Project1

October 8, 2024

1 ITCS 3162 Data Mining Project 1:

2 Music's Impact on Mental Health

2.0.1 Shan Raheim

Kaggle Link for the Dataset:

https://www.kaggle.com/datasets/catherinerasgaitis/mxmh-survey-results

2.0.2 Problem Introduction

With the growing issues of mental health in our society today, solutions and methods to handle and manage the symptoms experienced by millions are being discussed. Not everyone has access to healthcare and professional help, so people are left to deal with their issues on their own or seek help that is accessible to them, which may include simple things they can incorporate into their day-to-day lives. Hobbies, friends, and family are some of the most common ways people can find support for mental health issues, especially since professional help may not be an immediate option available to everyone. Music is something that most people can access and can be paired with many activities. Music has the ability to directly impact how a person behaves or feels. It can be a hobby or something played in the background, but it can affect someone on a deeper level. Since this impact can be essential to how people handle issues, I am curious to find out if music has a meaningful effect on people with mental illnesses, as it is something easy for people to access from almost anywhere at any time.

2.0.3 Data Introduction

The dataset I used for this project is called "Music & Mental Health Survey Results." This dataset contains 737 rows and 33 columns. It includes responses from hundreds of respondents who have mental illnesses and experience symptoms such as Depression, Anxiety, Insomnia, and OCD, gathered through a Google form. Some examples of columns in this dataset include the frequencies of genres listened to by the respondents, whether they believe music helps with their mental illnesses, the amount of time they listen to music, and whether they listen to music while studying or working. For the different illnesses, the respective columns are measured using self-reported data on a scale from 1 to 10.

2.0.4 Data Pre-processing

For my data pre-processing I actually did not have to do much. When I initially looked through the dataset it was clean. It did not have irrelevant data, duplicates, or missing values.

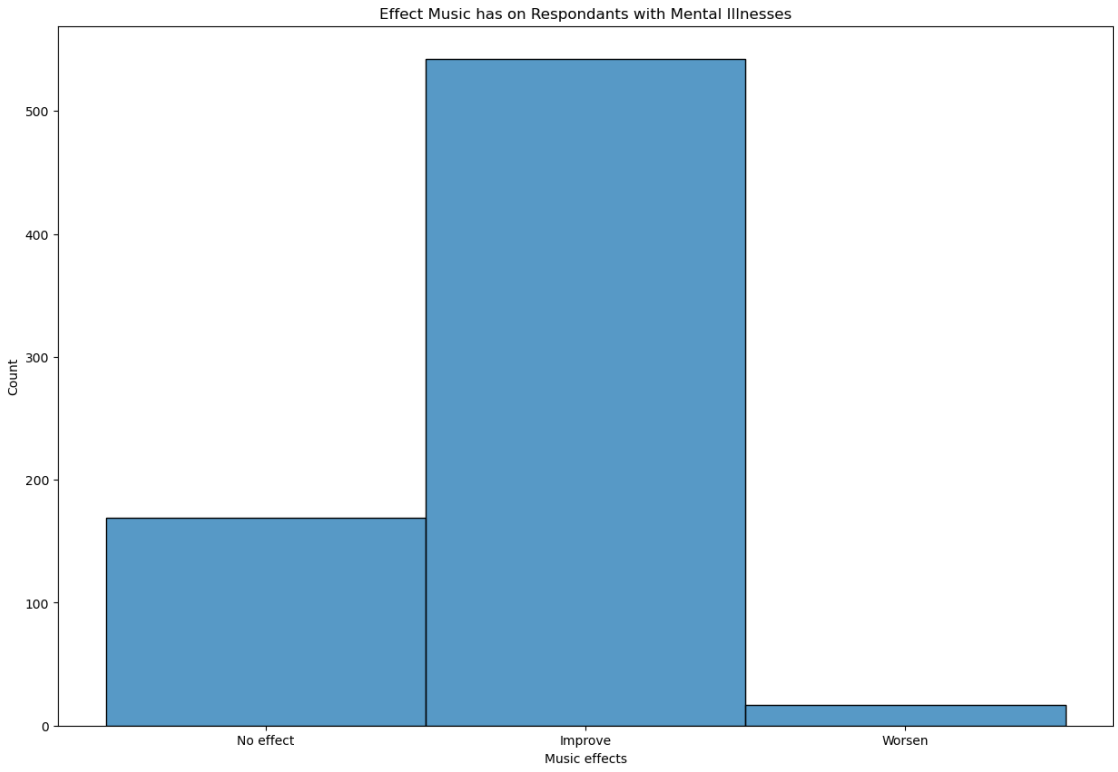
[10]:
```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
```

[12]:
```
data_filepath = "../ITSC3162/mxmh_survey_results.csv"
data = pd.read_csv(data_filepath)
```

2.0.5 Visualization 1: Bar Chart of the Mental Illnesses Affected by Music

[57]:
```
plt.figure(figsize=(15,10))
plt.title('Effect Music has on Respondants with Mental Illnesses')
sb.histplot(data['Music effects'])
```

[57]:
```
<Axes: title={'center': 'Effect Music has on Respondants with Mental
Illnesses'}, xlabel='Music effects', ylabel='Count'>
```



According to the histogram above from the survey results majority of the respondants say that music helps improve their mental condition. A small portion says it has no effect and even smaller amounts show that it actually worsens their mental state but compared the amount that says it

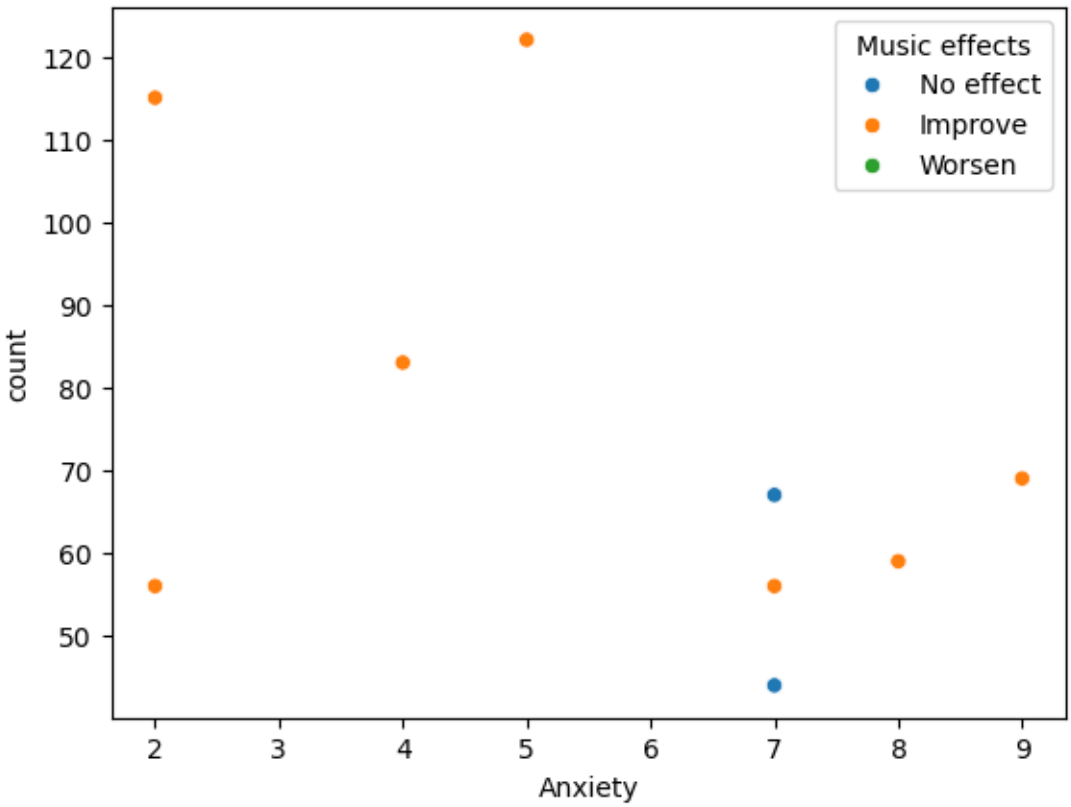
improves it, the difference is substancial. It shows music can help benefit their mental states with symptons from mental illnesses.

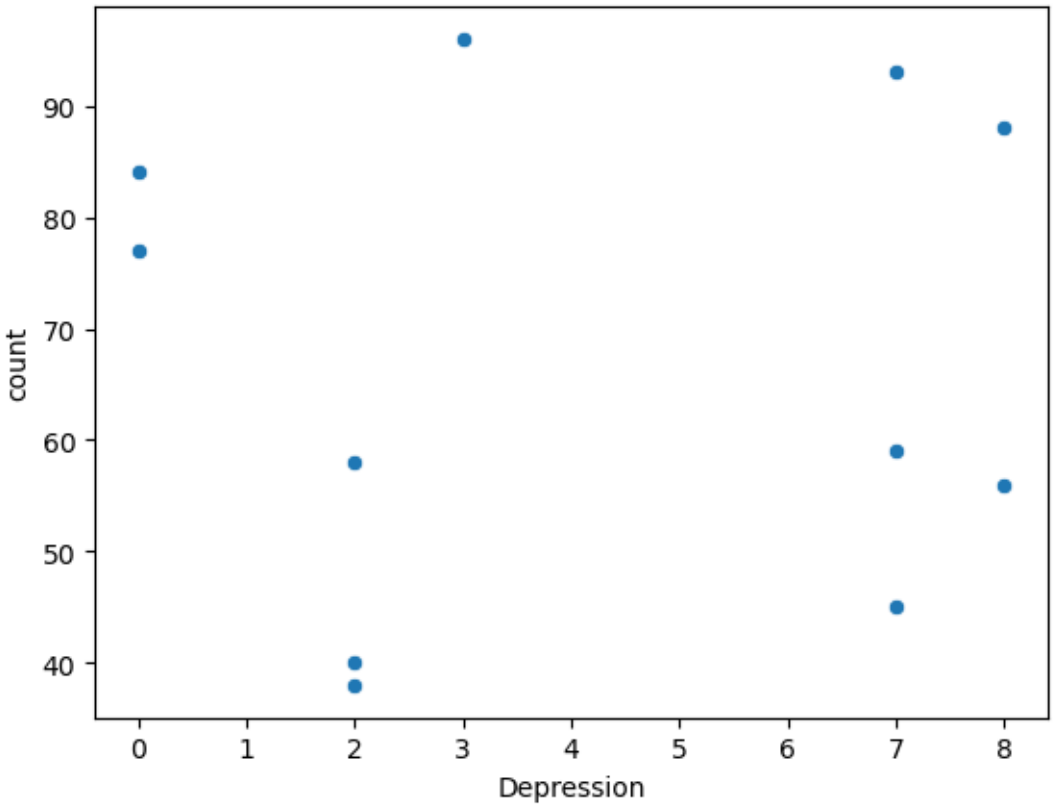
2.0.6 Most Common Mental Illnesses and Relation to the Data

According to the CDC Anxiety disorders and depression are the top two most common mental illnesses affecting Americans followed by others like PTSD. The statements, "In 2019, 301 million people were living with an anxiety disorder," and "In 2019, 280 million people were living with depression," demonstrates the recorded number of people who exeperienced mental illnesses related to depression and anxiety.

[133]:
```
anxiety_count = data['Anxiety'].value_counts()
depression_count= data['Depression'].value_counts()

plt.figure()
sb.scatterplot(x = data['Anxiety'], y = anxiety_count, hue = data['Music␣
↪effects'])

plt.figure()
sb.scatterplot(x = data['Depression'], y = depression_count)
```

[133]:
```
<Axes: xlabel='Depression', ylabel='count'>
```





In relation to the CDC article, bove are scatterplots for the self reported cases from the survey for Anxiety and Depression respectively rated on a scale from 0 - 10. Highlighted points show results from people who claimed music helps with their Anxiety or Depression, majority of the dots being orange which represents music improved their symptons of their respective mental illnesses. This shows a relation between the self reported cases of Anxiety and Depression along with music helping in accordance to the CDC article.

2.0.7 Impact

Overall the impact of this research and information can show how simple and close help for people suffering from mental illnesses. Though it may be limited it provide a source of assistance for people to dive into, potentially finding a new hobby or way to distract themselves from their respective illness symptons. A harm that I see from my visualizations is that some are maybe too "scarce". Part of the reason this is so is that the data is straightforward in a sense that it answers the general question of, if music helps with mental illnesses. A missing perspective that I think is missing from this data that could be included is asking if the respondants already recieve professional help or treatment for their illnesses along with using music to alleviate some of their symptons and helping them cope with the issues they face. This aspect could be included to compare whether or not if music combined with professional help actually helps or if one or the other could be doing more than the other.

2.0.8 Sources

https://www.kaggle.com/datasets/catherinerasgaitis/mxmh-survey-results

World Health Organization. (2022, June 8). Mental disorders. World Health Organization. https://www.who.int/news-room/fact-sheets/detail/mental-disorders

[22]:
```
data.head()
```

[22]:

	Timestamp	Age	Primary streaming service	Hours per day	While working	Instrumentalist	Composer	Fav genre	Exploratory	Foreign languages	…	Frequency [R&B]	Frequency [Rap]	Frequency [Rock]	Frequency [Video game music]	Anxiety	Depression	Insomnia	OCD	Music effects	Permissions
0	8/27/2022 19:29:02	18.0	Spotify	3.0	Yes	Yes	Yes	Latin	Yes	Yes	…	Sometimes	Very frequently	Never	Sometimes	3.0	0.0	1.0	0.0	NaN	I understand.
1	8/27/2022 19:57:31	63.0	Pandora	1.5	Yes	No	No	Rock	Yes	No	…	Sometimes	Rarely	Very frequently	Rarely	7.0	2.0	2.0	1.0	NaN	I understand.
2	8/27/2022 21:28:18	18.0	Spotify	4.0	No	No	No	Video game music	No	Yes	…	Never	Rarely	Rarely	Very frequently	7.0	7.0	10.0	2.0	No effect	I understand.
3	8/27/2022 21:40:40	61.0	YouTube Music	2.5	Yes	No	Yes	Jazz	Yes	Yes	…	Sometimes	Never	Never	Never	9.0	7.0	3.0	3.0	Improve	I understand.
4	8/27/2022 21:54:47	18.0	Spotify	4.0	Yes	No	No	R&B	Yes	No	…	Very frequently	Very frequently	Never	Rarely	7.0	2.0	5.0	9.0	Improve	I understand.

[5 rows x 33 columns]

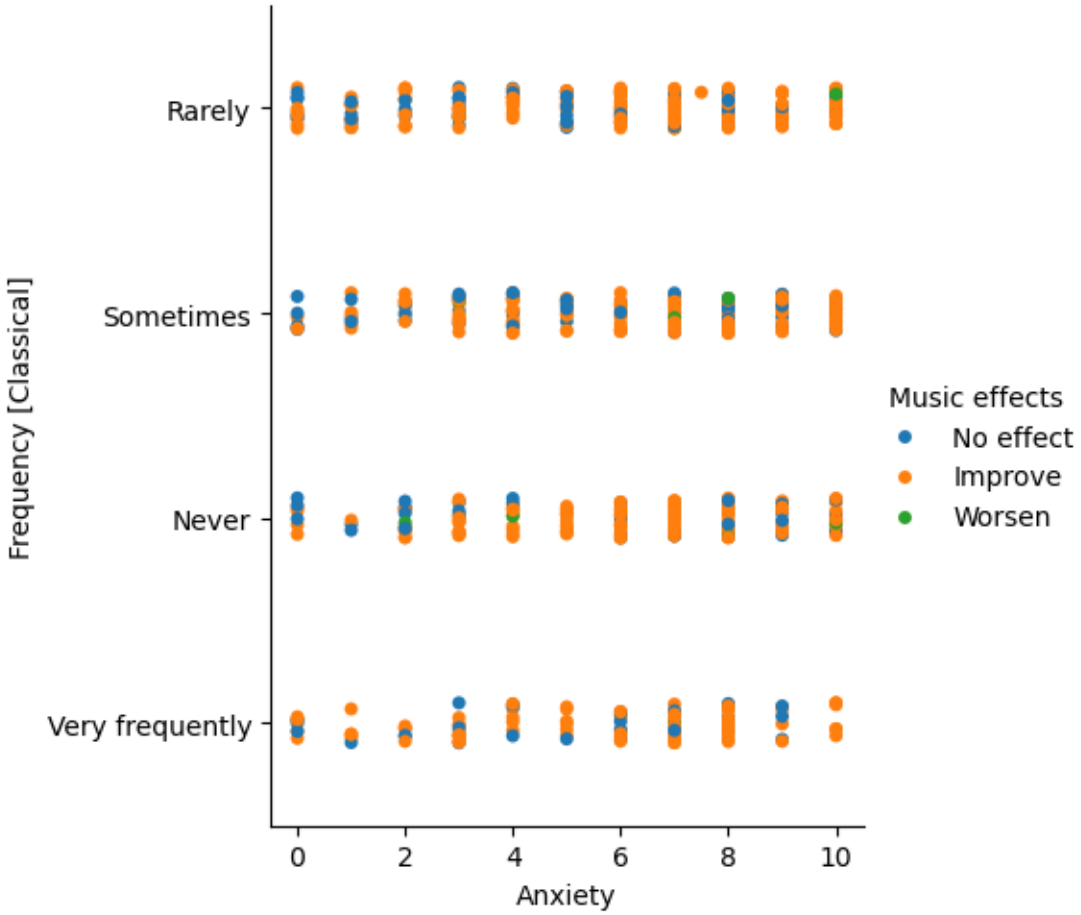
[24]:
```
data.columns
```

[24]:
```
Index(['Timestamp', 'Age', 'Primary streaming service', 'Hours per day',
       'While working', 'Instrumentalist', 'Composer', 'Fav genre',
       'Exploratory', 'Foreign languages', 'BPM', 'Frequency [Classical]',
       'Frequency [Country]', 'Frequency [EDM]', 'Frequency [Folk]',
       'Frequency [Gospel]', 'Frequency [Hip hop]', 'Frequency [Jazz]',
       'Frequency [K pop]', 'Frequency [Latin]', 'Frequency [Lofi]',
       'Frequency [Metal]', 'Frequency [Pop]', 'Frequency [R&B]',
       'Frequency [Rap]', 'Frequency [Rock]', 'Frequency [Video game music]',
       'Anxiety', 'Depression', 'Insomnia', 'OCD', 'Music effects',
       'Permissions'],
      dtype='object')
```

[28]:
```
import seaborn as sns
```

[32]:
```
sns.catplot(data=data, x="Anxiety", y="Frequency [Classical]", hue="Music␣
↪effects")
```

[32]:
```
<seaborn.axisgrid.FacetGrid at 0x27f7b505880>
```

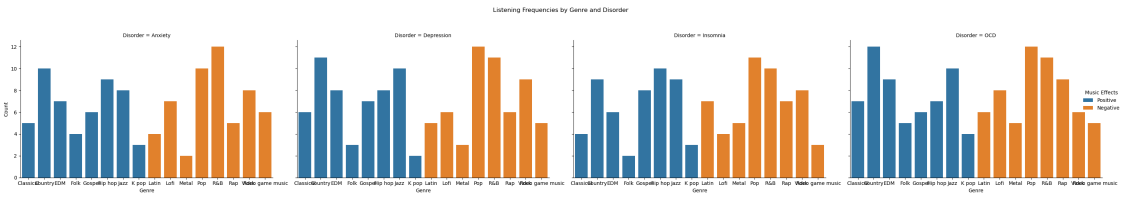


[ ]:
```
sns.barplot(data, x="Frequency [Classical]", y="body_mass_g", hue="sex")
```

y - counts of some sort
x - genres
hue - music effects

[43]:
```
# Sample genres array
genres = ['Classical', 'Country', 'EDM', 'Folk', 'Gospel', 'Hip hop', 'Jazz',
          'K pop', 'Latin', 'Lofi', 'Metal', 'Pop', 'R&B', 'Rap', 'Rock',␣
 ↪'Video game music']

# Assuming you have value counts for each disorder, replace with actual data
anxiety_count = [5, 10, 7, 4, 6, 9, 8, 3, 4, 7, 2, 10, 12, 5, 8, 6]  # Sample␣
 ↪data
depression_count = [6, 11, 8, 3, 7, 8, 10, 2, 5, 6, 3, 12, 11, 6, 9, 5]  #␣
 ↪Sample data
insomnia_count = [4, 9, 6, 2, 8, 10, 9, 3, 7, 4, 5, 11, 10, 7, 8, 3]  # Sample␣
 ↪data
OCD_count = [7, 12, 9, 5, 6, 7, 10, 4, 6, 8, 5, 12, 11, 9, 6, 5]  # Sample data

# Create a list of Music Effects categories (sample)
music_effects = ['Positive', 'Negative', 'Neutral']

# Combine the data into a DataFrame
data = {
    'Genre': genres * 4,  # Repeat genres for each disorder
    'Disorder': ['Anxiety'] * 16 + ['Depression'] * 16 + ['Insomnia'] * 16 +␣
 ↪['OCD'] * 16,
    'Count': anxiety_count + depression_count + insomnia_count + OCD_count,
    'Music Effects': (['Positive'] * 8 + ['Negative'] * 8) * 4  # Example:␣
 ↪Varying music effects
}

df = pd.DataFrame(data)

# Plotting the chart
plt.figure(figsize=(20, 20))
sns.catplot(data=df, x="Genre", y="Count", hue="Music Effects", col="Disorder",␣
 ↪kind="bar", height=5, aspect=1.5)

# Add a title
plt.suptitle('Listening Frequencies by Genre and Disorder', y=1.05)
plt.tight_layout()

# Show the plot
plt.show()
```

```
<Figure size 2000x2000 with 0 Axes>
```



[45]:
```
print(df)
```

```
                Genre Disorder  Count Music Effects
0           Classical  Anxiety      5      Positive
1             Country  Anxiety     10      Positive
2                 EDM  Anxiety      7      Positive
3               Folk   Anxiety      4      Positive
4              Gospel  Anxiety      6      Positive
..                 …       …    …            …
59               Pop       OCD     12      Negative
60               R&B       OCD     11      Negative
61               Rap       OCD      9      Negative
62              Rock       OCD      6      Negative
63  Video game music       OCD      5      Negative

[64 rows x 4 columns]
```

How to determine count?
(example) if a row has anxiety above a rating of 5, then it is added to the counts.
if out of those 5 ratings, most are positive, then the overall music effect is positive. if most are negative, then it is negative.

[ ]:
```
data[data['Anxiety'] > 5 or data['Depression'] > 5 or ] # example of filtering
```

[20]:
```
genres = ['Frequency [Classical]', 'Frequency [Country]',
'Frequency [EDM]', 'Frequency [Folk]', 'Frequency [Gospel]',
'Frequency [Hip hop]', 'Frequency [Jazz]', 'Frequency [K pop]',
'Frequency [Latin]', 'Frequency [Lofi]', 'Frequency [Metal]',
'Frequency [Pop]', 'Frequency [R&B]', 'Frequency [Rap]',
'Frequency [Rock]', 'Frequency [Video game music]']

anxiety_count = data['Anxiety'].value_counts()
depression_count= data['Depression'].value_counts()
insomnia_count = data['Insomnia'].value_counts()
OCD_count = data['OCD'].value_counts()

disorder_counts = [anxiety_count, depression_count, insomnia_count, OCD_count]

sb.catplot(x = genres, y = disorder_counts, hue = data['Music effects'],
```

```
            kind = 'Genre', height = 10, aspect = 10);
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[20], line 15
     11 OCD_count = data['OCD'].value_counts()
     13 disorder_counts = [anxiety_count, depression_count, insomnia_count,␣
 ↪OCD_count]
---> 15 sb.catplot(x = genres, y = disorder_counts, hue = data['Music effects']
     16             kind = 'Genre', height = 10, aspect = 10)

File ~\anaconda3latest\Lib\site-packages\seaborn\categorical.py:2782, in␣
 ↪catplot(data, x, y, hue, row, col, kind, estimator, errorbar, n_boot, seed,␣
 ↪units, weights, order, hue_order, row_order, col_order, col_wrap, height,␣
 ↪aspect, log_scale, native_scale, formatter, orient, color, palette, hue_norm,
 ↪legend, legend_out, sharex, sharey, margin_titles, facet_kws, ci, **kwargs)
   2779     elif x is not None and y is not None:
   2780         raise ValueError("Cannot pass values for both `x` and `y`.")
-> 2782 p = Plotter(
   2783     data=data,
   2784     variables=dict(
   2785         x=x, y=y, hue=hue, row=row, col=col, units=units, weight=weight
   2786     ),
   2787     order=order,
   2788     orient=orient,
   2789     # Handle special backwards compatibility where pointplot originally
   2790     # did *not* default to multi-colored unless a palette was specified
   2791     color="C0" if kind == "point" and palette is None and color is None
 ↪else color,
   2792     legend=legend,
   2793 )
   2795 for var in ["row", "col"]:
   2796     # Handle faceting variables that lack name information
   2797     if var in p.variables and p.variables[var] is None:

File ~\anaconda3latest\Lib\site-packages\seaborn\categorical.py:67, in␣
 ↪_CategoricalPlotter.__init__(self, data, variables, order, orient,␣
 ↪require_numeric, color, legend)
     56 def __init__(
     57     self,
     58     data=None,
   (…)
     64     legend="auto",
     65 ):
---> 67     super().__init__(data=data, variables=variables)
     69     # This method takes care of some bookkeeping that is necessary␣
 ↪because the
```

```
    70     # original categorical plots (prior to the 2021 refactor) had some␣
 ↪rules that
    71     # don't fit exactly into VectorPlotter logic. It may be wise to hav␣
 ↪a second
  (…)
    76     # default VectorPlotter rules. If we do decide to make orient part␣
 ↪of the
    77     # _base variable assignment, we'll want to figure out how to expres␣
 ↪that.
    78     if self.input_format == "wide" and orient in ["h", "y"]:

File ~\anaconda3latest\Lib\site-packages\seaborn\_base.py:634, in VectorPlotter
 ↪__init__(self, data, variables)
   629 # var_ordered is relevant only for categorical axis variables, and may
   630 # be better handled by an internal axis information object that tracks
   631 # such information and is set up by the scale_* methods. The analogous
   632 # information for numeric axes would be information about log scales.
   633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
   636 # TODO Lots of tests assume that these are called to initialize the
   637 # mappings to default values on class initialization. I'd prefer to
   638 # move away from that and only have a mapping when explicitly called.
   639 for var in ["hue", "size", "style"]:

File ~\anaconda3latest\Lib\site-packages\seaborn\_base.py:679, in VectorPlotter
 ↪assign_variables(self, data, variables)
   674 else:
   675     # When dealing with long-form input, use the newer PlotData
   676     # object (internal but introduced for the objects interface)
   677     # to centralize / standardize data consumption logic.
   678     self.input_format = "long"
--> 679     plot_data = PlotData(data, variables)
   680     frame = plot_data.frame
   681     names = plot_data.names

File ~\anaconda3latest\Lib\site-packages\seaborn\_core\data.py:58, in PlotData.
 ↪__init__(self, data, variables)
    51 def __init__(
    52     self,
    53     data: DataSource,
    54     variables: dict[str, VariableSpec],
    55 ):
    57     data = handle_data_source(data)
---> 58     frame, names, ids = self._assign_variables(data, variables)
    60     self.frame = frame
    61     self.names = names
```

```
File ~\anaconda3latest\Lib\site-packages\seaborn\_core\data.py:265, in PlotData
 ↪_assign_variables(self, data, variables)
    260             ids[key] = id(val)
    262 # Construct a tidy plot DataFrame. This will convert a number of
    263 # types automatically, aligning on index in case of pandas objects
    264 # TODO Note: this fails when variable specs *only* have scalars!
--> 265 frame = pd.DataFrame(plot_data)
    267 return frame, names, ids

File ~\anaconda3latest\Lib\site-packages\pandas\core\frame.py:778, in DataFrame
 ↪__init__(self, data, index, columns, dtype, copy)
    772     mgr = self._init_mgr(
    773         data, axes={"index": index, "columns": columns}, dtype=dtype,␣
 ↪copy=copy
    774     )
    776 elif isinstance(data, dict):
    777     # GH#38939 de facto copy defaults to False only in non-dict cases
--> 778     mgr = dict_to_mgr(data, index, columns, dtype=dtype, copy=copy,␣
 ↪typ=manager)
    779 elif isinstance(data, ma.MaskedArray):
    780     from numpy.ma import mrecords

File ~\anaconda3latest\Lib\site-packages\pandas\core\internals\construction.py:
 ↪503, in dict_to_mgr(data, index, columns, dtype, typ, copy)
    499     else:
    500         # dtype check to exclude e.g. range objects, scalars
    501         arrays = [x.copy() if hasattr(x, "dtype") else x for x in array
--> 503 return arrays_to_mgr(arrays, columns, index, dtype=dtype, typ=typ,␣
 ↪consolidate=copy)

File ~\anaconda3latest\Lib\site-packages\pandas\core\internals\construction.py:
 ↪114, in arrays_to_mgr(arrays, columns, index, dtype, verify_integrity, typ,␣
 ↪consolidate)
    111 if verify_integrity:
    112     # figure out the index, if necessary
    113     if index is None:
--> 114         index = _extract_index(arrays)
    115     else:
    116         index = ensure_index(index)

File ~\anaconda3latest\Lib\site-packages\pandas\core\internals\construction.py:
 ↪677, in _extract_index(data)
    675 lengths = list(set(raw_lengths))
    676 if len(lengths) > 1:
--> 677     raise ValueError("All arrays must be of the same length")
    679 if have_dicts:
    680     raise ValueError(
    681         "Mixing dicts with non-Series may lead to ambiguous ordering."
```

```
    682     )

ValueError: All arrays must be of the same length
```

[ ]:

[ ]: