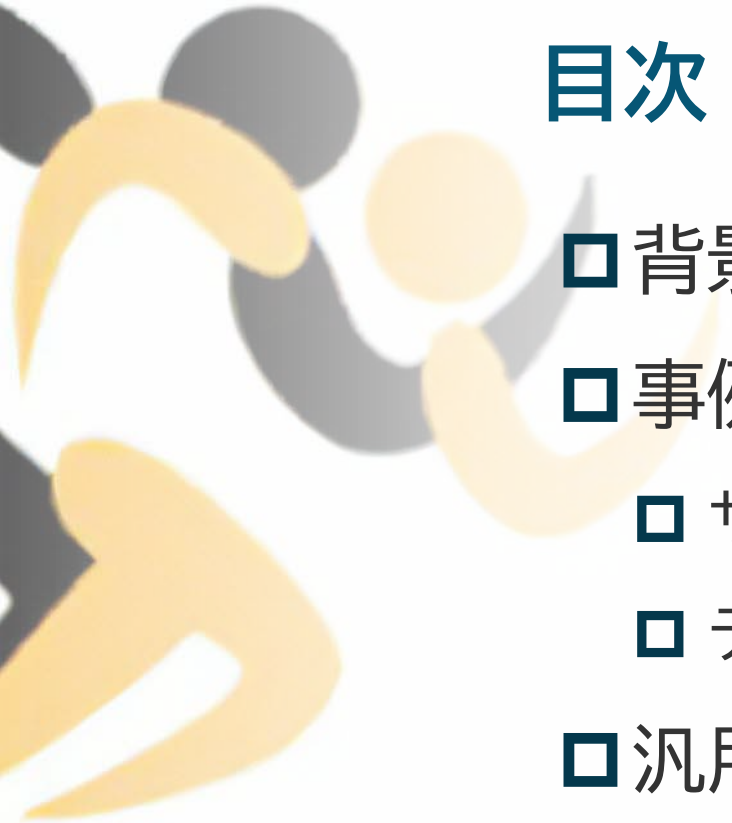
SPARK VISION
DATA SCIENCE

匿名データの利用における協働と省力化の試み

- IPUMSとのハーモナイゼーションの例より

株式会社 Spark Vision　中松 建

2024/11/18　公的統計ミクロデータ利活用に関する研究集会



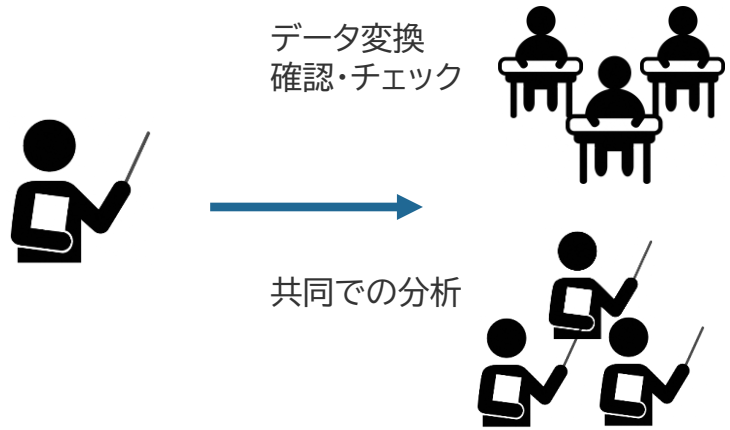
目次

- 背景
- 事例の紹介
  - サンプル(ダミー)データの作成
  - データ変換
- 汎用化とメタデータの検討
- まとめ
- 公開データ・プログラム

背景:匿名データ利用時の悩み

こんなお悩みありませんか？

一般的なデータ



匿名データ

- 作業・負担が集中
- 利用期間までに予定の作業が終わらない

大量の作業が必要なハーモナイゼーション(項目の調和)を
分担・協力して行った事例を紹介

背景：IPUMSとデータの変換

IPUMS(International Integrated Public Use Microdata Series)
ミネソタ人口センターと世界各国の統計局による共同プロジェクト
世界中の国勢調査や調査データを統合し、研究者が利用しやすい形式で提供

提携・協力

2017年～　一橋大学　公的統計ミクロデータ利用の研究基盤構築事業

2021年～　立正大学　匿名データとのハーモナイズ(項目の調和)などについて協働

国勢調査の匿名データをIPUMSのフォーマットへ変換

今回の実施体制

事例の紹介

- サンプル(ダミー)データの作成
- データ変換

サンプル(ダミー)データの作成

1. データのレイアウトが同じ	エンコーディング・列数・型
2. 定義から取りうる値が入力されている	年齢の値は 1 ～19, 99 労働力状態が主に家事の場合は、産業・職業は対象外　など
3. 対象が一般的でイメージしやすい - 項目間で整合性が取れている - ある程度多様な範囲をカバー	あまりよくない例: 年齢が勤続年数がほぼ同じ 都内在住で産業が農業 など

複雑なデータほど、高いレベルのものが欲しい　- データがイメージしやすくなる

数年前

作成を検討したものの、3のレベルは手間がかかるので断念…

例:ペルソナ(人物像)の設定 → 平均値などから妥当な回答の検討

＋ 調査票・項目の仕様の確認

2023 初夏

生成AIのテストとして作成

2024 春～

プログラムの動作確認等で利用

サンプル(ダミー)データ概要

以下の世帯員の構成を仮定して、サンプルのデータを**符号表・調査票**の情報だけで生成

- 夫婦+子供1人
- 夫婦のみの若い世帯
- 夫婦のみの高年齢の世帯
- 3世代(親世代+夫婦+子供)の世帯
- 働いている単身者の世帯
- アルバイトをしている学生の単身者の世帯

プロンプトによる指示と出力

世帯情報についての質問 → 各世帯員についての質問

コンテキスト情報
→ 項目間の整合性

手作業

コード化 → データ確認・微調整

＊GPT-4(0613相当)を使用、当時はAPIが利用不可

データ変換：マッピング情報

status	ipums varname	variable label	Var number	2000 CENSUS
GEOGRAPHY/HOUSEHOLD				GEOGRAPHY/HOUSEHOLD
lone	country	Country		
lone	year	Year		
lone	sample	Sample		
kip			1	Government statistics code
kip			2	Management code
kip			3	Survey date (Christian calendar year)
lone	pref_jp	Prefecture, Japan	4	Prefectures
lone	region_jp	Region, Japan		"
lone	munic_jp	Municipality, Japan	5	Municipality
kip			6	Record sequence number
lone	serial	Household serial number	7	Household serial number
lone	pernum	Person number	8	Household member number
lone	gq	Group quarters (collective)	9	Household type
lone	bldgtype	Building type	10	Residential building
lone	stories	Stories in structure	11	Floor number of building as a whole
lone	storyhh	Household story in structure	12	The floor where the household lives
lone	ownership	Ownership of dwelling	13	Relationship housing type to housing ownership
lone	livearea	Living area in square meters	14	Floor area of housing
lone	inctype	Type of household income	15	Types of household income
lone	persons	Number of persons in household	16	Household members
lone	hhtype	Household classification	17	Household family type
lone	multigen	Multigenerational household	18	3 generations household

変数間の対応情報

大量のファイルと情報

30項目ほど × **4**時点

どう扱う…？

CLASSWKD
Class of worker (detailed)

IPUMS code	IPUMS label	sample => source variable =>	japan2000 employee status	japan2005 employee status	japan2010 employee status
0	NIU (not in universe)		[blank] = Not covered	[blank] = Not covered	[blank] = Out of target
100	SELF-EMPLOYED		4 = Self-employed	4 = Self-employed	5 = Self-employed
101	Self-employed, unincorporated				
102	Self-employed, incorporated				
110	Employer				
111	Sharecropper, employer				
120	Working on own account				
121	Own account, agriculture				
122	Domestic worker, self-employed				
123	Subsistence worker, own consumption				
124	Own account, other				
125	Own account, without temporary/unpaid help				
126	Own account, with temporary/unpaid help				
130	Member of cooperative				
140	Sharecropper				
141	Sharecropper, self-employed				
142	Sharecropper, employee				
150	Kibbutz member				
199	Self-employed, not specified				
200	WAGE/SALARY WORKER				
201	Management		3 = Officer	3 = Officer	4 = Director of firm
202	Non-management				
203	White collar (non-manual)				
204	Blue collar (manual)				
205	White or blue collar				
206	Day laborer				
207	Employee, with a permanent job		1 = Regular employment	1 = Regular employment	1 = Regular employment
208	Employee, occasional, temporary, contract		2 = Temporary employmen	2 = Temporary employmen	2 = Temporary employmen
208	"				3 = Part-time job, other
209	Employee without legal contract				

