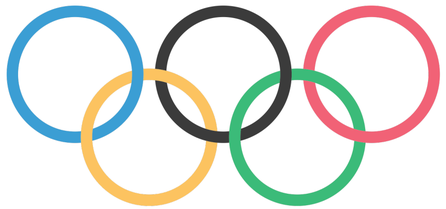
Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案

Demo 8 – Olympic



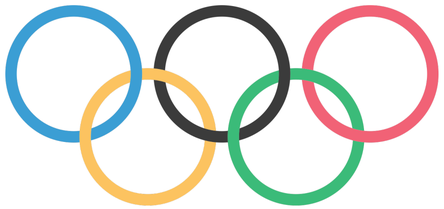
Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

Chapter Summary

- Scenario
- Data Import
- Medal Analysis
- Folium & Geopandas
- Df.replace()



Scenario

Over 11,000 athletes, with 47 disciplines, along with 743 Teams took part in the 2021(2020) Tokyo Olympics.

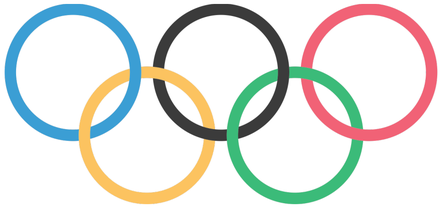
Our analysis goal is to find out the patterns of the winning teams and countries. Also we would like to know if the hosting countries may have advantage and improvement in getting medal.

Data Import

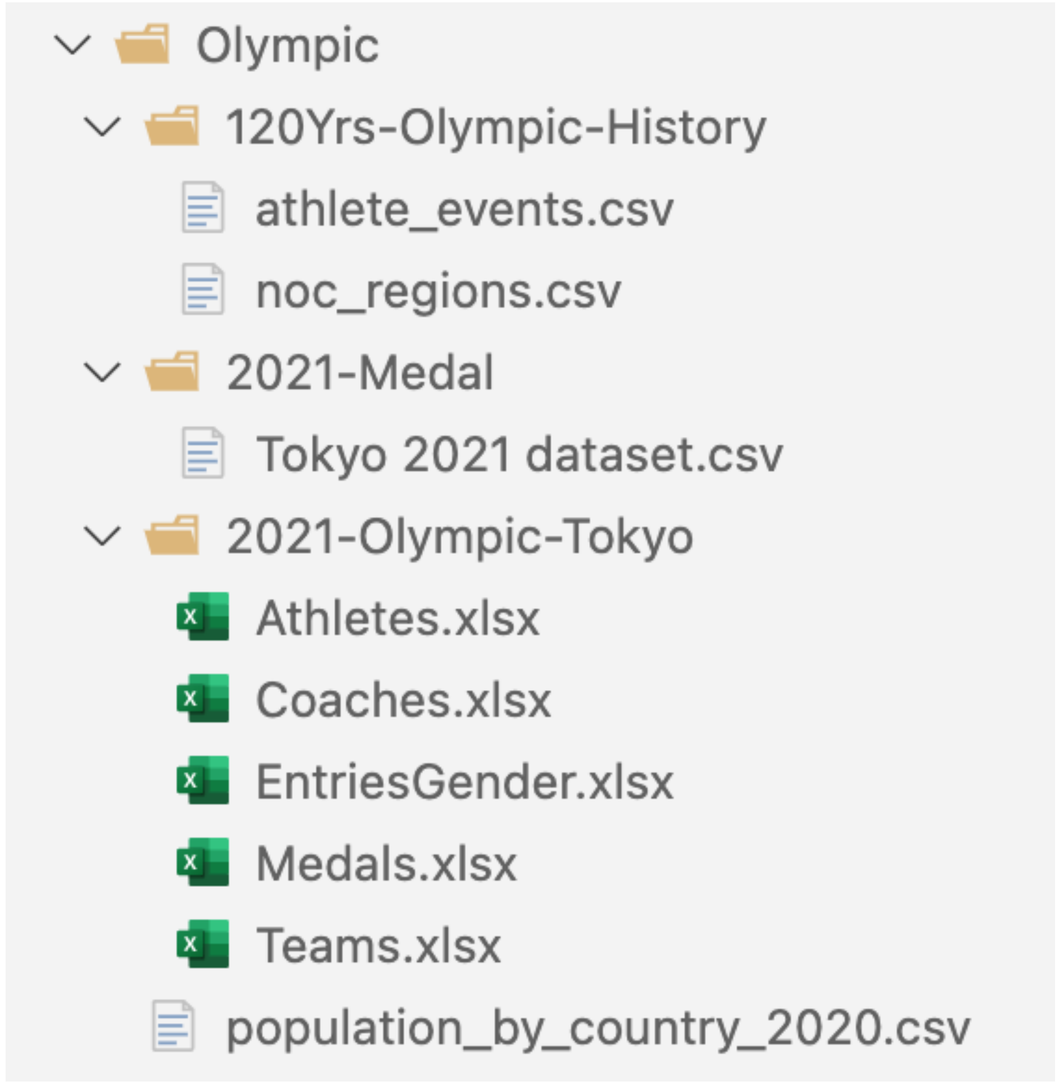
This dataset contains the details of the Athletes, Coaches, Teams participating as well as the Entries by gender. It contains their names, countries represented, discipline, gender of competitors, name of the coaches.

Will update the dataset with medals(gold, silver, bronze), more details about the athletes after few weeks.

Source: Tokyo Olympics 2020 Website



Dataset Structure



Import library

```
pip install folium
```

```
pip install geopandas
```

You may need to install folium and geopandas for map visualization.

```
import numpy as np
import pandas as pd

import folium
from folium import plugins
import geopandas as gpd
import branca

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from textwrap import wrap
from matplotlib.lines import Line2D
from matplotlib.offsetbox import OffsetImage, AnnotationBbox

from scipy import stats
from scipy.stats import ttest_ind
```

