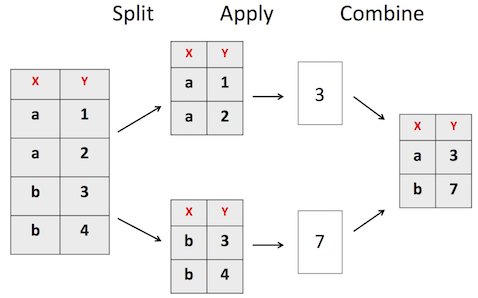
split-apply-combine for Data Analysis

The definition of this strategy is as per the name. **Splitting** the data into groups based on some criteria, **applying** a function to each group, **Combining** the outcome into a data structure. Here I will discuss how to use *GroupBy* objects from *pandas* for split-apply-combine strategy on a dataset. This can be done in R as well using dplyr package. Hadley William formalized the split-apply-combine procedure in 2009 with his paper *The Split-Apply-Combine Strategy for Data Analysis*. Below figure shows a pictorial representation of the process.



Lets take example of netflix data which contains titles, ratings, release year and user rating score, among several other columns. First we will load the data and review basic information about data.

>>> import pandas as pd  
>>> nflx = pd.read\_csv('netflix.csv')  
>>> nflx.head()

|  | **title** | **rating** | **ratingLevel** | **ratingDescription** | **release year** | **user rating score** | **user rating size** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | White Chicks | PG-13 | crude and sexual humor, language and some drug... | 80 | 2004 | 82.0 | 80 |
| 1 | Lucky Number Slevin | R | strong violence, sexual content and adult lang... | 100 | 2006 | NaN | 82 |
| 2 | Grey's Anatomy | TV-14 | Parents strongly cautioned. May be unsuitable ... | 90 | 2016 | 98.0 | 80 |
| 3 | Prison Break | TV-14 | Parents strongly cautioned. May be unsuitable ... | 90 | 2008 | 98.0 | 80 |
| 4 | How I Met Your Mother | TV-PG | Parental guidance suggested. May not be suitab... | 70 | 2014 | 94.0 | 80 |

# Drop rows with missing values and drop duplicate  
nflx.dropna(inplace=True)  
nflx.drop\_duplicates(inplace=True)

# Get summary stats of df  
df.describe()

|  | **ratingDescription** | **release year** | **user rating score** | **user rating size** |
| --- | --- | --- | --- | --- |
| count | 246.000000 | 246.000000 | 246.000000 | 246.0 |
| mean | 73.556911 | 2010.272358 | 81.390244 | 80.0 |
| std | 26.616145 | 8.887219 | 12.677883 | 0.0 |
| min | 10.000000 | 1940.000000 | 55.000000 | 80.0 |
| 25% | 60.000000 | 2007.000000 | 71.000000 | 80.0 |
| 50% | 80.000000 | 2015.000000 | 83.500000 | 80.0 |
| 75% | 90.000000 | 2016.000000 | 92.750000 | 80.0 |
| max | 124.000000 | 2017.000000 | 99.000000 | 80.0 |

**Step-1: Split**

We will use GroupBy method for splitting the movie data by release year.  It will create groupby object.

# Group by year  
nflx\_by\_year = nflx.groupby('release year')  
  
# Check type of GroupBy object  
type(nflx\_by\_year)

pandas.core.groupby.generic.DataFrameGroupBy

**Step-2: Apply**

There are many methods can be applied to DataFrameGroupBy object. For example we will apply .describe() method.

# Summary stats over years  
nflx\_by\_year.describe().head()

|  | **ratingDescription** | | | | | | | | **user rating score** | | | | | **user rating size** | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** | **count** | **mean** | **...** | **75%** | **max** | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| **release year** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1940 | 1.0 | 35.0 | NaN | 35.0 | 35.0 | 35.0 | 35.0 | 35.0 | 1.0 | 61.0 | ... | 61.0 | 61.0 | 1.0 | 80.0 | NaN | 80.0 | 80.0 | 80.0 | 80.0 | 80.0 |
| 1978 | 1.0 | 60.0 | NaN | 60.0 | 60.0 | 60.0 | 60.0 | 60.0 | 1.0 | 86.0 | ... | 86.0 | 86.0 | 1.0 | 80.0 | NaN | 80.0 | 80.0 | 80.0 | 80.0 | 80.0 |
| 1982 | 1.0 | 60.0 | NaN | 60.0 | 60.0 | 60.0 | 60.0 | 60.0 | 1.0 | 68.0 | ... | 68.0 | 68.0 | 1.0 | 80.0 | NaN | 80.0 | 80.0 | 80.0 | 80.0 | 80.0 |
| 1986 | 1.0 | 35.0 | NaN | 35.0 | 35.0 | 35.0 | 35.0 | 35.0 | 1.0 | 67.0 | ... | 67.0 | 67.0 | 1.0 | 80.0 | NaN | 80.0 | 80.0 | 80.0 | 80.0 | 80.0 |
| 1987 | 1.0 | 60.0 | NaN | 60.0 | 60.0 | 60.0 | 60.0 | 60.0 | 1.0 | 58.0 | ... | 58.0 | 58.0 | 1.0 | 80.0 | NaN | 80.0 | 80.0 | 80.0 | 80.0 | 80.0 |

**Step-3: Combine**

Lets say we need to determine mean() or median() of ratings by year, we can apply the .mean() or .median() method to groupby object and then store the result into a new dataframe.

# Get mean values by year and print head  
nflx\_mean\_yr = nflx\_by\_year.mean()  
nflx\_mean\_yr.head()

|  | **ratingDescription** | **user rating score** | **user rating size** |
| --- | --- | --- | --- |
| **release year** |  |  |  |
| 1940 | 35.0 | 61.0 | 80.0 |
| 1978 | 60.0 | 86.0 | 80.0 |
| 1982 | 60.0 | 68.0 | 80.0 |
| 1986 | 35.0 | 67.0 | 80.0 |
| 1987 | 60.0 | 58.0 | 80.0 |

***References:***

Wickham, H. (2011). The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, *40*(1), 1-29.

<https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html#splitting-an-object-into-groups>

<https://cmdlinetips.com/2018/02/introduction-to-split-apply-combine-with-pandas/>