証拠に基づく政策立案のための オープンデータを利活用した WebGIS 可視化によるデータフュージョン

長瀬　永遠

富山県立大学　情報基盤工学講座

December 17, 2021

提案システムの概要

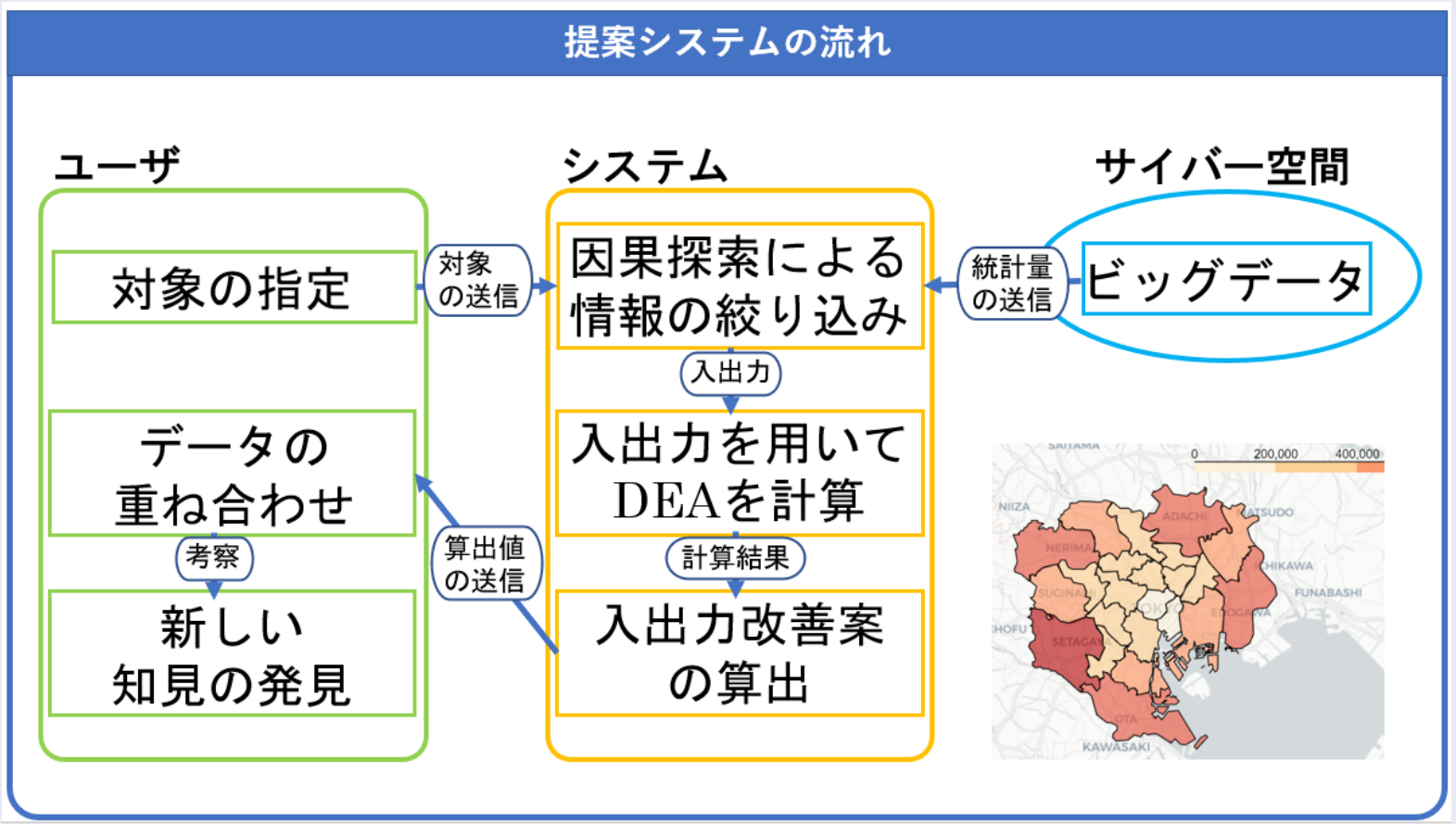

図 1: システムのフロー

ここまでの進捗 1

データベース

API の部分のみを項目に対応したものに変えることで全国の市区町村について最新年の統計データを取得してくる（＋自動的に必要な部分のみを抜き出す）ことができるようなコードを作成した.

```
12  #APIのデータ種部分の入力
13  url_base = 'https://opendata.resas-portal.go.jp/api/v1/municipality/employee/perYear'
14
15  l = [] #元データを格納するリスト
16
17
18  """
19  元データの取得（注1：存在しないcityCodeを入力するとエラーが出る）
20  """
21
22  #dataFrameの本体の作成
23  for city_code in range(1100, 1101): #range部分にcityCodeの範囲を入力（注1）
24      url = url_base + '?cityCode=0' + str(city_code) + '&prefCode=1' #url生成
25      req = urllib.request.Request(url, headers=api_key) #RESASへリクエスト
26      with urllib.request.urlopen(req) as response: #応答の読み込み
27          data = response.read()
28      d = json.loads(data.decode()) #json形式にデコード
29      l.append(pd.json_normalize(d['result'])) #データをpandas Seriesに直してリストに格納
30      print(url) #urlが正しいかの確認用
31  df_all = pd.concat(l, ignore_index=True) #リスト内のデータを結合（pandas dataFrame化）
32
33  #dataFrameにデータを追加する関数の定義
34  def get_data(s_cC, e_cC, c_head, pref_code, df_all):
35      l = [] #リストlのリフレッシュ
36      for city_code in range(s_cC, e_cC): #range部分にcityCodeの範囲を入力（注1）
37          url = url_base + '?cityCode=' + str(c_head) + format(city_code, '04') + '&prefCode=' + str(pref_code) #url生成
38          req = urllib.request.Request(url, headers=api_key) #RESASへリクエスト
39          with urllib.request.urlopen(req) as response: #応答の読み込み
40              data = response.read()
```

図 2: データ抽出のコード

ここまでの進捗 2.1

包絡分析法を用いた効率値および改善値の算出

包絡分析の入力指向・出力指向を線形計画法によって解くプログラムを Python で作成した．また，その結果をもとに入力，出力のそれぞれの項目について改善値を算出するプログラムも同時に作成した．

```
40  #目的関数の設定
41  problem1 += gamma
42  target = 3
43  #制約式の設定
44  for p_c_in in range(len(df_in.columns)):
45      problem_in = 0
46      problem = 0
47      for p_r in range(len(df_in)):
48          problem += df_in.iat[p_r, p_c_in] * l_lam[p_r]
49      problem += -(df_in.iat[target, p_c_in] * gamma)
50      problem_in = problem <= 0
51      problem1 += problem_in
52
53  for p_c in range(len(df_out.columns)):
54      problem_out = 0
55      problem = 0
56      for p_r in range(len(df_out)):
57          problem += df_out.iat[p_r, p_c] * l_lam[p_r]
58      problem_out = problem >= df_out.iat[target, p_c]
59      problem1 += problem_out
60
61  #計算の実行
62  problem1.solve()
```

図 3: 包絡分析法のプログラム

使用したデータセットと実行結果

	病院	A	B	C	D	E	F	G
入力	医師数	17	58	72	19	11	54	8
	平均入院日数	15.5	29.1	15.8	20.4	19.2	13.1	24.9
出力	1 日平均患者数	266	661	1695	514	543	1447	390
	医業収益（10 億円）	2.30	5.60	9.79	2.68	2.21	11.05	1.82

図 4: 使用したデータセット

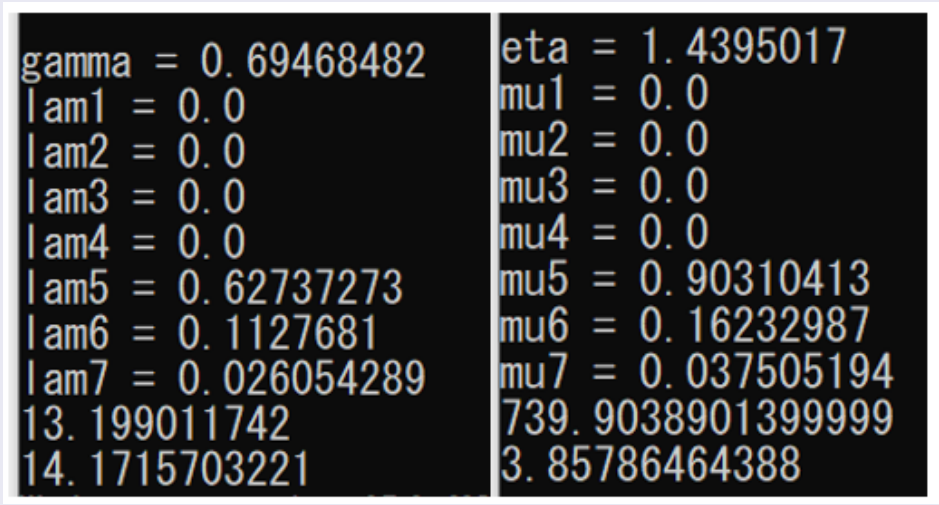

図 5: 実行結果

今後の課題

来週からやること

1. 全てのプログラムを一度に動かせるシステムのプロトタイプを作成.
2. 本論の作成.
3. 研究の有効性に関するエビデンスのサーベイと意見の構築
4. WIKI を書き足す.