UNIVERSITÄT LEIPZIG

Medizinische Fakultät

Universitätsklinikum Leipzig

Medizin ist unsere Berufung.

GMDS 2024

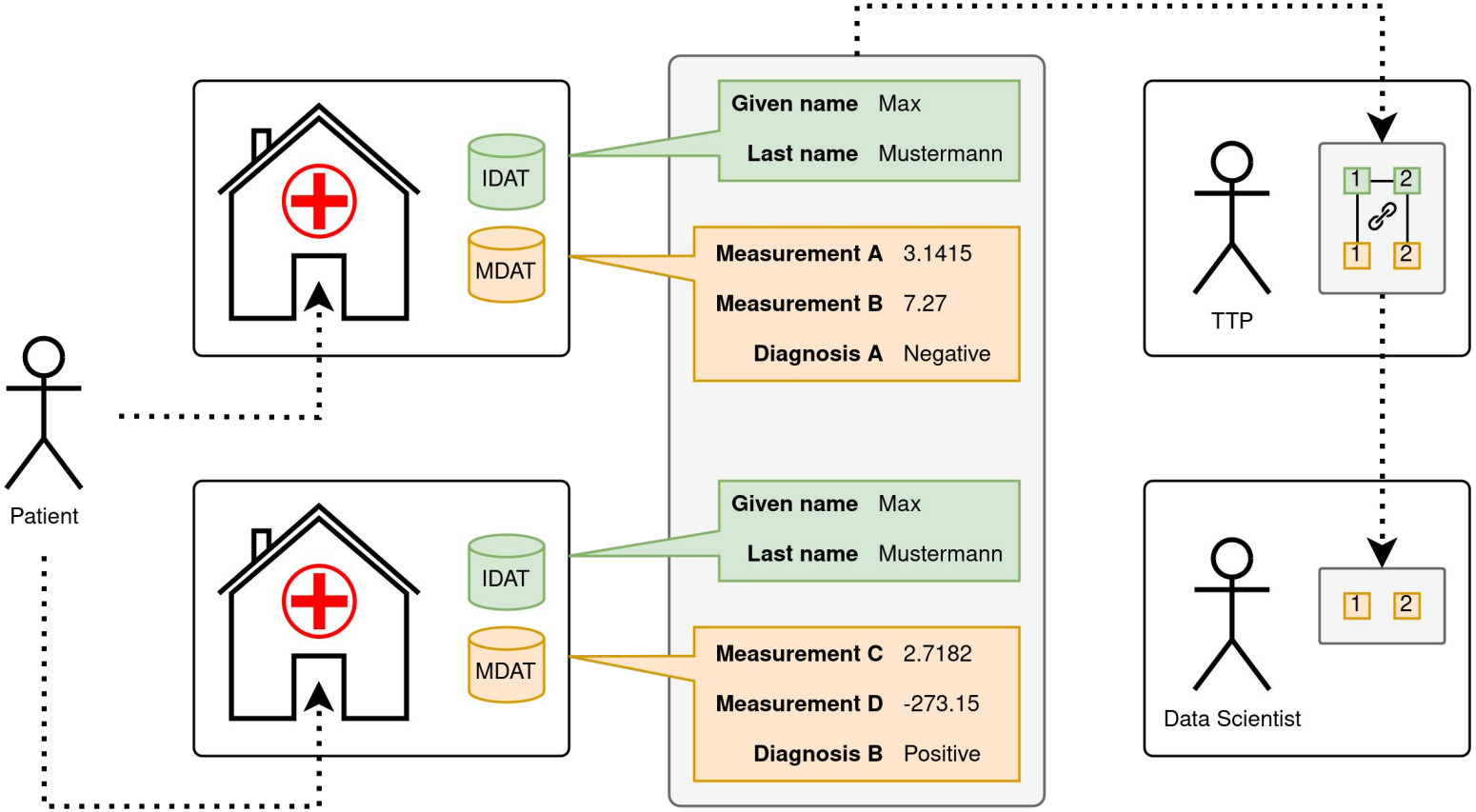
Generation and mutation of realistic personal identification data for the evaluation of record linkage algorithms

Dresden, 09.09.2024

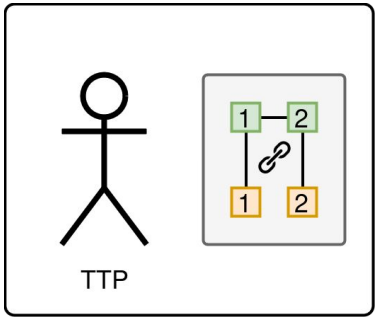
Maximilian Jugl, Toralf Kirsten

MDS

Medical Data Science


IDAT
MDAT
IDAT
MDAT
Patient
Given name Max
Last name Mustermann
Measurement A 3.1415
Measurement B 7.27
Diagnosis A Negative
Given name Max
Last name Mustermann
Measurement C 2.7182
Measurement D -273.15
Diagnosis B Positive
TTP
1 2
1 2
1 2
Data Scientist

A PRIMER ON RECORD LINKAGE



- Testing of various record linkage strategies
- Testing against error sources and varying data schemas
- Limited access to real-world data

⇒ **Generation of realistic-looking test data**

Given name	Last name	Gender	Date of birth
Axel	Schweiss	Male	1981-01-10
Grube	Claire		1970-10-01
Anna	Kond4	Female	1991-08-02

Typos?

Flipped values?

OCR errors?

Ambiguous format?

Missing values?

WHY ANOTHER TOOL?

- TDGen (Bachteler and Reiher, 2012)
  - No longer works with KNIME
- GeCo (Tran et al., 2013)
  - Python 2.7 only (deprecated)
  - Arbitrary software limitations
- GouDa (Restat et al., 2022)
  - No realistic distributions
- DaPo^{+} (Hildebrandt et al., 2023)
  - No source code or binaries available
  - Strict dependency on Apache Spark

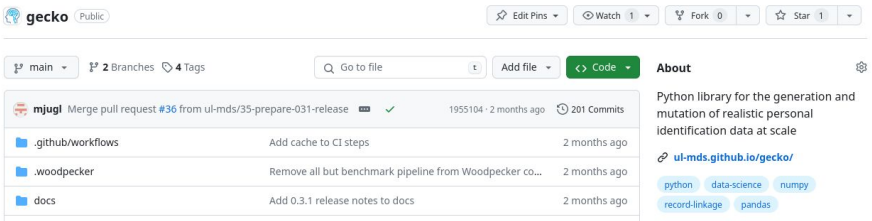
Software desiderata

- Data generation using shareable Python scripts
- Domain and schema independence
- Use of standardized file formats
- Data generation and mutation for multiple data columns
- Distribution as standard Python package
- Horizontal scalability
- Open source

PRESENTING GECKO

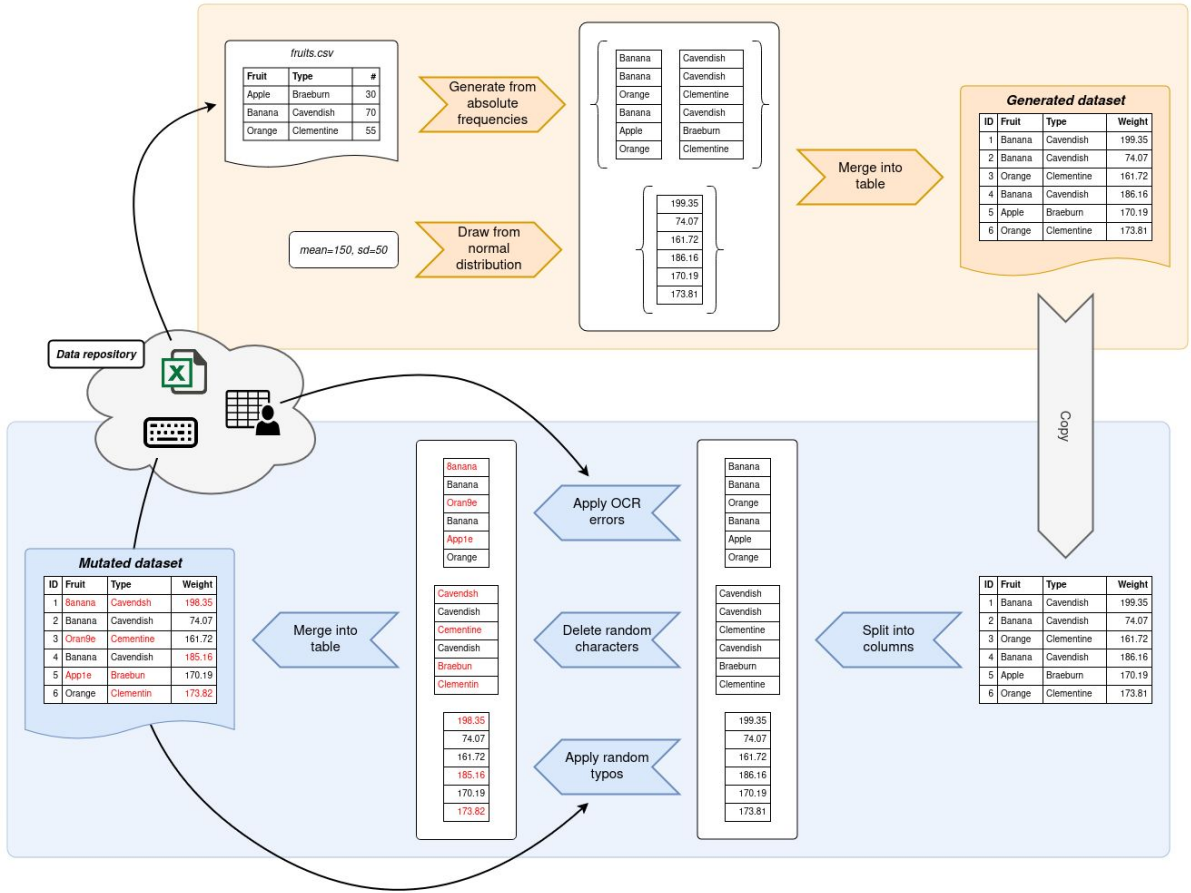
- Modern approach to the ideas put forward by GeCo (Tran et al. 2013)
  - Completely reworked from the ground up for modern Python
  - Based on NumPy and Pandas to integrate into data science applications
  - Domain-independent, highly configurable and scalable

