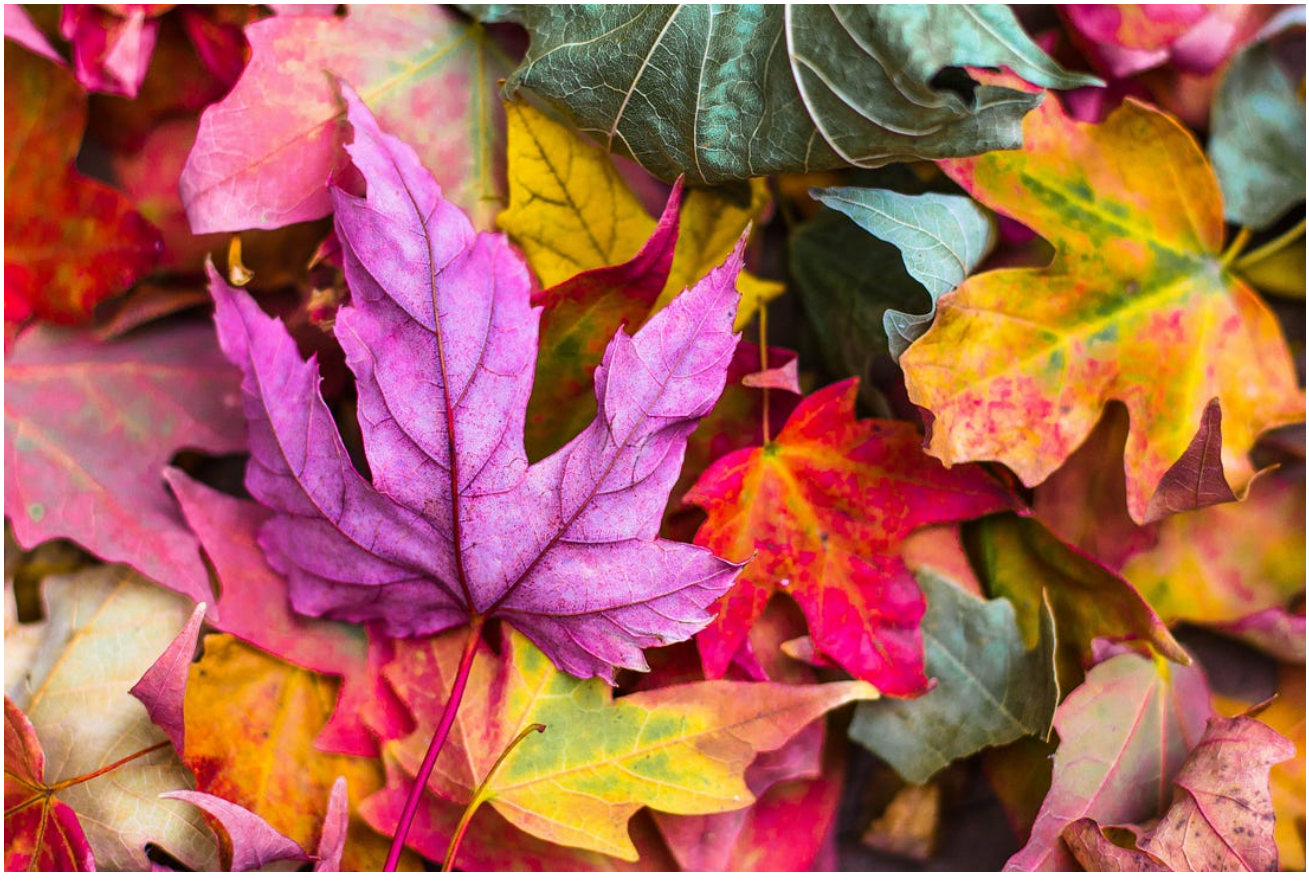


Photo by Jeremy Thomas on Unsplash

✦ Member-only story

HANDS-ON TUTORIAL

Create a Scientific Data Visualization from Scratch Using Plotnine in Python

Create a data visualization using Plotnine step by step

Audhi Aprilliant · Follow
Published in Towards Data Science
11 min read · Nov 29, 2020

Listen Share ••• More

Data pre-processing

As we have discussed in the previous story, the plotline can be used as an alternative to data visualization in Python. In this tutorial, we will continue the journey in data visualization that keeps focusing on one specific plot, namely the bar plot. Firstly, download the data of the *National Olympic Committee* (NOC) from my previous story.

```
# Dataframe manipulation
import pandas as pd
# Linear algebra
import numpy as np
# Data visualization with plotnine
from plotnine import *
import plotnine
# Set the figure size of plotnine
plotnine.options.figure_size = (6.4,4.8)
```

The `athlete event` the dataset has 271,116 records or rows and 15 columns or attributes.

```
# Import the athlete data
athlete_data = pd.read_csv('datasets/athlete_events.csv')
# Print the dimension of the data
print('Dimension of athlete data:\n{}'.format(len(athlete_data)),
      'rows and {}'.format(len(athlete_data.columns)),'columns')
athlete_data.head()
```

```
Dimension of athlete data:
271116 rows and 15 columns
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN

The athlete data in Olympics event 1896–2016 (Image by Author)

The `noc region` has 230 records or rows and 3 columns or attributes. The NaN in the `notes` column means that they are not recorded.

```
# Import the region data
regions_data = pd.read_csv('datasets/noc_regions.csv')
# Print the dimension of the data
print('Dimension of region data:\n{}'.format(len(regions_data)),
      'rows and {}'.format(len(regions_data.columns)),'columns')
regions_data.head()
```

```
Dimension of region data:
230 rows and 3 columns
```

	NOC	region	notes
0	AFG	Afghanistan	NaN
1	AHO	Curacao	Netherlands Antilles
2	ALB	Albania	NaN
3	ALG	Algeria	NaN
4	AND	Andorra	NaN

The National Olympic Committee data in Olympics event 1896–2016 (Image by Author)

We conduct the left join with the `athlete event` as left table and `NOC` column as the index level names to join on. It produces the cleaned data with 271,116 rows and 17 columns.

```
# Conduct left join between athlete and region data
full_data = athlete_data.merge(regions_data,on='NOC',how='left')
# Print the dimension of the data
print('Dimension of full data:\n{}'.format(len(full_data)),
      'rows and {}'.format(len(full_data.columns)),'columns')
full_data.head()
```

```
Dimension of full data:
271116 rows and 17 columns
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal	region	notes
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN	China	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN	China	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN	Denmark	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold	Denmark	NaN
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN	Netherlands	NaN

The Olympics data (athlete and NOC) in Olympics event 1896–2016 (Image by Author)

```
# Create cross tabulation
medal_noc = pd.crosstab([full_data['Year'], full_data['NOC']], full_data['Me
# Remove index name
medal_noc.columns.name = None
# Remove last row for total column attribute
medal_noc = medal_noc.drop([medal_noc.shape[0] - 1], axis = 0)
```

	Year	NOC	Bronze	Gold	Silver	All
0	1896	AUS	1	2	0	3
1	1896	AUT	2	2	1	5
2	1896	DEN	3	1	2	6
3	1896	FRA	2	5	4	11
4	1896	GBR	3	3	3	9
...	...	...	...	...	...	...
1437	2016	BRN	0	1	1	2
1438	2016	GRN	0	0	1	1
1439	2016	FIJ	0	13	0	13
1440	2016	JOR	0	1	0	1
1441	2016	KOS	0	1	0	1

1442 rows × 6 columns

Cross-tabulation between countries and their medal acquisition 1896–2016 (Image by Author)

Count the medals acquired by each country and find out a country that became the general winner each year.

```
# Data aggregation - group by
medal_noc_year = medal_noc.loc[medal_noc.groupby('Year')['All'].idxmax()].so
```

	Year	NOC	Bronze	Gold	Silver	All
6	1896	GRE	20	10	18	48
13	1900	FRA	82	52	101	235
42	1904	USA	125	128	141	394
52	1906	GRE	30	24	48	102
67	1908	GBR	90	147	131	368
97	1912	SWE	25	103	62	190
108	1920	USA	38	111	45	194
132	1924	USA	50	98	46	194
165	1928	USA	19	53	30	102
199	1932	USA	64	91	68	223
225	1936	GER	61	97	73	231
259	1948	USA	41	93	34	168

Countries that had been a general winner since 1896 (Image by Author)

Finally, we are able to count the number of countries that had been a general winner since 1896. This data is used to generate the data viz, like for this tutorial bar plot.

```
# Top ten countries that won the most Olympic medals
medal_noc_count = pd.DataFrame(medal_noc_year['NOC'].value_counts()).reset_i
medal_noc_count.columns = ['NOC','Count']
```

	NOC	Count
0	USA	16
1	URS	8
2	CAN	2
3	GER	2
4	GRE	2
5	SWE	1
6	EUN	1
7	FIN	1
8	GBR	1
9	FRA	1

Number of countries that had been a general winner since 1896 (Image by Author)

Hands-on with the Plotnine

Bar plot is the most common type of data visualization. It is a graph or plot that represents the categorical data in a bar. In another opportunity, you can also try to code and make visualization using another type of graph or plot.

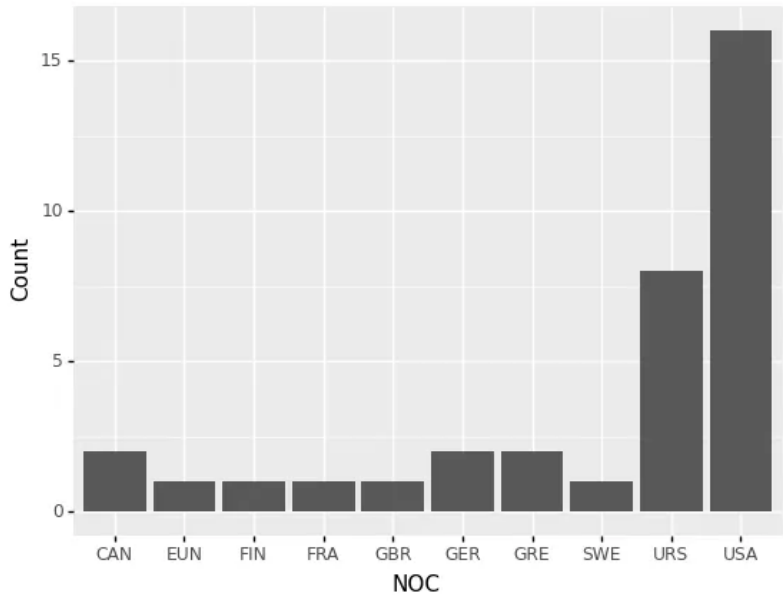
1 Basic bar plot

It will be the baseline of our bar plot. We just create a basic bar plot without any modification, theme, legend description, color manipulation, etc. We only need the `ggplot()` function to create the first layer of background and `geom_bar()` to convert our data into a bar plot. The horizontal axis (*X*) represents the categories or in this case, is *NOC* and the vertical (*Y*) represents their frequencies.

For `geom_bar()`, the default behaviour is to count the rows for each *X* value. It

doesn't expect a *Y*-value, since it's going to count that up itself — in fact, it will flag a warning if we give it one, since it thinks we're confused. How aggregation is to be performed is specified as an argument to `geom_bar()`, which is `stat = 'count'` for the default value. *Read more*.

```
# Create a basic bar plot
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             stat='identity')
)
```



The basic bar plot (Image by Author)

2 Add plot and axis title

In the previous section, our plot is successfully created but it has not enough information for the audience. When we present that to the conference or create a journal, the information must be misleading. What does that plot tell us? The information like a plot title must be included. The `labs()` function has an argument title to add the plot title to our plot. Then, we're able to modify both the horizontal and vertical labels using `xlab()` and `ylab()`.

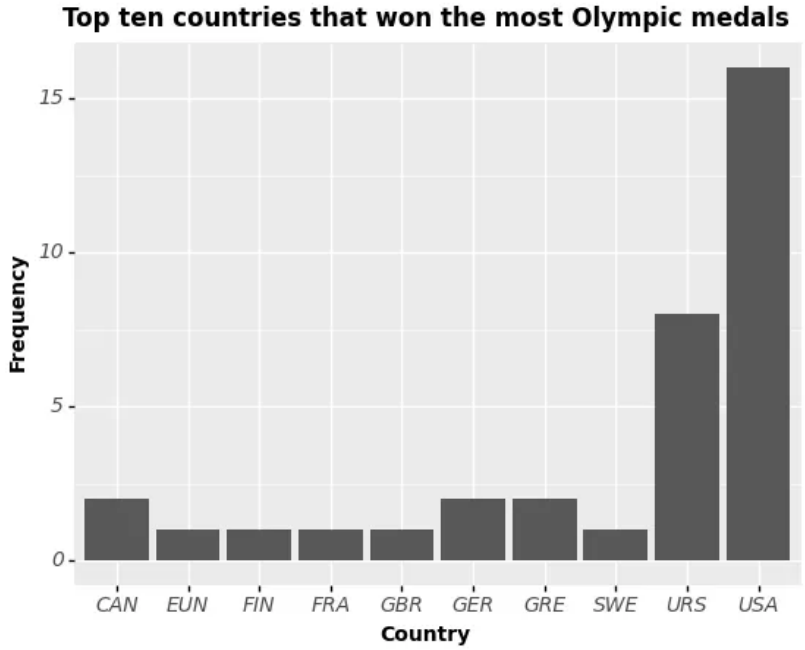
Furthermore, in order to manipulate the font style, just add other functions `theme()` to our main script. For the whole font setting like font size, font style, etc,

it's provided by an argument `text()`.

- `text()` — for the whole text element
- `axis_title()` — a setting for both the horizontal and vertical axis title
- `axis_text()` — a setting for the axis labels
- `plot_title()` — a setting for the plot title

> ***Note:*** *there are a lot of arguments in* `theme()` *function. To read more about it, please directly click here*

```
# Add plot and axis title
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             stat='identity')+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))
)
```



The bar plot with plot title and axis labels modification (Image by Author)

3 Order the categories decreasingly

The main objective of data viz is to summarize and represent the data. Data visualization contains both art and science aspects. So, that means we must consider the aesthetics aspect. In artwork, there are a lot of aspects to consider, such as ***balance***. It needs to implement in the data viz in order to help people understand the viz with a quick look.

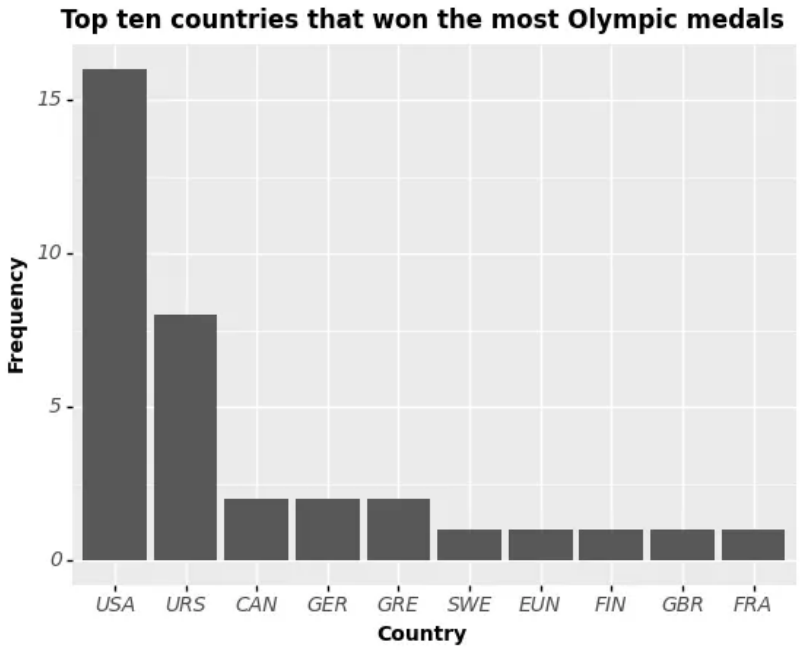
To reorder the horizontal labels which are categorical data, we can easily use the `scale_x_discrete()` function. An argument `limits` determines the ordered labels in the list.

```
# Input
medal_noc_count['NOC'].tolist()

# Output
# ['USA','URS','CAN','GER','GRE','SWE','EUN','FIN','GBR','FRA']
```

Now, let's create the bar plot with an ordered label! Does that look more beautiful and easier to understand, right?

```
# Order the categories decreasingly
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             stat='identity')+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))
)
```



The bar plot with ordered categories (Image by Author)

4 Fill the bar with a certain color

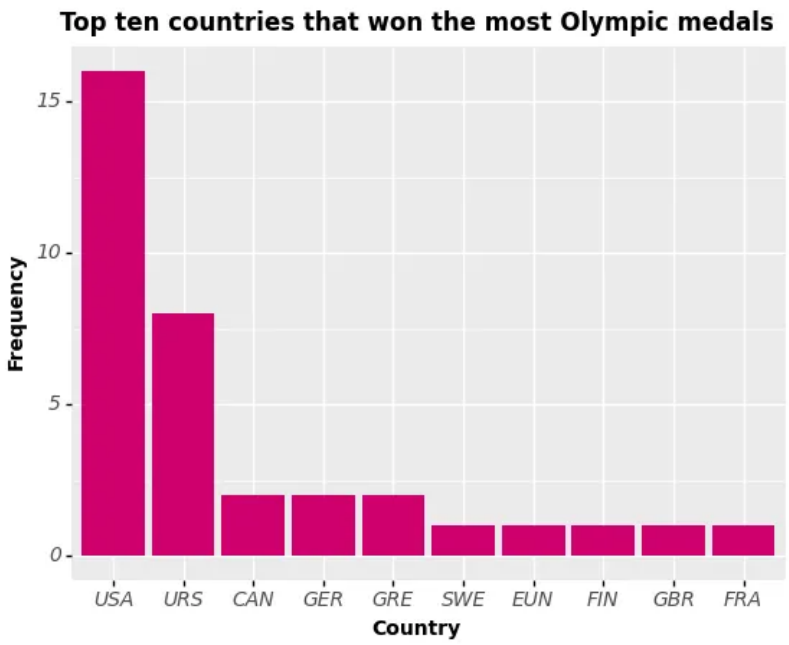
As art and also has the objective to make people understand the data, color becomes an important aspect to consider with.

The `plotnine` provides an argument `fill` to modify the plot's color. Colors and fills can be specified in the following ways:

- A name, for instance: `'red'`, `'blue'`, `'black'`, `'white'`, `'green'`, etc.
- An RGB specification

> ***Note:*** *to interact with color in plotnine, there are three argument to learn, such as* `colour`, `fill`, *and* `alpha`. *Read more here*

```
# Fill the bar with a certain colour
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill='#c22d6d',
             stat='identity')+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))
)
```



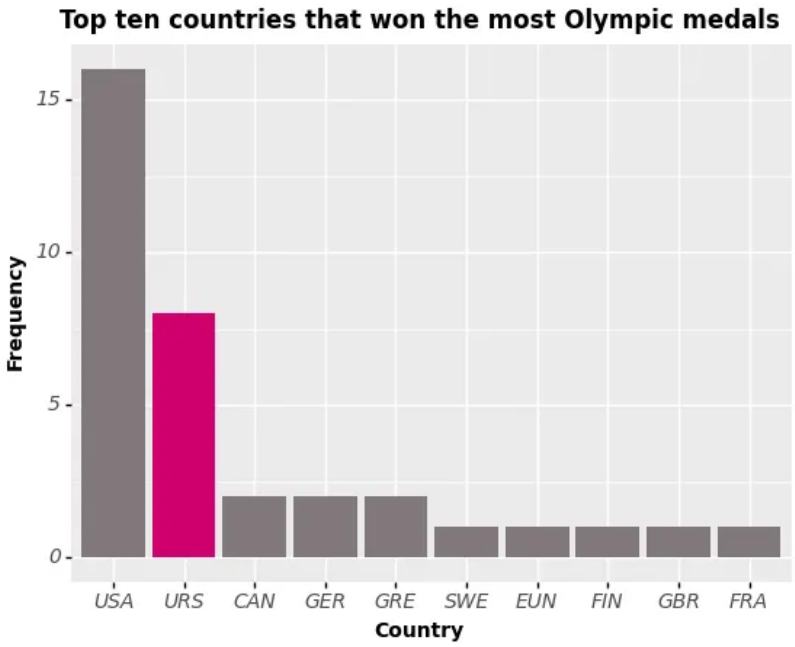
The bar plot with color modification (Image by Author)

5 Highlight the interesting category

As mentioned in the previous section, color is key to data viz. It helps us attract the audience's attention or focus and guide our data story. Color is often used to highlight the important part of our plot or graph. So, we will remake the color of the plot only to our highlight part, like URS, which is our focus right now.

The argument `fill` will be divided into two-part, URS and others. The URS will be `'#c22d6d'` and `'#80797c'` for others. The `np.where()` function determines the colors based on our criteria above.

```
# Highlight the interesting category
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill=np.where(medal_noc_count['NOC'] == 'URS','#c22d6d','#80797
             stat='identity')+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))
)
```



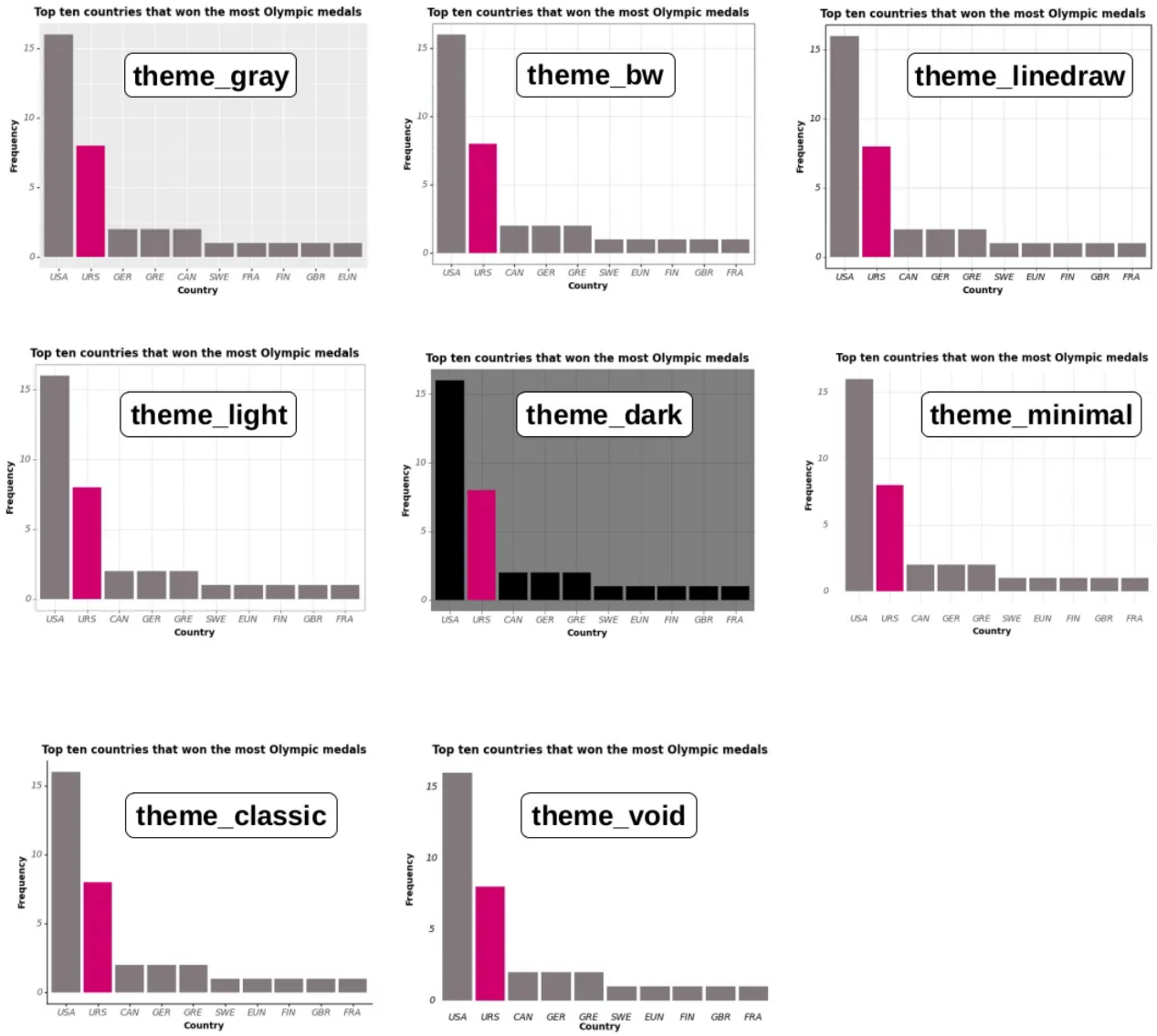
The bar plot with the highlighted part (Image by Author)

6 Change the plot theme

Plotnine provides us with a lot of themes. Which theme will we use for our data viz is our consideration? Each theme has both advantages and disadvantages. But to switch from one to the other, it's quite simple as click and drag (*we only need to write one row of code*).

The grammar of graphics themes:

- `theme_gray`
- `theme_bw`
- `theme_linedraw`
- `theme_light`
- `theme_dark`
- `theme_minimal`
- `theme_classic`
- `theme_void`

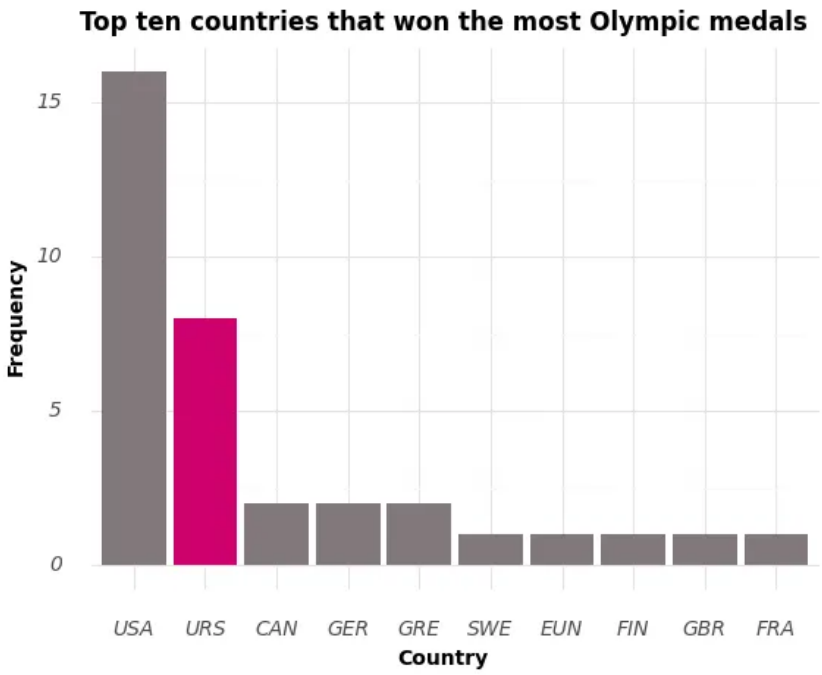


Plot themes provided by plotnine (Image by Author)

Let's create a data viz with a `theme_minimal()` function!

```
# Change the plot theme according to the needs
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill=np.where(medal_noc_count['NOC'] == 'URS','#c22d6d','#80797
             stat='identity')+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme_minimal()+
    theme(text=element_text(family='DejaVu Sans',
```

```
                              size=10),
            axis_title=element_text(face='bold'),
            axis_text=element_text(face='italic'),
            plot_title=element_text(face='bold',
                                    size=12))
  )
```



The bar plot with the minimal theme (Image by Author)

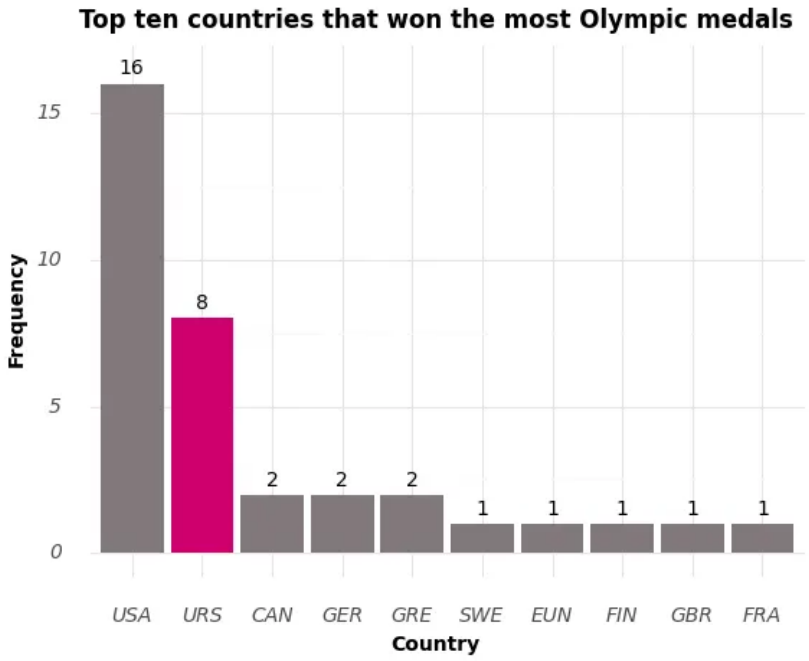
7 Annotate each category

To annotate the categories, there are several options to try, such as using `geom_text()` or `geom_label()`. We will demonstrate them in this tutorial and try to find out the differences. The `geom_text()` adds text directly to the plot while `geom_label()` draws a rectangle underneath the text, making it easier to read. There is an argument template for both `geom_text()` or `geom_label()` to use as follows.

- `x`: the position of labels in the *X-axis*
- `y`: the position of labels in the *Y-axis*
- `labels`: labels to annotate the categories
- `size`: font size of labels
- `nudge_x`: horizontal adjustment to nudge labels

- `nudge_y` : vertical adjustment to nudge labels

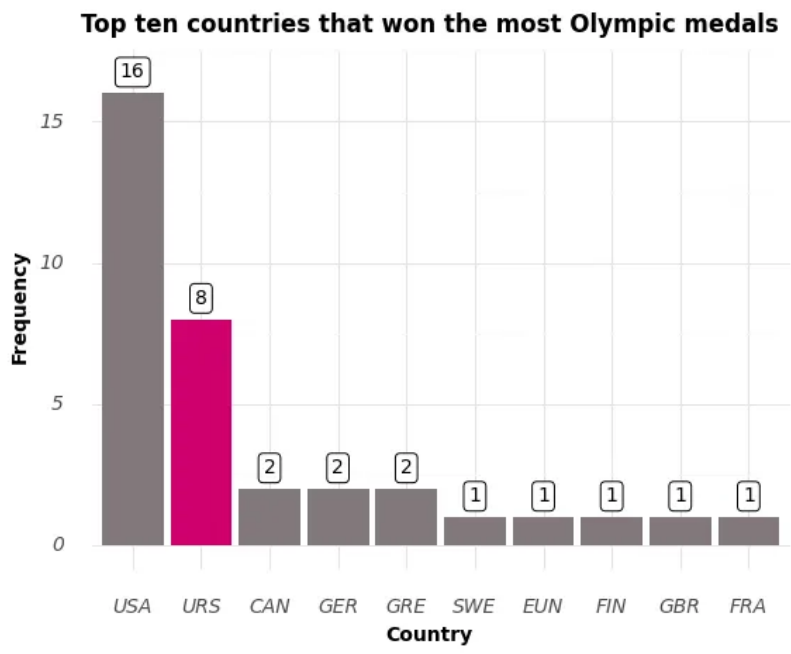
```
# Annotate each category using geom_text
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill=np.where(medal_noc_count['NOC'] == 'URS','#c22d6d','#80797
             stat='identity')+
    geom_text(aes(x='NOC',
                  y='Count',
                  label='Count'),
              size=10,
              nudge_y=0.5)+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme_minimal()+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))
)
```



The bar plot with text annotation using geom_text function (Image by Author)

Now, let's try the other option to annotate the categories, namely `geom_label()`!

```
# Annotate each category using geom_label
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill=np.where(medal_noc_count['NOC'] == 'URS','#c22d6d','#80797
             stat='identity')+
    geom_label(aes(x='NOC',
                   y='Count',
                   label='Count'),
               size=10,
               nudge_y=0.75)+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme_minimal()+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))
)
```



The bar plot with text annotation using geom_label function (Image by Author)

> *Note: to explore about technique to annotate text using plotnine, read this article here*

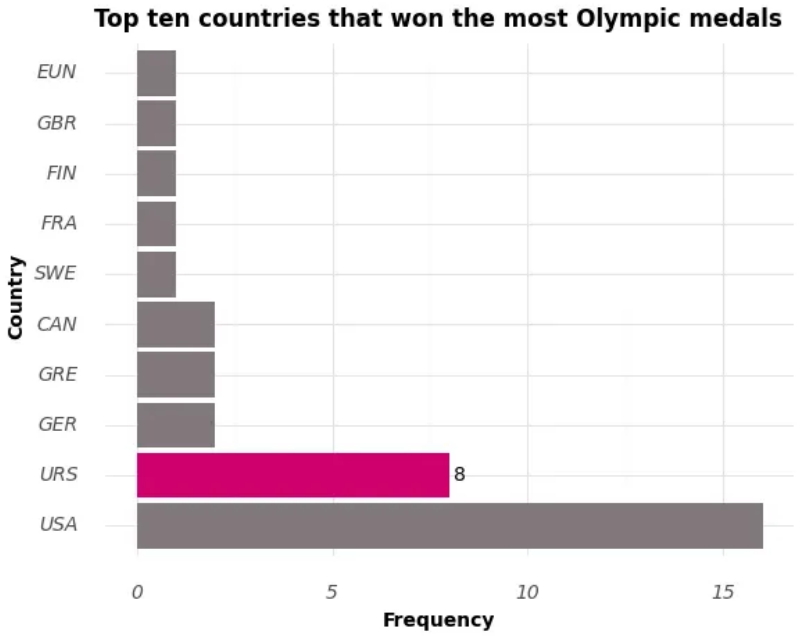
8 Flip the coordinate

Sometimes, for some data viz, that will be great if we can rotate the horizontal axis to vertical and vice versa or in other words, flip the coordinate. The plotnine gives a function `coord_flip()` to do that thing easily.

And in this, we are talking about how to get the audience's attention by showing them a highlighted part, for instance, giving a text annotation only for the URS. To do so, we might filter the data for only the URS rows and assign the label into `geom_text()` function.

