A Privacy-Preserving Tool for Assigning Census Tract Characteristics to Participant Data Using Non-PHI Feature Matching

Atharva Beesen

December 15, 2025

Abstract

Linking participant-level datasets to neighborhood characteristics is important for social science research. Many studies rely on census-tract information to capture local socioeconomic conditions, resources, and risks. However, conventional linkage methods depend on protected health information (PHI), such as residential addresses or geocoded coordinates, which creates privacy risks and barriers to Institutional Review Board (IRB) approval, data sharing, and reproducibility.

To address these issues, we developed a computational tool that lets researchers attach census-tract characteristics to participant records without using PHI. The tool compares non-identifying neighborhood features that already exist in both the participant file and an American Community Survey (ACS)-like tract file, and uses a relative-tolerance rule to find the tract whose characteristics best match each participant. It returns a matched file with tract-level attributes and hashed tract aliases, plus an unmatched file for participants who cannot be linked confidently. This paper describes the motivation, design, implementation, and simulation-based validation of the tool, and outlines how it can be used in future research. The package is openly available, easy to install, and usable through a single Python function call.

1 Introduction

Neighborhood context is closely tied to health. Countless studies have shown that community-level neighborhood features such as transit access, frequent social interaction, walkability, and green spaces are heavily linked to mental health, physical health, and well-being [2, 1]. Researchers often use census-tract measures such as income, poverty, education, and housing to capture these neighborhood conditions and study how they relate to individual outcomes.

In most projects, linking participants to census tracts requires address-level data. Analysts geocode home addresses or postal codes to obtain tract codes and then merge in ACS or census variables. Addresses and geocodes are treated as PHI under the Health Insurance Portability and Accountability Act (HIPAA): under the HIPAA Safe Harbor de-identification standard, "all geographic subdivisions smaller than a State, including street address and their equivalent geocodes" constitute identifiable information [5]. Large cohort studies report that geocoding itself can be technically demanding and time-consuming, especially when working with imperfect address data or very large samples [3]. These requirements can block secondary analysts from using neighborhood data, slow collaboration across institutions, and limit the ability to share or replicate analyses.

At the same time, many datasets already include tract-level characteristics that were attached upstream, for example by a health system or survey organization. Variables such as median household income or percent below poverty are not considered PHI. In practice, combinations of these variables can nearly identify a tract within a given region, even if the actual address or tract ID is never shared. This creates an opportunity: instead of starting from addresses, we can try to recover the most likely tract using only non-PHI features that are already present.

The tool described in this paper takes that approach. It accepts an ACS-like file (meaning any tract-level dataset with numeric socioeconomic indicators structured similarly to ACS summary tables) with tract-level features and a participant file with overlapping neighborhood variables. For each participant, it finds the tract whose characteristics best match the participant's values, using a simple relative-tolerance rule: that is, each tract is evaluated based on whether the relative difference between tract and participant values stays below a preset threshold across all shared features. When

a tract passes this tolerance test—a multidimensional check requiring all features to lie within the allowed preset percentage difference threshold—the tool transfers the chosen tract-level variable (the "new feature") to the participant and records a hashed tract alias instead of the raw census geographic identifier (GEOID). When no tract is a good match, the participant is placed in an unmatched file.

This design avoids direct use of PHI and does not require geocoding. It also aligns with broader work on protecting the confidentiality of census and administrative data, where new methods such as differential privacy are being introduced to balance privacy and data quality [4]. Here, instead of modifying the data with noise, we reduce risk by never handling addresses or true tract identifiers in the analyst-facing workflow.

In the sections that follow, we describe the tool's methods and software design, present a simulation study that tests its behavior under controlled conditions, and discuss how it might be used in real research settings, along with key limitations and possible extensions.

2 Methods

2.1 Public Availability and Installation

The tool is publicly accessible at:

`https://github.com/SustainableUrbanSystemsLab/NeighborhoodMatcher`

The package can be installed directly via:

```
pip install git+https://github.com/SustainableUrbanSystemsLab/
   ↪ NeighborhoodMatcher.git
```

Once installed, the primary function is imported with:

```
from acs_matcher import match_participants
```

This design allows any researcher with basic Python experience to use the tool without dealing with packaging details.

2.2 HIPAA Compliance Considerations

Because the tool never ingests, stores, or processes PHI such as addresses or geocodes, its workflow is compatible with HIPAA regulations. All matching is performed using non-identifying tract-level variables already present in the dataset, and GEOIDs are replaced with non-reversible hashed aliases. However, users must ensure they do not include address-like variables among the matching features; misconfiguration could reduce the intended privacy protections.

