クレジット:
UTokyo Online Education Education コンピュータシステム概論 2018 小林克志

東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

Pandas 追加

この資料はThe Python Tutorial (https://docs.python.org/3.6/tutorial/index.html#the-python-tutorial) (日本語版 (https://docs.python.jp/3/tutorial/)) および Python for Data Analysis:Wrangling with Pandas, Numpy and IPython (http://shop.oreilly.com/product/0636920050896.do)を参考に作成した。

インデックスの階層化

Multiindex は pandas の重要な機能のひとつ。 Multiindex による階層化によって多次元データをより低次元の処理系での取り扱いを実現する。

`pd.Series` の Multiindex

`pd.Series` クラスの Multiindex は:

In [10]:
```
import numpy as np
import pandas as pd
ser = pd.Series(np.random.randn(8),
                index=[["a", "a", "b", "c", "d", "d", "e", "e"],
                       ["2000","2010","2000","2000","2000","2010
","2000","2010"]])
ser
```

Out[10]:
```
a  2000    0.608605
   2010    0.117970
b  2000    0.165885
c  2000   -0.709062
d  2000   -0.562112
   2010   -0.277552
e  2000    0.171508
   2010   -1.111910
dtype: float64
```

これをインデックスで指定すると:

In [11]:
```
ser["a"]
```

Out[11]:
```
2000    0.608605
2010    0.117970
dtype: float64
```

In [12]:
```
ser["b":"c"]
```

Out[12]:
```
b  2000    0.165885
c  2000   -0.709062
dtype: float64
```



```
In [13]: ser.loc[["a", "d"]]
```

```
Out[13]: a  2000    0.608605
            2010    0.117970
         d  2000   -0.562112
            2010   -0.277552
         dtype: float64
```

`iloc()` の働きは（もちろん）同じ:

```
In [14]: ser.iloc[2:4]
```

```
Out[14]: b  2000    0.165885
         c  2000   -0.709062
         dtype: float64
```

低い階層のインデックスも指定するには、コンマで区切る:

```
In [15]: ser["a","2000"]
```

```
Out[15]: 0.6086054366042096
```

インデックスからデータセットを作り直すには、`pd.unstack()` が使える:

```
In [16]: ser.unstack()
```

Out[16]:

	2000	2010
a	0.608605	0.117970
b	0.165885	NaN
c	-0.709062	NaN
d	-0.562112	-0.277552
e	0.171508	-1.111910

`pd.stack()` は逆をおこなう:

```
In [17]: ser.unstack().stack()
```

```
Out[17]: a  2000    0.608605
            2010    0.117970
         b  2000    0.165885
         c  2000   -0.709062
         d  2000   -0.562112
            2010   -0.277552
         e  2000    0.171508
            2010   -1.111910
         dtype: float64
```


pd.DataFrame の Multiindex

pd.DataFrame では行・列いずれにも Multiindex は適用できる:

In [18]:
```
import numpy as np
import pandas as pd
df = pd.DataFrame(np.random.randn(24).reshape(6,4),
                  index =[["a", "a", "b", "b", "c", "c"],["2000",
"2010", "2000", "2010", "2000", "2010"]],
                  columns = [["Tokyo", "Tokyo", "Saitama", "Nagano
"],["Mainland", "Israndes", "Mainland", "Mainland"]])
df
```

Out[18]:

		Tokyo		Saitama	Nagano
		Mainland	Israndes	Mainland	Mainland
a	2000	0.069428	2.546802	1.457338	-0.569397
	2010	0.460472	0.031973	-0.063079	-0.608645
b	2000	1.841005	-0.667141	-0.110929	-0.327752
	2010	-0.277205	0.300749	-2.123452	1.265280
c	2000	-0.454824	1.104094	-1.671802	-0.148097
	2010	0.583424	-1.297794	-0.512486	-0.485872

インデックスの指定は、pd.Seriesと同じ:

In [19]:
```
df["Tokyo"]
```

Out[19]:

		Mainland	Israndes
a	2000	0.069428	2.546802
	2010	0.460472	0.031973
b	2000	1.841005	-0.667141
	2010	-0.277205	0.300749
c	2000	-0.454824	1.104094
	2010	0.583424	-1.297794

In [20]:
```
df["Tokyo","Mainland"]
```

Out[20]:
```
a  2000    0.069428
   2010    0.460472
b  2000    1.841005
   2010   -0.277205
c  2000   -0.454824
   2010    0.583424
Name: (Tokyo, Mainland), dtype: float64
```


In [21]:
```
df.loc["a"]
```

Out[21]:

	Tokyo		Saitama	Nagano
	Mainland	Isrands	Mainland	Mainland
2000	0.069428	2.546802	1.457338	-0.569397
2010	0.460472	0.031973	-0.063079	-0.608645

In [27]:
```
df["Tokyo","Mainland"]
```

Out[27]:
```
a  2000    0.069428
   2010    0.460472
b  2000    1.841005
   2010   -0.277205
c  2000   -0.454824
   2010    0.583424
Name: (Tokyo, Mainland), dtype: float64
```

ただし、pd.DataFrame なのでスライスは行に対して適用される:

In [ ]:
```
df["a":"b"]
```

Pandas のデータ読み込み

Pandas では CSV, XLS, JSON 形式といった様々な形式をシステムコールを使うよりも簡単にあつかうことができる。

CSV 形式

CSV 形式の読み込みには、pd.read_csv()を使う。
以下の例では、Yahoo! Finance から取得したビットコイン-USD 交換レートデータ(https://finance.yahoo.com/quote/BTC-USD/history/)を読み込む。
pd.read_csv() では Default で 1 行目、1 列目がインデックスとして利用される:

In [35]:
```
import pandas as pd
df = pd.read_csv("BTC-USD.csv", index_col = 0, parse_dates=True)
df
```

Out[35]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2010-07-16	0.049510	0.049510	0.049510	0.049510	0.049510	0
2010-07-17	0.049510	0.085850	0.059410	0.085840	0.085840	5
2010-07-18	0.085840	0.093070	0.077230	0.080800	0.080800	49

2010-07-19	0.080800	0.081810	0.074260	0.074740	0.074740	20
2010-07-20	0.074740	0.079210	0.066340	0.079210	0.079210	42
2010-07-21	0.079210	0.081810	0.050500	0.050500	0.050500	129
2010-07-22	0.050500	0.067670	0.050500	0.062620	0.062620	141
2010-07-23	0.062620	0.061610	0.050490	0.054540	0.054540	26
2010-07-24	0.054540	0.059410	0.050500	0.050500	0.050500	85
2010-07-25	0.050500	0.056000	0.050000	0.056000	0.056000	46
2010-07-26	0.056000	0.060500	0.053000	0.060000	0.060000	196
2010-07-27	0.060000	0.062000	0.054000	0.058900	0.058900	255
2010-07-28	0.058900	0.069900	0.057100	0.069900	0.069900	528
2010-07-29	0.069900	0.069800	0.058200	0.062700	0.062700	198
2010-07-30	0.062700	0.068890	0.056000	0.067850	0.067850	243
2010-07-31	0.067850	0.065000	0.060000	0.061100	0.061100	162
2010-08-01	0.061100	0.063300	0.060000	0.060000	0.060000	221
2010-08-02	0.060000	0.065000	0.059000	0.060000	0.060000	606
2010-08-03	0.060000	0.062310	0.057000	0.057000	0.057000	210
2010-08-04	0.057000	0.061000	0.058000	0.061000	0.061000	303
2010-08-05	0.061000	0.062400	0.060700	0.062300	0.062300	85
2010-08-06	0.062300	0.062200	0.059000	0.059000	0.059000	157
2010-08-07	0.059000	0.061000	0.059000	0.060900	0.060900	132

2010-08-08	0.060900	0.073500	0.059300	0.071000	0.071000	886
2010-08-09	0.071000	0.070900	0.066510	0.070000	0.070000	88
2010-08-10	0.070000	0.075410	0.060000	0.067000	0.067000	1015
2010-08-11	0.067000	0.070000	0.061410	0.070000	0.070000	134
2010-08-12	0.070000	0.068000	0.064500	0.064500	0.064500	233
2010-08-13	0.064500	0.069500	0.064500	0.067000	0.067000	295
2010-08-14	0.067000	0.067000	0.065000	0.065290	0.065290	294
...	...	...	...	...	...	...
2018-04-02	7074.649902	7528.970215	7039.879883	7434.299805	7434.299805	9053267
2018-04-03	7434.299805	7442.419922	6727.089844	6815.500000	6815.500000	8771997
2018-04-04	6815.500000	6933.660156	6602.379883	6790.450195	6790.450195	7771451
2018-04-05	6790.450195	6869.529785	6526.669922	6634.859863	6634.859863	6056512
2018-04-06	6634.859863	7083.850098	6624.470215	6917.200195	6917.200195	5648831
2018-04-07	6917.200195	7132.029785	6911.020020	7049.919922	7049.919922	4258255
2018-04-08	7049.919922	7204.279785	6633.689941	6789.529785	6789.529785	9018836
2018-04-09	6789.529785	6922.839844	6676.720215	6871.069824	6871.069824	4245109
2018-04-10	6871.069824	6999.370117	6839.160156	6977.129883	6977.129883	4501413
2018-04-11	6977.040039	8051.950195	6791.330078	7927.729980	7927.729980	1490759
2018-04-12	7927.729980	8237.160156	7758.799805	7899.109863	7899.109863	1171435
2018-04-13	7898.740234	8195.339844	7841.189941	8022.509766	8022.509766	5372863

2018-04-14	8022.509766	8437.030273	8021.040039	8376.730469	8376.730469	570527
2018-04-15	8376.730469	8430.929688	7929.240234	8079.770020	8079.770020	713471
2018-04-16	8079.770020	8187.600098	7858.120117	7921.629883	7921.629883	580880
2018-04-17	7921.649902	8246.570313	7908.919922	8189.959961	8189.959961	593500
2018-04-18	8189.919922	8321.009766	8136.419922	8301.820313	8301.820313	602099
2018-04-19	8301.820313	8945.139648	8243.490234	8877.080078	8877.080078	979994
2018-04-20	8877.150391	9047.820313	8631.179688	8935.719727	8935.719727	829073
2018-04-21	8935.849609	9043.980469	8788.440430	8823.360352	8823.360352	610183
2018-04-22	8823.459961	9025.730469	8804.200195	8968.250000	8968.250000	553892
2018-04-23	8967.860352	9741.910156	8957.679688	9655.769531	9655.769531	127646
2018-04-24	9657.690430	9765.230469	8757.059570	8873.620117	8873.620117	177980
2018-04-25	8873.570313	9315.129883	8669.379883	9282.120117	9282.120117	997427
2018-04-26	9289.009766	9385.870117	8923.480469	8938.469727	8938.469727	758182
2018-04-27	8938.469727	9435.900391	8892.519531	9351.469727	9351.469727	774111
2018-04-28	9349.940430	9552.669922	9189.070313	9407.040039	9407.040039	640931
2018-04-29	9407.349609	9459.809570	9133.599609	9248.450195	9248.450195	574099
2018-04-30	9248.250000	9251.660156	8851.099609	9077.280273	9077.280273	669555
2018-05-01	9077.280273	9271.620117	8993.820313	9232.190430	9232.190430	527488

2847 rows × 6 columns

Microsoft Excel XLS 形式

Microsoft Exel の XLS 形式の読み込みには、`pd.read_xls()`を使う。
以下の例は、総務省が公開している、住民基本台帳に基づく人口動態データ(http://www.soumu.go.jp/menu_news/s-news/01gyosei02_02000148.html)を読み込んでいる:

- 列インデックスは、2-4 行目を Multiindex としてあつかっている
- 行インデックスは都道府県名で置き換えている

In [38]:
```
import pandas as pd
import numpy as np

df = pd.read_excel("000494956.xls", sheet_name=0, header=[1,2,3],
skiprows=[4])
# Omit "Unnamed" indices and adjust
for i, col in enumerate(df.columns.levels):
    columns = np.where(col.str.contains("Unnamed"), "", col)
    df.columns.set_levels(columns, level=i, inplace=True)
df.set_index("都道府県名", inplace=True)
df.index.name="都道府県名"
df
```

Out[38]:

団体コード	平成29年				平成28年				
	人口			世帯数	住民票記載数				
	男	女	計		転入者数（国内）	転入者数（国外）	転入者数（計）	出生者数	その他（計）
都道府県名									
北海道	2537340	2833467	5370807	2761826	245573	13047	258620	35452	2519
青森県	627006	696855	1323861	589887	34255	1880	36135	8684	519
岩手県	613838	663433	1277271	523065	36207	1870	38077	8363	325

宮城県	1131759	1187679	2319438	980808	106795	5049	111844	17569	1158
秋田県	485257	543939	1029196	426020	21489	1270	22759	5692	330
山形県	538338	580130	1118468	411919	28062	1356	29418	7578	240
福島県	950430	988129	1938559	779244	54421	2781	57202	13810	837
茨城県	1482072	1478386	2960458	1221978	98859	15798	114657	21383	1325
栃木県	993019	998578	1991597	817370	58970	10576	69546	14904	806
群馬県	988955	1009320	1998275	831970	60784	8721	69505	14201	1038
埼玉県	3679223	3664584	7343807	3212080	321112	26278	347390	56108	6306
千葉県	3137814	3145788	6283602	2811702	279946	33494	313440	46656	4155
東京都	6675004	6855049	13530053	6994147	886814	117674	1004488	115742	16667
神奈川県	4574910	4580479	9155389	4236072	441970	38089	480059	72794	16088
新潟県	1116170	1184753	2300923	890293	57884	3967	61851	15853	267
富山県	520105	554600	1074705	414865	24318	3737	28055	7463	294
石川	558006	595621	1153627	478395	34315	4021	38336	8985	412

県									
福井県	385110	409323	794433	289825	18126	2712	20838	6171	143
山梨県	413919	430798	844717	356363	27740	2982	30722	5925	443
長野県	1036838	1089226	2126064	861074	63213	7930	71143	15393	917
岐阜県	1004919	1061347	2066266	809888	60365	11798	72163	15254	1199
静岡県	1855174	1901691	3756865	1557733	128828	14885	143713	28423	1548
愛知県	3775270	3756961	7532231	3214669	329622	45586	375208	66567	5519
三重県	899701	942052	1841753	782840	55244	8045	63289	13797	992
滋賀県	699932	720328	1420260	566148	48032	5806	53838	12306	717
京都府	1232089	1337321	2569410	1202380	109701	11820	121521	19686	1780
大阪府	4283835	4577602	8861437	4223735	387230	35800	423030	70281	5454
兵庫県	2689585	2916960	5606545	2507945	196458	17726	214184	44135	3236
奈良県	656316	723865	1380181	587413	41808	3475	45283	9539	704
和歌山県	466136	518553	984689	440150	22502	1500	24002	6704	438
鳥									

取県	274794	300470	575264	235502	15404	1135	16539	4473	168
島根県	333255	363127	696382	288790	18968	1989	20957	5337	257
岡山県	929475	998157	1927632	835989	65197	7542	72739	15627	392
広島県	1385401	1472074	2857475	1300322	110217	13268	123485	23074	1662
山口県	668030	740558	1408588	659804	41468	3139	44607	9948	417
徳島県	364700	399513	764213	334117	21316	2032	23348	5381	313
香川県	482078	515733	997811	436123	30147	3551	33698	7606	430
愛媛県	666134	739191	1405325	651763	36137	3731	39868	9947	485
高知県	345444	387091	732535	352694	20632	1228	21860	4838	284
福岡県	2436091	2690298	5126389	2371459	256749	18896	275645	44603	3661
佐賀県	397243	440734	837977	328015	25637	1722	27359	6871	370
長崎県	656314	736636	1392950	635020	44442	5167	49609	10974	496
熊本県	850608	947541	1798149	770607	75804	4141	79945	15026	1033
大分県	558646	618245	1176891	533406	34224	3091	37315	9127	565
宮									

崎県	529293	590251	1119544	521627	34839	1604	36443	9009	647
鹿児島県	785168	882835	1668003	807169	60982	2150	63132	13756	743
沖縄県	723531	743540	1467071	632826	71989	5015	77004	16766	1271

47 rows × 22 columns