符号表間の対応情報 - 30ファイル程度

データ変換：アプローチの検討

Excelのマッピング情報からプログラムを生成 → **省力化・ミスの予防**

- 単一の変数の変換を生成の対象とする（年齢の5歳区分から10歳区分への変換など）
- 複数変数を参照する処理は少なく（1割未満ほど）、個別に対応

プログラム関連

- 動作環境を選ばない（一般的でないライブラリやバージョンに依存しない）
- 容易に処理の追加・修正が可能

~~キーに対応する変数のマージ~~

~~条件分岐による処理（case whenやif thenなど）~~

→ 辞書・連想配列の使用

- PythonではMap関数により対応可能
- Rでもlistによる処理を無名関数かunnameで行うことで1行で処理が可能

```
{
"1": " 2", # 0 to 4: 2 years
"2": " 7", # 5 to 9: 7
"3": "12", # 10 to 14: 12
}
```

データ変換：プログラムの生成

1. 符号表間の対応情報を辞書として読み込み
2. 列を順番に処理　- 列のソートを不要に
3. 必要に応じて辞書を変換に利用

→ 列・辞書の指定や辞書の値の変更だけで、追加・修正可能

```python
import pandas as pd
import json

df = pd.read_csv('jp_data/2015kokucho.csv', header=None, dtype=str,encoding="cp932")
df.columns = ['var' + str(i+1) for i in range(len(df.columns))]

with open('code/dict.json', 'r') as f:
    codebook = json.load(f)

out = pd.DataFrame()

out["country"] = "" # 位置の確定のため 個別処理は下部にまとめて記載
out["year"] = ""
out["sample"] = ""

out["pref_jp"] =df["var4"]
out["region_jp"] = df["var4"].map(codebook["region_jp"]["japan2015"])
```

```r
# R言語の場合　　sapplyでなくunnameによる方法もあり
out$region_jp <- sapply(df$var4, function(x) codebook$region_jp$japan2015[x])
```

```
"region_jp": {
    "japan2000": {
        "01": "1",
        "02": "2",
        "03": "2",
        "04": "2",
    }
},
"trnwrk": {
    "japan2000": {
        "  ": "0",
        "01": "20",
        "04": "31",
```

汎用化とメタデータの検討

- ファイル形式
- メタデータ マッピング情報例

汎用化とメタデータの検討

マッピング情報(Excel)がプログラムでは利用しづらい・・・

空白(null)、0・空白埋めの対応
セル内の変換に不要な情報(ラベルなど)の除外　など

CLASSWKD
Class of worker (detailed)

IPUMS code	IPUMS label	sample => source variable =>	japan2000 employee status	japan2005 employee status	japan2010 employee status
0	NIU (not in universe)		[blank] = Not covered	[blank] = Not covered	[blank] = Out of target
100	SELF-EMPLOYED		4 = Self-employed	4 = Self-employed	5 = Self-employed
101	Self-employed, unincorporated				
102	Self-employed, incorporated				
110	Employer				
111	Sharecropper, employer				
120	Working on own account				
121	Own account, agriculture				
122	Domestic worker, self-employed				
123	Subsistence worker, own consumption				
124	Own account, other				
125	Own account, without temporary/unpaid help				
126	Own account, with temporary/unpaid help				
130	Member of cooperative				
140	Sharecropper				
141	Sharecropper, self-employed				
142	Sharecropper, employee				
150	Kibbutz member				
199	Self-employed, not specified				
200	WAGE/SALARY WORKER				
201	Management		3 = Officer	3 = Officer	4 = Director of firm
202	Non-management				
203	White collar (non-manual)				
204	Blue collar (manual)				
205	White or blue collar				
206	Day laborer				
207	Employee, with a permanent job		1 = Regular employment	1 = Regular employment	1 = Regular employment
208	Employee, occasional, temporary, contract		2 = Temporary employmen	2 = Temporary employmen	2 = Temporary employmen
208	"				3 = Part-time job, other
209	Employee without legal contract				

CSV
XLSX

???

