In this practice, you will learn to create statistical charts using data visualization libraries Plotly and Seaborn.

1. You will create 10 charts.
2. Load HollywoodsMostProfitableStories.csv and use Plotly to create the following charts. Every figure must include a title. Each axis must be labeled.

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
import pandas as pd
film=pd.read_csv("HollywoodsMostProfitableStories.csv")
film.head()
```

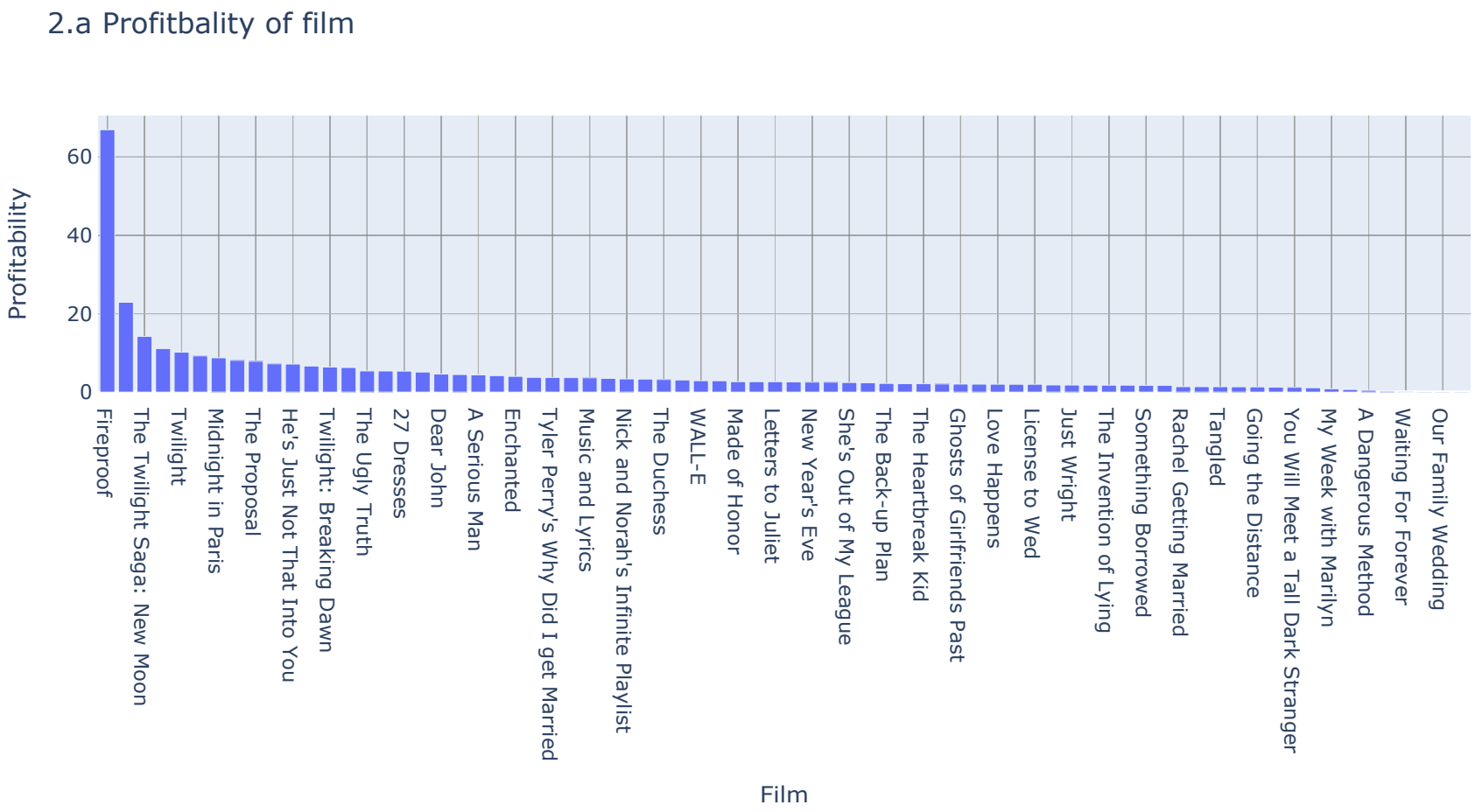
Out[2]:

	Film	Genre	Lead Studio	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross	Year
0	27 Dresses	Comedy	Fox	71.0	5.343622	40.0	160.308654	2008
1	(500) Days of Summer	Comedy	Fox	81.0	8.096000	87.0	60.720000	2009
2	A Dangerous Method	Drama	Independent	89.0	0.448645	79.0	8.972895	2011
3	A Serious Man	Drama	Universal	64.0	4.382857	89.0	30.680000	2009
4	Across the Universe	Romance	Independent	84.0	0.652603	54.0	29.367143	2007

a. A bar chart showing the profitability of the film. The X-axis is the film. The Y-axis is profitability. The bars should be sorted from the most to the least profitable.

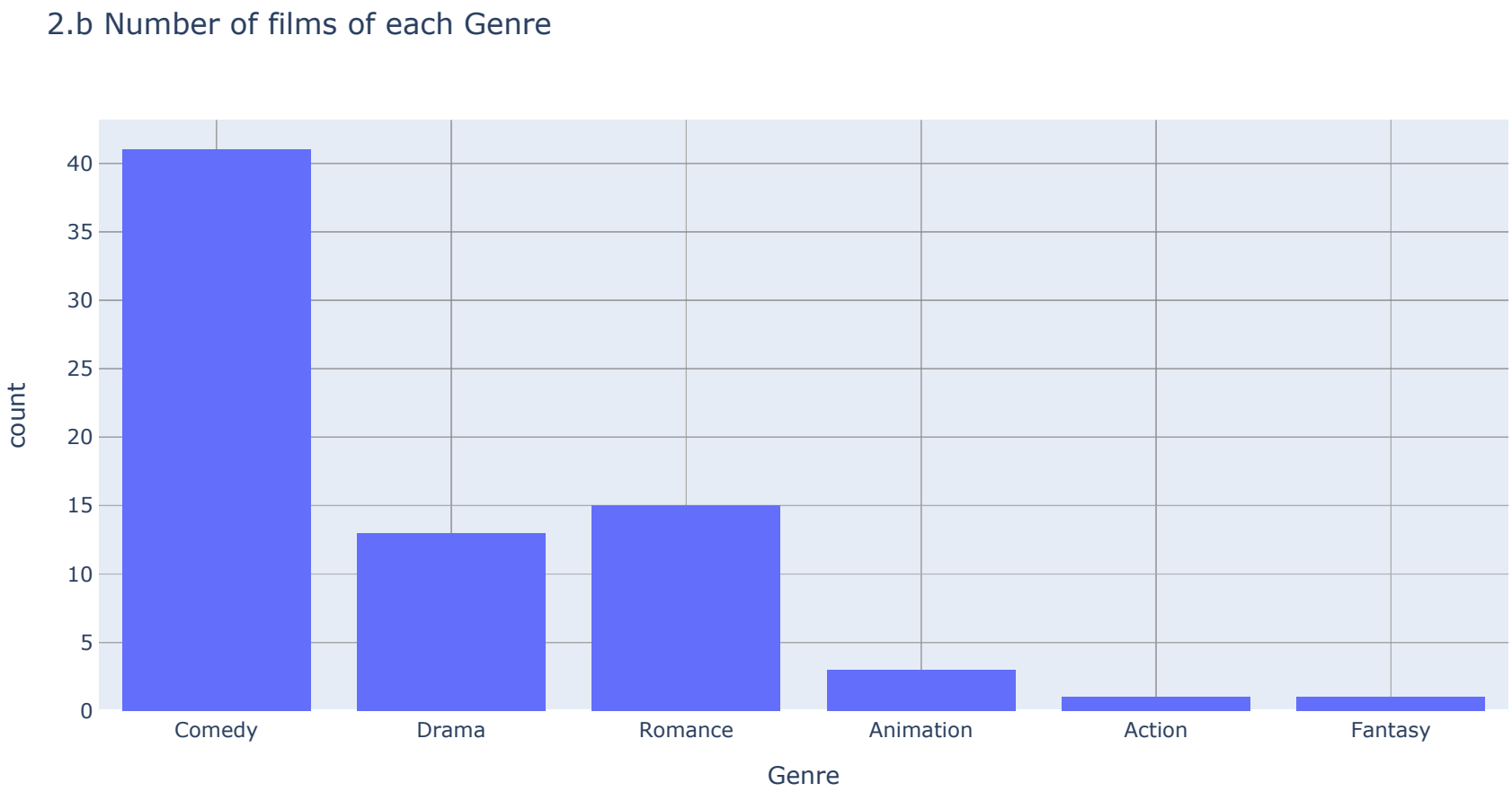
In [3]:
```
import plotly.express as px
```

In [4]:
```
film_sort=film.sort_values('Profitability', ascending=False)
fig_2a=px.bar(film_sort, x='Film', y='Profitability', title='2.a Profitbality of film')
fig_2a.show()
# fig_2a.write_image('fig_2a.png')
```



b. A histogram showing the number of films for each Genre. The X-axis is the Genre. The Y-axis is the number of films in the spreadsheet from each Genre.

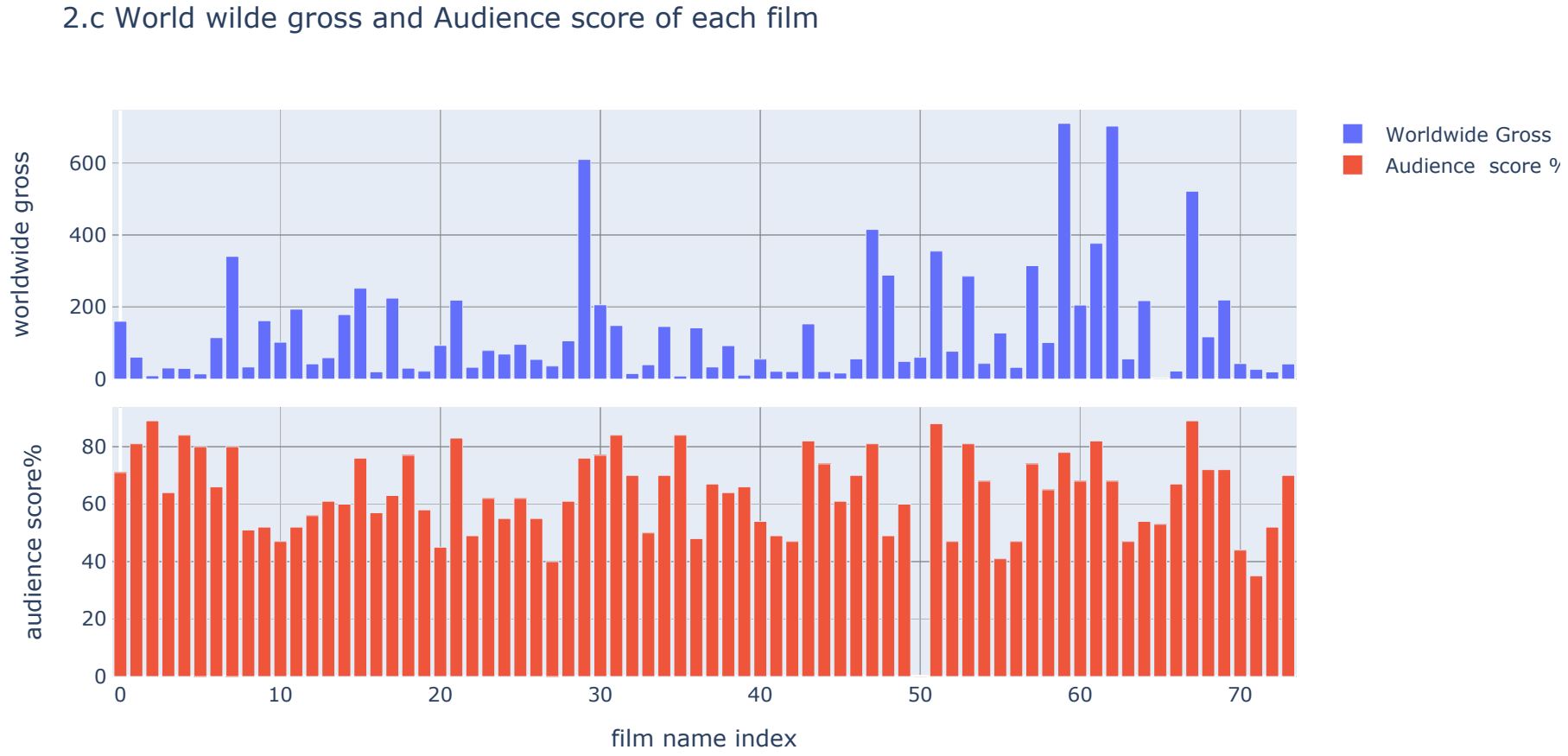
In [5]:
```
fig_2b=px.histogram(film, x='Genre', histfunc='sum',title='2.b Number of films of each Genre')
fig_2b.show()
# fig_2b.write_image('fig_2b.png')
```



c. A figure with two bar plots: Worldwide Gross and Audience. The X-axis is the film index. When the mouse cursor hovers over a bar, the film's title should be displayed in the tooltip.

