visualization-lecture-2-dec-batch

June 22, 2023

0.1 Data Visualization Lecture - 2

[1]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
import pandas as pd
import numpy as np
```

[3]:
```
data = pd.read_excel('final_vg.xlsx', sheet_name='final_vg')
```

[4]:
```
data
```

[4]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales
0	2061.0	1942.0	NES	1985.0	Shooter	Capcom	4.569217	3.033887
1	9137.0	¡Shin Chan Flipa en colores!	DS	2007.0	Platform	505 Games	2.076955	1.493442
2	14279.0	.hack: Sekai no Mukou ni + Versus	PS3	2012.0	Action	Namco Bandai Games	1.145709	1.762339
3	8359.0	.hack//G.U. Vol.1//Rebirth	PS2	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856
4	7109.0	.hack//G.U. Vol.2//Reminisce	PS2	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054
…	…	…	…	…	…	…	…	…
16647	7925.0	Zumba Fitness Rush	X360	2012.0	Sports	505 Games	4.409308	3.167419
16648	6279.0	Zumba Fitness: World Party	Wii	2013.0	Misc	Majesco Entertainment	3.033887	2.792725
16649	6977.0	Zumba Fitness: World Party	XOne	2013.0	Misc	Majesco Entertainment	3.228043	2.004268
16650	15422.0	Zwei!!	PSP	2008.0	Role-Playing	Falcom Corporation	1.087977	0.592445
16651	12919.0	Zyuden Sentai Kyoryuger: Game de Gaburincho!!	3DS	2013.0	Action	Namco Bandai Games	1.081046	1.714664

```
       JP_Sales  Other_Sales  Global_Sales
0      3.439352     1.991671     12.802935
1      3.033887     0.394830      7.034163
2      1.493442     0.408693      4.982552
3      3.228043     0.394830      7.226880
4      1.440483     1.493442      8.363113
…         …            …            …
16647  4.168474     1.087977     13.053204
16648  1.596852     1.493442      8.878837
16649  1.833151     1.087977      7.954274
16650  1.087977     0.394830      3.509168
16651  2.004268     0.394830      5.132196

[16652 rows x 11 columns]
```

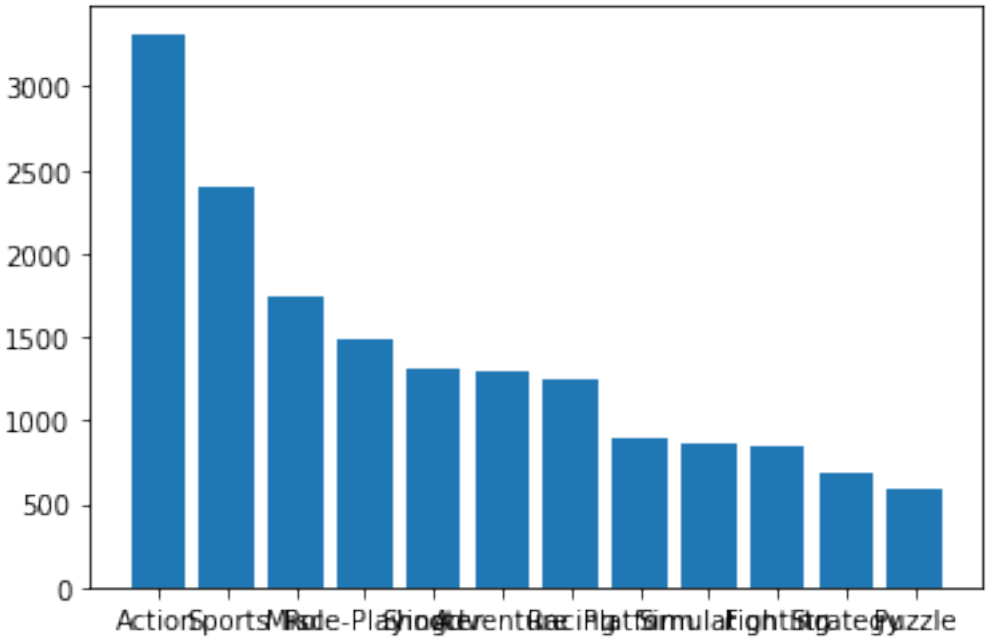
0.1.1 Univariate Analysis - Categorical data

[5]:
```
cat_counts = data['Genre'].value_counts()
cat_counts[:5]
```

[5]:
```
Action          3316
Sports          2400
Misc            1739
Role-Playing    1488
Shooter         1310
Name: Genre, dtype: int64
```

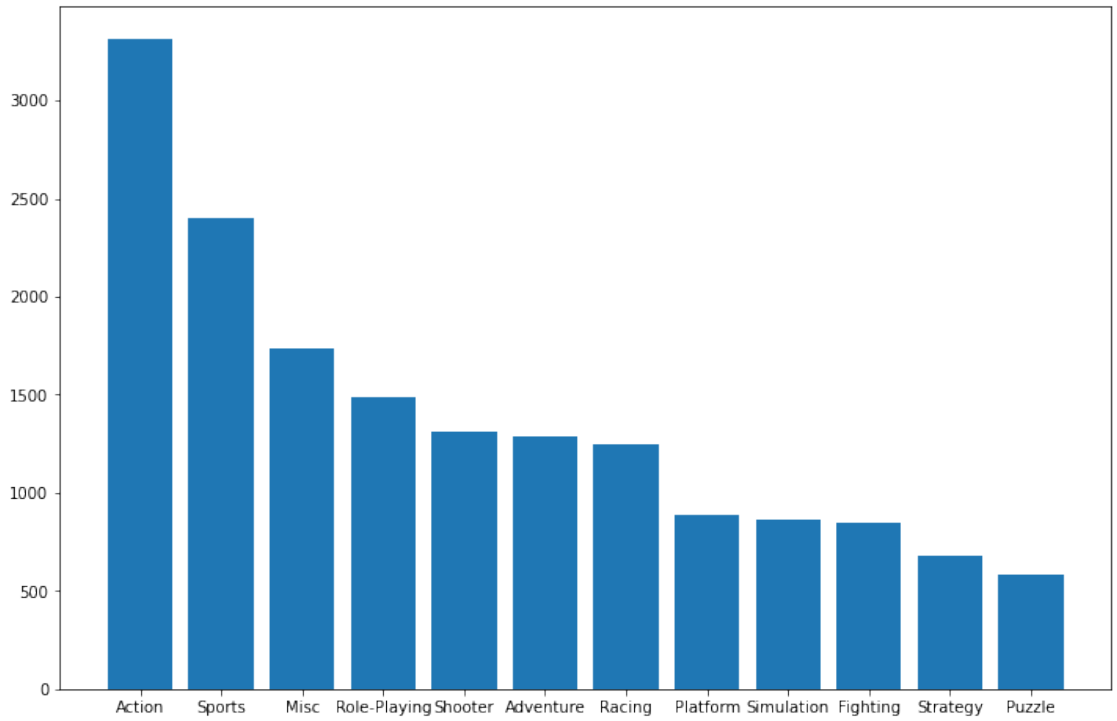
[6]:
```
x_bar = cat_counts.index
y_bar = cat_counts
plt.bar(x_bar, y_bar)
```

[6]:
```
<BarContainer object of 12 artists>
```

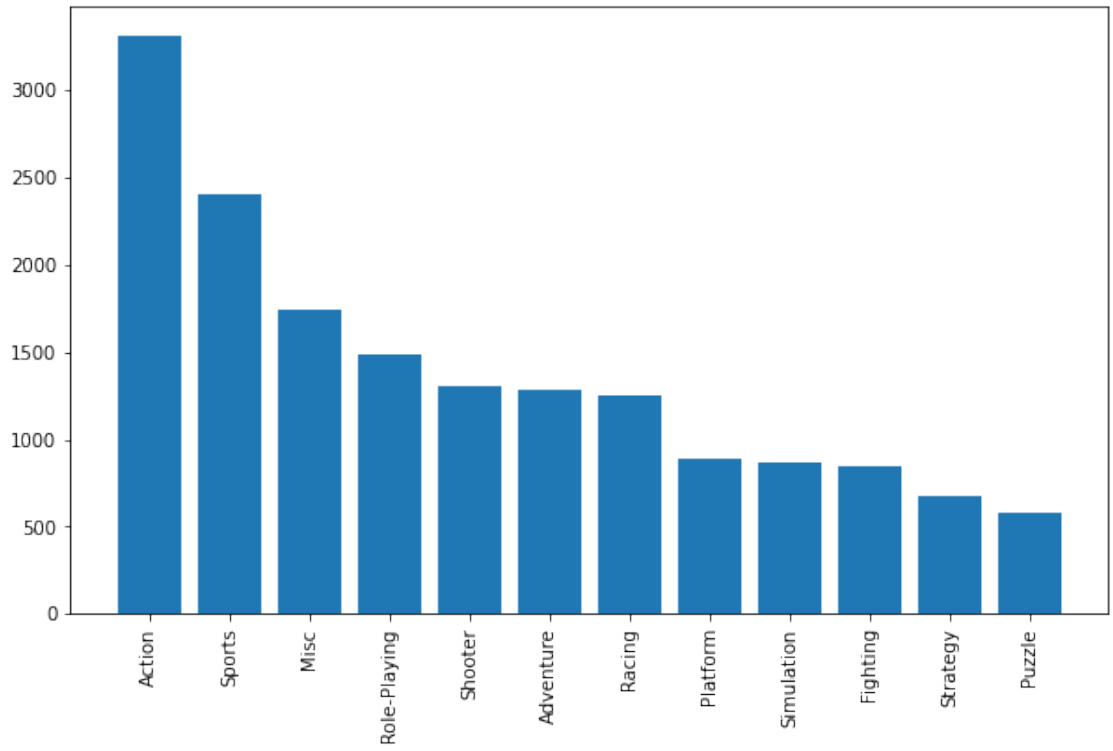


[7]:
```
plt.figure(figsize=(12, 8))
plt.bar(x_bar, y_bar)
```

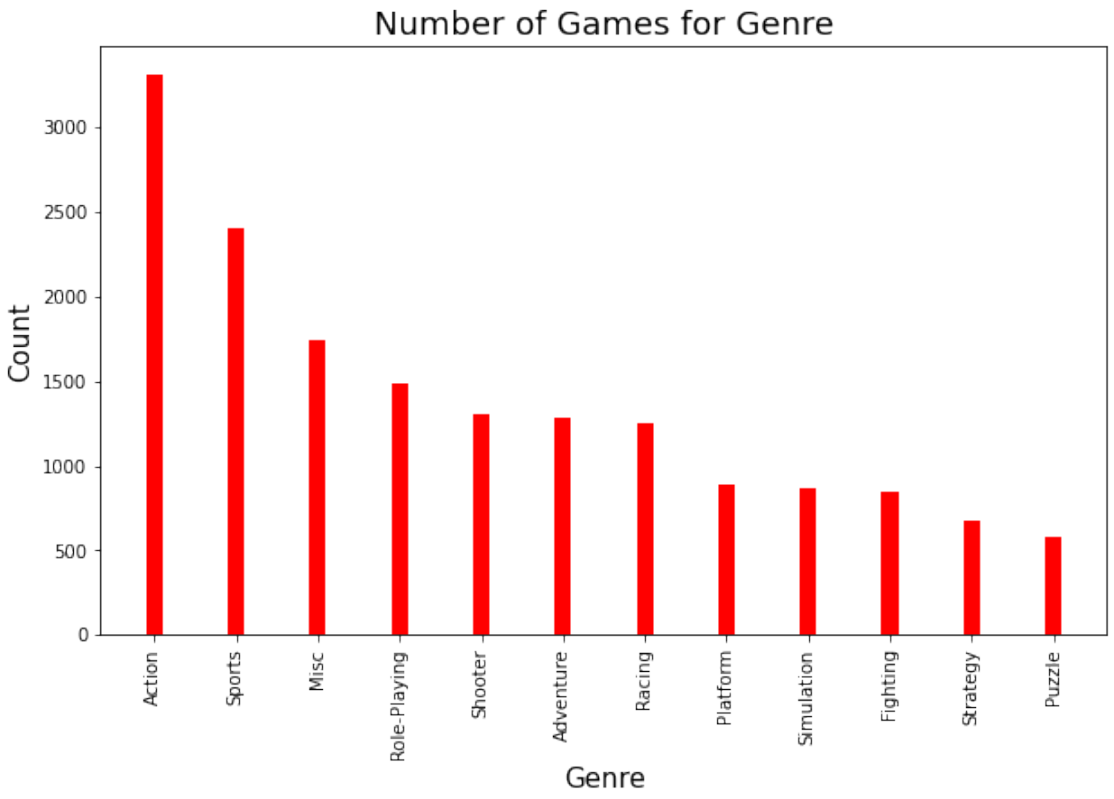
[7]: <BarContainer object of 12 artists>



[8]:
```
plt.figure(figsize=(10, 6))
plt.bar(x_bar, y_bar)
plt.xticks(rotation=90, fontsize=10)
plt.show()
```



[9]:
```
plt.figure(figsize=(10, 6))
plt.bar(x_bar, y_bar, width=0.2, color='red')
plt.xticks(rotation=90, fontsize=10)
plt.xlabel('Genre', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.title('Number of Games for Genre', fontsize=18)
plt.show()
```



[10]:
```
help(plt.bar)
```

```
Help on function bar in module matplotlib.pyplot:

bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)
    Make a bar plot.

