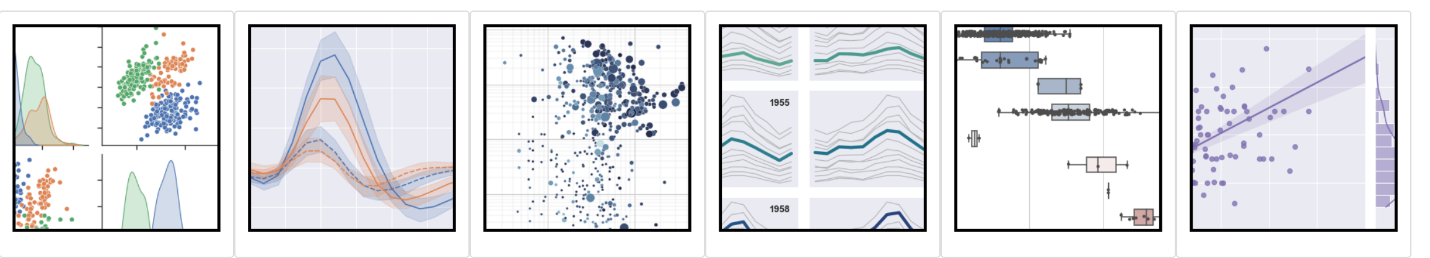
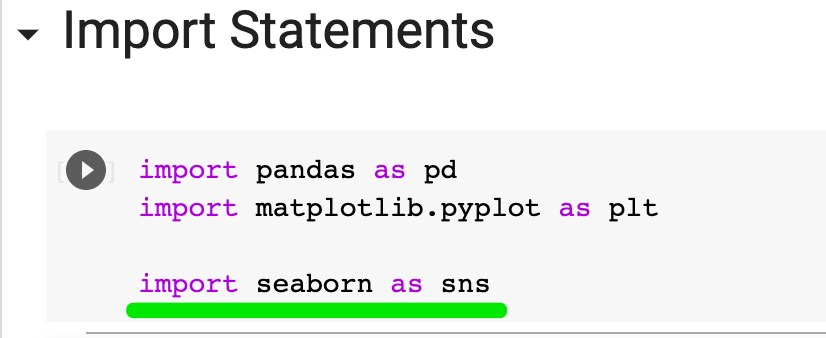
**Seaborn Data Visualisation: Bubble Charts**

We're now ready to visualise our data. Today I want to introduce you to another popular data visualisation tool that you can use alongside plotly and Matplotlib: Seaborn. [Seaborn](https://seaborn.pydata.org/) is built on top of Matplotlib and it makes creating certain visualisations very convenient.



**Import Seaborn**

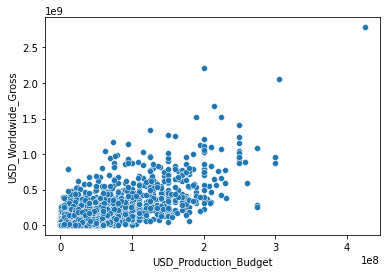
The first step is adding Seaborn to our notebook. By convention we'll use the name sns.