2.3 Required Inputs

The function operates on two comma-separated values (CSV) files:

- **ACS-like file** containing:
  - a `geoid` column,
  - one or more numeric tract-level features,
  - a `new_feature` column representing the attribute to be appended to participants.
- **Participant file** containing:
  - an `id` column,
  - any subset of the ACS-like feature columns.

Column order does not matter, but column *names* do: overlapping features are identified purely by matching column names after trimming whitespace.

Table 1 illustrates an example ACS input.

Table 1: Table 1: ACS input example.

geoid	feature_1	feature_2	feature_3	new_feature
100000	0.10	100	0.50	1.0
100001	0.20	200	0.60	2.0
100002	0.30	300	0.70	3.0
100003	0.40	400	0.80	4.0
100004	0.50	500	0.90	5.0

Table 2 shows an example participant input.

Table 2: Table 2: Participant input example.

id	feature_1	feature_2	feature_3
0	0.10	100	0.50
1	0.21	199	0.61
2	0.29	301	0.71
3	0.40	400	0.81
4	9.99	999	9.99

These inputs were constructed so that participants 0–3 match directly to the first four tracts, while participant 4 is too different to match any tract.

2.4 Overview of Algorithmic Design (with Code Examples)

The matching algorithm is designed to identify, for each participant, the census tract whose characteristics most closely resemble the participant's available neighborhood features. To make the logic transparent, each major step is explained briefly and paired with a minimal code snippet from the implementation.

2.4.1 Loading and Normalizing Input Data

Both CSVs are initially loaded as strings to avoid unintended coercion. Column names are stripped and standardized.

```
acs_df_raw = pd.read_csv(acs_csv_path, dtype=str)
```

```
user_df_raw = pd.read_csv(participant_csv_path, dtype=str)
acs_df_raw.columns = [c.strip() for c in acs_df_raw.columns]
user_df_raw.columns = [c.strip() for c in user_df_raw.columns]
```

This step keeps early parsing robust and avoids hidden type errors.

2.4.2 Identifying Overlapping Feature Columns

The tool uses only the feature columns that appear in both datasets. These shared columns form the basis of the matching.

```
user_feats = [c for c in user_df_raw.columns if c != "id"]
acs_feats = [c for c in acs_df_raw.columns
          if c not in ("geoid", "new_feature")]
overlap = [c for c in user_feats if c in acs_feats]
```

Participants can have a subset of the ”ACS” features (if there are other columns, these will be ignored); only the overlapping features are used for comparisons.

2.4.3 Converting Features to Numeric

Because ACS and participant data may include commas, symbols, or mixed formats, a flexible numeric parser is used.

```
def _to_numeric_series(s):
   if pd.api.types.is_numeric_dtype(s):
      return pd.to_numeric(s, errors="coerce")
   # fallback regex extraction
   x = str(s).replace(",", "")
   m = _num_re.search(x)
   return float(m.group(0)) if m else np.nan
```

This converts messy real-world values into usable numeric columns while keeping the original strings for output.

2.4.4 Computing Relative Differences

For each participant, the algorithm measures how far each tract is from that participant on every overlapping feature, using a relative difference. For a given feature, with tract value a and participant value u, we define:

$$\text{rel\_diff} = \frac{|a - u|}{\max\big(|a|, |u|, \varepsilon\big)}.$$

```
uv = row_cmp[overlap].to_numpy(dtype=float)[valid_cols]
av = acs_vals[:, valid_cols]


denom = np.maximum(np.maximum(np.abs(av), np.abs(uv)), eps)
rel = np.abs(av - uv) / denom
```

2.4.5 Applying the Tolerance Rule

A tract is considered a match only if *all* overlapping features are within the relative tolerance.

```
within = (rel <= rtol).all(axis=1)
```

This rule is conservative: if any feature is too different, the tract is not considered a candidate.

2.4.6 Selecting the Best Match

If multiple tracts pass the tolerance test, the algorithm chooses the one with the smallest average relative error.

```
cand_rel = rel[within]
best_idx = np.argmin(cand_rel.mean(axis=1))
```

```
best_global = np.flatnonzero(within)[best_idx]
```

This favors tracts whose overall feature profile is closest to the participant.

