Stacked Bar: Lego Dataset

In [154…
```
import pandas as pd
import matplotlib as mlp
import matplotlib.pyplot as plt
import numpy as np
pd.set_option("display.precision", 2)

mlp.__version__
```

Out[154]:
```
'3.5.3'
```

Dataset

Using the original dataset for the stacked bar: Lego Dataset and exporting it onto this assignment page

In [155…
```
url_lego = 'https://raw.githubusercontent.com/3eltran23/InfoViz/master/LegoTotalSetsPe

# reading dataframe/set
# borrowed from github
df_lego = pd.read_csv(url_lego)
df_lego = df_lego[df_lego['year']> 2017]
df_lego.head()
```

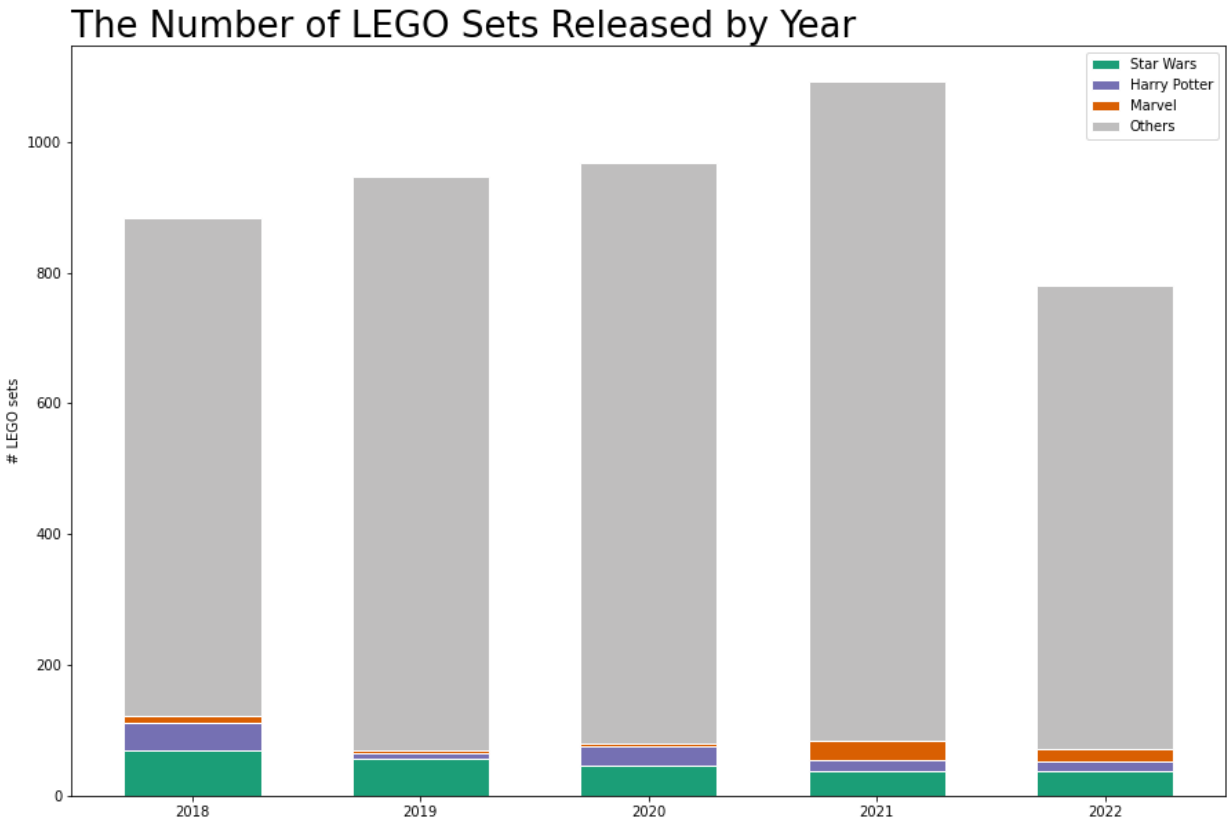
Out[155]:

	year	Harry Potter	Marvel	Others	Star Wars
6	2018	42.0	10.0	763.0	69.0
7	2019	9.0	3.0	878.0	57.0
8	2020	30.0	4.0	887.0	46.0
9	2021	18.0	29.0	1008.0	37.0
10	2022	15.0	18.0	710.0	38.0

