Department of Computer Science

University of Reading

# CSMAD21 – Applied Data Science with Python

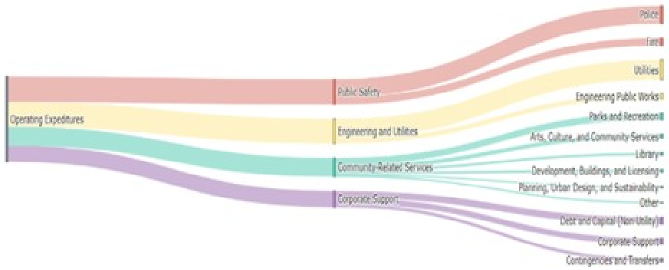
## Data Visualisation 2

# Lecture Objectives

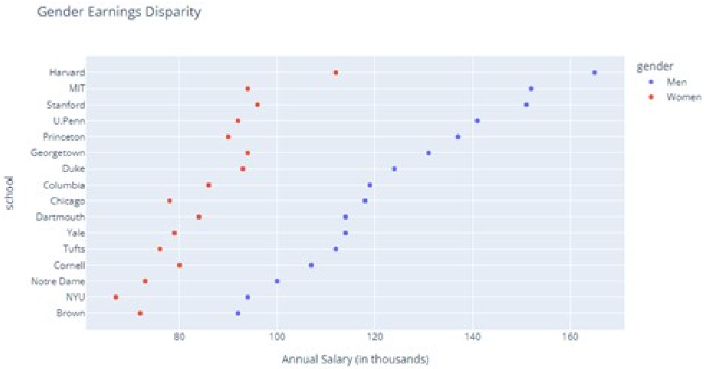
- Implement data discovery and analysis with interactive graphs via Plotly.

# Outline

- Plotly interactive graphs:
  - Sankey
  - Sunburst
  - Dot Plot
- Summary
- Q&A



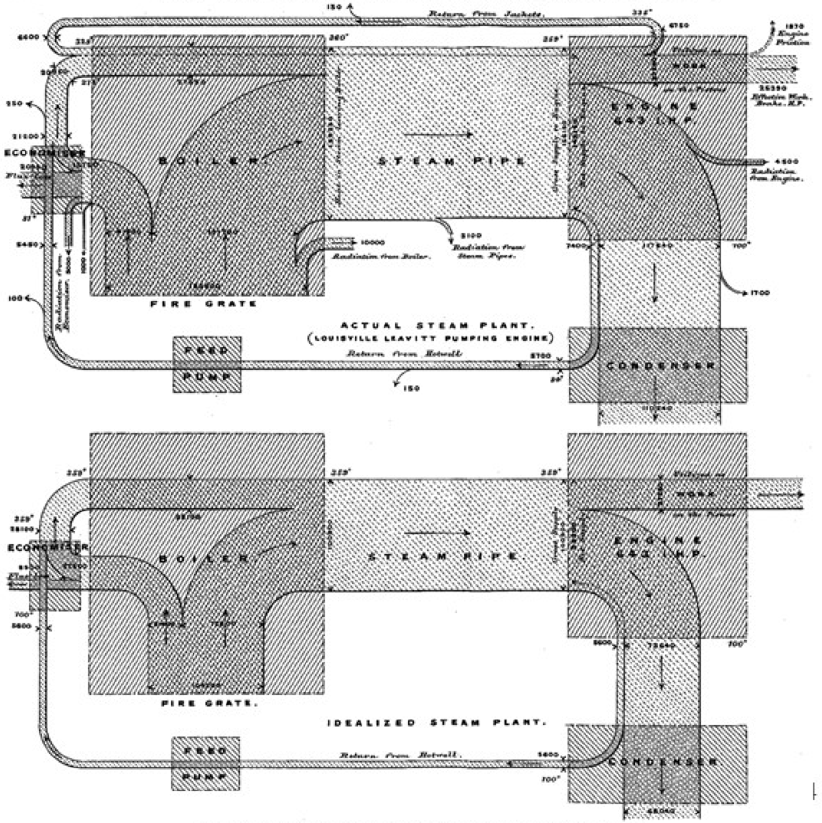
SCAA
Coffee Taster's Flavor Wheel



# Sankey Diagrams

- Sankey diagrams are named after Irish Captain Matthew Henry Phineas Riall Sankey, who used this type of diagram in 1898 in a classic figure showing the energy efficiency of a steam engine.
- They’re a convenient chart for visualizing any kind of measurable flow — Some examples are the flow of travelers, energy, and money.

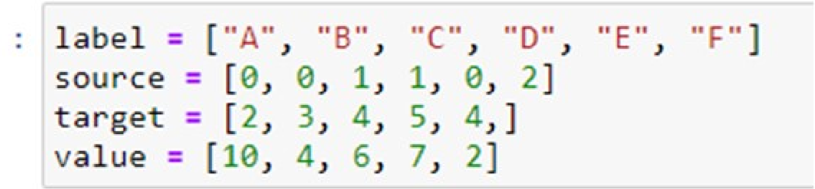
THE THERMAL EFFICIENCY OF STEAM-ENGINES. PLATE 5.



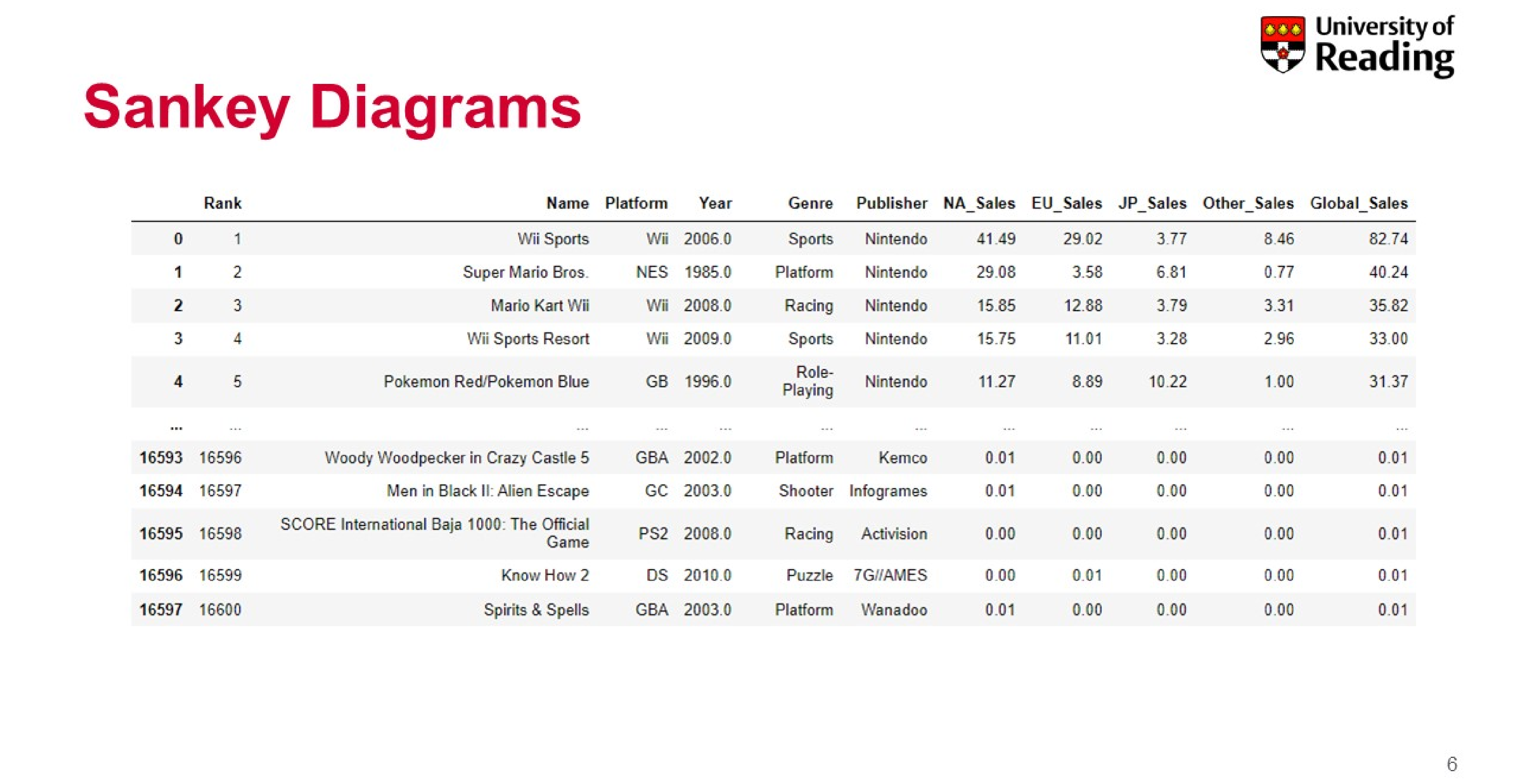
Minutes of Proceedings of The Institution of Civil Engineers, Vol: CXXXIV Session 1897-98, Part IV.

# Sankey Diagrams

- Sankey diagrams visualize the contributions to a flow by defining source to represent the source node, target for the target node, value to set the flow volume, and label that shows the node name.



```
: label = ["A", "B", "C", "D", "E", "F"]
  source = [0, 0, 1, 1, 0, 2]
  target = [2, 3, 4, 5, 4,]
  value = [10, 4, 6, 7, 2]
```



University of Reading

# Sankey Diagrams

| | Rank | Name | Platform | Year | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.49 | 29.02 | 3.77 | 8.46 | 82.74 |
| **1** | 2 | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 | 0.77 | 40.24 |
| **2** | 3 | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.85 | 12.88 | 3.79 | 3.31 | 35.82 |
| **3** | 4 | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.75 | 11.01 | 3.28 | 2.96 | 33.00 |
| **4** | 5 | Pokemon Red/Pokemon Blue | GB | 1996.0 | Role-Playing | Nintendo | 11.27 | 8.89 | 10.22 | 1.00 | 31.37 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **16593** | 16596 | Woody Woodpecker in Crazy Castle 5 | GBA | 2002.0 | Platform | Kemco | 0.01 | 0.00 | 0.00 | 0.00 | 0.01 |
| **16594** | 16597 | Men in Black II: Alien Escape | GC | 2003.0 | Shooter | Infogrames | 0.01 | 0.00 | 0.00 | 0.00 | 0.01 |
| **16595** | 16598 | SCORE International Baja 1000: The Official Game | PS2 | 2008.0 | Racing | Activision | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 |
| **16596** | 16599 | Know How 2 | DS | 2010.0 | Puzzle | 7G//AMES | 0.00 | 0.01 | 0.00 | 0.00 | 0.01 |
| **16597** | 16600 | Spirits & Spells | GBA | 2003.0 | Platform | Wanadoo | 0.01 | 0.00 | 0.00 | 0.00 | 0.01 |

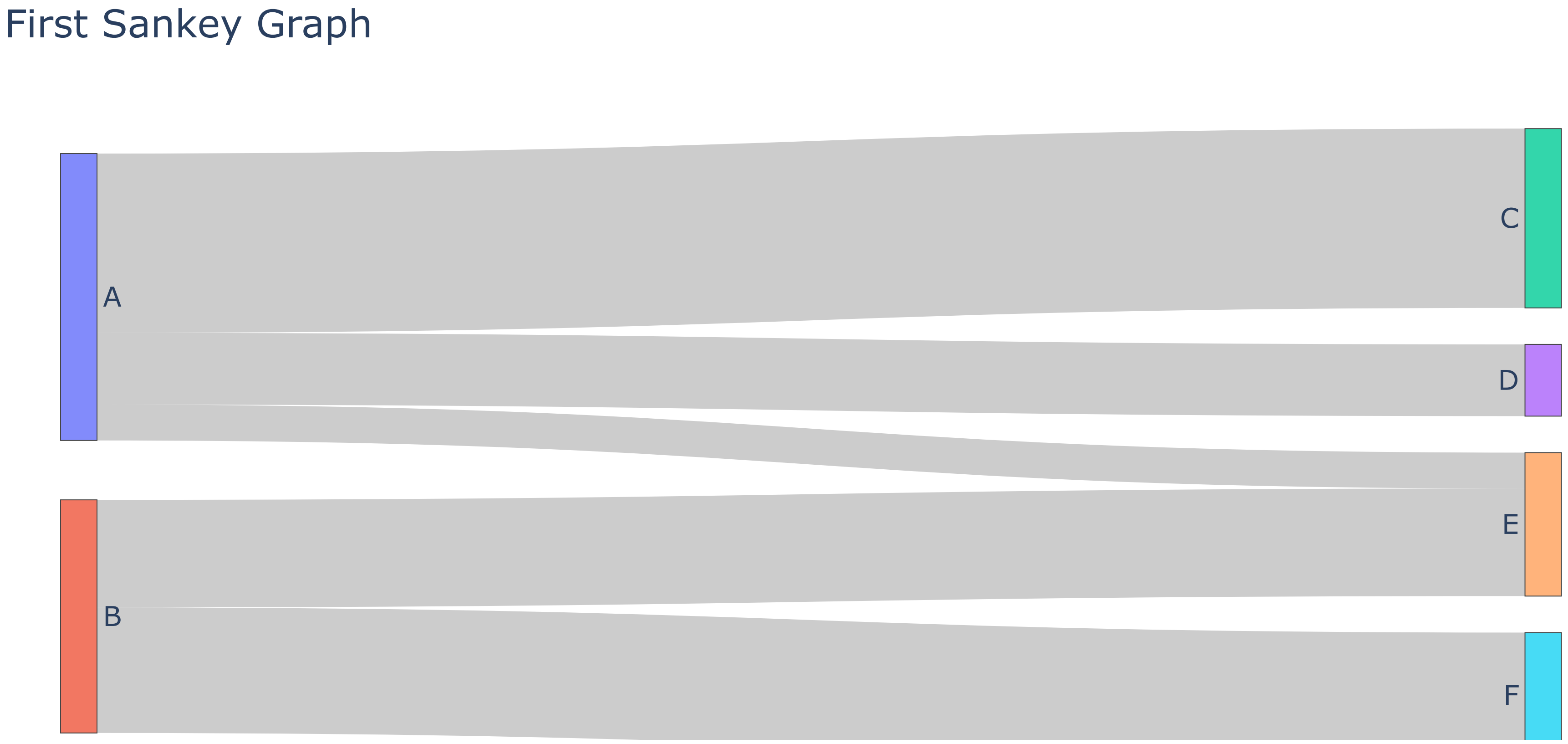
## General Example

In [26]:
```
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
```

In [27]:

```
label = ["A", "B", "C", "D", "E", "F"]
source = [0, 0, 1, 1, 0, 2]
target = [2, 3, 4, 5, 4,]
value = [10, 4, 6, 7, 2]
```

```
In [28]: fig = go.Figure(data=[go.Sankey(
             # Define nodes
             node = dict(
                           label =  label,
                         ),

             # Add links
             link = dict(
                         source =  source,
                         target =  target,
                         value =  value,
             ))])

         fig.update_layout(title_text="First Sankey Graph", font_size=15)
         fig.show()
```



## VGChartz data

The data set containing data for video games sales greater than 100,000 copies generated by a scrape of vgchartz.com. The columns are:

- Rank - Ranking of overall sales
- Name - The games name
- Platform - Platform of the games release (i.e. PC, PS4, etc.)
- Year - Year of the game's release
- Genre - Genre of the game
- Publisher - Publisher of the game
- NA_Sales - Sales in North America (in millions)
- EU_Sales - Sales in Europe (in millions)
- JP_Sales - Sales in Japan (in millions)
- Other_Sales - Sales in the rest of the world (in millions)
- Global_Sales - Total worldwide sale

In [29]:

```
games = pd.read_csv('Datasets/VGChartz.csv')
games.head()
```

Out[29]:

| | Rank | Name | Platform | Year | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.49 | 29.02 | 3.77 | 8.46 | 82.74 |
| **1** | 2 | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 | 0.77 | 40.24 |
| **2** | 3 | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.85 | 12.88 | 3.79 | 3.31 | 35.82 |
| **3** | 4 | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.75 | 11.01 | 3.28 | 2.96 | 33.00 |
| **4** | 5 | Pokemon Red/Pokemon Blue | GB | 1996.0 | Role-Playing | Nintendo | 11.27 | 8.89 | 10.22 | 1.00 | 31.37 |

In [30]:

```
print('Number of records:', len(games))
print('Before droping NAN values \n', games.isna().sum())
games = games.dropna()
print('After droping NAN values number of records:', len(games))
```

```
Number of records: 16598
Before droping NAN values
 Rank             0
Name             0
Platform         0
Year           271
Genre            0
Publisher       58
NA_Sales         0
EU_Sales         0
JP_Sales         0
Other_Sales      0
Global_Sales     0
dtype: int64
After droping NAN values number of records: 16291
```

In [31]:

```
##Extracting the top 5 publishers by "Global Sales"
top_publishers = games.groupby(['Publisher']).sum('Global Sales').reset_index().sort_values(by = 'Global_Sales', ascendi
top_publishers
```

Out[31]:

| | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales |
|---|---|---|---|---|---|
| **359** | Nintendo | 815.75 | 418.30 | 454.99 | 95.19 |
| **138** | Electronic Arts | 584.22 | 367.38 | 13.98 | 127.63 |
| **21** | Activision | 426.01 | 213.72 | 6.54 | 74.79 |
| **455** | Sony Computer Entertainment | 265.22 | 187.55 | 74.10 | 80.40 |
| **524** | Ubisoft | 252.81 | 163.03 | 7.33 | 50.16 |

In [32]:

```
##All nodes:
nodes = top_publishers.Publisher.tolist() + top_publishers.columns[1:].tolist()
nodes
```

Out[32]:

```
['Nintendo',
 'Electronic Arts',
 'Activision',
 'Sony Computer Entertainment',
 'Ubisoft',
 'NA_Sales',
 'EU_Sales',
 'JP_Sales',
 'Other_Sales']
```

In [33]:

```
##Source values
source = []
for a1 in range(0, len(top_publishers.Publisher.tolist())):
    for a2 in range(0, len(top_publishers.columns[1:].tolist())):
        source.append(a1)
source
```

Out[33]:

```
[0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4]
```

In [34]:

```
##Target
target = []
for a1 in range(0, len(top_publishers.Publisher.tolist())):
    for a2 in range(0, len(top_publishers.columns[1:].tolist())):
        target.append(a2 + len(top_publishers.Publisher.tolist()))
target
```

Out[34]:

```
[5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8, 5, 6, 7, 8]
```

In [35]:

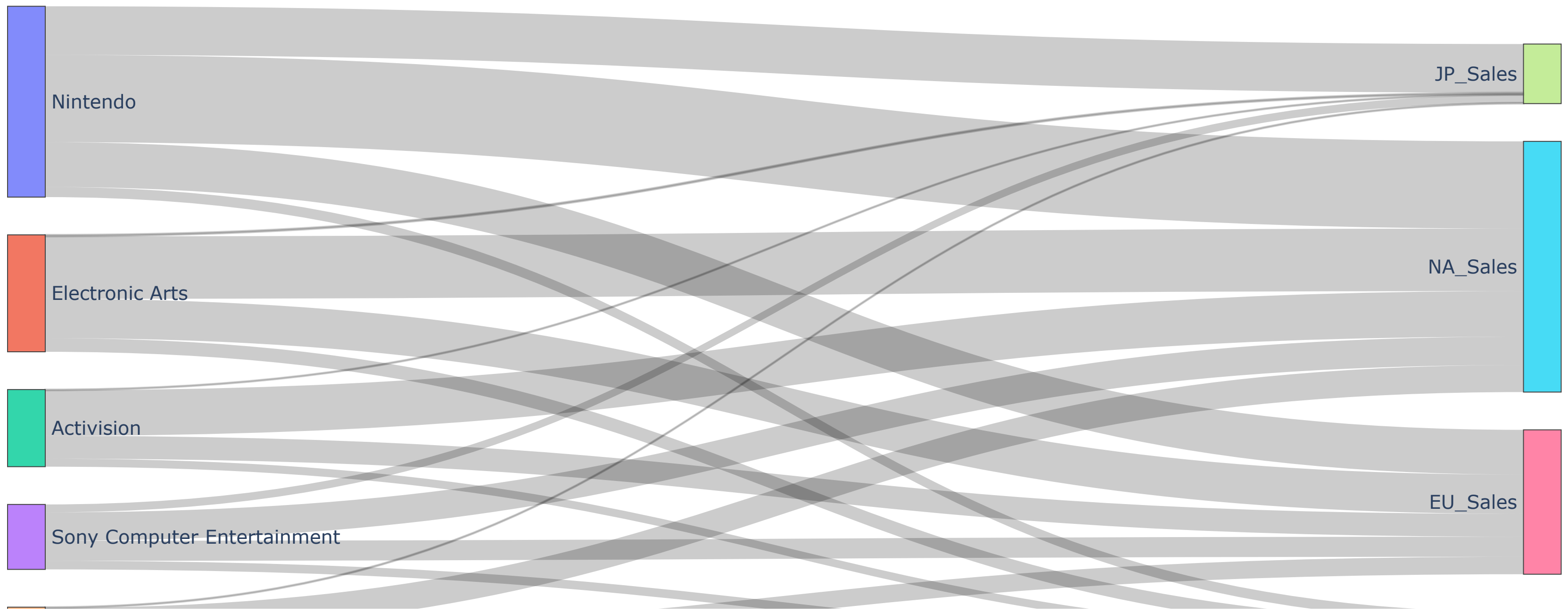
```
##Value
values = []
top_publishers.loc[top_publishers['Publisher'] == 'Nintendo',:].values[0][1:].tolist()
for a1 in top_publishers['Publisher'].tolist():
    values = values + top_publishers.loc[top_publishers['Publisher'] == a1,:].values[0][1:].tolist()
values
```

Out[35]:

```
[815.7500000000001,
 418.3000000000002,
 454.98999999999955,
 95.19000000000037,
 584.2199999999979,
 367.3799999999965,
 13.979999999999958,
 127.63000000000072,
 426.0099999999991,
 213.72000000000037,
 6.539999999999987,
 74.79000000000089,
 265.2199999999993,
 187.55000000000018,
 74.10000000000004,
 80.4000000000002,
 252.81000000000046,
 163.0299999999997,
 7.329999999999993,
 50.15999999999995]
```

In [36]:
```
fig = go.Figure(data=[go.Sankey(
    # Define nodes
                                node = dict(
                                label =  nodes,
                                ),

    # Adding links
                                link = dict(
                                source =  source,
                                target =  target,
                                value =  values,
))])

fig.update_layout(title_text="Top 5 Publishers",font_size=10)
fig.show()
```

Top 5 Publishers



In [37]:

```
##Adding a third layer
###We are going to analyse Nintendo and add a second layer in the graph
top_publishers_nintendo = games.loc[games['Publisher'] =='Nintendo' ]
top_publishers_nintendo = top_publishers_nintendo.groupby(['Publisher', 'Genre']).sum('Global Sales').reset_index().sort_
top_publishers_nintendo
```

Out[37]:

| | Publisher | Genre | Global_Sales | NA_Sales | EU_Sales | JP_Sales | Other_Sales |
|---|---|---|---|---|---|---|---|
| **4** | Nintendo | Platform | 426.18 | 219.46 | 84.90 | 102.36 | 19.43 |
| **7** | Nintendo | Role-Playing | 284.57 | 105.63 | 63.92 | 101.95 | 13.03 |
| **10** | Nintendo | Sports | 218.01 | 98.77 | 66.18 | 35.87 | 17.18 |
| **3** | Nintendo | Misc | 180.67 | 61.98 | 51.62 | 55.25 | 11.78 |
| **6** | Nintendo | Racing | 151.30 | 73.55 | 39.75 | 29.22 | 8.81 |
| **0** | Nintendo | Action | 128.10 | 63.49 | 29.02 | 29.16 | 6.48 |
| **5** | Nintendo | Puzzle | 124.88 | 55.74 | 26.42 | 37.09 | 5.53 |
| **9** | Nintendo | Simulation | 85.25 | 29.70 | 26.05 | 23.65 | 5.86 |
| **8** | Nintendo | Shooter | 69.69 | 51.39 | 9.85 | 6.03 | 2.39 |
| **2** | Nintendo | Fighting | 53.35 | 27.10 | 8.64 | 14.94 | 2.65 |
| **1** | Nintendo | Adventure | 35.71 | 17.72 | 7.66 | 9.01 | 1.28 |
| **11** | Nintendo | Strategy | 26.72 | 11.22 | 4.29 | 10.46 | 0.77 |

In [38]:

```
##All nodes:
#First level nodes
fl_nodes = list(set(top_publishers_nintendo.Publisher.tolist())) ##Extracting the first level, in this case just Nintend
scndl_nodes = top_publishers_nintendo.columns[3:].tolist()##Extracting second level, in this case sales per region
thrdl_nodes = top_publishers_nintendo.Genre.tolist()## Extracting third level, in this case Genre

nodes = fl_nodes + scndl_nodes + thrdl_nodes ##Merging all nodes
nodes
```

Out[38]:

```
['Nintendo',
 'NA_Sales',
 'EU_Sales',
 'JP_Sales',
 'Other_Sales',
 'Platform',
 'Role-Playing',
 'Sports',
 'Misc',
 'Racing',
 'Action',
 'Puzzle',
 'Simulation',
 'Shooter',
 'Fighting',
 'Adventure',
 'Strategy']
```

In [39]:
```
##Source values
source = []
source_lv1 = []
source_lv2 = []
for a1 in range(0, len(fl_nodes)):
    for a2 in range(1, len(scndl_nodes)+1):
        source_lv1.append(a1)##Extracting the source index/node of the first layer; because it is Nintendo it is just 0
        for a3 in range(1, len(thrdl_nodes) + 1):
            source_lv2.append(a2) ##Extracting the source index/node of the second layer (e.g. NA_Sales -> 1, EU_Sales -
source = source_lv1 + source_lv2
source[0:20]
```

Out[39]:
```
[0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2]
```

In [40]:
```
##Target
target_lv1 = []
target_lv2 = []

for a1 in range(0, len(fl_nodes)):
    for a2 in range(0, len(scndl_nodes)):
        target_lv1.append(a2 + len(fl_nodes)) ##Extracting the target index/node of the first layer (e.g. Nintendo (0) t
        for a3 in range(0, len(thrdl_nodes)):
            target_lv2.append(a3 + len(scndl_nodes) + len(fl_nodes))##Extracting the target index/node of the second Lay

target = target_lv1 + target_lv2
target[0:20]
```

Out[40]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 5, 6, 7, 8]
```

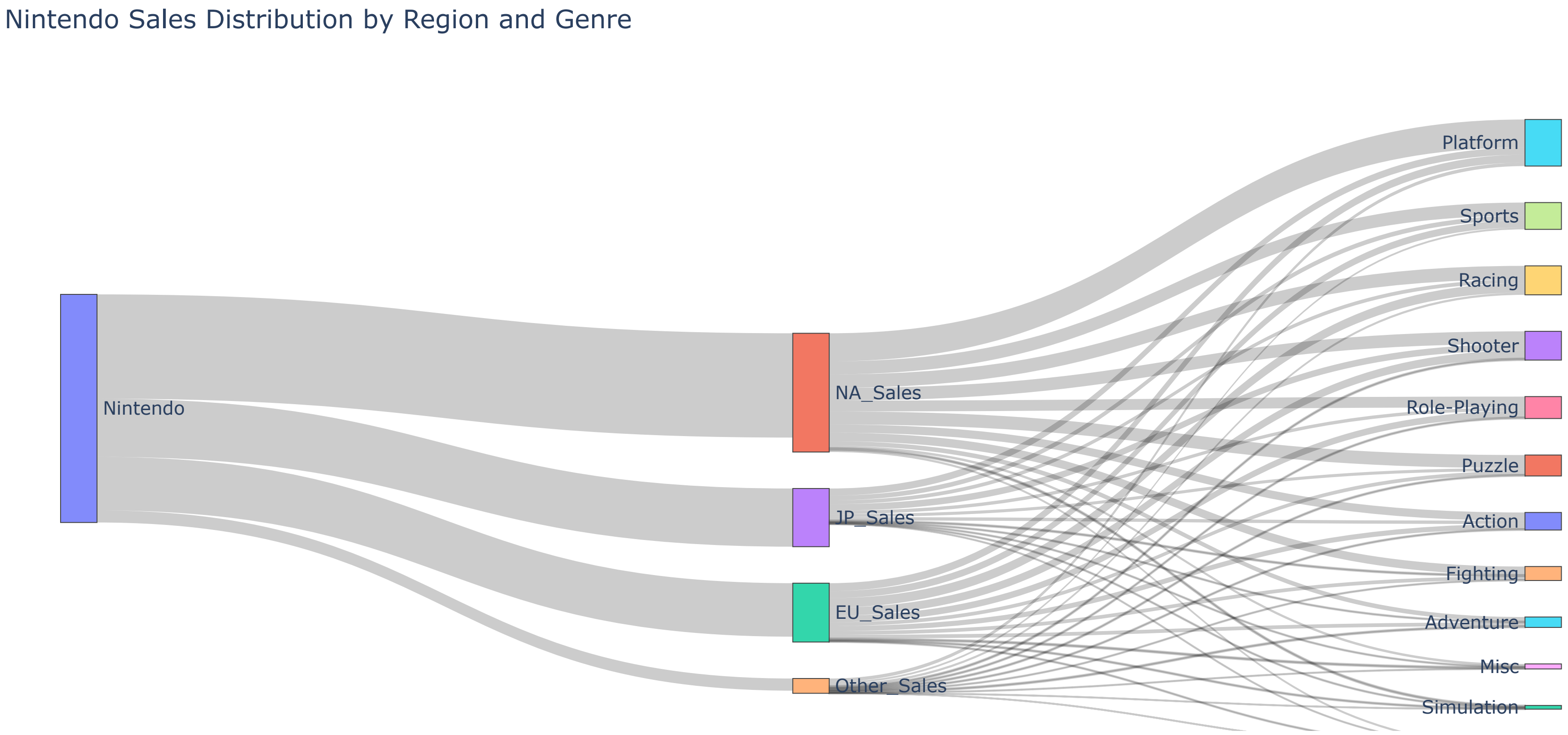
In [41]:

```
##Values
values = [] ##All the values
##First level values:
values_flvl = []
for a1 in top_publishers_nintendo['Publisher'].unique():##Just Nintendo is part of the analysis
    for a2 in top_publishers_nintendo.columns[3:]: ##Will go through the values 'NA_Sales' to 'Other_Sales'
        values_flvl.append(sum(top_publishers_nintendo.loc[top_publishers_nintendo['Publisher'] == a1, a2].values))##Sum

##Second level values:
values_slvl = []
for a1 in top_publishers_nintendo['Genre'].unique():##Will go through the values 'Platform' to 'Strategy'
    values_slvl = values_slvl + top_publishers_nintendo.loc[top_publishers_nintendo['Genre'] == a1, top_publishers_ninte
values = values_flvl + values_slvl
values[:5]
```

Out[41]:

```
[815.7499999999999,
 418.3000000000002,
 454.9900000000001,
 95.18999999999996,
 219.4599999999999]
```

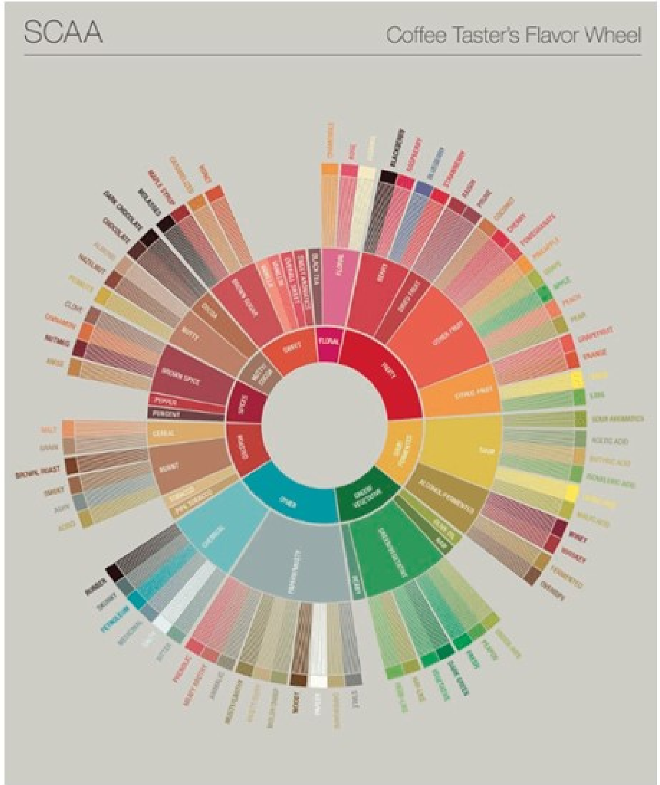
```
In [42]: fig = go.Figure(data=[go.Sankey(
             # Define nodes
                                     node = dict(
                                     label =  nodes,
                                     ),

             # Adding links
                                     link = dict(
                                     source =  source,
                                     target =  target,
                                     value =  values,
         ))])

         fig.update_layout(title_text="Nintendo Sales Distribution by Region and Genre",font_size=10)
         fig.show()
```



Nintendo Sales Distribution by Region and Genre

# Sunburst

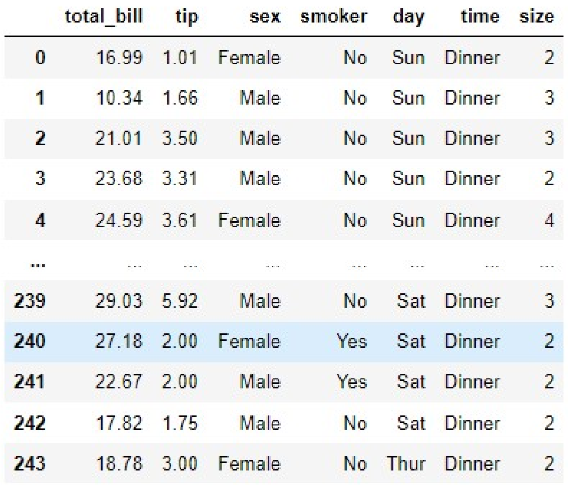
- A Sunburst Diagram is used to visualize hierarchical data, depicted by concentric circles. The circle in the centre represents the root node, with the hierarchy moving outward from the center. A segment of the inner circle bears a hierarchical relationship to those segments of the outer circle which lie within the angular sweep of the parent segment.



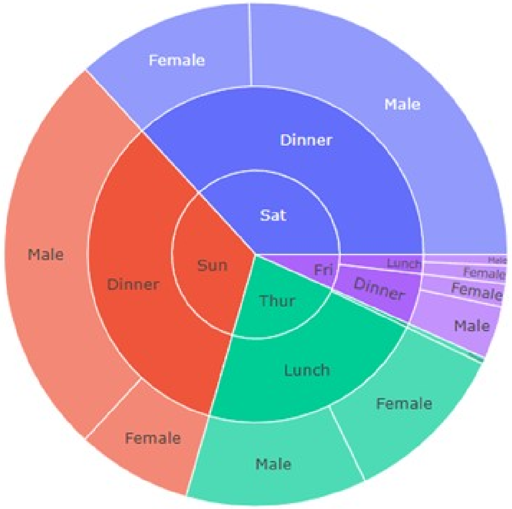
Source: http://food52.com/blog/15618-what-it-means-to-reinvent-the-coffee-flavor-wheel?utm_source=Facebook&utm_medium=SocialMarketing&utm_campaign=Social

# Sunburst

- Hierarchical data are often stored as a rectangular dataframe, with different columns corresponding to different levels of the hierarchy.



| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| 0 | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| 1 | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| 2 | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| 3 | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| 4 | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 239 | 29.03 | 5.92 | Male | No | Sat | Dinner | 3 |
| 240 | 27.18 | 2.00 | Female | Yes | Sat | Dinner | 2 |
| 241 | 22.67 | 2.00 | Male | Yes | Sat | Dinner | 2 |
| 242 | 17.82 | 1.75 | Male | No | Sat | Dinner | 2 |
| 243 | 18.78 | 3.00 | Female | No | Thur | Dinner | 2 |



## General Example

In [43]:

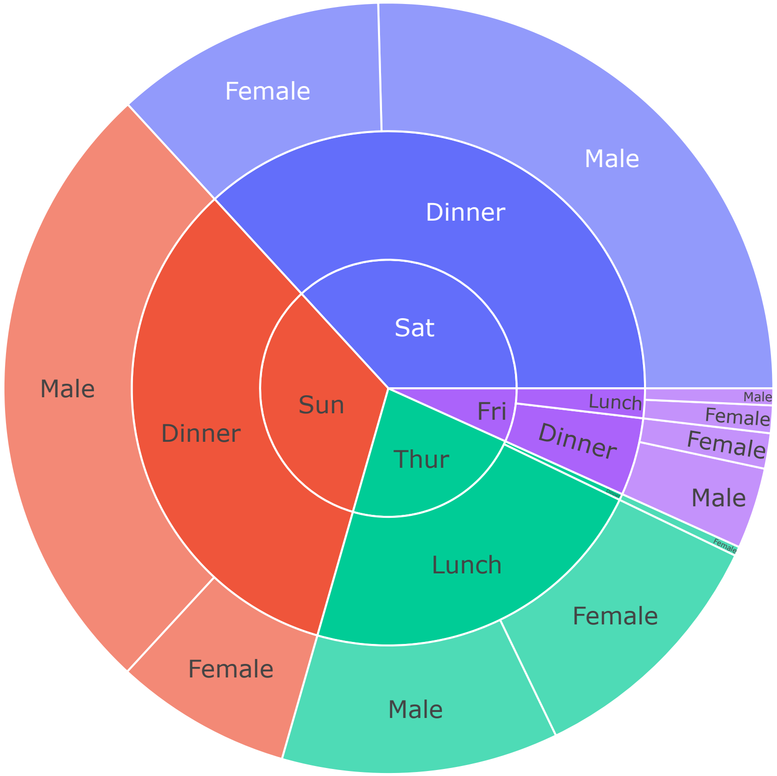
```python
import plotly.express as px
df = px.data.tips()
df.head()
```

Out[43]:

|  | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| **0** | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| **1** | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| **2** | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| **3** | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| **4** | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

In [44]:

```python
fig = px.sunburst(df, path=['day', 'time', 'sex'], values='total_bill')
fig.show()
```



## VGChartz data

In [45]:

```
games.head()
```

Out[45]:

| | Rank | Name | Platform | Year | Genre | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales | Global_Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Wii Sports | Wii | 2006.0 | Sports | Nintendo | 41.49 | 29.02 | 3.77 | 8.46 | 82.74 |
| **1** | 2 | Super Mario Bros. | NES | 1985.0 | Platform | Nintendo | 29.08 | 3.58 | 6.81 | 0.77 | 40.24 |
| **2** | 3 | Mario Kart Wii | Wii | 2008.0 | Racing | Nintendo | 15.85 | 12.88 | 3.79 | 3.31 | 35.82 |
| **3** | 4 | Wii Sports Resort | Wii | 2009.0 | Sports | Nintendo | 15.75 | 11.01 | 3.28 | 2.96 | 33.00 |
| **4** | 5 | Pokemon Red/Pokemon Blue | GB | 1996.0 | Role-Playing | Nintendo | 11.27 | 8.89 | 10.22 | 1.00 | 31.37 |

In [46]:

```
##We are going to analyse the top 5 publishers by Global Sales
top_publishers = games.groupby(['Publisher']).sum('Global Sales').reset_index().sort_values(by = 'Global_Sales', ascendi
top_publishers
```

Out[46]:

| | Publisher | NA_Sales | EU_Sales | JP_Sales | Other_Sales |
|---|---|---|---|---|---|
| **359** | Nintendo | 815.75 | 418.30 | 454.99 | 95.19 |
| **138** | Electronic Arts | 584.22 | 367.38 | 13.98 | 127.63 |
| **21** | Activision | 426.01 | 213.72 | 6.54 | 74.79 |
| **455** | Sony Computer Entertainment | 265.22 | 187.55 | 74.10 | 80.40 |
| **524** | Ubisoft | 252.81 | 163.03 | 7.33 | 50.16 |

In [47]:

```
##Working out the data to represent it in a hierarchy:
top_publishers_2 = pd.DataFrame()
for a1 in top_publishers.columns[1:]:##Will go through the "NA_Sales" to "Other_Sales"
    df1 = top_publishers.loc[:, ['Publisher', a1]]##Extract the data per sales region
    df1['Region'] = a1.split('_')[0]##Add a colun called "Region" that stores the first part of the string (e.g. "NA_Sale
    df1 = df1.rename(columns={a1: "Sales"})## Rename the sales per region to sales (e.g. "NA_Sales" -> "Sales")
    top_publishers_2 = pd.concat([top_publishers_2, df1])##Concat the dataset
top_publishers_2
```

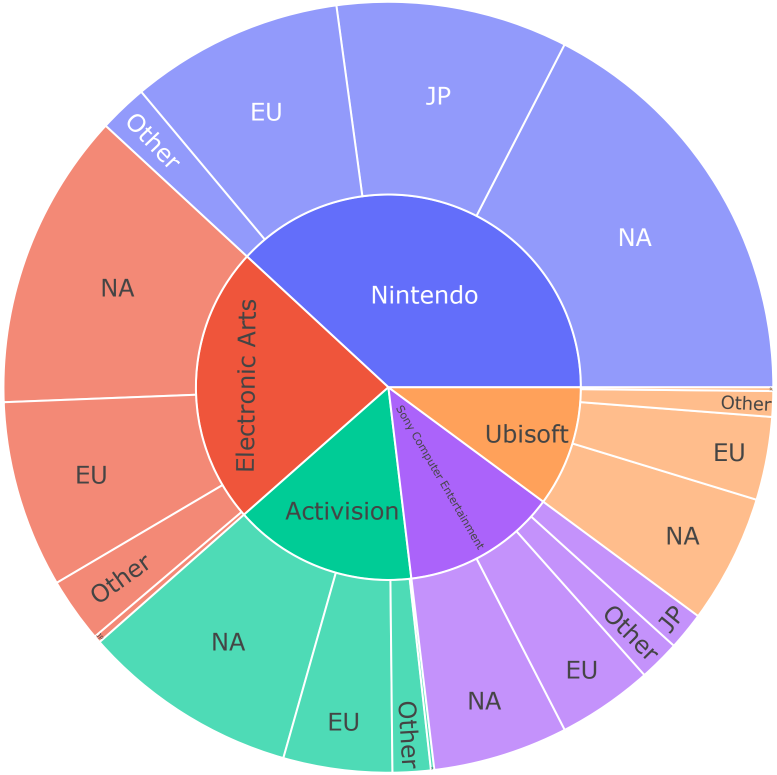
Out[47]:

| | Publisher | Sales | Region |
|---|---|---|---|
| **359** | Nintendo | 815.75 | NA |
| **138** | Electronic Arts | 584.22 | NA |
| **21** | Activision | 426.01 | NA |
| **455** | Sony Computer Entertainment | 265.22 | NA |
| **524** | Ubisoft | 252.81 | NA |
| **359** | Nintendo | 418.30 | EU |
| **138** | Electronic Arts | 367.38 | EU |
| **21** | Activision | 213.72 | EU |
| **455** | Sony Computer Entertainment | 187.55 | EU |
| **524** | Ubisoft | 163.03 | EU |
| **359** | Nintendo | 454.99 | JP |
| **138** | Electronic Arts | 13.98 | JP |
| **21** | Activision | 6.54 | JP |
| **455** | Sony Computer Entertainment | 74.10 | JP |
| **524** | Ubisoft | 7.33 | JP |
| **359** | Nintendo | 95.19 | Other |
| **138** | Electronic Arts | 127.63 | Other |
| **21** | Activision | 74.79 | Other |
| **455** | Sony Computer Entertainment | 80.40 | Other |

|  | Publisher | Sales | Region |
|---|---|---|---|
| **524** | Ubisoft | 50.16 | Other |

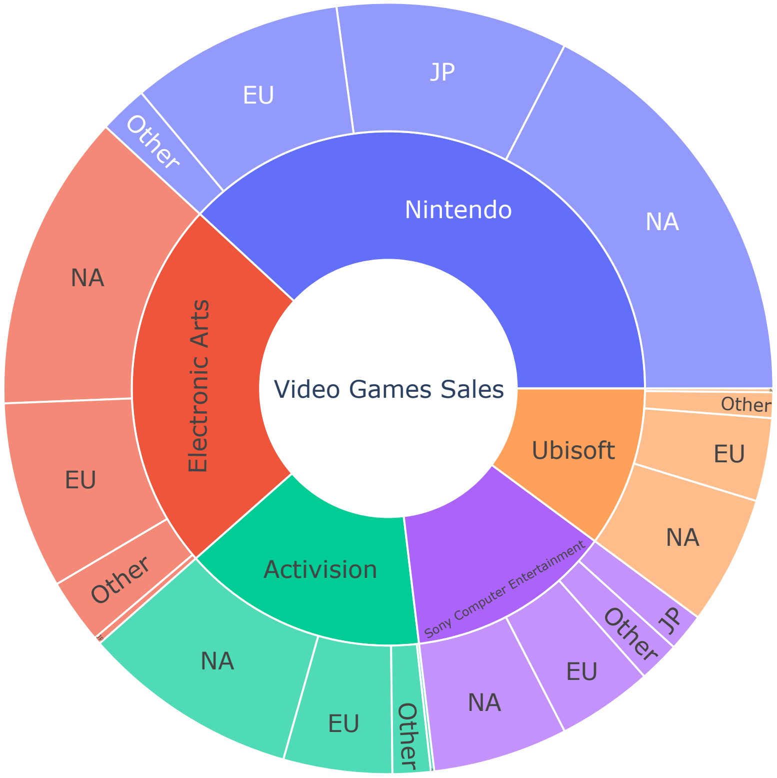
In [48]:

```
import plotly.express as px
fig = px.sunburst(top_publishers_2, path=['Publisher', 'Region'], values='Sales')
fig.show()
```



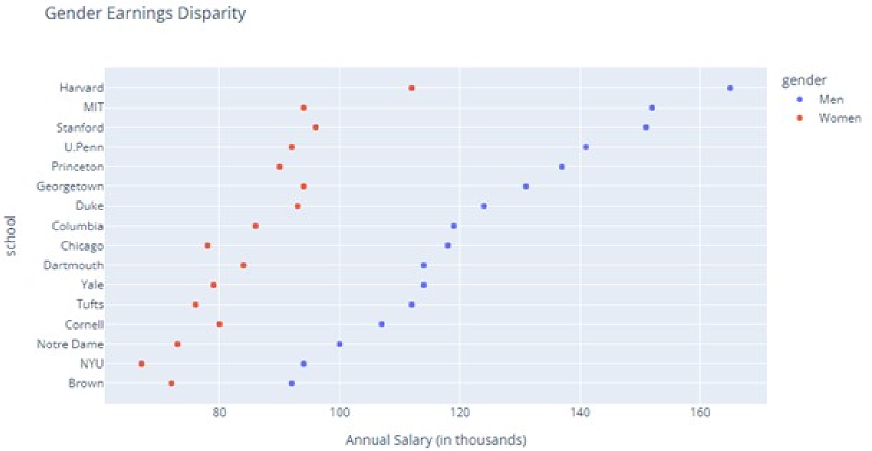
In [49]:

```
##Adding some labels
top_publishers_2['Data'] = 'Video Games Sales'
import plotly.express as px
fig = px.sunburst(top_publishers_2, path=['Data', 'Publisher', 'Region'], values='Sales')
fig.show()
```

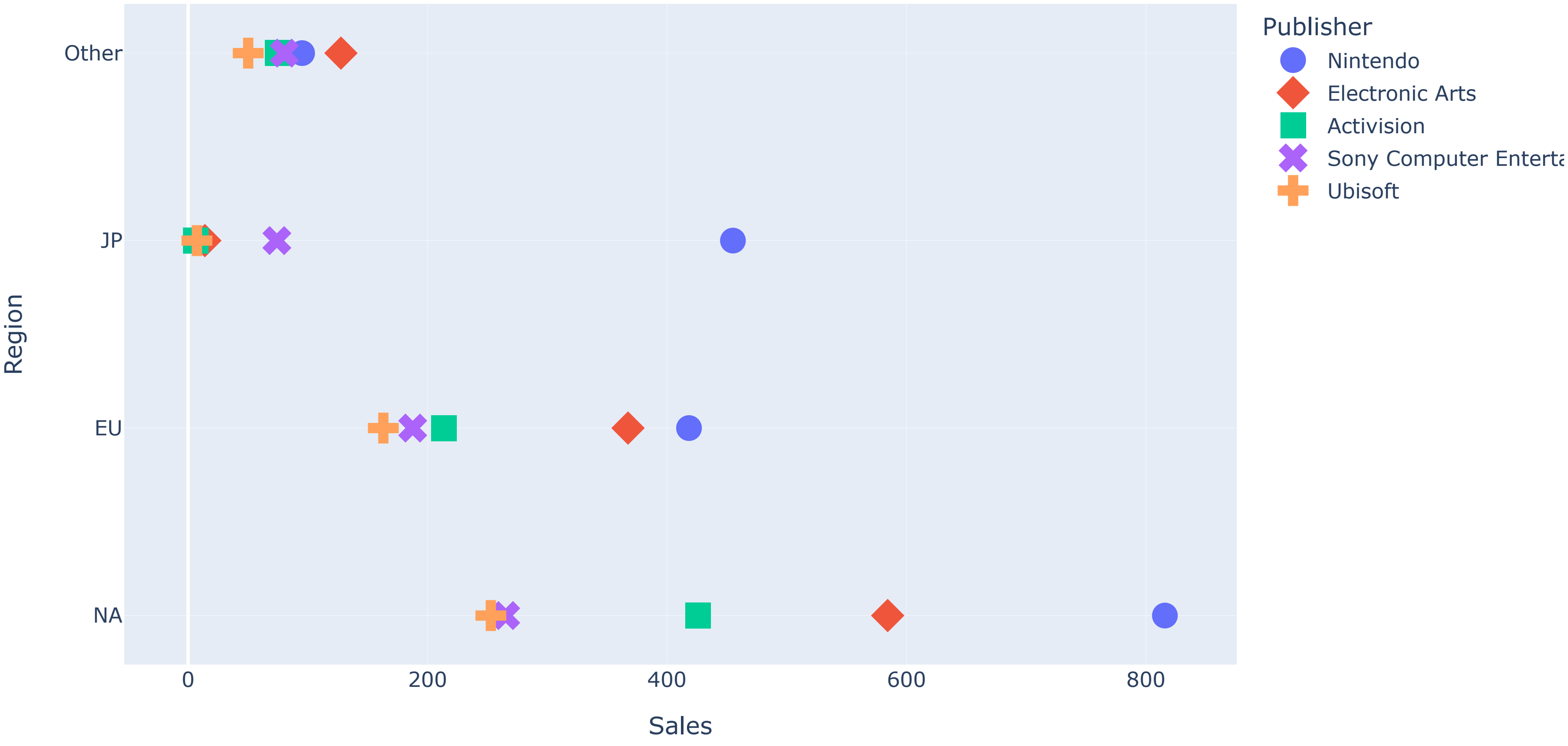


# Dot Plots

- Dot plots (also known as Cleveland dot plots) are scatter plots with one categorical axis and one continuous axis. They can be used to show changes between two (or more) points in time or between two (or more) conditions. Compared to a bar chart, dot plots can be less cluttered and allow for an easier comparison between conditions.
- Are one of the simplest statistical plots and are suitable for small to moderate sized data sets. They are useful for highlighting clusters and gaps, as well as outliers.



In [50]:
```
fig = px.scatter(top_publishers_2, y="Region", x="Sales", color="Publisher", symbol="Publisher")
fig.update_traces(marker_size=15)
fig.show()
```



# Summary

- Plotly allows to create a complex and interactive graphs that can be implemented to analyse and describe the data in a different way. Some of these graphs allows us to depict the data as a flow, hierarchy and to analyse changes between variables and dimensions.

# Questions

# References

- Plotly Python Open Source Graphing Library Basic Charts, https://plotly.com/python/basic-charts/
- Dot Plot, https://en.wikipedia.org/wiki/Dot_plot_(statistics)
- Sankey diagram, https://en.wikipedia.org/wiki/Sankey_diagram
- Sunburst Diagram, https://datavizproject.com/data-type/sunburst-diagram/