Session 14: Data Aggregation using Groupby

Setting up

[2]:
```
import pandas as pd
base='https://raw.githubusercontent.com/chendaniely/pandas_for_everyone/master/data/'
filename='gapminder.tsv'
gapminder=pd.read_csv(base+filename,sep='\t')
gapminder.head()
```

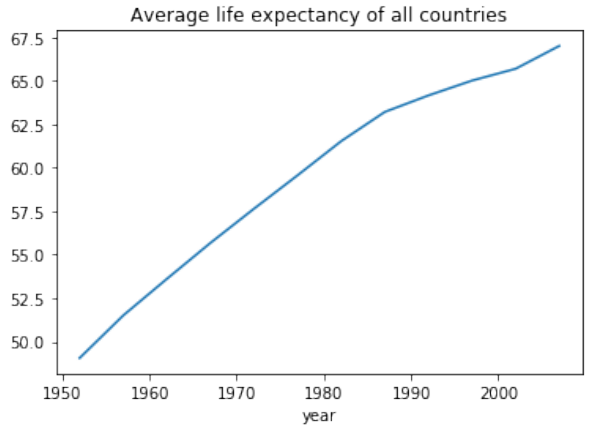
	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
4	Afghanistan	Asia	1972	36.088	13079460	739.981106

1. Grouping by One Column

[2]:
```
gapminder.groupby('year')['lifeExp'].mean().head()
```

```
year
1952    49.057620
1957    51.507401
1962    53.609249
1967    55.678290
1972    57.647386
Name: lifeExp, dtype: float64
```

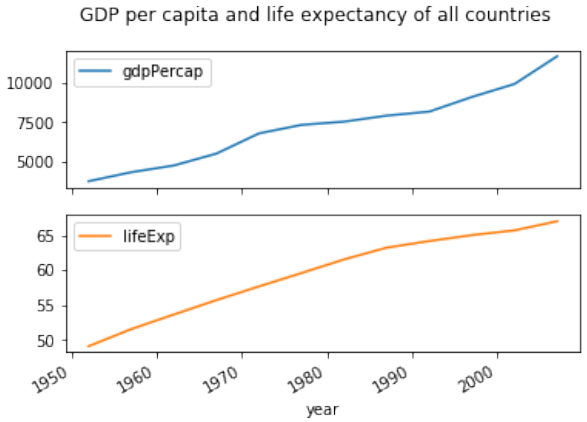
[4]:
```
gapminder.groupby('year')['lifeExp'].mean()\
        .plot(title='Average life expectancy of all countries')
```



[4]:
```
gapminder.groupby('year')[['gdpPercap','lifeExp']].mean().head()
```

year	gdpPercap	lifeExp
1952	3725.276046	49.057620
1957	4299.408345	51.507401
1962	4725.812342	53.609249
1967	5483.653047	55.678290
1972	6770.082815	57.647386

[5]:
```
gapminder.groupby('year')[['gdpPercap','lifeExp']].mean()\
        .plot(title='GDP per capita and life expectancy of all countries',subplots=True)
```



List of Pandas methods built into groupby:

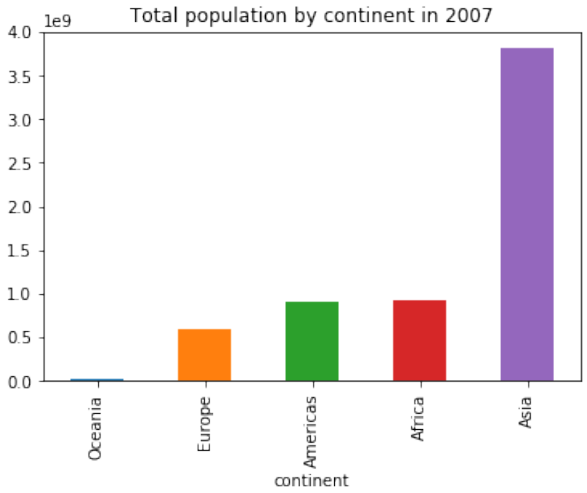
Method	Description
`count`	Number of elements not including NaN values.
`size`	Number of elements including NaNs.
`sum`	Total.
`mean`	Average.
`median`	Median.
`var`	Variance.
`std`	Standard deviation.
`min`	Minimum.
`max`	Maximum.
`quantile(0.28)`	28th percentile from the bottom.
`describe`	count, mean, std, min, 25%, 50%, 75%, and max
`first`	first non-null value.
`last`	last non-null value.
`nth(3)`	3rd value (does not skip null values).