- Source code: https://github.com/ul-mds/gecko
- Documentation: https://ul-mds.github.io/gecko/
- Python Package Index: https://pypi.org/project/gecko-syndata/



Source Code

Publication


fruits.csv
Fruit | Type | #
Apple | Braeburn | 30
Banana | Cavendish | 70
Orange | Clementine | 55
Generate from absolute frequencies
Banana | Cavendish
Banana | Cavendish
Orange | Clementine
Banana | Cavendish
Apple | Braeburn
Orange | Clementine
mean=150, sd=50
Draw from normal distribution
199.35
74.07
161.72
186.16
170.19
173.81
Merge into table
Generated dataset
ID | Fruit | Type | Weight
1 | Banana | Cavendish | 199.35
2 | Banana | Cavendish | 74.07
3 | Orange | Clementine | 161.72
4 | Banana | Cavendish | 186.16
5 | Apple | Braeburn | 170.19
6 | Orange | Clementine | 173.81
Copy
Data repository
Mutated dataset
ID | Fruit | Type | Weight
1 | 8anana | Cavendsh | 198.35
2 | Banana | Cavendish | 74.07
3 | Oran9e | Cementine | 161.72
4 | Banana | Cavendish | 185.16
5 | App1e | Braebun | 170.19
6 | Orange | Clementin | 173.82
Merge into table
8anana
Banana
Oran9e
Banana
App1e
Orange
Cavendsh
Cavendish
Cementine
Cavendish
Braebun
Clementin
198.35
74.07
161.72
185.16
170.19
173.82
Apply OCR errors
Delete random characters
Apply random typos
Banana
Banana
Orange
Banana
Apple
Orange
Cavendish
Cavendish
Clementine
Cavendish
Braeburn
Clementine
199.35
74.07
161.72
186.16
170.19
173.81
Split into columns
ID | Fruit | Type | Weight
1 | Banana | Cavendish | 199.35
2 | Banana | Cavendish | 74.07
3 | Orange | Clementine | 161.72
4 | Banana | Cavendish | 186.16
5 | Apple | Braeburn | 170.19
6 | Orange | Clementine | 173.81

```
from pathlib import Path
import numpy as np
from gecko import generator, mutator

rng = np.random.default_rng(727)
gecko_data_dir = Path(__file__).parent / "gecko-data"
```

Script setup

Imports and RNG instances

```
df_generated = generator.to_data_frame(
    {
        ("given_name", "gender"): generator.from_multicolumn_frequency_table(
            gecko_data_dir / "de_DE" / "given-name-gender.csv",
            value_columns=["given_name", "gender"],
            freq_column="count",
            rng=rng,
        ),
    },
    10_000,
)
```

Data generation

Assignment of generators to single or multiple columns

```
df_mutated = mutator.mutate_data_frame(
    df_generated,
    {
        "gender": (.01, mutator.with_categorical_values(
            gecko_data_dir / "de_DE" / "given-name-gender.csv",
            value_column="gender",
            rng=rng,
        )),
    },
    rng,
)
```

Data mutation

Assignment of generators to single or multiple columns

```
df_generated.to_csv("german-generated.csv", index_label="id")
df_mutated.to_csv("german-mutated.csv", index_label="id")
```

Data export

OUTPUT AND PERFORMANCE

ID	Given name	Last name	Gender	Street name	Municipality	Postcode
254	Helmut	Jahn	m	Peenestraße	Stolpe	17 391
M-254	Jahn	Helmut	m	Peenestraße	Stolpe	17 391
1226	Rudolf	Franzen	m	Birkenweg	Suthfeld	31555
M-1226	Rudolf	Franzen	m	Birkenweg	Suthfeld	31565
2397	Erna	Eickhoff	f	Schulweg	Krautheim	74 238
M-2397	Erna	Eickhoff	*(empty)*	Schulweg	Krautheim	74 238
9960	Ingrid	Reinhold	f	Hochstraße	Mogendorf	56 424
M-9960	Ingrid	Reinhold	m	Hochstraße	Mogendorf	56 424

- Frequency tables sourced from publicly available sources
- Arbitrary configuration of mutators across single and multiple columns

⇒ https://github.com/ul-mds/gecko-examples

OUTPUT AND PERFORMANCE

Dataset	Records	CPU time in s				
		Min	Q5	Q50	Q95	Max
American	100 000	0.30	0.30	0.31	0.32	0.33
	1 000 000	2.82	2.83	2.87	3.07	3.11
	10 000 000	28.00	28.05	28.28	30.18	30.62
German	100 000	0.80	0.80	0.81	0.85	0.87
	1 000 000	6.63	6.63	6.74	6.84	6.86
	10 000 000	65.12	65.26	66.09	66.86	67.09

- Benchmark with generation and mutation of 100k to 10m records
- Evaluation of single-core performance

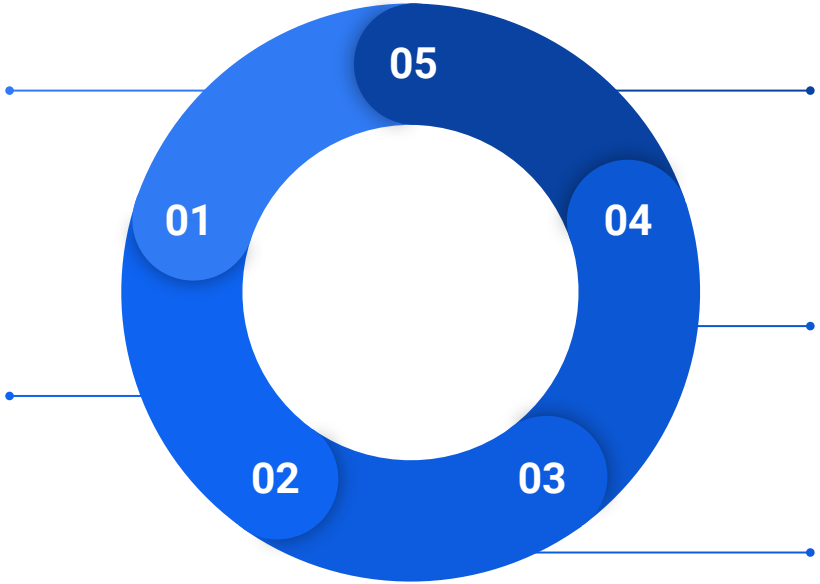
⇒ Gecko is 15~100x faster than its modern alternatives

USE CASE: THRESHOLD ESTIMATION FOR PPRL

Analyse error sources in patient cohort

≈14k records and ≈7.5k unique persons

Derive error configuration for Gecko



Evaluate match quality against cohort

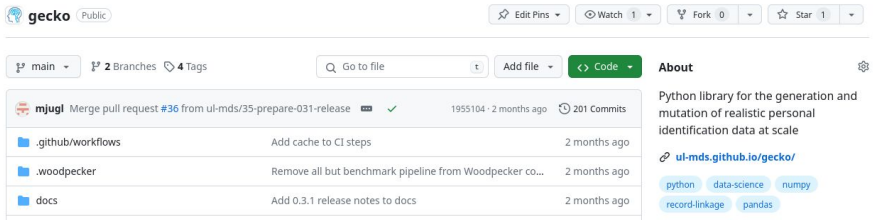
Derive ideal threshold and apply to cohort

Generate datasets using Gecko and perform matching

WHERE DO WE GO FROM HERE?

- Continuous testing of old and new record linkage algorithms
- Stress-testing of input forms that validate user-generated data
- PoC training data for machine learning models
- *This line could summarize your use case!*

⇒ **Reach out!** Maximilian.Jugl@medizin.uni-leipzig.de



UNIVERSITÄT LEIPZIG
Medizinische Fakultät

Universitätsklinikum Leipzig
Medizin ist unsere Berufung.

Thank you!

Dresden, 09.09.2024

Maximilian Jugl, Toralf Kirsten

MDS
Medical Data Science