Equitable Lunch Invites

February 12, 2026

1 What this planner is doing

Hi, this was written to aid in the planning of lunches with many people from various disciplines to ensure equity and fair representation.

We want to schedule a *series* of lunch events. Each event picks:

- a fixed number of hosts (usually 1), and
- a fixed number of guests (plus an optional waitlist).

The plannercarries forward what happened in earlier events. We aim to be fair (rotate people), robust to no-shows (down-prioritize unreliable invitees), and be balanced across demographic categories and disciplines.

2 Entities and state

2.1 Participants

Each participant has:

- a **name**,
- a **discipline** from a fixed allowed list (the planner can auto-fill missing disciplines),
- an optional **demographic category** (e.g. sex); missing/unknown values are treated as category `U`.

Participants appear in two roles: **hosts** and **guests**.

2.2 Per-role stats

For each role (hosts, guests), the state stores per-person statistics:

$$\text{stats}(i) = (a_i, n_i, c_i)$$

where:

- a_i (`assigned_count`) is the number of times person i has been assigned for that role,
- n_i (`no_show_count`) is the number of times person i no-showed for that role,
- c_i (`cooldown`) is a small integer penalty window after a no-show.

The planner compares people by the *fairness / reliability score key*

$$\text{score}(i) = (a_i, n_i)$$

and always prefers smaller tuples lexicographically: first fewer prior assignments, then fewer no-shows.

3 Deterministic seeds (reproducible tie-breaks)

You provide an integer **seed key** $k \in [0, 1000]$. The CLI derives deterministic seeds by hashing a labeled string. The planner then creates separate pseudo-random generators for hosts and guests.

These RNGs are used only to generate *per-person tie-break numbers* $t_i \in [0, 1)$. When two candidates are otherwise equal, the smaller t_i wins.

4 Input preprocessing that affects the math

4.1 Auto-filling missing disciplines

If a roster row has a blank discipline, the program fills it in by cycling through the canonical discipline list. If the canonical list has length L, the m-th missing row (in file order) receives discipline

$$\text{DISCIPLINES}[m \bmod L].$$

This is way to avoid dropping those participants and spreads them evenly through the disciplines

4.2 Demographic normalization

If the demographic column is named `Sex`, then values are normalized to:

`F`, `M`, or `U` (unknown).

For other demographic columns, blank values become `U`; short tokens are uppercased.

5 The two allocation primitives

The planner uses two allocation methods, each for a different job:

- a **randomized proportional allocator** for building the guest cohort (it uses randomness only to break ties when multiple valid allocations exist), and
- a **deterministic apportionment allocator** for setting per-event demographic targets across the series (same inputs always give the same targets).

5.1 Randomized proportional allocation with minimums and caps

This routine takes:

- a total number of seats T,
- nonnegative integer weights w_k over categories k,
- minimums m_k and caps M_k (both clipped to be nonnegative, with $m_k \le M_k$),
- an RNG for tie-breaking.

It returns nonnegative integers x_k such that $\sum_k x_k = T$ and $m_k \le x_k \le M_k$ whenever feasible.

Step 0: trivial cases. If $T \le 0$, return all zeros.

Step 1: apply minimums. Let $m'_k = \min(m_k, M_k)$ and let $m' = \sum_k m'_k$.

- If $m' > T$ (minimums oversubscribed), the algorithm satisfies as much of the minimums as possible by sorting categories by larger (m'_k, w_k) (with an RNG shuffle used as a tie-break), then greedily taking
$$x_k \leftarrow \min(m'_k, M_k, \text{remaining}).$$
- Otherwise, initialize $x_k \leftarrow m'_k$ and set the remaining seats $R \leftarrow T - m'$.

Step 2: proportional share of remaining seats. If $\sum_k w_k \le 0$, the remaining seats are assigned in an RNG-shuffled category order, filling each category up to its cap. Otherwise, for each category:
$$\text{desired}_k = R \cdot \frac{w_k}{\sum_j w_j}.$$
The algorithm takes:
$$\text{base}_k = \min\Big(\lfloor \text{desired}_k \rfloor,\ M_k - x_k\Big), \qquad x_k \leftarrow x_k + \text{base}_k,$$
and stores the remainder
$$r_k = \text{desired}_k - \text{base}_k.$$

Step 3: largest-remainder cleanup (with RNG tie-break). Let $\widehat{R} = T - \sum_k x_k$ be the remaining unassigned seats (typically small). The algorithm orders categories by descending (r_k, u_k) where $u_k \sim \text{Uniform}(0, 1)$ is an RNG tie-break. It then walks this order once, giving one seat to each category that is still below its cap, until $\widehat{R} = 0$.

Step 4: fill any leftover capacity. If seats still remain (this can happen when caps prevented earlier assignments), the algorithm does one more RNG-shuffled pass over categories and fills any remaining capacity until the total is T.

5.2 Deterministic allocation (Hamilton-style apportionment)

This routine is used to define per-event demographic targets. It takes a total seat count T and nonnegative integer weights w_k over categories k and returns integers x_k with $\sum_k x_k = T$.

Step 0: trivial cases. If $T \le 0$, return all zeros.

Step 1: compute exact shares. If $\sum_k w_k \le 0$, it assigns one seat to each category in sorted key order until seats run out. Otherwise, for each category:
$$\text{exact}_k = T \cdot \frac{w_k}{\sum_j w_j}, \qquad x_k \leftarrow \lfloor \text{exact}_k \rfloor.$$

Step 2: assign leftover seats by largest remainder. Let $R = T - \sum_k x_k$. Order categories by descending $(\text{exact}_k - x_k,\ k)$ and give one additional seat to the first R categories in that order. This is fully deterministic: ties are broken by the category key (descending order).

6 Demographic modes (turn counts into weights)

Let the roster demographic counts be c_k for categories k (including `U` for unknowns).

6.1 Proportional mode

Weights equal counts:

$$w_k = c_k.$$

6.2 Women-to-parity mode (only when the demographic column is `Sex`)

If both `F` and `M` exist and $0 < c_{\mathtt{F}} < c_{\mathtt{M}}$, the planner *inflates* the female weight up to the male count:

$$w_{\mathtt{F}} \leftarrow c_{\mathtt{M}}, \qquad w_{\mathtt{M}} \leftarrow c_{\mathtt{M}},$$

and all other categories keep their original counts. If `Sex` is not the demographic column, this mode behaves the same as proportional mode.

7 Per-event demographic targets

Suppose each event selects s guests. Let $A(T)$ be the deterministic allocation of T seats across demographic categories using weights w_k.

Define the **cumulative target after event** e as:

$$C_e \;=\; A(e \cdot s).$$

Then the **event-e target** is the increment:

$$\text{target}_e \;=\; C_e - C_{e-1} \;=\; A(e \cdot s) - A((e-1) \cdot s).$$

This "difference of cumulatives" trick makes rounding behave well across timeas even when s is small and individual events must use integers the cumulative targets follow the weighted proportions as closely as possible.

Important detail. The planner computes these demographic weights and targets from the *full eligible guest roster* (excluding anyone in the host roster), not from the cohort subset. So the target is an aspiration for the series, while the cohort may limit what is feasible on a particular event.

8 Guest cohort construction (max unique guests)

The planner may restrict invitations to a fixed-size **guest cohort** so that only up to `guest_max_unique` distinct guests appear across the series.

Let $C = \min(\texttt{guest\_max\_unique},\ \#\text{eligible guests})$. The cohort is recomputed only when needed (changed roster, changed cohort seed, changed max-unique); otherwise it is reused.

8.1 Discipline seat allocation for the cohort

Let disciplines be ordered by the canonical discipline list. Let d_ℓ be the number of guests in discipline ℓ.

The cohort enforces minimum discipline coverage:

- If $C < \#\{\ell : d_\ell > 0\}$, it chooses C disciplines (favoring larger d_ℓ with RNG tie-breaks) and sets minimum 1 seat for those chosen disciplines, 0 for the rest.
- Otherwise it uses a base minimum of 1 seat per discipline, or 2 seats if $C \geq 2 \cdot \#\{\ell\}$ (clipped by availability).

It then uses the *randomized proportional allocator* with:

$$T = C, \quad w_\ell = d_\ell, \quad m_\ell = \text{minimum}_\ell, \quad M_\ell = d_\ell.$$

This returns an integer number of cohort seats per discipline.

8.2 Demographic seat allocation for the cohort

Let demographic counts be c_k over categories k in the roster (including `U`). Apply the configured demographic mode to get weights w_k. Then the cohort computes a demographic target using the randomized proportional allocator:

$$T = C, \quad w_k, \quad m_k = 0, \quad M_k = c_k.$$

8.3 Picking names to satisfy discipline and demographic targets

Within each discipline ℓ, the planner repeatedly selects the "best" remaining participant until it hits the allocated number of seats for that discipline. Across all picks, it tries to satisfy the demographic targets by tracking **remaining** demographic needs r_k.

The per-pick candidate ranking is:

1. candidates in a demographic category with $r_k > 0$ (needed) are preferred,
2. lower score$(i) = (a_i, n_i)$ is preferred,
3. smaller deterministic tie-break t_i is preferred,
4. then a stable name ordering is used (numeric names sort before alphabetic names).

After discipline quotas are satisfied, if the cohort still has fewer than C names, it fills remaining slots from all leftover participants using the same ranking.

9 Eligibility pools and cooldowns

Before selecting hosts or guests for an event, the planner filters out participants with cooldown $c_i > 0$.

9.1 Discipline-preserving cooldown override

Cooldown is overridden *only* to avoid completely removing a discipline from the pool:

- Let P be the input participant pool for a role (hosts or guests).
- Let $P_0 \subseteq P$ be those with cooldown ≤ 0.
- If every discipline present in P still appears in P_0, use P_0.
- Otherwise, re-add *all* participants from any "missing" disciplines, even if they are on cooldown.

This is why the README says cooldown is only overridden for discipline coverage.

10 Selecting participants for a single event

The host and guest selection routines are the same, except:

- hosts are selected *without* demographic targets, and
- guests are selected *with* the per-event demographic targets target_e defined earlier.

10.1 Demographic deficit for the current partial selection

Let selected_k be how many already-selected people are in demographic category k. Define the deficit

$$\Delta_k = \text{target}_e(k) - \text{selected}_k.$$

Only *positive* deficits matter; the algorithm does not prioritize between different positive magnitudes.

10.2 The candidate sort key (the <3 of "equity")

Each candidate i is ranked by the tuple:

$$\Big(\underbrace{\pi(i)}_{\text{demographic priority}}, \underbrace{(a_i, n_i)}_{\text{fairness/reliability}}, \underbrace{t_i}_{\text{seeded tie-break}}, \underbrace{\text{name\_order}(i)}_{\text{stability}} \Big)$$

and the planner always chooses the minimum lexicographically.

The demographic priority $\pi(i)$ is:

- If there is *no* category with $\Delta_k > 0$, then $\pi(i) = 0$ for everyone.
- Otherwise, $\pi(i) = 0$ if i is in any category with $\Delta_k > 0$, and $\pi(i) = 1$ otherwise.

So demographic targeting is a *first-pass filter*

10.3 Discipline anchoring

To avoid discipline drop-out within an event, the algorithm uses an **anchoring** phase:

- Compute a discipline order from the canonical discipline list.
- Let $A = \min(\text{seats},\ \#\text{disciplines present in the pool})$.
- For each of these A anchor slots, force the next pick to come from a discipline that has not yet been anchored.

Operationally, at each anchor step it:

1. finds the best candidate *within each not-yet-anchored discipline* using the sort key above, then
2. selects the best among those discipline-best candidates (again using the same sort key).

After anchoring, remaining seats are filled from the whole pool using the same sort key.

10.4 Waitlist ordering

After selecting the main set, the waitlist is the remaining pool sorted by:

$$(\text{score}(i),\ t_i,\ \text{name\_order}(i)),$$

i.e. *without* demographic-deficit priority. The planner takes the first `waitlist_size` people from this ordering.

11 Attendance feedback (how the state changes the math)

In the CLI/GUI workflow, state updates are appliedwhen you plan a later event. Before planning event e, the program records any earlier planned events that have not yet been recorded (in increasing event-index order) by reading the plan CSV.

When recording a single event:

- guest cooldowns are decremented first, then
- attendance outcomes for that event are applied.

Selected guests are required to have a non-empty recorded outcome (otherwise planning fails). Waitlist guest outcomes are optional; `filled` (and `attended`) count as attendance.

The per-role stat updates for outcomes are:

- `attended`: $a_i \leftarrow a_i + 1$.
- `no_show`: $a_i \leftarrow a_i + 1$, $n_i \leftarrow n_i + 1$, and $c_i \leftarrow \max(c_i, 1)$.
- `cant_attend`: no stat changes (acknowledged non-attendance without penalty).

Cooldown then decays by 1 *each time an event is recorded*:

$$c_i \leftarrow \max(0,\ c_i - 1).$$

Waitlist fill-ins. Waitlist guests with attendance marked `filled` (or `attended`) are treated as attended and get $a_i \leftarrow a_i + 1$. This means the waitlist can become a true selection mechanism over time, not just a backup list.