perplexity

Final C project plan (GIH, curl+gunzip, Premier+Trad)

1) Data inputs and downloads

Use 17Lands Public Datasets (not API scraping) and decompress with `gunzip`.[1] [2]

Download URLs:

- Powered Cube PremierDraft game data: `https://17lands-public.s3.amazonaws.com/analysis_data/game_data/game_data_public.Cube_-_Powered.PremierDraft.csv.gz`[2]
- Powered Cube TradDraft game data: `https://17lands-public.s3.amazonaws.com/analysis_data/game_data/game_data_public.Cube_-_Powered.TradDraft.csv.gz`[2]
- Card ID mapping: `https://17lands-public.s3.amazonaws.com/analysis_data/cards/cards.csv`[2]

Shell commands:

```
mkdir -p data/raw data/tmp data/out

curl -L -o data/raw/powered_premier_games.csv.gz \
  "https://17lands-public.s3.amazonaws.com/analysis_data/game_data/game_data_public.Cube_
curl -L -o data/raw/powered_trad_games.csv.gz \
  "https://17lands-public.s3.amazonaws.com/analysis_data/game_data/game_data_public.Cube_
curl -L -o data/raw/cards.csv \
  "https://17lands-public.s3.amazonaws.com/analysis_data/cards/cards.csv"

gunzip -c data/raw/powered_premier_games.csv.gz > data/tmp/powered_premier_games.csv
gunzip -c data/raw/powered_trad_games.csv.gz    > data/tmp/powered_trad_games.csv
```

17Lands notes gzip and `gunzip`/`tar` for decompression on Unix.[2]

2) Definitions (GIH and four buckets)

Presence definition: **GIH-style present** = "card was in the kept opening hand OR drawn later in the game," consistent with 17Lands' definition that #GIH counts opening-hand plus drawn-later instances.[3]

Synergy label for a pair (A, B):

$$\mathrm{syn}_{\Delta}(A, B) = p_{11} - p_{10} - p_{01} + p_{00}$$

where:

- $p_{11} = P(\text{win} \mid A \text{ present}, B \text{ present})$
- $p_{10} = P(\text{win} \mid A \text{ present}, B \text{ not present})$
- $p_{01} = P(\text{win} \mid A \text{ not present}, B \text{ present})$
- $p_{00} = P(\text{win} \mid A \text{ not present}, B \text{ not present})$

Compute from game-level rows (one row per game) in the public dump.[2]

3) Compute all buckets without “missing pair” iteration

Track only:

- Global counts: N, W (games, wins)
- Per-card present counts: N_A, W_A
- Pair both-present counts: N_{AB}, W_{AB}

Derive remaining buckets:

- $N_{10} = N_A - N_{AB}$, $W_{10} = W_A - W_{AB}$
- $N_{01} = N_B - N_{AB}$, $W_{01} = W_B - W_{AB}$
- $N_{00} = N - N_A - N_B + N_{AB}$, $W_{00} = W - W_A - W_B + W_{AB}$

This is valid because the four buckets partition games under a fixed present/not-present definition.[3] [2]

4) Repository structure

- `data/`
  - `raw/` gz + cards mapping
  - `tmp/` decompressed CSVs
  - `out/` generated labels/models
- `src/`
  - `csv.h/.c` streaming CSV reader
  - `hash.h/.c` open-addressing hash map (uint64 key → struct)
  - `cards.h/.c` loads `cards.csv` and provides `mtga_id ↔ name`[2]
  - `labels.h/.c` builds synergy labels from game CSV
  - `train.h/.c` trains model from labels
  - `infer.c` CLI for querying two card names
  - `main_labels.c`, `main_train.c`
- `Makefile`

5) Program: `labels` (build `labels.csv`)

Inputs:

- `data/tmp/powered_premier_games.csv`
- `data/tmp/powered_trad_games.csv`
- `data/raw/cards.csv`[2]

Outputs:

- `data/out/labels_premier.csv`
- `data/out/labels_trad.csv`
- Optional: `data/out/labels_both.csv` with a `format` column

Per row:

1. Read `won` (0/1).
2. Parse "opening hand cards" list and "drawn cards" list; union them to form GIH-present set. Game data includes opening-hand and drawn-later card lists.[2]
3. Update:
   - global N, W
   - for each present card A: N_A, W_A
   - for each unordered pair in the present set: N_{AB}, W_{AB}

Smoothing + thresholds:

- Use binary per-game presence (present at least once) to keep bucket algebra consistent with the partition logic.[3]
- Beta smoothing: $\hat{p} = \frac{w+\alpha}{n+\alpha+\beta}$ (start $\alpha = \beta = 1$).
- Emit pair labels only if $N_{AB} \geq$ `MIN_BOTH_PRESENT` (e.g., 500).

Label output schema:
`card_a,card_b,n11,w11,p11,n10,w10,p10,n01,w01,p01,n00,w00,p00,syn_delta`

6) Program: `train` (learn to predict synergy)

Baseline model (efficient in C, good generalization):

- Per-card embedding $v_i \in \mathbb{R}^d$, per-card bias b_i, global bias c
- Prediction: $\hat{s}(A, B) = v_A^\top v_B + b_A + b_B + c$

Training:

- Read `labels*.csv`
- Optimize weighted L2-regularized squared error with SGD:
  - Weight ω_{AB} based on sample size (e.g., N_{AB} or capped)

Outputs:

- `data/out/model_premier.bin`
- `data/out/model_trad.bin`
- Optional: `data/out/model_both.bin`

7) Program: `infer` (CLI)

Usage:

```
./infer data/out/model_both.bin "Tinker" "Blightsteel Colossus"
```

Steps:

- Name→ID via `cards.csv`[2]
- Compute $\hat{s}(A, B)$
- Optionally also print observed $\mathrm{syn}_{\Delta}(A, B)$ from labels if available.

8) Validation checklist

- Bucket consistency: derived counts non-negative; totals match partition identities.
- Spot-check: a small set of known cube combos should skew positive.
- Cross-format stability: train on Premier, evaluate on Trad (and vice versa).

9) Compliance / data hygiene

- Use 17Lands public dumps; they provide usage guidance that prefers public dataset use over scraping.[1]
- Cache raw downloads locally and record dataset "last updated" metadata from the public datasets page.[2]

⁂

1. https://www.statisticshowto.com/interaction-effect-interacting-variable/
2. https://statisticsbyjim.com/regression/interaction-effects/
3. https://www.statisticssolutions.com/statistical-interaction-more-than-the-sum-of-its-parts/