In [6]:
```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
```

In [7]:
```
fig_2c=make_subplots(rows=2, cols=1, shared_xaxes=True, vertical_spacing=0.05)
fig_2c.add_trace(go.Bar(x=film['Film'].index, y=film['Worldwide Gross'], name='Worldwide Gross',
                        text=film['Film']), row=1, col=1)
fig_2c.add_trace(go.Bar(x=film['Film'].index, y=film['Audience  score %'], name='Audience  score %',
                        text=film['Film']), row=2, col=1)

fig_2c['layout']['xaxis2'].update(title="film name index")
fig_2c['layout']['yaxis1'].update(title="worldwide gross")
fig_2c['layout']['yaxis2'].update(title="audience score%")
fig_2c['layout'].update(title="2.c World wilde gross and Audience score of each film")
fig_2c.show()
# fig_2c.write_image('fig_2c.png')
```



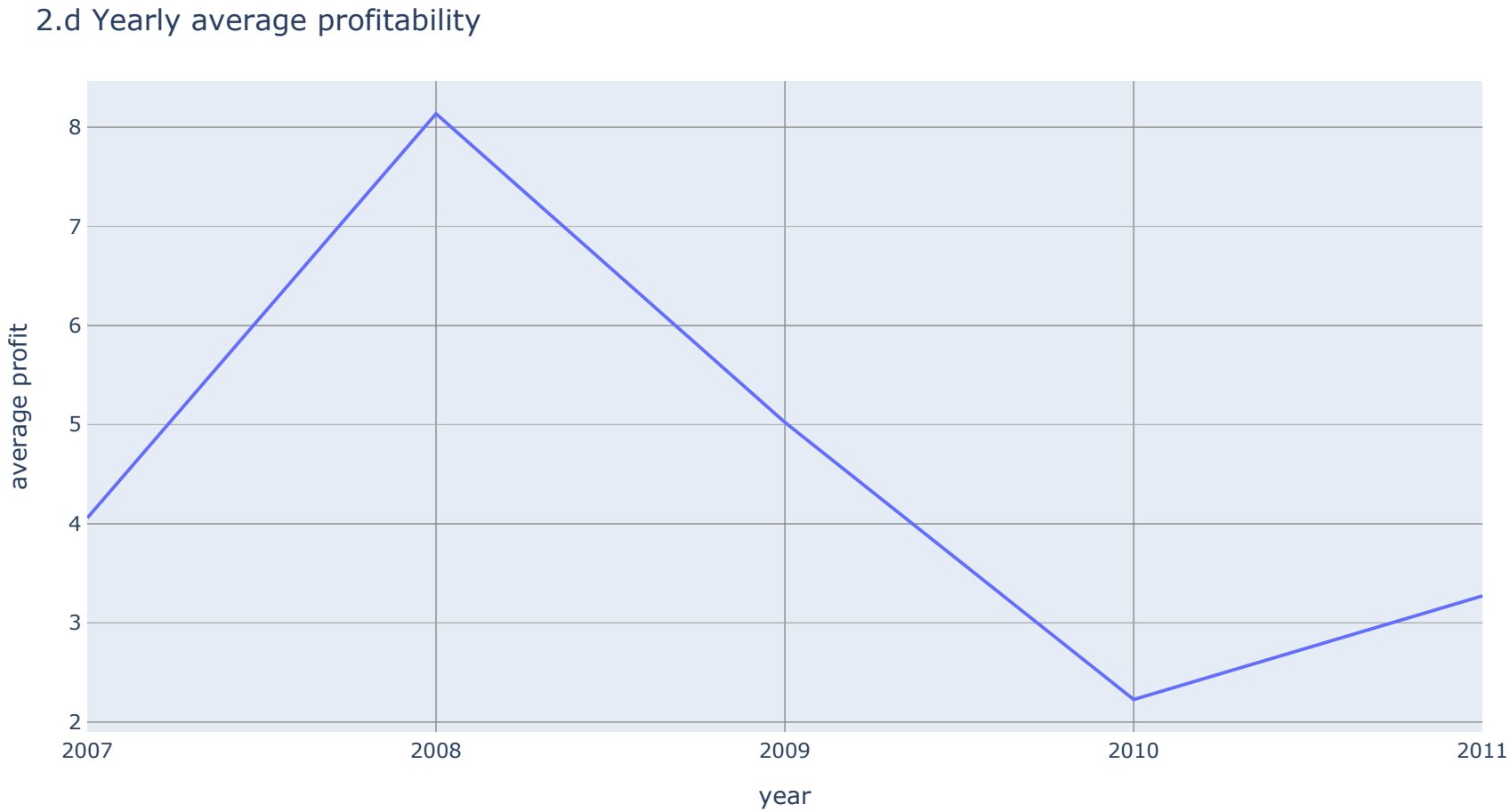
d. A line chart showing the profitability of the films over the years. The X-axis is the Year. The Y-axis is the average profitability.

In [8]:
```
per_year=pd.DataFrame(film.groupby(['Year']).mean())
per_year.head()
```

Out[8]:

Year	Audience score %	Profitability	Rotten Tomatoes %	Worldwide Gross
2007	66.727273	4.058520	48.818182	119.523219
2008	69.789474	8.136442	53.789474	186.272113
2009	62.750000	5.023312	44.250000	167.282500
2010	55.684211	2.227516	37.473684	89.894684
2011	67.583333	3.272555	54.583333	116.979377

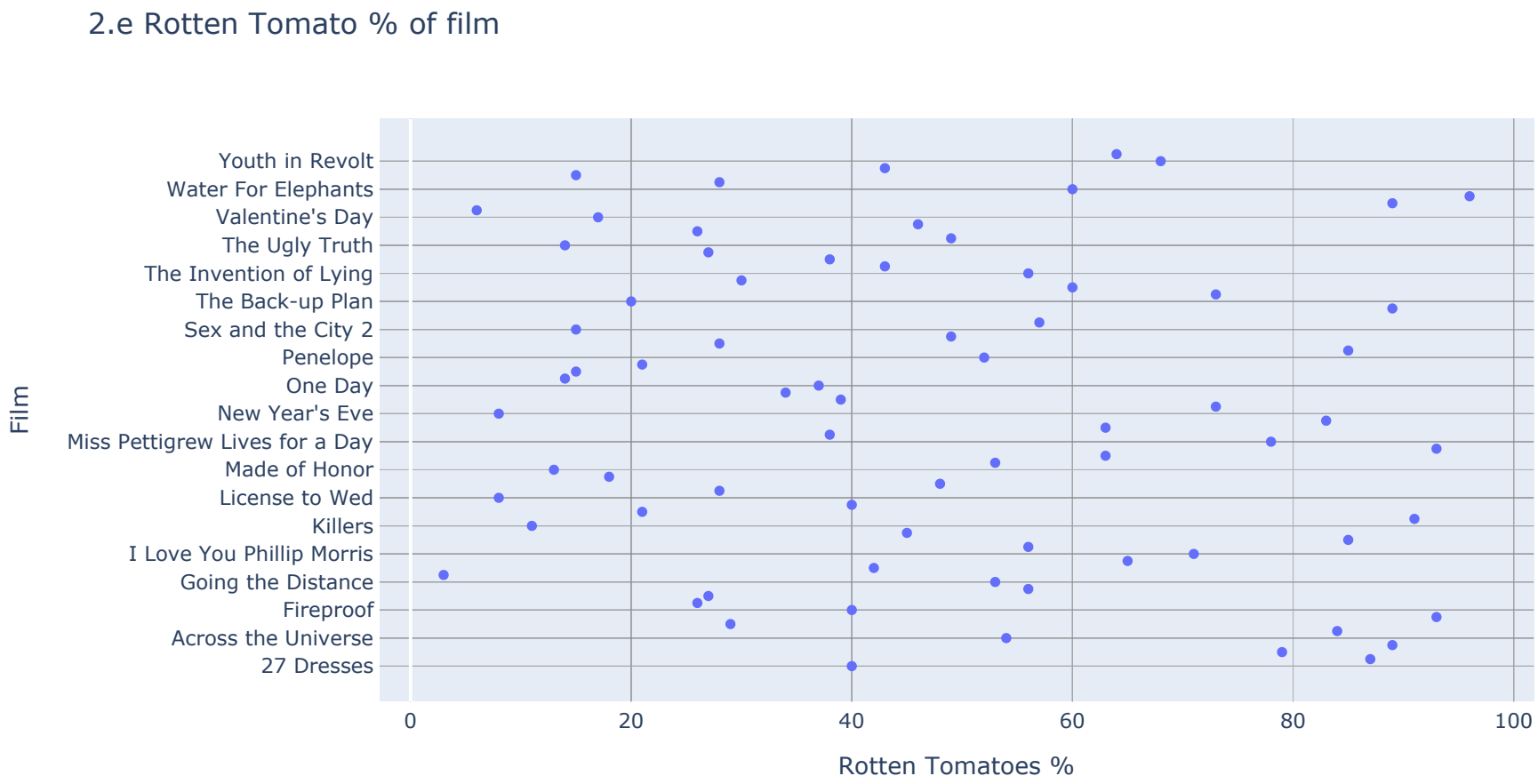
In [9]:
```
fig_2d=px.line(per_year, x=per_year.index, y='Profitability')
fig_2d['layout']['xaxis'].update(title="year", type='category')
fig_2d['layout']['yaxis'].update(title="average profit")
fig_2d['layout'].update(title="2.d Yearly average profitability")
fig_2d.show()
# fig_2d.write_image('fig_2d.png')
```



plotly express: dot plot

e. A dot plot showing the Rotten Tomato %. The X-axis is the Rotten Tomato %. The Y-axis is the film title.

In [10]:
```
fig_2e=px.scatter(film, x='Rotten Tomatoes %', y='Film', title='2.e Rotten Tomato % of film')
fig_2e.show()
# fig_2e.write_image('fig_2e.png')
```



1. Load Housing_price.csv and use Plotly to create the following charts. Every figure must include a title. Each axis must be labeled.

In [11]:
```
house=pd.read_csv("Housing_price.csv")
house.head()
#print(house.info())
```

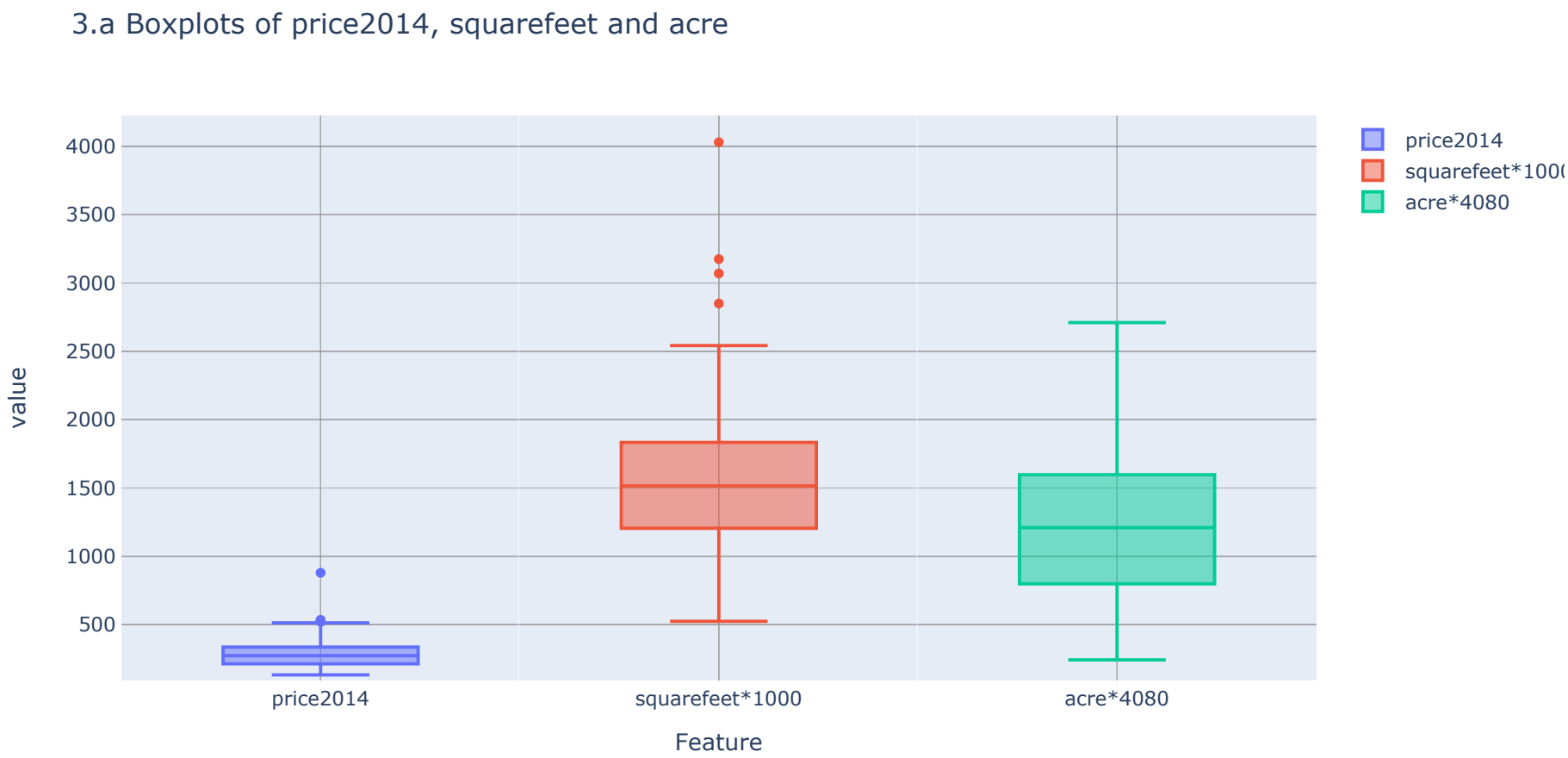
Out[11]:

	housenum	acre	acregroup	adj1998	adj2007	adj2011	bedgroup	bedrooms	bikescore	diff2014	...	price1998	price2007	price2011	price2014
0	1	0.28	> 1/4 acre	148.3626	233.8419	191.8211	3 beds	3	35	62.36645	...	101.5	203.5	181.1	210.729
1	2	0.29	> 1/4 acre	135.2073	261.4203	206.9677	3 beds	3	44	68.96375	...	92.5	227.5	195.4	204.171
2	3	0.36	> 1/4 acre	256.5284	401.0359	347.9472	3 beds	3	66	82.13365	...	175.5	349.0	328.5	338.662
3	4	0.26	> 1/4 acre	231.6795	305.0861	257.4915	3 beds	3	61	44.57055	...	158.5	265.5	243.1	276.250
4	5	0.31	> 1/4 acre	271.8762	298.7660	236.6253	4+ beds	4	53	-102.70300	...	186.0	260.0	223.4	169.173

5 rows × 31 columns

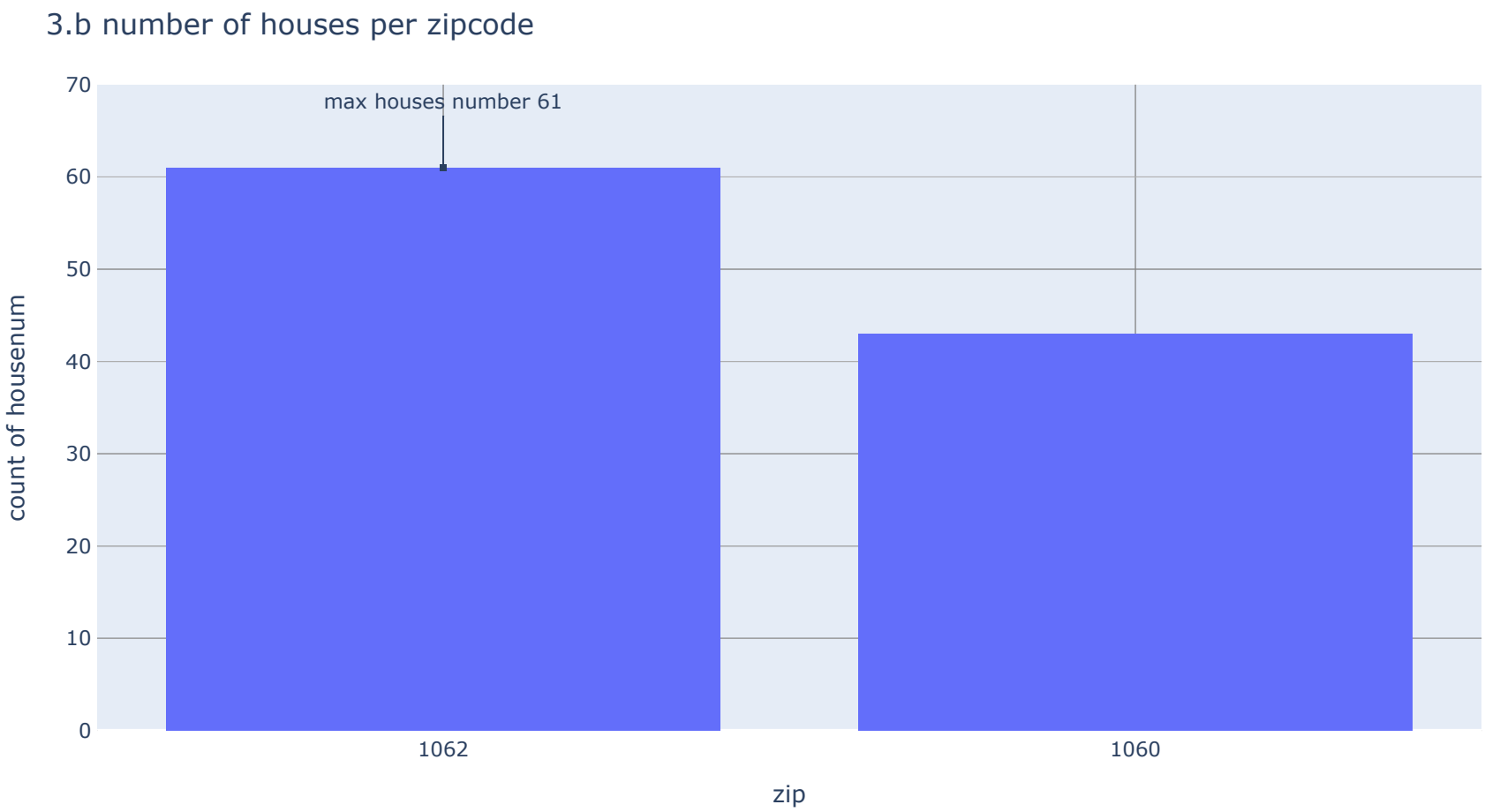
a. A figure that contains three boxplots: price2014, squarefeet, and acre. i. The values in “square feet” and “acre” columns are too small compared with the price column. To make the chart look better, you may multiply “squarefeet” by 1000. (The data in the squarefeet column was divided by 1000.) You may convert the values in the “acre” column to squareyard by multiplying 4840

In [12]:
```
fig_3a=make_subplots(rows=1, cols=3, shared_yaxes=True,horizontal_spacing=0)
fig_3a.add_trace(go.Box(y=house['price2014'], name='price2014'),row=1, col=1)
fig_3a.add_trace(go.Box(y=house['squarefeet']*1000, name='squarefeet*1000'),row=1, col=2)
fig_3a.add_trace(go.Box(y=house['acre']*4840, name='acre*4080'),row=1, col=3)
fig_3a['layout']['xaxis2'].update(title="Feature")
fig_3a['layout']['yaxis'].update(title="value")
fig_3a['layout'].update(title="3.a Boxplots of price2014, squarefeet and acre")
fig_3a.show()
# fig_3a.write_image('fig_3a.png')
```



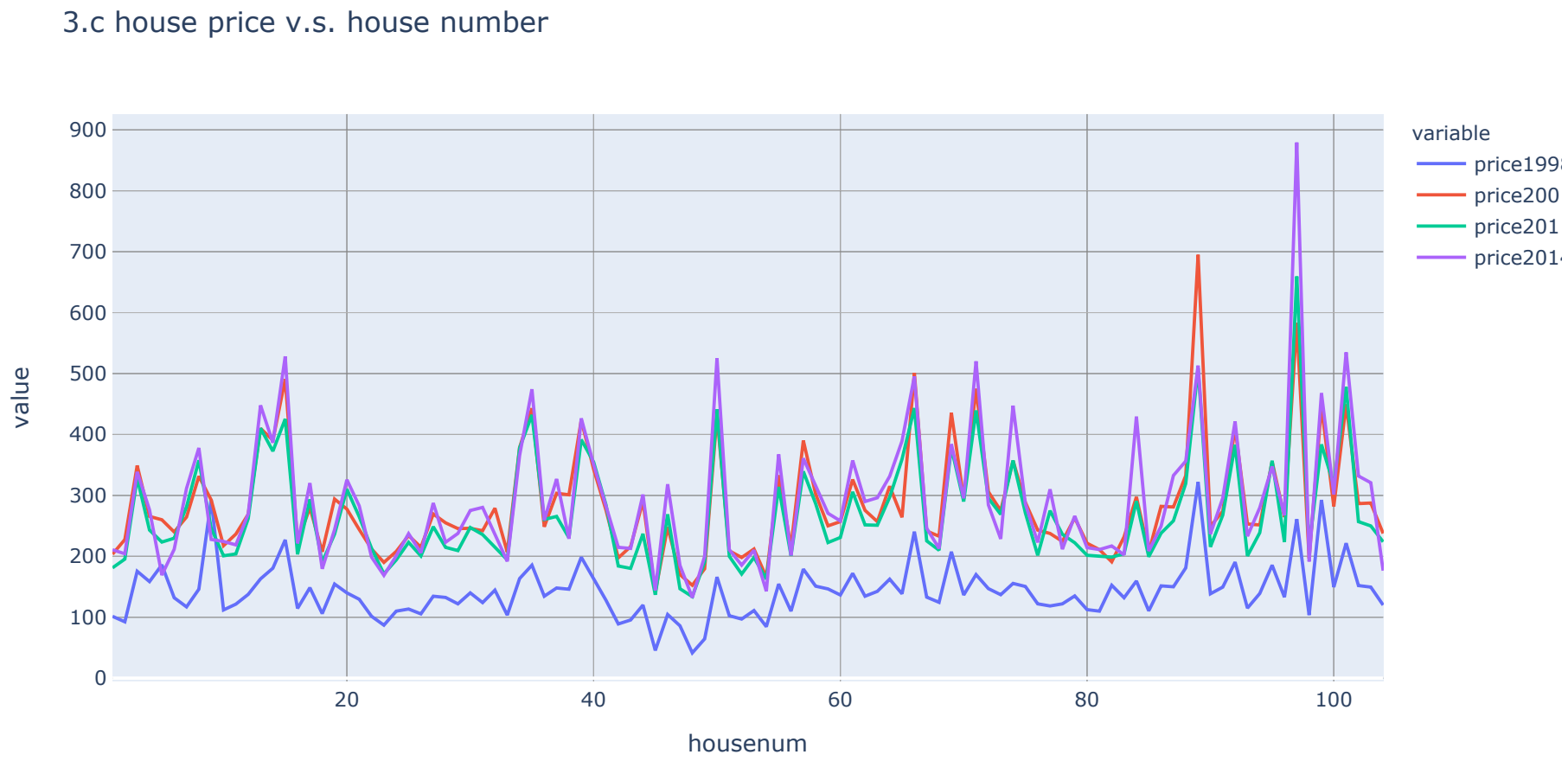
b. A histogram showing the number of houses per Zip code. The X-axis is the Zip code. Create an annotation pointing to the Zip code with the most houses.

In [13]:
```
fig_3b=px.histogram(house, x='zip', y='housenum',histfunc='count')
fig_3b.update_layout(title='3.b number of houses per zipcode',
                     xaxis=dict(type='category'),
                     annotations=[ dict( x="1062", y="61", xref='x', yref='y',
                                         text='max houses number 61',
                                         showarrow=True, arrowhead=7, ax=0, ay=-40)])
fig_3b.show()
# fig_3b.write_image('fig_3b.png')
```



c. A line chart with four lines: price1998, price2007, price2011, and price2014. The X-axis is the house num. The Y-axis is the value from the four columns listed above. When the mouse cursor hovers over a marker, the street address of the house should be displayed in the tooltip.

In [14]:
```
house['Address']=house['streetno'].map(str)+''+house['streetname'].map(str)
house_c=pd.melt(house, id_vars=['housenum','Address'],
                value_vars=['price1998','price2007','price2011','price2014'])
fig_3c=px.line(house_c, x=house_c['housenum'], y=house_c['value'],color='variable',
               hover_name='Address',
               title='3.c house price v.s. house number')
fig_3c.show()
# fig_3c.write_image('fig_3c.png')
```



1. Load wimbledons_champions.csv and use Seaborn to create the following charts.

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [16]:
```
champ=pd.read_csv('wimbledons_champions.csv')
champ.head()
# print(champ.info())
```

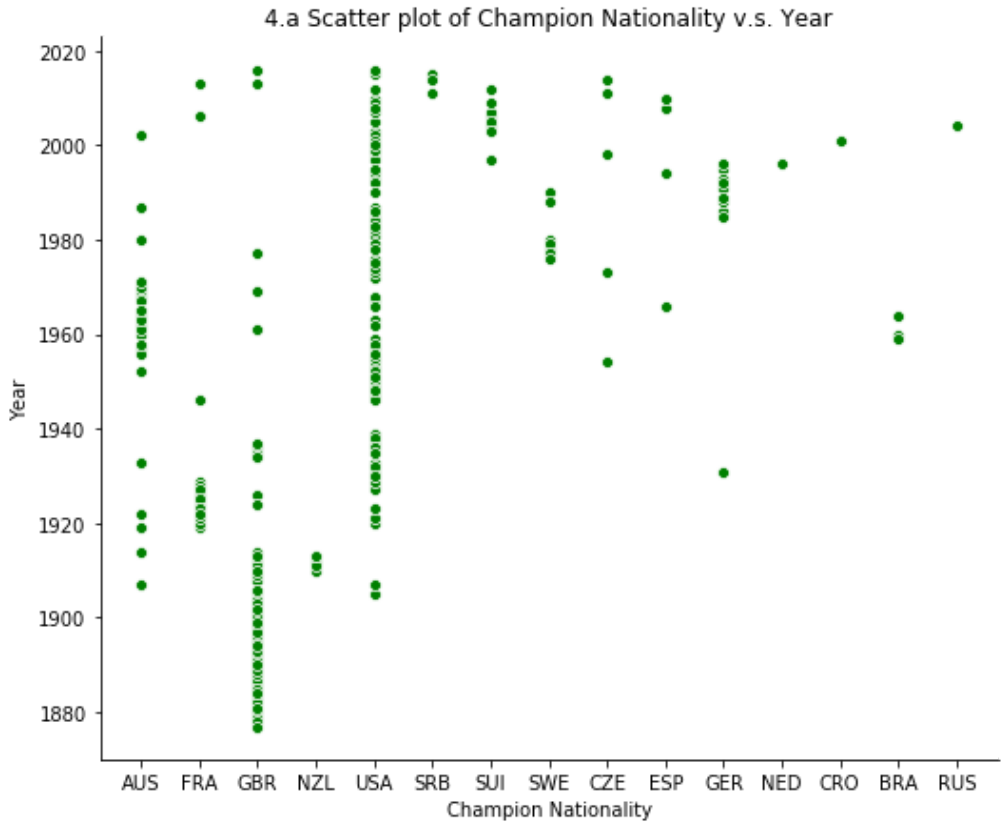
Out[16]:

	Gender	Champion	Mins	Runner-up Nationality	Champion Nationality	Runner-up	Score	Runner-up Seed	Champion Seed	Year	Runner-up Nationality (Men's)	Runner-Up
0	Men's	G.L. Patterson	NaN	AUS	AUS	N.E. Brookes	6-3, 7-5, 6-2	NaN	NaN	1919	NaN	NaN
1	Men's	G.L. Patterson	NaN	GBR	AUS	R. Lycett	6-3, 6-4, 6-2	NaN	NaN	1922	NaN	NaN
2	Men's	N.E. Brookes	NaN	GBR	AUS	A.W. Gore	6-4, 6-2, 6-2	NaN	NaN	1907	NaN	NaN
3	Men's	N.E. Brookes	NaN	NZL	AUS	A.F. Wilding	6-4, 6-4, 7-5	NaN	NaN	1914	NaN	NaN
4	Men's	J.R. Borotra	80.0	FRA	FRA	J.R. Lacoste	6-1, 3-6, 6-1, 3-6, 6-4	NaN	NaN	1924	NaN	NaN

a. A scatterplot showing when a player from different countries won the championship. The X-axis is the country. The Y-axis is the year. Each circle/dot indicates a player from a certain country won the championship in a certain year. The circles should be filled with green color.

In [17]:
```
fig_4a=sns.relplot(x='Champion Nationality', y='Year', height=6, aspect=1.25, color='green',data=champ)
fig_4a.set(title='4.a Scatter plot of Champion Nationality v.s. Year')
fig_4a
```

Out[17]: <seaborn.axisgrid.FacetGrid at 0x7f8d0a8ffc50>



b. Create a grid with four cells. In the first row, show two charts: a histogram showing the number of men's champions for different countries and a histogram showing the number of women's champions for different countries. In the second row, show two charts: a histogram showing the number of men's runners-up for different countries and a histogram showing the number of women's runners-up for different countries.

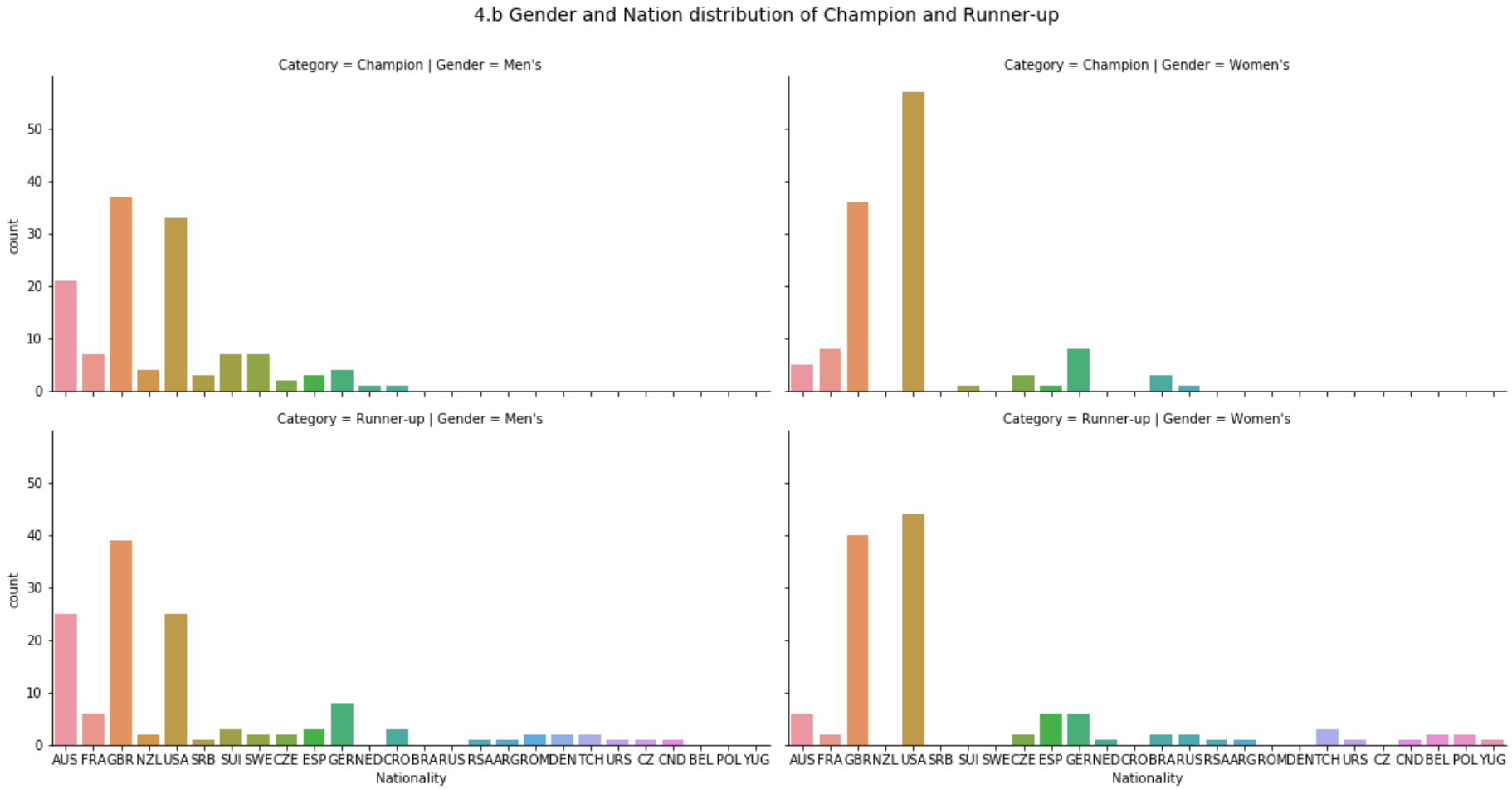
In [18]:
```
champ['Runner-up Nationality']=champ['Runner-up Nationality'].fillna(champ["Runner-up Nationality (Men's)"])
champ['Runner-up']=champ['Runner-up'].fillna(champ['Runner-up'])

champ1=pd.DataFrame({"Gender":champ["Gender"],"Nationality":champ["Champion Nationality"],"Category":"Champion"})
champ2=pd.DataFrame({"Gender":champ["Gender"],"Nationality":champ["Runner-up Nationality"],"Category":"Runner-up"})
champ_b=pd.concat([champ1,champ2])
champ_b.head()
```

Out[18]:

	Gender	Nationality	Category
0	Men's	AUS	Champion
1	Men's	AUS	Champion
2	Men's	AUS	Champion
3	Men's	AUS	Champion
4	Men's	FRA	Champion

In [19]:
```
f=sns.catplot(x='Nationality',row='Category', col='Gender',kind='count', height=4, aspect=2, data=champ_b)
fig_4b=f.fig
fig_4b.suptitle("4.b Gender and Nation distribution of Champion and Runner-up", y=1.05, fontsize=14)
```

Out[19]:
```
Text(0.5, 1.05, '4.b Gender and Nation distribution of Champion and Runner-up')
```



In [ ]:
```
```