labassignment11bn

April 15, 2025

1 Lab Assignment 11: Data Visualizations

1.1 DS 6001: Practice and Application of Data Science

1.1.1 Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

1.2 Problem 0

Import the following libraries:

[192]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

1.3 Problem 1

Write a short paragraph that provides a critique of the following data visualizations. What's good about each figure, and what's not good? Pay particular attention to how well the figure communicates information to a general audience and tells a complete story. Make specific references to the ideas discussed in the first section of the Module 11 Jupyter notebook.

1.3.1 Part a

[1 point]

This visualization has a clear title, clearly labels each bar, and has the percentage of each category above each bar, which is nice. The bars are equal width and the categories are dispalyed in a logical order. But good Lord are there problems. There is no scale on the y axis so I am not sure what the scale is. There are horizontal bars across as reference which is nice, but they aren't labelled so we don't know what the reference lines are. The bars don't match their percentages! The 13% has the biggest bar, despite being the lowest percentage. The 28% has no bar at all, potentially insinuating that the y scale doesn't start at 0 since there is no 0%.

1.3.2 Part b

[1 point]

This is a pie chart with clearly labelled slices with percentages and a title in the upper right hand corner. However, this pie chart clearly violates the principle of proportional ink in mulitple ways. First, it is a 3D pie chart, which adds more area to certain areas slices but not others. Second, they have this spiraling type feature where the smallest percentage has a small slice, then each slices has a slightly bigger section (like a bigger radius). This makes the areas VERY misleading. Also, they repeated colors which makes it confusing.

1.3.3 Part c

[1 point]

I'm not sure where to start on this one. I guess the graph is clearly labelled and there's both an x and a y scale (though the x could be a little clearer). Without too much thought it's easy to deduce that each dot is a year (though the fact that I had to think about it is not great and it is hard to tell which year is which). They also labelled some helpful items like the first and last data point as well as when an important legislation was passed pertaining to the topic. While the graph is clearly labelled, it has two title? Or at least a title and a subtitle. For one graph? That seems like they could have done a better job. The thing that baffles me is why is 0 on the top for the y scale and then the graph goes down when the numbers are increasing? That's so confusing. Also, why is it red? I assume that it's red for blood since they're talking about deaths, but it does not add any clarity. I guess I give them creativity points becuase it kind of looks like blood dripping down a wall, like after a gunshot?

1.4 Problem 2

For the rest of this lab, we will once again be working with the 2019 General Social Survey.

[193]:
```
%%capture
gss = pd.read_csv("https://github.com/jkropko/DS-6001/raw/master/localdata/
 ↪gss2018.csv",
                  encoding='cp1252', na_values=['IAP','IAP,DK,NA,uncodeable',␣
 ↪'NOT SURE',
                                                'DK', 'IAP, DK, NA, uncodeable',␣
 ↪'.a', "CAN'T CHOOSE"])
```

Here is code that cleans the data and gets it ready to be used for data visualizations:

[194]:
```
mycols = ['id', 'wtss', 'sex', 'educ', 'region', 'age', 'coninc',
          'prestg10', 'mapres10', 'papres10', 'sei10', 'satjob',
          'fechld', 'fefam', 'fepol', 'fepresch', 'meovrwrk']
gss_clean = gss[mycols]
gss_clean = gss_clean.rename({'wtss':'weight',
                              'educ':'education',
                              'coninc':'income',
                              'prestg10':'job_prestige',
                              'mapres10':'mother_job_prestige',
```

```
                                'papres10':'father_job_prestige',
                                'sei10':'socioeconomic_index',
                                'fechld':'relationship',
                                'fefam':'male_breadwinner',
                                'fehire':'hire_women',
                                'fejobaff':'preference_hire_women',
                                'fepol':'men_bettersuited',
                                'fepresch':'child_suffer',
                                'meovrwrk':'men_overwork'},axis=1)
gss_clean.age = gss_clean.age.replace({'89 or older':'89'})
gss_clean.age = gss_clean.age.astype('float')
```

The `gss_clean` dataframe now contains the following features:

- `id` - a numeric unique ID for each person who responded to the survey
- `weight` - survey sample weights
- `sex` - male or female
- `education` - years of formal education
- `region` - region of the country where the respondent lives
- `age` - age
- `income` - the respondent's personal annual income
- `job_prestige` - the respondent's occupational prestige score, as measured by the GSS using the methodology described above
- `mother_job_prestige` - the respondent's mother's occupational prestige score, as measured by the GSS using the methodology described above
- `father_job_prestige` -the respondent's father's occupational prestige score, as measured by the GSS using the methodology described above
- `socioeconomic_index` - an index measuring the respondent's socioeconomic status
- `satjob` - responses to "On the whole, how satisfied are you with the work you do?"
- `relationship` - agree or disagree with: "A working mother can establish just as warm and secure a relationship with her children as a mother who does not work."
- `male_breadwinner` - agree or disagree with: "It is much better for everyone involved if the man is the achiever outside the home and the woman takes care of the home and family."
- `men_bettersuited` - agree or disagree with: "Most men are better suited emotionally for politics than are most women."
- `child_suffer` - agree or disagree with: "A preschool child is likely to suffer if his or her mother works."
- `men_overwork` - agree or disagree with: "Family life often suffers because men concentrate too much on their work."

1.4.1 Part a

Reorder the categories of `relationship` to "strongly agree", "agree", "disagree", and "strongly disagree".

Then create a simple barplot that shows the frequencies of the categories of `relationship` three times: * once using `matplotlib` alone, * once using `seaborn`, * and once using the `.plot()` method from `pandas`.

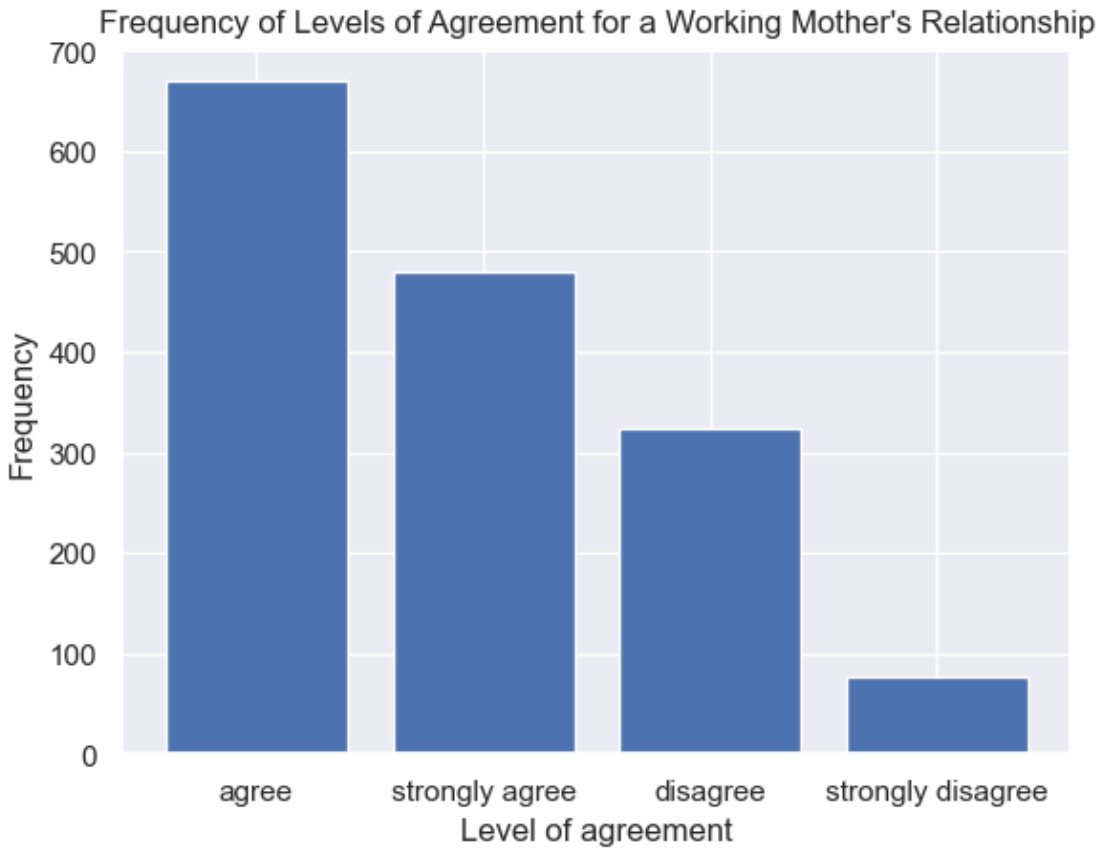
Make sure each barplot has descriptive axis labels and a title, and set a good size for each figure

displayed in the Jupyter notebook. [2 points]