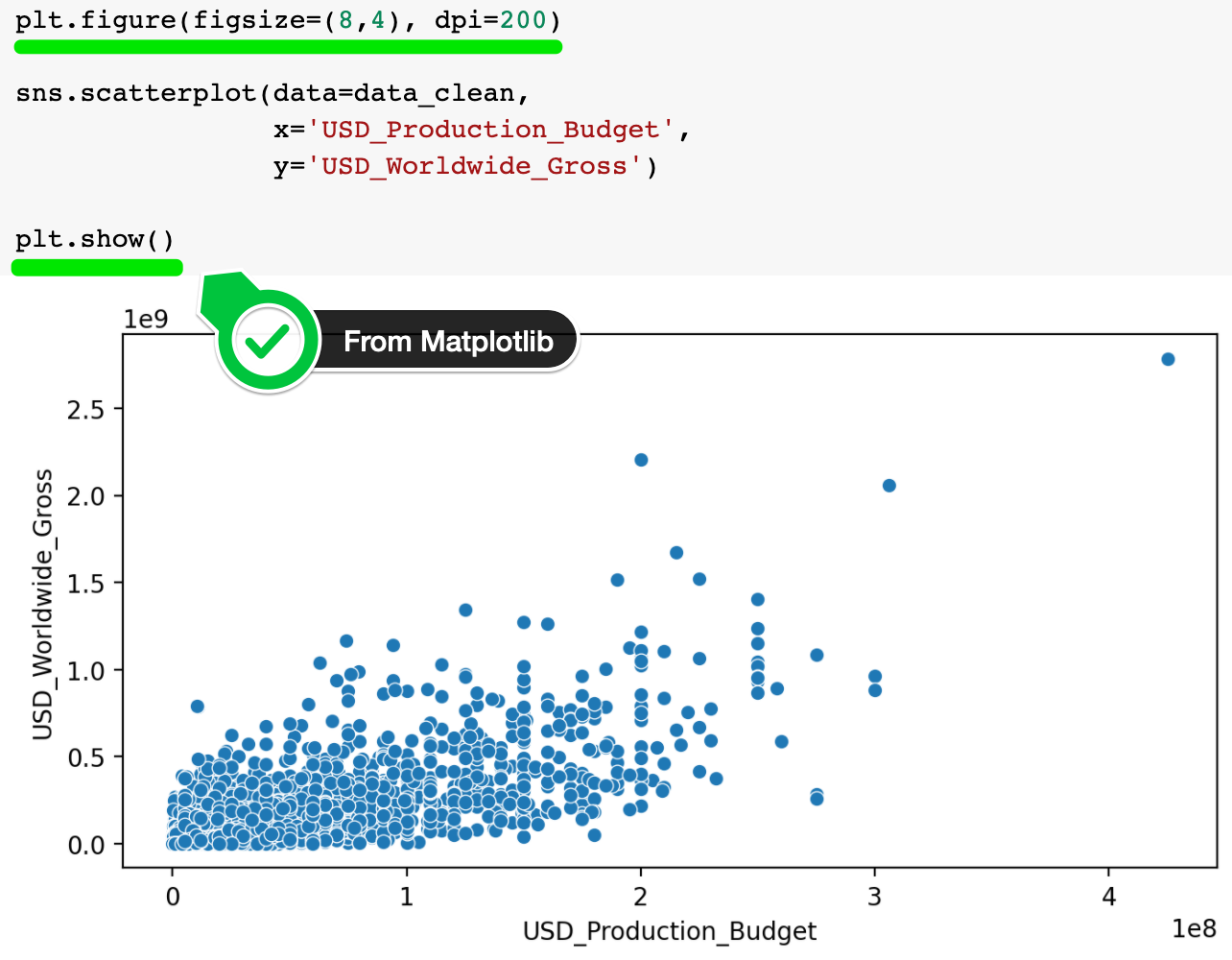
**Seaborn Scatter Plots**

To create a [.scatterplot()](https://seaborn.pydata.org/generated/seaborn.scatterplot.html?highlight=scatterplot#seaborn.scatterplot), all we need to do is supply our DataFrame and the column names that we'd like to see on our axes.

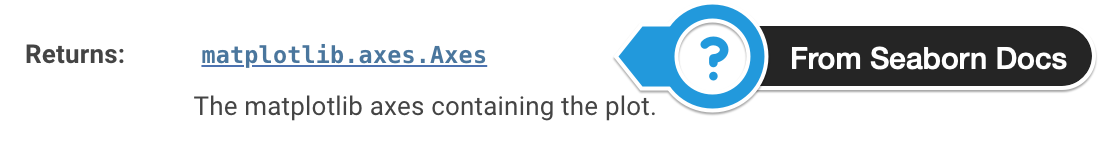
1. sns.scatterplot(data=data\_clean,
2. x='USD\_Production\_Budget',
3. y='USD\_Worldwide\_Gross')



That should look familiar. 😊 Because Seaborn is built on top of Matplotlib, we can dive into the Matplotlib layer anytime to configure our chart. For example, we can increase the size of our figure:



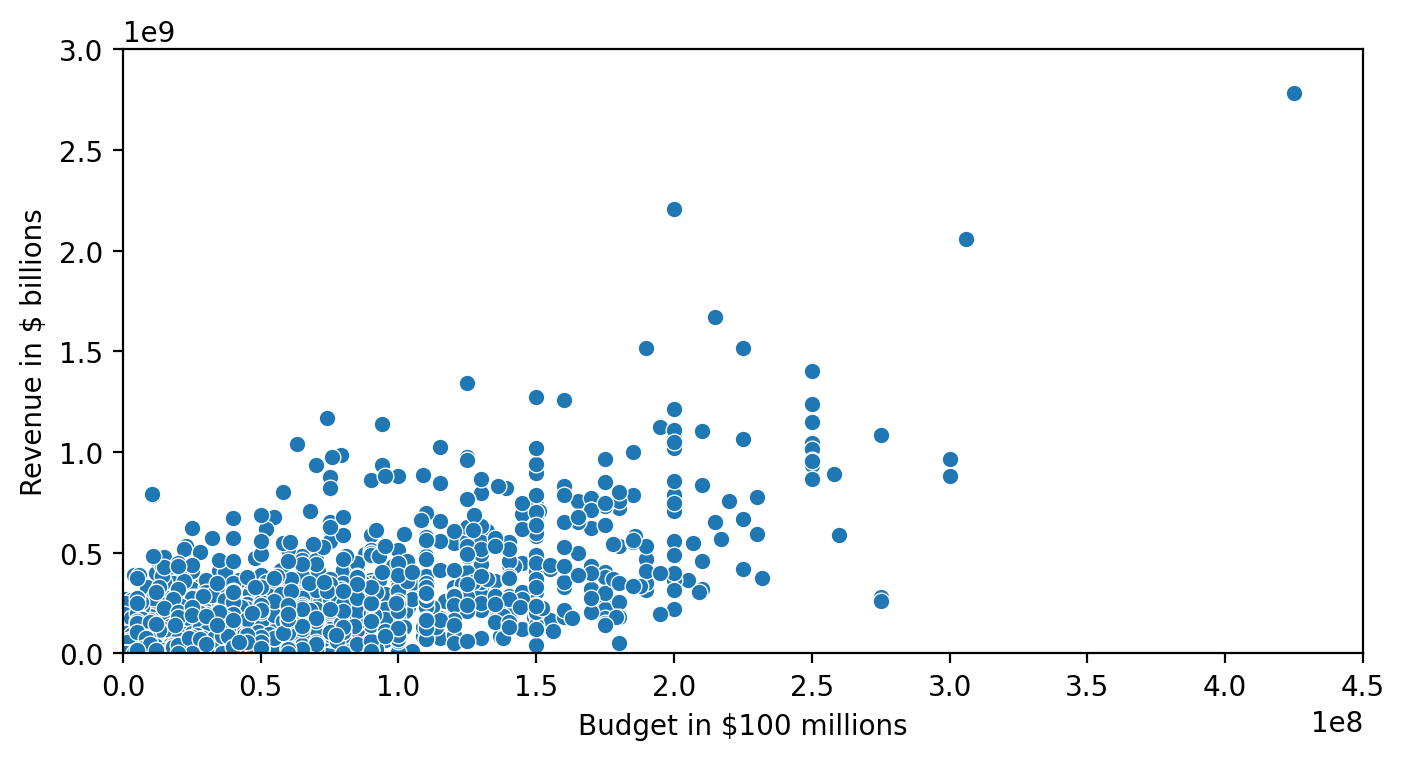
And to style our chart we can simply configure the Axes object that is returned from sns.scatterplot().



Here's how:

1. plt.figure(figsize=(8,4), dpi=200)
3. ax = sns.scatterplot(data=data\_clean,
4. x='USD\_Production\_Budget',
5. y='USD\_Worldwide\_Gross')
7. ax.set(ylim=(0, 3000000000),
8. xlim=(0, 450000000),
9. ylabel='Revenue in $ billions',
10. xlabel='Budget in $100 millions')
12. plt.show()

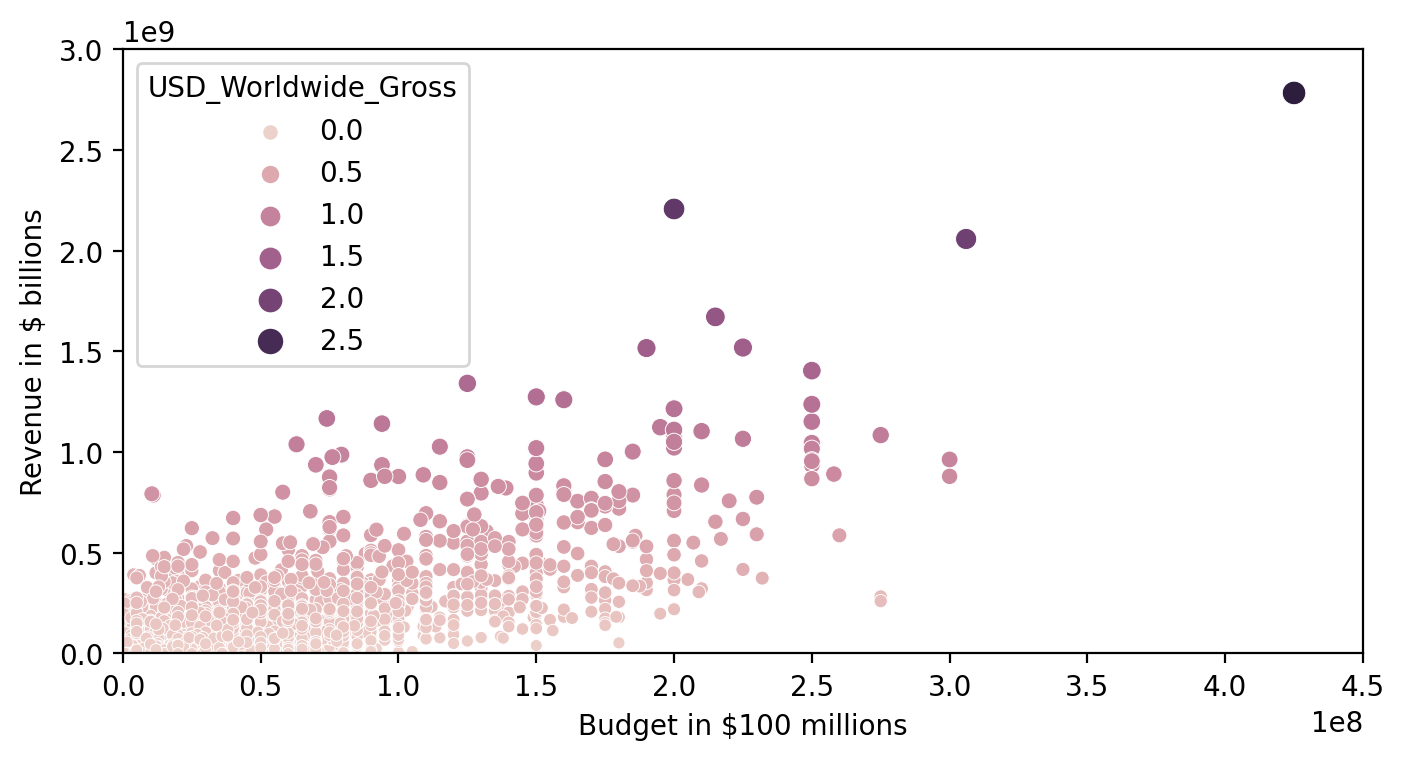
Here we're diving into the Matplotb layer to set the limits on the axes and change the labels.



**From Scatter Plot to Bubble Chart**

But the reason we're using Seaborn is because of the hue and size parameters that make it very easy to create a bubble chart. These parameters allow us to colour the data and change their size according to one of the columns in our DataFrame.

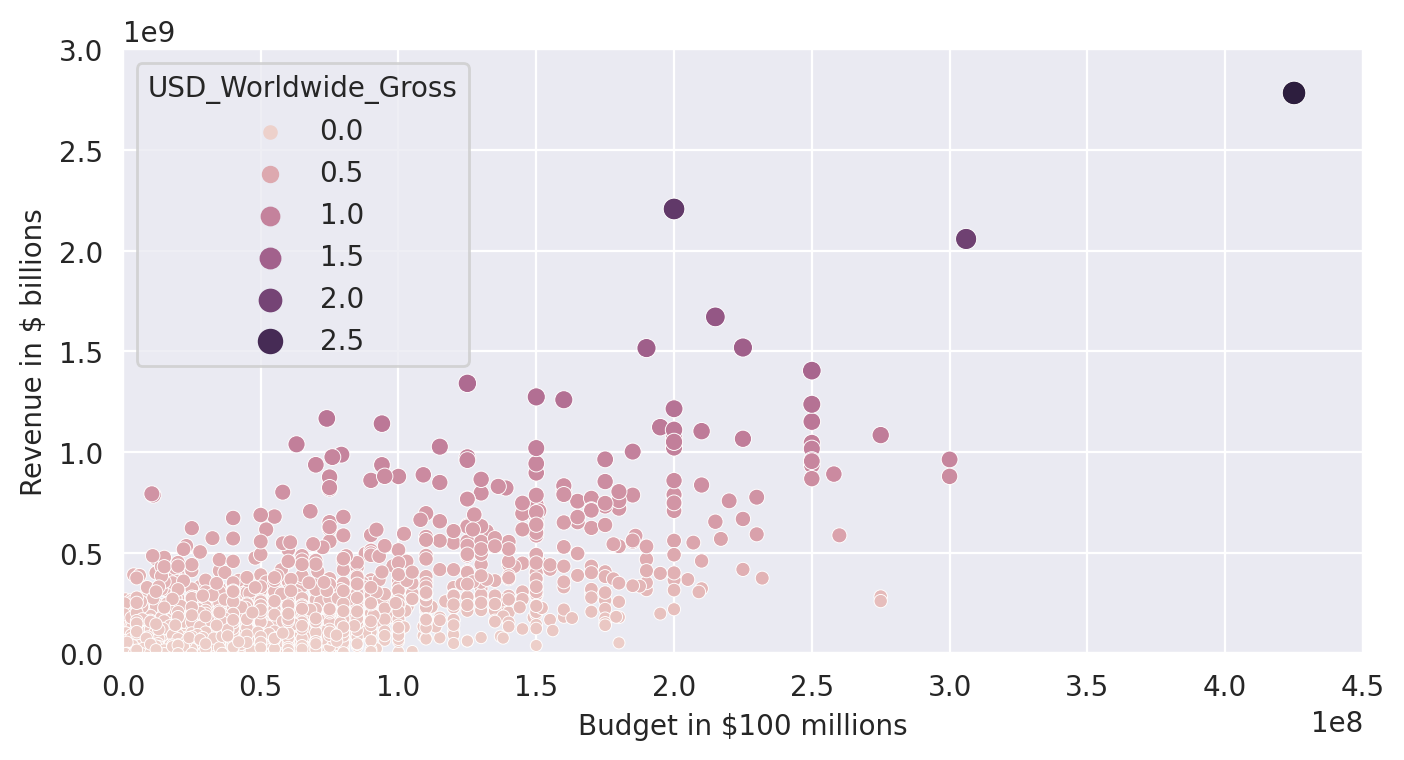
1. plt.figure(figsize=(8,4), dpi=200)
2. ax = sns.scatterplot(data=data\_clean,
3. x='USD\_Production\_Budget',
4. y='USD\_Worldwide\_Gross',
5. hue='USD\_Worldwide\_Gross', # colour
6. size='USD\_Worldwide\_Gross',) # dot size
8. ax.set(ylim=(0, 3000000000),
9. xlim=(0, 450000000),
10. ylabel='Revenue in $ billions',
11. xlabel='Budget in $100 millions',)
13. plt.show()



Now our higher grossing movies are bigger and darker on our chart. That's super handy. But Seaborn offers a number of convenient styling options as well.

To set the styling on a single chart (as opposed to all the charts in the entire notebook) we can use Python's with keyword. We've seen with used already when it comes to opening files in previous lessons.

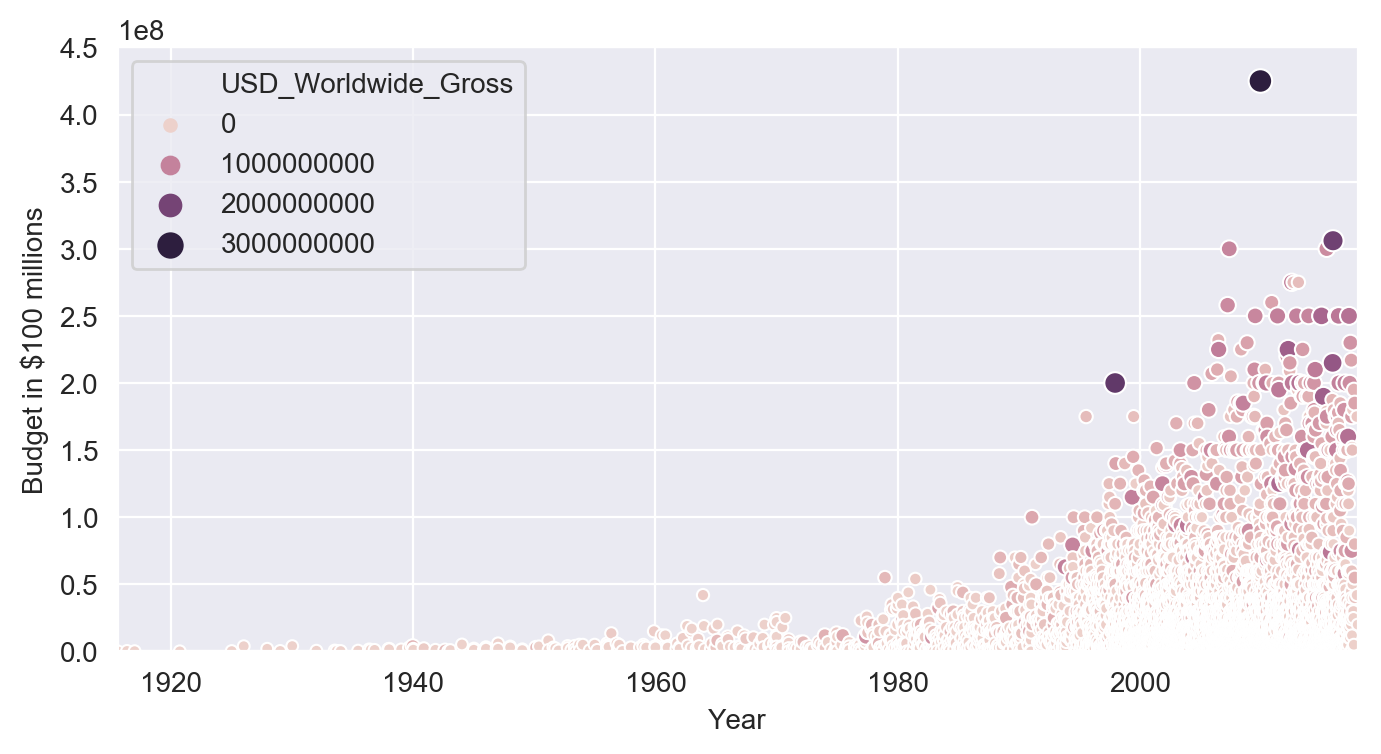
1. plt.figure(figsize=(8,4), dpi=200)
3. # set styling on a single chart
4. with sns.axes\_style('darkgrid'):
5. ax = sns.scatterplot(data=data\_clean,
6. x='USD\_Production\_Budget',
7. y='USD\_Worldwide\_Gross',
8. hue='USD\_Worldwide\_Gross',
9. size='USD\_Worldwide\_Gross')
11. ax.set(ylim=(0, 3000000000),
12. xlim=(0, 450000000),
13. ylabel='Revenue in $ billions',
14. xlabel='Budget in $100 millions')



In addition to 'darkgrid', Seaborn has a number of [built-in themes](https://python-graph-gallery.com/104-seaborn-themes/). so you can style your chart very quickly. Try out 'whitegrid', 'dark',  or 'ticks' for example.

Challenge

Now that you've seen how to create a beautiful bubble chart in Seaborn, it's time to create your own. Can you write the code to replicate this chart? Notice how we are actually representing THREE dimensions in this chart: the budget, the release date, and the worldwide revenue. This is what makes bubble charts so awesomely informative.



.

.

..

...

..

.

.

**Solution: Movie Budgets over Time**

Alright, I hope that was fairly straightforward. All you needed to do is change a few arguments:

1. plt.figure(figsize=(8,4), dpi=200)
3. with sns.axes\_style("darkgrid"):
4. ax = sns.scatterplot(data=data\_clean,
5. x='Release\_Date',
6. y='USD\_Production\_Budget',
7. hue='USD\_Worldwide\_Gross',
8. size='USD\_Worldwide\_Gross',)
10. ax.set(ylim=(0, 450000000),
11. xlim=(data\_clean.Release\_Date.min(), data\_clean.Release\_Date.max()),
12. xlabel='Year',
13. ylabel='Budget in $100 millions')

**Analysis**

What do we see here? What is this chart telling us? Well, first off, movie budgets have just exploded in the last 40 years or so. Up until the 1970s, the film industry appears to have been in an entirely different era. Budgets started growing fast from the 1980s onwards and continued to grow through the 2000s. Also, the industry has grown massively, producing many more films than before. The number of data points is so dense from 2000 onwards that they are overlapping.