> *Note: because we have already assign the **medal_noc_count** in `ggplot()` function, we need to mapping the filtered URS data in `geom_text()` using argument `mapping`*

```
# Flip the coordinate
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill=np.where(medal_noc_count['NOC'] == 'URS','#c22d6d','#80797
             position = position_dodge(0.9),
             stat='identity')+
    geom_text(data=medal_noc_count[medal_noc_count['NOC'] == 'URS'],
              mapping=aes(x='NOC',
                          y='Count',
                          label='Count'),
              size=10,
              nudge_y=0.25)+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme_minimal()+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))+
    coord_flip()
)
```



The bar plot flipped by coord_flip function (decreasingly) (Image by Author)

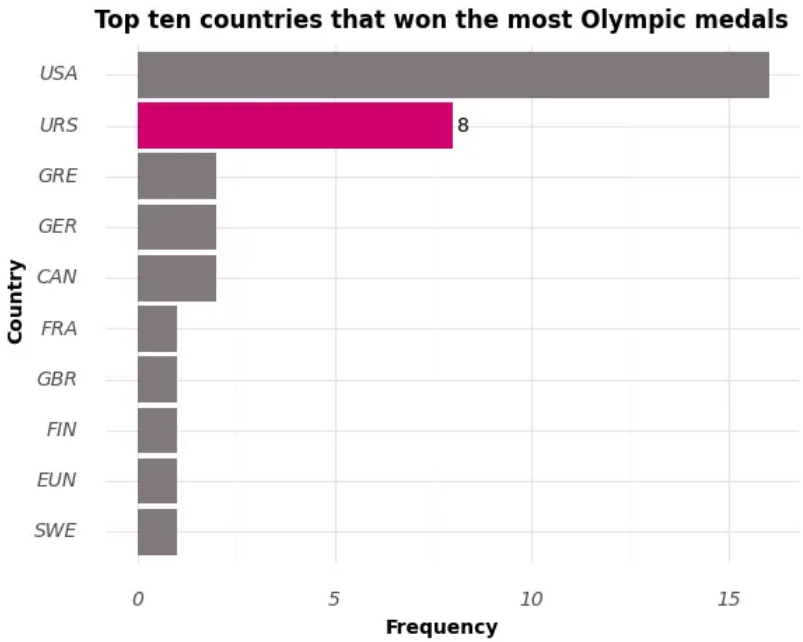
Did you get something wrong? Yes, it's about the order. We must rearrange the category based on their numbers (increasingly). It's too tricky but quite simple to do. We just need to sort the numbers increasingly instead of decreasingly.

```
# Reverse order the categories for flipping the coordinate
medal_noc_count.sort_values('Count',ascending=True,inplace=True)
```

	NOC	Count
5	SWE	1
6	FRA	1
7	FIN	1
8	GBR	1
9	EUN	1
2	GER	2
3	GRE	2
4	CAN	2
1	URS	8
0	USA	16

An ordered data increasingly (Image by Author)

When the data is already sorted increasingly, we can visualize it!



The bar plot flipped by coord_flip function (Increasingly) (Image by Author)

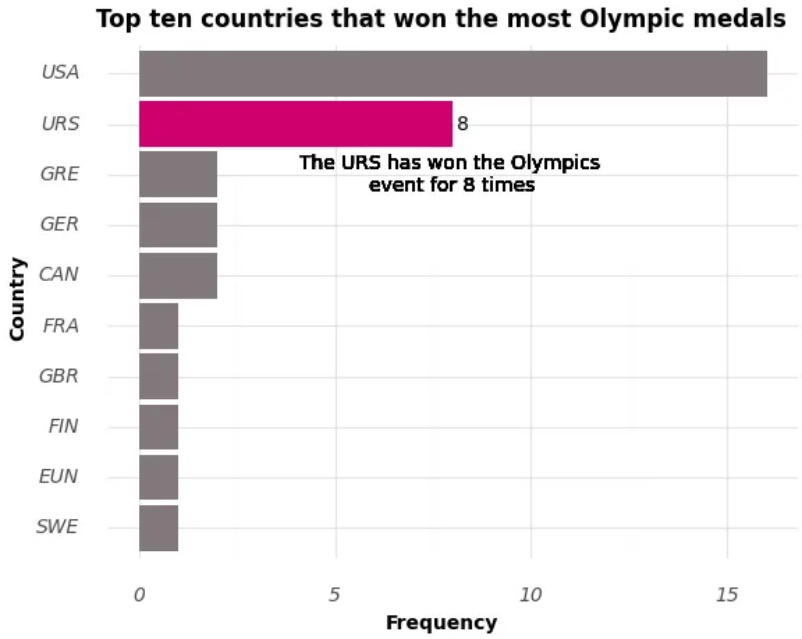
9 Add a description for the highlighted part

When the URS data becomes the highlighted part for this tutorial, it's recommended to annotate some text to it. It might be the findings or insights. To do so, we use `geom_text()` function. The `x` is assigned to `8` because of the URS position in the vertical axis (from the bottom) and the `y` is assigned to `8` because of the URS label. The statement `y=8` directly makes the centre paragraph of annotation to the `8`.

> ***Note:*** *the* `\n` *is an escape character which means insert a newline in the text*

```
# Add the description about the highlighted bar
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill=np.where(medal_noc_count['NOC'] == 'URS','#c22d6d','#80797
             position=position_dodge(0.9),
             stat='identity')+
    geom_text(data=medal_noc_count[medal_noc_count['NOC'] == 'URS'],
              mapping=aes(x='NOC',
                  y='Count',
                  label='Count'),
              size=10,
```

```
                nudge_y=0.25)+
    geom_text(x=8,
                y=8,
                label='The URS has won the Olympics \nevent for 8 times',
                size=10,
                nudge_y=0.25)+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme_minimal()+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          axis_text=element_text(face='italic'),
          plot_title=element_text(face='bold',
                                  size=12))+
    coord_flip()
)
```

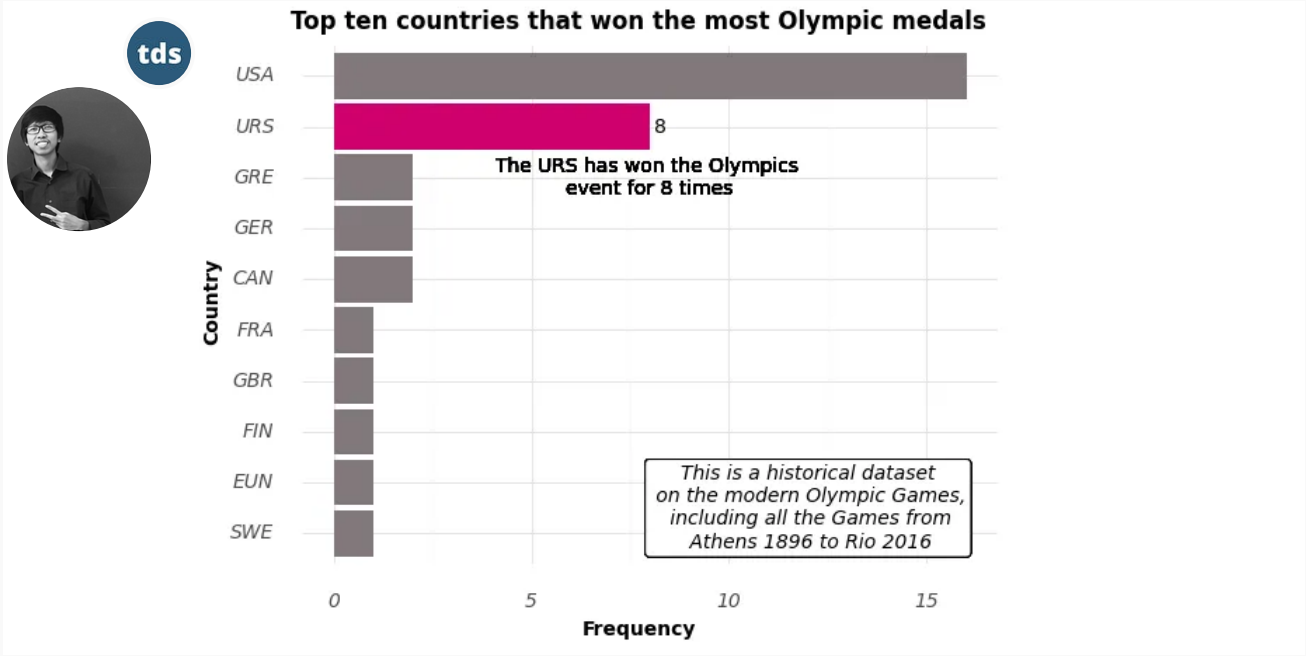


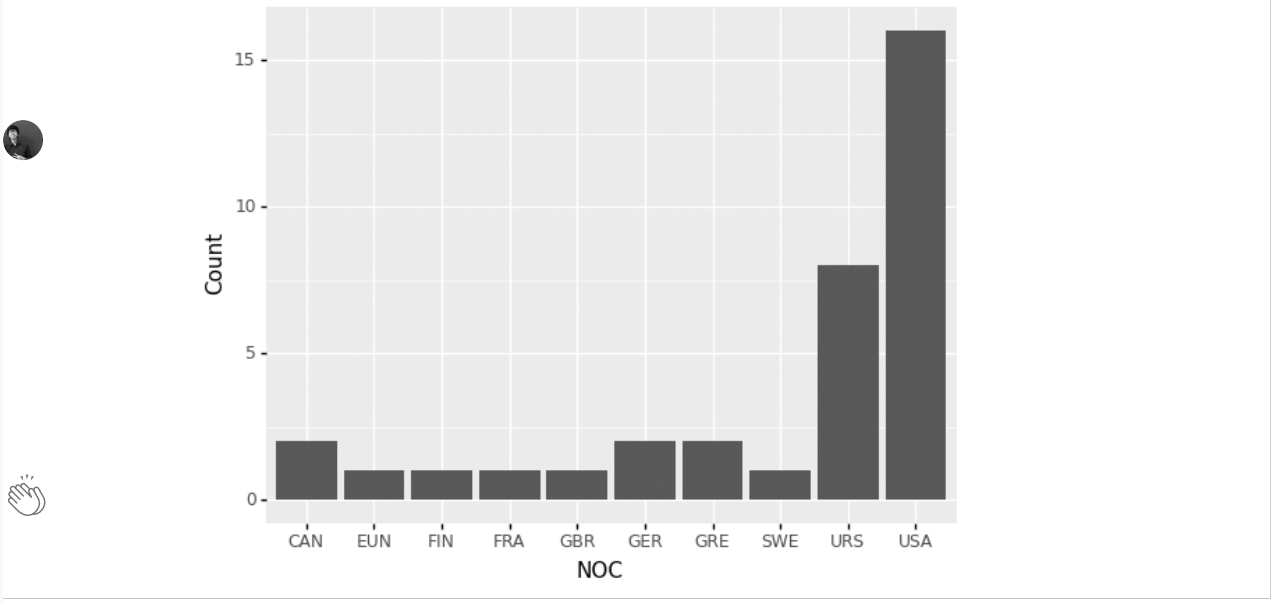
Highlight the bar plot using text annotation (Image by Author)

10 Add the data description in an empty space

The last section about how to create scientific data viz is a description of the data. It helps the audience understand the data source, its periods, sampling methods, etc. To make it tidy and clear, we are able to try `geom_label()` function to it. The label position will be too tricky because we need to conduct trial and error to make sure the description is in the right position.

```
# Add the description about the data in an empty space
(
    ggplot(data=medal_noc_count)+
    geom_bar(aes(x='NOC',
                 y='Count'),
             fill=np.where(medal_noc_count['NOC'] == 'URS','#c22d6d','#80797
             position = position_dodge(0.9),
             stat='identity')+
    geom_text(data=medal_noc_count[medal_noc_count['NOC'] == 'URS'],
              mapping=aes(x='NOC',
                  y='Count',
                  label='Count'),
              size=10,
              nudge_y=0.25)+
    geom_text(x=8,
              y=8,
              label='The URS has won the Olympics \nevent for 8 times',
              size=10,
              nudge_y=0.25)+
    geom_label(x=1.5,
               y=12,
               label='This is a historical dataset\n on the modern Olympic G
               nudge_y=3,
               family='DejaVu Sans',
               size=10,
               fontstyle='italic',
               boxstyle = 'round')+
    labs(title='Top ten countries that won the most Olympic medals')+
    xlab('Country')+
    ylab('Frequency')+
    scale_x_discrete(limits=medal_noc_count['NOC'].tolist())+
    theme_minimal()+
    theme(text=element_text(family='DejaVu Sans',
                            size=10),
          axis_title=element_text(face='bold'),
          a                                    ),
          p                                    size=12))+
    _flip()
)
```





Create a scientific plot from basic (Image by Author)