Rajeev_download_process_files

February 10, 2018

0.1 Download data and create pandas dataframes

- A .py version of this script is %run in other notebooks to load primary dataframes
- Last cell contains descriptions of dataframe schemas
- Script is smart enough not to download or perform lengthy procedures if the files already exist

Set working path and import libraries

In [1]:
```
data_path = "rajeev_data" names

        import os
        if not os.path.isdir(data_path): # creates path if it does not exist
            os.makedirs(data_path)

        import pandas as pd
        import urllib.request
        import zipfile
```

Download files from US Social Security Administration website unless files already exist in working_path

In [ ]:
```
os.chdir(data_path)

        ssa_url = 'http://www.socialsecurity.gov/oact/babynames/names.zip'

        if not os.path.isfile("names.zip"):
            #print "Downloading."
            import urllib
            urllib.request.urlretrieve(ssa_url, 'names.zip')
        else: print("Data already downloaded.")

        if not os.path.isfile("yob1880.txt") or not os.path.isfile("yob2017.txt"):
            #print "Extracting."
            import zipfile
            with zipfile.ZipFile('names.zip') as zf:
                for member in zf.infolist():
                    zf.extract(member)
```

```
        else: print("Data already extracted.")

        os.chdir("../")
```

Create pandas dataframes from U.S. Social Security baby names database and pickle for later use in other notebooks This block takes well under a minute on my medium-quality desktop Windows PC.

```
In [3]: redo_dataframes = False
        os.chdir(data_path)

        if (redo_dataframes == True or
            not os.path.isfile("yob.pickle") or
            not os.path.isfile("names.pickle") or
            not os.path.isfile("years.pickle")):

            print("Processing.")

            # read individual files, yob1880.txt, yob1881.txt, etc. and assemble into a datafr
            years = range(1880, 2017) # stops at 2017: update this when Social Security Admini
            parts = []
            part_columns = ['name', 'sex', 'births']
            for year in years:
                path = 'yob' + str(year) + '.txt'
                part_df = pd.read_csv(path, names=part_columns)
                part_df['year'] = year
                parts.append(part_df)
            yob = pd.concat(parts, ignore_index=True)

            # add column 'pct': the number of births of that name and sex in that year
            # divided by the total number of births of that sex in that year, multiplied by
            # 100 to turn into a percentage and reduce leading zeroes
            def add_pct(group):
                births = group.births.astype(float)
                group['pct'] = (births / births.sum() * 100)
                return group
            yob = yob.groupby(['year', 'sex']).apply(add_pct)
            #add rank of each name each year each sex
            yob['ranked'] = yob.groupby(['year', 'sex'])['births'].rank(ascending=False)
            yob.to_pickle("yob.pickle")

            # names dataframe: discards individual birth or pct values, and instead collects d
            # There is one row per unique combination of name and sex.
            yobf = yob[yob.sex == 'F']
            yobm = yob[yob.sex == 'M']
            names_count = pd.DataFrame(yobf['name'].value_counts())
            names_count.columns= ['year_count']
            names_min = pd.DataFrame(yobf.groupby('name').year.min())
```

```
names_min.columns = ['year_min']
names_max = pd.DataFrame(yobf.groupby('name').year.max())
names_max.columns = ['year_max']
names_pctsum = pd.DataFrame(yobf.groupby('name').pct.sum())
names_pctsum.columns = ['pct_sum']
names_pctmax = pd.DataFrame(yobf.groupby('name').pct.max())
names_pctmax.columns = ['pct_max']
names_f = names_count.join(names_min)
names_f = names_f.join(names_max)
names_f = names_f.join(names_pctsum)
names_f = names_f.join(names_pctmax)
names_f['sex'] = "F"
names_f.reset_index(inplace=True, drop=False)
names_f.columns = ['name', 'year_count', 'year_min', 'year_max', 'pct_sum', 'pct_ma
names_f = names_f[['name', 'sex', 'year_count', 'year_min', 'year_max', 'pct_sum',
names_count = pd.DataFrame(yobm['name'].value_counts())
names_count.columns=['year_count']
names_min = pd.DataFrame(yobm.groupby('name').year.min())
names_min.columns = ['year_min']
names_max = pd.DataFrame(yobm.groupby('name').year.max())
names_max.columns = ['year_max']
names_pctsum = pd.DataFrame(yobm.groupby('name').pct.sum())
names_pctsum.columns = ['pct_sum']
names_pctmax = pd.DataFrame(yobm.groupby('name').pct.max())
names_pctmax.columns = ['pct_max']
names_m = names_count.join(names_min)
names_m = names_m.join(names_max)
names_m = names_m.join(names_pctsum)
names_m = names_m.join(names_pctmax)
names_m['sex'] = "M"
names_m.reset_index(inplace=True, drop=False)
names_m.columns = ['name', 'year_count', 'year_min', 'year_max', 'pct_sum', 'pct_ma
names_m = names_m[['name', 'sex', 'year_count', 'year_min', 'year_max', 'pct_sum',
names = pd.concat([names_f, names_m], ignore_index=True)
names.to_pickle('names.pickle')

# create years dataframe. Discards individual name data, aggregating by year.
total = pd.DataFrame(yob.pivot_table('births', index='year', columns = 'sex', aggfu
total.reset_index(drop=False, inplace=True)
total.columns = ['year', 'births_f', 'births_m']
total['births_t'] = total.births_f + total.births_m
newnames = pd.DataFrame(names.groupby('year_min').year_min.count())
newnames.columns = ['firstyearcount']
newnames.reset_index(drop=False, inplace=True)
newnames.columns = ['year', 'new_names']
uniquenames = pd.DataFrame()
for yr in range(1880, 2017):
    uniquenames = uniquenames.append(pd.DataFrame([{'year':yr, 'unique_names':len(y
```

```
            years = pd.merge(left=total, right=newnames, on='year', right_index=False, left_inc
            years = pd.merge(left=years, right=uniquenames, on='year', right_index=False, left_
            years['sexratio'] = 100.0 * years.births_m / years.births_f
            years.to_pickle('years.pickle')

        else:

            print("Reading from pickle.")
            yob = pd.read_pickle('yob.pickle')
            names = pd.read_pickle('names.pickle')
            years = pd.read_pickle('years.pickle')

        os.chdir("../")

Processing.
```

1 Dataframe schemas:

Note dataframes have only an arbitrary ordinal index. Indexes and multi-indexes are added later where needed.

---

yob = a dataframe with each record comprising a unique name, sex and year.

```
name     String
sex      M or F
births   Number of birth with that name of that sex during that year;
         names with fewer than 5 births in a given year are omitted due to privacy concerns
year     1880-2012
pct      Percentage of births of that sex during that year with that name (float)
ranked   Rank of number of births of that name among all births of that sex during that year
```

---

names = a dataframe with each record comprising a unique name and sex, with data for individual years discarded but summary and additional data added.

```
name             Same as in df
sex              Same as in df
year_count       Number of different years in which that name appears in dataframe, from 1 to
year_min         First year name appears in database
year_max         Last year name appears in database
pct_sum          Sum of pct field for that name for all years. Not a statistically meaningful
                 (because the underlying distribution of names varies from year to year),
                 but I have found it a useful rough metric during development
pct_max          Maximum value in pct field for all years, indicating the most popular that na
```

years = a dataframe with each record comprising a unique year, with individual name data discarded but summary and additional data added.

```
year              Same as in df
births_f          Number of female births during that year
births_m          Number of male births during that year
births_t          Total number of births during that year
new_names         Number of names that appear for the first time during that year
unique_names      Number of different names that appear during that year
sexratio          Number of boys born per hundred girls
```

Make versions from 1900 on:

```
In [4]: os.chdir(data_path)

        if (redo_dataframes == True or
            not os.path.isfile("yob1900.pickle") or
            not os.path.isfile("names1900.pickle") or
            not os.path.isfile("years1900.pickle")):

            yob1900 = yob[yob.year >= 1900]
            yob1900.to_pickle("yob1900.pickle")

            yobf = yob1900[yob1900.sex == 'F']
            yobm = yob1900[yob1900.sex == 'M']
            names_count = pd.DataFrame(yobf['name'].value_counts())
            names_count.columns= ['year_count']
            names_min = pd.DataFrame(yobf.groupby('name').year.min())
            names_min.columns = ['year_min']
            names_max = pd.DataFrame(yobf.groupby('name').year.max())
            names_max.columns = ['year_max']
            names_pctsum = pd.DataFrame(yobf.groupby('name').pct.sum())
            names_pctsum.columns = ['pct_sum']
            names_pctmax = pd.DataFrame(yobf.groupby('name').pct.max())
            names_pctmax.columns = ['pct_max']
            names_f = names_count.join(names_min)
            names_f = names_f.join(names_max)
            names_f = names_f.join(names_pctsum)
            names_f = names_f.join(names_pctmax)
            names_f['sex'] = "F"
            names_f.reset_index(inplace=True, drop=False)
            names_f.columns = ['name', 'year_count', 'year_min', 'year_max', 'pct_sum', 'pct_ma
            names_f = names_f[['name', 'sex', 'year_count', 'year_min', 'year_max', 'pct_sum',
            names_count = pd.DataFrame(yobm['name'].value_counts())
            names_count.columns=['year_count']
            names_min = pd.DataFrame(yobm.groupby('name').year.min())
```

```
        names_min.columns = ['year_min']
        names_max = pd.DataFrame(yobm.groupby('name').year.max())
        names_max.columns = ['year_max']
        names_pctsum = pd.DataFrame(yobm.groupby('name').pct.sum())
        names_pctsum.columns = ['pct_sum']
        names_pctmax = pd.DataFrame(yobm.groupby('name').pct.max())
        names_pctmax.columns = ['pct_max']
        names_m = names_count.join(names_min)
        names_m = names_m.join(names_max)
        names_m = names_m.join(names_pctsum)
        names_m = names_m.join(names_pctmax)
        names_m['sex'] = "M"
        names_m.reset_index(inplace=True, drop=False)
        names_m.columns = ['name', 'year_count', 'year_min', 'year_max', 'pct_sum', 'pct_ma
        names_m = names_m[['name', 'sex', 'year_count', 'year_min', 'year_max', 'pct_sum',
        names1900 = pd.concat([names_f, names_m], ignore_index=True)
        names1900.to_pickle('names1900.pickle')

        # create years dataframe. Discards individual name data, aggregating by year.
        total = pd.DataFrame(yob1900.pivot_table('births', index='year', columns = 'sex', a
        total.reset_index(drop=False, inplace=True)
        total.columns = ['year', 'births_f', 'births_m']
        total['births_t'] = total.births_f + total.births_m
        newnames = pd.DataFrame(names.groupby('year_min').year_min.count())
        newnames.columns = ['firstyearcount']
        newnames.reset_index(drop=False, inplace=True)
        newnames.columns = ['year', 'new_names']
        uniquenames = pd.DataFrame()
        for yr in range(1900, 2017):
            uniquenames = uniquenames.append(pd.DataFrame([{'year':yr, 'unique_names':len(y
        years1900 = pd.merge(left=total, right=newnames, on='year', right_index=False, left
        years1900 = pd.merge(left=years, right=uniquenames, on='year', right_index=False, l
        years1900['sexratio'] = 100.0 * years.births_m / years.births_f
        years1900.to_pickle('years.pickle')

    else:

        print("Reading from pickle (1900+ versions).")
        yob1900 = pd.read_pickle('yob1900.pickle')
        names1900 = pd.read_pickle('names1900.pickle')
        years1900 = pd.read_pickle('years1900.pickle')

    os.chdir("../")
```

1.0.1 Tails of all three dataframes:

```
In [5]: print("Tail of dataframe 'yob':")
        yob.tail()
```

```
Tail of dataframe 'yob':
```

Out[5]:

	name	sex	births	year	pct	ranked
1891889	Zolton	M	5	2016	0.000266	13106.5
1891890	Zurich	M	5	2016	0.000266	13106.5
1891891	Zyahir	M	5	2016	0.000266	13106.5
1891892	Zyel	M	5	2016	0.000266	13106.5
1891893	Zylyn	M	5	2016	0.000266	13106.5

```
In [6]: print("\nTail of dataframe 'names':")
        names.tail()
```

```
Tail of dataframe 'names':
```

Out[6]:

	name	sex	year_count	year_min	year_max	pct_sum	pct_max
106690	Jovari	M	1	2012	2012	0.000264	0.000264
106691	Porsha	M	1	1989	1989	0.000300	0.000300
106692	Ebbin	M	1	1921	1921	0.000454	0.000454
106693	Dannen	M	1	1968	1968	0.000288	0.000288
106694	Maryon	M	1	1924	1924	0.000441	0.000441

```
In [7]: print("\nTail of dataframe 'years':")
        years.tail()
```

```
Tail of dataframe 'years':
```

Out[7]:

	year	births_f	births_m	births_t	new_names	unique_names	sexratio
132	2012	1756347	1892094	3648441	1536	31266	107.728940
133	2013	1749061	1885683	3634744	1418	30819	107.811163
134	2014	1779496	1913434	3692930	1400	30709	107.526738
135	2015	1776538	1907211	3683749	1258	30553	107.355486
136	2016	1756647	1880674	3637321	1264	30294	107.060440

```
In [8]: print("Tail of dataframe 'yob1900':")
        yob1900.tail()
```

```
Tail of dataframe 'yob1900':
```

Out[8]:

	name	sex	births	year	pct	ranked
1891889	Zolton	M	5	2016	0.000266	13106.5
1891890	Zurich	M	5	2016	0.000266	13106.5
1891891	Zyahir	M	5	2016	0.000266	13106.5
1891892	Zyel	M	5	2016	0.000266	13106.5
1891893	Zylyn	M	5	2016	0.000266	13106.5

```
In [9]: print("Tail of dataframe 'names1900':")
        names1900.tail()
```

```
Tail of dataframe 'names1900':
```

Out[9]:

	name	sex	year_count	year_min	year_max	pct_sum	pct_max
106633	Braecyn	M	1	2016	2016	0.000266	0.000266
106634	Toki	M	1	2015	2015	0.000262	0.000262
106635	Erny	M	1	1995	1995	0.000315	0.000315
106636	Jaymi	M	1	1993	1993	0.000255	0.000255
106637	Greenwood	M	1	1918	1918	0.000592	0.000592

```
In [10]: years1900 = years1900[['year', 'births_f', 'births_m', 'births_t', 'new_names', 'uniqu
         years1900.columns = ['year', 'births_f', 'births_m', 'births_t', 'new_names', 'unique_
         # above lines correct outer merge problem
         print("Tail of dataframe 'years1900':")
         years1900.tail()
```

```
Tail of dataframe 'years1900':
```

Out[10]:

	year	births_f	births_m	births_t	new_names	unique_names	sexratio
112	2012	1756347	1892094	3648441	1536	31266	108.306454
113	2013	1749061	1885683	3634744	1418	30819	108.403568
114	2014	1779496	1913434	3692930	1400	30709	108.185819
115	2015	1776538	1907211	3683749	1258	30553	108.247621
116	2016	1756647	1880674	3637321	1264	30294	108.057782