[195]:
```
gss_clean['relationship'] = gss_clean['relationship'].astype('category')
gss_clean['relationship'] = gss_clean['relationship'].cat.
  ↪reorder_categories(new_categories=["strongly agree", "agree", "disagree",␣
  ↪"strongly disagree"], ordered=True)
```

[196]:
```
plt.bar(gss_clean['relationship'].value_counts().index,␣
  ↪gss_clean['relationship'].value_counts().values)
plt.xlabel('Level of agreement')
plt.ylabel('Frequency')
plt.title('Frequency of Levels of Agreement for a Working Mother\'s␣
  ↪Relationship')
```

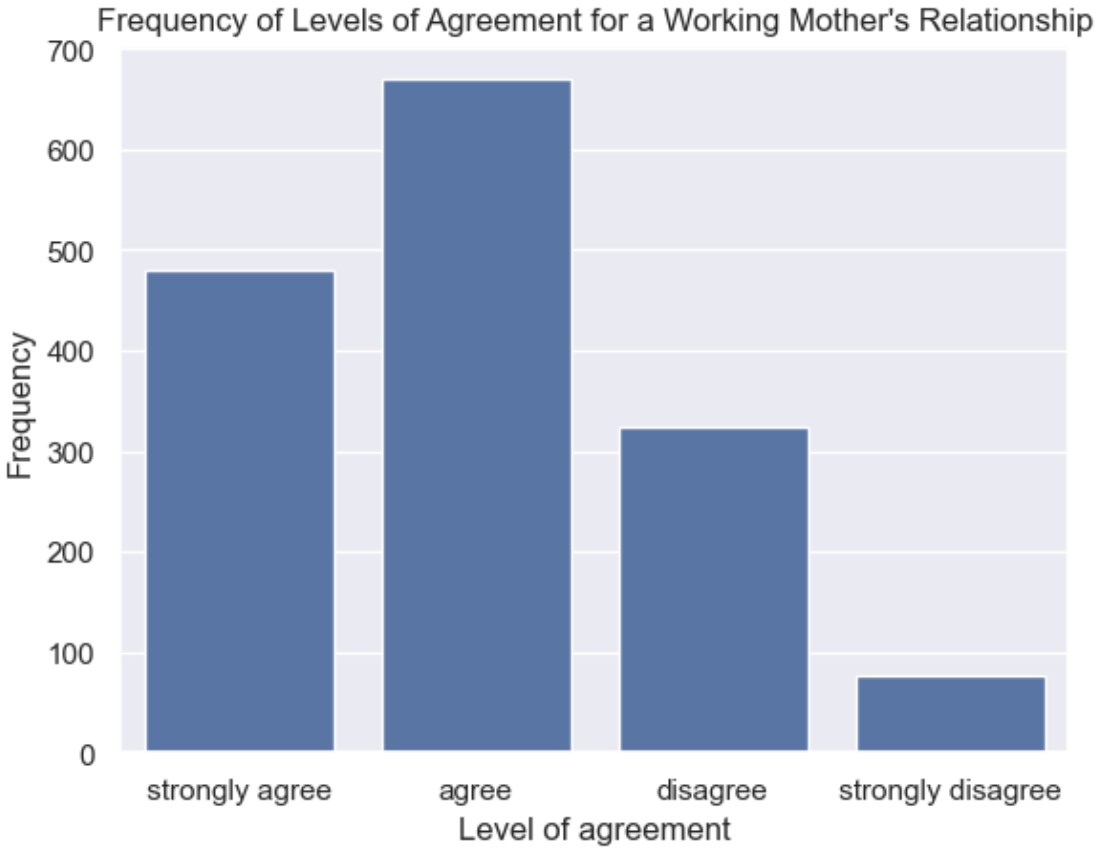
[196]:
```
Text(0.5, 1.0, "Frequency of Levels of Agreement for a Working Mother's
Relationship")
```



[197]:
```
sns.barplot(x=gss_clean['relationship'].value_counts().index,␣
  ↪y=gss_clean['relationship'].value_counts().values)
plt.xlabel('Level of agreement')
```

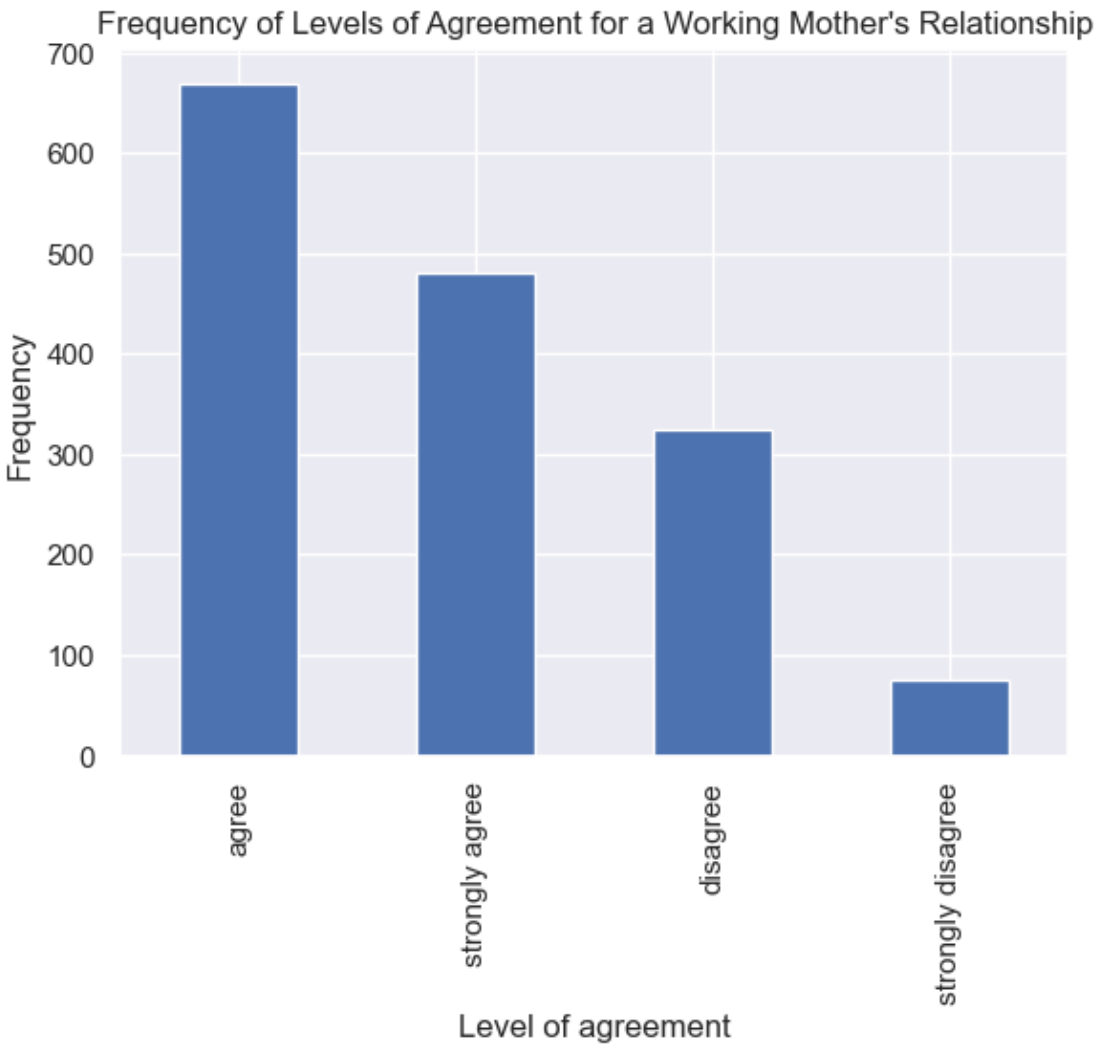
```
plt.ylabel('Frequency')
plt.title('Frequency of Levels of Agreement for a Working Mother\'s␣
 ↪Relationship')
```

[197]:
```
Text(0.5, 1.0, "Frequency of Levels of Agreement for a Working Mother's
Relationship")
```



[198]:
```
gss_clean['relationship'].value_counts().plot(kind='bar')
plt.xlabel('Level of agreement')
plt.ylabel('Frequency')
plt.title('Frequency of Levels of Agreement for a Working Mother\'s␣
 ↪Relationship')
```

[198]:
```
Text(0.5, 1.0, "Frequency of Levels of Agreement for a Working Mother's
Relationship")
```



I know I didn't change the figure size on each graph, but they show up plenty big on my screen.

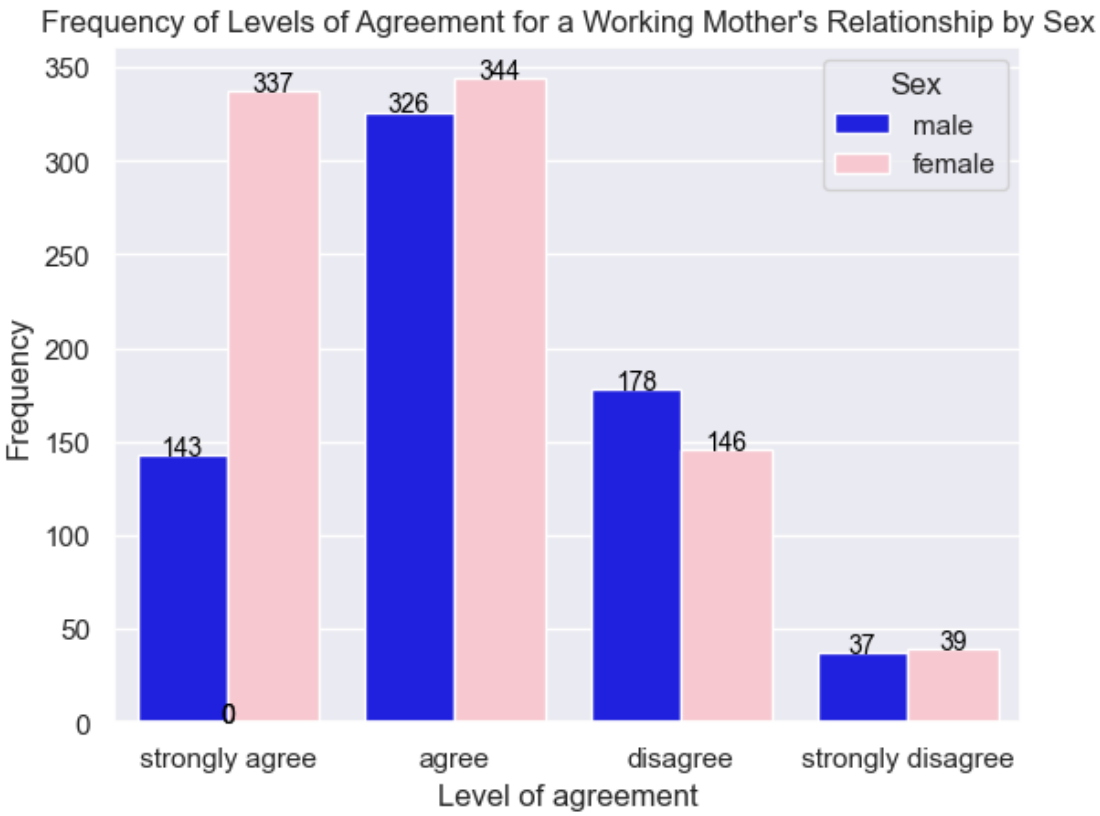
1.4.2 Part b

Create two barplots that show * the frequency of the different levels of agreement for `relationship` for men and for women on the same plot, * with bars for men and bars for women side-by-side, * using different colors for the bars for men and the bars for women, * listing these colors and the sex they refer to in a legend, * and labeling each bar with the number the bar represents.

Create the first barplot using `seaborn` with the bars oriented vertically, and create the second barplot using the `.plot()` method with the bars oriented horizontally. [2 points]

[199]:
```
sns.countplot(x='relationship', hue='sex', data=gss_clean, palette={'male':␣
↪'blue', 'female': 'pink'})
plt.xlabel('Level of agreement')
```

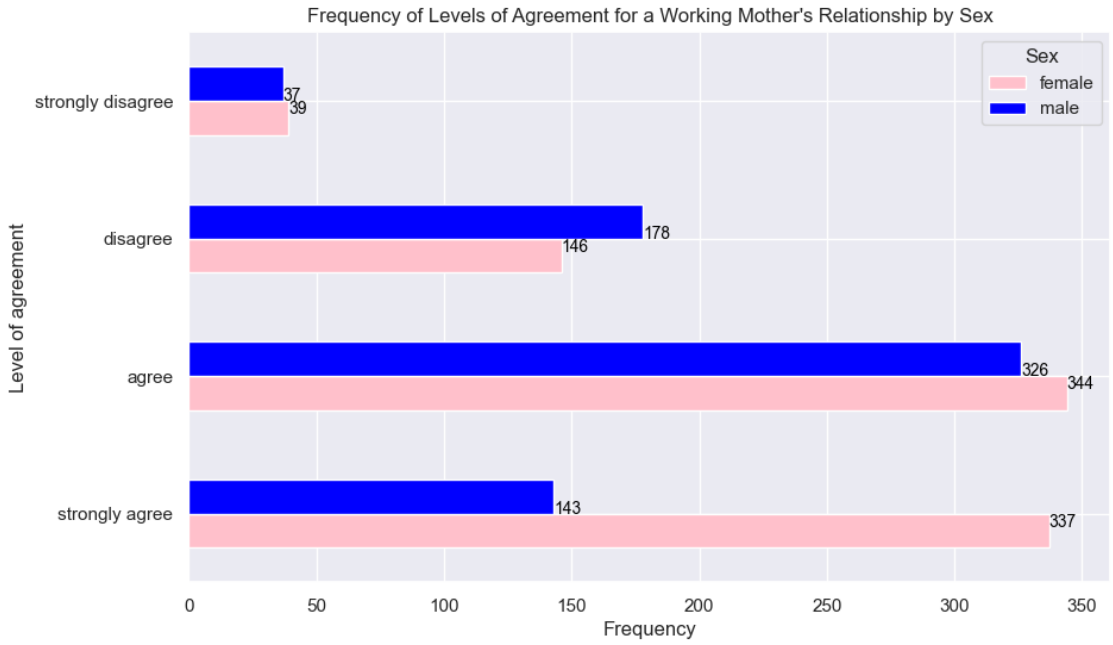
```
plt.ylabel('Frequency')
plt.title('Frequency of Levels of Agreement for a Working Mother\'s␣
 ↪Relationship by Sex')
plt.legend(title='Sex')
for p in plt.gca().patches:
    plt.gca().annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2.
 ↪, p.get_height()), ha='center', va='baseline', fontsize=10, color='black')
```



[200]:
```
gss_clean['count'] = 1
pivot_df = gss_clean.pivot_table(index='relationship', columns='sex',␣
 ↪values='count', aggfunc='sum')
pivot_df.plot(kind='barh', stacked=False, color={'male': 'blue', 'female':␣
 ↪'pink'}, figsize=(10, 6))
plt.xlabel('Frequency')
plt.ylabel('Level of agreement')
plt.title('Frequency of Levels of Agreement for a Working Mother\'s␣
 ↪Relationship by Sex')
plt.legend(title='Sex')
for index, value in enumerate(pivot_df.values):
```

```
    plt.gca().text(value[0], index, str(int(value[0])), color='black',␣
↪va='top', ha='left', fontsize=10)
    plt.gca().text(value[1], index, str(int(value[1])), color='black',␣
↪va='baseline', ha='left', fontsize=10)
```

```
C:\Users\brian\AppData\Local\Temp\ipykernel_34860\2412445982.py:2:
FutureWarning: The default value of observed=False is deprecated and will change
to observed=True in a future version of pandas. Specify observed=False to
silence this warning and retain the current behavior
  pivot_df = gss_clean.pivot_table(index='relationship', columns='sex',
values='count', aggfunc='sum')
```



1.4.3 Part c

Create a visualization with * nine barplots, arranged in a 3x3 grid. * The barplots should refer to each of the nine categories of `region`, * and each barplot should be given a label that contains the name of the region. * Within each barplot, list the categories of `relationship`, * and display horizontal bars.

Only one figure is required. Use `seaborn`, `matplotlib`, and `.plot()` as you see fit. [2 points]

[201]:
```
gss_facet = gss_clean.groupby(['relationship', 'region']).agg({'count': 'sum'}).
↪reset_index()

c = sns.FacetGrid(gss_facet, col='region', hue='relationship', col_wrap=3)
c.map(sns.barplot, 'count', 'relationship', palette={'strongly agree': 'green',␣
↪'agree': '#c6f4d6', 'disagree': '#ffc5c5', 'strongly disagree': 'red'})
```

```
c.set_titles('{col_name}')
c.set_axis_labels('count')
c.fig.subplots_adjust(top=.9)
c.fig.suptitle('Distribution of Relationship Levels by Region', fontsize=16)
```

```
C:\Users\brian\AppData\Local\Temp\ipykernel_34860\1760358911.py:1:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  gss_facet = gss_clean.groupby(['relationship', 'region']).agg({'count':
'sum'}).reset_index()
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:718: UserWarning: Using the barplot function
without specifying `order` is likely to produce an incorrect plot.
  warnings.warn(warning)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
```

```
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
```

```
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
```

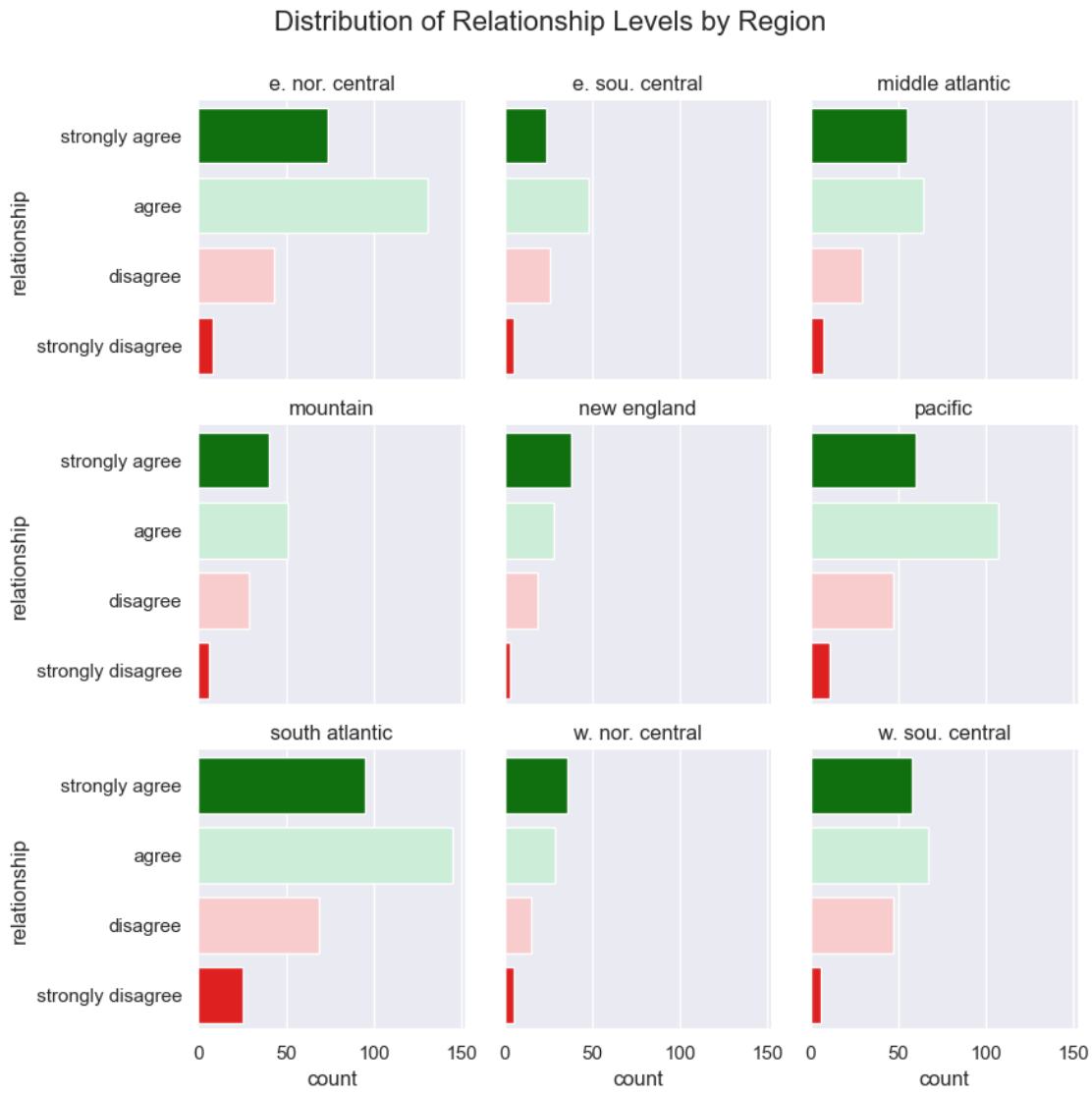
```
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
```

```
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
```

```
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
```

```
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
c:\Users\brian\AppData\Local\Programs\Python\Python312\Lib\site-
packages\seaborn\axisgrid.py:854: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  func(*plot_args, **plot_kwargs)
```

[201]: `Text(0.5, 0.98, 'Distribution of Relationship Levels by Region')`



1.5 Problem 3

Write code that exactly replicates the following figures, including all aesthetic choices. **Don't worry, however, about making the size of the figures exactly the same as that varies from browser to browser.** All of the following figures are generated by a primary graphing function from `seaborn`.

1.5.1 Part a

Replicate the following figure:

[Hint: the values of occupational prestige and socioeconomic status are the means of `job_prestige` and `socioeconomic_index` within years of `education`. Note that values of `education` less than 8 are excluded.] [2 points]

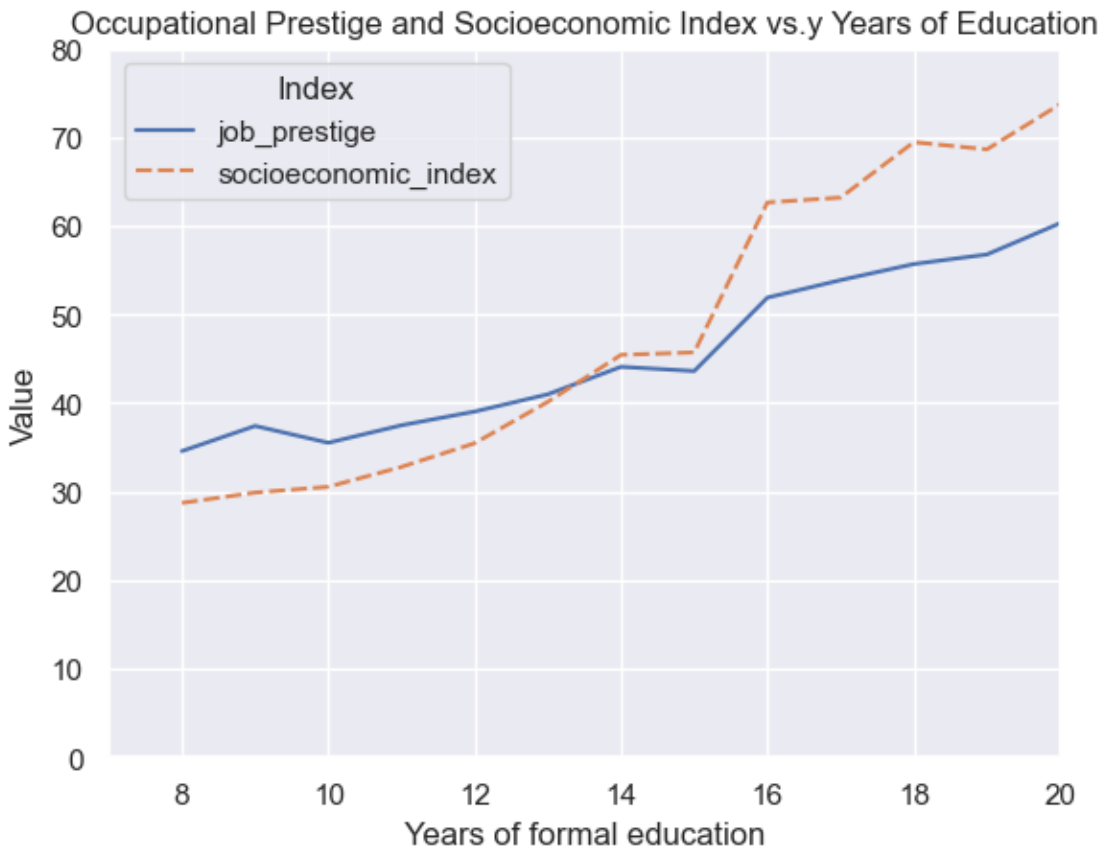
[202]:
```
gss_3a = gss_clean[gss_clean['education'].between(8, 20)].groupby('education').
  ↪agg({'job_prestige': 'mean', 'socioeconomic_index': 'mean'}).reset_index()
gss_3a = pd.melt(gss_3a, id_vars=['education'], value_vars=['job_prestige',␣
  ↪'socioeconomic_index'], var_name='Index', value_name='mean')
gss_3a
```

[202]:

	education	Index	mean
0	8.0	job_prestige	34.575758
1	9.0	job_prestige	37.416667
2	10.0	job_prestige	35.516667
3	11.0	job_prestige	37.494118
4	12.0	job_prestige	39.041667
5	13.0	job_prestige	40.988571
6	14.0	job_prestige	44.095710
7	15.0	job_prestige	43.616667
8	16.0	job_prestige	51.905213
9	17.0	job_prestige	53.885417
10	18.0	job_prestige	55.698276
11	19.0	job_prestige	56.777778
12	20.0	job_prestige	60.314286
13	8.0	socioeconomic_index	28.724242
14	9.0	socioeconomic_index	29.893750
15	10.0	socioeconomic_index	30.540000
16	11.0	socioeconomic_index	32.792941
17	12.0	socioeconomic_index	35.465224
18	13.0	socioeconomic_index	40.141714
19	14.0	socioeconomic_index	45.442904
20	15.0	socioeconomic_index	45.720833
21	16.0	socioeconomic_index	62.653318
22	17.0	socioeconomic_index	63.201042
23	18.0	socioeconomic_index	69.461207
24	19.0	socioeconomic_index	68.648889
25	20.0	socioeconomic_index	73.742857

[203]:
```
lg = sns.lineplot(x='education', y='mean',
             hue='Index', style='Index',
             data=gss_3a)
lg.set(xlim=(7,20), ylim=(0, 80))
plt.xlabel('Years of formal education')
plt.ylabel('Value')
plt.title('Occupational Prestige and Socioeconomic Index vs.y Years of␣
  ↪Education')
```

[203]:
```
Text(0.5, 1.0, 'Occupational Prestige and Socioeconomic Index vs.y Years of
Education')
```



1.5.2 Part b

Replicate the following figure:

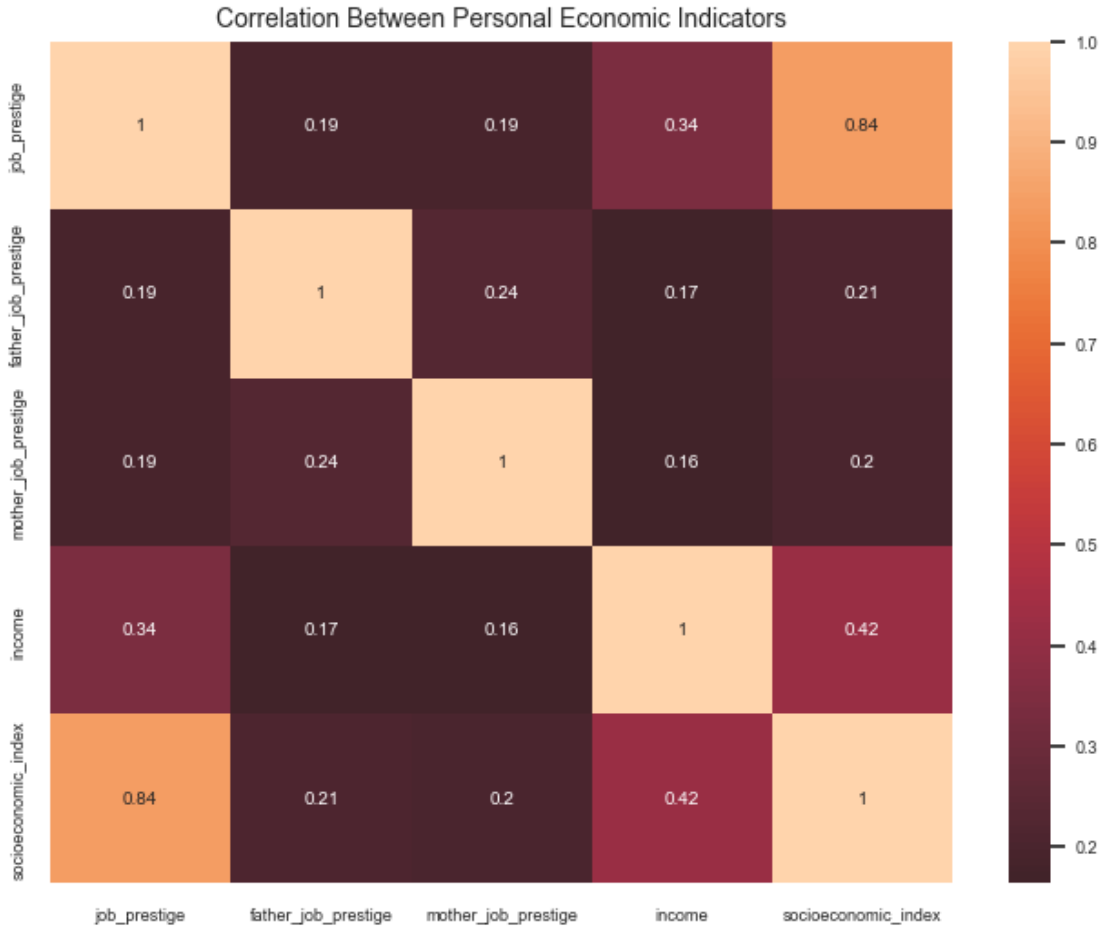
[Hint: to match the color scheme, you will need to set `center=0`.] [2 points]

[204]:
```
gss_3b = gss_clean.loc[:, ['job_prestige', 'father_job_prestige',␣
↪'mother_job_prestige', 'income', 'socioeconomic_index']].corr()
```

[205]:
```
plt.figure(figsize=(8, 6))
sns.set(font_scale=.6)
sns.heatmap(gss_3b, annot=True, center=0)
plt.title('Correlation Between Personal Economic Indicators', fontsize=10)
plt.xticks(rotation=0)
plt.yticks(rotation=90)
```

[205]:
```
(array([0.5, 1.5, 2.5, 3.5, 4.5]),
 [Text(0, 0.5, 'job_prestige'),
  Text(0, 1.5, 'father_job_prestige'),
  Text(0, 2.5, 'mother_job_prestige'),
```

```
Text(0, 3.5, 'income'),
Text(0, 4.5, 'socioeconomic_index')])
```



1.5.3 Part c

Replicate the following figure:

[Hint: The individual plots inside the grid have `height=4` and `aspect=1`, and to include the overall title I used

```
g.fig.subplots_adjust(top=.95)
g.fig.suptitle('Income vs. Prestige by Education', fontsize=16)
```

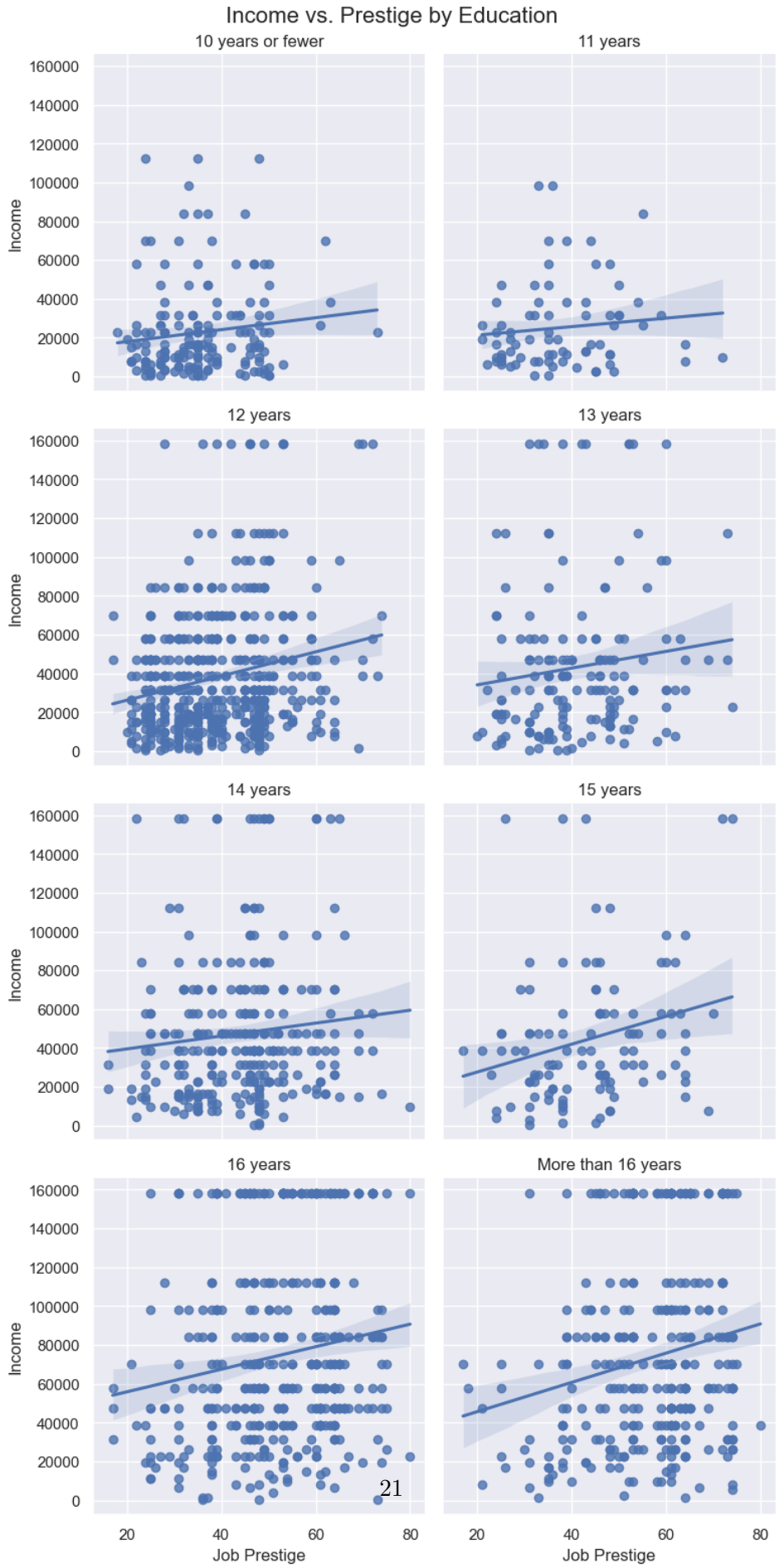
You will first need to create a version of `education` that collapses values from 0 to 10 to "10 years or fewer" and collapses values from 17 to 20 to "More than 16 years". You can use `.map()`, `.replace()`, or `pd.cut()` to do that.] [3 points]

[206]:
```
gss_3c = gss_clean
gss_3c = gss_3c.assign(education_years =
```

```
                        pd.cut(gss_3c.education,
                               bins=[-1, 10, 11, 12, 13, 14, 15, 16, 20],
                               labels=("10 years or fewer", "11 years", "12␣
↪years", "13 years", "14 years", "15 years", "16 years", "More than 16␣
↪years")))
```

[207]:
```
sns.set(font_scale=1)
g = sns.FacetGrid(gss_3c, col='education_years', height=4, aspect=1, col_wrap=2)
g.map(sns.regplot, 'job_prestige', 'income')
g.set_titles('{col_name}')
g.fig.subplots_adjust(top=.95)
g.fig.suptitle('Income vs. Prestige by Education', fontsize=16)
g.set_axis_labels('Job Prestige', 'Income', fontsize=12)
```

[207]:
```
<seaborn.axisgrid.FacetGrid at 0x157a0e19610>
```


Income vs. Prestige by Education
10 years or fewer
11 years
12 years
13 years
14 years
15 years
16 years
More than 16 years
Income
Job Prestige
0
20000
40000
60000
80000
100000
120000
140000
160000
20
40
60
80

1.6 Problem 4

There is a consistent finding that in the United States that women get paid only 80% of what men get paid. Other research however finds that the gap is much smaller when comparing men and women who hold the same job. In this problem you will use the GSS data to investigate the following questions:

1. Do men have higher incomes than women?
2. If there is a difference, is this difference due to the fact that men have jobs with higher occupational prestige than women?

You may use any kind of data visualization and you may use multiple visualizations to find an answer to these questions. In order to receive credit for this problem, you must write in text what parts of your visualizations are important and what we should learn from the visualizations to answer the questions. Please consider the entire distributions of income and occupational prestige, not just the means or medians. [4 points]

[208]:
```
gss_clean.groupby('sex').agg({'income': 'mean'})
```

[208]:
```
              income
sex
female  47191.021452
male    53314.626187
```

[209]:
```
53314.626187 - 47191.021452
```

[209]:
```
6123.604735000001
```

[210]:
```
47191.021452 / 53314.626187
```

[210]:
```
0.8851421237856648
```

[211]:
```
gss_clean.groupby('sex').agg({'income': 'median'})
```

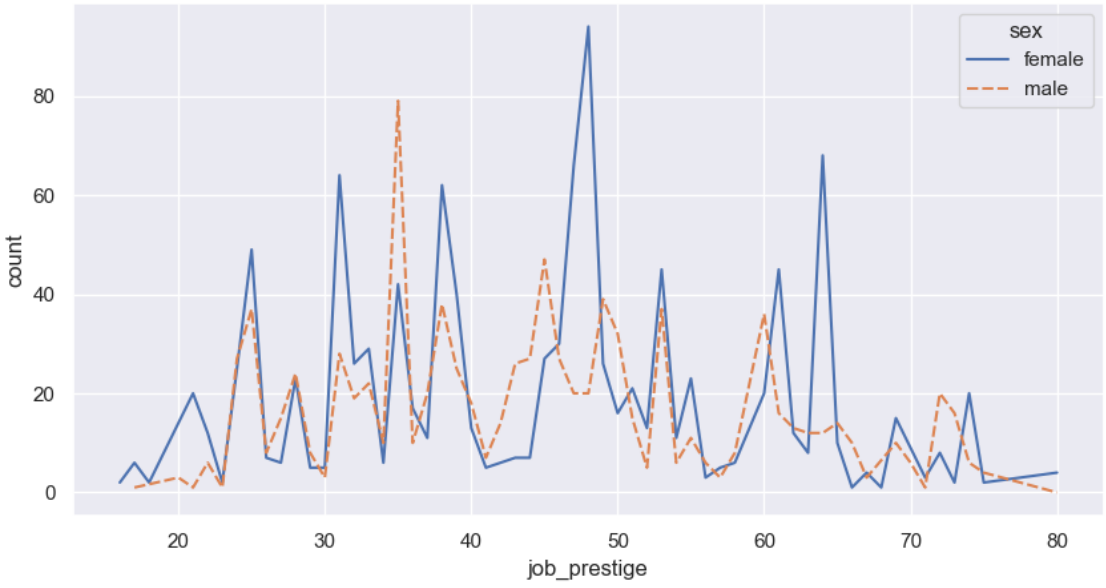
[211]:
```
         income
sex
female  38555.0
male    38555.0
```

[212]:
```
p4a = gss_clean.groupby(['job_prestige', 'sex']).agg({'income': 'count'}).
 ↪reset_index()
p4a = p4a.rename(columns={'income': 'count'})
p4b = gss_clean.groupby(['job_prestige', 'sex']).agg({'income': 'mean'}).
 ↪reset_index()
p4 = pd.merge(p4a, p4b, how='inner', on=['job_prestige', 'sex'])
p4
```

[212]:
```
     job_prestige     sex  count         income
0            16.0  female      2   25411.250000
1            17.0  female      6   52575.000000
2            17.0    male      1   47317.500000
3            18.0  female      2   40307.500000
4            20.0    male      3   12267.500000
..            ...     ...    ...            ...
110          74.0    male      6   62585.306867
111          75.0  female      2  114150.920600
112          75.0    male      4   75476.710300
113          80.0  female      4   57294.522800
114          80.0    male      0            NaN

[115 rows x 4 columns]
```

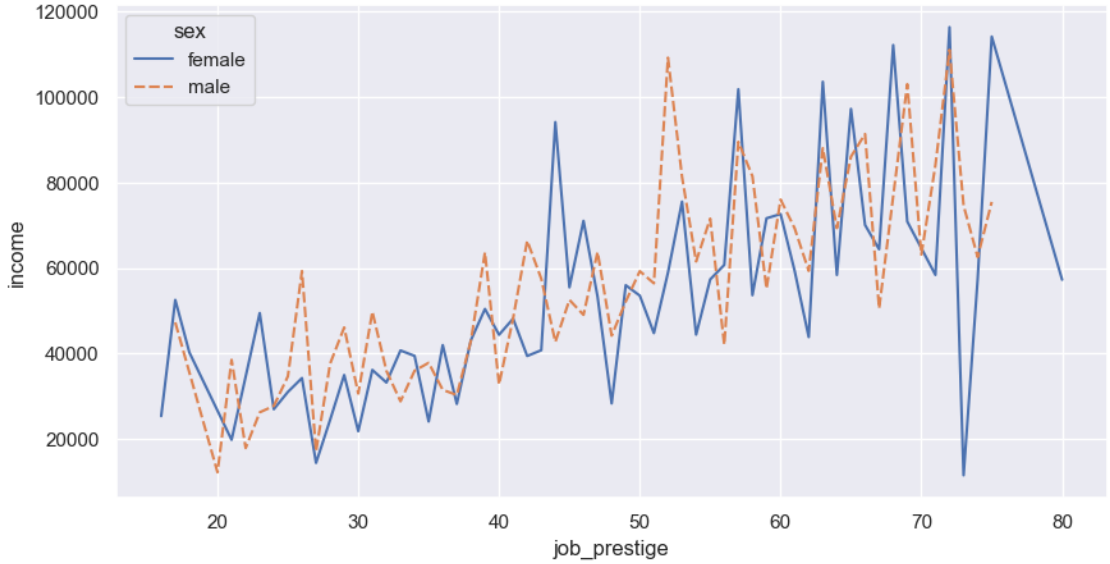
[213]:
```
plt.figure(figsize=(10, 5))
sns.lineplot(x='job_prestige', y='count',
             hue='sex', style='sex',
             data=p4)
```

[213]:
```
<Axes: xlabel='job_prestige', ylabel='count'>
```



[214]:
```
plt.figure(figsize=(10, 5))
sns.lineplot(x='job_prestige', y='income',
             hue='sex', style='sex',
             data=p4)
```

[214]: <Axes: xlabel='job_prestige', ylabel='income'>



[222]:
```
p4_men = p4[p4.sex == 'male']
p4_men = p4_men.rename(columns={'sex': 'male', 'count': 'men_count', 'income':␣
 ↪'men_income'})
p4_women = p4[p4.sex == 'female']
p4_women = p4_women.rename(columns={'sex': 'female','count': 'women_count',␣
 ↪'income': 'women_income'})
p4_diff = pd.merge(p4_men, p4_women, how='outer', on='job_prestige')
p4_diff = p4_diff.assign(count_diff = p4_diff.men_count - p4_diff.women_count)
p4_diff = p4_diff.assign(income_diff = p4_diff.men_income - p4_diff.
 ↪women_income)
p4_diff
```

[222]:

	job_prestige	male	men_count	men_income	female	women_count	\
0	16.0	NaN	NaN	NaN	female	2.0	
1	17.0	male	1.0	47317.500000	female	6.0	
2	18.0	NaN	NaN	NaN	female	2.0	
3	20.0	male	3.0	12267.500000	NaN	NaN	
4	21.0	male	1.0	38555.000000	female	20.0	
5	22.0	male	6.0	17933.916667	female	12.0	
6	23.0	male	1.0	26287.500000	female	2.0	
7	24.0	male	27.0	27780.370370	female	25.0	
8	25.0	male	37.0	34712.387600	female	49.0	
9	26.0	male	8.0	59316.011400	female	7.0	
10	27.0	male	15.0	17139.450000	female	6.0	
11	28.0	male	24.0	37669.410050	female	23.0	
12	29.0	male	8.0	46134.562500	female	5.0	

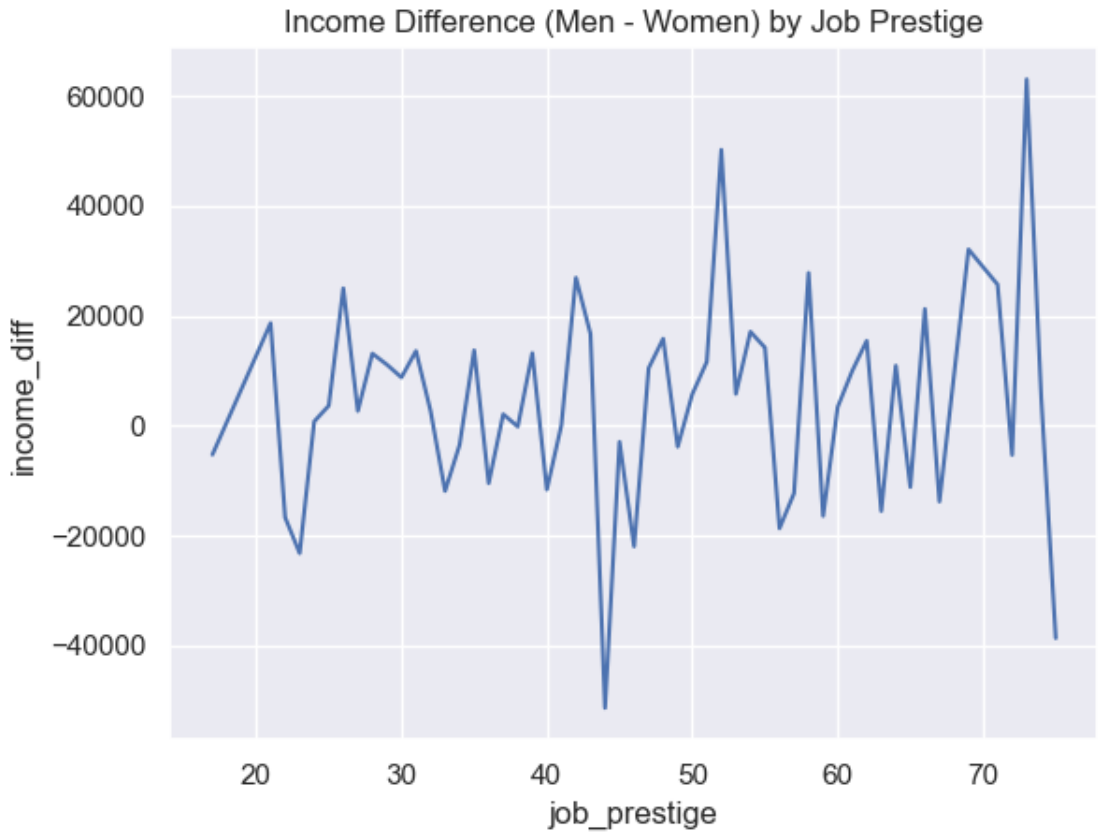
13	30.0	male	3.0	30668.750000	female	5.0
14	31.0	male	28.0	49811.319014	female	64.0
15	32.0	male	19.0	35972.368421	female	26.0
16	33.0	male	22.0	28860.488636	female	29.0
17	34.0	male	10.0	36013.875000	female	6.0
18	35.0	male	79.0	37864.472673	female	42.0
19	36.0	male	10.0	31545.000000	female	17.0
20	37.0	male	20.0	30335.775000	female	11.0
21	38.0	male	38.0	42990.305032	female	62.0
22	39.0	male	25.0	63672.970944	female	40.0
23	40.0	male	18.0	32791.222222	female	13.0
24	41.0	male	7.0	48286.905886	female	5.0
25	42.0	male	14.0	66412.763029	female	6.0
26	43.0	male	26.0	57532.160862	female	7.0
27	44.0	male	27.0	42752.697822	female	7.0
28	45.0	male	47.0	52575.606885	female	27.0
29	46.0	male	27.0	49079.358607	female	30.0
30	47.0	male	20.0	63782.237500	female	66.0
31	48.0	male	20.0	44215.575000	female	94.0
32	49.0	male	39.0	52175.801887	female	26.0
33	50.0	male	32.0	59297.161950	female	16.0
34	51.0	male	15.0	56450.606080	female	21.0
35	52.0	male	5.0	109196.236480	female	13.0
36	53.0	male	37.0	81335.936811	female	45.0
37	54.0	male	6.0	61563.015200	female	11.0
38	55.0	male	11.0	71577.212836	female	23.0
39	56.0	male	6.0	42060.000000	female	3.0
40	57.0	male	3.0	89536.447067	female	5.0
41	58.0	male	8.0	81500.929050	female	6.0
42	59.0	male	22.0	55225.424600	female	13.0
43	60.0	male	36.0	76032.234889	female	20.0
44	61.0	male	16.0	69313.157725	female	45.0
45	62.0	male	13.0	59352.064708	female	12.0
46	63.0	male	12.0	88036.293633	female	8.0
47	64.0	male	12.0	69351.111767	female	68.0
48	65.0	male	14.0	86071.329629	female	10.0
49	66.0	male	10.0	91360.677360	female	1.0
50	67.0	male	3.0	50588.833333	female	4.0
51	68.0	NaN	NaN	NaN	female	1.0
52	69.0	male	10.0	103014.802360	female	15.0
53	70.0	male	6.0	63169.473533	NaN	NaN
54	71.0	male	1.0	84120.000000	female	3.0
55	72.0	male	20.0	111016.391040	female	8.0
56	73.0	male	16.0	74650.386400	female	2.0
57	74.0	male	6.0	62585.306867	female	20.0
58	75.0	male	4.0	75476.710300	female	2.0
59	80.0	male	0.0	NaN	female	4.0

	women_income	count_diff	income_diff
0	25411.250000	NaN	NaN
1	52575.000000	-5.0	-5257.500000
2	40307.500000	NaN	NaN
3	NaN	NaN	NaN
4	19864.587500	-19.0	18690.412500
5	34637.007600	-6.0	-16703.090933
6	49508.125000	-1.0	-23220.625000
7	27016.540000	2.0	763.830370
8	31058.591837	-12.0	3653.795763
9	34298.928571	1.0	25017.082829
10	14458.125000	9.0	2681.325000
11	24535.000000	1.0	13134.410050
12	35014.950000	3.0	11119.612500
13	21871.200000	-2.0	8797.550000
14	36202.497888	-36.0	13608.821127
15	33255.176585	-7.0	2717.191836
16	40753.003145	-7.0	-11892.514508
17	39481.515200	4.0	-3467.640200
18	24138.601190	37.0	13725.871483
19	42015.887718	-7.0	-10470.887718
20	28247.113636	9.0	2088.661364
21	43143.841652	-24.0	-153.536620
22	50457.123900	-15.0	13215.847044
23	44419.134615	5.0	-11627.912393
24	48193.750000	2.0	93.155886
25	39460.458333	8.0	26952.304695
26	40808.214286	19.0	16723.946576
27	94145.347486	20.0	-51392.649663
28	55513.494119	20.0	-2937.887233
29	71058.304613	-3.0	-21978.946006
30	53306.813727	-46.0	10475.423773
31	28372.680132	-74.0	15842.894868
32	56006.952831	13.0	-3831.150944
33	53568.677575	16.0	5728.484375
34	44805.583333	-6.0	11645.022747
35	58947.641631	-8.0	50248.594849
36	75540.266738	-8.0	5795.670073
37	44449.772727	-5.0	17113.242473
38	57361.327113	-12.0	14215.885723
39	60753.333333	3.0	-18693.333333
40	101835.736480	-2.0	-12299.289413
41	53676.765200	2.0	27824.163850
42	71656.244800	9.0	-16430.820200
43	72648.868240	16.0	3383.366649
44	59354.774773	-29.0	9958.382952

```
45   43885.520833         1.0  15466.543874
46  103576.315450         4.0 -15540.021817
47   58415.367994       -56.0  10935.743773
48   97231.552360         4.0 -11160.222731
49   70100.000000         9.0  21260.677360
50   64404.375000        -1.0 -13815.541667
51  112160.000000         NaN           NaN
52   70904.790987        -5.0  32110.011373
53            NaN         NaN           NaN
54   58416.666667        -2.0  25703.333333
55  116341.545600        12.0  -5325.154560
56   11566.500000        14.0  63083.886400
57   57418.217060       -14.0   5167.089807
58  114150.920600         2.0 -38674.210300
59   57294.522800        -4.0           NaN
```

[216]:
```
sns.lineplot(x='job_prestige', y='income_diff', data=p4_diff)
plt.title('Income Difference (Men - Women) by Job Prestige')
```

[216]:
```
Text(0.5, 1.0, 'Income Difference (Men - Women) by Job Prestige')
```



[217]:
```
print(p4_diff[p4_diff.income_diff > 0].shape[0])
print(p4_diff[p4_diff.income_diff == 0].shape[0])
print(p4_diff[p4_diff.income_diff < 0].shape[0])
```

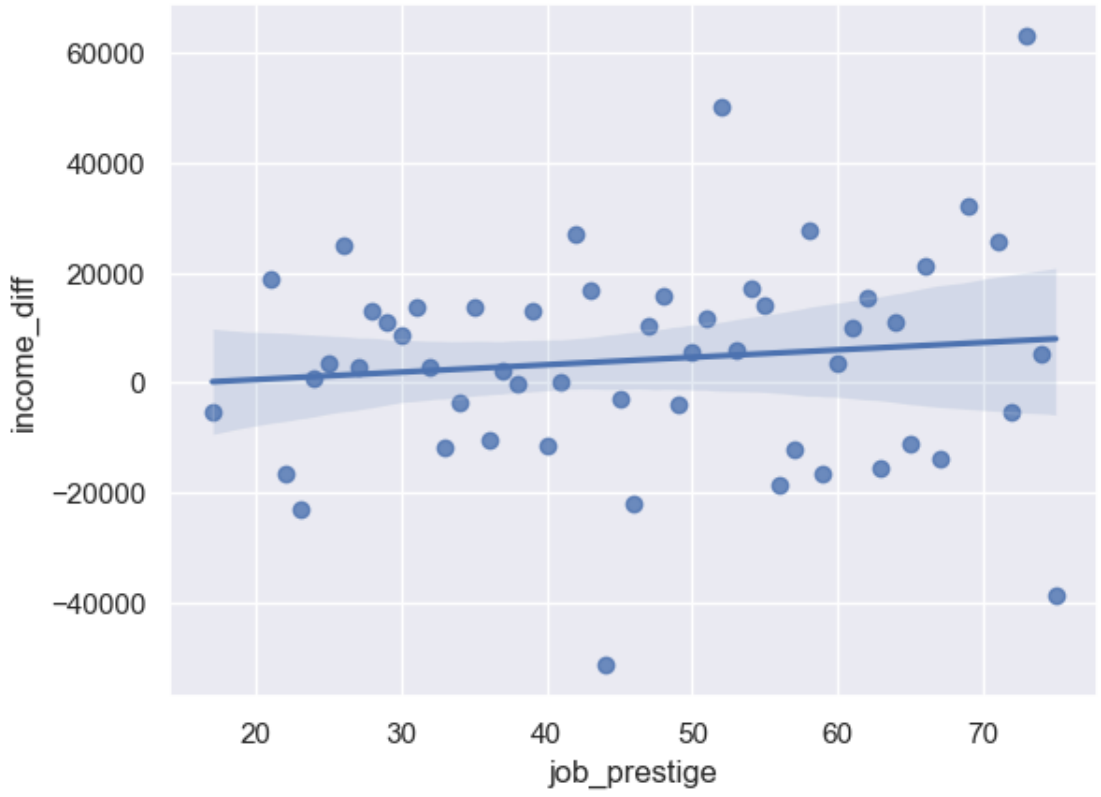
```
34
0
20
```

[218]:
```
print(p4_diff['income_diff'].mean())
print(p4_diff.loc[p4_diff['income_diff'].idxmin()]['job_prestige'])
print(p4_diff['income_diff'].min())
print(p4_diff.loc[p4_diff['income_diff'].idxmax()]['job_prestige'])
print(p4_diff['income_diff'].max())
```

```
4149.431519145973
44.0
-51392.64966349207
73.0
63083.8864
```

[219]:
```
sns.regplot(x='job_prestige', y='income_diff', data=p4_diff)
```

[219]:
```
<Axes: xlabel='job_prestige', ylabel='income_diff'>
```



[227]:
```
p4_diff2 = p4_diff.dropna()

from scipy import stats

slope, intercept, r_value, p_value, std_err = stats.
 ↪linregress(p4_diff2['job_prestige'], p4_diff2['income_diff'])

print(f"y = {intercept:.2f} + {slope:.2f}x, R^2 = {r_value**2:.2f}, p =␣
 ↪{p_value:.2f}")
```

```
y = -2135.23 + 134.72x, R^2 = 0.01, p = 0.42
```

To answer this question I first found the mean of the men's and women's salaries, which are $53,314.63 and $47,191.02 respectively for a difference of $6,123.60. As a ratio of women to men it's .885 meaning that women make 88-89% of what men make. Then I calculated the median of both men and women which was equal at $38,555 indicating there is no difference. To investigate further I grouped the dataframe by prestige level and sex and made a line plot of mean income per prestige level for men and women. While the graphic was interesting I ultimately found it not helpful in answering the question. Next I sought to find the difference in mean salary between men and women at each prestige level. I graphed this with another line plot and found it easier to read though but it needed some help since there were many point above the 0 line as below. To help understand I found that there were 34 prestige levels with a positive difference (men making more than women), 20 prestige levels with a negative difference, and no presitge levels with no difference (equal pay). The mean of the differences is $4,149.43 indicating that men make more overall, but we knew that. There are some extreme differences both directions. For example, with a prestige level of 44, women make more than men of average by $51,392.65 and at a prestige level of 73 men make more on average by $63,083.89. Both of these could have an effect on the mean of the difference, and since the prestige level is higher for the male difference could explain the perception that men make more at higher levels of prestige. Looking at the line plot however, there is no real trend (up or down) as the graph bounces up and down throughout. To get a clearer look at the trend I plotted a regplot of the prestige level vs income difference and you can see a slight positive slope. With the 95% confidence shaded on the regplot, the 0 line was completely contained within the interval making no difference in salary a realistic possibility at each prestige level. The linear regression equation is y = -2135.23 + 134.72x meaning that for every 1 increase in prestige, men are expected to make more than women by $134.72. This has a p-value of 0.42 meaning there is not convincing evidence that this slope is not 0. All this leads me to believe that the difference in mean salary between men and women is not due to prestige level.

I started down a road of doing something with counts, but then I realized the difference are most likely due to the unequal surveying of men and women and therefore not useful. I just didn't take out the code…