

# 파이썬 라이브러리를 활용한 데이터 분석

## 14장 데이터 분석 예제

2020.07.10금 2h

# 14장 데이터 분석 예제

## 미국 신생아 이름 분석

2h

# 미국의 신생아 이름 정보

## • 미국사회보장국(SSA)

- 1880년부터 현재까지 가장 빈도가 높은 신생아 이름 정보 제공

## • 실습 파일

- dataset/babynames/yob1880.txt
  - 1880년, 최소 5명 이상 중복되는 이름만 제공
    - 이름(name)
    - 성별(sex)
    - 출생아수(births)
- 년도별 csv 파일 제공
  - 2010년까지 파일
    - 파일 yob2010.txt

## • 성별 출생아수, 1880년

- names1880.groupby('sex').births.sum()
  - 4명 이하의 이름은 없으나
    - 편의상 합이 출생아 수

```
In [34]: import pandas as pd
names1880 = pd.read_csv('datasets/babynames/yob1880.txt',
                        names=['name', 'sex', 'births'])
names1880
```

Out[34]:

	name	sex	births
0	Mary	F	7065
1	Anna	F	2604
2	Emma	F	2003
3	Elizabeth	F	1939
4	Minnie	F	1746
...	...	...	...
1995	Woodie	M	5
1996	Worthy	M	5
1997	Wright	M	5
1998	York	M	5
1999	Zachariah	M	5

2000 rows × 3 columns

```
In [35]: names1880.groupby('sex').births.sum()
```

Out[35]:

sex	
F	90993
M	110493
Name: births, dtype: int64	

# 모든 자료를 하나의 DataFrame으로

## • 년도별 파일을 읽어

- 데이터프레임의 리스트를 생성
- Pandas.concat(리스트)로 이 리스트를 합침
  - 옵션 `ignore_index=True`
    - 원래의 순서를 무시하고 전체 인덱스로 사용

```
In [39]: pd.concat(pieces)
```

Out[39]:

	name	sex	births	year
0	Mary	F	7065	1880
1	Anna	F	2604	1880
2	Emma	F	2003	1880
3	Elizabeth	F	1939	1880
4	Minnie	F	1746	1880

기본 옵션 `ignore_index=False`인 경우 ...

33833	Zymaire	M	5	2010
33834	Zyonne	M	5	2010
33835	Zyquarius	M	5	2010
33836	Zyran	M	5	2010
33837	Zzyzx	M	5	2010

1690784 rows × 4 columns

```
In [36]: years = range(1880, 2011)

pieces = []
columns = ['name', 'sex', 'births']

for year in years:
    path = 'datasets/babynames/yob%d.txt' % year
    frame = pd.read_csv(path, names=columns)

    frame['year'] = year
    pieces.append(frame)

# Concatenate everything into a single DataFrame
names = pd.concat(pieces, ignore_index=True)
```

```
In [38]: names
```

Out[38]:


	name	sex	births	year
0	Mary	F	7065	1880
1	Anna	F	2604	1880
2	Emma	F	2003	1880
3	Elizabeth	F	1939	1880
4	Minnie	F	1746	1880
...	...	...	...	...
1690779	Zymaire	M	5	2010
1690780	Zyonne	M	5	2010
1690781	Zyquarius	M	5	2010
1690782	Zyran	M	5	2010
1690783	Zzyzx	M	5	2010

1690784 rows × 4 columns

# Pandas concat()

## • '이어 붙이기'

- 두 데이터프레임을 무조건 세로로 '이어 붙이기'
  - 구조가 같으면 이해가 매우 쉬움
  - 구조가 다르다면?



### Concat Pandas DataFrame

**student\_df1**

index	Roll No	Name	Class
0	101	Ankit	AI
1	102	John	ML
2	103	Ali	DS
3	104	Ram	Python

**student\_df2**

index	Roll No	Name	Class
0	104	Juhi	AI
1	105	Malika	ML
2	106	Josef	DS
3	107	Akansha	Python

**student\_df1 + student\_df2**

index	Roll No	Name	Class
0	101	Ankit	AI
1	102	John	ML
2	103	Ali	DS
3	104	Ram	Python
0	104	Juhi	AI
1	105	Malika	ML
2	106	Josef	DS
3	107	Akansha	Python

기본 옵션 ignore\_index=False인 경우

www.PandasTutorial.net

# 연도와 성별에 따른 출생아 수

## 메소드 `pivot_table()`, `groupby()` 사용

– 함수 `sum()`으로 총 출생아 수

### • 최근 남아가 많은 원인?

– 여아의 이름이 다양한 것도 하나의 원인일 수 있음

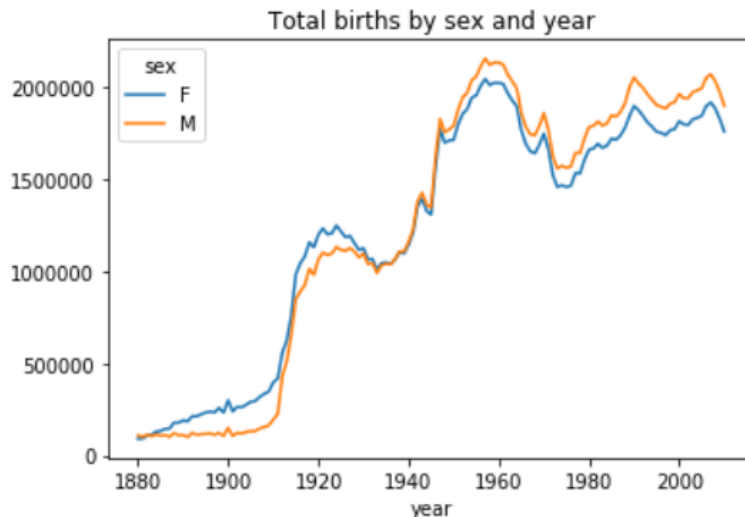
```
In [44]: total_births = names.pivot_table('births', index='year',
total_births.head()
```

Out[44]:

sex	F	M
year		
1880	90993	110493
1881	91955	100748
1882	107851	113687
1883	112322	104632
1884	129021	114445

```
In [47]: total_births.plot(title='Total births by sex and year')
```

Out[47]: <matplotlib.axes.\_subplots.AxesSubplot at 0x1e0ec7511c8>



```
5]: total_births2 = names.groupby('year')['births'].sum()
total_births.head()
```

5]:

sex	F	M
year		
1880	90993	110493
1881	91955	100748
1882	107851	113687
1883	112322	104632
1884	129021	114445

# 열 prop 추가

- 열 prop

- 성별로 해당년도 전체 출생아 수에서 차지하는 비율
- 0.02 의미
  - 100명중 2명이 같은 이름
- 연도와 성별로 그룹화하고 각 그룹에 열 prop을 추가

```
In [58]: def add_prop(group):  
         group['prop'] = group.births / group.births.sum()  
         return group  
  
names = names.groupby(['year', 'sex']).apply(add_prop)  
names[:5]
```

Out [58]:

	name	sex	births	year	prop
0	Mary	F	7065	1880	0.077643
1	Anna	F	2604	1880	0.028618
2	Emma	F	2003	1880	0.022013
3	Elizabeth	F	1939	1880	0.021309
4	Minnie	F	1746	1880	0.019188

```
In [55]: names.groupby(['year', 'sex']).sum()
```

Out [55]:

		births	prop
year	sex		
1880	F	90993	1.0
	M	110493	1.0
1881	F	91955	1.0
	M	100748	1.0
1882	F	107851	1.0
...	...	...	...
2008	M	2032310	1.0
2009	F	1827643	1.0
	M	1973359	1.0
2010	F	1759010	1.0
	M	1898382	1.0

# 연도별/성별 선호하는 이름 1000개 추출(1)

## • 연도별, 성별 그룹핑

- 출생아 수인 births로 오름차순으로 정렬한 결과
  - 1000개만 추출
- 메소드 reset\_index()
  - Index를 모두 열로 이동
  - 옵션 drop=True
    - 삽입되는 index를 제거

```
In [70]: def get_top1000(group):
          return group.sort_values(by='births', ascending=False)[:1000]

grouped = names.groupby(['year', 'sex'])
top1000 = grouped.apply(get_top1000)
top1000[:5]
```

Out[70]:

		name	sex	births	year	prop
year	sex					
1880	F	0	Mary	F	7065	1880 0.077643
		1	Anna	F	2604	1880 0.028618
		2	Emma	F	2003	1880 0.022013
		3	Elizabeth	F	1939	1880 0.021309
		4	Minnie	F	1746	1880 0.019188

```
In [71]: # Drop the group index, not needed
top1000.reset_index(inplace=True, drop=True)
top1000[:5]
```

Out[71]:

	name	sex	births	year	prop
0	Mary	F	7065	1880	0.077643
1	Anna	F	2604	1880	0.028618
2	Emma	F	2003	1880	0.022013
3	Elizabeth	F	1939	1880	0.021309
4	Minnie	F	1746	1880	0.019188



# 연도별/성별 선호하는 이름 1000개 추출(2)

## • 함수 없이 처리

- 리스트 pieces
  - 연도별 상위 1000등의 데이터프레임의 리스트
- 연도별, 성별 그룹에서
  - 년도별로 출생아 수 등수 1000개를 추출한 데이터프레임을 계속 추가
  - 결과 리스트를 concat

```
In [73]: pieces = []
for year, group in names.groupby(['year', 'sex']):
    pieces.append(group.sort_values(by='births', ascending=False)[:1000])

top1000 = pd.concat(pieces, ignore_index=True)
top1000
```

Out[73]:

	name	sex	births	year	prop
0	Mary	F	7065	1880	0.077643
1	Anna	F	2604	1880	0.028618
2	Emma	F	2003	1880	0.022013
3	Elizabeth	F	1939	1880	0.021309
4	Minnie	F	1746	1880	0.019188
...	...	...	...	...	...
261872	Camilo	M	194	2010	0.000102
261873	Destin	M	194	2010	0.000102
261874	Jaquan	M	194	2010	0.000102
261875	Jaydan	M	194	2010	0.000102
261876	Maxton	M	193	2010	0.000102

261877 rows × 5 columns

# 유행 분석 사전 작업

- 남녀 분리
- 연도와 이름의 피벗테이블
  - 열이 이름
    - 6868 개의 이름이 열

```
In [80]: boys = top1000[top1000.sex == 'M']
          boys[:5]
          girls = top1000[top1000.sex == 'F']
          girls[:5]
```

Out[80]:

	name	sex	births	year	prop
0	Mary	F	7065	1880	0.077643
1	Anna	F	2604	1880	0.028618
2	Emma	F	2003	1880	0.022013
3	Elizabeth	F	1939	1880	0.021309
4	Minnie	F	1746	1880	0.019188

```
In [81]: total_births = top1000.pivot_table('births', index='year',
                                             columns='name', aggfunc=sum)
          total_births.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 131 entries, 1880 to 2010
Columns: 6868 entries, Aaden to Zuri
dtypes: float64(6868)
memory usage: 6.9 MB
```

```
In [82]: total_births[:5]
```

Out[82]:

	name	Aaden	Aaliyah	Aarav	Aaron	Aarush	Ab	Abagail	Abb	Abbey	Abbie	...	Zoa	Zoe	Zoey	Zoie	Zola	Zollie	Z
year																			
1880		NaN	NaN	NaN	102.0	NaN	NaN	NaN	NaN	NaN	71.0	...	8.0	23.0	NaN	NaN	7.0	NaN	
1881		NaN	NaN	NaN	94.0	NaN	NaN	NaN	NaN	NaN	81.0	...	NaN	22.0	NaN	NaN	10.0	NaN	
1882		NaN	NaN	NaN	85.0	NaN	NaN	NaN	NaN	NaN	80.0	...	8.0	25.0	NaN	NaN	9.0	NaN	
1883		NaN	NaN	NaN	105.0	NaN	NaN	NaN	NaN	NaN	79.0	...	NaN	23.0	NaN	NaN	10.0	NaN	
1884		NaN	NaN	NaN	97.0	NaN	NaN	NaN	NaN	NaN	98.0	...	13.0	31.0	NaN	NaN	14.0	6.0	

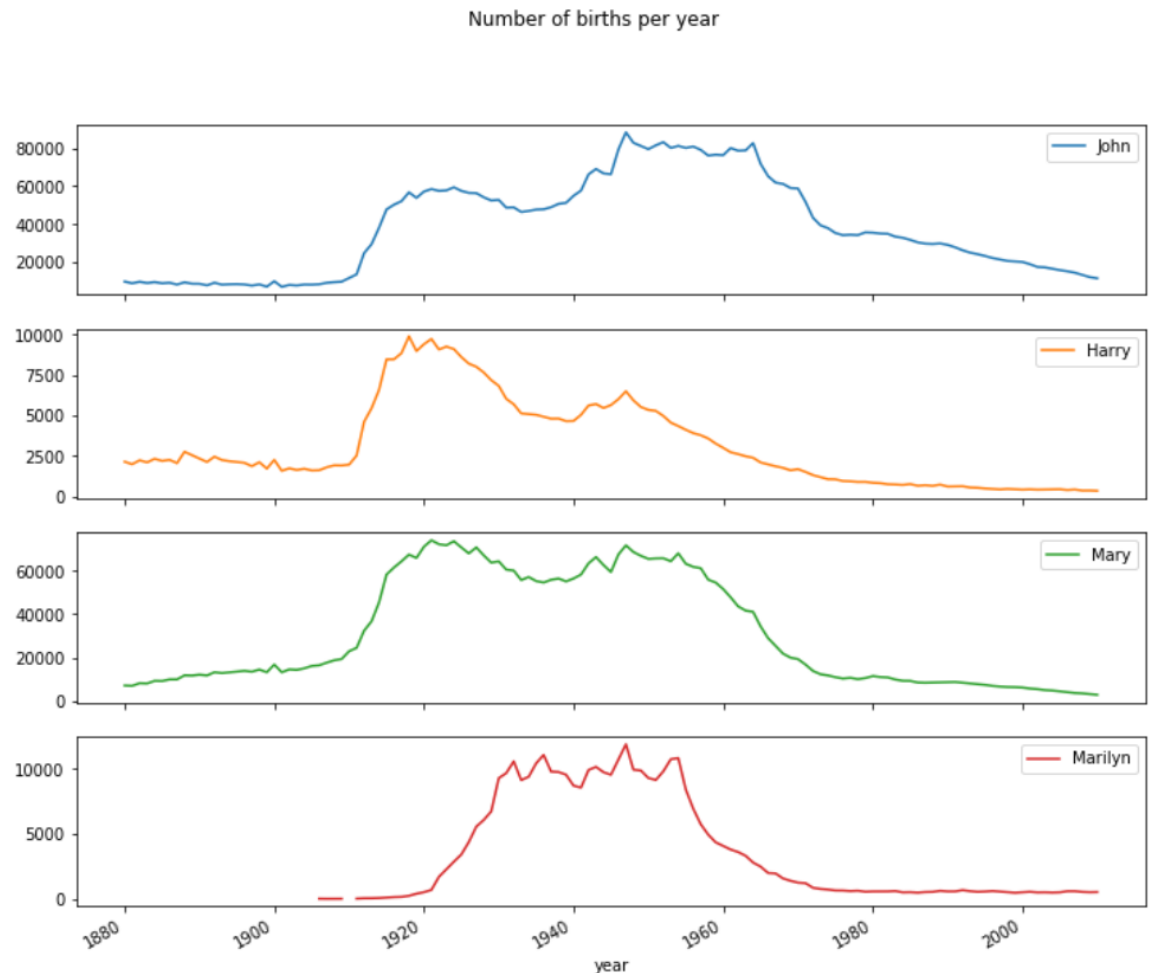
5 rows × 6868 columns

# 주요 이름

- **년도별 출생아 수**
  - John 등 4개

```
In [83]: subset = total_births[['John', 'Harry', 'Mary', 'Marilyn']]
subset.plot(subplots=True, figsize=(12, 10), grid=False,
           title="Number of births per year")
```

```
Out[83]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x000001E0E384CE88>,
<matplotlib.axes._subplots.AxesSubplot object at 0x000001E0EE041088>,
<matplotlib.axes._subplots.AxesSubplot object at 0x000001E0E384CB08>,
<matplotlib.axes._subplots.AxesSubplot object at 0x000001E0EE2263C8>],
dtype=object)
```



# 인기 이름 비율

- 전체 출생아 수에서 차지하는 비율
  - 인기있는 상위 1000개의 이름
- 의미
  - 흔한 이름을 기피
  - 다양성이 증가

```
In [88]: table = top1000.pivot_table('prop', index='year',  
                                     columns='sex', aggfunc=sum)  
table[:5]
```

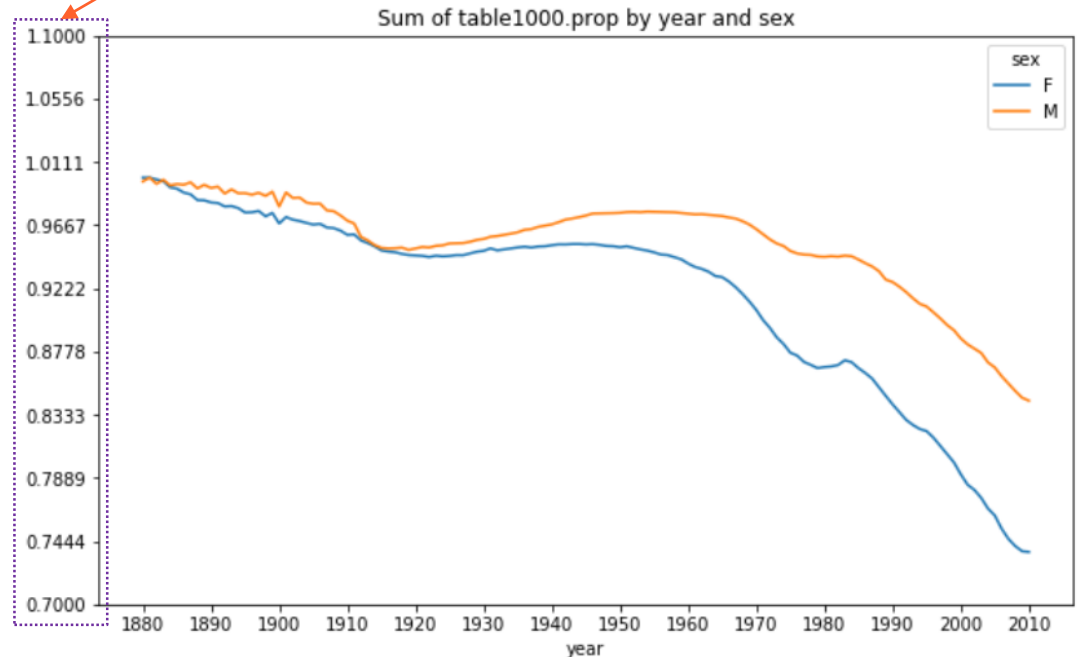
```
Out[88]:
```

sex	F	M
year		
1880	1.000000	0.997375
1881	1.000000	1.000000
1882	0.998702	0.995646
1883	0.997596	0.998566
1884	0.993156	0.994539

```
In [93]: plt.figure()  
table.plot(title='Sum of table1000.prop by year and sex',  
           yticks=np.linspace(0.7, 1.1, 10), xticks=range(1880, 2020, 10))
```

```
Out[93]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0e3368e88>
```

```
<Figure size 720x432 with 0 Axes>
```



# 두 연도에서 50%까지 이름의 종류

- 열 `prop`의 누계를 `prop_cumsum`에 저장
  - 상위 누계 50%에 몇 개의 이름이 있는 지?
    - 메소드 `searchsorted(5)`
      - .5에 이르는 첨자를 반환
- 결과
  - 1900: 25개
  - 2010: 117개

```
In [106]: prop_cumsum = df.sort_values(by='prop', ascending=False).prop.cumsum()
prop_cumsum[:5]
```

```
Out[106]: 40877    0.065319
40878    0.122308
40879    0.170437
40880    0.206338
40881    0.233584
Name: prop, dtype: float64
```

```
In [104]: prop_cumsum.values[searchsorted(0.5)] + 1
```

```
Out[104]: 117
```

```
In [105]: df = boys[boys.year == 1900]
in1900 = df.sort_values(by='prop', ascending=False).prop.cumsum()
in1900.values.searchsorted(0.5) + 1
```

```
Out[105]: 25
```

```
In [100]: df = boys[boys.year == 2010]
df
```

```
Out[100]:
```

	name	sex	births	year	prop
260877	Jacob	M	21875	2010	0.011523
260878	Ethan	M	17866	2010	0.009411
260879	Michael	M	17133	2010	0.009025
260880	Jayden	M	17030	2010	0.008971
260881	William	M	16870	2010	0.008887
...	...	...	...	...	...
261872	Camilo	M	194	2010	0.000102
261873	Destin	M	194	2010	0.000102
261874	Jaquan	M	194	2010	0.000102
261875	Jaydan	M	194	2010	0.000102
261876	Maxton	M	193	2010	0.000102

1000 rows × 5 columns

```
In [101]: df.sort_values(by='prop', ascending=False)
df.sort_values(by='prop', ascending=False).prop.cumsum()
```

```
Out[101]: 260877    0.011523
260878    0.020934
260879    0.029959
260880    0.038930
260881    0.047817
...
261872    0.842748
261873    0.842850
261874    0.842953
261875    0.843055
261876    0.843156
Name: prop, Length: 1000, dtype: float64
```

# 모든 연도에서 50%까지 이름의 종류

## • DataFrame diversity

- 연도와 성을 그룹핑해 묶고 각 그룹에 apply() 적용

## • 이름이 다양해지고 있으며, 특히 여자가 뚜렷

```
In [111]: def get_quantile_count(group, q=0.5):
            group = group.sort_values(by='prop', ascending=False)
            return group.prop.cumsum().values.searchsorted(q) + 1

diversity = top1000.groupby(['year', 'sex']).apply(get_quantile_count)
diversity[:6]
```

```
Out[111]: year sex
          1880 F      38
           1880 M      14
          1881 F      38
           1881 M      14
          1882 F      38
           1882 M      15
dtype: int64
```

```
In [112]: diversity = diversity.unstack('sex')
diversity.head()
```

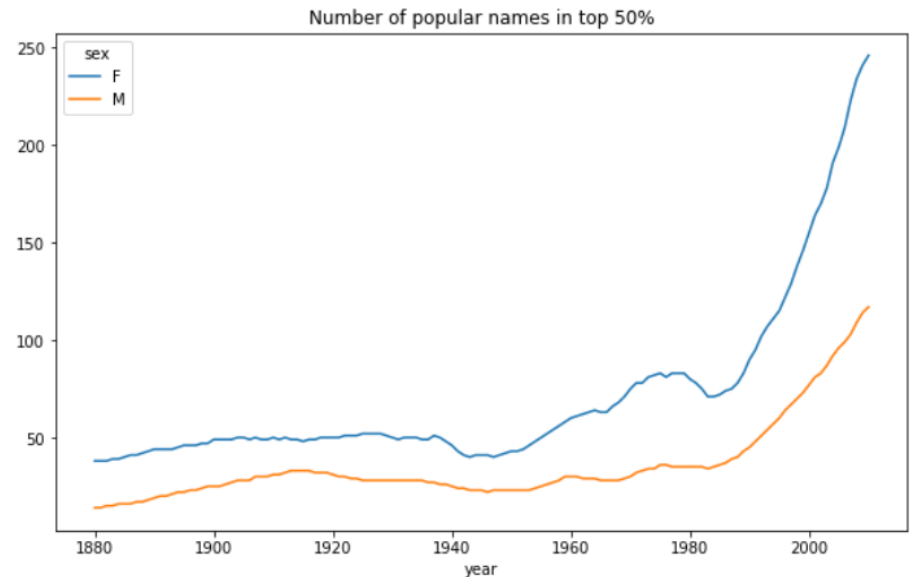
```
Out[112]:
```

sex	F	M
year		
1880	38	14
1881	38	14
1882	38	15
1883	39	15
1884	39	16

```
In [113]: fig = plt.figure()
diversity.plot(title="Number of popular names in top 50%")

Out[113]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0f0ed8f48>

<Figure size 720x432 with 0 Axes>
```



# 마지막 글자의 변화

- 로라 와튼버그 분석
  - 이름 연구가
    - 남아 이름의 마지막 철자의 변화
- DF subtable
  - 1910, 1960, 2010

```
In [114]: # extract last letter from name column
get_last_letter = lambda x: x[-1]
last_letters = names.name.map(get_last_letter)
last_letters.name = 'last_letter'
last_letters.head()
```

```
Out[114]: 0    y
          1    a
          2    a
          3    h
          4    e
          Name: last_letter, dtype: object
```

```
In [115]: table = names.pivot_table('births', index=last_letters,
                                   columns=['sex', 'year'], aggfunc=sum)
table.head()
```

```
Out[115]:
```

sex	F									
year	1880	1881	1882	1883	1884	1885	1886	1887	1888	1889
last_letter										
a	31446.0	31581.0	36536.0	38330.0	43680.0	45408.0	49100.0	48942.0	59442.0	58631.0
b	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
c	NaN	NaN	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN
d	609.0	607.0	734.0	810.0	916.0	862.0	1007.0	1027.0	1298.0	1374.0
e	33378.0	34080.0	40399.0	41914.0	48089.0	49616.0	53884.0	54353.0	66750.0	66663.0

5 rows × 262 columns

```
In [116]: subtable = table.reindex(columns=[1910, 1960, 2010], level='year')
subtable.head()
```

```
Out[116]:
```

sex	F			M		
year	1910	1960	2010	1910	1960	2010
last_letter						
a	108376.0	691247.0	670605.0	977.0	5204.0	28438.0
b	NaN	694.0	450.0	411.0	3912.0	38859.0
c	5.0	49.0	946.0	482.0	15476.0	23125.0
d	6750.0	3729.0	2607.0	22111.0	262112.0	44398.0
e	133569.0	435013.0	313833.0	28655.0	178823.0	129012.0

# 마지막 철자가 차지하는 비율

- 출생아 수 구하고
- 각각의 철자에서 출생아 수로 나눔

```
In [117]: subtable.sum()
```

```
Out[117]: sex  year
          F    1910    396416.0
          1960    2022062.0
          2010    1759010.0
          M    1910    194198.0
          1960    2132588.0
          2010    1898382.0
dtype: float64
```

```
In [119]: letter_prop = subtable / subtable.sum()
          letter_prop.head()
```

```
Out[119]:
```

	sex	F			M		
	year	1910	1960	2010	1910	1960	2010
	last_letter						
a		0.273390	0.341853	0.381240	0.005031	0.002440	0.014980
b		NaN	0.000343	0.000256	0.002116	0.001834	0.020470
c		0.000013	0.000024	0.000538	0.002482	0.007257	0.012181
d		0.017028	0.001844	0.001482	0.113858	0.122908	0.023387
e		0.336941	0.215133	0.178415	0.147556	0.083853	0.067959



# 3개 년도 마지막 철자가 차지하는 비율 그리기

## • 남자

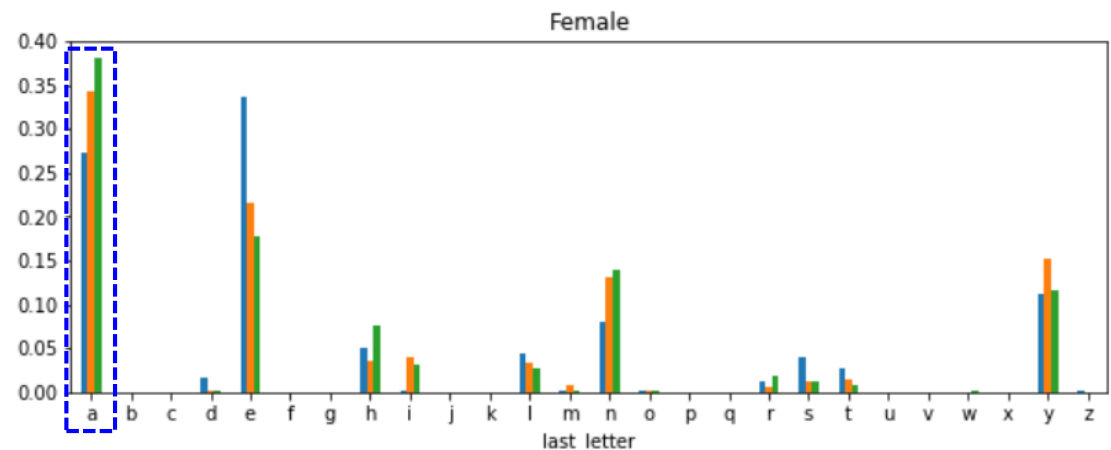
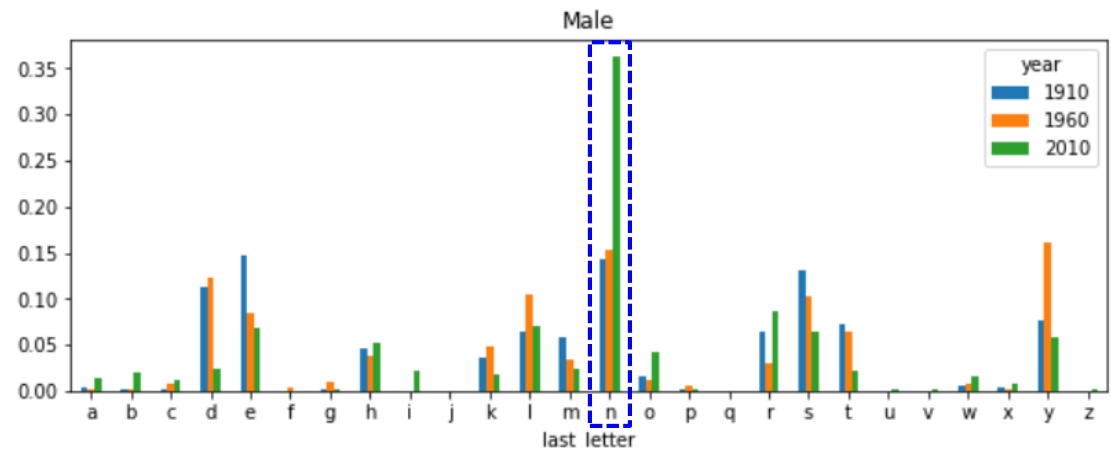
– 철자 n의 빈도

- 1960년 이후에 급격히 증가

In [125]: `import matplotlib.pyplot as plt`

```
fig, axes = plt.subplots(2, 1, figsize=(10, 8))
letter_prop['M'].plot(kind='bar', rot=0, ax=axes[0], title='Male')
letter_prop['F'].plot(kind='bar', rot=0, ax=axes[1], title='Female',
                      legend=False)
```

```
plt.subplots_adjust(hspace=0.30)
```



# 남자의 마지막 철자가 차지하는 비율 그리기

- 남자 아이
  - 철자 d, n, y의 변화

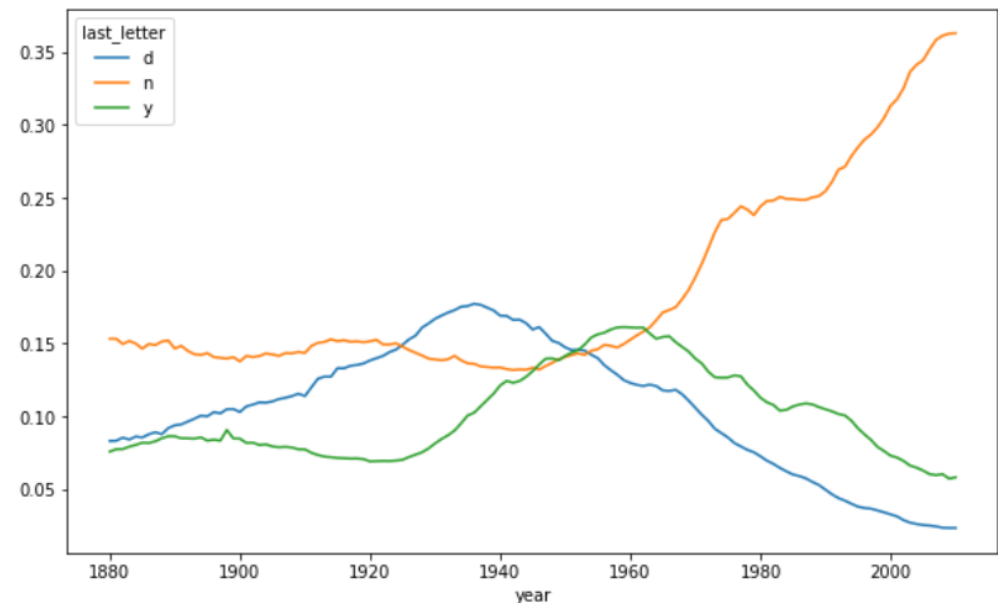
```
In [126]: letter_prop = table / table.sum()
dny_ts = letter_prop.loc[['d', 'n', 'y']].T
dny_ts.head()
```

```
Out[126]:
```

	last_letter	d	n	y
year				
1880		0.083055	0.153213	0.075760
1881		0.083247	0.153214	0.077451
1882		0.085340	0.149560	0.077537
1883		0.084066	0.151646	0.079144
1884		0.086120	0.149915	0.080405

```
In [127]: dny_ts.plot()
```

```
Out[127]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0dea38c8>
```



# 여자의 마지막 철자가 차지하는 비율 그리기

## • 여자 아이

- 철자 a, e, n, y의 변화

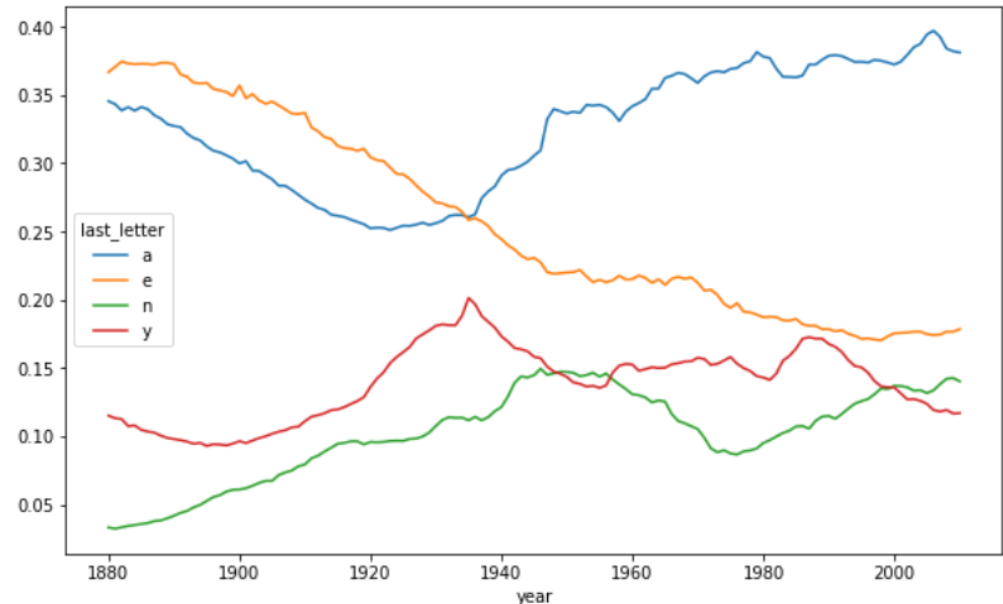
```
In [131]: letter_prop = table / table.sum()
aeny_ts = letter_prop.loc[['a', 'e', 'n', 'y'], 'F'].T
aeny_ts.head()
```

Out[131]:

last_letter	a	e	n	y
year				
1880	0.345587	0.366819	0.033057	0.115053
1881	0.343440	0.370616	0.032179	0.113142
1882	0.338764	0.374582	0.033157	0.112609
1883	0.341251	0.373159	0.034161	0.107397
1884	0.338550	0.372722	0.034932	0.107866

```
In [132]: aeny_ts.plot()
```

Out[132]: <matplotlib.axes.\_subplots.AxesSubplot at 0x1e0ebfa8148>



# 성별로 철자 **lesl**로 시작하는 이름의 수

- 시리즈 **lesley\_like**
  - 철자 **lesl**로 시작하는 이름을 포함하는 목록
- **lesley\_like**에서 각 이름의 출생아 수

```
In [133]: all_names = pd.Series(top1000.name.unique())
lesley_like = all_names[all_names.str.lower().str.contains('lesl')]
lesley_like
```

```
Out[133]: 632      Leslie
2294      Lesley
4262      Leslee
4728      Lesli
6103      Lesly
dtype: object
```

```
In [134]: filtered = top1000[top1000.name.isin(lesley_like)]
filtered.groupby('name').births.sum()
```

```
Out[134]: name
Leslee      1082
Lesley     35022
Lesli        929
Leslie     370429
Lesly       10067
Name: births, dtype: int64
```

```
In [135]: table = filtered.pivot_table('births', index='year',
                                         columns='sex', aggfunc='sum')
table.head()
```

```
Out[135]:
```

	sex	F	M
	year		
	1880	8.0	79.0
	1881	11.0	92.0
	1882	9.0	128.0
	1883	7.0	125.0
	1884	15.0	125.0

# 이름 **lesley**와 비슷한 이름의 남녀 비율의 변화

## • 출생 연도로 정규화

- `table =`  
`table.div(table.sum(1),`  
`axis=0)`

## • 의미

- 이름 Lesley, Leslie
  - 남자 이름에서 여자 이름으로 바뀐 이름

```
In [153]: table = table.div(table.sum(1), axis=0)
          table.tail()
```

```
Out[153]:
```

sex	F	M
year		
2006	1.0	NaN
2007	1.0	NaN
2008	1.0	NaN
2009	1.0	NaN
2010	1.0	NaN

```
In [154]: table.plot(style={'M': 'b-', 'F': 'r--'})
```

```
Out[154]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0ebf99fc8>
```