2.4.7 Attaching Tract Information with Hashed Identifiers

To protect PHI, the raw `geoid` is replaced with a hashed alias. Researchers never see tract identifiers, and the hashing is one-way.

```
def _alias_for_geoid(g):
   h = hashlib.sha1(g.encode("utf-8")).hexdigest()[:8]
   return f"alias_{h}"
```

The mapping from GEOID to alias is kept internal and does not appear in the output files.

2.4.8 Exporting Matched and Unmatched Participants

Finally, the tool writes two outputs next to the input participant file.

```
matched_df.to_csv(base + "_matched.csv", index=False)
unmatched_df.to_csv(base + "_unmatched.csv", index=False)
```

This separation makes it easy to see which participants were confidently matched and which were not.

2.5 Privacy Mechanism

As a further safeguard, the tool outputs only hashed versions of tract identifiers:

```
import hashlib


def alias_for_geoid(g):
   return "alias_" + hashlib.sha1(g.encode()).hexdigest()[:8]
```

This creates a non-reversible mapping, so analysts can group participants by tract alias without learning the underlying GEOID. If needed, a data steward could maintain a private crosswalk between GEOIDs and aliases, but this file is not produced or shared by default.

2.6 Expected Output Structure

The function `match_participants_to_new_feature` always produces two files:

1. a **matched output** file containing participants who were paired with a census tract within the specified tolerance, and
2. an **unmatched output** file containing participants for whom no tract met the tolerance criteria.

Both outputs closely mirror the participant input schema. The matched file appends two columns:

- `new_feature` — the tract-level variable transferred from the ACS file, and
- `tract_alias` — a Secure Hash Algorithm 1 (SHA-1)–based hashed identifier derived from the tract's GEOID.

This structure keeps the data easy to use and safe to share.

Example: Matched Output

Table 3: Table 3: Illustrative matched output structure.

id	feature_1	feature_2	feature_3	new_feature	tract_alias
0	0.10	100	0.50	1.0	alias_a2f4c9d1
1	0.21	199	0.61	2.0	alias_b771e8c3
2	0.29	301	0.71	3.0	alias_c118be29
3	0.40	400	0.81	4.0	alias_d9ef11aa

The matched output provides all original participant columns plus the neighborhood feature being transferred. The output has the first four participants in Table 2 – we mentioned earlier that we only expect these to have a matching tract.

Example: Unmatched Output

The unmatched file contains only participant columns, because no tract matched within the tolerance threshold.

Table 4: Table 4: Illustrative unmatched output structure.

id	feature_1	feature_2	feature_3
4	9.99	999	9.99

This unmatched row corresponds to the fifth participant in Table 2, whose features fall far outside the ACS feature ranges. In realistic applications, we do not expect 100% of participants to match; the match rate depends on how many tract characteristics are available and how tight the chosen tolerance is. Unmatched cases are treated as a safety valve rather than an error, and they signal records for which the available features do not support a confident tract assignment.

3 Results

To evaluate performance, we ran a controlled simulation that tested two main properties: (1) whether the algorithm correctly identifies census tracts when participant features closely match tract features, and (2) whether it leaves participants unmatched when their features do not resemble any tract within the chosen tolerance. This setup lets us see how the tool behaves when used only with non-PHI neighborhood characteristics.

All simulation code and synthetic datasets are available at:

`https://github.com/SustainableUrbanSystemsLab/NeighborhoodMatcher`

in the testing folder.

3.1 Simulation Inputs

We generated an ACS-like dataset with 100 synthetic census tracts and a participant dataset with 100 participants. For 95 participants, feature values were constructed so that each participant

corresponded closely (within 0.5% relative tolerance) to exactly one tract. For the remaining 5 participants, features were perturbed far beyond any realistic tolerance threshold so that no match should be found. Under this design, a correct implementation should return exactly 95 matched and 5 unmatched participants.

Tables 5 and 6 show the first ten rows of each dataset. These screenshots were generated directly in Python and saved as Portable Network Graphics (PNG) files.

