Automated Assembly of Item Pools Using Linear Programming

Jeffrey Patton, *Psychometrician*
Ray Yan, *Director, Measurement Services*
Financial Industry Regulatory Authority (FINRA)

FINRA
Financial Industry Regulatory Authority

Who We Are

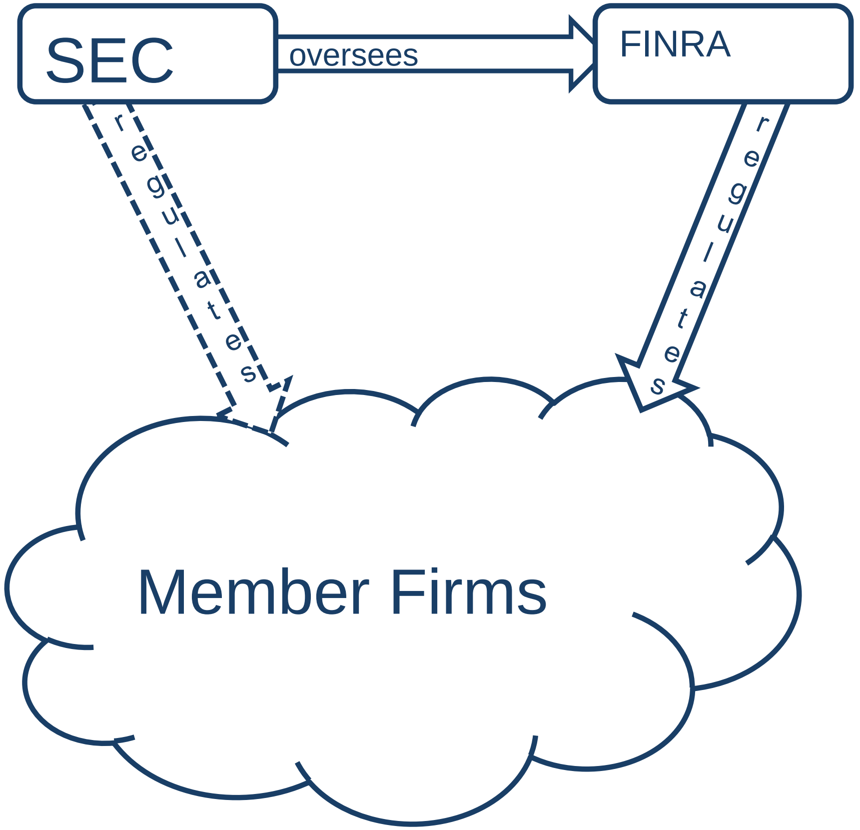
- The Financial Industry Regulatory Authority (FINRA) is an independent, nongovernmental regulator for all broker-dealers doing business with the public in the U.S.
- FINRA protects investors by regulating brokers and brokerage firms and by monitoring trading on U.S. stock markets.

Regulatory Bodies of US Financial Markets

SEC – Government agency that enforces federal securities laws

FINRA – Membership-based organization that creates and enforces rules for member firms based on federal securities laws

Member Firms – Responsible for monitoring the actions of their representatives.



What We Do

- **We oversee about 3,900 brokerage firms, nearly 160,000 branch offices, and nearly 635,000 registered securities representatives.**
  - Every firm and broker that wants to do business with the U.S. public must be licensed and registered with us and must follow our rules.
  - All brokers must pass our qualification exams and satisfy continuing education requirements.
- **We examine broker-dealers for compliance with our own rules and with federal securities laws.**
- **We monitor 99 percent of all trading in U.S. listed equities markets, processing an average of 37 billion market transactions per day.**

Concerning FINRA Exams

- We administer and maintain 28 high-stakes qualification exams.

- Our goal is to ensure that each candidate has sufficient knowledge of the securities industry, its markets, and its regulations.

Concerning FINRA Exams

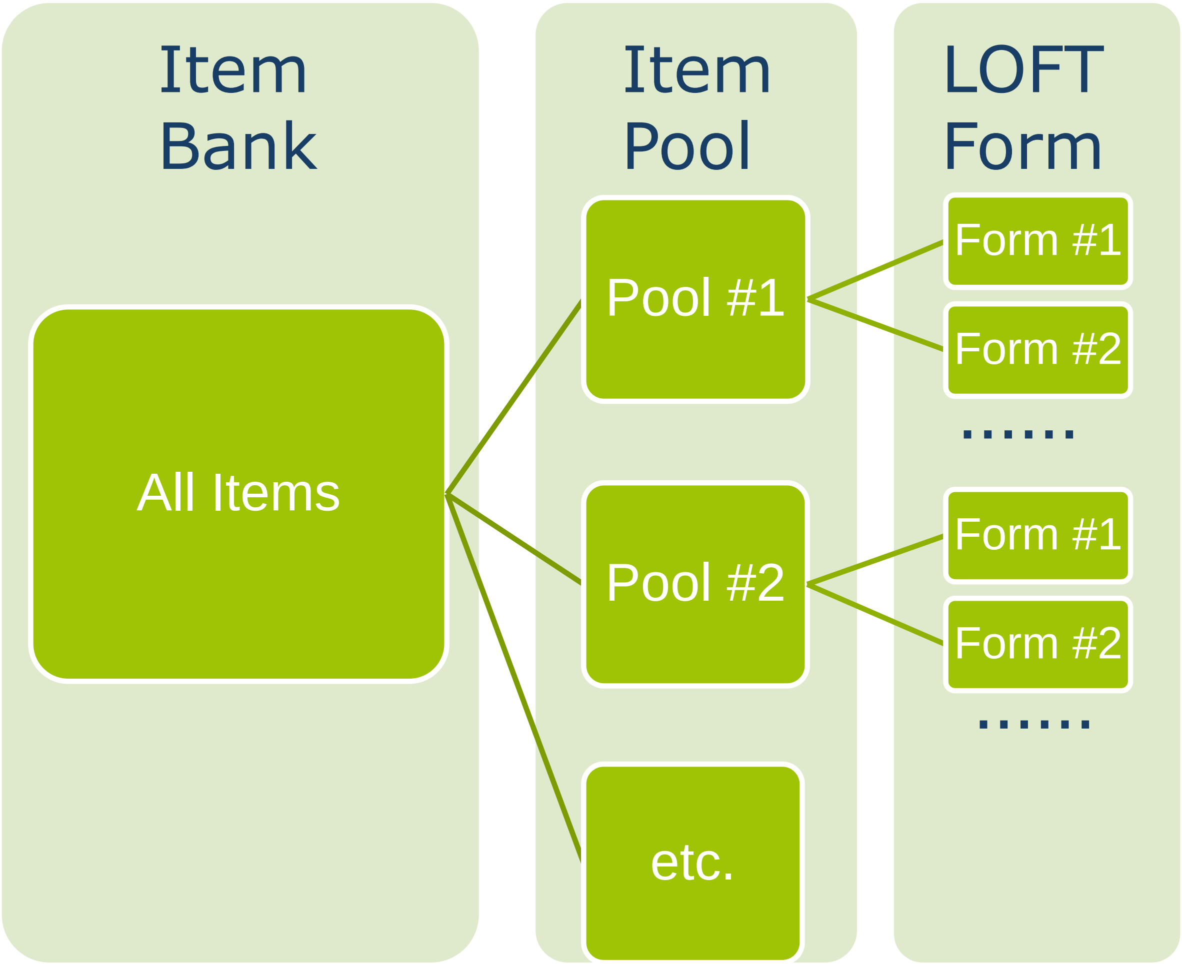
- Many exams employ linear-on-the-fly testing (LOFT).
- Before a candidate’s appointment, a test form is assembled “on the fly” from the item pool.
- Periodically, a new item pool is assembled from the item bank.

Item Pool Assembly

- A program’s item “bank” refers to the set of all available items, in all stages of development.
- The items used to assemble a test form may be drawn directly from the bank.
- However, it may be preferable to assemble a subset of operational items from the bank — an item “pool.”
- Only those items in the currently active pool are eligible to appear on a test form.

FINRA

Item Pool Assembly



Why bother assembling an item pool?

- **It's an aggressive form of exposure control.**
- **You can construct a pool with properties that resemble the desired properties of test forms.**
- **Security! An organized attempt to harvest items would compromise only the active pool. The remaining items in the bank would be untouched.**
  - We don't want all the same items to appear on test forms – use *item exposure control.*
  - Similarly, we don't want all the same items to appear in different item pools – use *item selection control.*

Goals

- **Our goal today is to share what we've learned about item pool assembly and "rotation."**
- **You can use a heuristic method to assemble pools (*i.e.*, sequential selection of items).**
- **We've chosen to use linear programming (*i.e.*, simultaneous selection).**
  - Much easier to realize complex or numerous constraints

Outline

- **Automated Test Assembly (ATA)**
  - Overview
  - Example
- **Item Pool Assembly**
  - Assemble entire pool vs. partitions
  - Simultaneous vs. sequential assembly
  - How to control item “selection” rates

Automated Test Assembly: Why Do It?

- **Often it is insufficient for a test form to simply meet a set of constraints (*e.g.*, content blueprint).**

- **You may want the form to be optimal in some way:**
  - Maximize Fisher information at the passing score
  - Minimize the number of MC items that have five response options
  - Maximize the number of items that can be administered in a given time limit

- **If the item pool is large, finding the optimal form manually would be quite laborious.**

Automated Test Assembly: Why Do It?

- **With linear programming (LP), a computer can:**
  - efficiently search among all possible item sets that meet the constraints.
  - find the set that optimizes the criterion of interest.

- **LP is a tool built for combinatorial optimization.**

- **LP has been around in mathematics since at least the 1940s; only in the last few decades has it been applied to measurement.**
  - van der Linden, W. J. (2005). *Linear models for optimal test design*. New York: Springer.

Automated Test Assembly: How To Do It?

1. Make a list of all constraints that the test form must meet. Decide what the objective function should be (*i.e.*, the criterion to be optimized).

2. "Translate" constraints and objective function into linear equations/inequalities; provide these and the item bank to the LP solver.

3. The solver will return the optimal set of items (within some tolerance) that also meets all constraints.

Automated Test Assembly: An Example

- **Suppose we have a pool of 16 items; each item belongs to content area A or B.**

- **We want to assemble a five-item test.**
  - Two items from content area A
  - Three items from content area B

- **We want an expected test score of three at $\theta = 0.3$.**

Automated Test Assembly: An Example

Item	Content Area	$P_j(\theta = 0.3)$
1	A	.30
2	A	.41
3	A	.27
4	A	.68
5	A	.49
6	A	.51
7	A	.70
8	B	.61

Item	Content Area	$P_j(\theta = 0.3)$
9	B	.42
10	B	.36
11	B	.66
12	B	.69
13	B	.65
14	B	.47
15	B	.80
16	B	.43

Automated Test Assembly: An Example

- **First, we define the necessary decision variables, one for each item in the pool:**

$$x_j = \begin{cases} 1 \text{ if included in the test} \\ 0 \text{ otherwise} \end{cases} \qquad j = 1, 2, \dots, 16$$

- **Any constraint will be expressed as a linear combination of these variables.**

- **The LP solver will return optimal values of the decision variable (*i.e.*, the optimal set of items that also meets all constraints).**

Automated Test Assembly: An Example

- **First constraint: Two items must come from content area A.**

- **Coefficients:**

$$k_{jA} = \begin{cases} 1 \text{ if item } j \text{ belongs to A} \\ 0 \text{ otherwise} \end{cases}$$

- **Linear combination and constraint:**

$$\sum_{j=1}^{16} k_{jA} x_j = 2$$

Automated Test Assembly: An Example

- **Next constraint: Three items must come from content area B.**

- **Coefficients:**

$$k_{jB} = \begin{cases} 1 \text{ if item } j \text{ belongs to B} \\ 0 \text{ otherwise} \end{cases}$$

- **Linear combination and constraint:**

$$\sum_{j=1}^{16} k_{jB} x_j = 3$$

Automated Test Assembly: An Example

- Finally, for the objective function, we need one more decision variable: x_{17}
- With the following constraints, we define x_{17} as the (positive) difference between the target expected score (3) and the actual expected score:

$$\sum_{j=1}^{16} P_j(\theta = 0.3)x_j \geq 3 - x_{17}$$

$$\sum_{j=1}^{16} P_j(\theta = 0.3)x_j \leq 3 + x_{17}$$

- Our objective function is simply to minimize x_{17}

Automated Test Assembly: An Example

Decision Variable	Solution
x_1	0
x_2	1
x_3	0
x_4	0
x_5	0
x_6	0
x_7	1
x_8	1

Decision Variable	Solution
x_9	0
x_{10}	0
x_{11}	0
x_{12}	0
x_{13}	0
x_{14}	1
x_{15}	1
x_{16}	0
x_{17}	.01

Software

- **Commercial software is quite expensive.**
- **An excellent free option: "lpSolveAPI" R package**
  - Built-in LP solver "lp_solve" (used to be stand-alone)
  - Not specific to measurement but still straightforward
  - Sufficient for small- to moderate-sized problems
- **Specify the number and scale of decision variables**
- **Add constraints one by one**

Outline

- **Automated Test Assembly (ATA)**
  - Overview
  - Example

- **Item Pool Assembly**
  - Assemble entire pool vs. partitions
  - Simultaneous vs. sequential assembly
  - How to control item “selection” rates

Item Pool Assembly

- **Lots of available literature concerning test assembly, for example:**
  - Constrain the overlap among *n* test forms
  - Control how much time a candidate will likely need to finish a test
  - The “shadow test” approach to adaptive testing

- **Much less literature on item pool assembly**
  - Though many of the principles of test assembly are applicable
  - Any pool assembly procedure that could support a test program is probably proprietary

Unique Issues with Item Pool Assembly

1. **If there are lots of constraints, should the pool be assembled piecemeal or all at once?**

2. **If you intend to rotate through pools periodically, should you build multiple pools simultaneously? Or should they be built sequentially?**

3. **What's a good way to control item "selection" rates (*i.e.*, the proportion of pools in which an item appears)?**

Assemble Partitions or the Entire Pool?

- **For a fixed bank size, the number of possible pools will (almost certainly) be much larger than the number of possible test forms.**

- **For example, with a 1,000-item bank:**
  - Number of unique 50-item tests: $\binom{1{,}000}{50} = 9 \times 10^{84}$
  - Number of unique 400-item pools: $\binom{1{,}000}{400} = 5 \times 10^{290}$

- **The more item sets to search through, the longer the LP solver is likely to take!**

Assemble Partitions or the Entire Pool?

- **Using lp_solve, assembling a pool with several hundred items can take a while — or it might not even work!**
  - How can you speed things up?
- **Partition the item pool (*e.g.*, by content area):**
  - Assemble the entire pool at once, but specify constraints separately for each partition, or
  - Assemble each partition separately.
- **Adjust parameters in the solver's algorithm:**
  - 2.00000001 can be stored as an integer.
  - If the maximum information at a point, for example, is actually 30.34674, is 30.3 high enough?

Simultaneous vs. Sequential Assembly

Simultaneous

- **PROS:**
  - Easier to control inter-pool properties (conceptually at least)
- **CONS:**
  - Problem may become too large.
    - Have constraints for each pool & among pools
    - For each partition, could do simultaneous assembly
  - Bank changes over time (promote pretest items, retire old/bad items)

Sequential

- **PROS:**
  - Can react quickly to changes in the bank
  - Assembly problem is more manageable
- **CONS:**
  - Not as straightforward to implement constraints between pools

Item “Selection” Control

Exposure Rates vs Selection Rates

- **Suppose we want to assemble test forms that have maximum information at the passing score.**
  - We want each form to have lots of informative items at the passing score.
  - But we don't want the "best" items to be overexposed.
  - That is, we want item exposure rates to be as even as possible and average form overlap to be low.

- **There are analogous concerns with pool assembly!**
  - Let's say that we want lots of informative items at the passing score.
  - But we want item "selection" rates to be as even as possible, and we want average pool overlap to be low.
  - We want the "best" items to be spread evenly across pools and don't want systematic differences among pools.

Exposure Rates vs Selection Rates

- **We say “selection” rates because an item in a pool will not necessarily be exposed (and to differentiate it from exposure rates).**

- **Often, you’ll want to use an aggressive method to ensure that selection rates are as even as possible.**
  - An item must appear in a pool to have a chance to appear on a form.

- **We’ll go through four different ways to control item selection rates.**

Overlap Constraints

- **Explicitly state the maximum allowed overlap between two item pools.**
- **Of course, lowest possible level of overlap depends on bank size & number of item pools.**

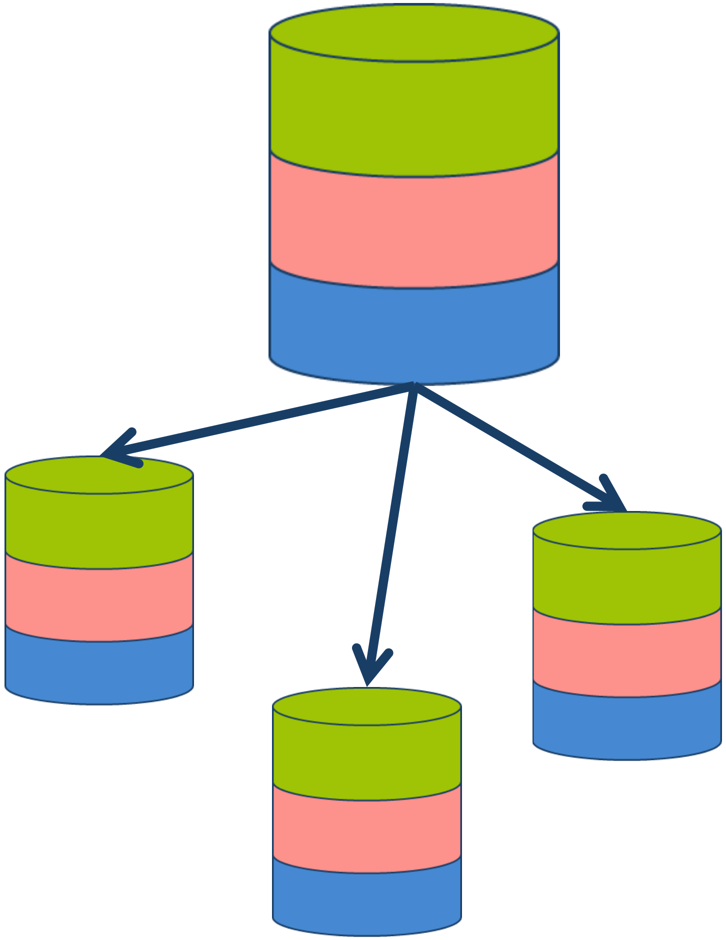
Overlap Constraints

- **Simultaneous assembly:**
  - Must put constraint on all unique pairs of pools that you're assembling
  - Makes the assembly problem much more complex
  - Assembling pools with zero overlap is simpler but impractical

- **Sequential assembly:**
  - Must keep track of the composition of the preceding *N* pools and specify constraints with each one
  - Much less complex constraints
  - But might run into infeasibility

Bank Stratification

- **Analogous to *a*-stratification!**
  - Organize items into strata and put constraints on how many items should come from each.
- **Easy to implement with either assembly method but requires:**
  - Intimate knowledge of the bank; might run into infeasibility
  - Fine-tuning: must conduct simulations to estimate item selection rates and pool overlap



Selection Penalties

- **Can only be used with sequential assembly**
- **Keep track of the composition of the preceding *N* pools**
- **Instead of maximizing information at the passing point, for example, maximize a penalized function of information:**
  - If selected last month, item receives a large penalty
  - If last selected a year ago, item receives no penalty
- **Likely need to fine-tune the penalty function**

Randomesque

- Analogous to the randomesque item exposure control method; can be used with either pool assembly method
- Instead of maximizing information, maximize a “noisy” function of information
- For example: take the log of information and multiply by a random number from a U(0,1) distribution
- Again, will need to fine-tune the function

Questions and Answers

Thank you for your time!

Jeffrey.Patton@FINRA.org

Ray.Yan@FINRA.org