Olympic Swimming History 1912 - 2020

Introduction Visualizations Conclusion

In [1]:

```
#Import necessary Libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.patches import Patch
from matplotlib.ticker import PercentFormatter
from IPython.display import display
import seaborn as sns
import os
```

In [2]:

```
import random
import datetime as datetime
import matplotlib.dates as dates
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from contextlib import contextmanager
from time import time
from tqdm import tqdm
import lightgbm as lgbm
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
```

In [3]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:

```
from IPython.core.display import HTML
HTML("""
<style>
.output_png {
    display: table-cell;
    text-align: center;
    vertical-align: middle;
}
</style>
""")
```

Out[4]:

In [22]:

```
import plotly.express as px

import plotly.io as pio
pio.renderers.default = 'notebook'

!pip install Pyppeteer
!pyppeteer-install
```

```
Requirement already satisfied: Pyppeteer in c:\users\hamza\anaconda3\lib\site-packages
```

```
(1.0.2)
Requirement already satisfied: certifi>=2021 in c:\users\hamza\anaconda3\lib\site-packag
es (from Pyppeteer) (2023.5.7)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\hamza\anaconda3\lib\si
te-packages (from Pyppeteer) (1.26.14)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\hamza\anaconda3\lib\site-
packages (from Pyppeteer) (4.64.1)
Requirement already satisfied: websockets<11.0,>=10.0 in c:\users\hamza\anaconda3\lib\si
te-packages (from Pyppeteer) (10.4)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\hamza\anaconda3\lib\s
ite-packages (from Pyppeteer) (4.11.3)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\hamza\anaconda3\lib\sit
e-packages (from Pyppeteer) (1.4.4)
Requirement already satisfied: pyee<9.0.0,>=8.1.0 in c:\users\hamza\anaconda3\lib\site-p
ackages (from Pyppeteer) (8.2.2)
Requirement already satisfied: zipp>=0.5 in c:\users\hamza\anaconda3\lib\site-packages
(from importlib-metadata>=1.4->Pyppeteer) (3.11.0)
Requirement already satisfied: colorama in c:\users\hamza\anaconda3\lib\site-packages (f
rom tqdm<5.0.0,>=4.42.1->Pyppeteer) (0.4.6)
```

```
[INFO] Starting Chromium download.

  0%|          | 0.00/137M [00:00<?, ?b/s]
  0%|          | 666k/137M [00:00<00:20, 6.64Mb/s]
  2%|2         | 3.32M/137M [00:00<00:07, 18.2Mb/s]
  5%|4         | 6.47M/137M [00:00<00:05, 24.2Mb/s]
  7%|6         | 9.48M/137M [00:00<00:04, 26.5Mb/s]
  9%|8         | 12.3M/137M [00:00<00:04, 27.0Mb/s]
 11%|#1        | 15.4M/137M [00:00<00:04, 28.3Mb/s]
 14%|#3        | 18.6M/137M [00:00<00:04, 29.4Mb/s]
 16%|#6        | 21.9M/137M [00:00<00:03, 30.7Mb/s]
 18%|#8        | 25.0M/137M [00:00<00:03, 30.1Mb/s]
 20%|##        | 28.0M/137M [00:01<00:03, 28.8Mb/s]
 23%|##2       | 30.9M/137M [00:01<00:03, 28.6Mb/s]
 25%|##4       | 34.2M/137M [00:01<00:03, 29.7Mb/s]
 27%|##7       | 37.5M/137M [00:01<00:03, 30.4Mb/s]
 30%|##9       | 40.5M/137M [00:01<00:03, 29.2Mb/s]
 32%|###1      | 43.5M/137M [00:01<00:03, 28.9Mb/s]
 34%|###3      | 46.4M/137M [00:01<00:03, 28.9Mb/s]
 36%|###6      | 49.3M/137M [00:01<00:03, 29.0Mb/s]
 39%|###8      | 52.7M/137M [00:01<00:02, 30.2Mb/s]
 41%|####1     | 56.2M/137M [00:01<00:02, 31.6Mb/s]
 43%|####3     | 59.4M/137M [00:02<00:02, 30.3Mb/s]
 46%|####5     | 62.4M/137M [00:02<00:02, 30.0Mb/s]
 48%|####7     | 65.5M/137M [00:02<00:02, 30.0Mb/s]
 51%|#####     | 69.3M/137M [00:02<00:02, 32.2Mb/s]
 53%|#####3    | 72.9M/137M [00:02<00:01, 32.4Mb/s]
 56%|#####5    | 76.2M/137M [00:02<00:01, 32.6Mb/s]
 58%|#####8    | 79.4M/137M [00:02<00:01, 32.3Mb/s]
 61%|######    | 83.4M/137M [00:02<00:01, 34.1Mb/s]
 63%|######3   | 86.8M/137M [00:02<00:01, 34.1Mb/s]
 66%|######5   | 90.2M/137M [00:02<00:01, 34.0Mb/s]
 69%|######8   | 94.1M/137M [00:03<00:01, 35.2Mb/s]
 71%|#######1  | 97.6M/137M [00:03<00:01, 35.1Mb/s]
 74%|#######4  | 101M/137M [00:03<00:01, 35.4Mb/s]
 77%|#######6  | 105M/137M [00:03<00:00, 35.2Mb/s]
 79%|#######9  | 108M/137M [00:03<00:00, 34.4Mb/s]
 82%|########1 | 112M/137M [00:03<00:00, 34.5Mb/s]
 84%|########4 | 115M/137M [00:03<00:00, 34.0Mb/s]
 87%|########6 | 119M/137M [00:03<00:00, 33.0Mb/s]
 89%|########9 | 122M/137M [00:03<00:00, 33.5Mb/s]
 92%|#########2| 126M/137M [00:04<00:00, 34.1Mb/s]
 95%|#########4| 130M/137M [00:04<00:00, 34.4Mb/s]
 97%|#########7| 133M/137M [00:04<00:00, 34.4Mb/s]
100%|#########9| 136M/137M [00:04<00:00, 34.3Mb/s]
100%|##########| 137M/137M [00:04<00:00, 31.5Mb/s]
```

```
[INFO] Beginning extraction
[INFO] Chromium extracted to: C:\Users\hamza\AppData\Local\pyppeteer\pyppeteer\local-chr
omium\588429
```

In [5]:

```
%%html
<style>
table {float:left; width:100%;}
th {float:center; color:orange;}
tr {float:center; color:red;}
</style>
```

In [6]:

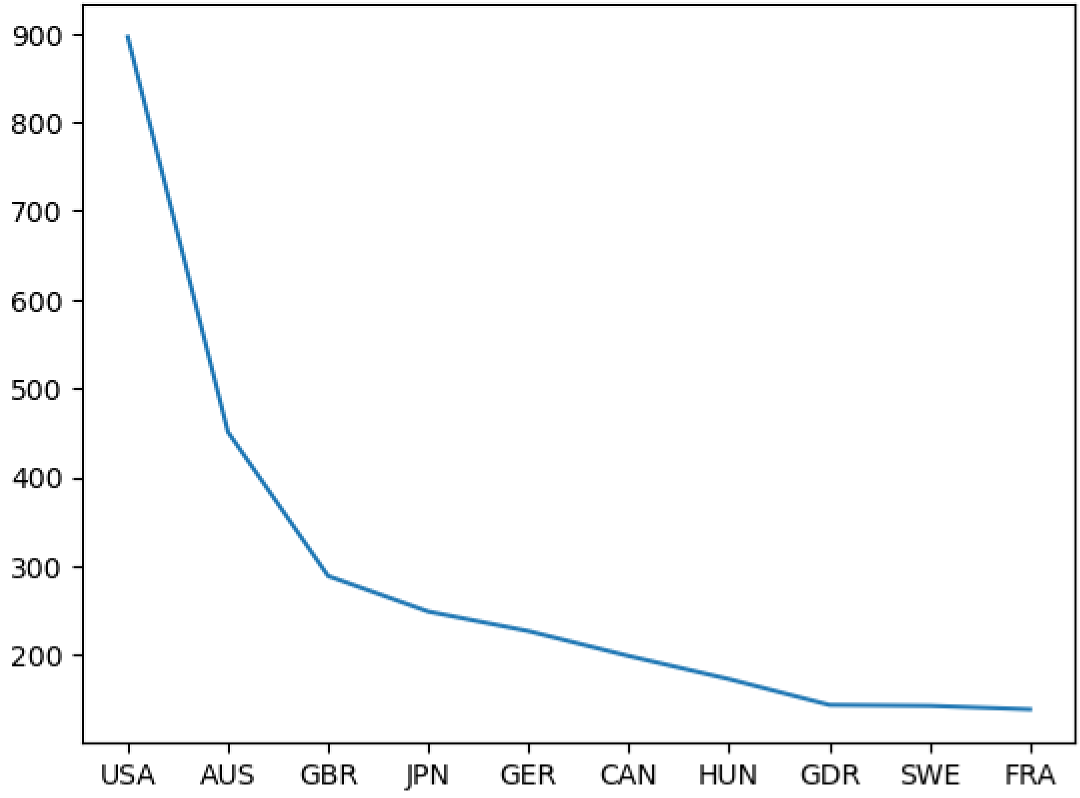
```
swimmers = pd.read_csv('Olympic_Swimming_Results_1912to2020.csv')
```

Top 10 Teams with most Apperances in Rankings

In [7]:

```
display(swimmers['Team'].value_counts().nlargest(10))
plt.plot(swimmers['Team'].value_counts().nlargest(10))
plt.show()
```

```
USA    896
AUS    451
GBR    289
JPN    249
GER    227
CAN    199
HUN    173
GDR    144
SWE    143
FRA    139
Name: Team, dtype: int64
```

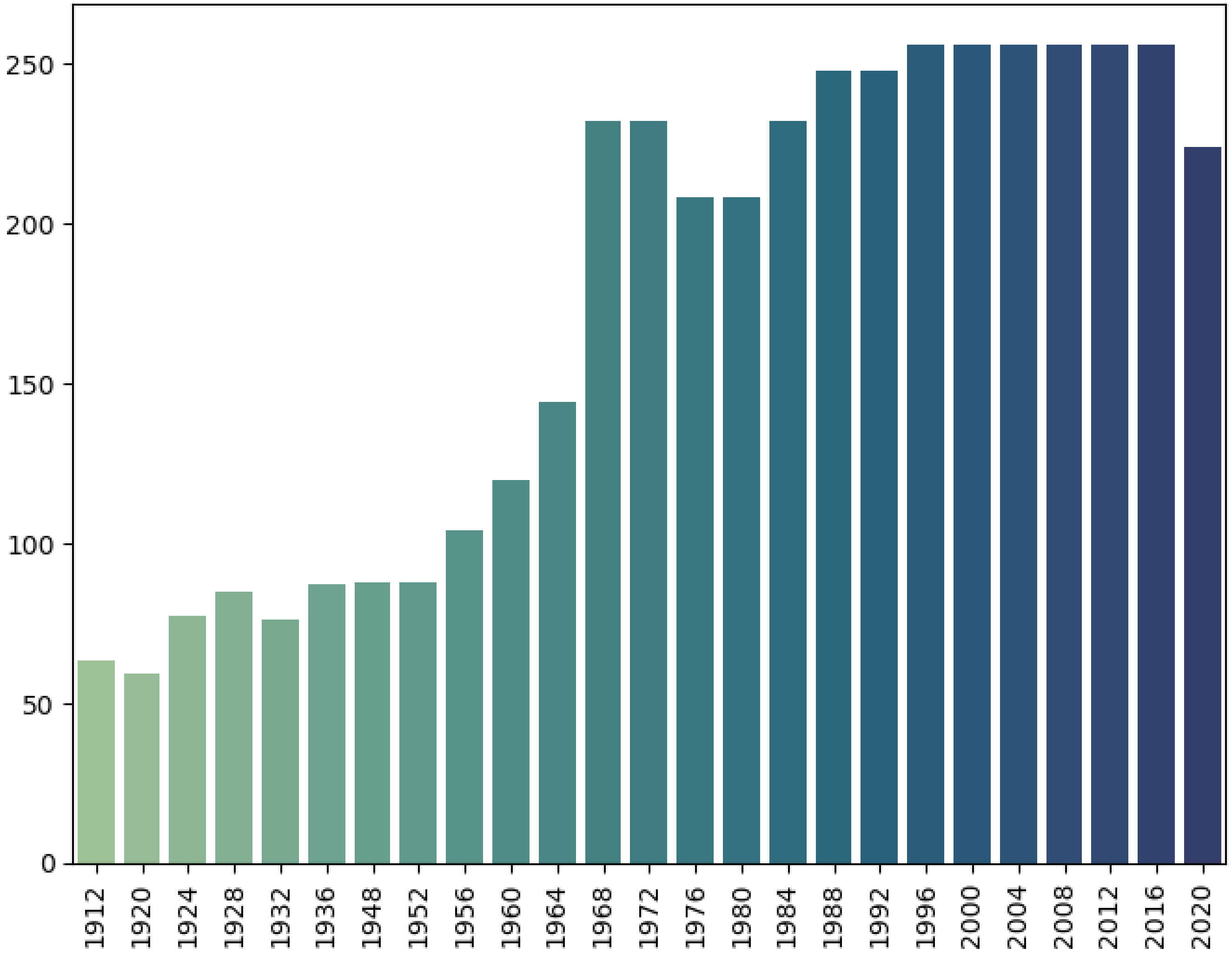


Canada has placed 6 for number of apperances which proves the nation can complete nicely at the world stage

How much Competition are They Facing

In [8]:

```
plt.figure(figsize=(8,6))
sns.barplot(x=swimmers['Year'].value_counts().index, y=swimmers['Year'].value_counts().v
plt.xticks(rotation=90)
plt.show()
```



Number of participants have greatly increased over the years before stabilizing in 2000

Any Sudden Jumps in Data?

In [9]:

```
swimmers.describe()
```

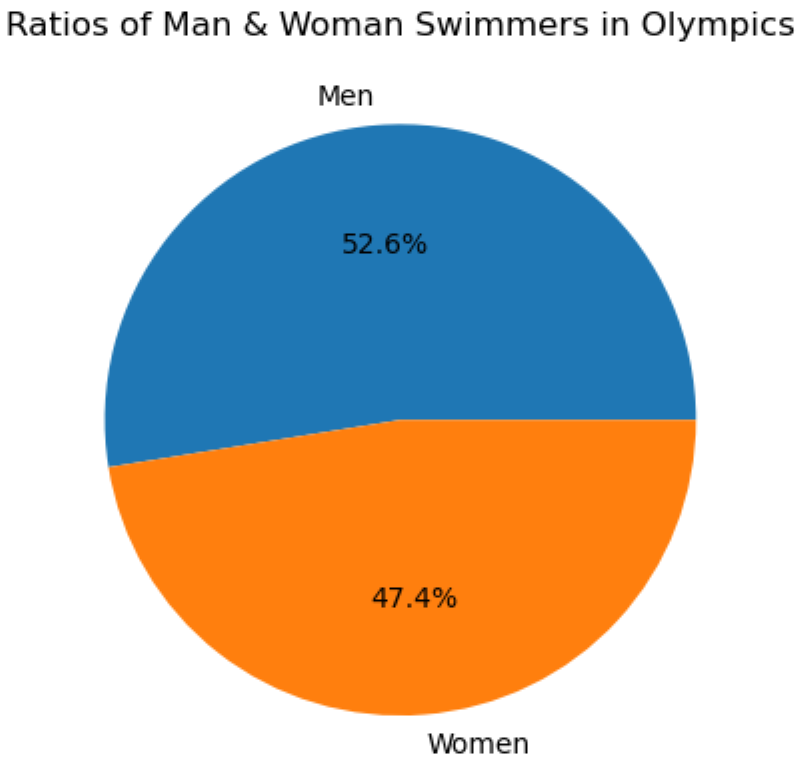
Out[9]:

	Year	Relay?	Rank
count	4359.000000	4359.000000	4359.000000
mean	1982.936453	0.169764	3.164946
std	26.928344	0.375468	1.189715
min	1912.000000	0.000000	0.000000
25%	1968.000000	0.000000	2.000000
50%	1988.000000	0.000000	4.000000
75%	2004.000000	0.000000	4.000000

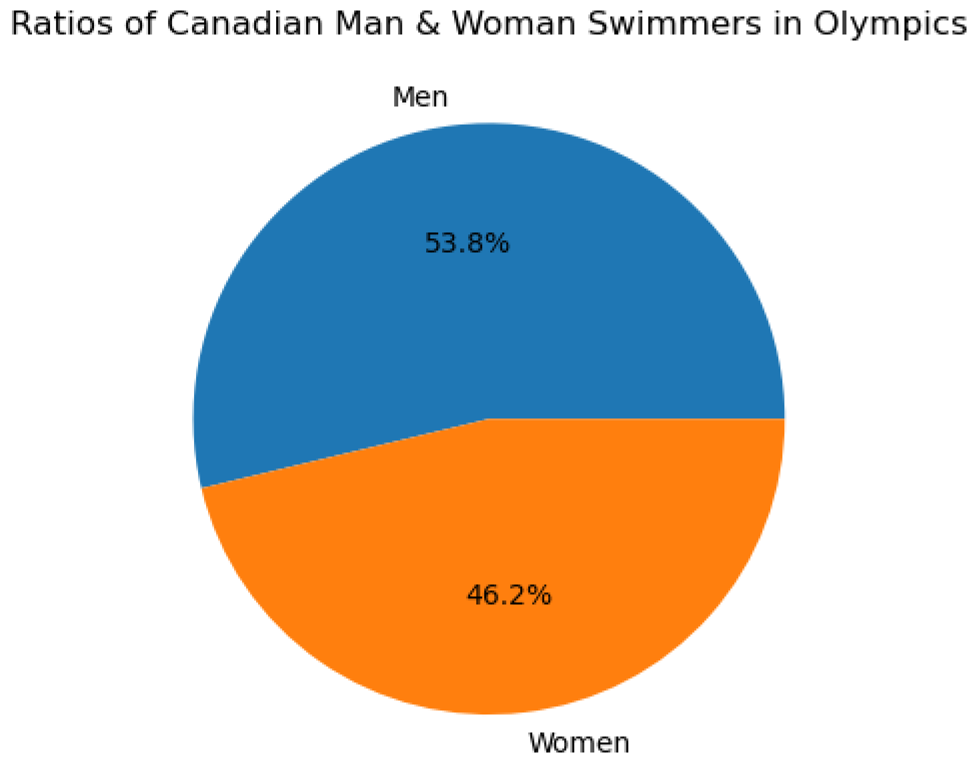
max	2020.000000	1.000000	5.000000

Comparing diversity in Canadian Swimmers to the General Competition

In [10]:
```
plt.pie(swimmers['Gender'].value_counts().values, labels = swimmers['Gender'].value_coun
plt.title('Ratios of Man & Woman Swimmers in Olympics')
plt.show()
```



In [11]:
```
topcan = swimmers[swimmers.Team == 'CAN']
plt.pie(topcan['Gender'].value_counts().values, labels = swimmers['Gender'].value_counts
plt.title('Ratios of Canadian Man & Woman Swimmers in Olympics')
plt.show()
```

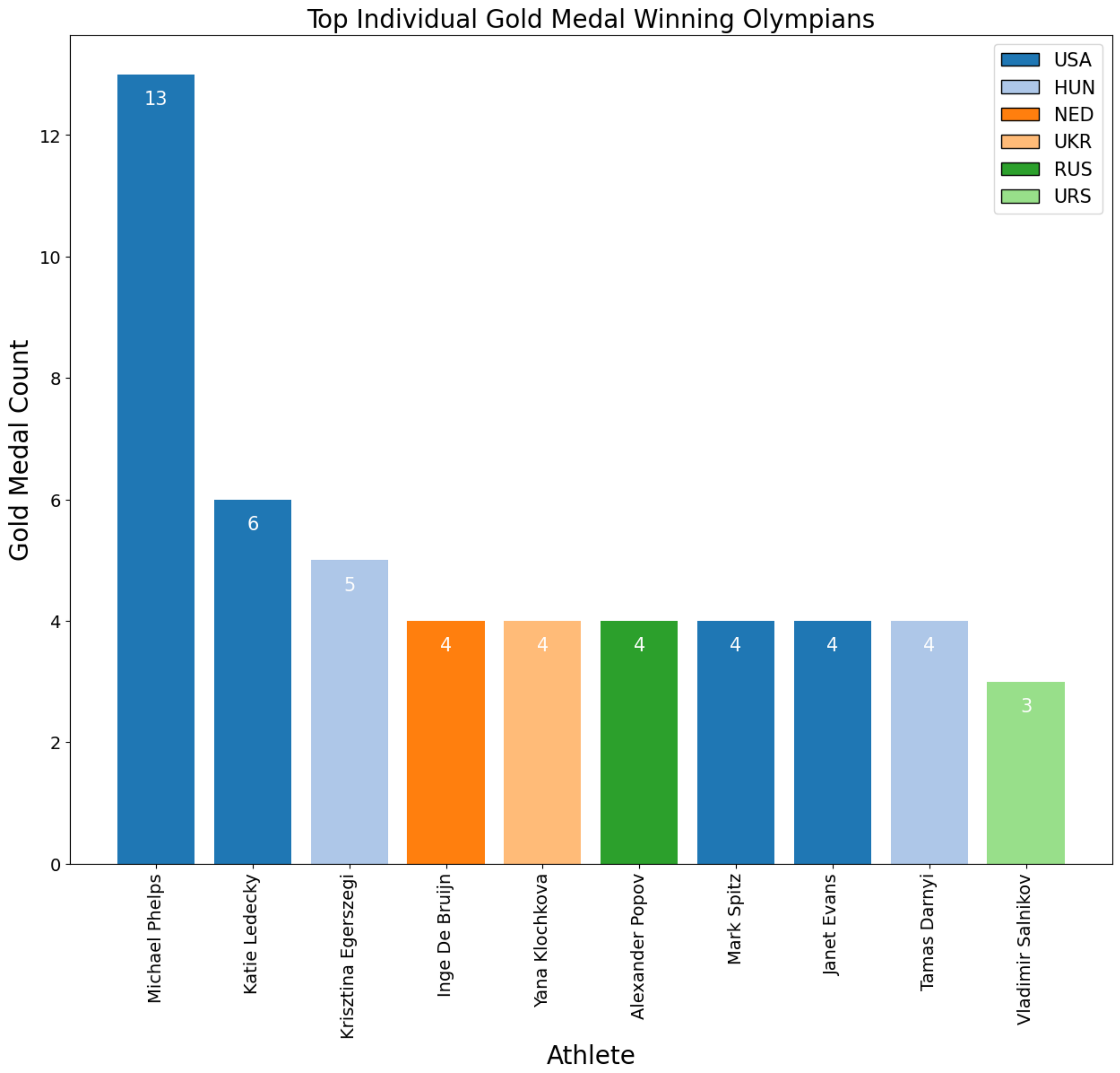


Canada is essentialy around the average of the breakdown, historically canadian woman have done better in summer sports then men so perhaps Canda should invest more in the womans department

Comparing Top Athlethes with Top Canadian Swimmers

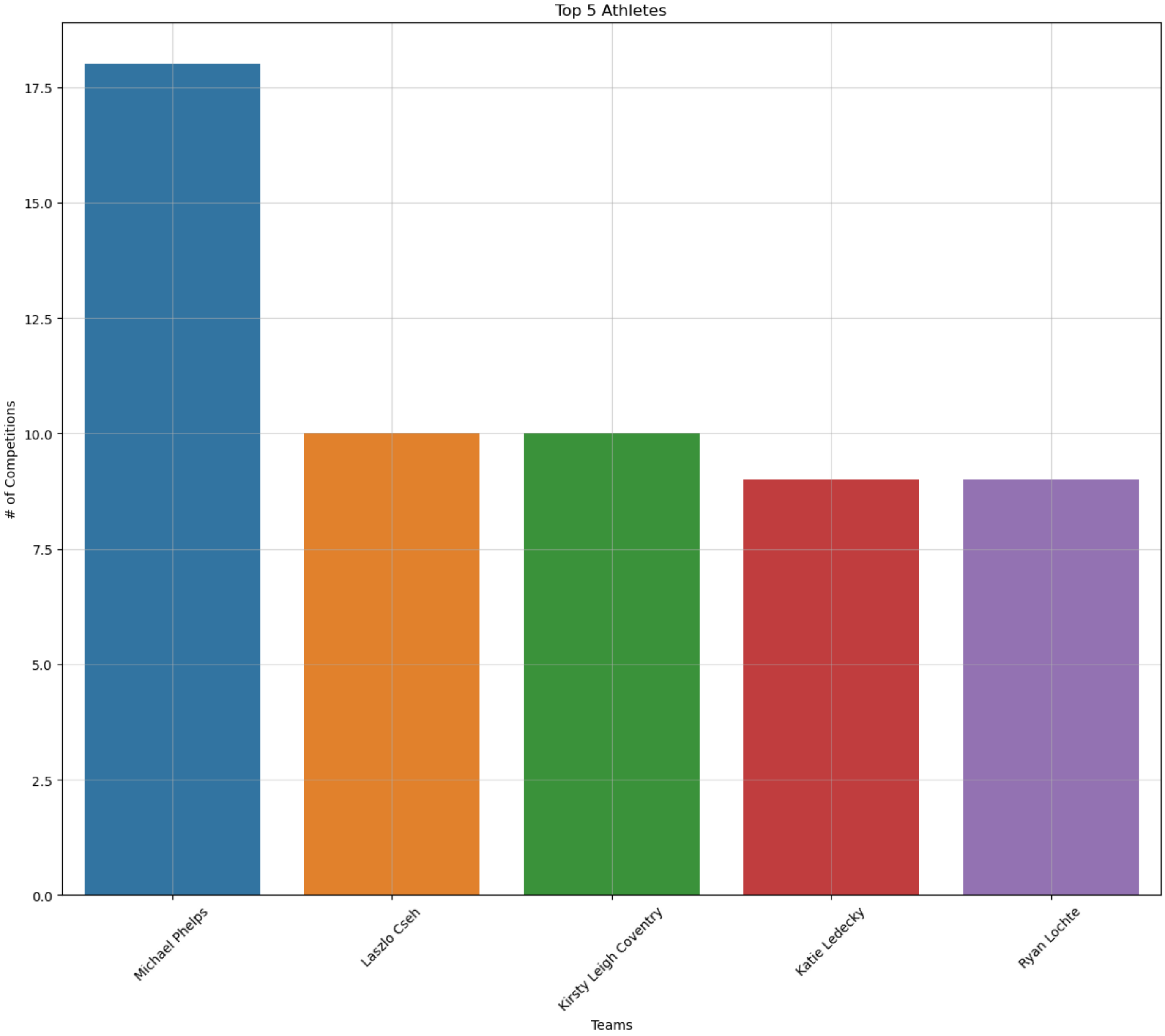
In [12]:

```
gold_medals_pd = swimmers[swimmers['Rank'] == 1]
gold_medal_table = pd.pivot_table(gold_medals_pd, values=['Rank', 'Team'], index='Athlet
gold_medal_df = gold_medal_table.reset_index()
gold_medal_df = gold_medal_df.sort_values('Rank', ascending=False)
gold_medal_df = gold_medal_df[:10].reset_index(drop=True)

#Create Bar Graph

%matplotlib inline
plt.rcParams['figure.figsize'] = [15, 12]

gold_medal_df = gold_medal_df.set_index('Athlete')
color_dict = {val: plt.cm.tab20(i) for i, val in enumerate(gold_medal_df['Team'].unique(

fig, ax = plt.subplots()
ax.bar(gold_medal_df.index, gold_medal_df['Rank'], color=gold_medal_df['Team'].map(color
legend_elements = [Patch(facecolor=color_dict[val], edgecolor='black', label=val) for va
ax.legend(handles=legend_elements, fontsize=15)

# Add text annotations to the bars
for i in range(len(gold_medal_df.index)):
    ax.text(gold_medal_df.index[i], gold_medal_df['Rank'][i] - .5, str(gold_medal_df['Ra

ax.set_xlabel('Athlete', fontsize=20)
ax.set_ylabel('Gold Medal Count', fontsize=20)
plt.xticks(rotation=90, fontsize=14)
plt.yticks(fontsize=14)

plt.title("Top Individual Gold Medal Winning Olympians", fontsize=20)
plt.show()
```



The top 10 gold medalists are from 6 countries

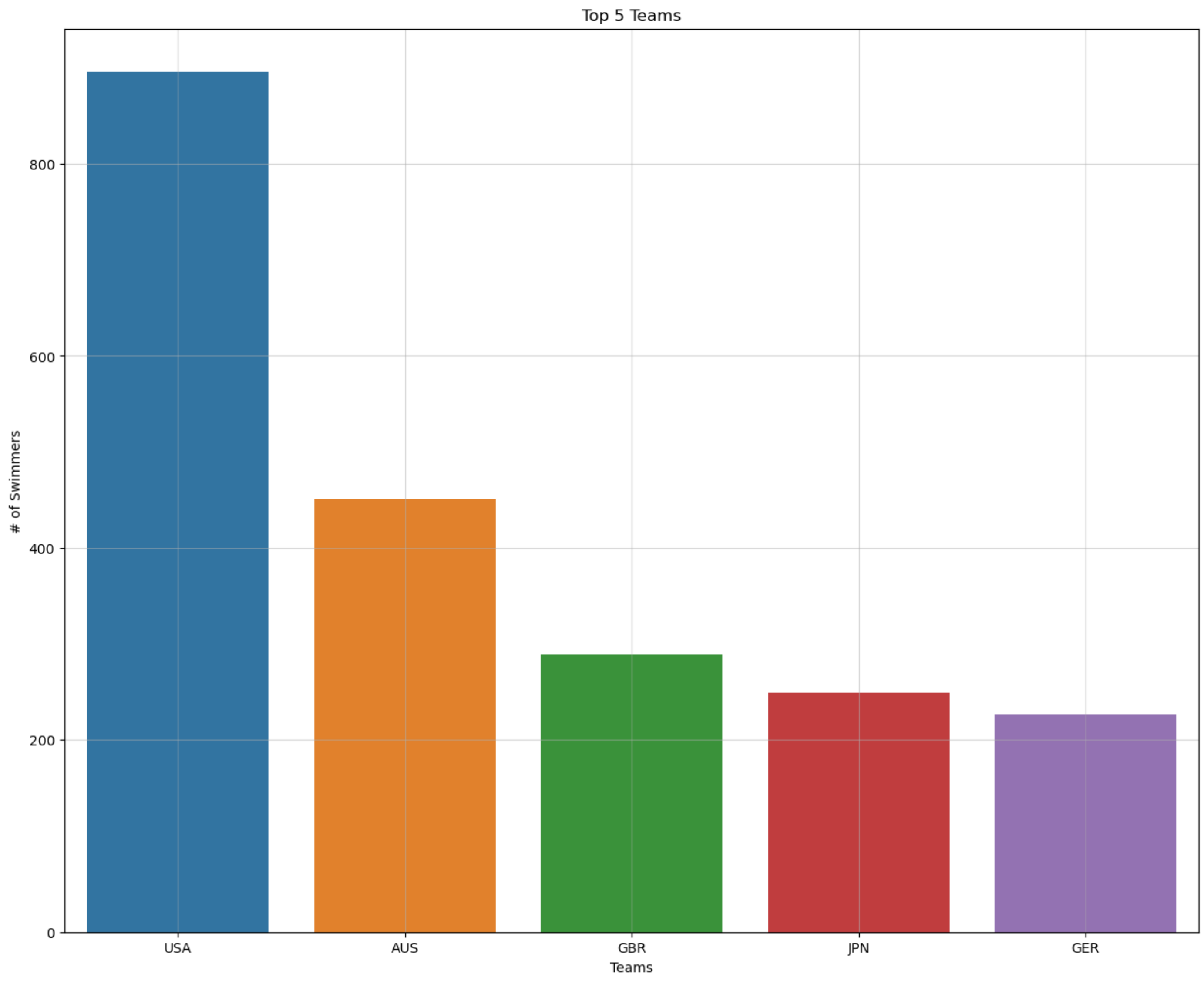
In [13]:
```
top5_athlete = swimmers['Athlete'].value_counts()[:5]
sns.barplot(x=top5_athlete.index, y=top5_athlete.values)
plt.xticks(rotation=45)
plt.title('Top 5 Athletes')
plt.xlabel('Teams')
plt.ylabel('# of Competitions')
plt.grid(alpha=0.5)
plt.show()
```



With american swimmers being the sole exception is it clear that participating in more competions doesn't translate to more medals

In [14]:

```
top5_team = swimmers['Team'].value_counts()[:5]
sns.barplot(x=top5_team.index, y=top5_team.values)
plt.title('Top 5 Teams')
plt.xlabel('Teams')
plt.ylabel('# of Swimmers')
plt.grid(alpha=0.5)
plt.show()
```

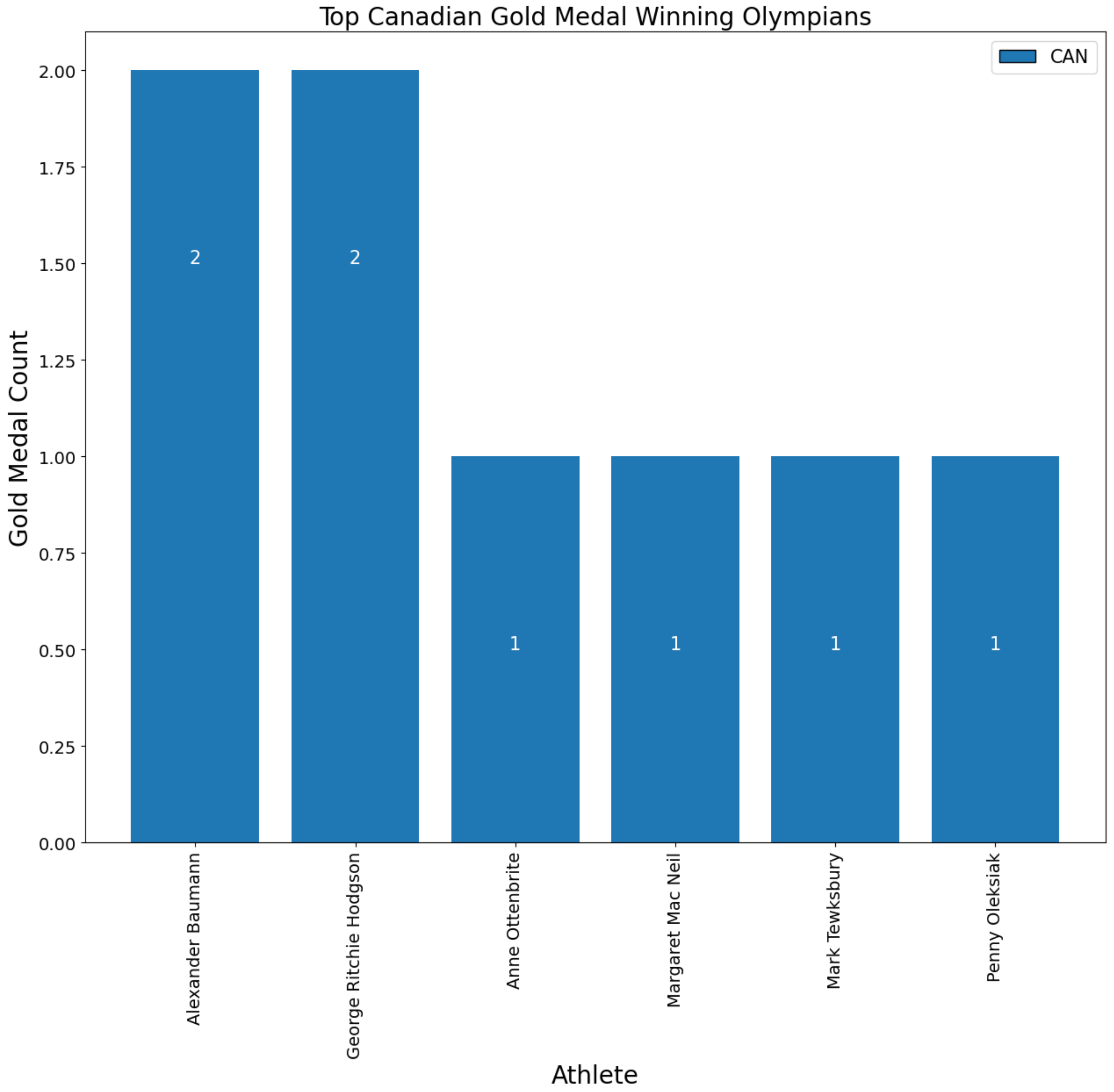


Its clear that the US aside none of the other top medalist are from the countries with most participants, so quality beats quantity here

In [15]:

```
gold_medals_pd = topcan[swimmers['Rank'] == 1]
gold_medal_table = pd.pivot_table(gold_medals_pd, values=['Rank', 'Team'], index='Athlet
gold_medal_df = gold_medal_table.reset_index()
gold_medal_df = gold_medal_df.sort_values('Rank', ascending=False)
gold_medal_df = gold_medal_df[:10].reset_index(drop=True)

#Create Bar Graph

%matplotlib inline
plt.rcParams['figure.figsize'] = [15, 12]

gold_medal_df = gold_medal_df.set_index('Athlete')
color_dict = {val: plt.cm.tab20(i) for i, val in enumerate(gold_medal_df['Team'].unique(

fig, ax = plt.subplots()
ax.bar(gold_medal_df.index, gold_medal_df['Rank'], color=gold_medal_df['Team'].map(color
legend_elements = [Patch(facecolor=color_dict[val], edgecolor='black', label=val) for va
ax.legend(handles=legend_elements, fontsize=15)

# Add text annotations to the bars
for i in range(len(gold_medal_df.index)):
    ax.text(gold_medal_df.index[i], gold_medal_df['Rank'][i] - .5, str(gold_medal_df['Ra

ax.set_xlabel('Athlete', fontsize=20)
ax.set_ylabel('Gold Medal Count', fontsize=20)
```

```
plt.xticks(rotation=90, fontsize=14)
plt.yticks(fontsize=14)

plt.title("Top Canadian Gold Medal Winning Olympians", fontsize=20)

plt.show()
```



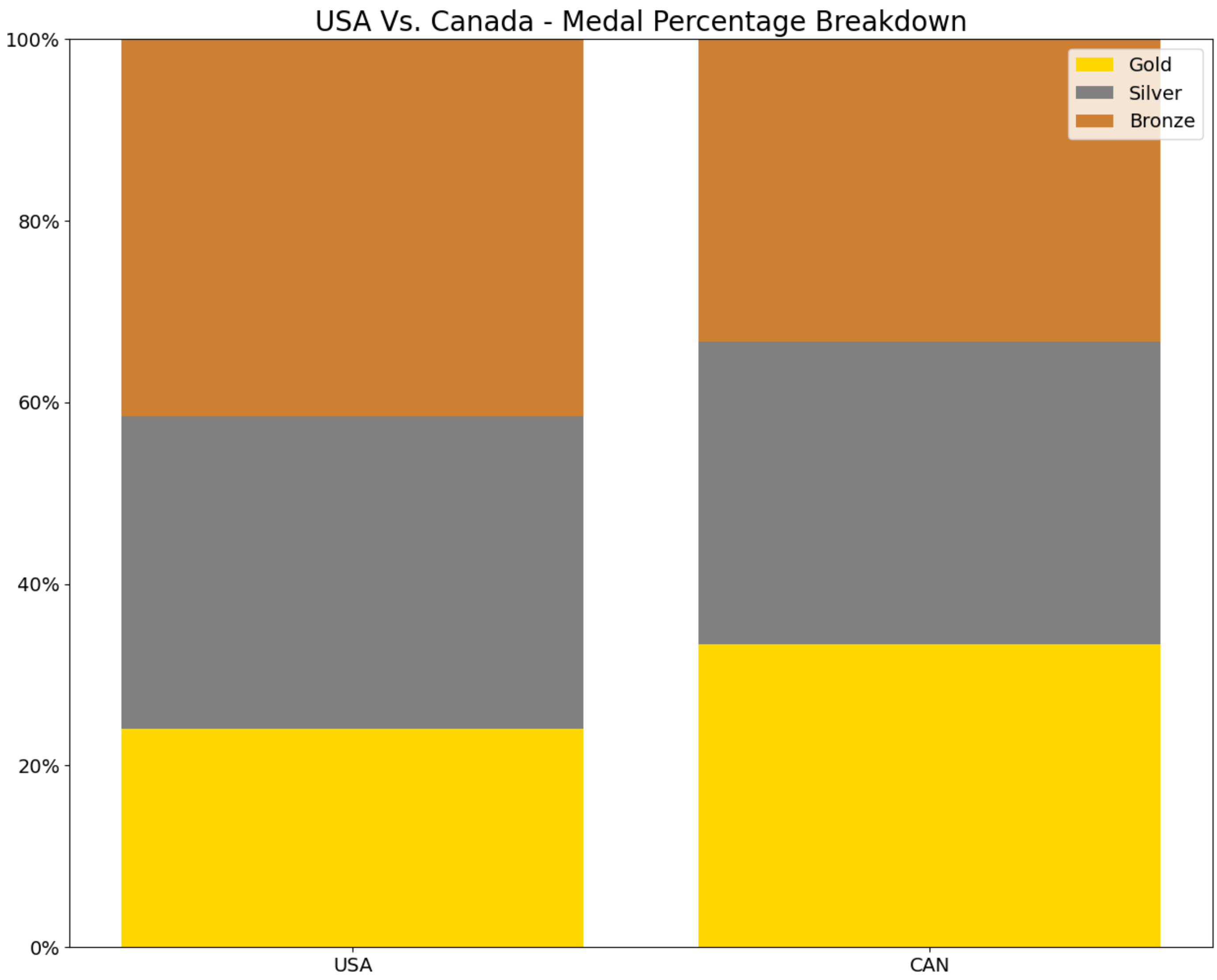
The top canadian gold medalist has 2 medals which would put him right outside the rankings

USA vs Canada Medals Breakdown

In [16]:

```
usa_df = swimmers[swimmers['Team'] == 'USA']
can_df = swimmers[swimmers['Team'] == 'CAN']

usa_gold_df = usa_df[usa_df['Rank'] == 1]
usa_gold_total = usa_gold_df['Rank'].sum()
usa_silver_df = usa_df[usa_df['Rank'] == 2]
usa_silver_total = usa_silver_df['Rank'].sum()
usa_bronze_df = usa_df[usa_df['Rank'] == 3]
```

```
usa_bronze_total = usa_bronze_df['Rank'].sum()

can_gold_df = can_df[can_df['Rank'] == 1]
can_gold_total = can_gold_df['Rank'].sum()
can_silver_df = can_df[can_df['Rank'] == 2]
can_silver_total = can_gold_df['Rank'].sum()
can_bronze_df = can_df[can_df['Rank'] == 3]
can_bronze_total = can_gold_df['Rank'].sum()

# Noramlize Data
labels = ['USA', 'CAN']
golds = [usa_gold_total, can_gold_total]
silvers = [usa_silver_total, can_silver_total]
bronzes = [usa_bronze_total, can_bronze_total]

total = np.array(golds) + np.array(silvers) + np.array(bronzes)
gold_means = 100 * np.array(golds) / total
silver_means = 100 * np.array(silvers) / total
bronze_means = 100 * np.array(bronzes) / total

# Create Stacked Bar Chart
fig, ax = plt.subplots()
ax.bar(labels, gold_means, label='Gold', color='#FFD700')
ax.bar(labels, silver_means, bottom=gold_means, label='Silver', color='#808080')
ax.bar(labels, bronze_means , bottom=gold_means+silver_means, label='Bronze', color='#CD
ax.yaxis.set_major_formatter(PercentFormatter())
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)


plt.title('USA Vs. Canada - Medal Percentage Breakdown', fontsize=20)
plt.legend(fontsize=14)
plt.ylim(0, 100)

plt.show()
```



While the US have more bronze medals the Canadians are more balanced

Specialization is Important

In [17]:

```
display(swimmers[swimmers.Athlete == 'Penny Oleksiak'])
```

	Location	Year	Distance (in meters)	Stroke	Relay?	Gender	Team	Athlete	Results	Rank
139	Tokyo	2020	100m	Freestyle	0	Women	CAN	Penny Oleksiak	52.59	4
178	Tokyo	2020	200m	Freestyle	0	Women	CAN	Penny Oleksiak	1:54.70	3
265	Rio	2016	100m	Butterfly	0	Women	CAN	Penny Oleksiak	56.460	3
281	Rio	2016	100m	Freestyle	0	Women	CAN	Penny Oleksiak	52.700	1

Oleksiak has won 3 medals in 4 events proving that.

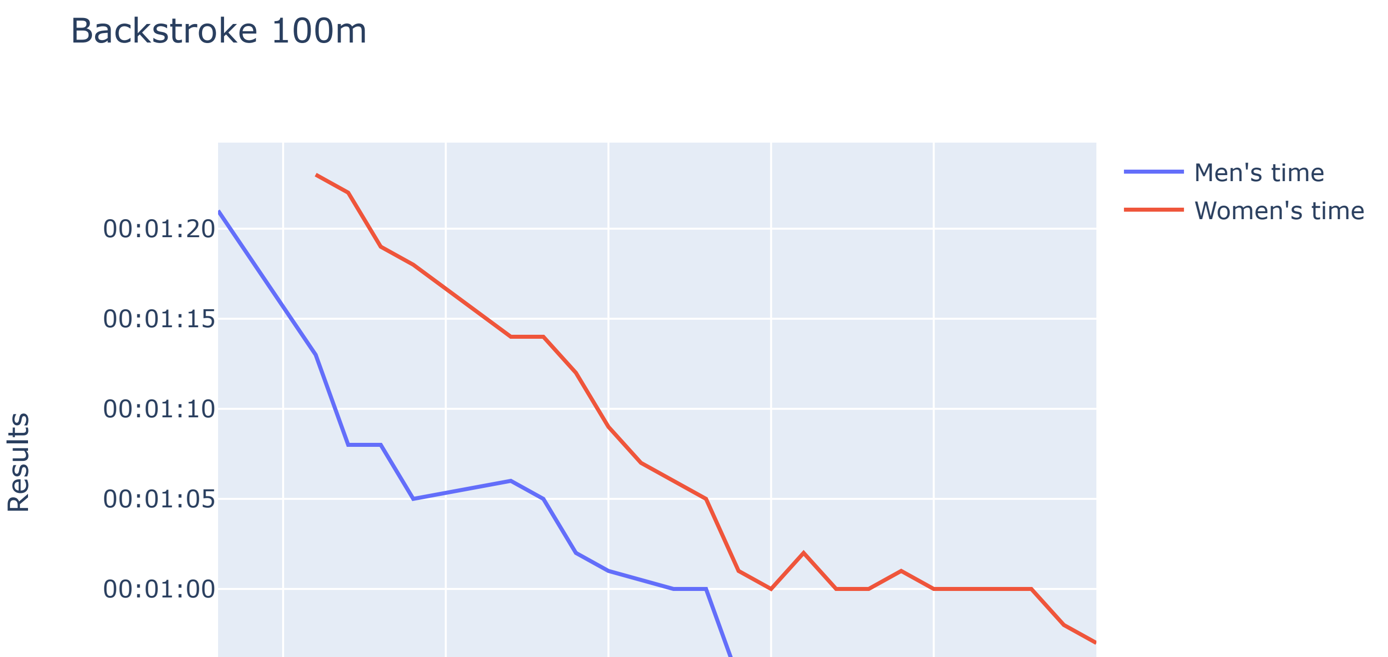
In [18]:
```python
data0=pd.read_csv('Olympic_Swimming_Results_1912to2020.csv')
data0=data0.dropna()
data0=data0[data0['Rank']==1]
```

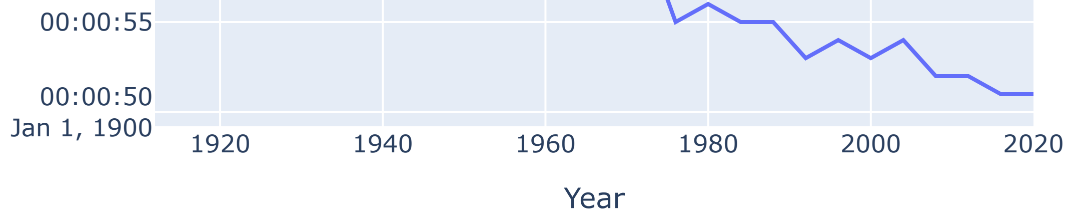
In [19]:
```python
data0=data0.reset_index(drop=True)
for i in range(len(data0)):
    item = data0.iloc[i,8].split('.')[0]
    if item.count(':')==0:
        data0.iloc[i,8]='00:00:'+item
    elif item.count(':')==1:
        data0.iloc[i,8]='00:'+item
    elif item.count(':')==2:
        data0.iloc[i,8]=item

data0['time']=pd.to_datetime(data0['Results'], format='%H:%M:%S')
```

In [20]:
```python
data1=data0[['Distance (in meters)','Stroke']].drop_duplicates()
dist_stroke=[]
for i in range(len(data1)):
    dist_stroke+=[data1.iloc[i,0:2].tolist()]
```

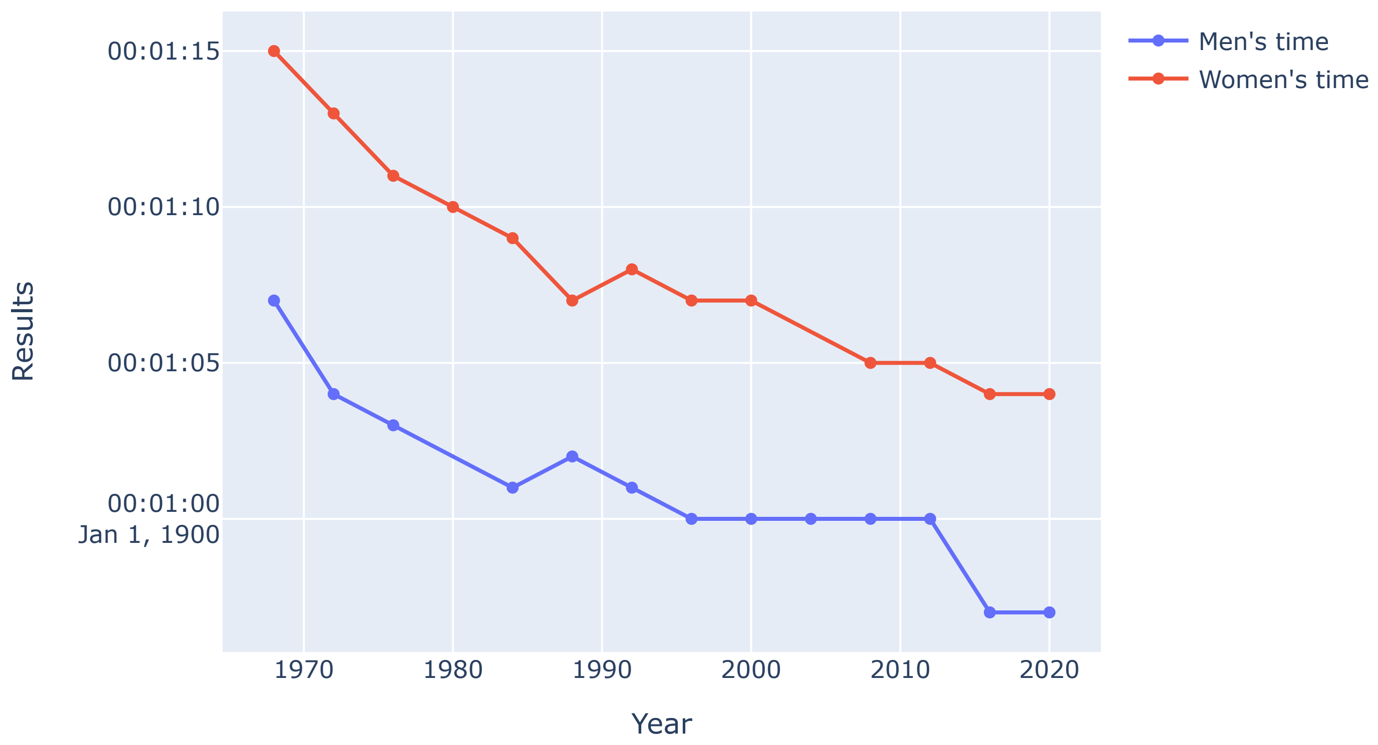
Women and Men Comparisons in Individual Competitions

In [21]:
```python
for item in dist_stroke:
    disti=item[0]
    strokei=item[1]
    datai=data0[data0['Distance (in meters)']==disti][data0['Stroke']==strokei].sort_val
    dataim=datai[datai['Gender']=='Men']
    dataif=datai[datai['Gender']=='Women']
    if len(dataim)>5 or len(dataif)>5:
        fig=make_subplots(specs=[[{"secondary_y":False}]])
        fig.add_trace(go.Scatter(x=dataim['Year'],y=dataim['time'],name="Men's time"),se
        fig.add_trace(go.Scatter(x=dataif['Year'],y=dataif['time'],name="Women's time"),
        fig.update_layout(autosize=False,width=700,height=500,title_text=strokei+' '+dis
        fig.update_xaxes(title_text="Year")
        fig.update_yaxes(title_text="Results",secondary_y=False)
        fig.show()
```

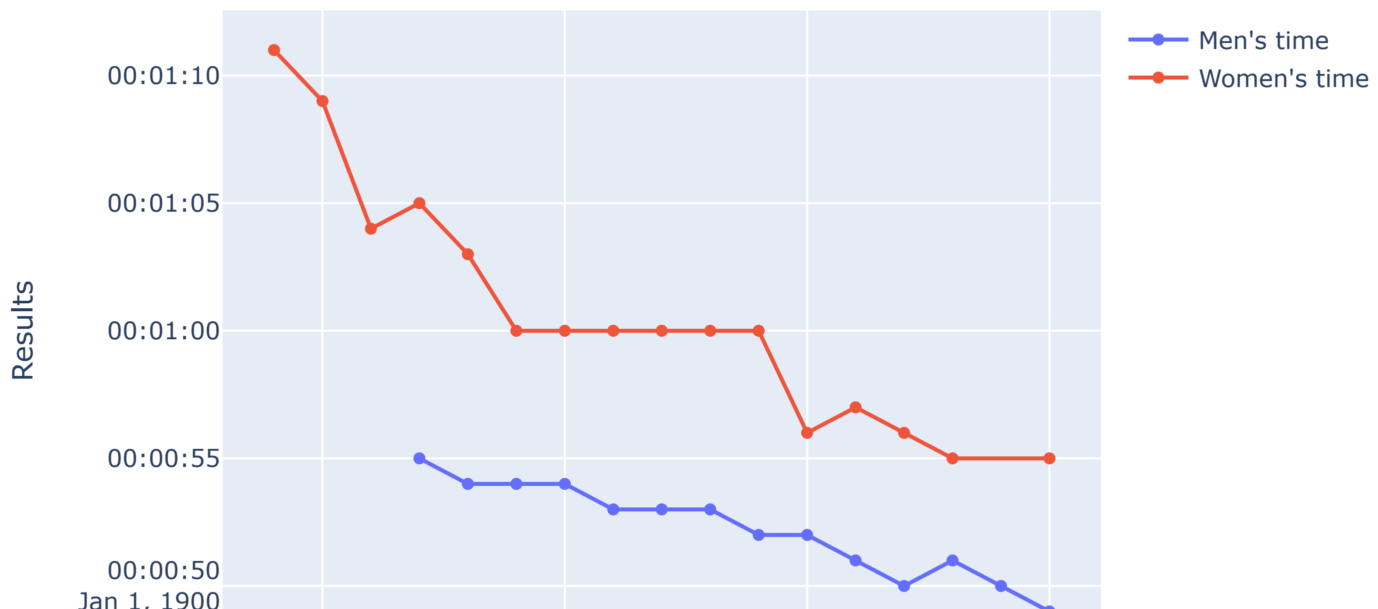


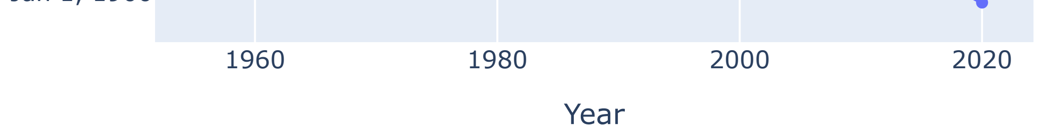


Breaststroke 100m

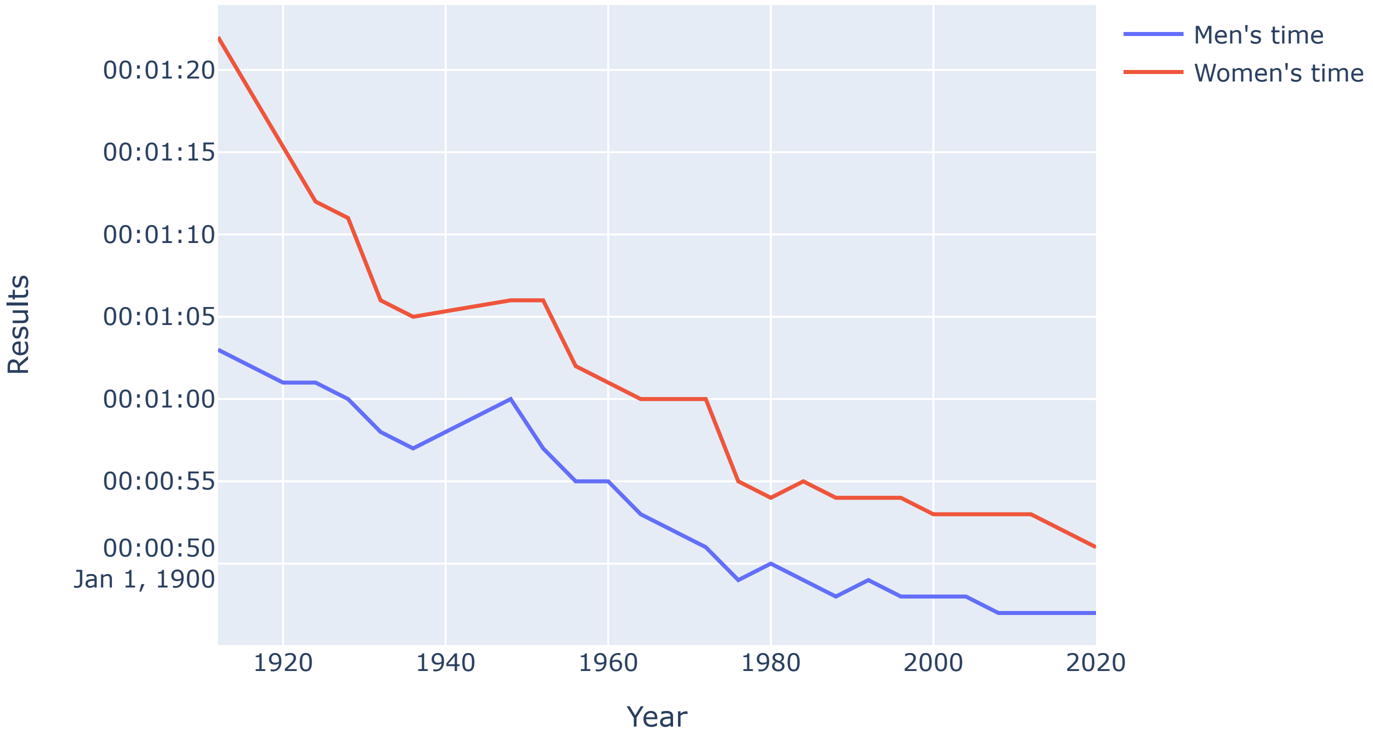


Butterfly 100m

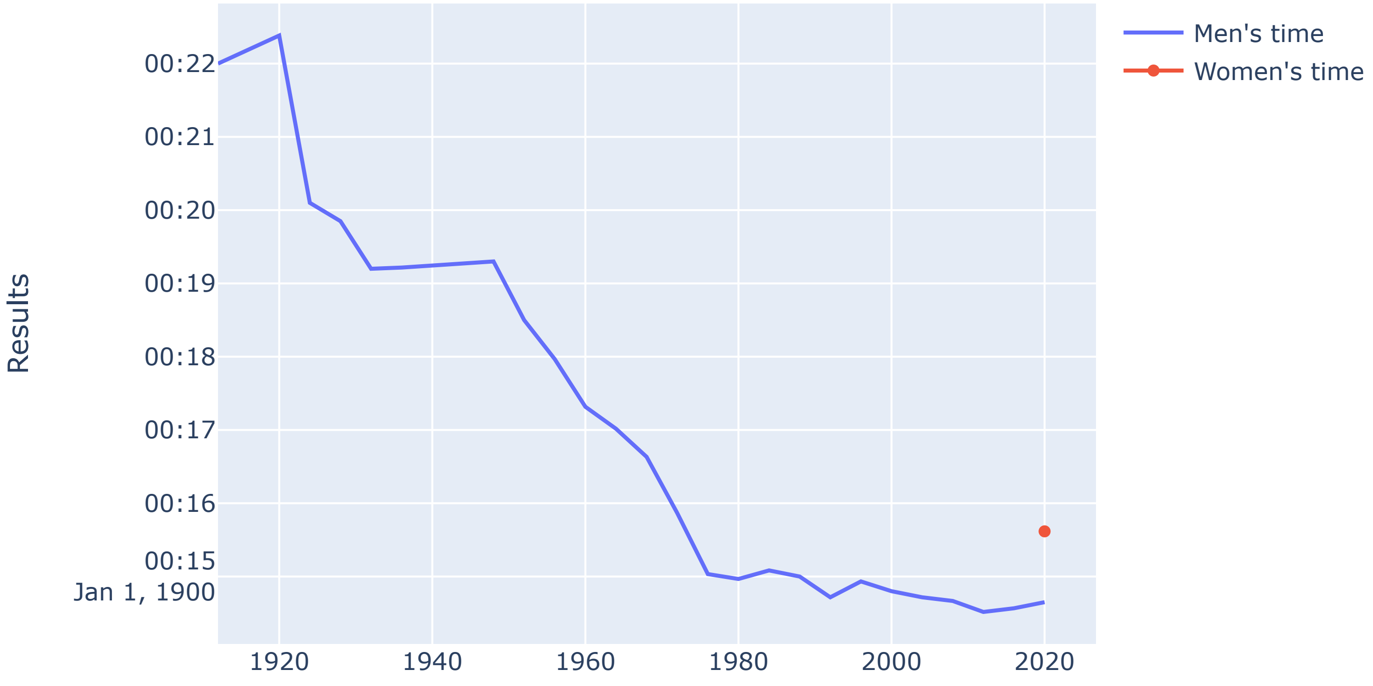




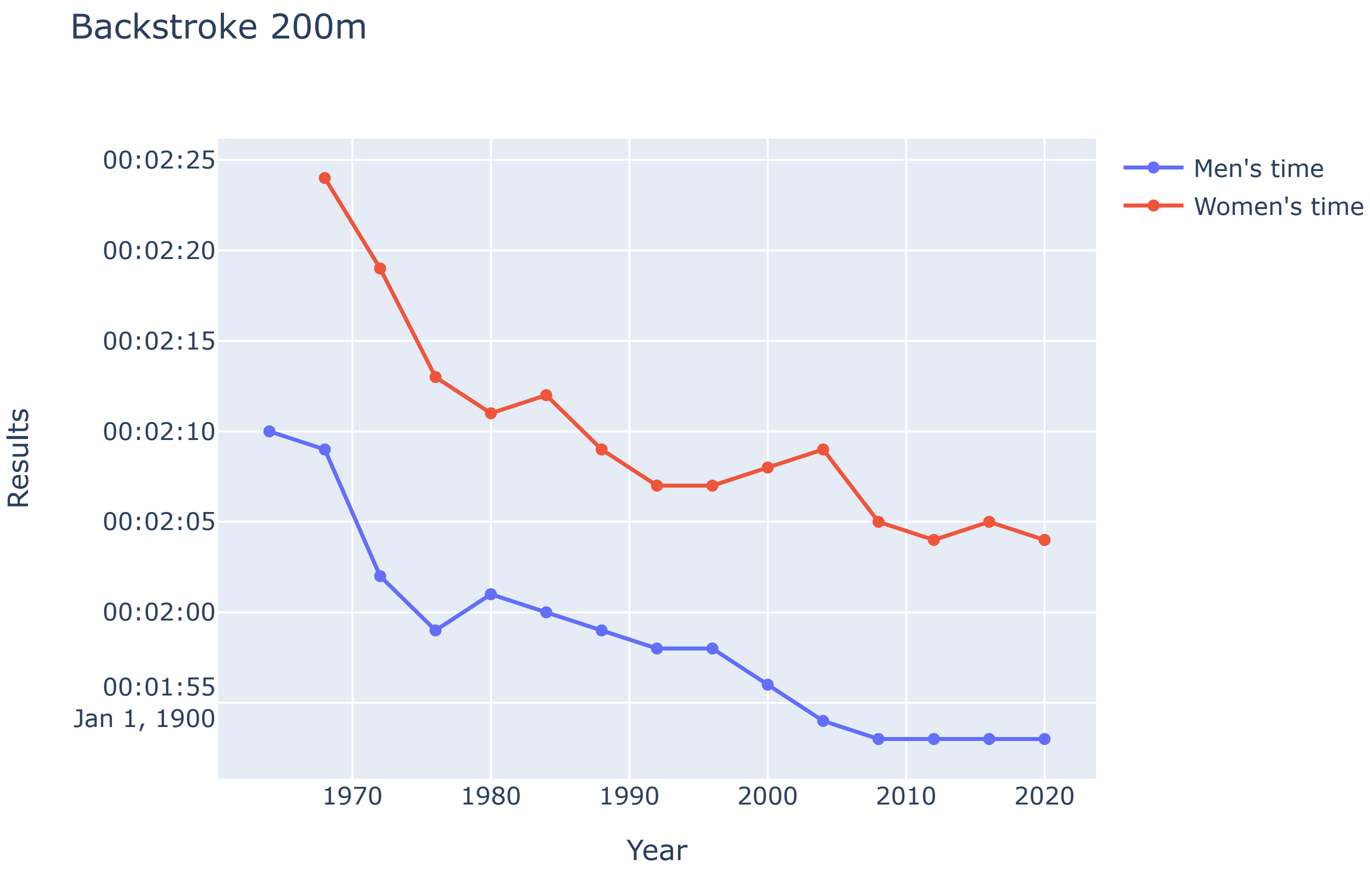
Freestyle 100m

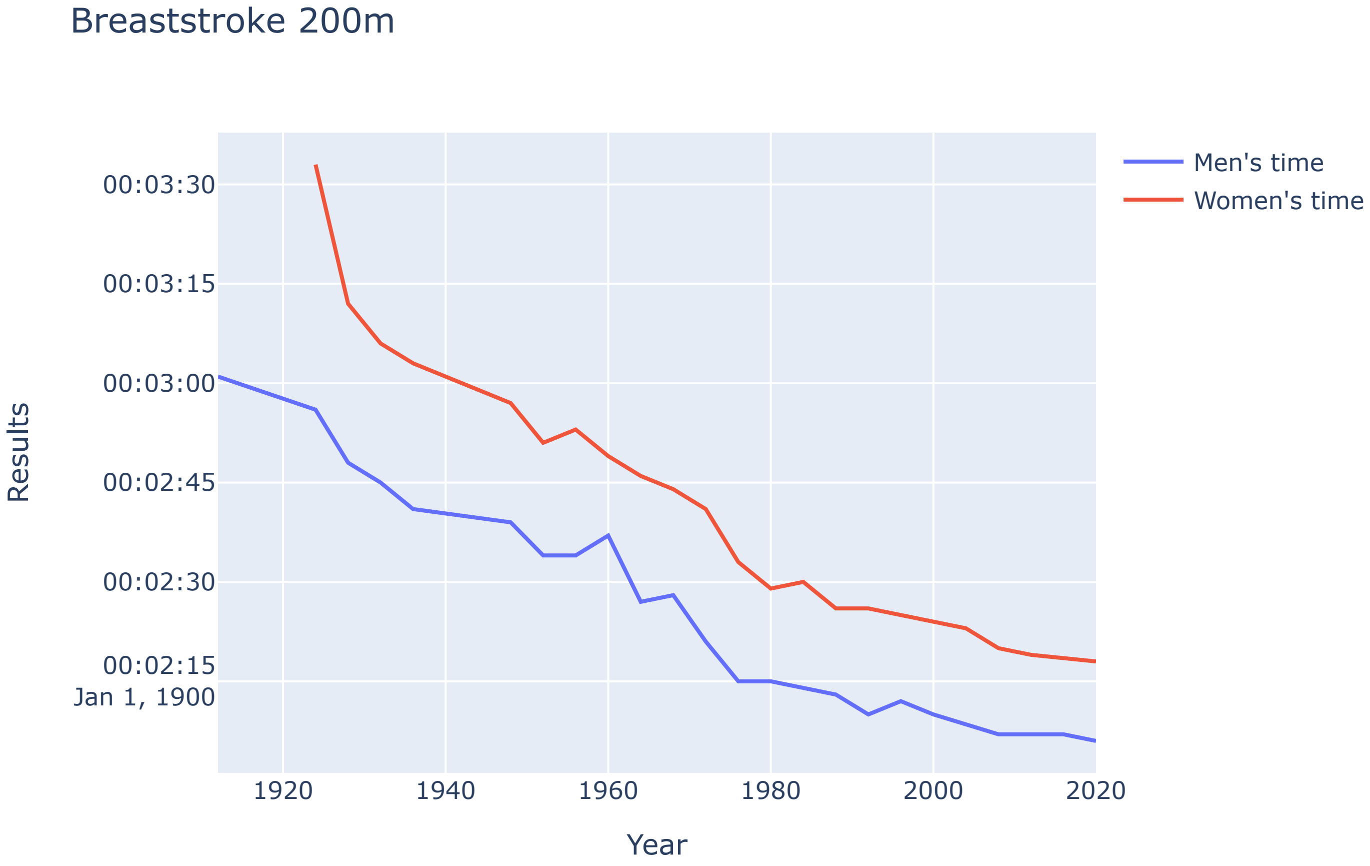


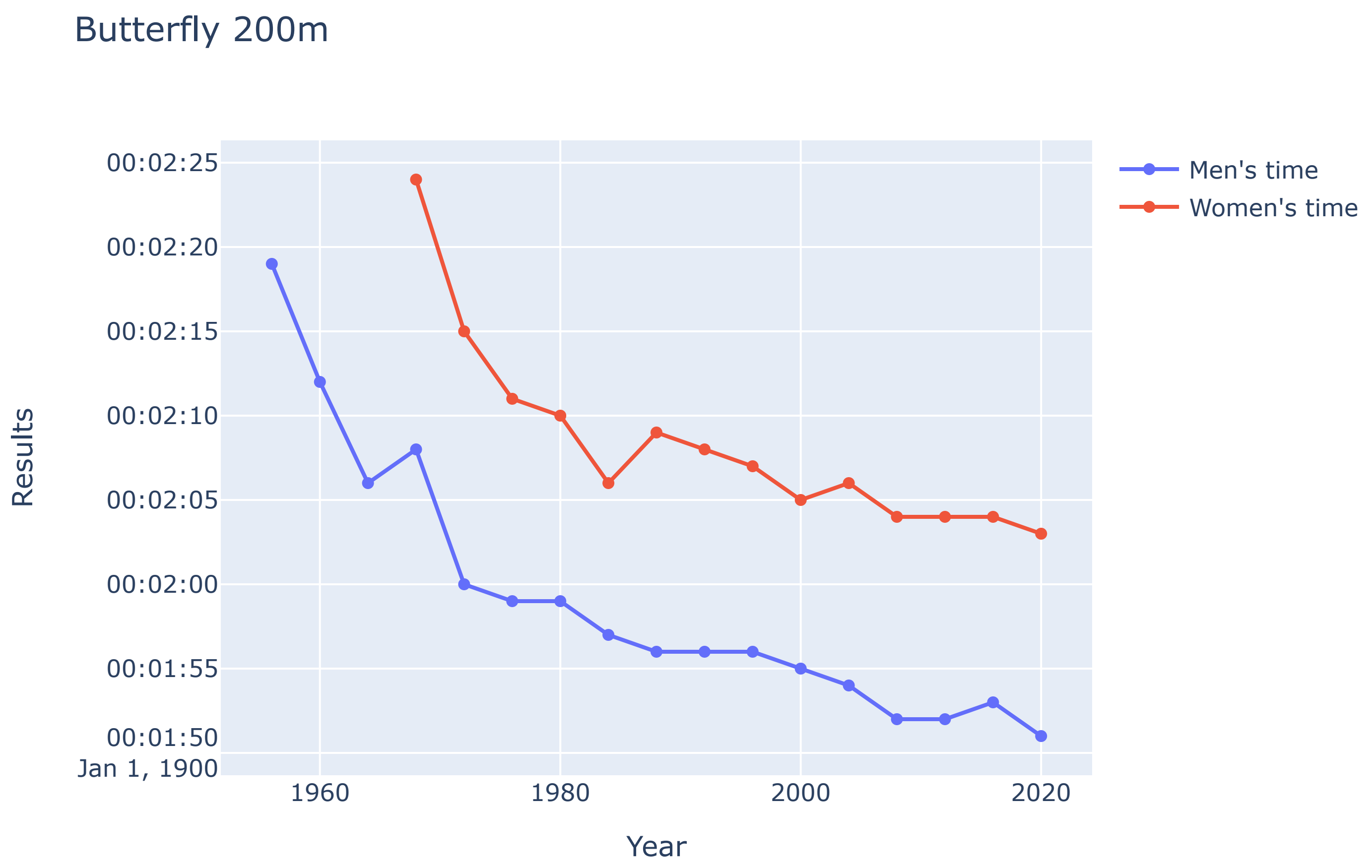
Freestyle 1500m

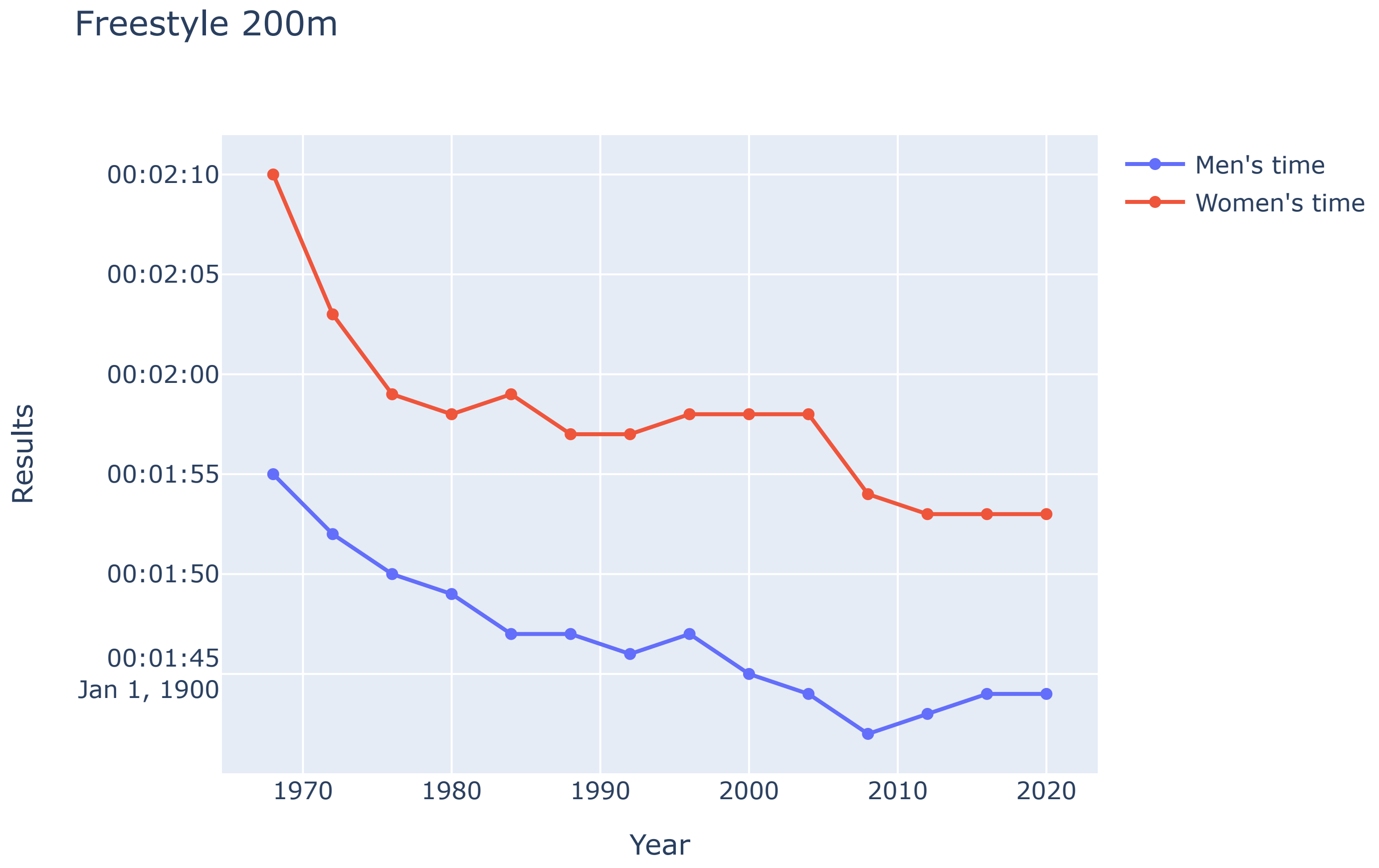


Year

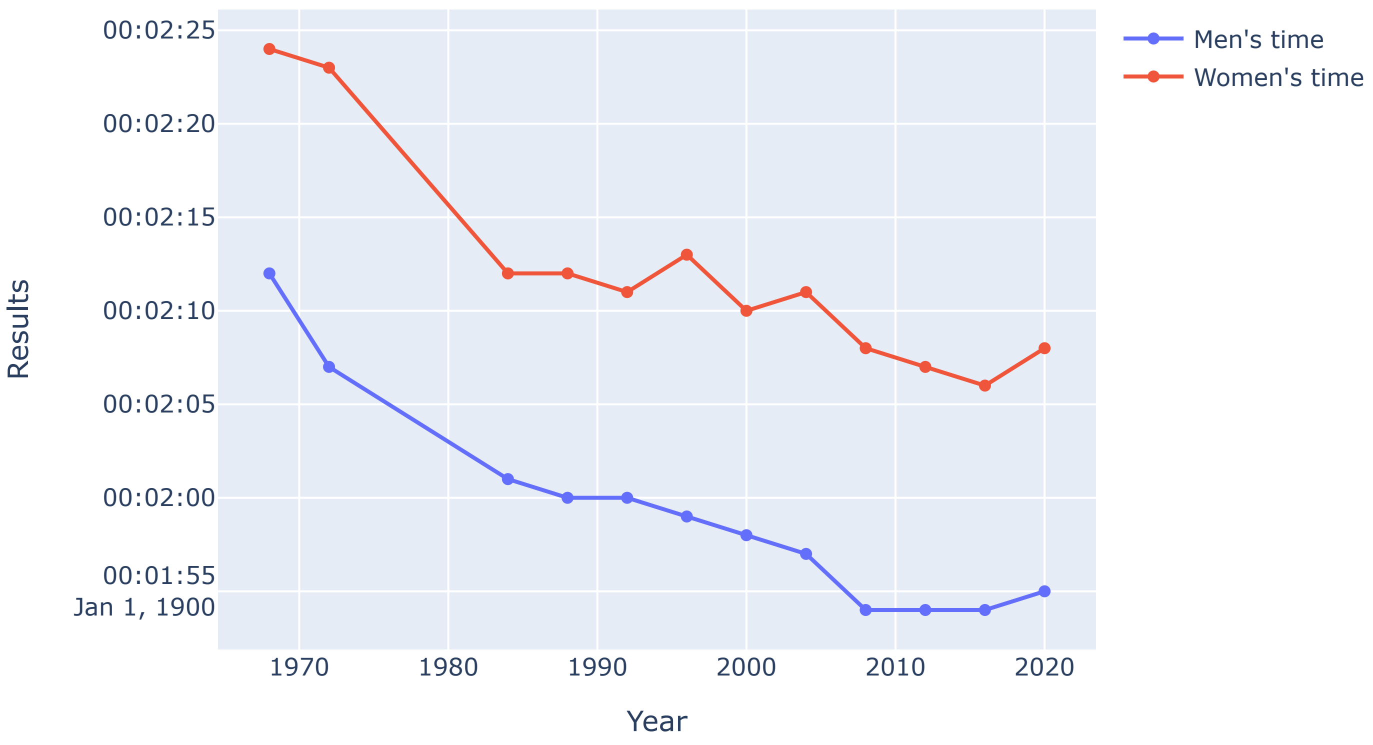

Backstroke 200m
Men's time
Women's time
Results
00:02:25
00:02:20
00:02:15
00:02:10
00:02:05
00:02:00
00:01:55
Jan 1, 1900
1970
1980
1990
2000
2010
2020
Year


Breaststroke 200m
Men's time
Women's time
Results
00:03:30
00:03:15
00:03:00
00:02:45
00:02:30
00:02:15
Jan 1, 1900
1920
1940
1960
1980
2000
2020
Year

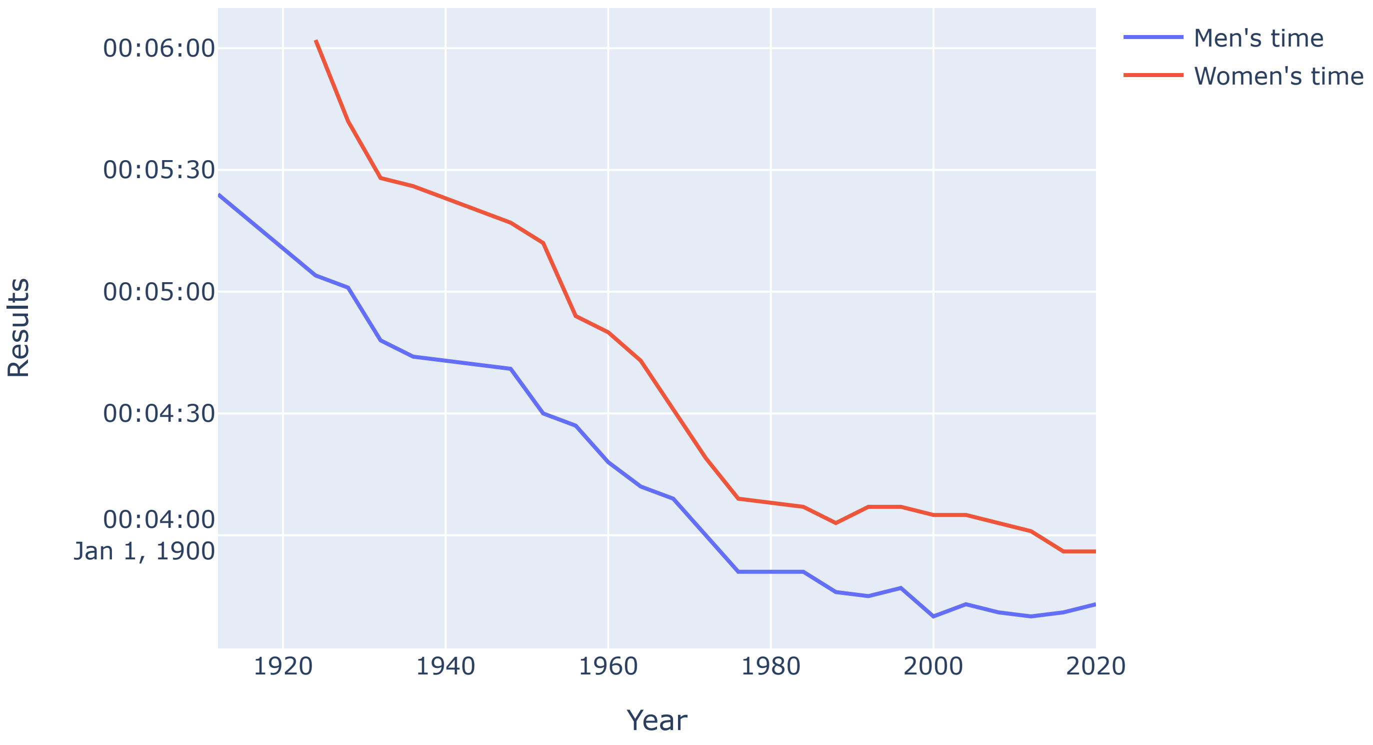

Butterfly 200m
Results
00:02:25
00:02:20
00:02:15
00:02:10
00:02:05
00:02:00
00:01:55
00:01:50
Jan 1, 1900
1960
1980
2000
2020
Year
Men's time
Women's time


Freestyle 200m
Results
00:02:10
00:02:05
00:02:00
00:01:55
00:01:50
00:01:45
Jan 1, 1900
1970
1980
1990
2000
2010
2020
Year
Men's time
Women's time

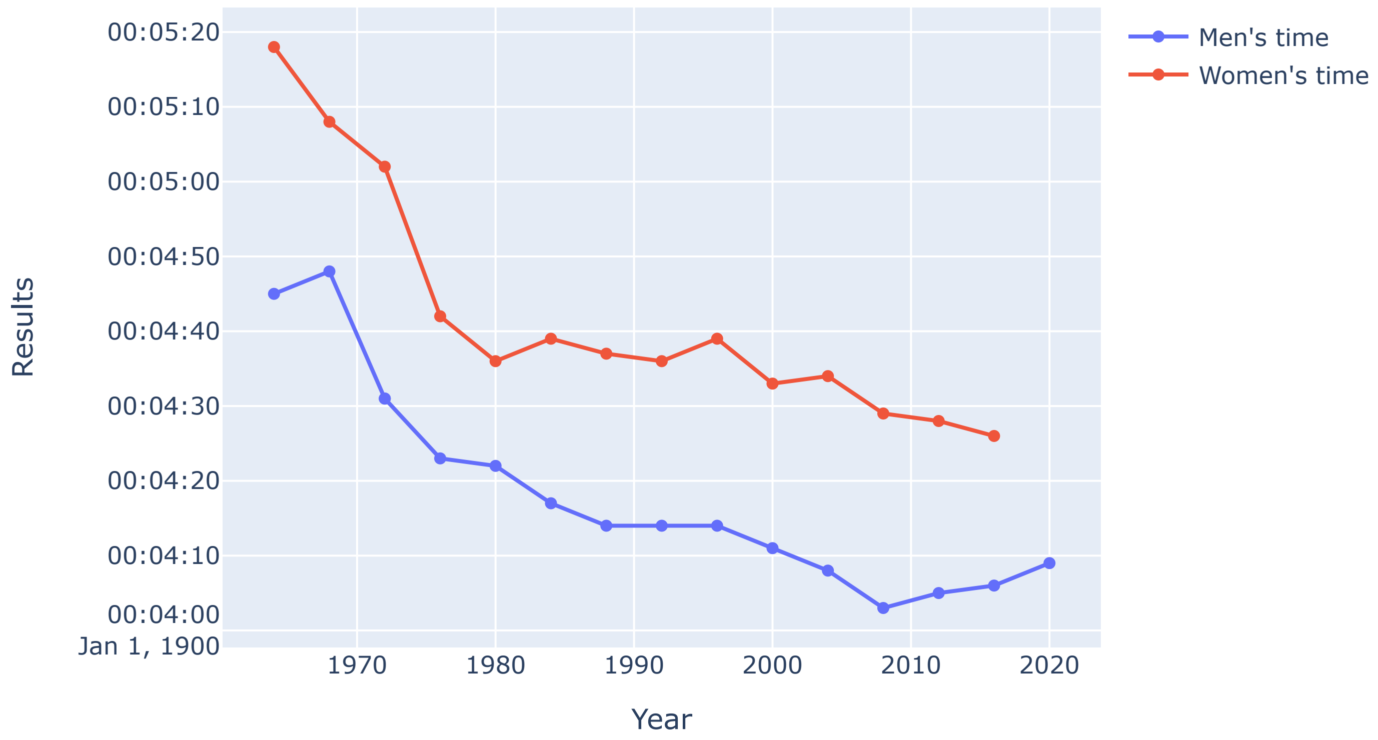
Individual medley 200m



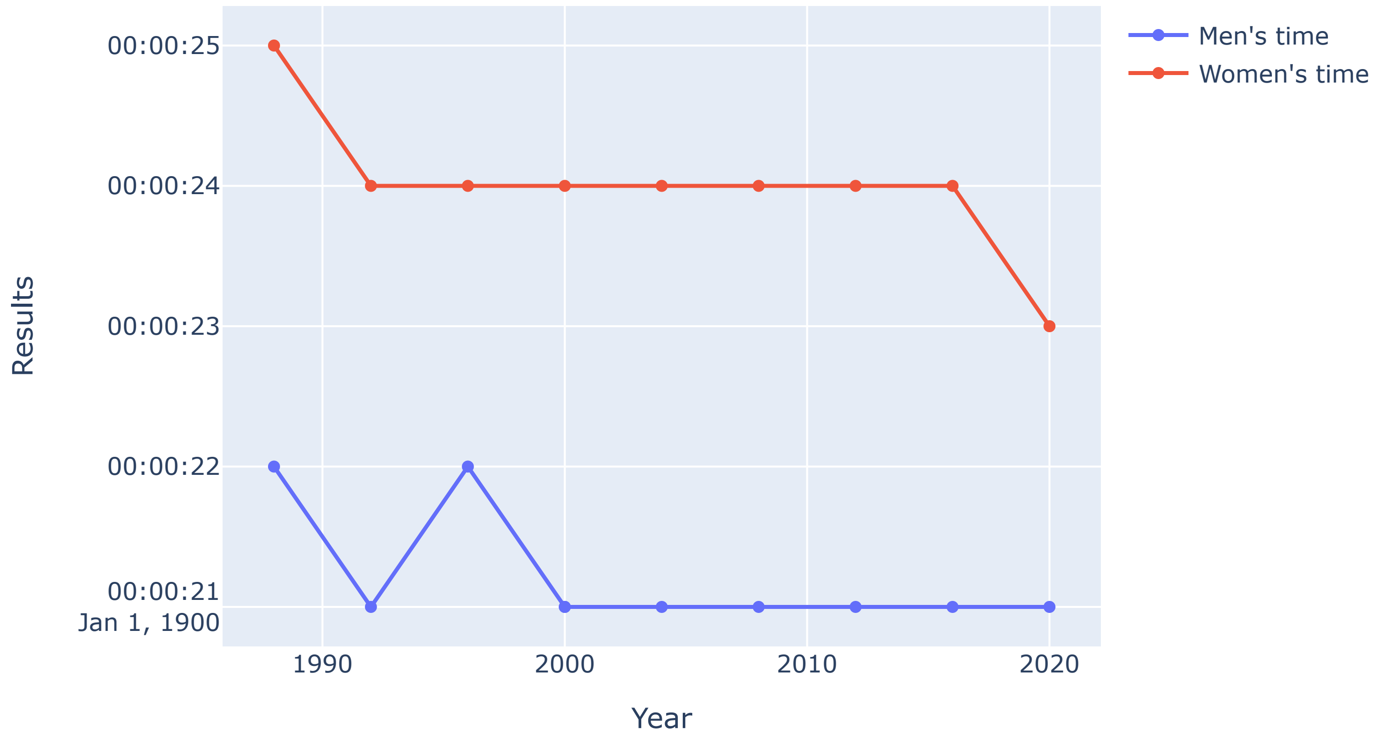
Freestyle 400m



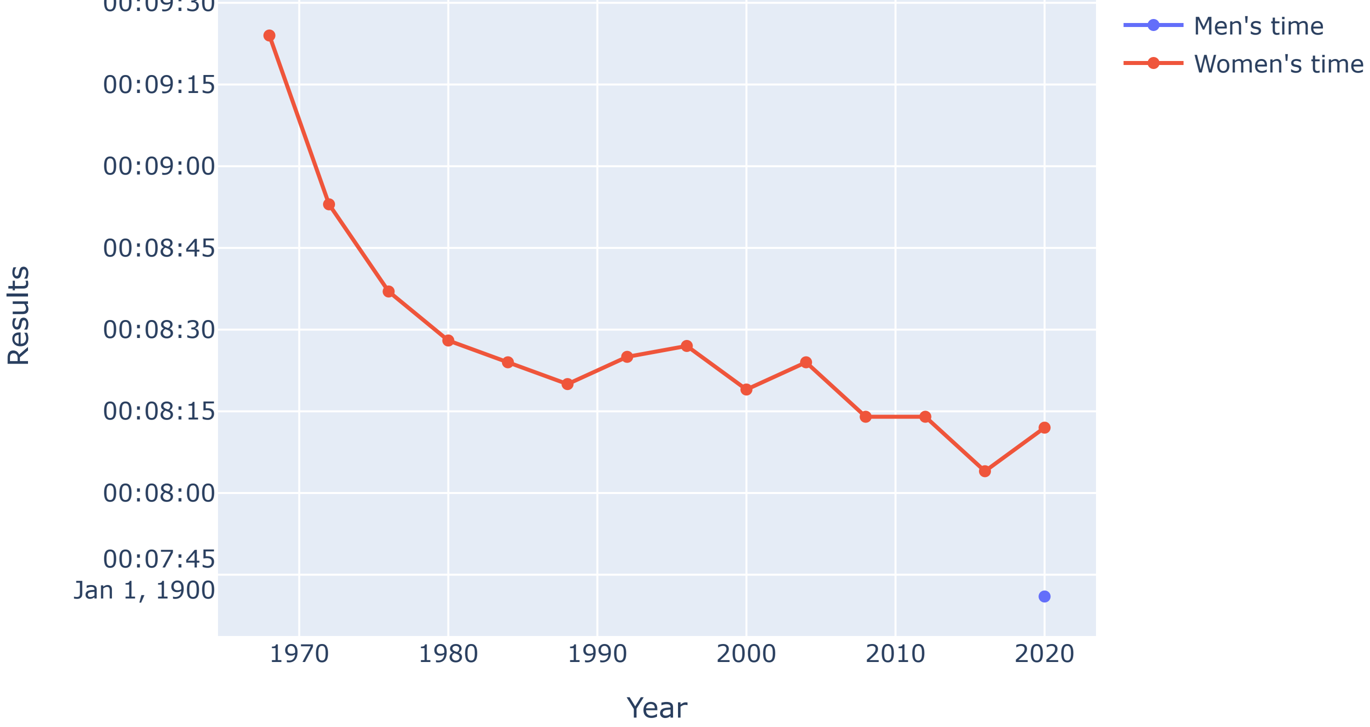
Individual medley 400m



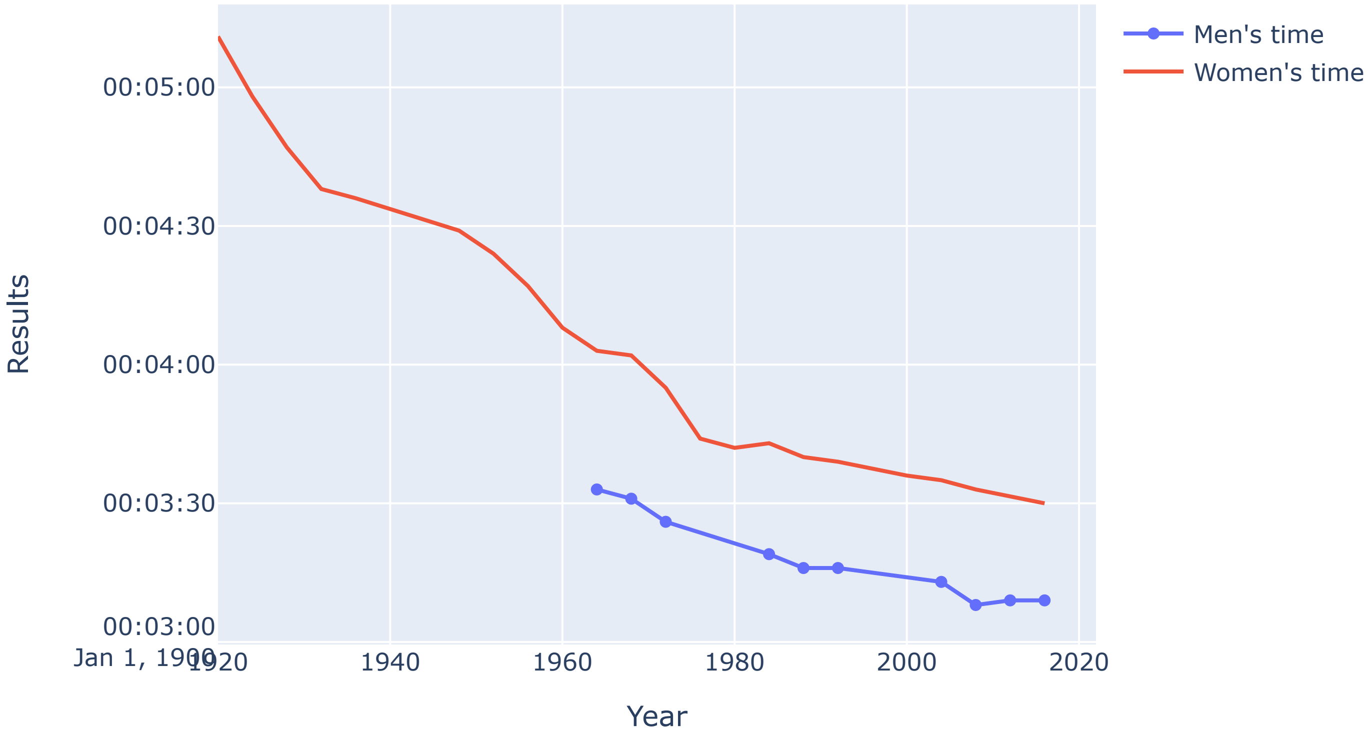
Freestyle 50m



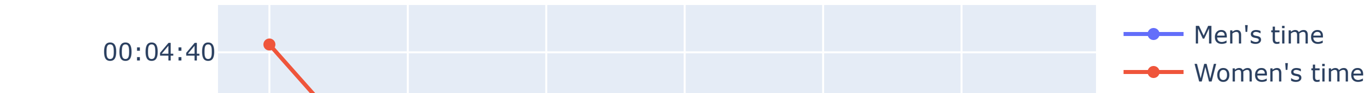
Freestyle 800m

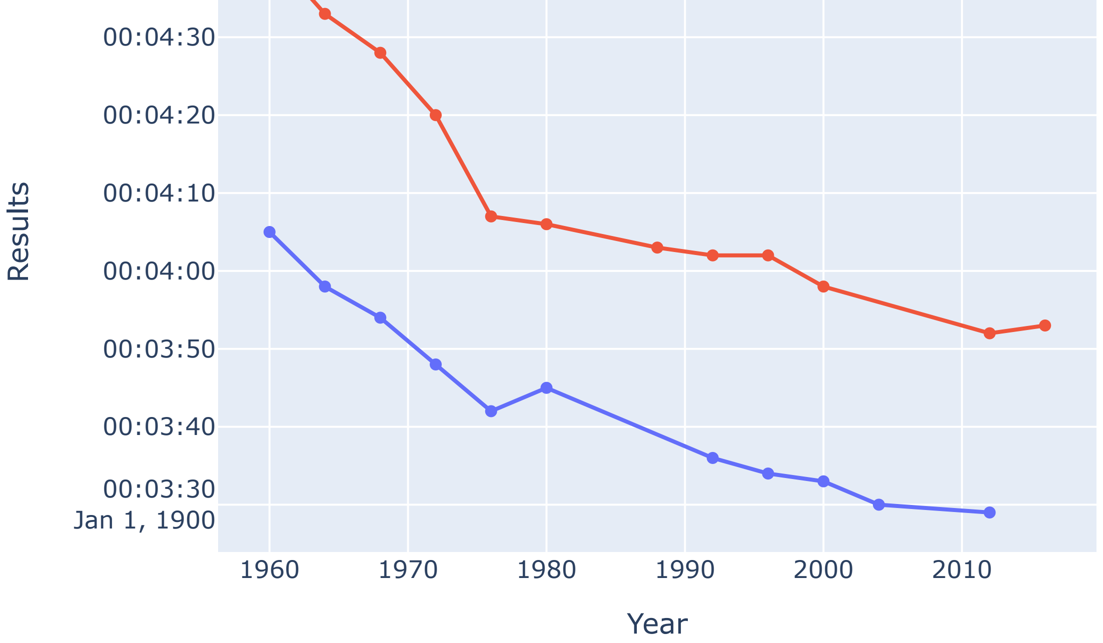


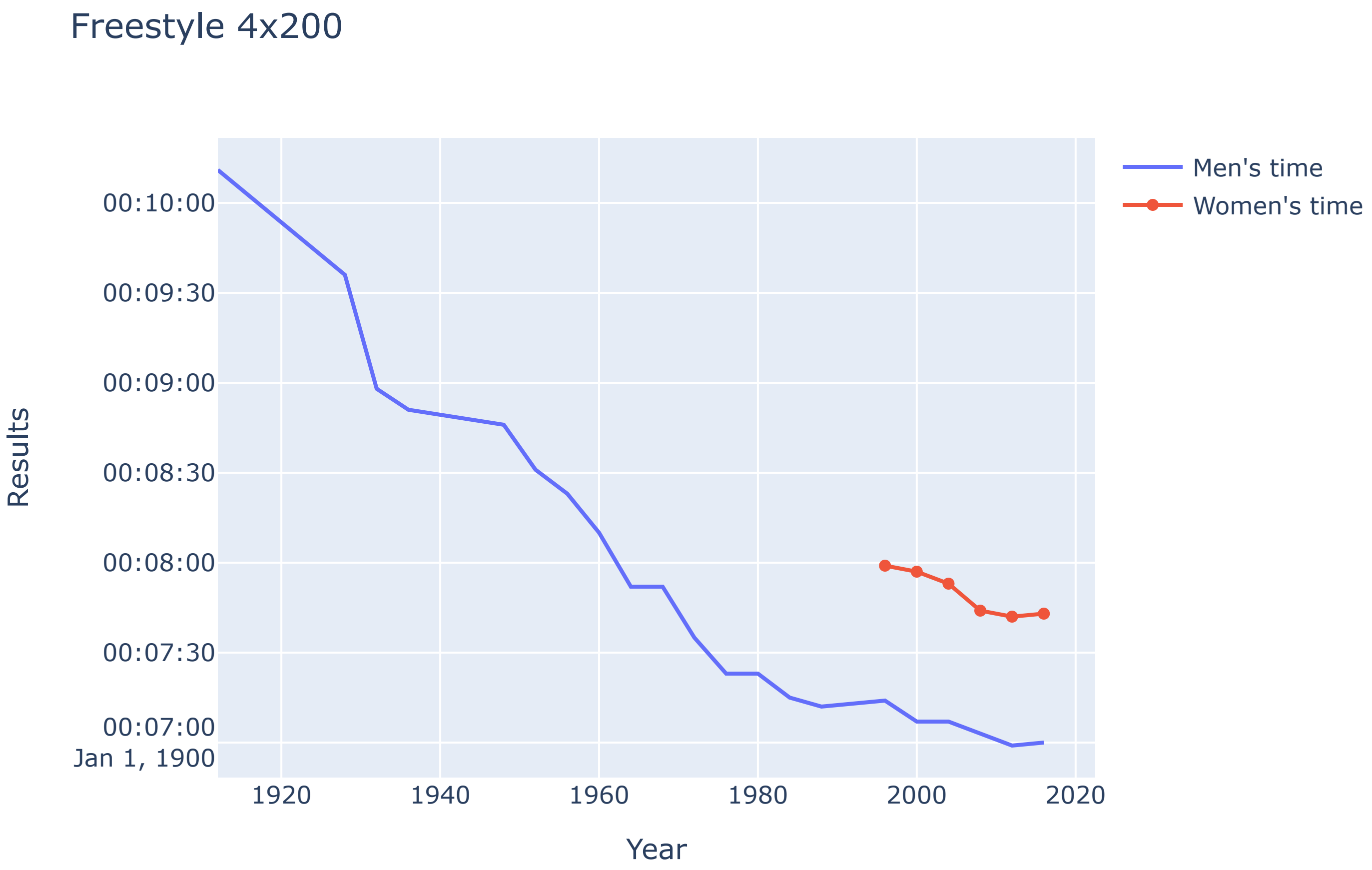
Freestyle 4x100



Medley 4x100






Freestyle 4x200

Both men and women follow similar trends in their timing behaviours

```
In [ ]:
```