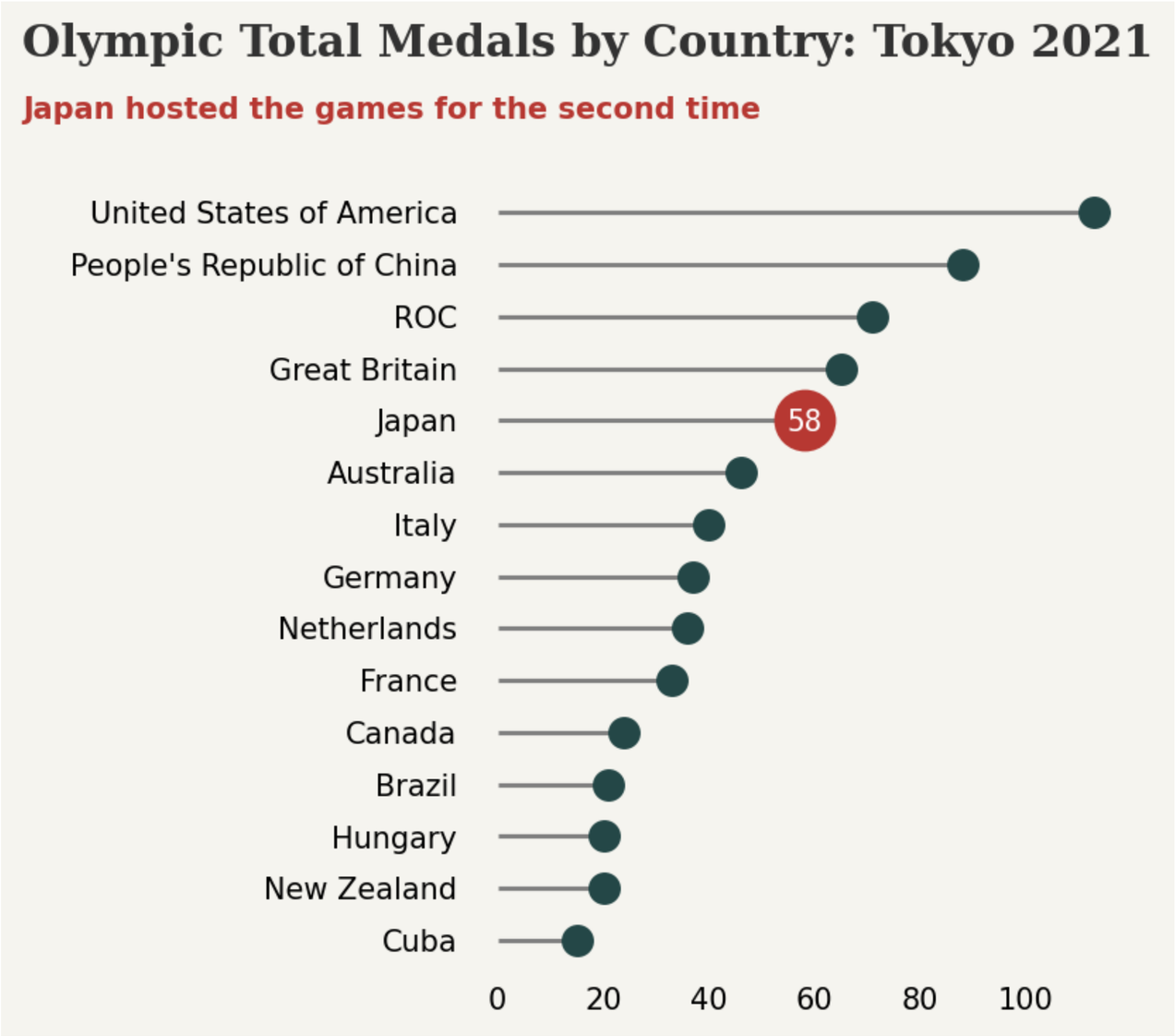
Setting up visual

```
18
19 plt.rcParams["font.family"] = "monospace"
20 plt.rcParams['figure.dpi'] = 150
21 background_color='#F5F4EF'
22
23 HEADER = '\033[95m'
24 OKBLUE = '\033[94m'
25 OKCYAN = '\033[96m'
26 OKGREEN = '\033[92m'
27 WARNING = '\033[93m'
28 FAIL = '\033[91m'
29 ENDC = '\033[0m'
30 BOLD = '\033[1m'
31 UNDERLINE = '\033[4m'
```

Plot the first graph

```
fig, ax = plt.subplots(figsize=(4, 5), facecolor=background_color)

temp = df_21_full[:15].sort_values(by='Total')
my_range=range(1,len(df_21_full[:15]['Team/NOC'])+1)

ax.set_facecolor(background_color)

plt.hlines(y=my_range, xmin=0, xmax=temp['Total'], color='gray')
plt.plot(temp['Total'], my_range, "o",markersize=10, color='#244747')
plt.plot(temp['Total'][2], my_range[10], "o",markersize=20,color='#B73832')

Xstart, Xend = ax.get_xlim()
Ystart, Yend = ax.get_ylim()

ax.tick_params(axis=u'both', which=u'both',length=0)
ax.set_xlabel("Total Medals",fontfamily='DejaVu Sans',loc='left',color='gray')
ax.set_axisbelow(True)

for s in ['top','right','bottom','left']:
    ax.spines[s].set_visible(False)

ax.text(-90,Yend+2.3, 'Olympic Total Medals by Country: Tokyo 2021', fontsize=15,
                              fontweight='bold',fontfamily='serif',color='#323232')
ax.text(-90,Yend+1.1, 'Japan hosted the games for the second time', fontsize=10,
                           fontweight='bold',fontfamily='sansserif',color='#B73832')
#ax.text(-100,Yend+1, 'Not Hosting', fontsize=10,fontweight='bold',fontfamily='sansserif',color='#244747')

plt.yticks(my_range, temp['Team/NOC'])
plt.xlabel('')

ax.annotate(temp['Total'][2], xy=(54.86,10.95), va = 'center', ha='left',fontweight='light',
            fontfamily='monospace',fontsize=10, color='white',rotation=0)
plt.show()
```

Olympic 2021 Tokyo



Top 15 medal countries

```
def highlight(nation):
    if nation['Team/NOC'] == 'Japan':
        return ['background-color: #f3f2f1']*6
    else:
        return ['background-color: white']*6

df_21_full[['Rank','Team/NOC','Bronze Medal','Silver Medal','Gold Medal','Total']]\
.iloc[:15].style.set_caption('Medals by Country: Summer Olympic Games sorted by Gold Medals [Top 15]')\
.bar(subset=['Gold Medal'], color='#f0c05a')\
.bar(subset=['Silver Medal'], color='Lightgray')\
.bar(subset=['Bronze Medal'], color='#a97142')\
.hide(axis="index").apply(highlight, axis=1)
```

The “\” means the code is not end but change other line for easy reading only.

Top 15 medal countries

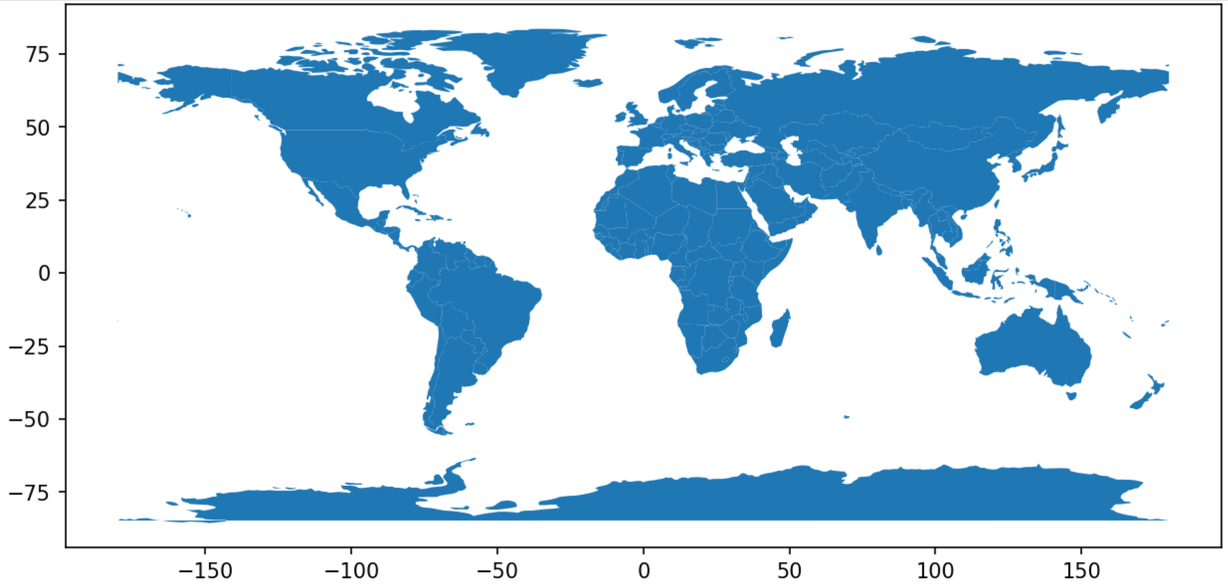
Medals by Country: Summer Olympic Games sorted by Gold Medals [Top 15]

Rank	Team/NOC	Bronze Medal	Silver Medal	Gold Medal	Total
1	United States of America	33	41	39	113
2	People's Republic of China	18	32	38	88
3	Japan	17	14	27	58
4	Great Britain	22	21	22	65
5	ROC	23	28	20	71
6	Australia	22	7	17	46
7	Netherlands	14	12	10	36
8	France	11	12	10	33
9	Germany	16	11	10	37
10	Italy	20	10	10	40
11	Canada	11	6	7	24
12	Brazil	8	6	7	21
13	New Zealand	7	6	7	20
14	Cuba	5	3	7	15
15	Hungary	7	7	6	20

Geographic visualization

```
# For geographic plotting
global_polygons = gpd.read_file(country_shapes)
global_polygons.to_file('global_polygons.geojson', driver = 'GeoJSON')

global_polygons.plot(figsize=(10,5)) # test map of the globe

# Tabular
df = pd.merge(df,regions,left_on='NOC',right_on='NOC')
df = df.query('Season == "Summer"') # Only interested in Summer Olympics for this project
```

Prepare for geographic plot.



Replace the long country name

```
#Replacing the country name with common values
df.replace('USA', "United States of America", inplace = True)
df.replace('Tanzania', "United Republic of Tanzania", inplace = True)
df.replace('Democratic Republic of Congo', "Democratic Republic of the Congo", inplace = True)
df.replace('Congo', "Republic of the Congo", inplace = True)
df.replace('Lao', "Laos", inplace = True)
df.replace('Syrian Arab Republic', "Syria", inplace = True)
df.replace('Serbia', "Republic of Serbia", inplace = True)
df.replace('Czechia', "Czech Republic", inplace = True)
df.replace('UAE', "United Arab Emirates", inplace = True)
df.replace('UK', "United Kingdom", inplace = True)

population.replace('United States', "United States of America", inplace = True)
population.replace('Czech Republic (Czechia)', "Czech Republic", inplace = True)
population.replace('DR Congo', "Democratic Republic of the Congo", inplace = True)
population.replace('Serbia', "Republic of Serbia", inplace = True)
population.replace('Tanzania', "United Republic of Tanzania", inplace = True)

df_21_full.replace('Great Britain', "United Kingdom", inplace = True)
df_21_full.replace("People's Republic of China", "China", inplace = True)
df_21_full.replace("ROC", "Russia", inplace = True)
```

Map countries to cities

```python
# Function to map country to city
def host_country(col):
    if col == "Rio de Janeiro":
        return "Brazil"
    elif col == "London":
        return "United Kingdom"
    elif col == "Beijing":
        return  "China"
    elif col == "Athina":
        return  "Greece"
    elif col == "Sydney" or col == "Melbourne":
        return  "Australia"
    elif col == "Atlanta" or col == "Los Angeles" or col == "St. Louis":
        return  "United States of America"
    elif col == "Barcelona":
        return  "Spain"
    elif col == "Seoul":
        return  "South Korea"
    elif col == "Moskva":
        return  "Russia"
    elif col == "Montreal":
        return  "Canada"
    elif col == "Munich" or col == "Berlin":
        return  "Germany"
    elif col == "Mexico City":
        return  "Mexico"
    elif col == "Tokyo":
        return  "Japan"
    elif col == "Roma":
        return  "Italy"
    elif col == "Paris":
        return  "France"
    elif col == "Helsinki":
        return  "Finland"
    elif col == "Amsterdam":
        return  "Netherlands"
    elif col == "Antwerpen":
        return  "Belgium"
    elif col == "Stockholm":
        return  "Sweden"
    else:
        return "Other"

# Applying this function
df['Host_Country'] = df['City'].apply(host_country)
```

Create new DF to contain the hosting nation data

```
df_new = df.groupby(['Year','Host_Country','region','Medal']
                    )['Medal'].count().unstack().fillna(0).astype(int).reset_index()

df_new['Is_Host'] = np.where(df_new['Host_Country'] == df_new['region'],1,0)
df_new['Total Medals'] = df_new['Bronze'] + df_new['Silver'] + df_new['Gold']
```

```
# Preparing to add 2021 data to our historic df
df_21_full_refined = df_21_full[['Team/NOC', "Gold Medal", "Silver Medal", "Bronze Medal"]]
df_21_full_refined.loc[df_21_full_refined.index,['Total Medals']] = df_21_full_refined[["Gold Medal",
                                                                  "Silver Medal", "Bronze Medal"]].sum(axis=1)
df_21_full_refined.loc[df_21_full_refined.index,['Year']] = 2021

df_21_full_refined = df_21_full_refined.rename(columns={'Gold Medal':'Gold', 'Silver Medal':'Silver',
                                                        'Bronze Medal':'Bronze'})


df_21_full_refined['Is_Host'] = np.where(df_21_full_refined['Team/NOC'] == 'Japan',1,0)
df_21_full_refined['Host_Country'] = 'Japan'
df_21_full_refined = df_21_full_refined.rename(columns={'Team/NOC':'region'})

# Adding 2021 data to historic
df_new = pd.concat([df_new, df_21_full_refined],axis = 0)

# Removing Russia as many Olympic games were competed in as the Soviet Union,
# containing several modern day nations
df_new = df_new.query("region != 'Russia' | region != 'ROC'")
```

Distribution of medal on map

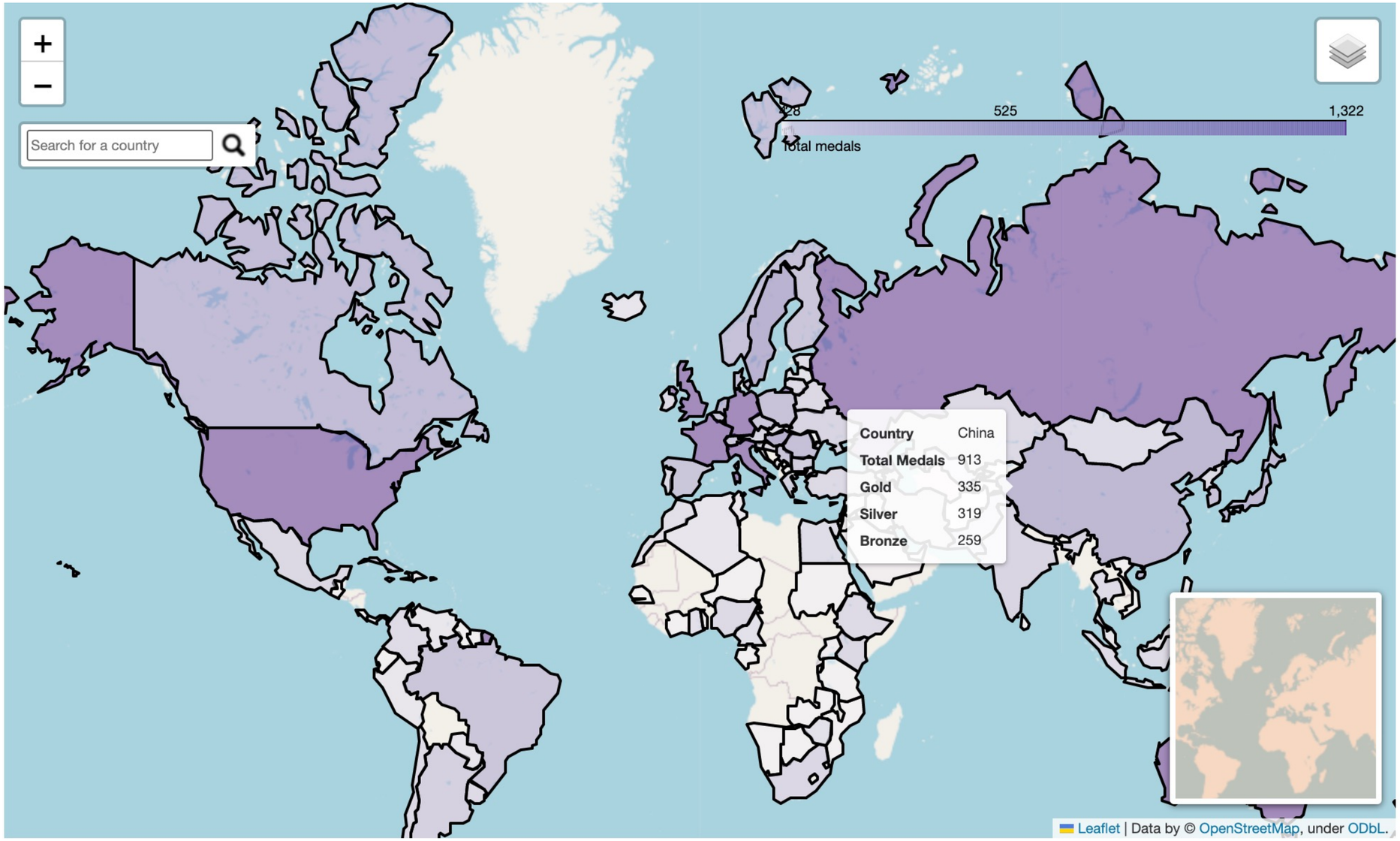
```
host_list = list(df_new.query("Is_Host == 1")['Host_Country'].value_counts().index)

medals_temp = medals.reset_index()
medals_map = pd.merge(global_polygons, medals_temp, left_on='name',right_on='region')
medals_map['Hosted'] = np.where(medals_map['name'].isin(host_list),1,0)
```

```
def rd2(x):
    return round(x, 2)

Temp = medals_map

minimum, maximum = Temp["Total"].quantile([0.05, 0.95]).apply(rd2)
mean = round(Temp["Total"].mean(), 2)

colormap = branca.colormap.LinearColormap(
    colors=["#f2f0f7", "#cbc9e2", "#9e9ac8", "#756bb1", "#54278f"],
    index=Temp["Total"].quantile([0.25, 0.5, 0.85, 0.95]),
    vmin=minimum,
    vmax=maximum,
)

colormap.caption = "Total medals"

from folium.plugins import Search
from folium.plugins import HeatMap

m = folium.Map(location=(50,0),zoom_start=3)

```

Distribution of medal on map

```
23 def style_function(x):
24     return { "fillColor": colormap(x["properties"]["Total"]),
25                 "color": "black",
26                 "weight": 2,
27                 "fillOpacity": 0.5, }
28 
29 Map_Layer = folium.GeoJson(
30     Temp,
31     name="Medals",
32     style_function=style_function,
33     tooltip=folium.GeoJsonTooltip(
34         fields=["name","Total","Gold","Silver","Bronze"], aliases=["Country","Total Medals",
35                                                                    "Gold","Silver","Bronze"], localize=True ),
36 ).add_to(m)
37 
38 plugins.Search(Map_Layer,position='topleft',
39                            search_zoom=5,placeholder="Search for a country",weight=3,
40                            search_label='region',
41                            geom_type='Polygon').add_to(m)
42 
43 minimap = plugins.MiniMap()
44 m.add_child(minimap)
45 
46 folium.LayerControl().add_to(m)
47 colormap.add_to(m)
48 
49 m
```

Distribution of medal on map



Top 15 Medal Countries in all year history

```
medals = df.groupby(['region','Medal'])['Medal'].count().unstack().fillna(0).astype(int)

medals['Total'] = medals['Bronze'] + medals['Silver'] + medals['Gold']

medals = medals[['Bronze','Silver','Gold','Total']].sort_values(by='Total', ascending=False)

medals.iloc[:15].style.set_caption('Medals by Country: Summer Olympic Games [Top 15]')\
.bar(subset=['Gold'], color='#f0c05a')\
.bar(subset=['Silver'], color='Lightgray')\
.bar(subset=['Bronze'], color='#a97142')\
.background_gradient(subset=['Total'], cmap='BuGn')
```

Top 15 Medal Countries in all year history

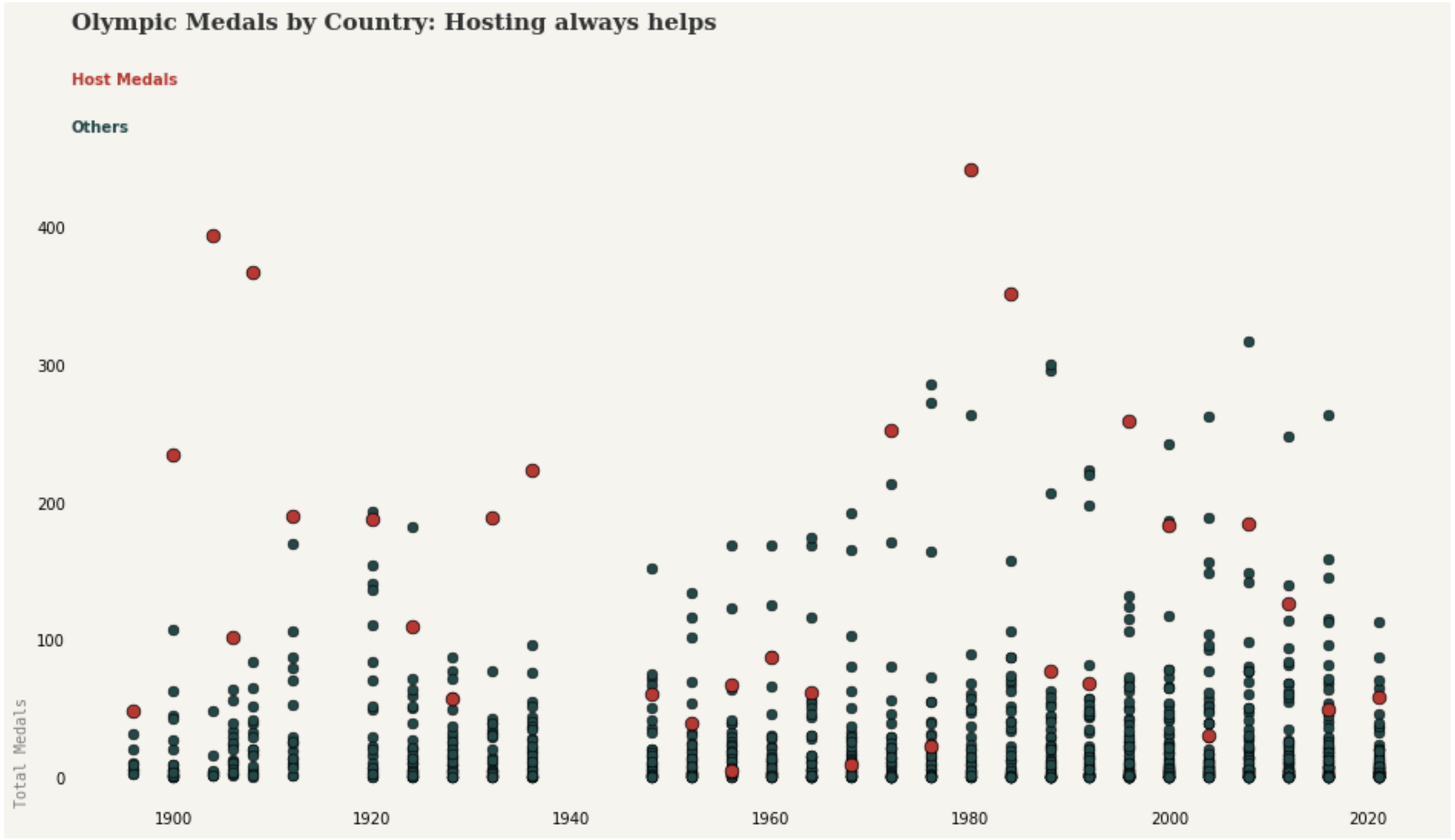
Medals by Country: Summer Olympic Games [Top 15]

Medal region	Bronze	Silver	Gold	Total
United States of America	1197	1333	2472	5002
Russia	994	974	1220	3188
Germany	1064	987	1075	3126
United Kingdom	620	729	636	1985
France	587	575	465	1627
Italy	454	474	518	1446
Australia	515	456	362	1333
Hungary	363	328	432	1123
Sweden	358	396	354	1108
Netherlands	371	302	245	918
China	259	319	335	913
Japan	333	287	230	850
Canada	344	239	158	741
Romania	290	200	161	651
Denmark	177	236	179	592

Medal by countries with host nation marked

```
fig, ax = plt.subplots(1,1, figsize=(16,8), facecolor=background_color)

sns.scatterplot(data=df_new.query("Is_Host == 0"), x='Year', y='Total Medals', s=45,
                ec='black', color='#244747',ax=ax)
sns.scatterplot(data=df_new.query("Is_Host == 1"), x='Year', y='Total Medals', s=75,
                ec='black', color='#B73832',ax=ax)

Xstart, Xend = ax.get_xlim()
Ystart, Yend = ax.get_ylim()

ax.tick_params(axis=u'both', which=u'both',length=0)
ax.set_ylabel("Total Medals",fontfamily='monospace',loc='bottom',color='gray')
ax.set_xlabel("")

ax.set_facecolor(background_color)
ax.set_axisbelow(True)


for s in ['top','right','bottom','left']:
    ax.spines[s].set_visible(False)


ax.text(Xstart,Yend+80, 'Olympic Medals by Country: Hosting always helps', fontsize=15,
        fontweight='bold',fontfamily='serif',color='#323232')
ax.text(Xstart,Yend+40, 'Host Medals', fontsize=10,fontweight='bold',color='#B73832')
ax.text(Xstart,Yend+5, 'Others', fontsize=10,fontweight='bold',color='#244747')

plt.show()
```

Medal by countries with host nation marked



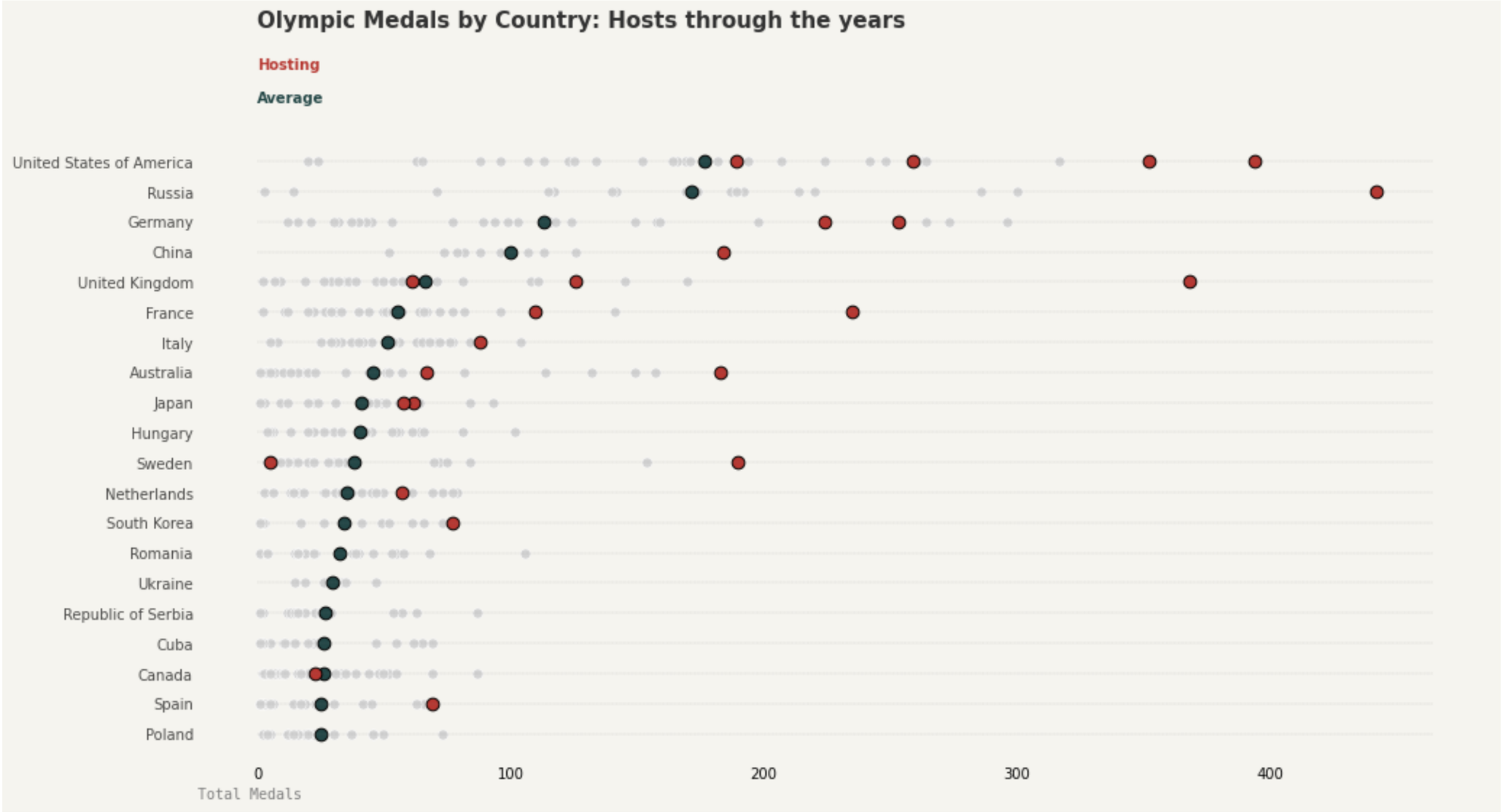
Olympic Medals by Country: Hosts through the years

```
fig, ax = plt.subplots(1,1, figsize=(16, 8), facecolor=background_color)

# top 20
top_list_ = df_new.groupby('region')['Total Medals'].mean().sort_values(ascending=False)\
.reset_index()[:20].sort_values(by='Total Medals',ascending=True)

plot = 1
for country in top_list_['region']:
    mean = df_new[df_new['region'] == country].groupby('region')['Total Medals'].mean()
    # historic scores
    sns.scatterplot(data=df_new[df_new['region'] == country], y=plot, x='Total Medals',
                    color='lightgray',s=50,ax=ax)
    # mean score
    sns.scatterplot(data=df_new[df_new['region'] == country], y=plot, x=mean,color='#244747',
                    ec='black',linewidth=1,s=75,ax=ax)
    # Hosting score
    sns.scatterplot(data=(df_new[(df_new['region'] == country) & (df_new['Is_Host'] == 1)]),
                    y=plot, x='Total Medals',color='#B73832',ec='black',linewidth=1,s=75,ax=ax)
    plot += 1

```

Olympic Medals by Country: Hosts through the years

```
Xstart, Xend = ax.get_xlim()
Ystart, Yend = ax.get_ylim()

ax.set_yticks(top_list_.index+1)
ax.set_yticklabels(top_list_['region'][::-1], fontdict={'horizontalalignment': 'right'}, alpha=0.7)
ax.tick_params(axis=u'both', which=u'both',length=0)
ax.set_xlabel("Total Medals",fontfamily='monospace',loc='left',color='gray')
ax.set_facecolor(background_color)
ax.hlines(y=top_list_.index+1, xmin=0, xmax=Xend, color='gray', alpha=0.5, linewidth=.3, linestyles='--')
ax.set_axisbelow(True)

for s in ['top','right','bottom','left']:
    ax.spines[s].set_visible(False)

ax.text(0,Yend+3.5, 'Olympic Medals by Country: Hosts through the years', fontsize=15,
                                                        fontweight='bold', color='#323232')
ax.text(0,Yend+2.1, 'Hosting', fontsize=10,fontweight='bold', color='#B73832')
ax.text(0,Yend+1, 'Average', fontsize=10,fontweight='bold', color='#244747')

plt.show()
```

Olympic Medals by Country: Hosts through the years



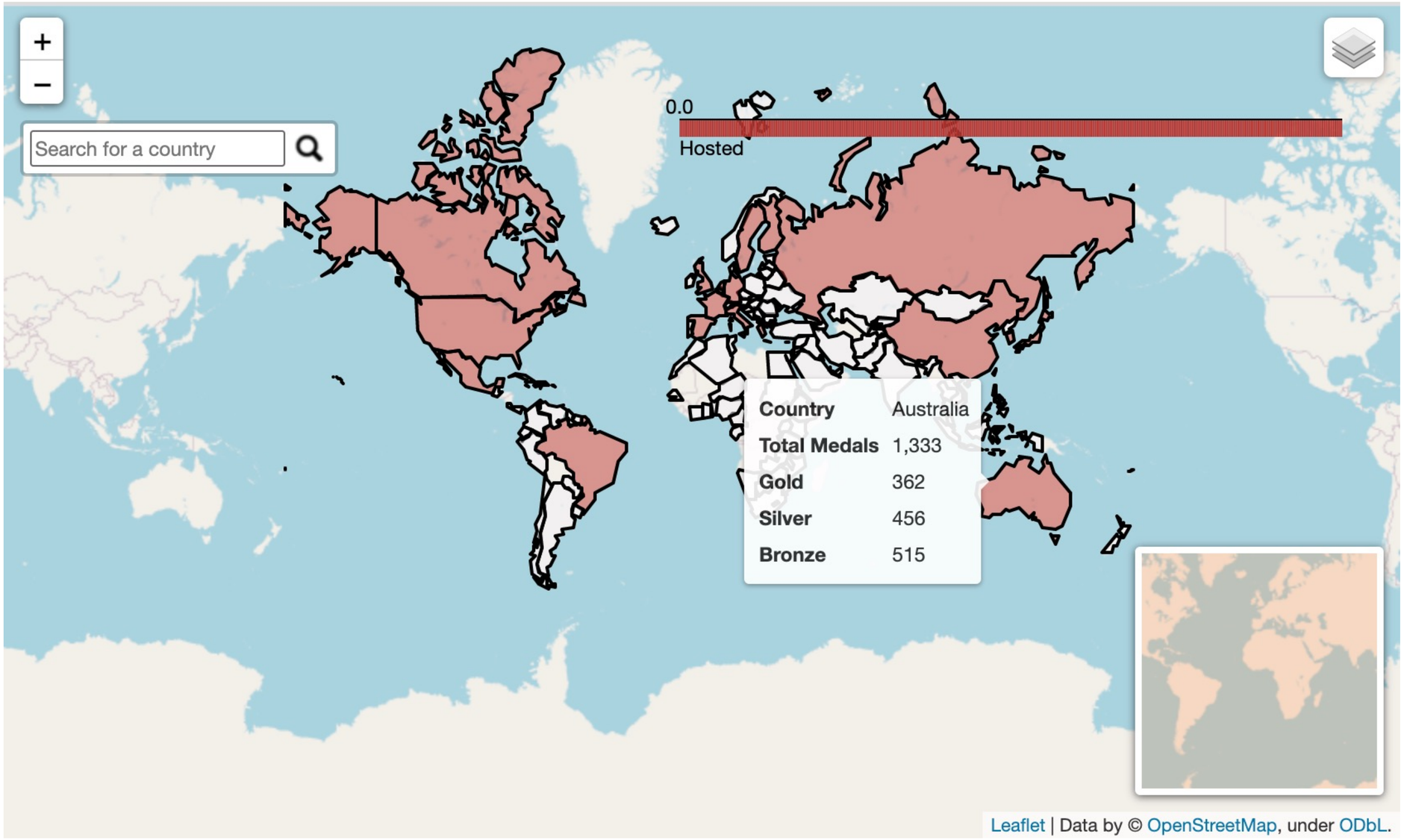
Host countries on map with total medal over years

```
Temp = medals_map

colormap = branca.colormap.LinearColormap(
    colors=["#f2f0f7",  "#B73832"],
    index=Temp["Hosted"].quantile([0.05, 0.5]),
    vmin=0,
    vmax=1, )

colormap.caption = "Hosted"

from folium.plugins import Search
from folium.plugins import HeatMap

m = folium.Map(location=(50,0),zoom_start=3)

def style_function(x):
    return { "fillColor": colormap(x["properties"]["Hosted"]),
                "color": "black",
                "weight": 2,
                "fillOpacity": 0.5,   }

```

Host countries on map with total medal over years

```
Map_Layer = folium.GeoJson(
    Temp,
    name="Hosted",
    style_function=style_function,
    tooltip=folium.GeoJsonTooltip(
        fields=["name","Total","Gold","Silver","Bronze"], aliases=["Country",
                                "Total Medals","Gold","Silver","Bronze"], localize=True
    ),
).add_to(m)

plugins.Search(Map_Layer,position='topleft',
                            search_zoom=5,placeholder="Search for a country",weight=3,
                            search_label='region',
                            geom_type='Polygon').add_to(m)

minimap = plugins.MiniMap()
m.add_child(minimap)

folium.LayerControl().add_to(m)
colormap.add_to(m)

m
```

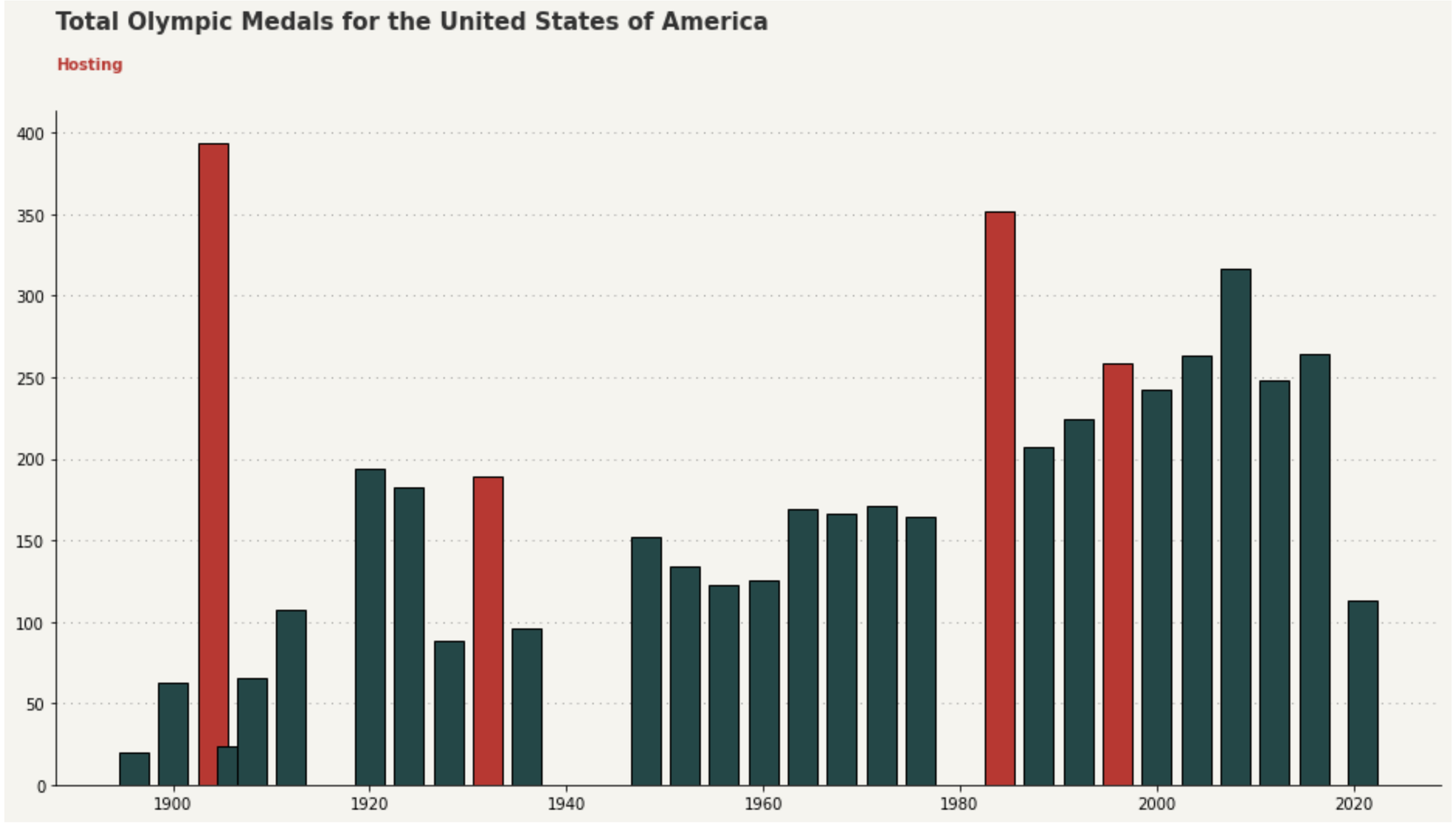
Host countries on map with total medal over years



View the USA's medal tally over time

```
temp = df_new.query("region == 'United States of America'")

fig, ax = plt.subplots(1, 1, figsize=(16, 8), facecolor=background_color)

color = ['#B73832' if i == 1 else '#244747' for i in temp['Is_Host']]

ax.bar(temp['Year'], temp['Total Medals'],width=3, color=color, ec='black')
ax.set_facecolor(background_color)
ax.grid(which='both', axis='y', zorder=5, color='gray', linestyle=':', dashes=(1,5))
ax.set_axisbelow(True)

for s in ['top', 'right']:
    ax.spines[s].set_visible(False)

Xstart, Xend = ax.get_xlim()
Ystart, Yend = ax.get_ylim()

ax.text(Xstart,Yend+50, 'Total Olympic Medals for the United States of America', fontsize=15,
        fontweight='bold',color='#323232')
ax.text(Xstart,Yend+25, 'Hosting', fontsize=10,fontweight='bold',color='#B73832')

plt.show()
```

View the USA's medal tally over time



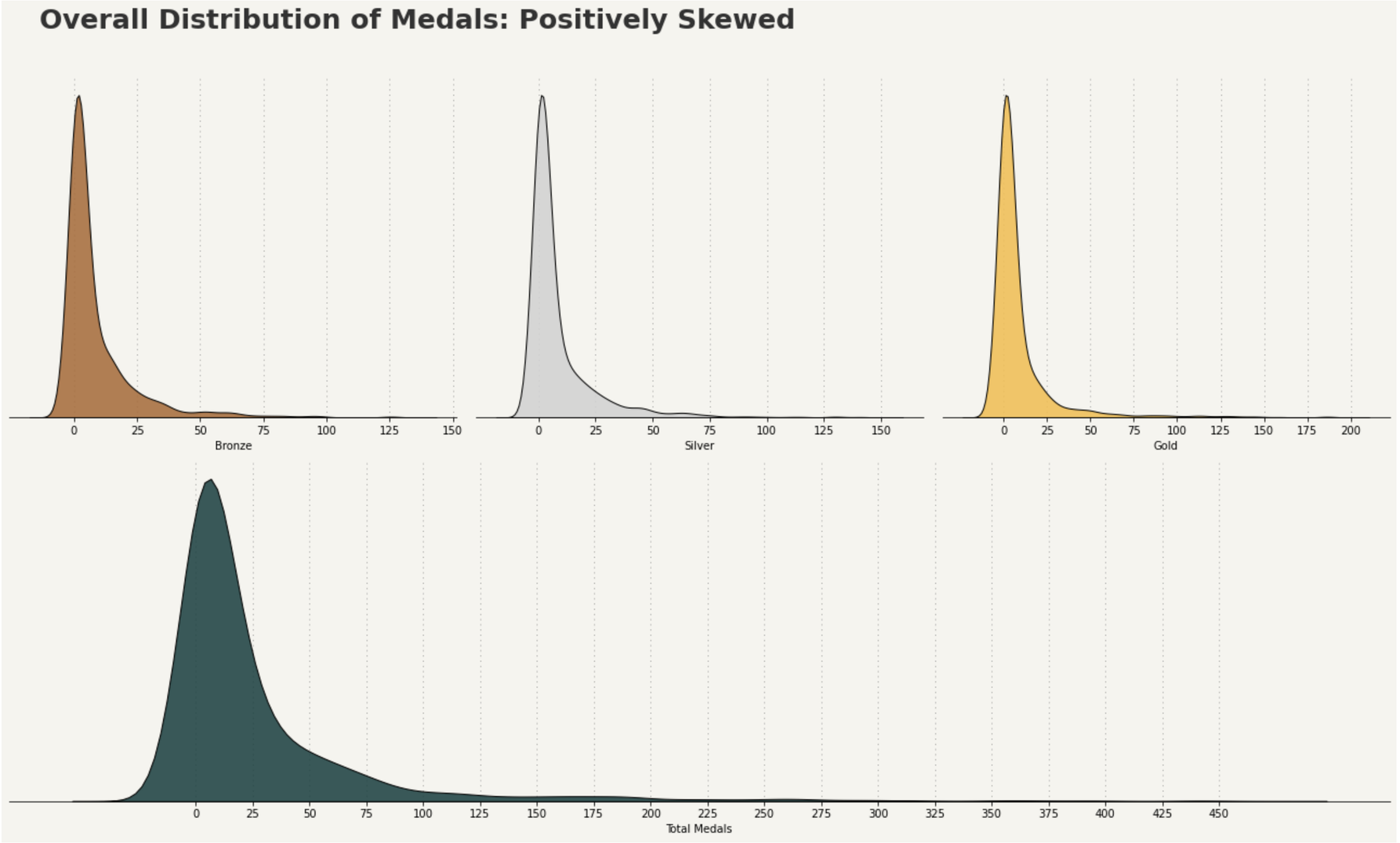
Overall medal distribution

```
fig = plt.figure(figsize=(18,10), facecolor=background_color)

colors = ['#a97142', 'lightgray', '#f0c05a', '#244747']

num = 0
variables = ['Bronze', 'Silver', 'Gold', 'Total Medals']

plots = [(0,0), (0,1), (0,2), (1,0)]

data = df_new

for item in variables:
    if plots[num] == plots[-1]:
        colspan=3
    else:
        colspan=1

    plt.subplot2grid((2,3), (plots[num]), colspan=colspan)

    ax = sns.kdeplot(data=data, x=item,color=colors[num],fill=True,  alpha=0.9, ec='black', cut=5)
    ax.set_facecolor(background_color)
    ax.tick_params(axis='y', left=False)
    ax.get_yaxis().set_visible(False)
    ax.set_axisbelow(True)
    ax.set_xlabel(item)
    ax.grid(which='major', axis='x', zorder=5, color='gray', linestyle=':', dashes=(1,5))
    for s in ["top","right","left"]:
        ax.spines[s].set_visible(False)

    plt.xticks(list(np.arange(0,data[item].max()+25,25)))
    num +=1

```

Overall medal distribution

```
33
34 Xstart, Xend = ax.get_xlim()
35 Ystart, Yend = ax.get_ylim()
36
37 fig.text(0.03,1.05, 'Overall Distribution of Medals: Positively Skewed', fontsize=25,
38           fontweight='bold',color='#323232')
39
40 plt.tight_layout()
41 plt.show()
```

Overall medal distribution



Statistics

We found that the medal distribution is not normal distributed. Therefore we use natural log to reduce the positive skewed effect.

```
host_list = list(df_new.query("Is_Host == 1")['Host_Country'].value_counts().index)

df_new['Total_Medals_NaturalLog'] = np.log(df_new['Total Medals'])

Not_hosting_samples = df_new[df_new['region'].isin(host_list)]\
.query("Is_Host == 0")['Total_Medals_NaturalLog'].count()

hosting_samples = df_new[df_new['region'].isin(host_list)]\
.query("Is_Host == 1")['Total_Medals_NaturalLog'].count()

Not_hosting_mean = df_new[df_new['region'].isin(host_list)]\
.query("Is_Host == 0")['Total_Medals_NaturalLog'].mean()

hosting_mean = df_new[df_new['region'].isin(host_list)]\
.query("Is_Host == 1")['Total_Medals_NaturalLog'].mean()

Not_hosting_std = df_new[df_new['region'].isin(host_list)]\
.query("Is_Host == 0")['Total_Medals_NaturalLog'].std()

hosting_std = df_new[df_new['region'].isin(host_list)].\
query("Is_Host == 1")['Total_Medals_NaturalLog'].std()
```

Statistics

```
print(color_font.S+"Key Statistics"+color_font.E)
print("Mean Natural Log of Total Medals when Not Hosting:\n",Not_hosting_mean)
print("Mean Natural Log of Total Medals when Hosting:\n",hosting_mean,"\n")


Not_hosting_std_error = Not_hosting_std/np.sqrt(Not_hosting_samples)
print("Standard Error of Natural Log of Total Medals when Not Hosting:\n",Not_hosting_std_error)

hosting_std_error = hosting_std/np.sqrt(hosting_samples)
print("Standard Error of Natural Log of Total Medals when Not Hosting:\n",hosting_std_error)
```

```
Key Statistics
Mean Natural Log of Total Medals when Not Hosting:
 3.086281690114543
Mean Natural Log of Total Medals when Hosting:
 4.566512430214569

Standard Error of Natural Log of Total Medals when Not Hosting:
 0.06927049040753175
Standard Error of Natural Log of Total Medals when Not Hosting:
 0.19153671012884035
```

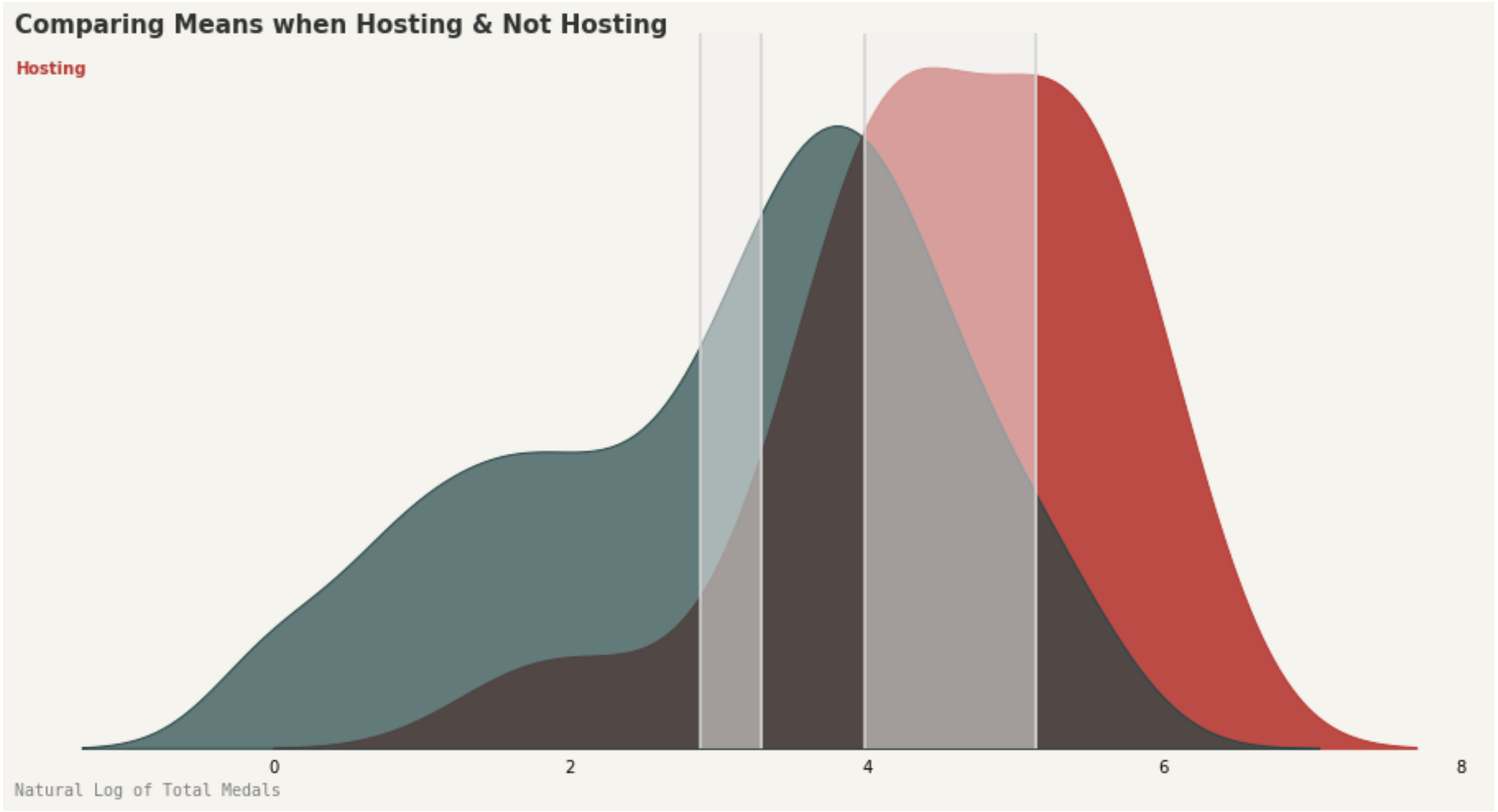
Comparing Means when Hosting & Not Hosting

```
fig, ax = plt.subplots(1,1, figsize=(18,6), facecolor=background_color)
ax.set_facecolor(background_color)
sns.kdeplot(data=df_new[df_new['region'].isin(host_list)].query("Is_Host == 1"),
            x='Total_Medals_NaturalLog',fill=True,color='#B73832',alpha=0.9, ax=ax)
sns.kdeplot(data=df_new[df_new['region'].isin(host_list)].query("Is_Host == 0"),
            x='Total_Medals_NaturalLog', fill=True, color='#244747',alpha=0.7, ax=ax)

ax.axvline(color='lightgray',x=Not_hosting_mean+(3*Not_hosting_std_error))
ax.axvline(color='lightgray',x=Not_hosting_mean-(3*Not_hosting_std_error))
ax.axvspan(Not_hosting_mean-(3*Not_hosting_std_error), Not_hosting_mean+(3*Not_hosting_std_error),
           alpha=0.5, color='#f3f2f1')

#ax.axvline(x=Not_hosting_mean)

#ax.axvline(x=hosting_mean)
ax.axvline(color='lightgray',x=hosting_mean+(3*hosting_std_error))
ax.axvline(color='lightgray',x=hosting_mean-(3*hosting_std_error))
ax.axvspan(hosting_mean-(3*hosting_std_error), hosting_mean+(3*hosting_std_error),
           alpha=0.5, color='#f3f2f1')

```

Comparing Means when Hosting & Not Hosting

```
21 for s in ['top','left', 'bottom', 'right']:
22     ax.spines[s].set_visible(False)
23
24 ax.set_xlabel("Natural Log of Total Medals",fontfamily='monospace',loc='left',color='gray')
25 ax.get_yaxis().set_visible(False)
26 ax.tick_params(axis = 'both', which = 'major', labelsize = 10)
27
28 ax.tick_params(axis='both', which='both',left=False, bottom=False,labelbottom=True)
29
30 Xstart, Xend = ax.get_xlim()
31 Ystart, Yend = ax.get_ylim()
32
33 ax.text(Xstart,Yend, 'Comparing Means when Hosting & Not Hosting', fontsize=15,
34         fontweight='bold',color='#323232')
35 ax.text(Xstart,Yend-0.02, 'Hosting', fontsize=10,fontweight='bold',color='#B73832')
36
37 plt.show()
```

Comparing Means when Hosting & Not Hosting



Chapter Wrap Up

In this chapter, we had tried a few new visualization tools, such as folium and geopandas. Some of the code would be long and difficult to understand in short time. However, please remember you can always get help from search engine or OpenAi.

And there are many sample coding from those library documentation.

Once again, PRACTICE MADE PERFECT!

Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:
https://plotly.com/python/graph-objects/

Kaggle dataset:
https://www.kaggle.com/datasets

WorldBank API:

- https://blogs.worldbank.org/opendata/introducing-wbgapi-new-python-package-accessing-world-bank-data
- https://nbviewer.org/github/tgherzog/wbgapi/blob/master/examples/wbgapi-cookbook.ipynb
- https://pypi.org/project/wbgapi/

GitHub Open Source Code:
https://github.com/plotly/plotly.py