Original Stacked Bar Plot

In [156…
```
labels = ['Star Wars', 'Harry Potter', 'Marvel','Others' ]
qcs =['#1b9e77','#7570b3','#d95f02','#BFBEBE']
width = 0.6

bottom = np.zeros((len(df_lego),), dtype=int)  # bottom accumulates starting points


fig, ax = plt.subplots(figsize=(15,10))

for pos, label in enumerate(labels):

    ax.bar(df_lego['year'], df_lego[label], width=width, label=label, color=qcs[pos],
    bottom = bottom +  df_lego[label]
```

```
ax.set_ylabel('# LEGO sets')
ax.legend()
ax.set_title('The Number of LEGO Sets Released by Year', fontsize=26, loc='left')

plt.show()
```



Add alternative text

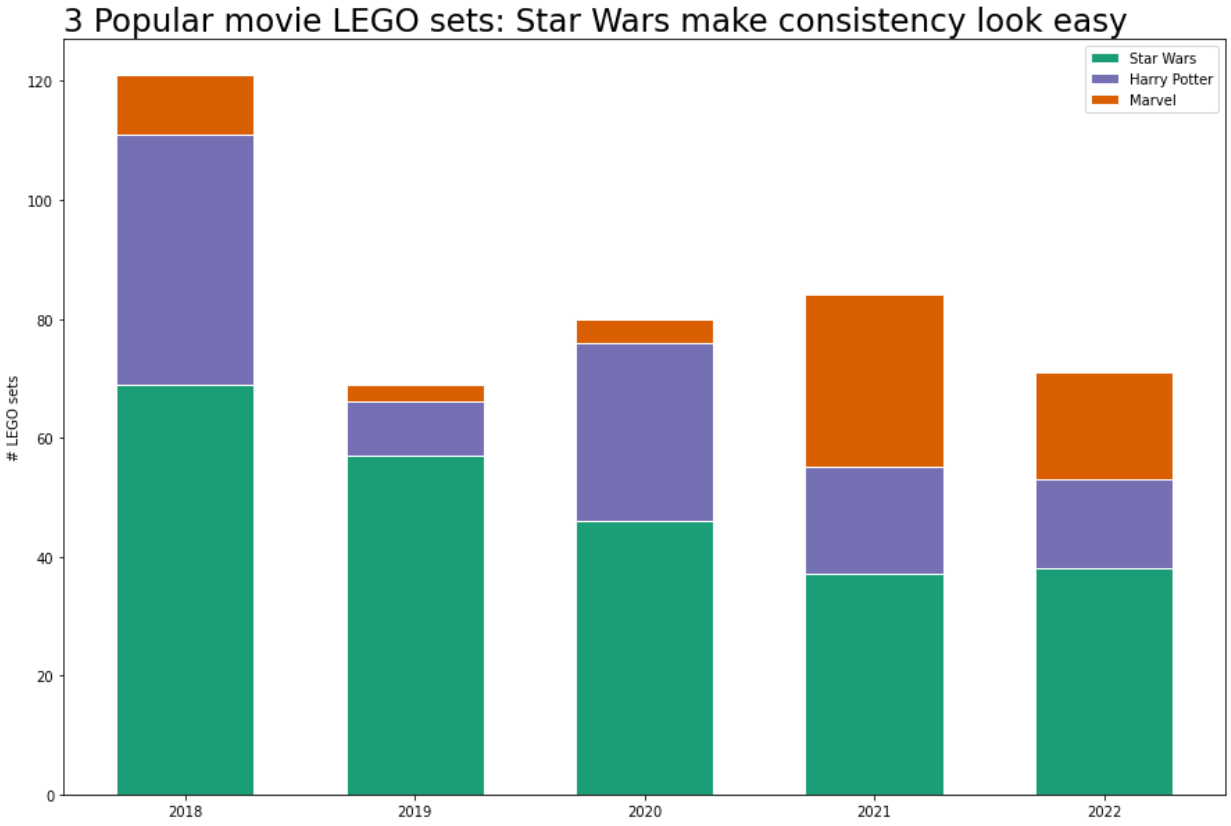
"A Horizontal stacked bar graph displaying the amount of LEGO sets produced in a given year. LEGO sets overall accumulation of releases from different movies ranging from Star Wars, Harry Potter, Marvel, and others each year starting from year 2018-2022. The "other" stands at majority of the graph outreleasing Star Wars, Harry Potter, and Marvel by more than 5 times even with them combined."

A change in this graph will be made however. The graphs has more of the "other" brand name/movie LEGO sets which makes the graph harder to read for the other brands. The change will consist of only mentioning the "other" as majority and then only show the other 3 lego sets which can be possible to use a horizontal stacked, making reading easier. Also the "other" category saturates the graph too much and will render other data points unknown and useless.

Employ a takeaway title

Here, I will display the change of the 4 categories to only 3 categories (Marvel, Harry Potter, and Star Wars). The replacement of the title "The Number of LEGO Sets Released by Year" to "3 Popular movie LEGO sets: Star Wars make consistency look easy".

In [157…
```
labels = ['Star Wars', 'Harry Potter', 'Marvel' ]
qcs =['#1b9e77','#7570b3','#d95f02',]
width = 0.6

bottom = np.zeros((len(df_lego),), dtype=int)  # bottom accumulates starting points


fig, ax = plt.subplots(figsize=(15,10))

for pos, label in enumerate(labels):

    ax.bar(df_lego['year'], df_lego[label], width=width, label=label, color=qcs[pos],
    bottom = bottom +  df_lego[label]

ax.set_ylabel('# LEGO sets', )
ax.legend()
ax.set_title('3 Popular movie LEGO sets: Star Wars make consistency look easy', fontsi

plt.show()
```

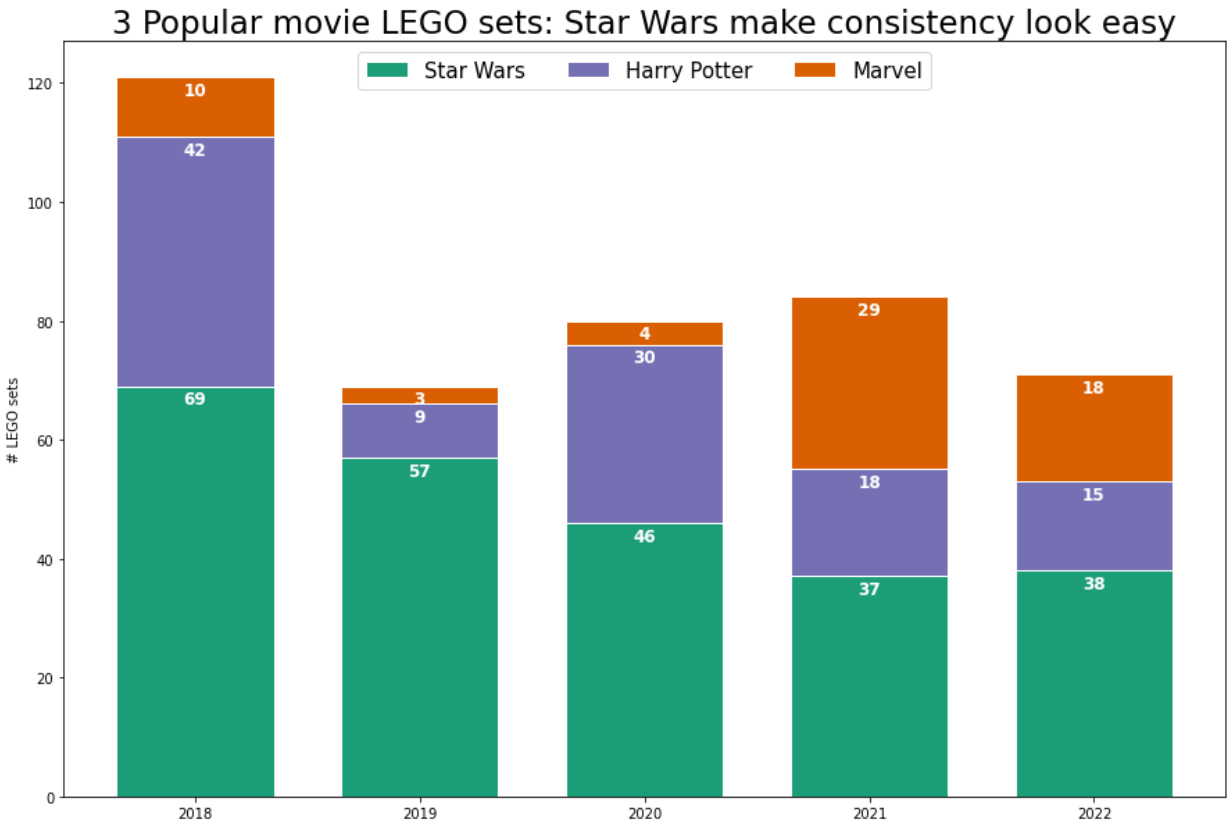


Label Data Directly:

Currently there is a legend at the corner of the graph that displays the colors and the LEGO sets with their affiliated movie series. I will be relocating the legend as it is more work to track the colors onto the graph in small text and its positioned far from the graph. Also having to filter out the "other" category will make the graph much easier to view. There is some number labels embedded in the graph indicating the actual quantity of the LEGO sets from each movie.

In [158…
```
labels = ['Star Wars', 'Harry Potter', 'Marvel' ]
```

```
qcs =['#1b9e77','#7570b3','#d95f02',]
width = 0.7

bottom = np.zeros((len(df_lego),), dtype=int)  # bottom accumulates starting points


fig, ax = plt.subplots(figsize=(15,10))

for pos, label in enumerate(labels):

    ax.bar(df_lego['year'], df_lego[label], width=width, label=label, color=qcs[pos],
    bottom = bottom +  df_lego[label]

ax.set_ylabel('# LEGO sets')

x_pos = np.arange(len(df_lego))
ax.legend(loc='upper center', ncol = 3, fontsize = 15)

#reference to how to put labels or numbers in a graph
#https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.text.html
#https://www.pythoncharts.com/matplotlib/stacked-bar-charts-labels/

y_offset = -3

for bar in ax.patches:
  ax.text(
      # Put the text in the middle of each bar. get_x returns the start
      # so we add half the width to get to the middle.
      bar.get_x() + bar.get_width() / 2,
      # Vertically, add the height of the bar to the start of the bar,
      # along with the offset.
      bar.get_height() + bar.get_y() + y_offset,
      # This is actual value we'll show.
      round(bar.get_height()),
      # Center the labels and style them a bit.
      ha='center',
      color='w',
      weight='bold',
      size=12
  )

ax.set_title('3 Popular movie LEGO sets: Star Wars make consistency look easy', fontsi

plt.show()
```



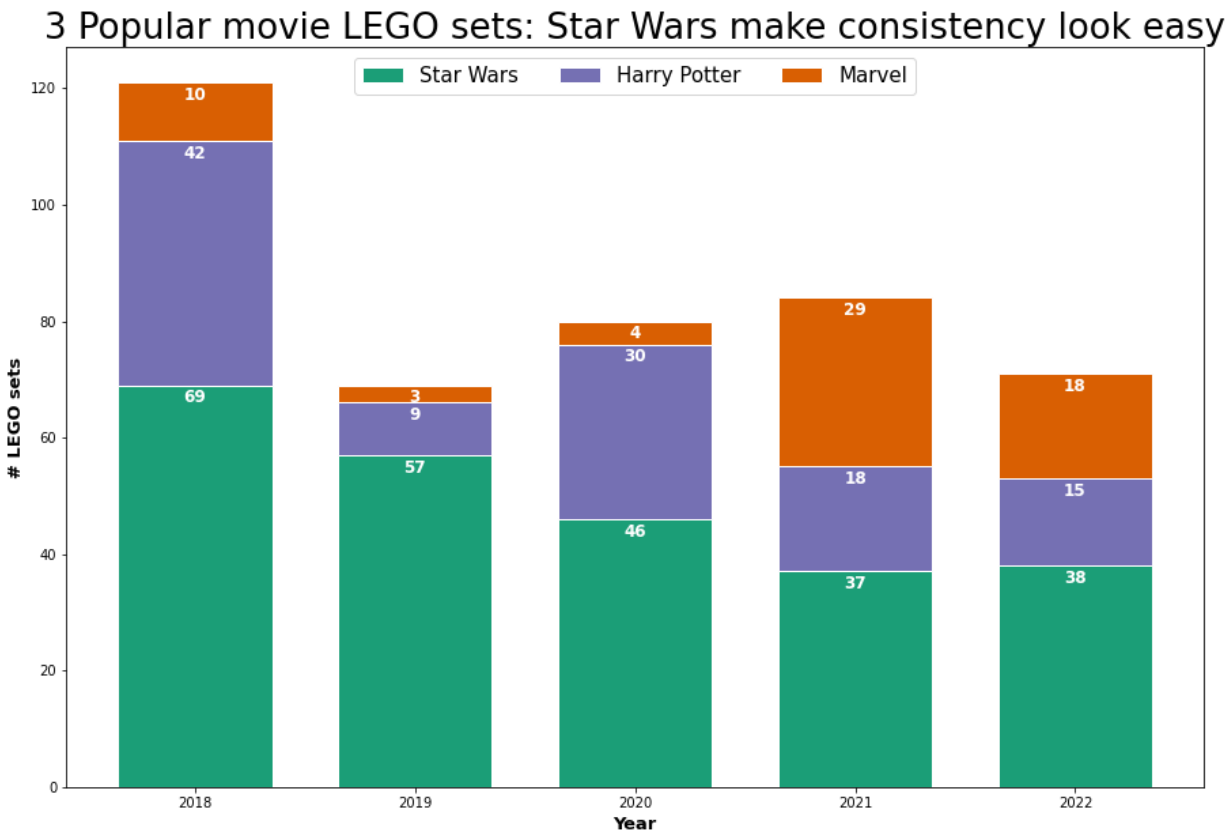
Use sufficient contrast

Contrast is important as it will make it easier to read the data graph at hand and for some people that are colorblind. Some colors that are not relative to each other will be beneficial from one LEGO set to another. Color should express polar opposite of its color like orange to blue to brown.

In [159…

```
bottom = np.zeros((len(df_lego),), dtype=int)  # bottom accumulates starting points


fig, ax = plt.subplots(figsize=(15,10))

for pos, label in enumerate(labels):

    ax.bar(df_lego['year'], df_lego[label], width=width, label=label, color=qcs[pos],
    bottom = bottom +  df_lego[label]

ax.set_ylabel('# LEGO sets', fontsize = 13, weight='bold')
ax.set_xlabel('Year', fontsize = 13, weight='bold')

x_pos = np.arange(len(df_lego))
ax.legend(loc='upper center', ncol = 3, fontsize = 15)

#reference to how to put labels or numbers in a graph
#https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.text.html
#https://www.pythoncharts.com/matplotlib/stacked-bar-charts-labels/

y_offset = -3

for bar in ax.patches:
```

```
  ax.text(
      # have the text in the middle of each bar. get_x returns the start
      # so we add half the width to get to the middle.
      bar.get_x() + bar.get_width() / 2,
      # vertically, add height of the bar to the start of the bar,
      # along with the offset.
      bar.get_height() + bar.get_y() + y_offset,
      # This is actual value we'll show.
      round(bar.get_height()),
      # Center the labels and style them.
      ha='center',
      va ='baseline',
      color='w',
      weight='bold',
      size=12
  )

ax.set_title('3 Popular movie LEGO sets: Star Wars make consistency look easy', fontsi

plt.show()
```