Table 5: Table 5: ACS-like simulation input (top 10 rows).

geoid	feature_1	feature_2	feature_3	new_feature
100000	0.4371	21885.75	0.6494	0.5997575
100001	0.9556	58184.62	0.2589	1.1243712
100002	0.7588	38861.36	0.3131	0.8462886
100003	0.6388	50514.24	0.8290	1.0317924
100004	0.2404	74453.99	0.6245	1.0208649
100005	0.2404	34957.53	0.2064	0.5213753
100006	0.1523	44622.98	0.2710	0.5901298
100007	0.8796	65333.07	0.6645	1.2592557
100008	0.6410	33727.89	0.2035	0.7086539
100009	0.7373	24618.79	0.3126	0.6929879

Table 6: Table 6: Participant-level simulation input (top 10 rows).

id	feature_1	feature_2	feature_3
0	0.437060808016065	21887.63548942777	0.6493942406990518
1	0.955605992497185	58188.79082195383	0.2589062204659142
2	0.758806166786056	38860.80812226713	0.3130860596096783
3	0.638817558044215	50516.77451226550	0.8289482904882822
4	0.240410870471325	74457.78034652035	0.6244954659307771
5	0.240422878967981	34954.75523814771	0.2063939463799729
6	0.152300496508609	44626.57262206075	0.2710045341612624
7	0.879568854502720	65333.13863072336	0.6644438809322527
8	0.641037842870169	33730.09214430131	0.2035193078686720
9	0.737266206923771	24617.90396778348	0.3126303978957448

3.2 Running the Matching Tool

We first installed the package:

```
pip install git+https://github.com/SustainableUrbanSystemsLab/
   ↪ NeighborhoodMatcher.git
```

and then imported the tool and applied the matching function to the synthetic inputs:

```
from acs_matcher import match_participants


matched_path, unmatched_path = match_participants(
   acs_csv_path="acs_sim_for_report_acs.csv",
   participant_csv_path="participants_sim_for_report.csv",
   rtol=0.005
)
```

Because the simulation was fully deterministic, we expected exactly 95 matched participants and exactly 5 unmatched participants.

3.3 Simulation Outputs

Tables 7 and 8 show the first ten rows of the matched and unmatched output files produced by the tool.

The outputs illustrate two key behaviors:

- **Correct matches:** All 95 participants that should match were assigned tract aliases linked to the correct synthetic tracts.
- **Correct non-matches:** All 5 deliberately unmatchable participants were placed in the unmatched file, indicating that the algorithm did not force an arbitrary match.

Table 7: Table 7: Matched output (top 10 rows).

id	feature_1	feature_2	feature_3	new_feature	tract_alias
0	0.437060808016065	21887.63548942777	0.6493942406990518	0.5997575	alias_409e9519
1	0.955605992497185	58188.79082195383	0.2589062204659142	1.1243712	alias_66eafce8
2	0.758806166786056	38860.80812226713	0.3130860596096783	0.8462886	alias_1748b719
3	0.638817558044215	50516.77451226550	0.8289482904882822	1.0317924	alias_f21d1e28
4	0.240410870471325	74457.78034652035	0.6244954659307771	1.0208649	alias_0c1516d4
5	0.240422878967981	34954.75523814771	0.2063939463799729	0.5213753	alias_b4db53fd
6	0.152300496508609	44626.57262206075	0.2710045341612624	0.5901298	alias_44cd9fad
7	0.879568854502720	65333.13863072336	0.6644438809322527	1.2592557	alias_2c81c96f
8	0.641037842870169	33730.09214430131	0.2035193078686719	0.7086539	alias_84d97444
9	0.737266206923771	24617.90396778348	0.3126303978957448	0.6929879	alias_89fd7010

Table 8: Table 8: Unmatched output (top 10 rows).

id	feature_1	feature_2	feature_3
95	6.20072809389097	213882.7172649411	7.369808201314362
96	5.37931664054332	264087.4744803215	8.337788692605136
97	5.64439860955325	218188.0084399145	5.861599356008149
98	5.64022919478886	234566.7283323863	5.961445094043354
99	5.75951346756147	289678.8409906012	5.204343081332395

3.4 Simulation Summary

Table 9 summarizes the main performance metrics. As intended, the tool returned 95 matched participants and 5 unmatched participants. For the matched participants, the transferred `new_feature` values were recovered almost exactly, with numerical differences only at machine precision.

Table 9: Table 9: Simulation summary for 100 synthetic participants.

Metric	Value
Relative tolerance (rtol)	0.005
Total participants	100
Matched participants	95
Unmatched participants	5
Percent matched	95%
Percent unmatched	5%
Correct alias assignment	100%
Mean abs. relative error (new_feature)	$\approx 10^{-16}$
Median abs. relative error (new_feature)	$\approx 10^{-16}$

3.5 Interpretation

The simulation shows that the tool behaves as designed. When participant features align closely with those of a tract, the algorithm finds the correct match and transfers tract-level attributes with negligible numerical error. When participant features fall far outside the range of all tracts, the algorithm appropriately leaves those participants unmatched. This behavior is important in practice: it means that analysts can trust matches that pass the tolerance rule, while clearly seeing which records were not suitable for linkage.

