Import modules

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import math
from matplotlib.ticker import FuncFormatter
import plotly
import plotly.figure_factory as ff
from pandas.plotting import parallel_coordinates
import numpy as np

%matplotlib inline
```

Data load and transformation

In [2]:

```
education = pd.read_csv('ex6-2/education.csv')
crime = pd.read_csv('ex6-2/crimeratesbystate-formatted.csv')
birthrate = pd.read_csv('ex6-2/birth-rate.csv')

# remove whitespaces from crime dataset (sine we have already encountered it)
education = education.applymap(lambda x: x.strip() if type(x) is str else x)
crime = crime.applymap(lambda x: x.strip() if type(x) is str else x)
birthrate = birthrate.applymap(lambda x: x.strip() if type(x) is str else x)
```

Histogram

Distribution of birth rate

In [3]:

```
birthrate_hist = pd.melt(birthrate, id_vars="Country", var_name="Year", value_name = 'Birt
birthrate_hist["BirthRate_int"] = birthrate_hist["BirthRate"].apply(lambda x: math.ceil(x)
birthrate_hist.head()
```

Out[3]:

	Country	Year	BirthRate	BirthRate_int
0	Aruba	1960	36.400	37
1	Afghanistan	1960	52.201	53
2	Angola	1960	54.432	55
3	Albania	1960	40.886	41
4	Netherlands Antilles	1960	32.321	33

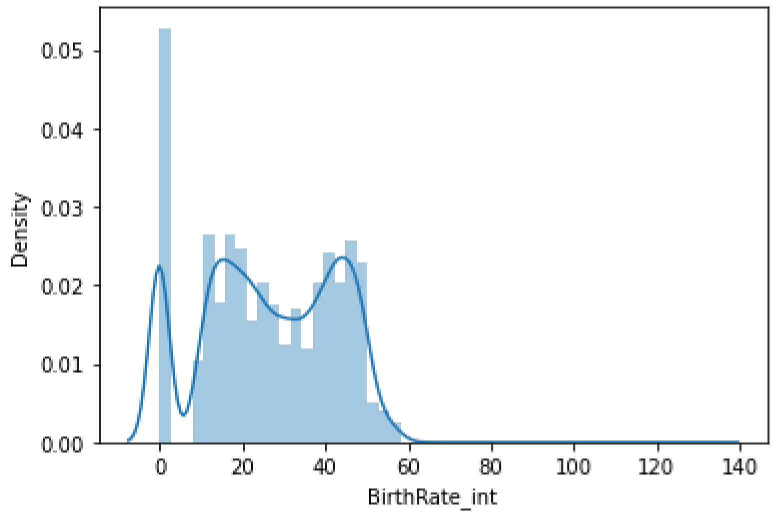
In [4]:

```
sns.distplot( birthrate_hist["BirthRate_int"] )
```

```
/Users/veerareddykoppula/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.p
y:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level function with simi
lar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[4]:

```
<AxesSubplot:xlabel='BirthRate_int', ylabel='Density'>
```

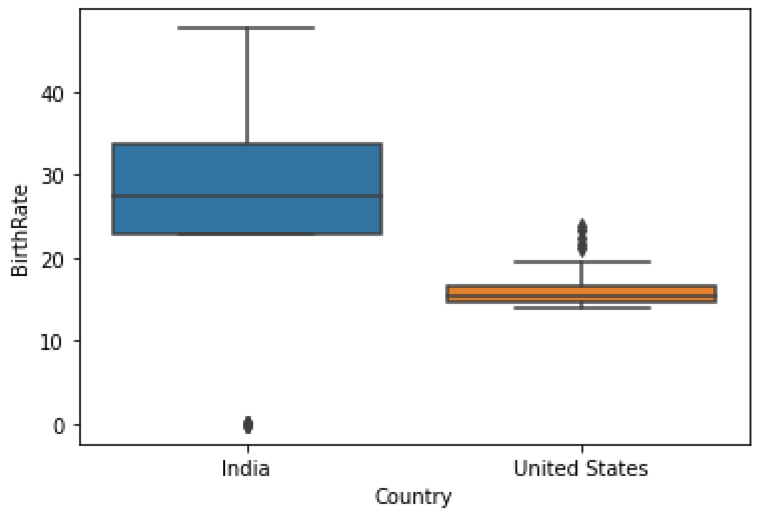


Box plot

Comparison of birthrate betwen India and USA

In [5]:
```
birthrate_box = birthrate_hist[(birthrate_hist["Country"]=="United States") | (birthrate_h
sns.boxplot(x = birthrate_box["Country"], y=birthrate_box["BirthRate"])
```

Out[5]:
```
<AxesSubplot:xlabel='Country', ylabel='BirthRate'>
```



Bullet chart

US burglary statistics against some dummy benchmark

In [6]:
```
# transform data
crime_bullet = crime[crime["state"]=="United States"][["state","burglary"]]
crime_bullet['target'] = 500
crime_bullet_tuple = [tuple(x) for x in crime_bullet.values][0]


# set parameter for bullet chart
limits = [300, 500, 1000]
palette = sns.color_palette("Blues_r", len(limits))
fig, ax = plt.subplots()
ax.set_aspect('equal')
ax.set_yticks([1])
ax.set_yticklabels='United States'

prev_limit = 0
```

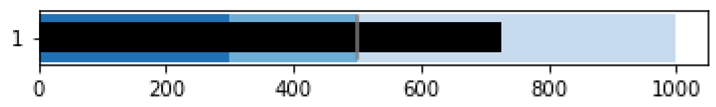
```
for idx, lim in enumerate(limits):
    ax.barh([1], lim-prev_limit, left=prev_limit, height=75, color=palette[idx])
    prev_limit = lim

# draw the value we're measuring
ax.barh([1], crime_bullet_tuple[1], color='black', height=45)

ax.axvline(crime_bullet_tuple[2], color="gray", ymin=0.10, ymax=0.9)
```

Out[6]:

```
<matplotlib.lines.Line2D at 0x7fdf0d76b640>
```



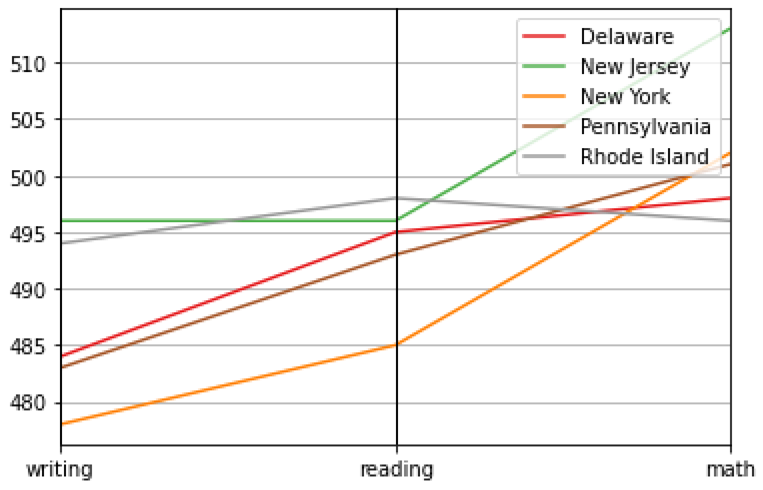
Parallel Coordinate plot

Comparison of reading, writing and math numbers between 5 states

In [7]:

```
# transform data
education_parallel = education[education['state'].isin(['New York','New Jersey','Delaware'

# make the plot
parallel_coordinates(education_parallel, 'state', colormap=plt.get_cmap("Set1"))
plt.show()
```



Pie chart

Comparison of reading numbers between 5 states

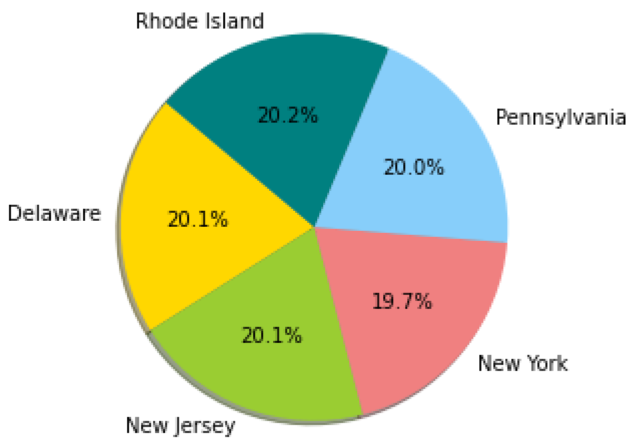
In [8]:

```
# transform data
education_pie = education_parallel[['state','reading']]

# set colors
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue','teal']

# plot
plt.pie(education_pie['reading'], labels=education_pie['state'], colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```



Donought chart

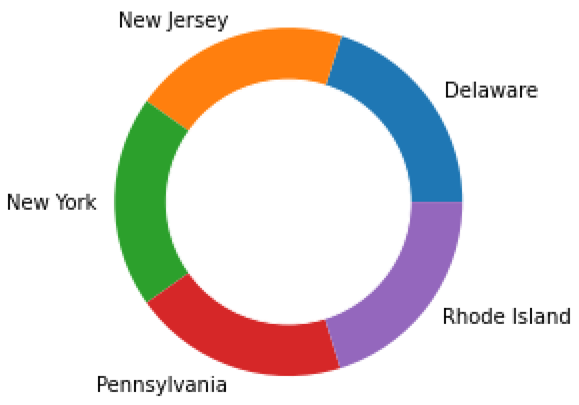
Comparison of reading, writing and math numbers between 5 states

In [9]:
```
# transform data
education_donut = education_pie

# create a pieplot
plt.pie(education_donut['reading'], labels=education_donut['state'])

# add a circle at the center
my_circle=plt.Circle( (0,0), 0.7, color='white')
p=plt.gcf()
p.gca().add_artist(my_circle)

plt.show()
```



In [ ]:
```
```