6.1) The Data Science Lifecycle

Vitor Kamada

December 2019

Reference

Tables, Graphics, and Figures from

Principles and Techniques of Data Science

Lau et al. (2019): Ch 1 The Data Science Lifecycle

https://www.textbook.ds100.org/ch/01/lifecycle_intro.html

Data Science Lifecycle

1. **Question/Problem Formulation**
   - Student first names give us additional information about the students themselves?
2. **Data Acquisition and Cleaning**
3. **Exploratory Data Analysis**
4. **Prediction and Inference**

UC Berkeley: Student First Names in Data 100

```
import numpy as np
import pandas as pd
path ='https://github.com/DS-100/textbook/raw/master/content/'
students = pd.read_csv(path + 'ch/01/roster.csv')
students
```

```
students['Name'] = students['Name'].str.lower()
```

	Name	Role
0	Keeley	Student
1	John	Student
2	BRYAN	Student

	Name	Role
0	keeley	Student
1	john	Student
2	bryan	Student

What is the Meaning of the Role Field?

```
print("There are", len(students),
      "students on the roster.")
```

```
There are 279 students on the roster.
```

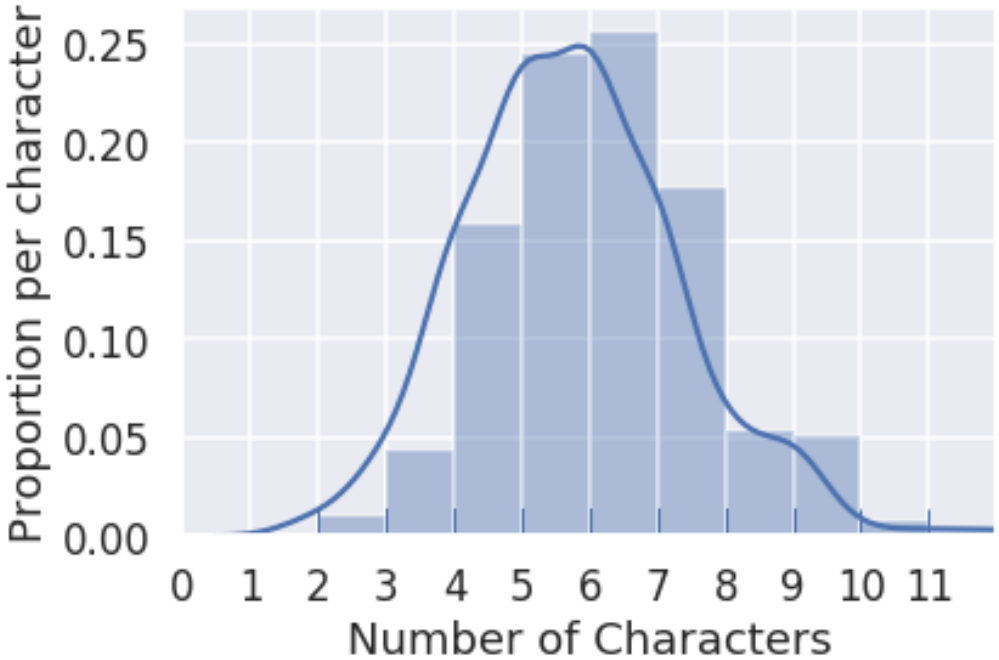
```
students['Role'].value_counts().to_frame()
```

	Role
Student	237
Waitlist Student	42

```
students['Name'][5]
```

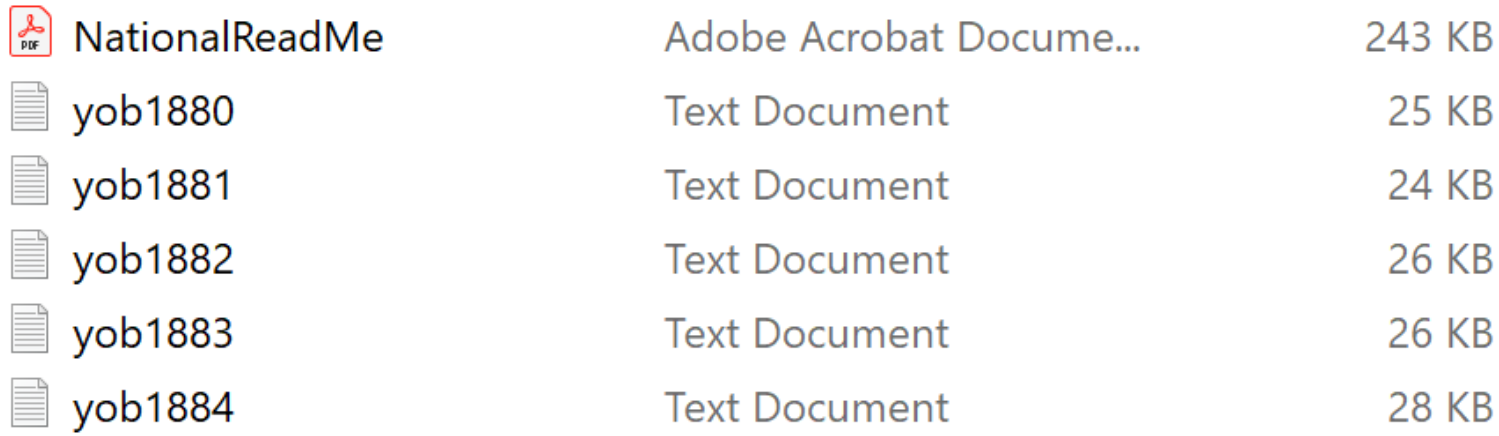
```
'jerry'
```

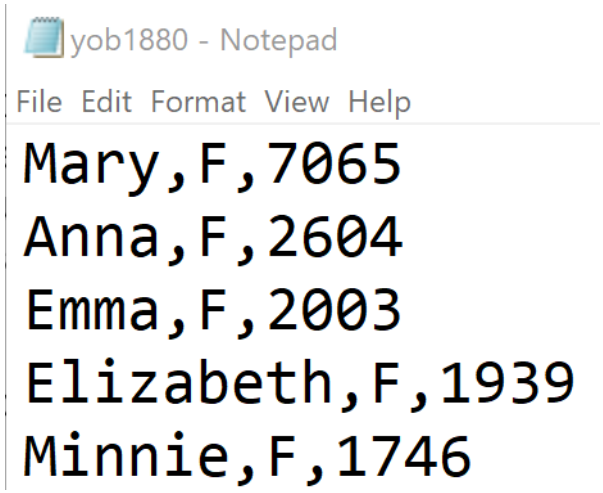
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
sns.set_context('talk')
sns.distplot(students['Name'].str.len(),  rug=True,
 bins=np.arange(12), axlabel="Number of Characters")
plt.xlim(0, 12)
plt.xticks(np.arange(12))
plt.ylabel('Proportion per character');
```



Baby Names Dataset from US Social Security

https://www.ssa.gov/oact/babynames/index.html





Downloading and Loading the Dataset

```
import urllib.request
import os.path
data_url = "https://www.ssa.gov/oact/babynames/names.zip"
local_filename = "babynames.zip"
if not os.path.exists(local_filename): # if the data exists don't download again
    with urllib.request.urlopen(data_url) as resp, open(local_filename, 'wb') as f:
        f.write(resp.read())

import zipfile
babynames = []
with zipfile.ZipFile(local_filename, "r") as zf:
    data_files = [f for f in zf.filelist if f.filename[-3:] == "txt"]
    def extract_year_from_filename(fn):
        return int(fn[3:7])
    for f in data_files:
        year = extract_year_from_filename(f.filename)
        with zf.open(f) as fp:
            df = pd.read_csv(fp, names=["Name", "Sex", "Count"])
            df["Year"] = year
            babynames.append(df)
```

```
babynames = pd.concat(babynames)
babynames
```

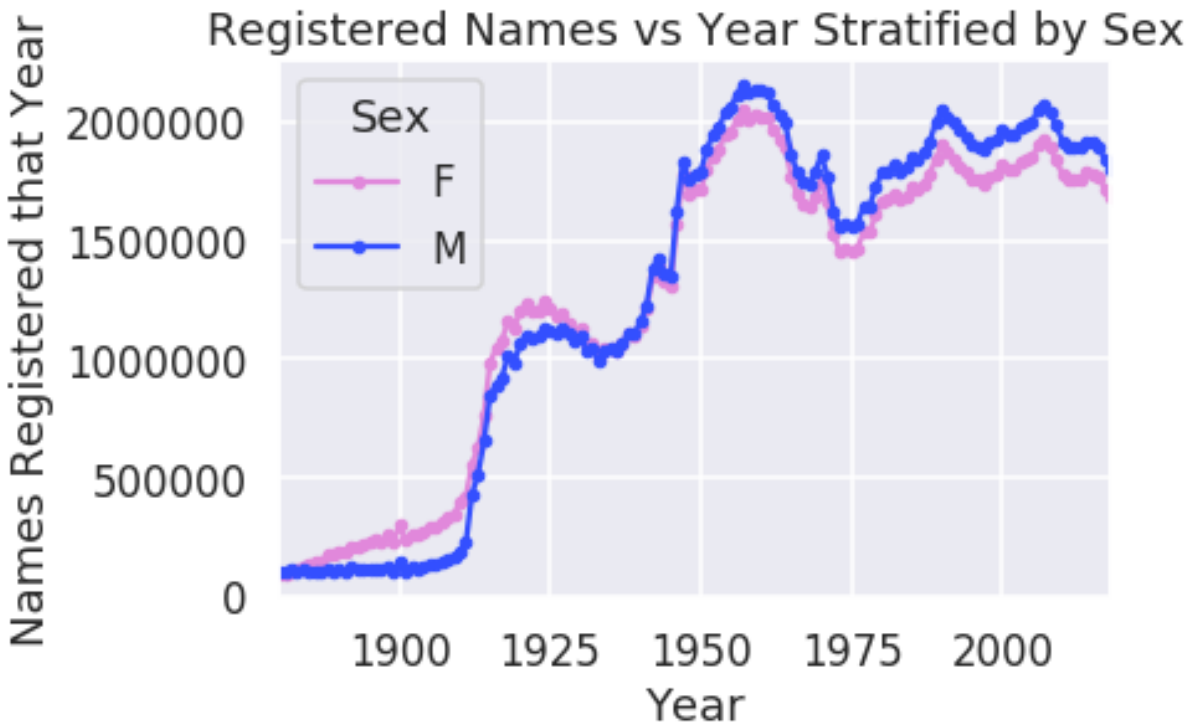
	Name	Sex	Count	Year
0	Mary	F	7065	1880
1	Anna	F	2604	1880
2	Emma	F	2003	1880

```
babynames['Name'] = babynames['Name'].str.lower()
```

	Name	Sex	Count	Year
0	mary	F	7065	1880
1	anna	F	2604	1880
2	emma	F	2003	1880

```
pivot_year_name_count = pd.pivot_table(
    babynames, index='Year', columns='Sex',
    values='Count', aggfunc=np.sum)

pink_blue = ["#E188DB", "#334FFF"]
with sns.color_palette(sns.color_palette(pink_blue)):
    pivot_year_name_count.plot(marker=".")
    plt.title("Registered Names vs Year Stratified by Sex")
    plt.ylabel('Names Registered that Year')
```



```
sex_counts = pd.pivot_table(babynames, index='Name',
  columns='Sex', values='Count', aggfunc='sum',
  fill_value=0., margins=True)
```

Sex	F	M	All
Name			
aaban	0	114	114
aabha	35	0	35
aabid	0	16	16

```
prop_female = sex_counts['F'] / sex_counts['All']
sex_counts['prop_female'] = prop_female
```

Sex	F	M	All	prop_female
Name				
aaban	0	114	114	0.000000
aabha	35	0	35	1.000000
aabid	0	16	16	0.000000

Inferring Sex From Name

```
def sex_from_name(name):
    if name in sex_counts.index:
        prop = sex_counts.loc[name, 'prop_female']
        return 'F' if prop > 0.5 else 'M'
    else:
        return 'Name not in dataset'

sex_from_name('sam')
```

```
'M'
```

```
students['sex'] = students['Name'].apply(sex_from_name)
```

	Name	Role	sex
0	keeley	Student	F
1	john	Student	M
2	bryan	Student	M

```
students['sex'].value_counts()
```

```
M                          145
F                           92
Name not in dataset         42
Name: sex, dtype: int64
```

```
import ipywidgets as widgets
from ipywidgets import interact
interact(sex_from_name, name='sam');
```

name | sam

'M'

Inferring Age From Name

```
def avg_year(group):
    return np.average(group['Year'], weights=group['Count'])

avg_years = (babynames.groupby('Name').apply(avg_year)
    .rename('avg_year').to_frame())
```

Name	avg_year
aaban	2013.333333
aabha	2013.714286
aabid	2012.687500

```
def year_from_name(name):
    return (avg_years.loc[name, 'avg_year']
            if name in avg_years.index
            else None)

interact(year_from_name, name='fernando');
```

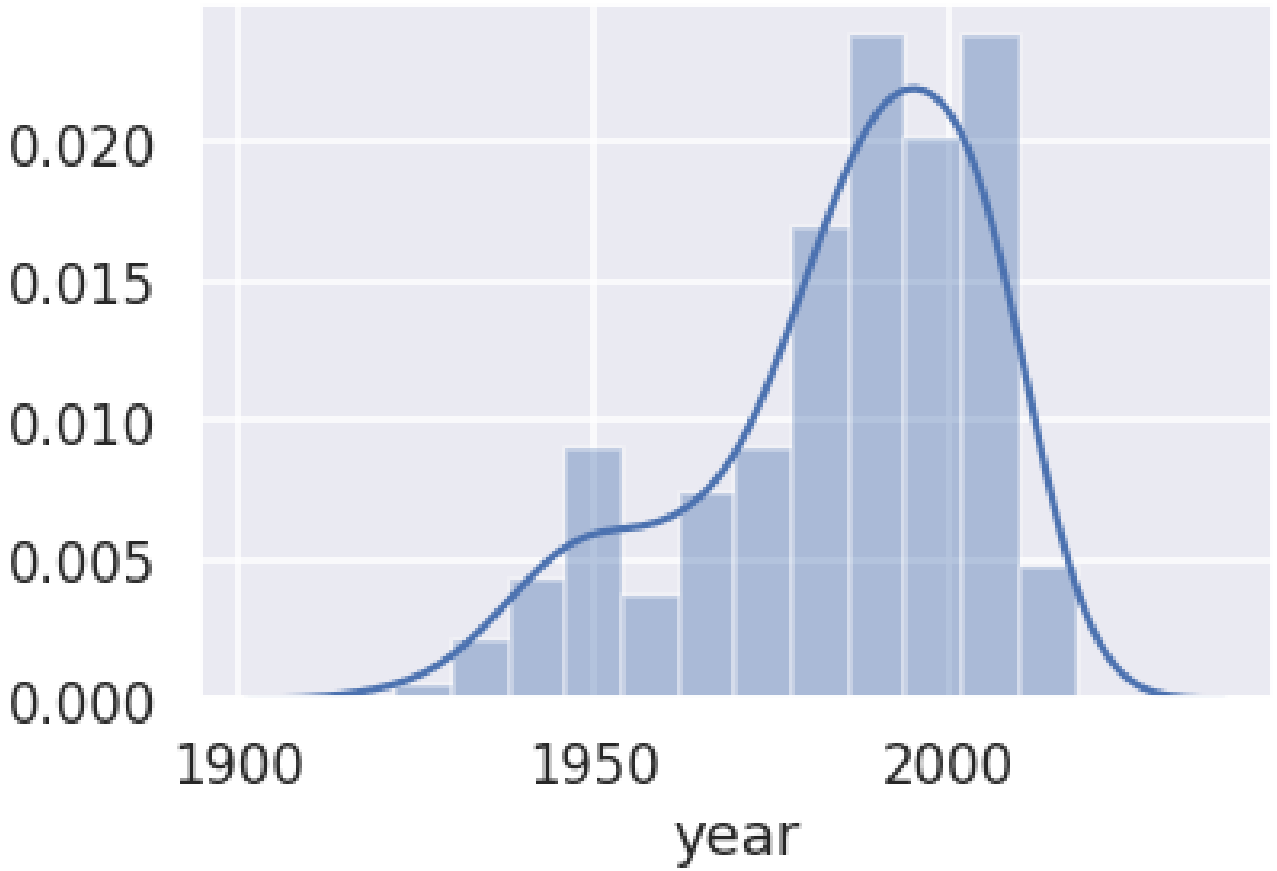
name fernando

1988.883829953841

```
students['year'] = students['Name'].apply(year_from_name)
```

	Name	Role	sex	year
0	keeley	Student	F	1998.668050
1	john	Student	M	1951.328445
2	bryan	Student	M	1983.875858

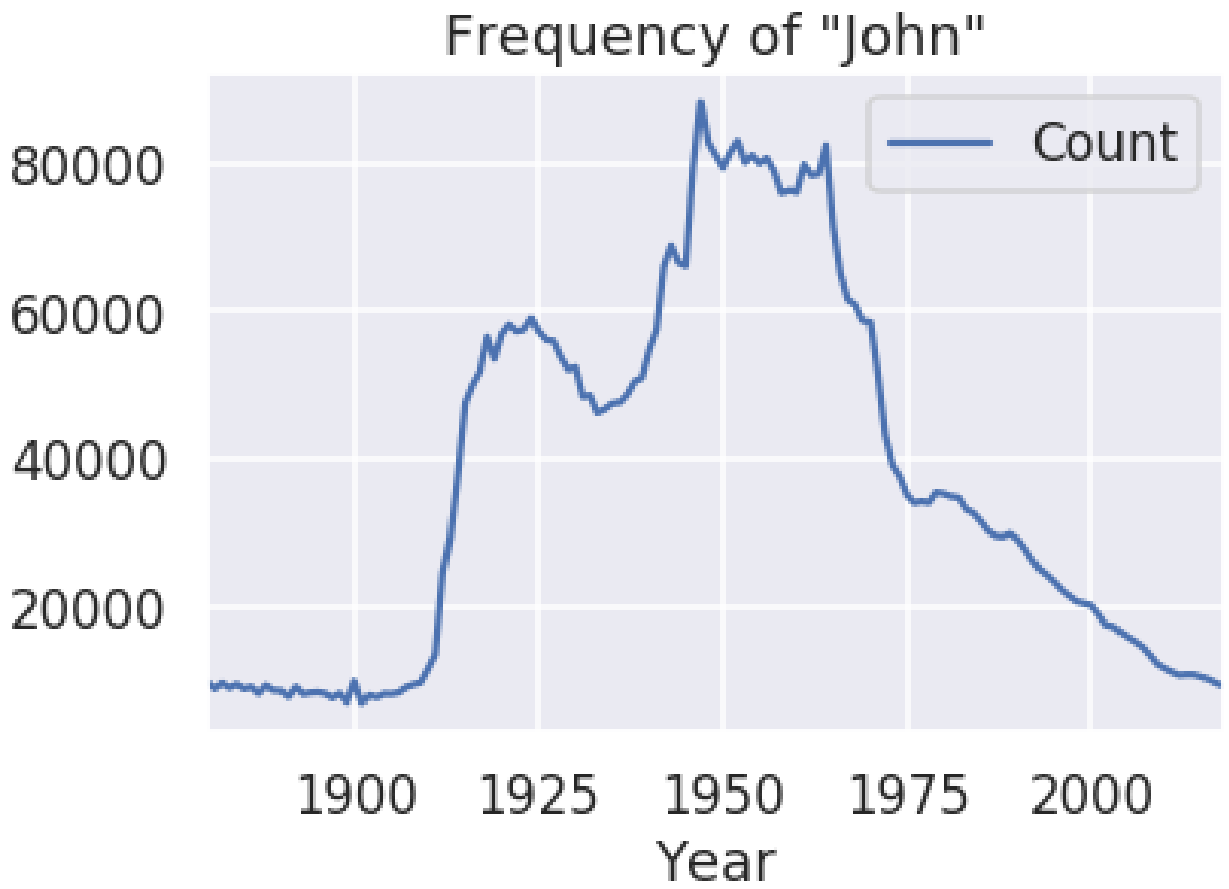
```
sns.distplot(students['year'].dropna());
```




```
students['year'].mean()
```

```
1984.714780016429
```

```
names = babynames.set_index('Name').sort_values('Year')
john = names.loc['john']
john[john['Sex'] == 'M'].plot('Year', 'Count')
plt.title('Frequency of "John"');
```



```
names = babynames.set_index('Name').sort_values('Year')
kanye = names.loc['kanye']
kanye[kanye['Sex'] == 'M'].plot('Year', 'Count')
plt.title('Frequency of "Kanye"');
```