4 Discussion

This tool offers a simple way to link participant-level data to census-tract characteristics without working directly with addresses or geocodes. It uses tract-level features that many datasets already contain, and relies on a transparent matching rule based on relative differences. The output keeps the familiar structure of the original participant file, adds tract-level variables of interest, and uses hashed tract aliases instead of true GEOIDs. In simulation, the tool successfully recovered the correct tract-level values for all matchable participants and correctly identified all intentionally unmatchable cases. This demonstrates that the method behaves reliably when appropriate tract-level features are available and the tolerance rule is well calibrated.

Methodologically, the tool sits between traditional geocoding and more recent privacy techniques. On one side, geocoding workflows have long been used to add neighborhood context to health data, but they depend on PHI and can be difficult to scale and manage securely [3]. On the other side, differential privacy and related methods add noise to protect confidentiality when publishing statistics from census or administrative data [4]. Our approach takes a different path: it avoids PHI altogether in the analyst-facing workflow and instead works only with tract-level summary variables that are already present in the dataset, never creating or exposing new row-level geographic identifiers.

The tool also has clear limitations. Its success depends on the quality and variety of tract-

level features available in both files. In areas where many tracts look very similar on the chosen variables, the algorithm may either leave more participants unmatched or assign matches that are statistically reasonable but not unique; both situations should be rare if the tolerance level is kept at a low number (it is automatically set to 0.5%) and the dataset does not contain essentially identical tracts. Additionally, the current implementation handles only numeric features and uses a fixed tolerance parameter. It treats each participant independently and does not calculate explicit measures of uncertainty about tract assignment. Computationally, runtime grows with the number of participants and tracts; this is acceptable for typical study sizes but could become slow for very large, nationwide datasets without additional optimization.

There are also practical risks if the tool is used incorrectly. If a user accidentally includes address-like variables among the matching features, or if tract aliases are later linked back to GEOIDs and merged with other sensitive information, some privacy benefits would be reduced. These risks can be limited through clear documentation, careful selection of features, and organizational policies that keep any GEOID-to-alias crosswalk separate and restricted.

Future work could extend the tool in several ways. One direction is to support categorical and count variables and allow different weights for each feature. Another is to return multiple candidate tracts with similarity scores, giving analysts a sense of uncertainty in ambiguous cases. Finally, applications to real-world cohorts could compare non-PHI matching against traditional address-based linkage, measuring how often the assignments agree and how any discrepancies affect downstream estimates of neighborhood effects on health.

5 Conclusion

We developed and tested an open-source Python tool that links participant-level data to census-tract characteristics using only non-PHI neighborhood features. By matching on overlapping tract-level variables rather than addresses or geocodes, the tool offers a practical option for researchers who cannot access PHI but still want to study the role of neighborhood context.

The tool is easy to install, requires only two CSV inputs, and returns matched and unmatched outputs that are straightforward to interpret and safe to share. A simple simulation study shows that the algorithm can achieve high matching accuracy while correctly flagging participants whose features do not align with any tract. In settings where direct geocoding is not feasible, this approach can help extend neighborhood-effects research and support more nuanced analyses of how place shapes health, without expanding the footprint of sensitive identifying information.

6 References

References

[1] H. Cohen-Cline, E. Turkheimer, and G. E. Duncan. Access to green space, physical activity and mental health: a twin study. *Journal of Epidemiology and Community Health*, 69(6):523–529, 2015.

[2] L. Ma, J. L. Kent, and C. Mulley. Transport disadvantage, social exclusion, and subjective well-being: The role of the neighborhood environment—evidence from sydney, australia. *Journal of Transport and Land Use*, 11(1):31–47, 2018.

[3] J. A. McElroy, P. L. Remington, A. Trentham-Dietz, S. A. Robert, and P. A. Newcomb. Geocoding addresses from a large population-based study: lessons learned. *Epidemiology*, 14(4):399–407, 2003.

[4] S. Ruggles, C. Fitch, D. Magnuson, and J. Schroeder. Differential privacy and census data: Implications for social and economic research. *AEA Papers and Proceedings*, 109:403–408, 2019.

[5] U.S. Department of Health and Human Services. 45 c.f.r. §164.514 — other requirements relating to uses and disclosures of protected health information. Electronic Code of Federal Regulations, 2023.