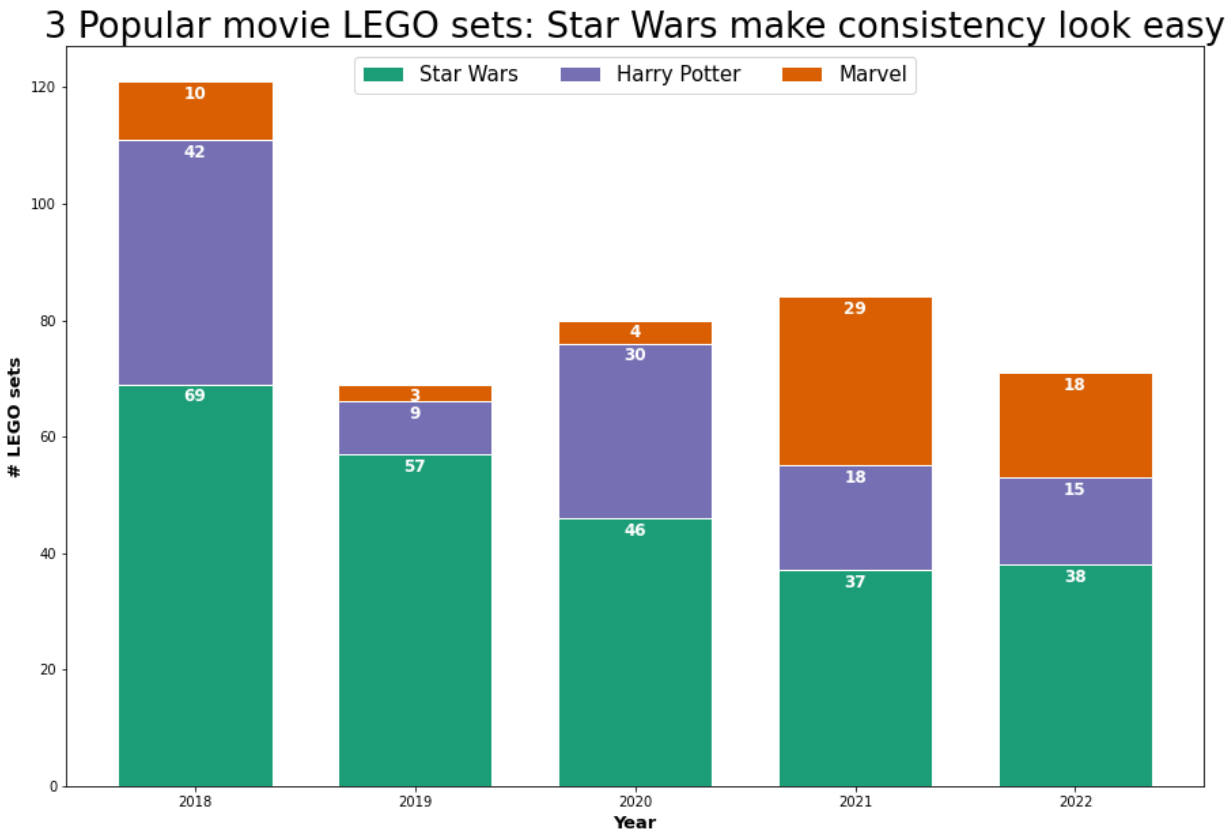
Leveraging white space

The whitespace from the previous step or the original already does a good job at separating the bars apart. There is no need to change anything for the white space.

Final Improved for Accessibility Plot

In [160…

```
bottom = np.zeros((len(df_lego),), dtype=int)  # bottom accumulates starting points


fig, ax = plt.subplots(figsize=(15,10))

for pos, label in enumerate(labels):

    ax.bar(df_lego['year'], df_lego[label], width=width, label=label, color=qcs[pos],
    bottom = bottom +  df_lego[label]

ax.set_ylabel('# LEGO sets', fontsize = 13, weight='bold')
ax.set_xlabel('Year', fontsize = 13, weight='bold')

x_pos = np.arange(len(df_lego))
ax.legend(loc='upper center', ncol = 3, fontsize = 15)

#reference to how to put labels or numbers in a graph
#https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.text.html
#https://www.pythoncharts.com/matplotlib/stacked-bar-charts-labels/
y_offset = -3

for bar in ax.patches:
  ax.text(
      # have the text in the middle of each bar. get_x returns the start
      # so we add half the width to get to the middle.
      bar.get_x() + bar.get_width() / 2,
      # vertically, add height of the bar to the start of the bar,
      # along with the offset.
      bar.get_height() + bar.get_y() + y_offset,
      # This is actual value we'll show.
      round(bar.get_height()),
      # Center the labels and style them.
      ha='center',
      va ='baseline',
      color='w',
      weight='bold',
      size=12
  )

ax.set_title('3 Popular movie LEGO sets: Star Wars make consistency look easy', fontsi

plt.show()
```



The plot shown is the finished product. We see the title is engaging and has a very forward and progressive statement that engages the audience. There is a change in labeling where I change the legend format and put it on top and made the font bigger so that it is easier to read. The color palette has contrast and reflects different colors that are truly different from each other.

In [ ]: