

# 13. 佐賀の気候

プログラミング・データサイエンス I

2024/7/4

## 1 今日の目的

今日の目的

- Web ページ内の表を取得
- データをきれいにする
- 作図する

Web ページの中に表がある場合があります。気象庁のページの中には、いくつかの観測地点での、毎月の気温などをまとめた表があります。今回は、以下の URL にある、佐賀市の最高気温の変化を使います。この表を取り出して、毎月の最高気温の変化を表すグラフを作成しましょう。

`http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?prec_no=85&block_no=47813&year=&month=&day=&elm=monthly&view=a2`

これまでの講義で使った佐賀県の人口推移の Excel などでは、不要な行の削除などの前処理が必要でした。Web 上の表も前処理が必要となります。

まずは、Web ブラウザを使って上記のページを見てください。一番右に「年の値」という列があります。毎月の変化を図示するには、不要な部分です。

また、2011 年 8 月のデータには、")"が入っています。2024 年のデータには、更"]"が入っています。")"は、観測データ数が完全では無いという意味で使われています。数字でないものは、データとして使えません。何か、対処が必要になります。なお、2024 年は、始まったばかりなので、今回は使わないことにします。

このような前処理を順番に進めていきましょう。今回は、作図できる値になるように一歩一歩進めていきます。それでは、サンプルプログラムを取得してください。

`https://github.com/first-programming-saga/withURL`

## 2 データをきれいにする

データをきれいにする

- HTML から対象となる表を取り出す
- 不要な列を削除する
- 行の内容を確認する
- データをきれいにする

ソースコード 2.1 SagaTenki.ipynb

```
1 import numpy
2 import pandas
3 import matplotlib.pyplot as plt
4 import japanize_matplotlib
5 import re
6
7 url = 'http://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?'
8 url +=
9     ↪ 'prec_no=85&block_no=47813&year=&month=&day=&elm=monthly&view=a2'
10
11 dataFrames = pandas.read_html(url)
12 print('表の数 ', len(dataFrames))
13 for i, d in enumerate(dataFrames):
14     print(f'----- {i} -----')
15     print(d.head())
```

それでは、SagaTenki.ipynb を開いてください。実行すると、この URL には、二つの表があり、最初が気温の変化、二番目が URL の下のほうにあるリンクであることが解ります。dataFrames[0] が、対象となる DataFrame であることが分かりました。なお、DataFrame の head() というメソッドは、表の最初の部分を表示します。

ソースコード 2.2 「年の値」の列を削除

```
1 dataFrames = pandas.read_html(url)
2 df = dataFrames[0].drop('年の値', axis = 1)
3 df.set_index('年', inplace = True)
4 df
```

最初に、表の「年の値」の列を削除しましょう (ソースコード 2.2)。drop() メソッドは、指定した行や列を削除した DataFrame を返します。axis='columns' が、列を削除することを表しています。また、「年」の列をインデクスに指定しましょう。実行し、「年の値」の列が無くなっていることを確かめてください。

前述のように、2011 年のデータには、")"が入っていました。他の年はどうでしょうか。例として 2010 年のデータを見ましょう (ソースコード 2.3)。出力結果 (出力例 2.4) をみると、object という型のデータと分かります。つまり、数値でないものが混じっていることがわかります。

#### ソースコード 2.3 2010 年の行

```
1 dataFrames = pandas.read_html(url)
2 df = dataFrames[0].drop('年の値', axis = 1)
3 df.set_index('年', inplace = True)
4 ser :pandas.Series = df.loc[2010] #type: ignore
5 ser
```

更に、Series 内の要素の型を確かめましょう (ソースコード 2.5)。出力例 2.6 のようになりました。numpy.float64 は、Pandas が表の中の小数に使う型です。問題なのは、二つ str 型、つまり文字列が含まれていました。これでは、気温をプロットすることはできません。

Series の値を一括して小数型に変換するには、astype() メソッドがあります。しかし、今回は、2011 年のデータのことも考えて、Series の要素を一つ一つ確かめて、必要

#### ソースコード 2.4 2010 年のデータ

```
1月      9.8
2月     13.2
3月     14.6
4月     19.1
5月     24.8
6月     28.3
7月     31.4
8月     34.7
9月     30.3
10月    24.1
11月    17.6
12月    12.0
Name: 2010, dtype: object
```

### ソースコード 2.5 Series の要素の型を調べる

```
1 for k in ser.index:
2     v = ser[k]
3     print(v,type(v))
```

### ソースコード 2.6 2010 年のデータの型

```
9.8 <class 'numpy.float64'>
13.2 <class 'str'>
14.6 <class 'numpy.float64'>
19.1 <class 'numpy.float64'>
24.8 <class 'numpy.float64'>
28.3 <class 'numpy.float64'>
31.4 <class 'numpy.float64'>
34.7 <class 'str'>
30.3 <class 'numpy.float64'>
24.1 <class 'numpy.float64'>
17.6 <class 'numpy.float64'>
12.0 <class 'numpy.float64'>
```

に応じて変更することを考えましょう。ソースコード 2.7 を見てください。この部分は、配布ファイルに含んでいます。

### ソースコード 2.7 データを数値に

```
1 pattern: re.Pattern[str] = re.compile(r'([-]*\d+\.\d*)\D*')
2 def cleanSeries(serIn:pandas.Series) -> pandas.Series:
3     """
4     列データに含まれる、数値以外の文字を除く
5     """
6     ser:pandas.Series = serIn.copy()
7     for k in ser.index:
8         v = ser[k]
9         if type(v) is str:
10            m: re.Match[str] | None = pattern.match(v)
11            if m:
12                v = m.group(1)
13            ser[k]=numpy.float64(str(v))
14     return ser
```

気温のデータの後ろに、スペースが入り、その後に) や] が、現れています。そこで、スペースが現れる前の部分だけを切り出して、数値の型に変換後、上書きすることにしま

す。ここでは、具体的な文字列ではなく、文字列のパターンを探す正規表現というものを使っています。正規表現を理解して使うことができると、テキストを扱うプログラムを書く際の強力な武器になります。興味のある人は、調べてみてください。

1行目の `(\S*)\s` が、空白以外の文字の後に空白が続くパターンを表しています。forループで、Seriesの要素を一つ一つ取り出し、それが文字列である場合(5行目)に処理します。文字列中に空白があれば、空白の前の部分を取り出します。9行目で、文字列を `numpy.float64` に変換します。このようにすれば、数値が文字列として表現されている場合も含めて、小数の型に変更することができます。ソースコード 2.8 を実行すると、全てが `numpy.float64` になっていることが解ります。これで、一番古い 1890 年と最新の年以外のデータは作図できそうです。

ソースコード 2.8 データの変換

```
1 dataFrames = pandas.read_html(url)
2 df = dataFrames[0].drop('年の値', axis = 'columns')
3 df.set_index('年', inplace = True)
4 ser = df.loc[2010]
5 cleanSeries(ser)
6 for k in ser.index:
7     v = ser[k]
8     print(v,type(v))
```

### 3 作図

#### 作図

- 1年だけ作図する
- 複数年作図する

最後に作図をしましょう。始めに、1年だけを作図しましょう。ソースコード 3.1 を見てください。もう、詳しい説明は不要でしょう。9行目は、横軸に毎月の名前を出すための処理です。これがないと、隔月に名前が出てしまいます。なお、ソースコード 3.2 は、DataFrame から指定した年の情報を取り出し、整形する部分です。ソースコード 3.1 まで作業したものは、`SagaTenkiFinal.ipynb` として配布しています。作図結果を図 1 に示します。

### ソースコード 3.1 2010年の最高気温変化

```
1 def drawTemperature(ser: pandas.Series, year: int) -> None:
2     fig, ax = plt.subplots(facecolor= 'w')
3     ax.set_title('佐賀市の最高気温')
4     ax.set_xlim(0, 11)
5     ax.set_ylim(0, 40)
6     ax.set_xlabel('月')
7     ax.set_ylabel('気温 (摂氏)')
8     ax.set_xticks([x for x in range(12)])
9
10    ser.plot(ax = ax, label = str(year), linewidth = 1)
11
12    ax.legend(loc = 'best')
13    fig.savefig('SagaTemperature.pdf')
14    fig.show()
```

### ソースコード 3.2 1年分の最高気温を取り出す

```
1 year = 2022
2 ser :pandas.Series = df.loc[year] #type: ignore
3 ser :pandas.Series = cleanSeries(ser)
```

**課題 3.1** ソースコード 3.1 を参考に、2005 年、2010 年、2015 年、2020 年の最高気温の毎月の変化を作図するプログラムを作成しなさい。また、その動作を確認しなさい。

ヒント：関数 `drawTemperature()` の引数に、`ser` のリスト、`year` のリストを渡す。

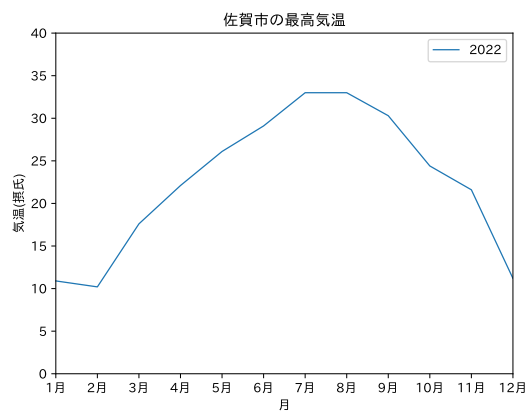


図1 2022年の佐賀市における最高気温変化