- 必要な情報(nullを含む)を保持できる
- 機械的に処理しやすい
- 人からでも扱いやすく、直接読み書きも可能
- 高いExcelとの親和性(インポートやエクスポートがしやすい)

ようなファイル形式や記法を検討

ファイル形式の検討

json

○ プログラムからは扱いやすい

✗ コメントが書けない(派生の形式で可能なものもある)

✗ 慣れていない人では直接扱いづらい

```
{
  "title": "Sample Metadata",
  "author": "John Doe",
  "date": "2023-10-01",
  "tags": ["sample", "metadata"],
  "version": 1.0
}
```

表形式(CSV, XLSXなど)

○ファイルの扱いに慣れている

✗ 制約がゆるく、形式が崩れがち

✗ 階層的な情報を扱いづらい

Item	Value
Title:	Sample Metadata
Author	John Doe
Date	2023-10-01
Tag	sample
Tag	metadata
Version	1.0

YAML

○ jsonに変換可能

○ 記号が少なく、書きやすい

△ 階層やリストに複数の書き方がある

```
title: Sample Metadata
author: John Doe
date: 2023-10-01
tags:
  - sample
  - metadata
version: 1.0
```

その他: ini, xml, toml等

変数間のマッピング情報例

必要な情報

- 変換後データ用の情報(変数名、ラベル)
- 処理の種類
- 符号表のマッピング情報
- 変換前変数の情報(列番号・変数名)

処理の種類

- copy: 値のコピー
- assign: 定数の代入
- dict: 辞書による変換
- skip: 対応なし(スキップ・作成されない)
- null: 個別処理が必要

```yaml
from: Population Census of Japan
to: IPUMS
entries: ["2000", "2005", "2010", "2015"]  # 繰り返し
notes:

map:
  age:
    label: Age
    type: dict
    vars: [21: age, 20: age, 20: age, 19: age]
    # 各繰り返しの列番号、または番号+列名

  inctype:
    label: Type of household income
    type: dict
    dict: inctype2
    vars: [15: income, null, null, null]

  year:
    label: year
    type: 'assign'
    values: ['2000', '2005', '2010', '2015']
```

符号表間のマッピング情報例

- 複数の繰り返し(時点や国など)を扱うことを想定
- 符号だけだとわかりづらいため、同行にコメント記載可能に
- 符号部分は比較的容易にexcelと変換が可能
  「:#」を区切り文字として扱う
- 括弧など極力余分な情報を持たせない

```yaml
name: age
from: Population Census of Japan
to: IPUMS
notes: "AGE is a continuous variable. For samples that
report age in 5-year groups, such as Japan, IPUMS codes AGE
to the mid-point of the interval.
In the Japanese censuses, AGE is top-coded at 85+ in 2000
and 2005 and 90+ in 2010 and 2015."

entries:
  "2000":
    "1": " 2" # 0 to 4: 2 years
    "2": " 7" # 5 to 9: 7
    "3": "12" # 10 to 14: 12

  "2005":
    "1": " 2" # 0 to 4: 2 years
    "2": " 7" # 5 to 9: 7
    "3": "12" # 10 to 14: 12
```

まとめ

- サンプル(ダミー)データ
- メタデータ
- 公開データ・プログラム

まとめ：サンプル(ダミー)データ

様々な面で使い勝手を高めてくれた

- データ仕様の理解促進(初めて利用する場合)
- 普段利用している環境での作業
- ツールの動作の再現・保証

直接データを扱わずに協働するには、**必須に近い**

フロー制御による作成にもトライ

(賃金構造基本統計調査)

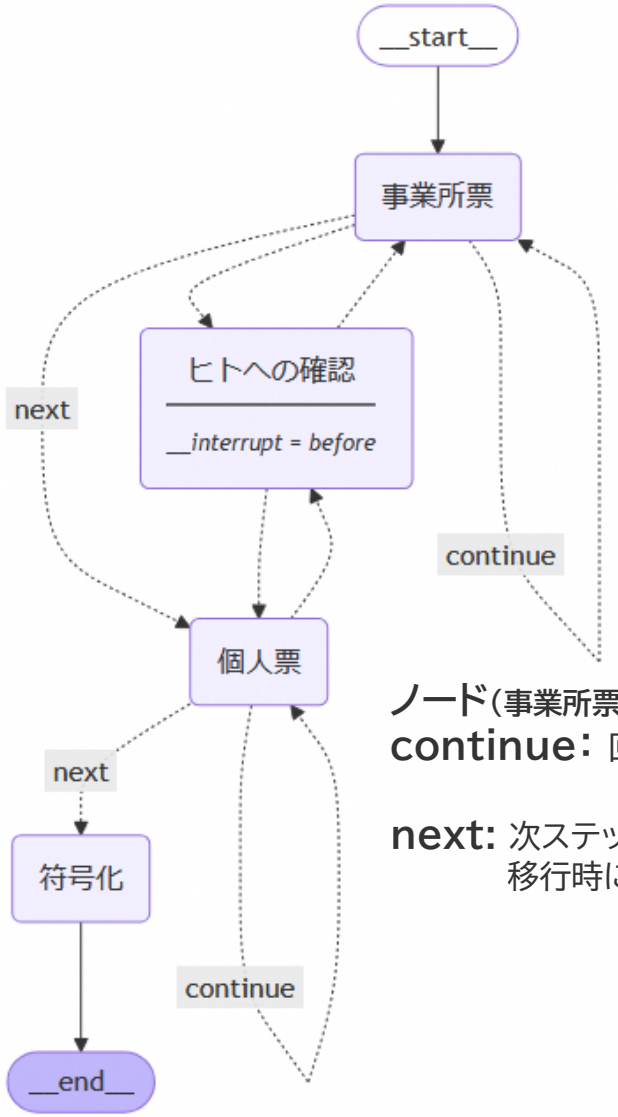
比較的単純なフロー

質問・項目はある程度まとめて扱う
Human In The Loop(ヒトによる確認・介入)を多用

↓

調査をエミュレートする形なども検討

Tool calling による厳密な回答の制限が可能



ノード(事業所票・個人票)： 回答・次の動作の判断
continue： 回答後のループ

next: 次ステップへの移行
移行時には履歴を整理

まとめ:メタデータ

協働の際には情報の整理・共有が必要

仕様の情報として独自の記法で書かれていることも多い

~~情報の整理・共有のためのメタデータ作成・・・作成するのが面倒になりがち~~

省力化にも活用しないともったいない

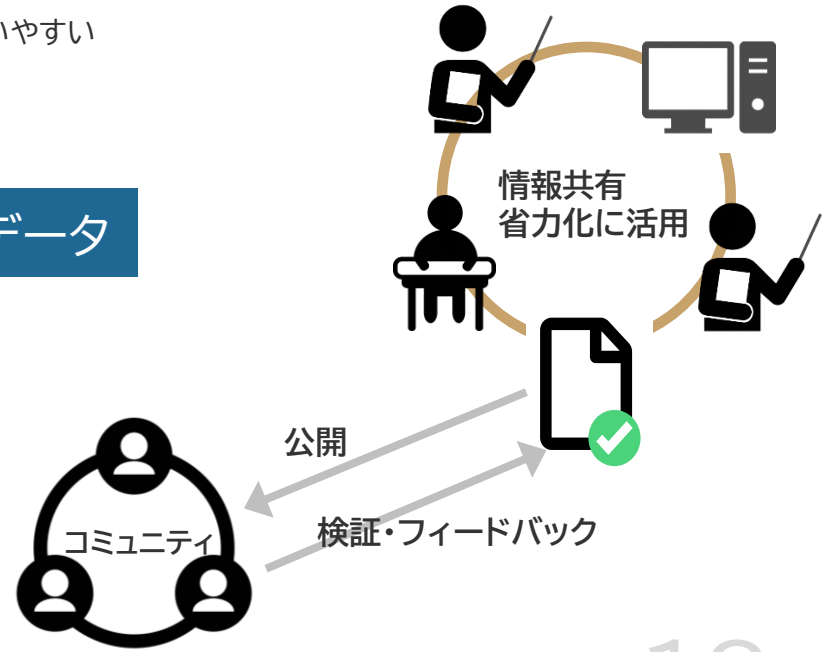
- プログラムの生成
- 生成AIを利用したマッピング(変数や符号の対応)

適度に分割・細分化されて、生成AIからでも扱いやすい
ドラフトとしては十分なマッピングが可能

協働の過程における情報の整理 ＋ 活用を通して高品質になったメタデータ

→ 限定公開のデータにおいても、**追跡・検証可能性が向上**
オープンサイエンスの流れにも合致

形式や含める情報など、**フォーマットの統一は課題**



公開データ・プログラム

サンプル(ダミー)データ：https://github.com/k-nkmt/synth processed collections

- 国勢調査：2000, 2005, 2010, 2015, 2020
- 賃金構造基本統計調査：2017, 2018, 2019

データ生成の条件などは各フォルダのreadmeに記載

変換関連プログラム：https://github.com/k-nkmt/harmonize with meta

- 変換時の辞書データ・プログラム
- YAML形式メタデータ利用例(ipynb)

  メタデータの読み込み、プログラムの生成等

フォルダ構造などはreadmeに記載

不具合や要望などあれば、issueまたはメールにお願いします

ご清聴ありがとうございました

お問い合わせ等

株式会社 Spark Vision

〒113-0022 東京都文京区千駄木 4-20-5 Glanz文京千駄木 201

URL: https://spark-v.com

E-mail: k.nakamatsu@spark-v.com