    The bars are positioned at *x* with the given *align*\ment. Their
    dimensions are given by *height* and *width*. The vertical baseline
    is *bottom* (default 0).

    Many parameters can take either a single value applying to all bars
    or a sequence of values, one for each bar.

    Parameters
    ----------
    x : float or array-like
        The x coordinates of the bars. See also *align* for the
        alignment of the bars to the coordinates.

    height : float or array-like
        The height(s) of the bars.
```

```
width : float or array-like, default: 0.8
    The width(s) of the bars.

bottom : float or array-like, default: 0
    The y coordinate(s) of the bars bases.

align : {'center', 'edge'}, default: 'center'
    Alignment of the bars to the *x* coordinates:

    - 'center': Center the base on the *x* positions.
    - 'edge': Align the left edges of the bars with the *x* positions.

    To align the bars on the right edge pass a negative *width* and
    ``align='edge'``.

Returns
-------
`.BarContainer`
    Container with all the bars and optionally errorbars.

Other Parameters
----------------
color : color or list of color, optional
    The colors of the bar faces.

edgecolor : color or list of color, optional
    The colors of the bar edges.

linewidth : float or array-like, optional
    Width of the bar edge(s). If 0, don't draw edges.

tick_label : str or list of str, optional
    The tick labels of the bars.
    Default: None (Use default numeric labels.)

xerr, yerr : float or array-like of shape(N,) or shape(2, N), optional
    If not *None*, add horizontal / vertical errorbars to the bar tips.
    The values are +/- sizes relative to the data:

    - scalar: symmetric +/- values for all bars
    - shape(N,): symmetric +/- values for each bar
    - shape(2, N): Separate - and + values for each bar. First row
      contains the lower errors, the second row contains the upper
      errors.
    - *None*: No errorbar. (Default)

    See :doc:`/gallery/statistics/errorbar_features`
```

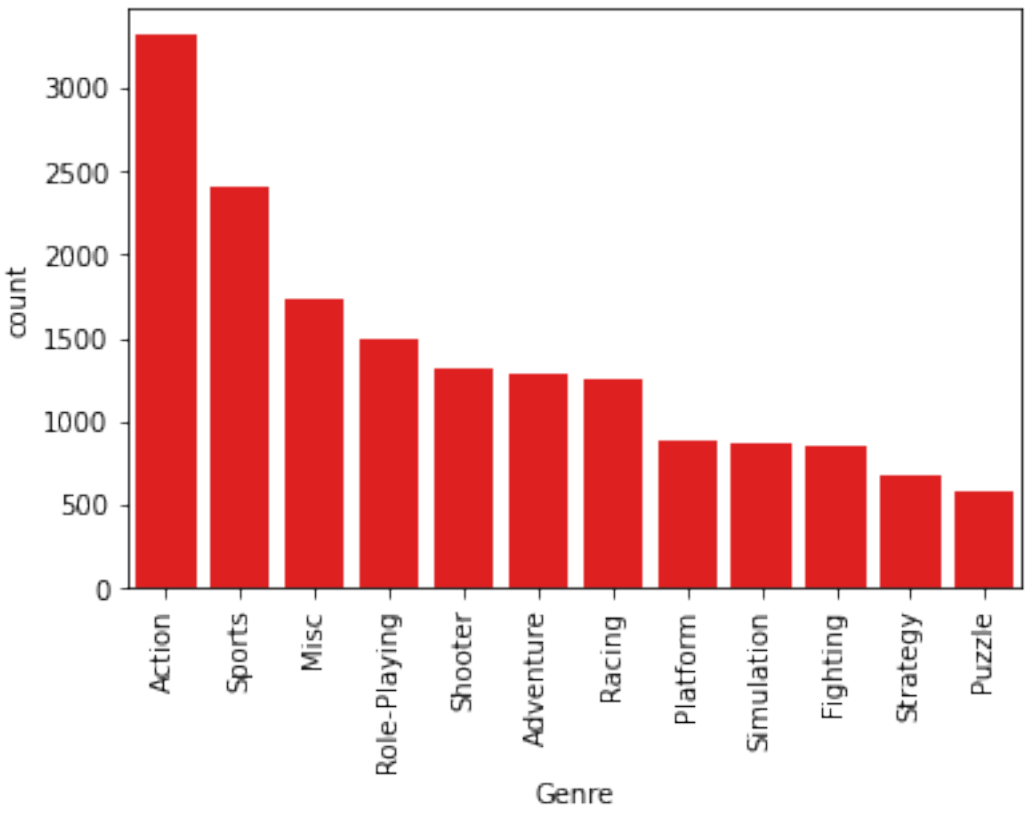
```
        for an example on the usage of ``xerr`` and ``yerr``.

    ecolor : color or list of color, default: 'black'
        The line color of the errorbars.

    capsize : float, default: :rc:`errorbar.capsize`
       The length of the error bar caps in points.

    error_kw : dict, optional
        Dictionary of kwargs to be passed to the `~.Axes.errorbar`
        method. Values of *ecolor* or *capsize* defined here take
        precedence over the independent kwargs.

    log : bool, default: False
        If *True*, set the y-axis to be log scale.

    data : indexable object, optional
        If given, all parameters also accept a string ``s``, which is
        interpreted as ``data[s]`` (unless this raises an exception).

    **kwargs : `.Rectangle` properties

    Properties:
        agg_filter: a filter function, which takes a (m, n, 3) float array and a
dpi value, and returns a (m, n, 3) array
        alpha: scalar or None
        angle: unknown
        animated: bool
        antialiased or aa: bool or None
        bounds: (left, bottom, width, height)
        capstyle: `.CapStyle` or {'butt', 'projecting', 'round'}
        clip_box: `.Bbox`
        clip_on: bool
        clip_path: Patch or (Path, Transform) or None
        color: color
        edgecolor or ec: color or None
        facecolor or fc: color or None
        figure: `.Figure`
        fill: bool
        gid: str
        hatch: {'/', '\\', '|', '-', '+', 'x', 'o', 'O', '.', '*'}
        height: unknown
        in_layout: bool
        joinstyle: `.JoinStyle` or {'miter', 'round', 'bevel'}
        label: object
        linestyle or ls: {'-', '--', '-.', ':', '', (offset, on-off-seq), …}
        linewidth or lw: float or None
        path_effects: `.AbstractPathEffect`
```

```
        picker: None or bool or float or callable
        rasterized: bool
        sketch_params: (scale: float, length: float, randomness: float)
        snap: bool or None
        transform: `.Transform`
        url: str
        visible: bool
        width: unknown
        x: unknown
        xy: (float, float)
        y: unknown
        zorder: float

    See Also
    --------
    barh : Plot a horizontal bar plot.

    Notes
    -----
    Stacked bars can be achieved by passing individual *bottom* values per
    bar. See :doc:`/gallery/lines_bars_and_markers/bar_stacked`.
```

[11]:
```
cat_counts.index
```

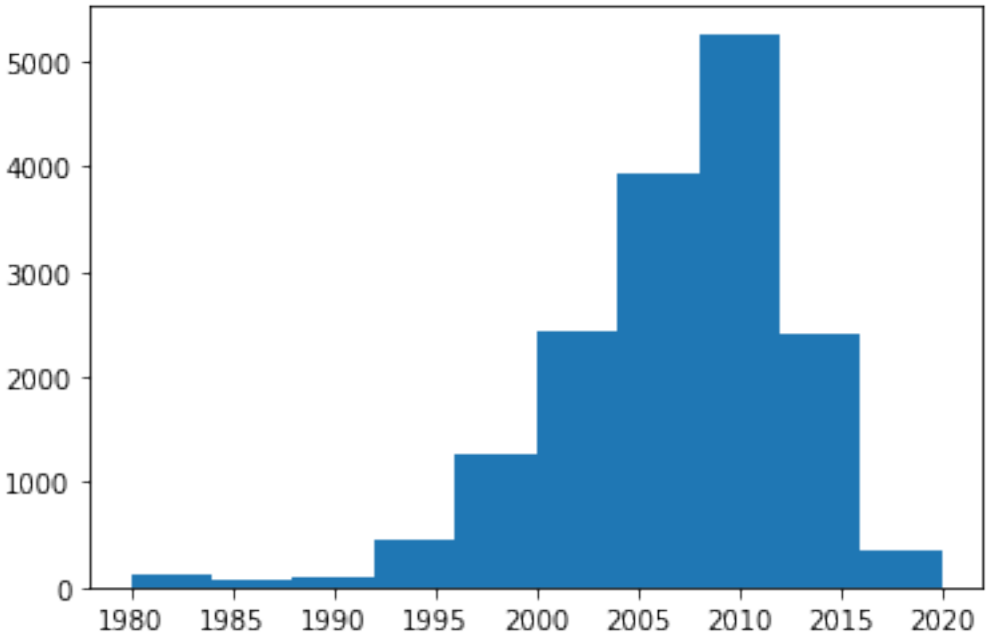
[11]:
```
Index(['Action', 'Sports', 'Misc', 'Role-Playing', 'Shooter', 'Adventure',
       'Racing', 'Platform', 'Simulation', 'Fighting', 'Strategy', 'Puzzle'],
      dtype='object')
```

[12]:
```
sns.countplot(data=data, x='Genre', order=cat_counts.index, color='red')
plt.xticks(rotation=90)
plt.show()
```

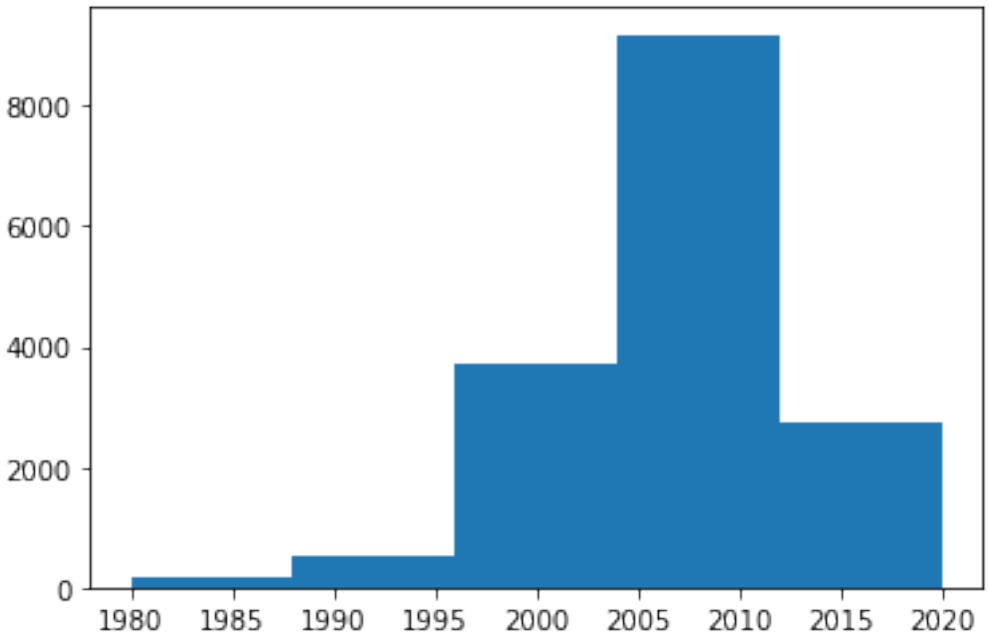


0.1.2 Univariate Analysis- Numerical Data

[13]:
```
plt.hist(data['Year'])
plt.show()
```

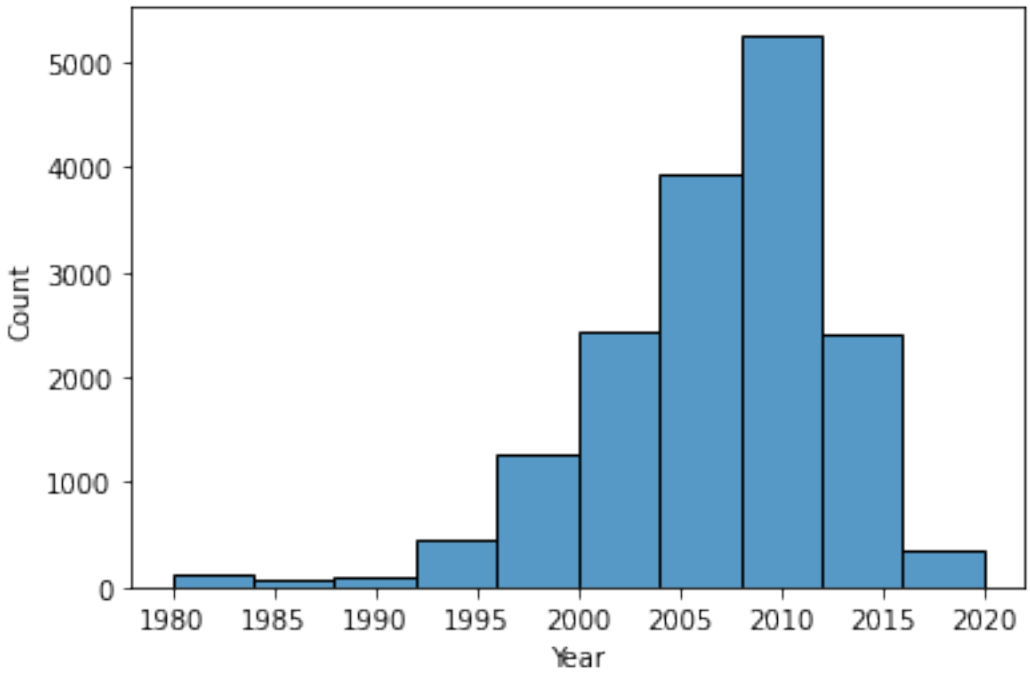


[14]:
```
plt.hist(data['Year'], bins=5)
plt.show()
```



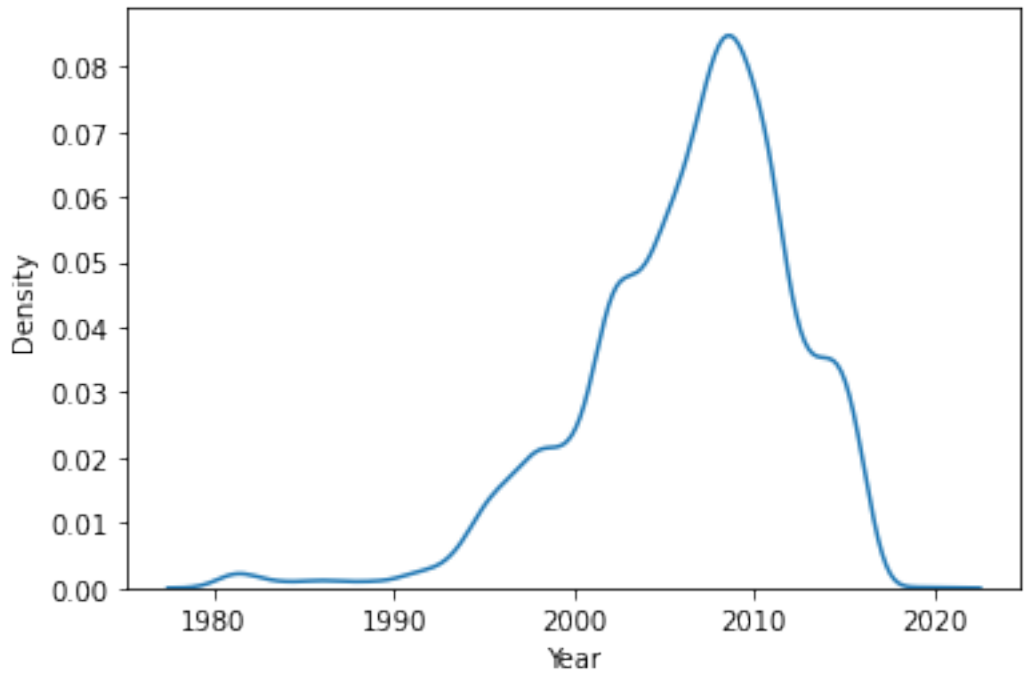
[15]:
```
sns.histplot(data['Year'], bins=10)
```

[15]: <AxesSubplot:xlabel='Year', ylabel='Count'>



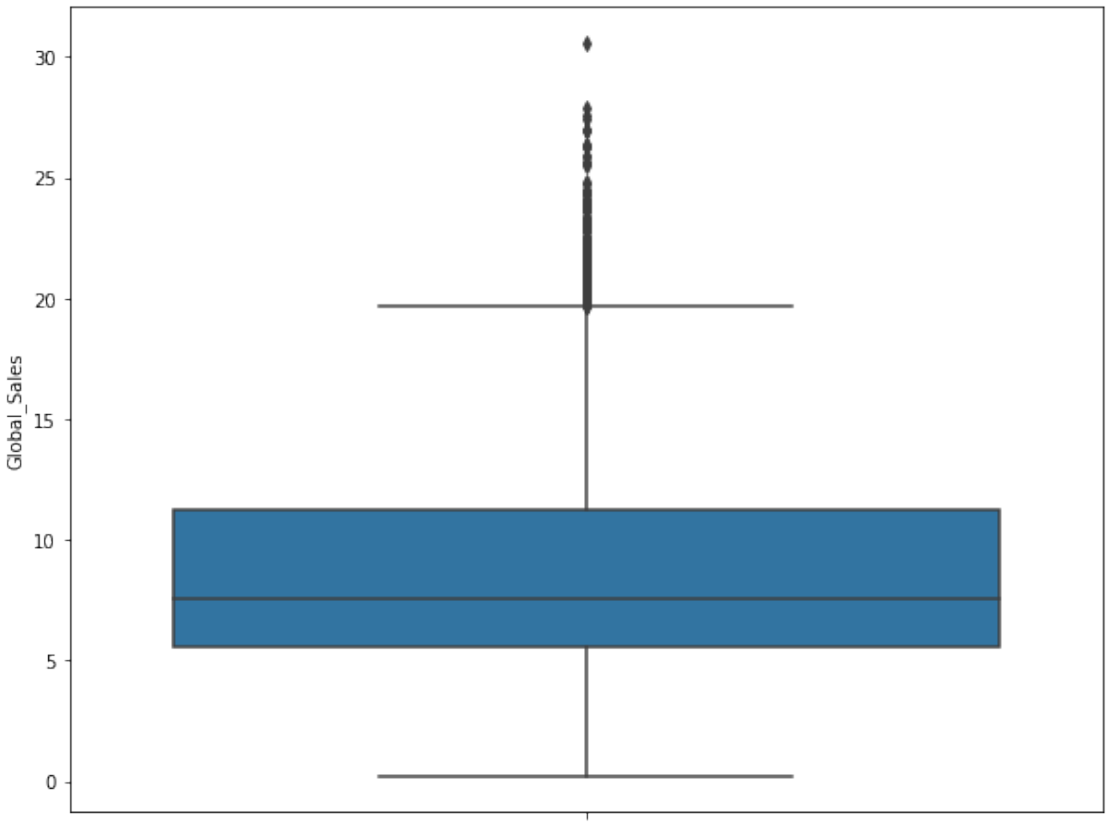
[16]:
```
sns.kdeplot(data['Year'])
```

[16]: <AxesSubplot:xlabel='Year', ylabel='Density'>



[17]:
```
plt.figure(figsize=(10, 8))
sns.boxplot(y=data['Global_Sales'])
```

[17]:
```
<AxesSubplot:ylabel='Global_Sales'>
```



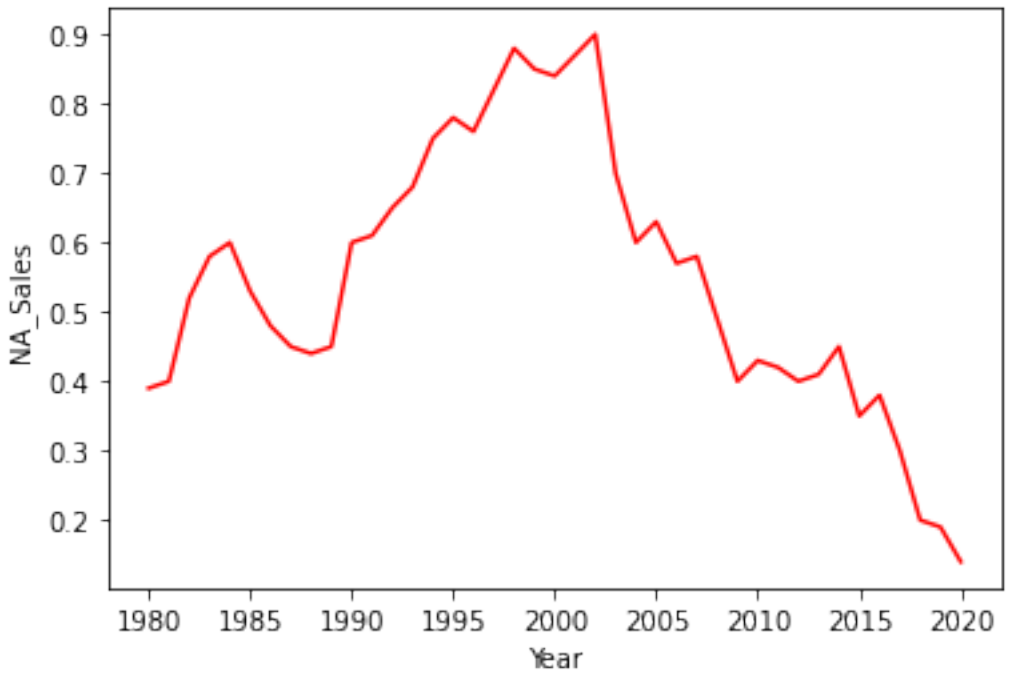
0.2 Bi Variate Analysis - (Numerical - Numerical)

[18]:
```
data['Name'].value_counts()
```

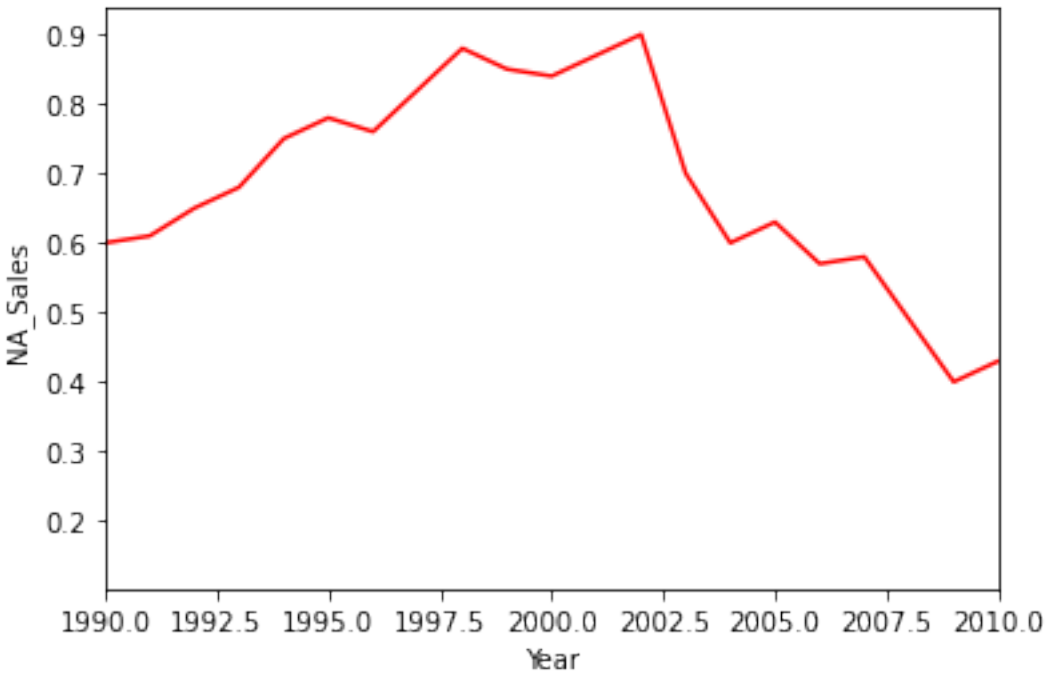
[18]:
```
Ice Hockey                          41
Baseball                            17
Need for Speed: Most Wanted         12
Ratatouille                          9
FIFA 14                              9
                                    ..
Indy 500                             1
Indy Racing 2000                     1
Indycar Series 2005                  1
```

```
inFAMOUS                                                1
Zyuden Sentai Kyoryuger: Game de Gaburincho!!           1
Name: Name, Length: 11493, dtype: int64
```

[20]:
```
ih = data.loc[data['Name']=='Ice Hockey']
sns.lineplot(data=ih, x='Year', y='NA_Sales', color='red')
plt.show()
```

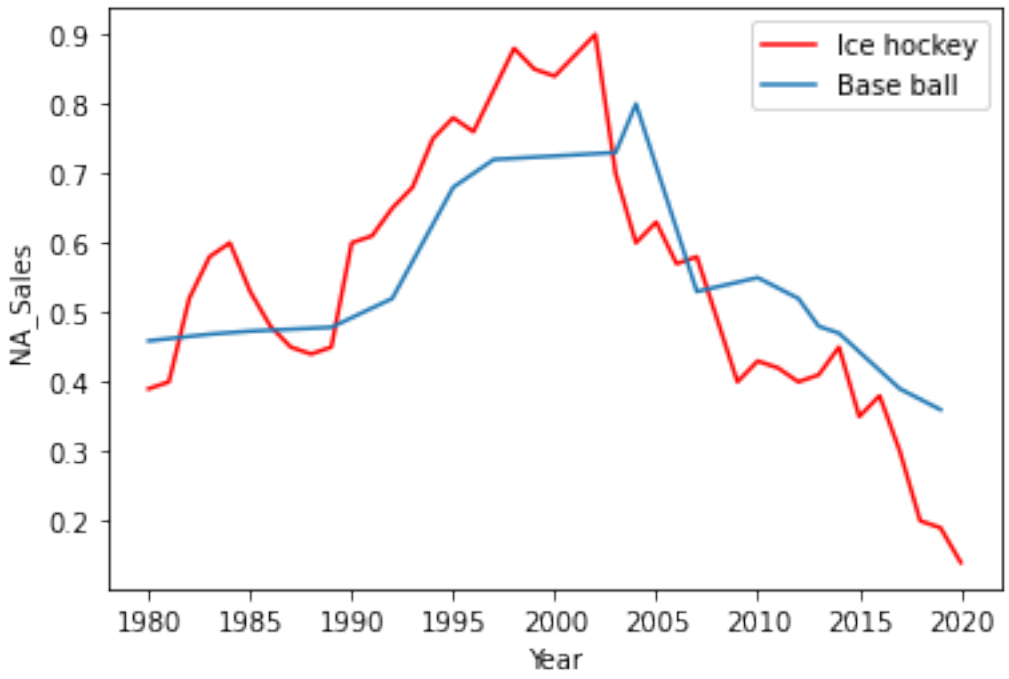


[21]:
```
ih = data.loc[data['Name']=='Ice Hockey']
sns.lineplot(data=ih, x='Year', y='NA_Sales', color='red')
plt.xlim(left=1990, right=2010)
plt.show()
```

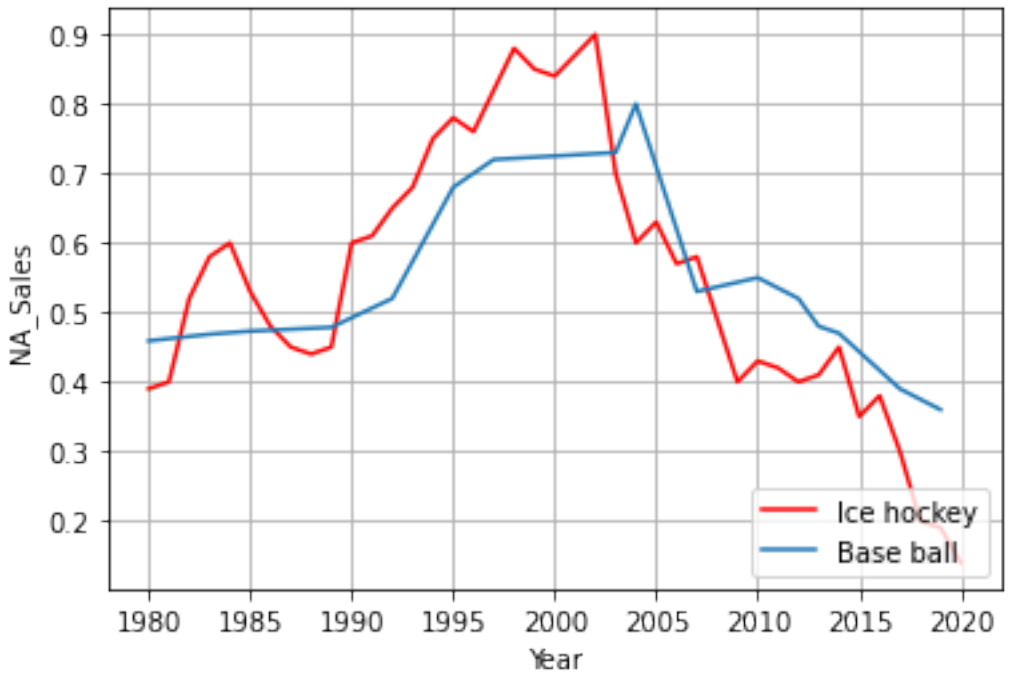


[ ]:
```
sns.lineplot?
```

[22]:
```
ih = data.loc[data['Name']=='Ice Hockey']
baseball = data.loc[data['Name']=='Baseball']
sns.lineplot(data=ih, x='Year', y='NA_Sales', color='red' )
sns.lineplot(data=baseball, x='Year', y='NA_Sales' )
plt.legend(['Ice hockey', 'Base ball'])
plt.show()
```



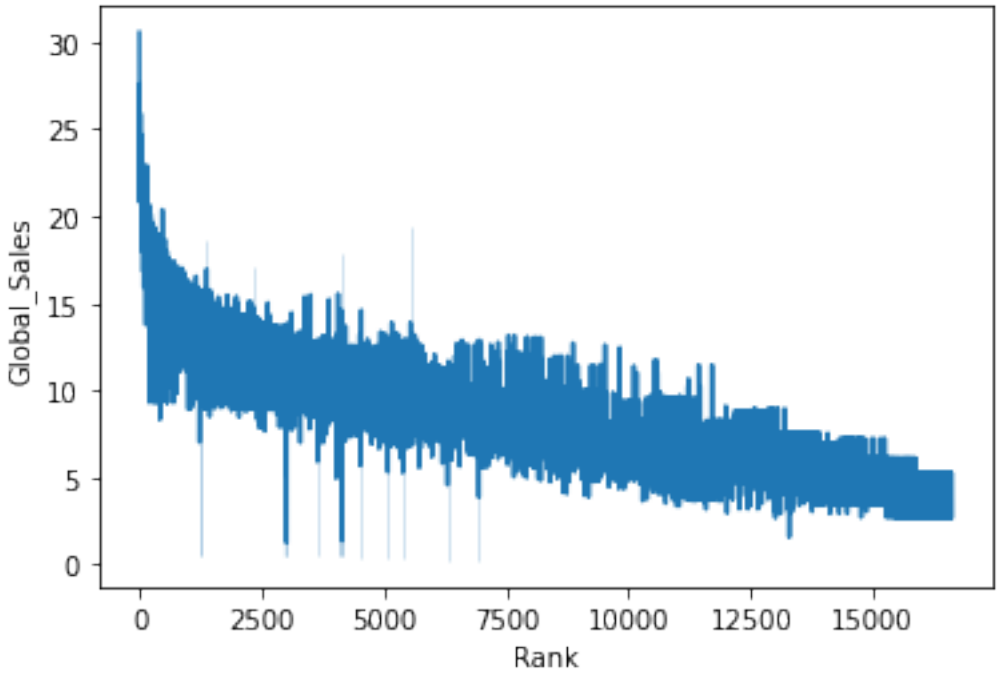
[26]:
```
ih = data.loc[data['Name']=='Ice Hockey']
baseball = data.loc[data['Name']=='Baseball']
sns.lineplot(data=ih, x='Year', y='NA_Sales', color='red' )
sns.lineplot(data=baseball, x='Year', y='NA_Sales' )
plt.legend(['Ice hockey', 'Base ball'], loc='lower right')
plt.grid()
plt.show()
```



0.2.1 Scatter Plot

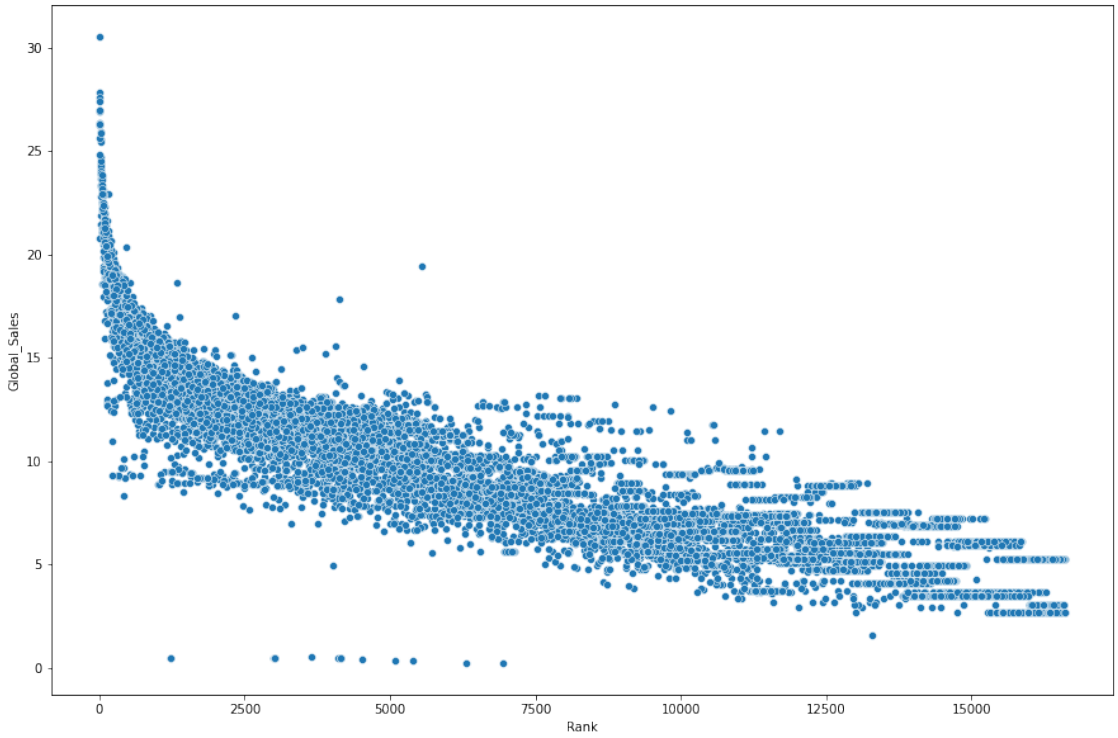
[27]:
```
sns.lineplot(data=data, x='Rank', y='Global_Sales')
```

[27]:
```
<AxesSubplot:xlabel='Rank', ylabel='Global_Sales'>
```



[30]:
```
plt.figure(figsize=(15, 10))
sns.scatterplot(data=data, x='Rank', y='Global_Sales')
```

[30]: <AxesSubplot:xlabel='Rank', ylabel='Global_Sales'>



[31]:
```
data
```

[31]:

	Rank	Name	Platform	\
0	2061.0	1942.0	NES	
1	9137.0	¡Shin Chan Flipa en colores!	DS	
2	14279.0	.hack: Sekai no Mukou ni + Versus	PS3	
3	8359.0	.hack//G.U. Vol.1//Rebirth	PS2	
4	7109.0	.hack//G.U. Vol.2//Reminisce	PS2	
…	…	…	…	
16647	7925.0	Zumba Fitness Rush	X360	
16648	6279.0	Zumba Fitness: World Party	Wii	
16649	6977.0	Zumba Fitness: World Party	XOne	
16650	15422.0	Zwei!!	PSP	
16651	12919.0	Zyuden Sentai Kyoryuger: Game de Gaburincho!!	3DS	

	Year	Genre	Publisher	NA_Sales	EU_Sales	\
0	1985.0	Shooter	Capcom	4.569217	3.033887	
1	2007.0	Platform	505 Games	2.076955	1.493442	
2	2012.0	Action	Namco Bandai Games	1.145709	1.762339	
3	2006.0	Role-Playing	Namco Bandai Games	2.031986	1.389856	
4	2006.0	Role-Playing	Namco Bandai Games	2.792725	2.592054	
…	…	…	…	…	…	
16647	2012.0	Sports	505 Games	4.409308	3.167419	

```
16648  2013.0          Misc  Majesco Entertainment  3.033887  2.792725
16649  2013.0          Misc  Majesco Entertainment  3.228043  2.004268
16650  2008.0  Role-Playing     Falcom Corporation  1.087977  0.592445
16651  2013.0        Action     Namco Bandai Games  1.081046  1.714664

       JP_Sales  Other_Sales  Global_Sales
0      3.439352     1.991671     12.802935
1      3.033887     0.394830      7.034163
2      1.493442     0.408693      4.982552
3      3.228043     0.394830      7.226880
4      1.440483     1.493442      8.363113
…         …            …            …
16647  4.168474     1.087977     13.053204
16648  1.596852     1.493442      8.878837
16649  1.833151     1.087977      7.954274
16650  1.087977     0.394830      3.509168
16651  2.004268     0.394830      5.132196

[16652 rows x 11 columns]
```

[35]:
```
top3_pub = data['Publisher'].value_counts().index[:3]
top3_gen = data['Genre'].value_counts().index[:3]
top3_plat = data['Platform'].value_counts().index[:3]
top3_data = data.loc[(data['Publisher'].isin(top3_pub)) & (data['Genre'].
 ↪isin(top3_gen)) & (data['Platform'].isin(top3_plat))]
top3_data
```

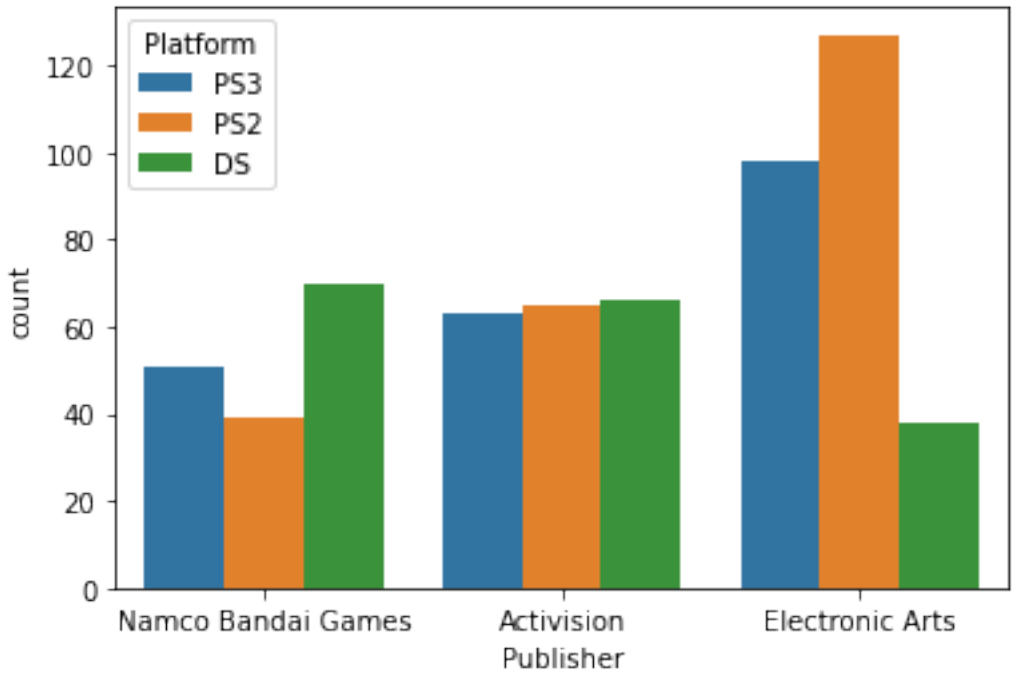
[35]:
```
          Rank                                                Name Platform  \
2      14279.0                  .hack: Sekai no Mukou ni + Versus      PS3
13      2742.0                                       [Prototype 2]      PS3
16      1604.0                                         [Prototype]      PS3
19      1741.0                             007: Quantum of Solace      PS3
21      4501.0                             007: Quantum of Solace      PS2
…         …                                                   …      …
16438  14938.0  Yes! Precure 5 Go Go Zenin Shu Go! Dream Festival       DS
16479  10979.0                           Young Justice: Legacy      PS3
16601  11802.0                       ZhuZhu Pets: Quest for Zhu       DS
16636   9196.0                         Zoobles! Spring to Life!       DS
16640   9816.0                                             Zubo       DS

         Year   Genre           Publisher  NA_Sales  EU_Sales  JP_Sales  \
2      2012.0  Action  Namco Bandai Games  1.145709  1.762339  1.493442
13     2012.0  Action          Activision  3.978349  3.727034  0.848807
16     2009.0  Action          Activision  4.569217  4.108402  1.187272
19     2008.0  Action          Activision  4.156030  4.346074  1.087977
21     2008.0  Action          Activision  3.228043  2.738800  2.585598
…         …       …                   …         …         …         …
```

```
16438  2008.0  Action  Namco Bandai Games  1.087977  0.592445  1.087977
16479  2013.0  Action  Namco Bandai Games  2.186589  1.087977  3.409089
16601  2011.0    Misc          Activision  2.340740  1.525543  3.103825
16636  2011.0    Misc          Activision  2.697415  1.087977  2.760718
16640  2008.0    Misc     Electronic Arts  2.592054  1.493442  1.493442

       Other_Sales  Global_Sales
2         0.408693      4.982552
13        2.792725     11.447989
16        3.339269     13.181205
19        3.390562     12.980643
21        3.652926     11.780257
…              …             …
16438     0.394830      3.509168
16479     0.394830      7.359902
16601     0.394830      7.372592
16636     0.394830      6.915540
16640     0.394830      5.969572

[617 rows x 11 columns]
```

0.2.2 Dodged Count plot

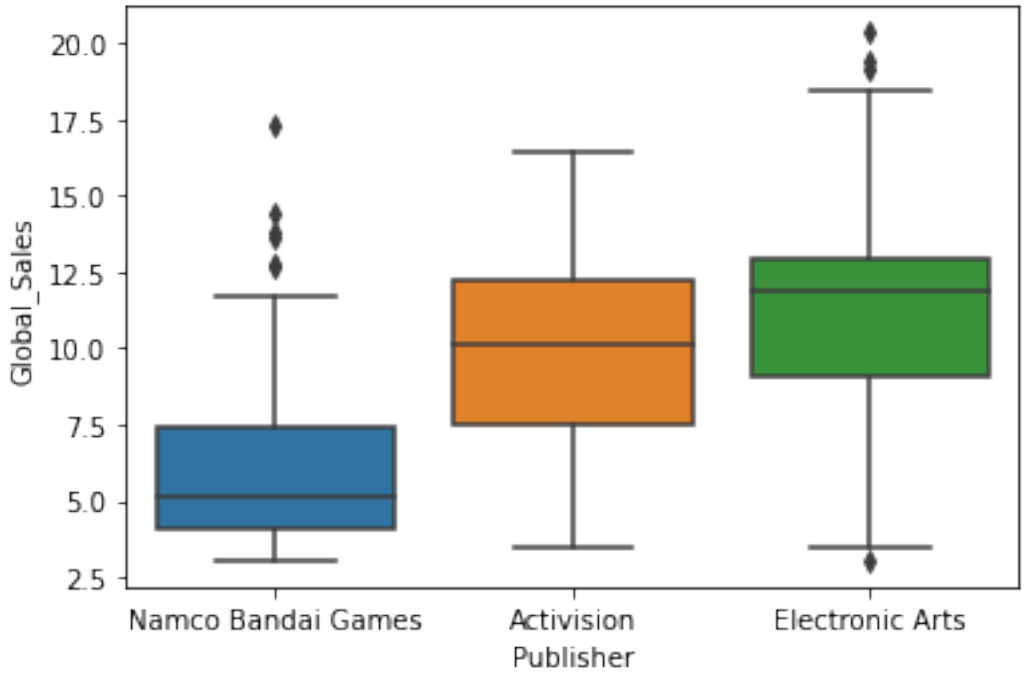
[36]:
```
sns.countplot(data=top3_data, x='Publisher', hue='Platform')
```

[36]:
```
<AxesSubplot:xlabel='Publisher', ylabel='count'>
```



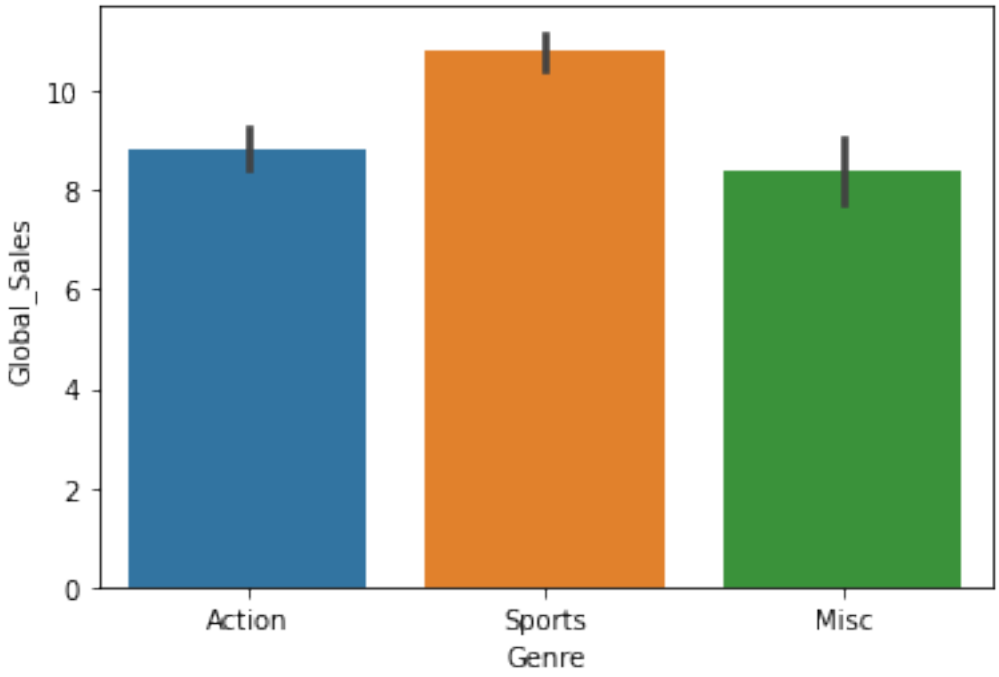
[37]:
```
#What is the distribution of sales for the top3 publishers


sns.boxplot(data=top3_data, x='Publisher', y='Global_Sales')
```

[37]: <AxesSubplot:xlabel='Publisher', ylabel='Global_Sales'>



[41]:
```
# Which genre give higher average global sales

sns.barplot(data=top3_data, x='Genre', y='Global_Sales', estimator=np.mean)
```

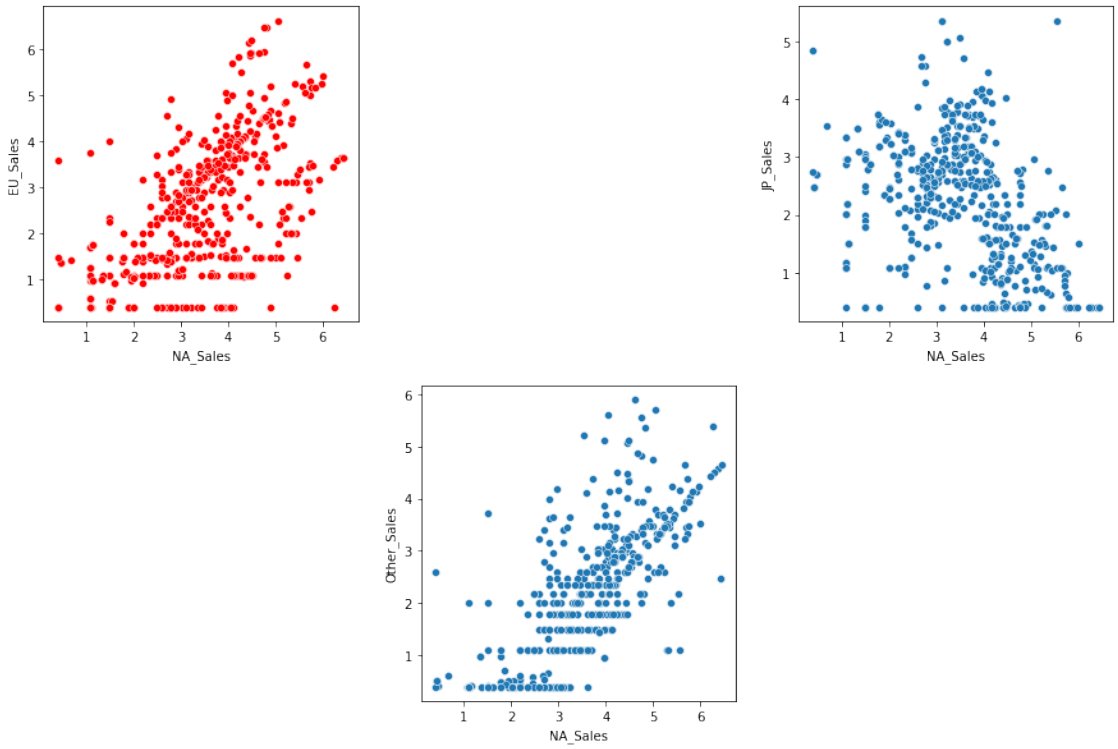
[41]: <AxesSubplot:xlabel='Genre', ylabel='Global_Sales'>



0.2.3 Sub plots

[44]:
```
plt.figure(figsize=(15, 10))
plt.subplot(2, 3, 1)
sns.scatterplot(data=top3_data, x='NA_Sales', y='EU_Sales', color='red')

plt.subplot(2, 3, 3)
sns.scatterplot(data=top3_data, x='NA_Sales', y='JP_Sales')

plt.subplot(2, 3, 5)
sns.scatterplot(data=top3_data, x='NA_Sales', y='Other_Sales')
```

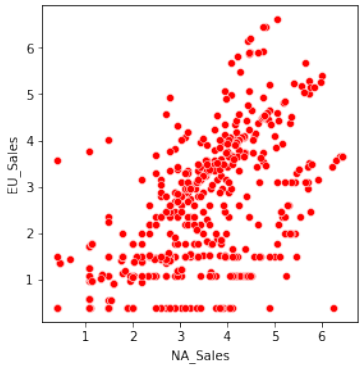
[44]:
```
<AxesSubplot:xlabel='NA_Sales', ylabel='Other_Sales'>
```

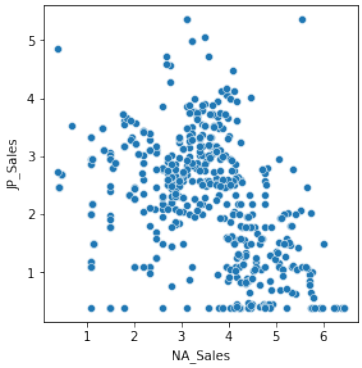


[45]:
```
plt.figure(figsize=(15, 10))
plt.subplot(2, 3, 1)
sns.scatterplot(data=top3_data, x='NA_Sales', y='EU_Sales', color='red')

plt.subplot(2, 3, 3)
sns.scatterplot(data=top3_data, x='NA_Sales', y='JP_Sales')

plt.subplot(2, 3, 8)
sns.scatterplot(data=top3_data, x='NA_Sales', y='Other_Sales')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [45], in <cell line: 8>()
      5 plt.subplot(2, 3, 3)
      6 sns.scatterplot(data=top3_data, x='NA_Sales', y='JP_Sales')
----> 8 plt.subplot(2, 3, 8)
      9 sns.scatterplot(data=top3_data, x='NA_Sales', y='Other_Sales')

File /usr/local/lib/python3.9/site-packages/matplotlib/pyplot.py:1268, in␣
 ↪subplot(*args, **kwargs)
   1265 fig = gcf()
   1267 # First, search for an existing subplot with a matching spec.
-> 1268 key = SubplotSpec._from_subplot_args(fig, args)
```

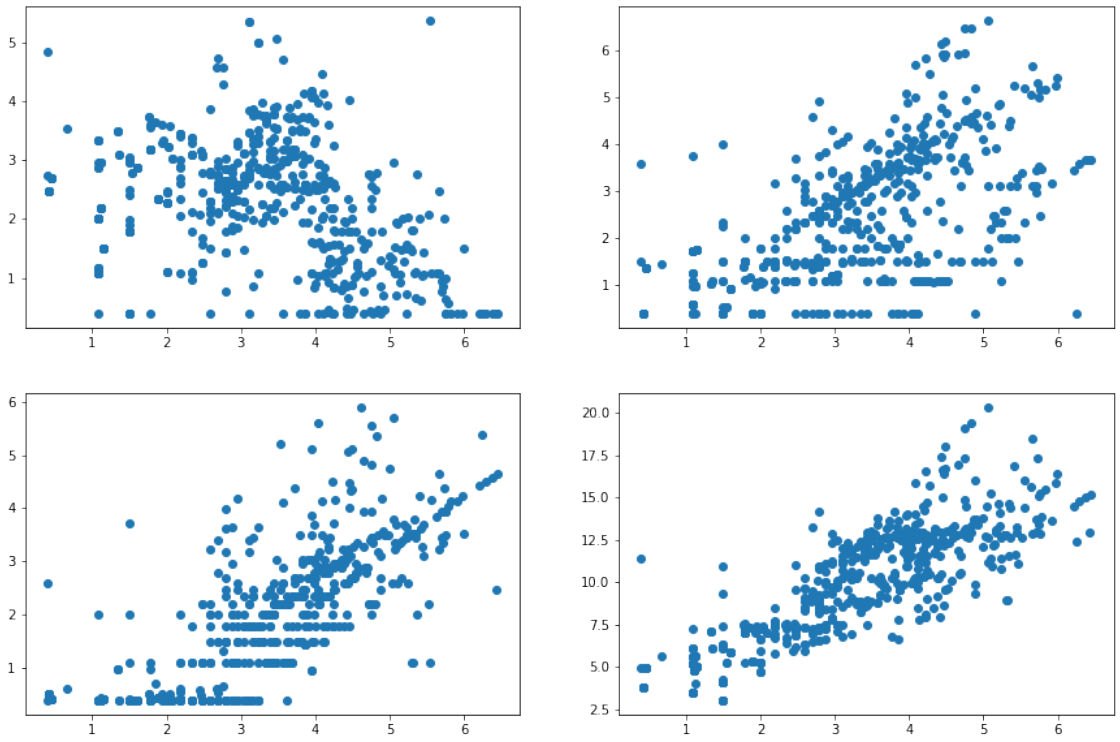
```
   1270 for ax in fig.axes:
   1271     # if we found an axes at the position sort out if we can re-use it
   1272     if hasattr(ax, 'get_subplotspec') and ax.get_subplotspec() == key:
   1273         # if the user passed no kwargs, re-use

File /usr/local/lib/python3.9/site-packages/matplotlib/gridspec.py:608, in␣
 ↪SubplotSpec._from_subplot_args(figure, args)
    606 else:
    607     if not isinstance(num, Integral) or num < 1 or num > rows*cols:
--> 608         raise ValueError(
    609             f"num must be 1 <= num <= {rows*cols}, not {num!r}")
    610     i = j = num
    611 return gs[i-1:j]

ValueError: num must be 1 <= num <= 6, not 8
```





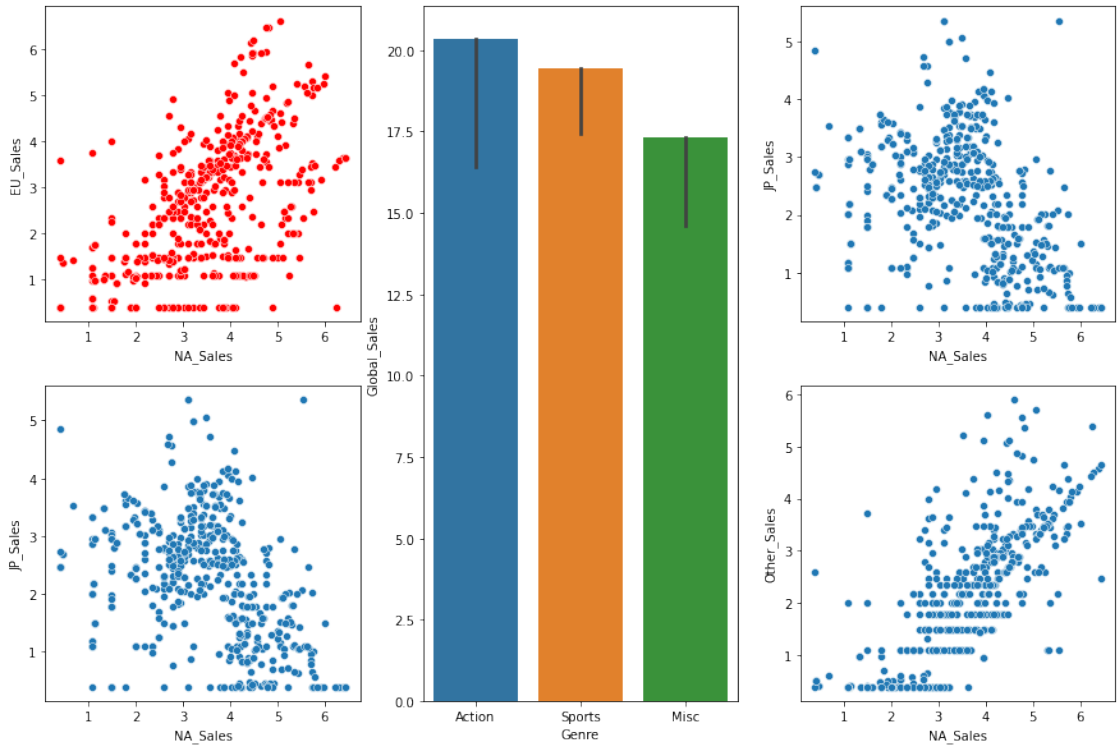
[48]:
```
fig, ax = plt.subplots(2, 2, figsize=(15, 10))
ax[0,0].scatter(data=top3_data, x='NA_Sales', y='JP_Sales')
ax[0,1].scatter(data=top3_data, x='NA_Sales', y='EU_Sales')
ax[1,0].scatter(data=top3_data, x='NA_Sales', y='Other_Sales')
ax[1,1].scatter(data=top3_data, x='NA_Sales', y='Global_Sales')
```

[48]:
```
<matplotlib.collections.PathCollection at 0x12cf3d190>
```



[52]:
```
plt.figure(figsize=(15, 10))
plt.subplot(2, 3, 1)
sns.scatterplot(data=top3_data, x='NA_Sales', y='EU_Sales', color='red')

plt.subplot(2, 3, 3)
sns.scatterplot(data=top3_data, x='NA_Sales', y='JP_Sales')

plt.subplot(2, 3, 6)
sns.scatterplot(data=top3_data, x='NA_Sales', y='Other_Sales')


plt.subplot(2, 3, 4)
sns.scatterplot(data=top3_data, x='NA_Sales', y='JP_Sales')


plt.subplot(1, 3, 2)
sns.barplot(data=top3_data, x='Genre', y='Global_Sales', estimator=np.max)
```

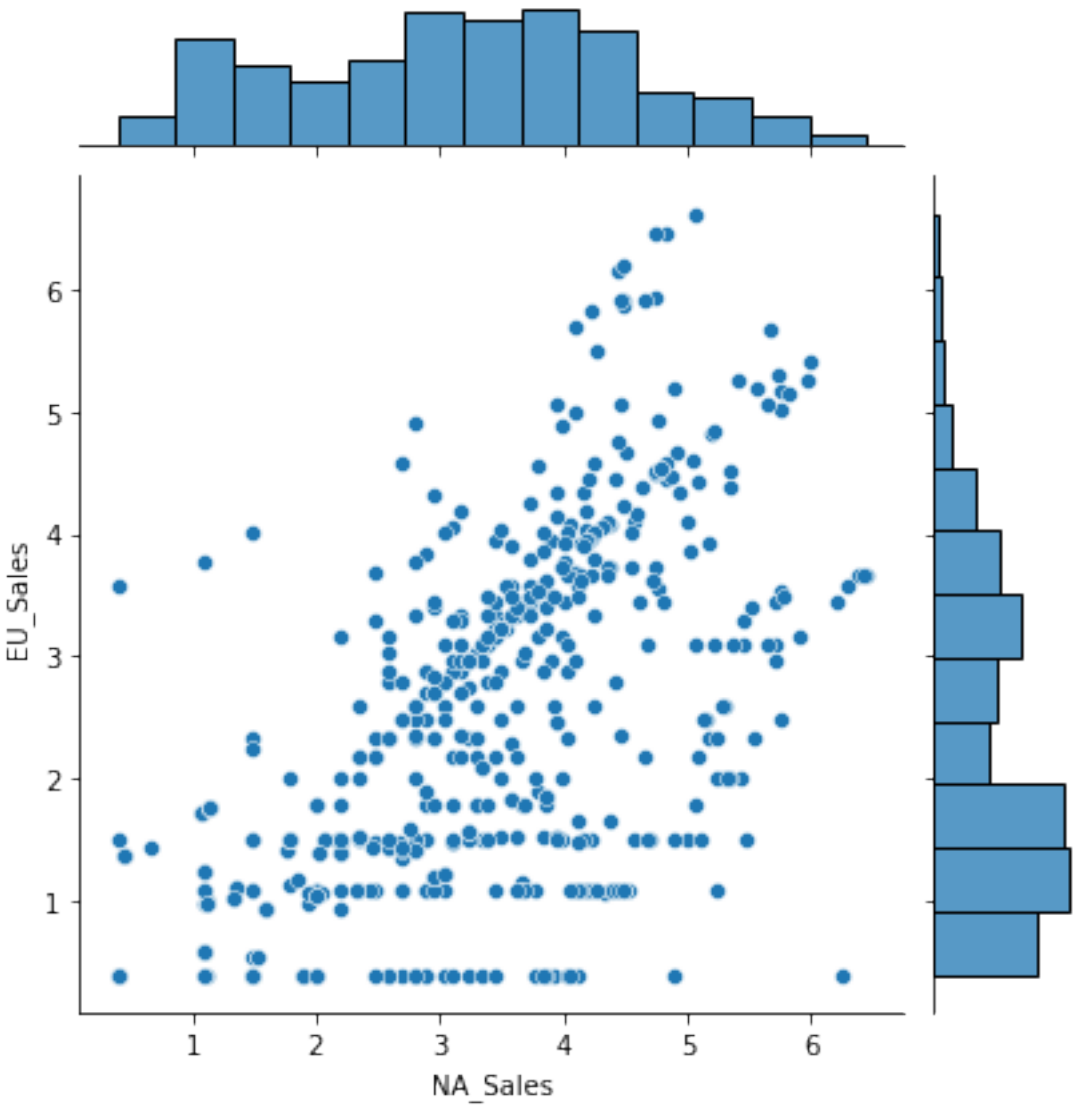
[52]:
```
<AxesSubplot:xlabel='Genre', ylabel='Global_Sales'>
```



0.3 Joint Plot

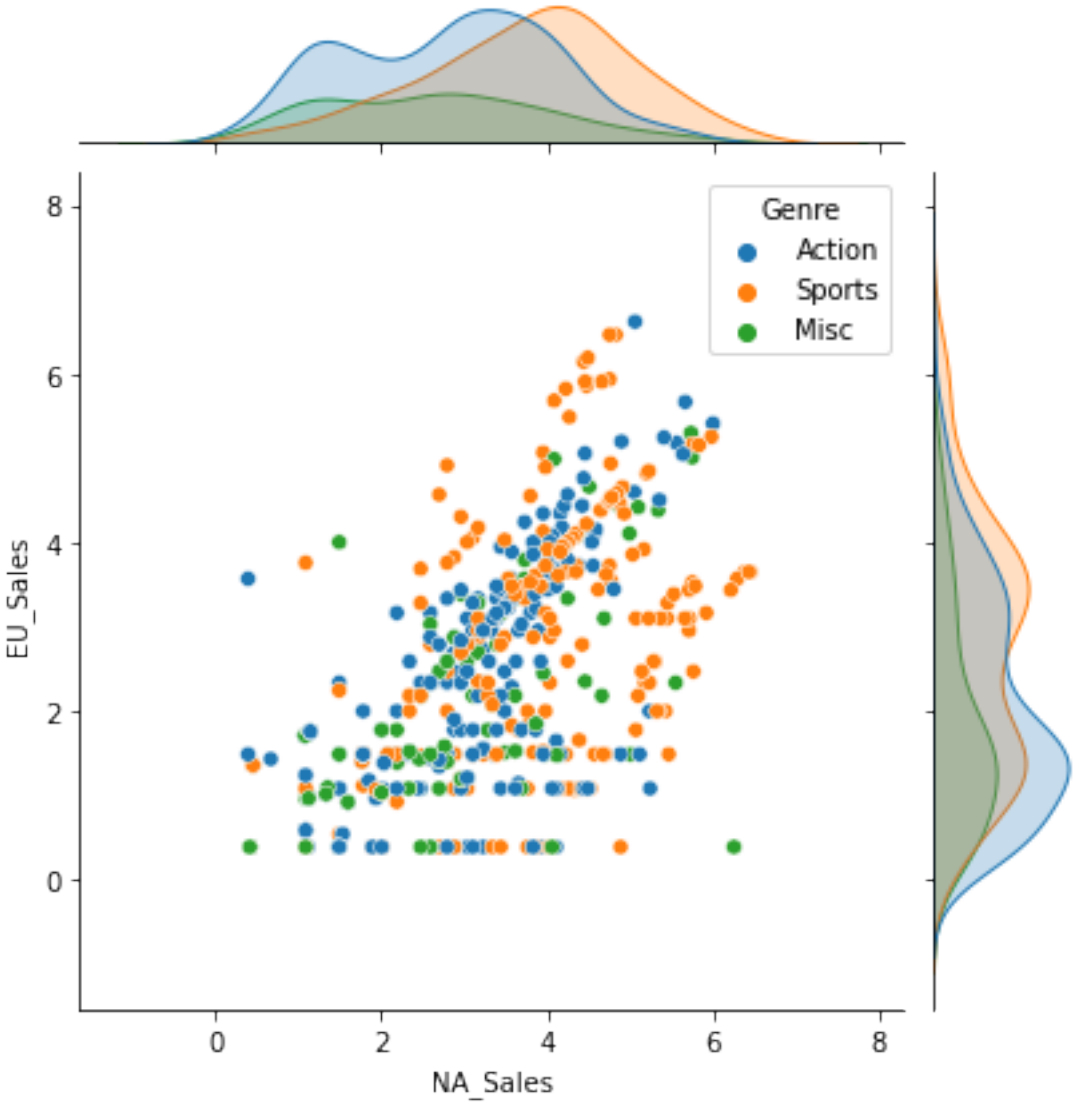
[53]:
```
sns.jointplot(data=top3_data, x='NA_Sales', y='EU_Sales')
```

[53]: <seaborn.axisgrid.JointGrid at 0x12e304280>



[54]:
```
sns.jointplot(data=top3_data, x='NA_Sales', y='EU_Sales', hue='Genre')
```

[54]: <seaborn.axisgrid.JointGrid at 0x12e33af70>



0.3.1 Pair Plots

[56]:
```
sns.pairplot(data=top3_data, hue='Genre')
```

[56]:
```
<seaborn.axisgrid.PairGrid at 0x12ecb0610>
```

[57]:
```
top3_data.corr()
```

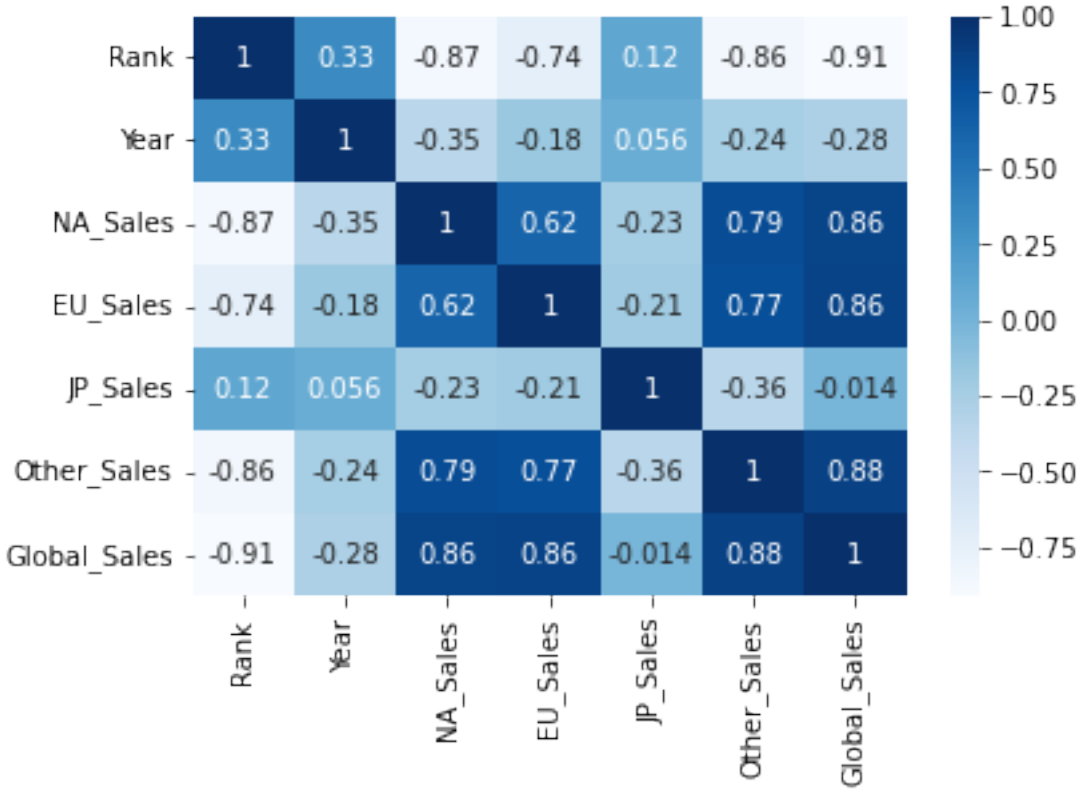
[57]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
Rank	1.000000	0.328705	-0.873726	-0.735711	0.115459	-0.857567	-0.911721
Year	0.328705	1.000000	-0.354256	-0.178026	0.055864	-0.239876	-0.280351
NA_Sales	-0.873726	-0.354256	1.000000	0.617483	-0.233315	0.794353	0.856300
EU_Sales	-0.735711	-0.178026	0.617483	1.000000	-0.208249	0.771105	0.864147
JP_Sales	0.115459	0.055864	-0.233315	-0.208249	1.000000	-0.355825	-0.014193
Other_Sales	-0.857567	-0.239876	0.794353	0.771105	-0.355825	1.000000	0.878816
Global_Sales	-0.911721	-0.280351	0.856300	0.864147	-0.014193	0.878816	1.000000

```
EU_Sales          0.864147
JP_Sales         -0.014193
Other_Sales       0.878816
Global_Sales      1.000000
```

[63]:
```
sns.heatmap(top3_data.corr(), annot=True, cmap='Blues')
```

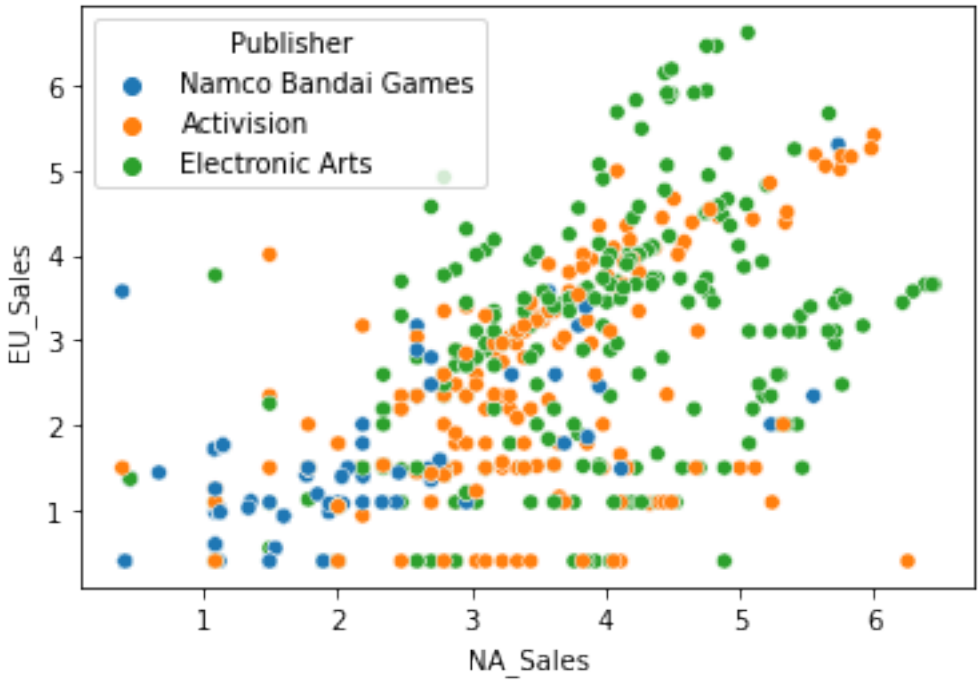
[63]:
```
<AxesSubplot:>
```



0.3.2 MultiVariate Data visualisation

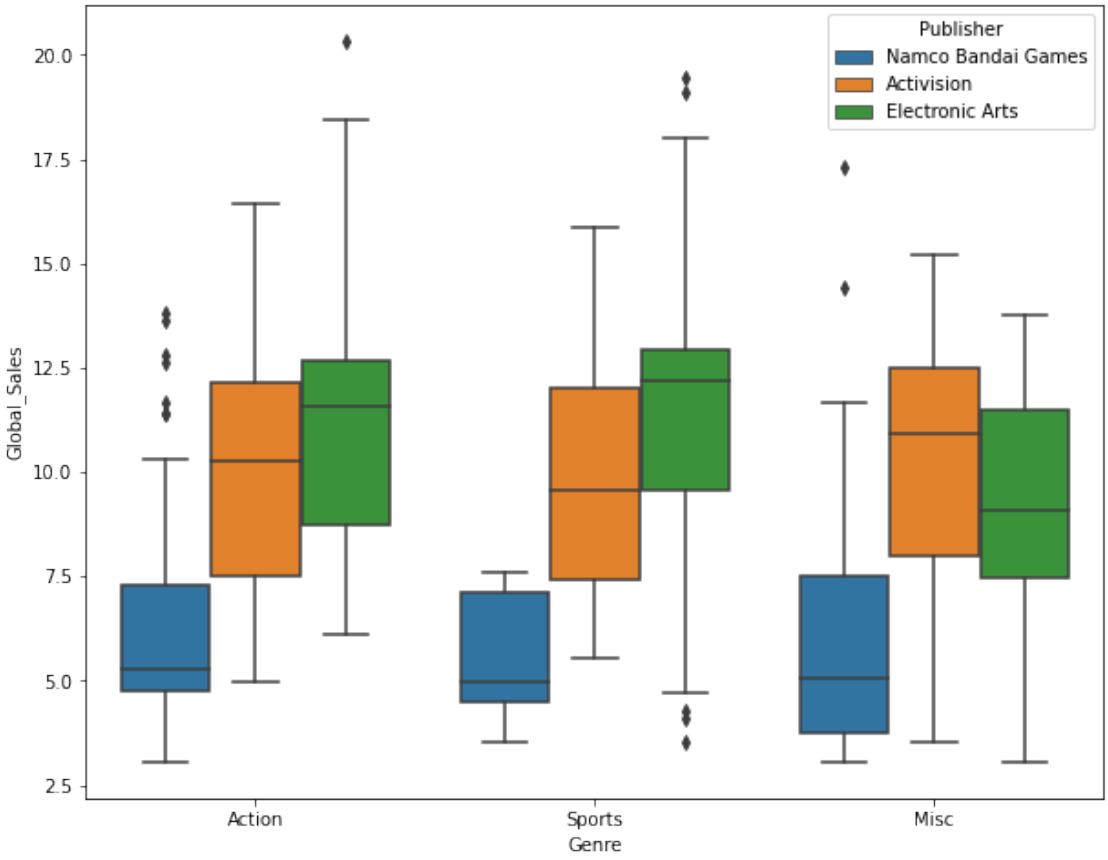
[64]:
```
sns.scatterplot(data=top3_data, x='NA_Sales', y='EU_Sales', hue='Publisher')
```

[64]:
```
<AxesSubplot:xlabel='NA_Sales', ylabel='EU_Sales'>
```



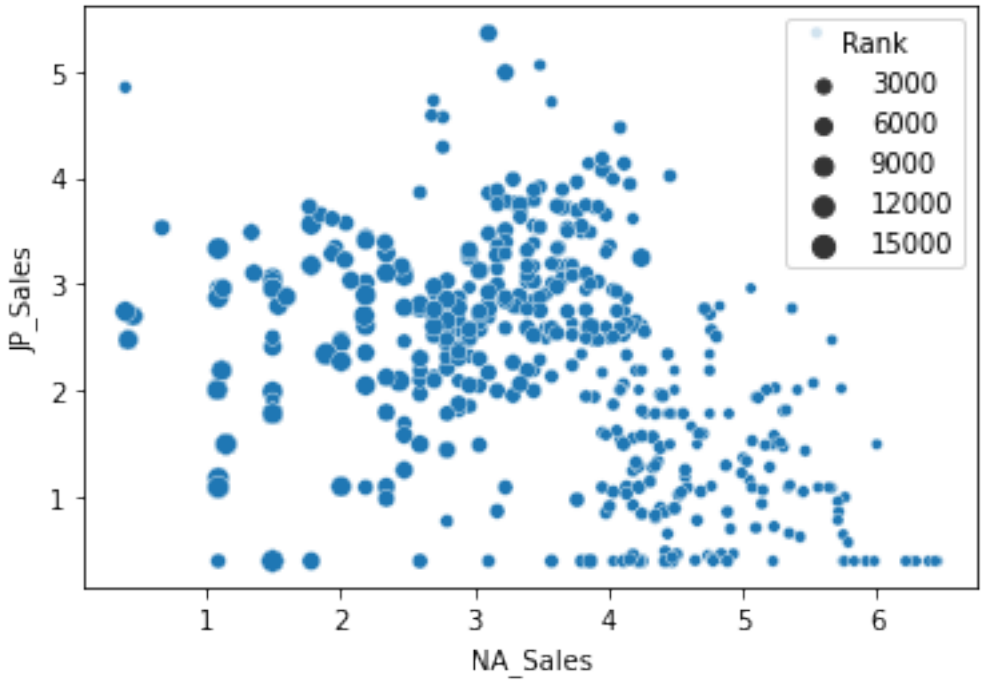
[68]:
```
plt.figure(figsize=(10, 8))
sns.boxplot(data=top3_data, x='Genre', y='Global_Sales', hue='Publisher')
```

[68]: <AxesSubplot:xlabel='Genre', ylabel='Global_Sales'>



[69]:
```
sns.scatterplot(data=top3_data, x='NA_Sales', y='JP_Sales', size='Rank')
```

[69]:
```
<AxesSubplot:xlabel='NA_Sales', ylabel='JP_Sales'>
```


Rank
3000
6000
9000
12000
15000
JP_Sales
NA_Sales
1
2
3
4
5
6