



MANIPULATING DATAFRAMES WITH PANDAS

Case Study: Olympic Medals



Olympic medals dataset

	City	Edition	Sport	Discipline	Athlete	NOC	Gender	Event	Event_gender	Medal
0	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	100m freestyle	M	Gold
1	Athens	1896	Aquatics	Swimming	HERSCHMANN, Otto	AUT	Men	100m freestyle	M	Silver
2	Athens	1896	Aquatics	Swimming	DRIVAS, Dimitrios	GRE	Men	100m freestyle for sailors	M	Bronze
3	Athens	1896	Aquatics	Swimming	MALOKINIS, Ioannis	GRE	Men	100m freestyle for sailors	M	Gold
4	Athens	1896	Aquatics	Swimming	CHASAPIS, Spiridon	GRE	Men	100m freestyle for sailors	M	Silver
5	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	1200m freestyle	M	Bronze
6	Athens	1896	Aquatics	Swimming	HAJOS, Alfred	HUN	Men	1200m freestyle	M	Gold
7	Athens	1896	Aquatics	Swimming	ANDREOU, Joannis	GRE	Men	1200m freestyle	M	Silver
8	Athens	1896	Aquatics	Swimming	CHOROPHAS, Efstathios	GRE	Men	400m freestyle	M	Bronze
9	Athens	1896	Aquatics	Swimming	NEUMANN, Paul	AUT	Men	400m freestyle	M	Gold

Reminder: indexing & pivoting

- Filtering and indexing
 - One-level indexing
 - Multi-level indexing
- Reshaping DataFrames with `pivot()`
- `pivot_table()`



Reminder: groupby

- Useful DataFrame methods
 - `unique()`
 - `value_counts()`
- Aggregations, transformations, filtering



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Understanding the column labels



“Gender” and “Event_gender”

	NOC	Gender	Event	Event_gender	Medal
145	GRE	Men	heavyweight - two hand lift	M	Bronze
146	DEN	Men	heavyweight - two hand lift	M	Gold
147	GBR	Men	heavyweight - two hand lift	M	Silver
148	GRE	Men	open event	M	Bronze
149	GER	Men	open event	M	Gold
150	GRE	Men	open event	M	Silver
151	HUN	Men	1500m freestyle	M	Bronze
152	GBR	Men	1500m freestyle	M	Gold
153	AUT	Men	1500m freestyle	M	Silver
154	NED	Men	200m backstroke	M	Bronze



Reminder: slicing & filtering

- Indexing and slicing
 - `.loc[]` and `.iloc[]` accessors
- Filtering
 - Selecting by Boolean Series
 - Filtering null/non-null and zero/non-zero values



Reminder: Handling categorical data

- Useful DataFrame methods for handling categorical data:
 - `value_counts()`
 - `unique()`
 - `groupby()`
- `groupby()` aggregations:
 - `mean()`, `std()`, `count()`



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Constructing alternative country rankings



Counting distinct events

```
In [1]: medals['Sport'].unique() # 42 distinct events
```

```
Out[1]:
```

```
array(['Aquatics', 'Athletics', 'Cycling', 'Fencing', 'Gymnastics',  
      'Shooting', 'Tennis', 'Weightlifting', 'Wrestling', 'Archery',  
      'Basque Pelota', 'Cricket', 'Croquet', 'Equestrian', 'Football',  
      'Golf', 'Polo', 'Rowing', 'Rugby', 'Sailing', 'Tug of War',  
      'Boxing', 'Lacrosse', 'Roque', 'Hockey', 'Jeu de paume', 'Rackets',  
      'Skating', 'Water Motorsports', 'Modern Pentathlon', 'Ice Hockey',  
      'Basketball', 'Canoe / Kayak', 'Handball', 'Judo', 'Volleyball',  
      'Table Tennis', 'Badminton', 'Baseball', 'Softball', 'Taekwondo',  
      'Triathlon'], dtype=object)
```



Ranking of distinct events

- Top five countries that have won medals in the most sports
- Compare medal counts of USA and USSR from 1952 to 1988



Two new DataFrame methods

- `idxmax()`: Row or column label where maximum value is located
- `idxmin()`: Row or column label where minimum value is located



idxmax() Example

```
In [2]: weather = pd.read_csv('monthly_mean_temperature.csv',  
....:                        index_col='Month')
```

```
In [3]: weather # DataFrame with single column
```

```
Out[3]:
```

	Mean TemperatureF
Month	
Apr	53.100000
Aug	70.000000
Dec	34.935484
Feb	28.714286
Jan	32.354839
Jul	72.870968
Jun	70.133333
Mar	35.000000
May	62.612903
Nov	39.800000
Oct	55.451613
Sep	63.766667



Using idxmax()

```
In [4]: weather.idxmax() # Returns month of highest temperature
Out[4]:
Mean TemperatureF    Jul
dtype: object
```



Using `idxmax()` along columns

```
In [5]: weather.T # Returns DataFrame with single row, 12 columns
```

```
Out[5]:
```

Month	Apr	Aug	Dec	Feb	Jan	Jul	\
Mean TemperatureF	53.1	70.0	34.935484	28.714286	32.354839	72.870968	

Month	Jun	Mar	May	Nov	Oct	Sep
Mean TemperatureF	70.133333	35.0	62.612903	39.8	55.451613	63.766667

```
In [6]: weather.T.idxmax(axis='columns')
```

```
Out[6]:
```

```
Mean TemperatureF    Jul
```

```
dtype: object
```



Using idxmin()

```
In [7]: weather.T.idxmin(axis='columns')  
Out[7]:  
Mean TemperatureF    Feb  
dtype: object
```



MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Reshaping DataFrames for visualization



Reminder: plotting DataFrames

```
In [1]: all_medals = medals.groupby('Edition')['Athlete'].count()
```

```
In [2]: all_medals.head(6) # Series for all medals, all years
```

```
Out[2]:
```

```
Edition
```

1896	151
1900	512
1904	470
1908	804
1912	885
1920	1298

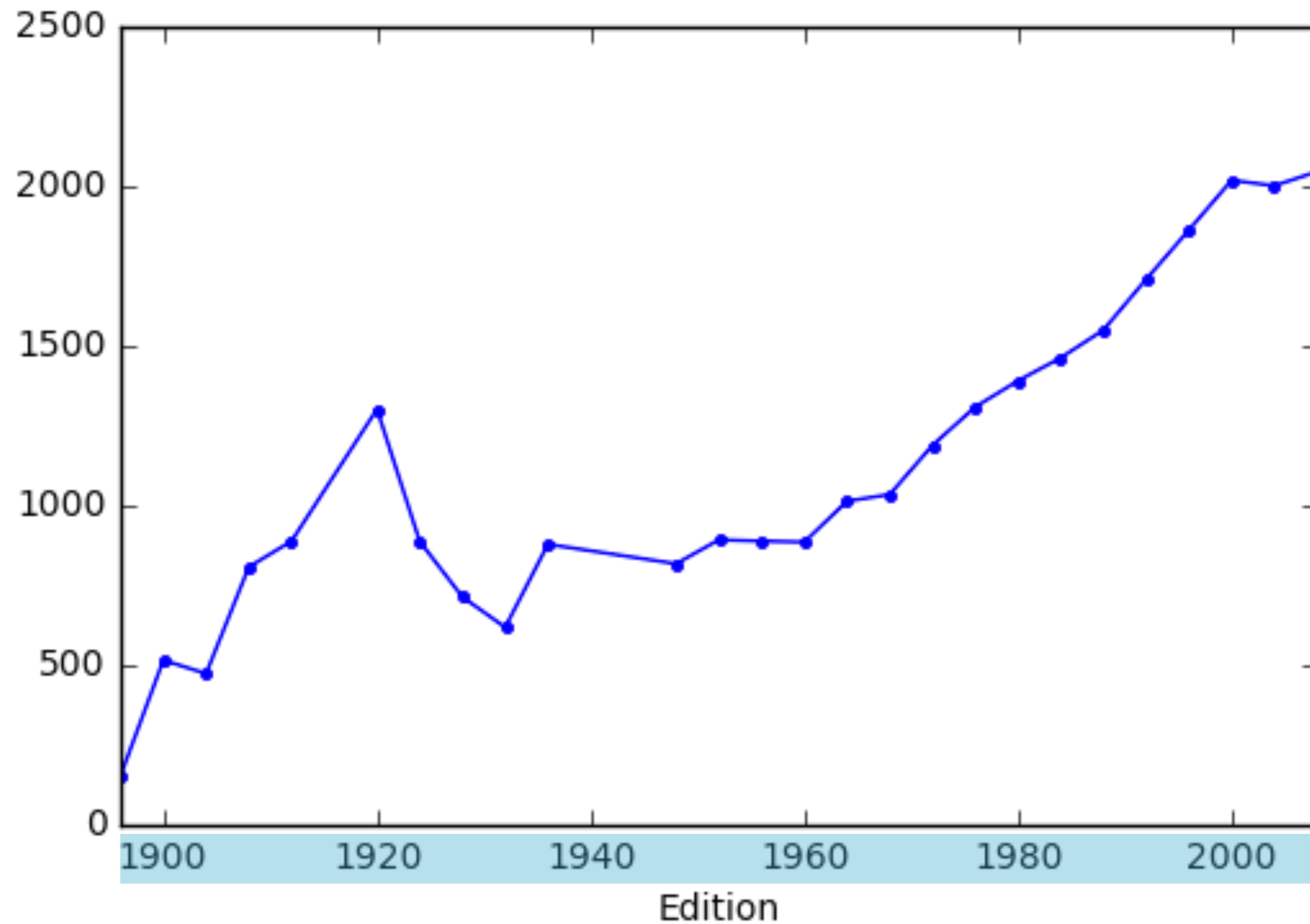
```
Name: Athlete, dtype: int64
```

```
In [3]: all_medals.plot(kind='line', marker='.')
```

```
In [4]: plt.show()
```



Plotting DataFrames





Grouping the data

```
In [5]: france = medals.NOC == 'FRA' # Boolean Series for France
```

```
In [6]: france_grps = medals[france].groupby(['Edition', 'Medal'])
```

```
In [7]: france_grps['Athlete'].count().head(10)
```

```
Out[7]:
```

Edition	Medal	
1896	Bronze	2
	Gold	5
	Silver	4
1900	Bronze	53
	Gold	46
	Silver	86
1908	Bronze	21
	Gold	9
	Silver	5
1912	Bronze	5

```
Name: Athlete, dtype: int64
```



Reshaping the data

```
In [8]: france_medals = france_grps['Athlete'].count().unstack()
```

```
In [9]: france_medals.head(12)    # Single level index
```

```
Out[9]:
```

Medal Edition	Bronze	Gold	Silver
1896	2.0	5.0	4.0
1900	53.0	46.0	86.0
1908	21.0	9.0	5.0
1912	5.0	10.0	10.0
1920	55.0	13.0	73.0
1924	20.0	39.0	63.0
1928	13.0	7.0	16.0
1932	6.0	23.0	8.0
1936	18.0	12.0	13.0
1948	21.0	25.0	22.0
1952	16.0	14.0	9.0
1956	13.0	6.0	13.0



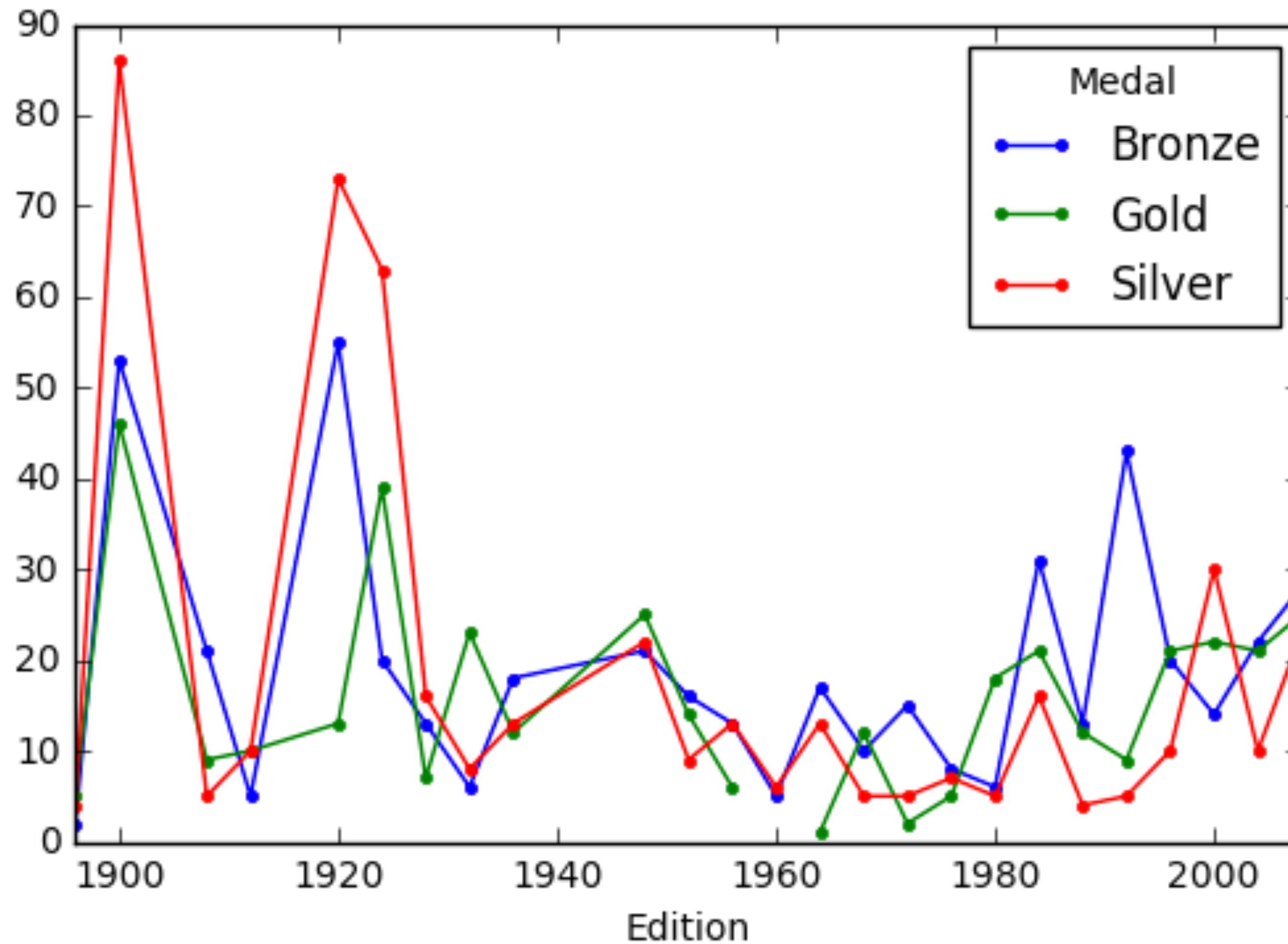
Plotting the result

```
In [10]: france_medals.plot(kind='line', marker='.')
```

```
In [11]: plt.show()
```



Plotting the result





MANIPULATING DATAFRAMES WITH PANDAS

Let's practice!



MANIPULATING DATAFRAMES WITH PANDAS

Final thoughts



You can now...

- Transform, extract, and filter data from DataFrames
- Work with pandas indexes and hierarchical indexes
- Reshape and restructure your data
- Split your data into groups and categories



MANIPULATING DATAFRAMES WITH PANDAS

**See you in the
next course!**