需求：

- 导入文件,查看原始数据
- 将人口数据和各州简称数据进行合并
- 将合并的数据中重复的abbreviation列进行删除
- 查看存在缺失数据的列
- 找到有哪些state/region使得state的值为NaN,进行去重操作
- 为找到的这些state/region的state项补上正确的值,从而去除掉state这一列的所有NaN
- 合并各州面积数据areas
- 我们会发现area(sq.mi)这一列有缺失数据, 找出是哪些行
- 去除含有缺失数据的行
- 找出2010年的全民人口数据
- 计算各州每一年的相应年龄段的人口密度
- 排序，并找出人口密度最高的五个州

为了解决上述问题，很多知识在那几个ipynb里可以找到，但是也有一些新内容。以后再工作或学习中，我们需要逐步往我们的知识体系 中去增添这些新内容。

In [1]:

```
import numpy as np
from pandas import DataFrame,Series
import pandas as pd
```

In [77]:

```
# 1. read 系列的 read_csv 读取数据
# 各州别名
abb = pd.read_csv('./data/population/state-abbrevs.csv')
# 人口数据
pop = pd.read_csv('./data/population/state-population.csv')
# 区域面积
area = pd.read_csv('./data/population/state-areas.csv')
```

In [9]:

```
# 2. 将人口数据和各州简称数据进行合并：使用merge进行数据合并
abb.head(1)
```

Out[9]:

	state	abbreviation
0	Alabama	AL

In [10]:

```
pop.head(1)
```

Out[10]:

	state/region	ages	year	population
0	AL	under18	2012	1117489.0

In [18]:

```
pop_abb = pd.merge(pop, abb, left_on="state/region", right_on="abbreviation", how="
```

In [19]:

```
# 3. 将合并的数据中重复的abbreviation列进行删除：使用删除列的drop方法即可
pop_abb.drop("abbreviation", axis=1, inplace=True)
```

In [28]:

```
# 4. 查看存在缺失数据的列： 这里查看的是列，使用isnull和any的组合或者notnull和all的组合
pop_abb.isnull().any(axis=0)
pop_abb.columns[pop_abb.isnull().any(axis=0)]
```

Out[28]:

```
Index(['population', 'state'], dtype='object')
```

In [29]:

```
# 5. 找到有哪些state/region使得state的值为NaN,进行去重操作：就是寻找state是NaN的行的state/re
pop_abb.head(2)
```

Out[29]:

	state/region	ages	year	population	state
0	AL	under18	2012	1117489.0	Alabama
1	AL	total	2012	4817528.0	Alabama

In [30]:

```
pop_abb["state/region"][pop_abb["state"].isnull()].unique()
```

Out[30]:

```
array(['PR', 'USA'], dtype=object)
```

In [32]:

```
# 6. 为找到的这些state/region的state项补上正确的值，从而去除掉state这一列的所有NaN
# 上面的5中寻找到的 state/region的state为NaN，说明在做merge的时候 pop的 state/region 列有值
# 这里不妨假设PR对应的state是Portuguese Republic(葡萄牙共和国)，USA对应America，使用replac
dic = {"PR": "Portuguese Republic", "USA": "America"}
pop_abb.replace(dic, inplace=True)
```

In [37]:

```
# 7. 合并各州面积数据areas
pop_abb.head(1)
```

Out[37]:

	state/region	ages	year	population	state
0	AL	under18	2012	1117489.0	Alabama

In [36]:

```
area.head(1)
```

Out[36]:

	state	area (sq. mi)
0	Alabama	52423

In [39]:

```
pop_abb_area = pd.merge(pop_abb, area, on="state", how='outer')
```

In [40]:

```
# 8. 我们会发现area(sq.mi)这一列有缺失数据, 找出是哪些行: 使用isnull和布尔索引
pop_abb_area.head(1)
```

Out[40]:

	state/region	ages	year	population	state	area (sq. mi)
0	AL	under18	2012.0	1117489.0	Alabama	52423.0

In [42]:

```
pop_abb_area.loc[pop_abb_area["area (sq. mi)"].isnull()].index
```

Out[42]:

```
Int64Index([2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 245
7, 2458,
            2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 246
8, 2469,
            2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 247
9, 2480,
            2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 249
0, 2491,
            2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 250
1, 2502,
            2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 251
2, 2513,
            2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 252
3, 2524,
            2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 253
4, 2535,
            2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543],
           dtype='int64')
```

In [44]:

```
# 9. 去除含有缺失数据的行: 使用drop并指定轴向即可
null_index = pop_abb_area.loc[pop_abb_area["area (sq. mi)"].isnull()].index
pop_abb_area.drop(labels=null_index, axis=0, inplace=True)
```

In [45]:

```
# 10. 找出2010年的全民人口数据:  这里的人口数据分为18岁以下的, 和全名的total
pop.head(1)
```

Out[45]:

	state/region	ages	year	population
0	AL	under18	2012	1117489.0

这里的场景是根据多个条件去查询DataFrame，使用query方法, 这是需要补充的一点

In [46]:

```
pop.query("year == 2010 & ages == 'total'")
```

Out[46]:

	state/region	ages	year	population
3	AL	total	2010	4785570.0
91	AK	total	2010	713868.0
101	AZ	total	2010	6408790.0
189	AR	total	2010	2922280.0
197	CA	total	2010	37333601.0
283	CO	total	2010	5048196.0
293	CT	total	2010	3579210.0
379	DE	total	2010	899711.0
389	DC	total	2010	605125.0
475	FL	total	2010	18846054.0
485	GA	total	2010	9713248.0
570	HI	total	2010	1363731.0
581	ID	total	2010	1570718.0
666	IL	total	2010	12839695.0
677	IN	total	2010	6489965.0
762	IA	total	2010	3050314.0
773	KS	total	2010	2858910.0
858	KY	total	2010	4347698.0
869	LA	total	2010	4545392.0
954	ME	total	2010	1327366.0
965	MD	total	2010	5787193.0
1050	MA	total	2010	6563263.0
1061	MI	total	2010	9876149.0
1146	MN	total	2010	5310337.0
1157	MS	total	2010	2970047.0
1242	MO	total	2010	5996063.0
1253	MT	total	2010	990527.0
1338	NE	total	2010	1829838.0
1349	NV	total	2010	2703230.0
1434	NH	total	2010	1316614.0
1445	NJ	total	2010	8802707.0
1530	NM	total	2010	2064982.0
1541	NY	total	2010	19398228.0

	state/region	ages	year	population
1626	NC	total	2010	9559533.0
1637	ND	total	2010	674344.0
1722	OH	total	2010	11545435.0
1733	OK	total	2010	3759263.0
1818	OR	total	2010	3837208.0
1829	PA	total	2010	12710472.0
1914	RI	total	2010	1052669.0
1925	SC	total	2010	4636361.0
2010	SD	total	2010	816211.0
2021	TN	total	2010	6356683.0
2106	TX	total	2010	25245178.0
2117	UT	total	2010	2774424.0
2202	VT	total	2010	625793.0
2213	VA	total	2010	8024417.0
2298	WA	total	2010	6742256.0
2309	WV	total	2010	1854146.0
2394	WI	total	2010	5689060.0
2405	WY	total	2010	564222.0
2490	PR	total	2010	3721208.0
2539	USA	total	2010	309326295.0

In [48]:

```
# 11. 计算各州每一年的相应年龄段的人口密度
pop_abb_area.head(2)
```

Out[48]:

	state/region	ages	year	population	state	area (sq. mi)
0	AL	under18	2012.0	1117489.0	Alabama	52423.0
1	AL	total	2012.0	4817528.0	Alabama	52423.0

In [49]:

```
pop_abb_area["density"] = pop_abb_area["population"] / pop_abb_area["area (sq. mi)"]
```

In [50]:

```
pop_abb_area.head(1)
```

Out[50]:

	state/region	ages	year	population	state	area (sq. mi)	density
0	AL	under18	2012.0	1117489.0	Alabama	52423.0	21.316769

In [75]:

```
# 12. 排序，并找出人口密度最高的五个州，使用sort_values
pop_abb_area.sort_values(by='density',axis=0,ascending=False)["state"].unique()[:5]
```

Out[75]:

```
array(['District of Columbia', 'New Jersey', 'Rhode Island',
       'Connecticut', 'Massachusetts'], dtype=object)
```

In [ ]:

```

```