Solutions 4

Jumping Rivers

Data manipulation

Start by loading the data and importing the packages we will need for this practical.

```
import pandas as pd
import numpy as np
import jrpyml.datasets as dat
movies = dat.load_movies()
```

1. The data has a lot of missing values in, particularly for budgets. We can remove them with

```
movies.dropna(inplace = True, subset=['budget'])
```

2. Calculate the average budget across all films.

```
movies.budget.mean()
```

```
## 13412513.24966443
```

3. Create a new column in your `DataFrame` that takes True of False, with True representing films that are more recent than 1970.

```
movies['recent'] = movies.year > 1970
```

4. Use this new column to calculate the average budget for the older films and the newer films. Hint: See `.groupby()`

```
movies.groupby('recent').agg({'budget': 'mean'})
```

```
##                budget
## recent
## False   1.926708e+06
## True    1.669094e+07
```

5. Calculate the average and standard deviations for lengths, budgets and ratings for the films in each year. Store all of the results in a single `DataFrame`.

```
x = movies.groupby('year').\
  agg({
    'length': ['mean','std'],
    'budget': ['mean','std'],
    'rating': ['mean','std']
  })
```

6. The previous calculation gives a multi index `DataFrame`. Essentially a heirarchy of indices, have a look at `.head()` on the result of the previous question to see what I mean. You can extract specific sub indicies with, for example, `x[('rating','mean')]` to get the mean column inside the length index.[1] Extract just the means from the previous result and store that answer.

```
y = x[[('length','mean'),('rating','mean'),('budget','mean')]]
```

7. To finish we will draw a plot with all 3 lines showing the averages evolving over time. To make the axes relevant, we will first scale all of our values to be on (0,1). Given a `DataFrame`, `y`, this could be acheived with

```
mins = y.min()
maxs = y.max()

rescaled = (1/(maxs-mins))*(y-maxs) + 1
```

8. Use `rescaled.plot()` to draw all 3 lines together. Is there anything interesting?

```
rescaled.plot()
# Interesting is of course subjective, but I think it is interesting that the budget
# has consistently gone up, but average ratings of films don't follow that,
# neither do the lengths
plt.show()
```

[1] Specifically here we are passing a tuple of indicies.