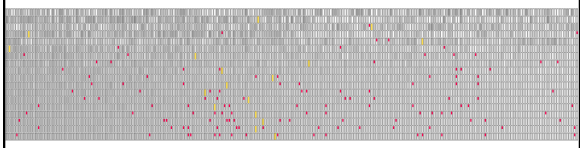


Phase 3: More Refined Comparison with Near Neighbours



Demonstration of Scaling

- The same simulation with 600 ‘artefacts’
- After 17 round sorting is very good
- (Image shows middle ~1/3 with one ‘artefact’ highlighted)



Adaptive Comparative Judgement

https://learn.gla.ac.uk/acjdemo/

This demonstration lets you try out ACJ by comparing photographs of wildlife and flowers. (It uses a development version of the software that doesn’t require a login)

Case Study

Functional Programming in Haskell: Supercharge Your Coding

- Futurelearn MOOC (n=1000)
- COMPSCI4021 (n=80)

Future Learn

Case Study Continued

In the Haskell MOOC, we asked students to peer assess using ACJ.

Students received:
1. Problem spec (to implement)
2. Quality guidelines as judgment criterion
3. Peers’ solutions (to compare)
4. Ranking of their own work (quartile bin)
5. A sample solution

Sample solution

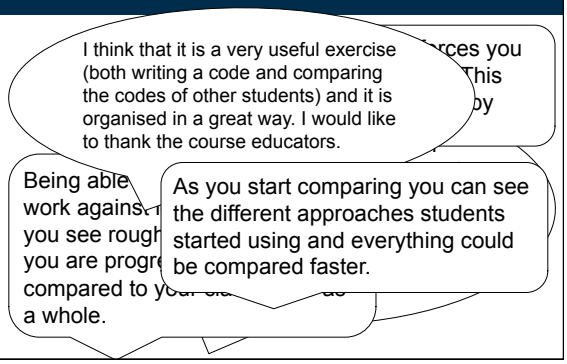
```
-- | The 'wordPhrase' function spells out an individual word
-- For example, "a is for apple"
wordPhrase :: String -> String
wordPhrase x = (head x) : " is for " ++ x


-- | The 'speller' function generates text for a spelling book
-- from a list of words
speller :: [String] -> String
speller [] = []
speller [x] = wordPhrase x
speller [x,y] = xPhrase ++ ", and " ++ yPhrase
  where
    xPhrase = wordPhrase x
    yPhrase = wordPhrase y
speller (x:xs) = wordPhrase x ++ ", " ++ speller xs
```

Student comments

- I can see different ways of thinking and I try to understand which one is better(more efficient) and I hope that I will be able to make my own codes more efficient in the future.
- The approach forces you to think differently. This can only be trained by doing it.
- Being able to compare your own work against lots of others lets you see roughly how well/poorly you are progressing in the course compared to your classmates as a whole.
- I think that it is a very useful exercise (both writing a code and comparing the codes of other students) and it is organised in a great way. I would like to thank the course educators.
- As you start comparing you can see the different approaches students started using and everything could be compared faster.

Student comments about their experience of ACJ



Interesting statistics

Can be set up to produce reports:

- Who was the most deviant marker?
- Which submission was the most divisive?
- How converged were the judgements?

Where next?

- Still a development / pilot tool
  - Further refinement possible
- Could this be useful in your teaching?
  - Scholarship / research
  - Not a ‘Service’

ACJ Software Components

Display User Interface	Input User Interface
Windows, panes for multiple docs, etc	Note taking, feedback private bank of comments
Stats Calculations	**Decision Making**
Estimates of order & distances	Select pairs for comparison

References

- Dale, V.H.M., & Singer, J., 2019. Learner experiences of a blended course incorporating a MOOC on Haskell functional programming. *Research in Learning Technology* vol.27. DOI: 10.25304/rlt.v27.2248
- Pollitt, A., 2012. The method of Adaptive Comparative Judgement. *Assessment in Education: Principles, Policy & Practice*, 19(3), pp.281–300.
- Thurstone, L. L., 1927. A law of comparative judgment. *Psychological Review, 34*(4), pp.273-286. http://dx.doi.org/10.1037/h0070288

Pointers

Sarah.Honeychurch@glasgow.ac.uk @NomadWarMachine
Niall.Barr@glasgow.ac.uk @niall_barr
Jeremy.Singer@glasgow.ac.uk
Steve.Draper@glasgow.ac.uk

Source code: https://github.com/niallb/ACJ-LTI
Further notes: http://www.psy.gla.ac.uk/~steve/apr/apr.html
This talk: http://www.psy.gla.ac.uk/~steve/talks/apr4.html

Minimise Incidental Complexity of Marking

- Several colleagues use ad-hoc software lash-ups to increase marking speed and quality
- e.g. pen input, PDF annotation, cloud storage, multi-window/screen views
- Plimmer (2006) systematically explores the possibilities for paperless, pen-based marking
- Our proposed UI components would be informed by their experiences
- Plimmer, B. & Mason, P., 2006. A pen-based paperless environment for annotating and marking student assignments. *7th Australasian User Interface Conference*.