[6]:
```
import numpy as np
gapminder.groupby('year').agg({'lifeExp':'mean','gdpPercap':'median'}).head()
```

```
        lifeExp    gdpPercap
year
1952  49.057620  1968.528344
1957  51.507401  2173.220291
1962  53.609249  2335.439533
1967  55.678290  2678.334741
1972  57.647386  3339.129407
```

Q1: Create a bar plot comparing the total populations of each continent in 2007, as below.
(Hint: First use "query" to filter for year being 2007, then group by the continent and compute the sum of the "pop" column. Then sort the result using "sort_values" and plot using "kind='bar'". All this can be chained together into one line.)

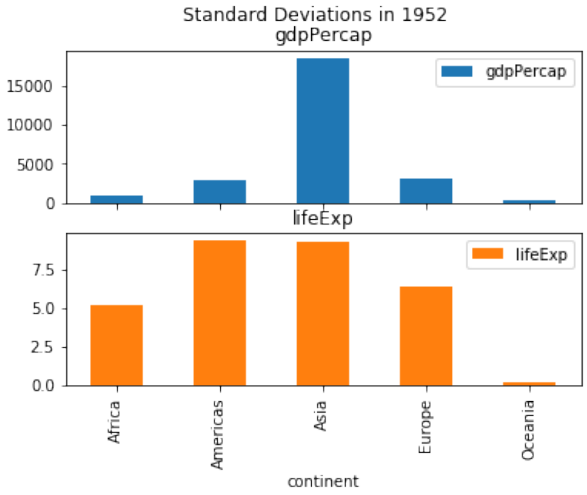
[7]:



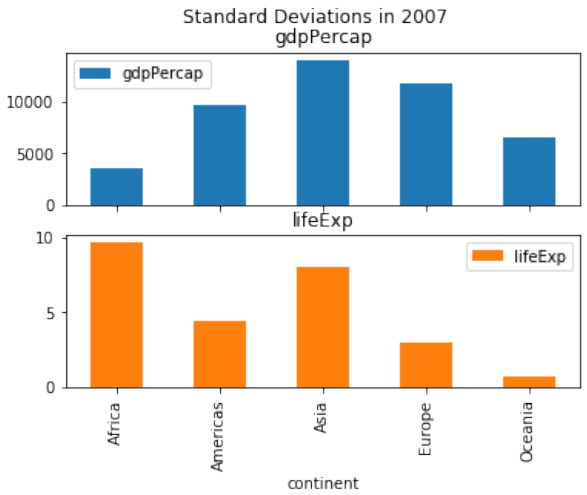
Q2: Create the following plots to compare the standard deviation in GDP per capita and life expectancy across countries within each continent, in 1952 and in 2007.

(Hint: for each graph, first use "query" to filter for the year, then group by the continent, and compute the standard deviation of both "gdpPercap" and "lifeExp". Then plot using "subplots=True". Each plot can be created using one line by chaining together commands.)

[8] :

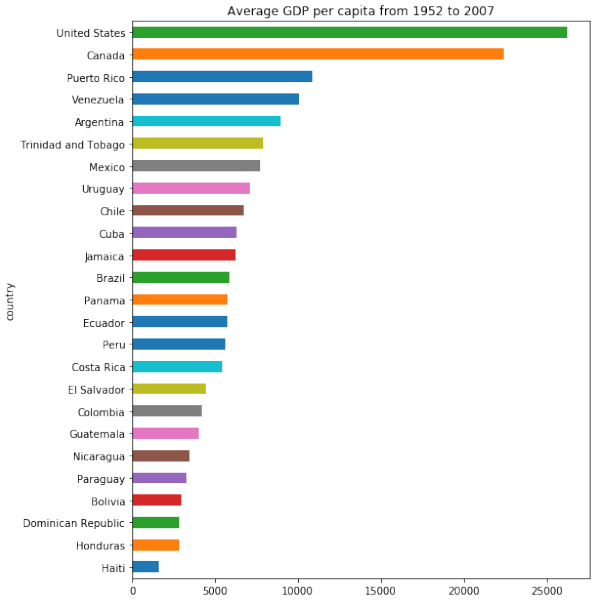


[9] :



Q3: Plot the average GDP per capita over the years in the dataset for all countries in the continent "Americas", as below.

[10]:



2. Grouping by Multiple Columns

Grouping by multiple columns is completely analogous, except for the result having multiple levels of index.

[11]:
```
result=gapminder.groupby(['year','continent'])['lifeExp'].mean()
result.head(10)
```

```
year  continent
1952  Africa       39.135500
      Americas     53.279840
      Asia         46.314394
      Europe       64.408500
      Oceania      69.255000
1957  Africa       41.266346
      Americas     55.960280
      Asia         49.318544
      Europe       66.703067
      Oceania      70.295000
Name: lifeExp, dtype: float64
```

[12]:
```
gapminder.groupby(['year','continent']).std().head()
```

```
                lifeExp           pop     gdpPercap
year continent
1952 Africa     5.151581  6.317450e+06    982.952116
     Americas   9.326082  3.234163e+07   3001.727522
     Asia       9.291751  1.132267e+08  18634.890865
     Europe     6.361088  1.724745e+07   3114.060493
     Oceania    0.190919  4.735083e+06    365.560078
```

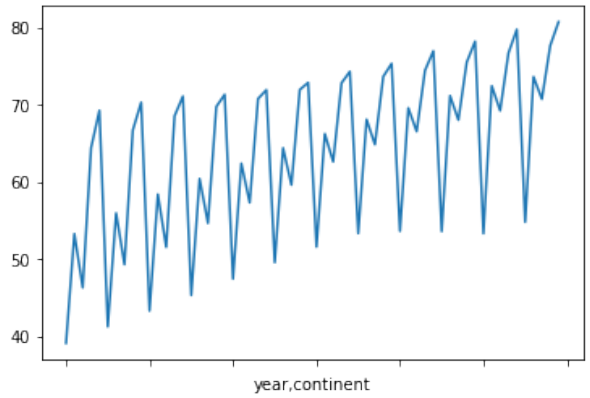
```
[13]: gapminder.groupby(['year','continent'])\
          .agg({'lifeExp':'mean','gdpPercap':'median'}).head()
```

```
                  lifeExp     gdpPercap
year continent
1952 Africa     39.135500    987.025569
     Americas   53.279840   3048.302900
     Asia       46.314394   1206.947913
     Europe     64.408500   5142.469716
     Oceania    69.255000  10298.085650
```

2.1 Working with MultiIndex Objects

The Series "result" has two levels of index, year and continent. This is an example of a Hierarchical Index or MultiIndex in Pandas. The data is the rightmost column above, while the two columns on the left are both row labels.

```
[14]: result.plot()
```



You can change a Series to a DataFrame by the command "reset_index", which get rid of the MultiIndex.

```
[15]: result.reset_index().head()
```

```
   year continent    lifeExp
0  1952    Africa  39.135500
1  1952  Americas  53.279840
2  1952      Asia  46.314394
3  1952    Europe  64.408500
4  1952   Oceania  69.255000
```

```
[16]: result.reset_index().groupby('continent')['lifeExp'].std()
```

```
continent
Africa      5.443393
Americas    6.663477
Asia        8.236002
Europe      4.042574
Oceania     3.843914
Name: lifeExp, dtype: float64
```

The "unstack" method moves one level of the MultiIndex to the columns, as below. The default behavior is to move the right most level.

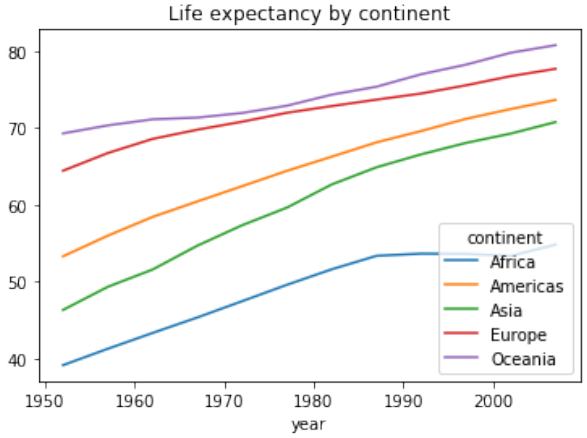
```
[17]: result.unstack().head()
```

```
continent     Africa  Americas       Asia     Europe  Oceania
year
1952       39.135500  53.27984  46.314394  64.408500   69.255
1957       41.266346  55.96028  49.318544  66.703067   70.295
1962       43.319442  58.39876  51.563223  68.539233   71.085
1967       45.334538  60.41092  54.663640  69.737600   71.310
1972       47.450942  62.39492  57.319269  70.775033   71.910
```

```
[18]: result.unstack(level=0).iloc[:5,:5]
```

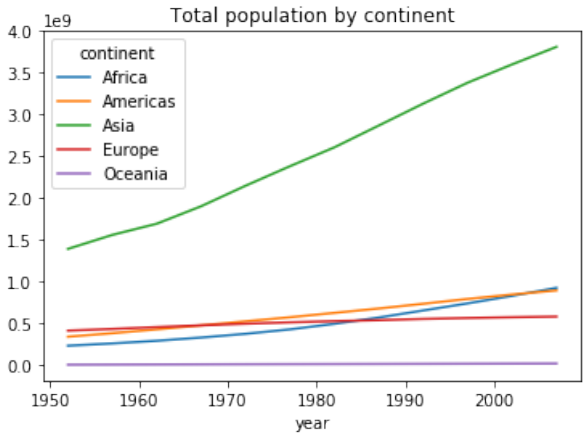
```
year            1952       1957       1962       1967       1972
continent
Africa     39.135500  41.266346  43.319442  45.334538  47.450942
Americas   53.279840  55.960280  58.398760  60.410920  62.394920
Asia       46.314394  49.318544  51.563223  54.663640  57.319269
Europe     64.408500  66.703067  68.539233  69.737600  70.775033
Oceania    69.255000  70.295000  71.085000  71.310000  71.910000
```

```
[19]: gapminder.groupby(['year','continent'])['lifeExp'].mean()\
          .unstack().plot(title="Life expectancy by continent")
```



Q4: Plot the trend in total population of each continent as below.

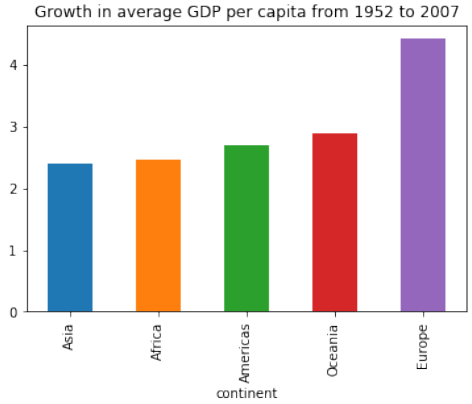
```
[20]:
```



Q5: Compute the average GDP per capita for each continent in 1952 and 2007, and plot the ratio.

(Hint: a quick way is to first group by the continent and year and compute the average GDP Per capita for each combination, then unstack it so that the years are the columns, similar to in Out[30]. Then you can compute the desired ratio by dividing the column for 2007 by the column for 1952.)

[22]:



Q6. Create a plot over time of the difference in total GDP between the richest and the poorest continent, as below.

(Hint: first add a "GDP" column in the gapminder DataFrame by multiplying the "gdpPercap" and "pop" columns. Then create a "gdpSum" DataFrame by grouping by the year and continent, and summing the GDPs. See below for what this DataFrame looks like. Using this DataFrame, you can compute a Series called "maxGDP" by grouping by the year and finding the max GDP, and similarly compute a Series called "minGDP". Both of these are indexed by year. Finally, subtract maxGDP by minGDP and plot the result.)

[23]:

```
   year continent           GDP
0  1952    Africa  3.115993e+11
1  1952  Americas  2.943475e+12
2  1952      Asia  1.125160e+12
```

[26]: