Hayden Schindler | Alexander Hasmuller CS 4330 - 010

Imports >>>

```
!pip install requests
!pip install html5lib
!pip install bs4
!pip install pandas
!pip install altair
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.27.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from re
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from reque
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: html5lib in /usr/local/lib/python3.10/dist-packages (1.1)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from html5lib) (0.5
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/dist-packages (from html5lib) (1.16.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting bs4
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from bs4) (4.11.2
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4
Building wheels for collected packages: bs4
  Building wheel for bs4 (setup.py) ... done
  Created wheel for bs4: filename=bs4-0.0.1-py3-none-any.whl size=1270 sha256=92a7ec4774efe71c2599a6f63db2b5
  Stored in directory: /root/.cache/pip/wheels/25/42/45/b773edc52acb16cd2db4cf1a0b47117e2f69bb4eb300ed0e70
Successfully built bs4
Installing collected packages: bs4
Successfully installed bs4-0.0.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from panda
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2022.7
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.22
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: altair in /usr/local/lib/python3.10/dist-packages (4.2.2)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from altair) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair) (4.3
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from altair) (1.22.4)
Requirement already satisfied: pandas>=0.18 in /usr/local/lib/python3.10/dist-packages (from altair) (1.5.3
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair) (0.12.0)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->a
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from panda
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->alta
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2
```

```
from google.colab import files
measure_data = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving 2022 County Health Rankings Data - v1.csv to 2022 County Health Rankings Data - v1.csv
```

```
from google.colab import files
suicide_data = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Suicides.csv to Suicides.csv
```

```
import pandas as pd


import io
df1 = pd.read_csv(io.BytesIO(measure_data['2022 County Health Rankings Data - v1.csv']))
```

Change the column headers to something that is esier to search with. >>

```
df1.columns=df1.iloc[0]


df1 = df1.drop(labels=0, axis=0)


df1['FIPS'] = df1.FIPS.astype(int)


from google.colab import files
add_measure_data = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving ADD 2022 County Health Rankings Data - v1.csv to ADD 2022 County Health Rankings Data - v1.csv
```

```
import io
df2 = pd.read_csv(io.BytesIO(add_measure_data['ADD 2022 County Health Rankings Data - v1.csv']))
```

```
<ipython-input-10-ece793bece60>:2: DtypeWarning: Columns (0,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
  df2 = pd.read_csv(io.BytesIO(add_measure_data['ADD 2022 County Health Rankings Data - v1.csv']))
```

```
import io
df3 = pd.read_csv(io.BytesIO(suicide_data['Suicides.csv']))
```

Change the column headers to something that is esier to search with. >>>

```
df2
```

	Unnamed: 0	Unnamed: 1	Unnamed: 2	COVID-19 age-adjusted mortality	Unnamed: 4	Life expectancy	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9
				# deaths during 2020					(AIAN)	Life 95% CI -

- Set the column headers to row 0

```
df2.columns
```

```
Index(['Unnamed: 0', 'Unnamed: 1', 'Unnamed: 2',
       'COVID-19 age-adjusted mortality', 'Unnamed: 4', 'Life expectancy',
       'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9',
       ...
       'Unnamed: 279', '% non-Hispanic white', 'Unnamed: 281',
       '% not proficient in English', 'Unnamed: 283', 'Unnamed: 284',
       'Unnamed: 285', '% female', '% rural', 'Unnamed: 288'],
      dtype='object', length=289)
```

```
df2.columns=df2.iloc[0]
```

- Drop the repeated titles for the columns

```
df2 = df2.drop(labels=0, axis=0)
```

- Fill the NaN spaces with blank

```
df2.fillna(0)
```

	FIPS	State	County	# deaths due to COVID-19 during 2020	COVID-19 death rate	Life Expectancy	95% CI - Low	95% CI - High	Life Expectancy (AIAN)	Life Expectancy (AIAN) 95% CI - Low	...	% Hispanic	F
1	01000	Alabama	0	6544	104	74.8	74.8	74.9	111.8	99.3	...	4.6	
2	01001	Alabama	Autauga	55	82	76.6	75.8	77.3	0	0	...	3.1	
3	01003	Alabama	Baldwin	165	50	77.7	77.3	78.1	104.6	67.6	...	4.6	
4	01005	Alabama	Barbour	38	110	72.9	71.6	74.1	0	0	...	4.7	
5	01007	Alabama	Bibb	40	147	73.6	72.4	74.8	0	0	...	2.9	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3189	56037	Wyoming	Sweetwater	22.0	60.0	76.5	75.6	77.3	0	0	...	16.2	
3190	56039	Wyoming	Teton	0	0	86.7	85.0	88.4	0	0	...	15.0	
3191	56041	Wyoming	Uinta	0	0	77.0	75.5	78.5	0	0	...	9.9	
3192	56043	Wyoming	Washakie	21.0	147.0	78.8	76.8	80.7	0	0	...	14.2	
3193	56045	Wyoming	Weston	0	0	80.7	78.7	82.6	0	0	...	4.1	

3193 rows × 289 columns

<<< Fix the FIPS number so that they're ints instead of strings >>>

```
df2['FIPS'] = df2.FIPS.astype(int)
```

- Set the column headers to row 0

```
df3.columns=df3.iloc[0]
```

- Drop the repeated titles for the columns

```
df3 = df3.drop(labels=0, axis=0)
```

```
df3
```

	# Deaths
1	4043
2	52
3	214
4	20
5	18
...	...
3184	67
3185	17
3186	31
3187	-
3188	-

3188 rows × 1 columns

Q1.) Is there a connection between areas that have higher broadband access and suicide rates or an increased amount of poor mental health days? (Uses Ranked measure data and additional measure data)

- In the excel file I added '-' to each blank space contained in the column to keep the spacing correct when exporting it to .csv. Removed those below

```
df3['# Deaths'] = df3['# Deaths'].str.replace('-', '')
```

```
df3
```

	# Deaths
1	4043
2	52
3	214
4	20
5	18

```
df3 = df3.replace('',0)
```

```
df3
```

	# Deaths
1	4043
2	52
3	214
4	20
5	18
...	...
3184	67
3185	17
3186	31
3187	0
3188	0

3188 rows × 1 columns

```
df2
```

```
                                        #
States = pd.DataFrame()
States  = df1['State']
FIPS = pd.DataFrame()
FIPS = df1['FIPS']
BA = pd.DataFrame()
BA = df2['% Broadband Access']
MUD = pd.DataFrame()
MUD = df1['Average Number of Mentally Unhealthy Days']
suicides = pd.DataFrame()
suicides = df3['# Deaths']
```

- Might need this function to re-map the FIPS to the counties. >>> (Not needed, just had to make the FIPS ints instead of strings)

```
from pandas.core.ops import common
def get_fips(lst):
  nums = {}
  for county in lst:
    fips = df2
    fips = fips.to_json()
    d = dict(fips[0])
    nums[county] = int(d['County'])
    # print(country, codes[country])
  return fips
```

- Combine selected columns into a sepearte DataFrame

```
Combine1 = pd.DataFrame()
Combine1 = [FIPS, States, BA, MUD, suicides]


Q1 = pd.concat(Combine1, axis=1)


Q1 = Q1.fillna(0)


Q1
```

	FIPS	State	% Broadband Access	Average Number of Mentally Unhealthy Days	# Deaths
1	1000	Alabama	80	5.6	4043
2	1001	Alabama	83	5.4	52
3	1003	Alabama	85	5.2	214
4	1005	Alabama	65	6.1	20
5	1007	Alabama	76	5.8	18
...	...	...	...	...	...
3189	56037	Wyoming	87	3.9	0
3190	56039	Wyoming	89	3.3	0
3191	56041	Wyoming	91	4.3	0
3192	56043	Wyoming	83	4.0	0
3193	56045	Wyoming	80	4.2	0

3193 rows × 5 columns

```
Q1['id'] = Q1['FIPS']
```

Q1

	FIPS	State	% Broadband Access	Average Number of Mentally Unhealthy Days	# Deaths	id
1	1000	Alabama	80	5.6	4043	1000
2	1001	Alabama	83	5.4	52	1001
3	1003	Alabama	85	5.2	214	1003
4	1005	Alabama	65	6.1	20	1005
5	1007	Alabama	76	5.8	18	1007
...	...	...	...	...	...	...
3189	56037	Wyoming	87	3.9	0	56037
3190	56039	Wyoming	89	3.3	0	56039
3191	56041	Wyoming	91	4.3	0	56041
3192	56043	Wyoming	83	4.0	0	56043
3193	56045	Wyoming	80	4.2	0	56045

3193 rows × 6 columns

- Uploads for sorting the data per capita

```
suicide_data = files.upload()
bad_mental_data = files.upload()
broadband_data = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving population - Suicide Rate.csv to population - Suicide Rate.csv
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving bad_mental - bad_mental.csv to bad_mental - bad_mental.csv
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving broadband - broadband.csv to broadband - broadband.csv
```

```
suicide = pd.read_csv(io.BytesIO(suicide_data['population - Suicide Rate.csv']))
bad_mental = pd.read_csv(io.BytesIO(bad_mental_data['bad_mental - bad_mental.csv']))
broadband = pd.read_csv(io.BytesIO(broadband_data['broadband - broadband.csv']))
```

```
suicide_alpha = suicide.sort_values('Suicide Rate')
bad_mental_alpha = bad_mental.sort_values('% Frequent Mental Distress')
broadband_alpha = broadband.sort_values('% Broadband Access')
```

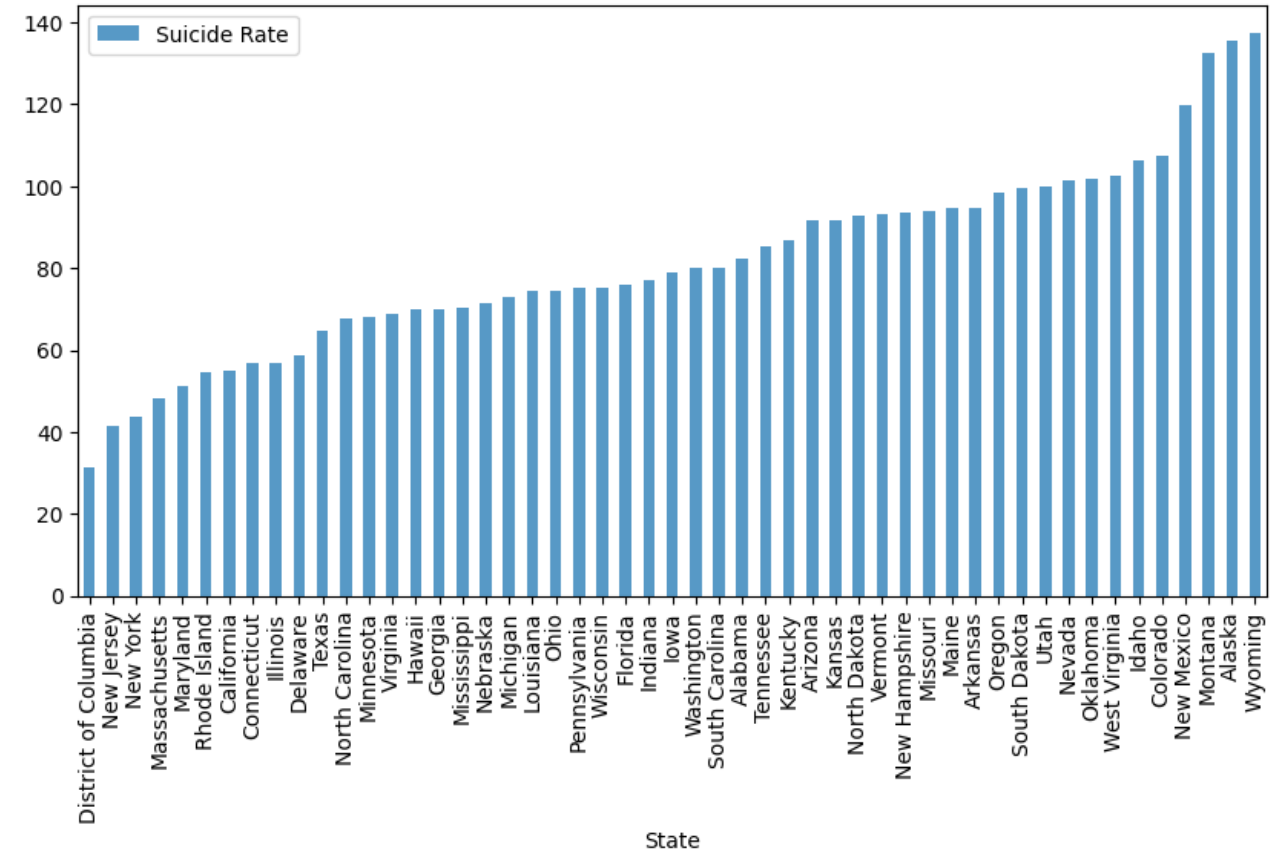
Organized the suicide data per capita here >>>

```
suicide_alpha.set_index(['State'])
```

State	Suicide Rate
District of Columbia	31.28
New Jersey	41.64
New York	43.61
Massachusetts	48.07
Maryland	51.31
Rhode Island	54.68
California	54.94
Connecticut	56.73
Illinois	56.97
Delaware	58.67
Texas	64.64
North Carolina	67.74
Minnesota	68.02
Virginia	68.92
Hawaii	69.94
Georgia	69.99
Mississippi	70.48
Nebraska	71.43
Michigan	73.04
Louisiana	74.48
Ohio	74.61
Pennsylvania	75.07
Wisconsin	75.20
Florida	75.95
Indiana	76.98
Iowa	78.93
Washington	79.98
South Carolina	80.16
Alabama	82.15
Tennessee	85.26
Kentucky	86.64
Arizona	91.76
Kansas	91.80
North Dakota	92.90
Vermont	93.05
New Hampshire	93.47
Missouri	93.91
Maine	94.66
Arkansas	94.67

```
suicide_alpha.plot(x='State', kind='bar',alpha=0.75, figsize=(10,5))
```

```
<Axes: xlabel='State'>
```

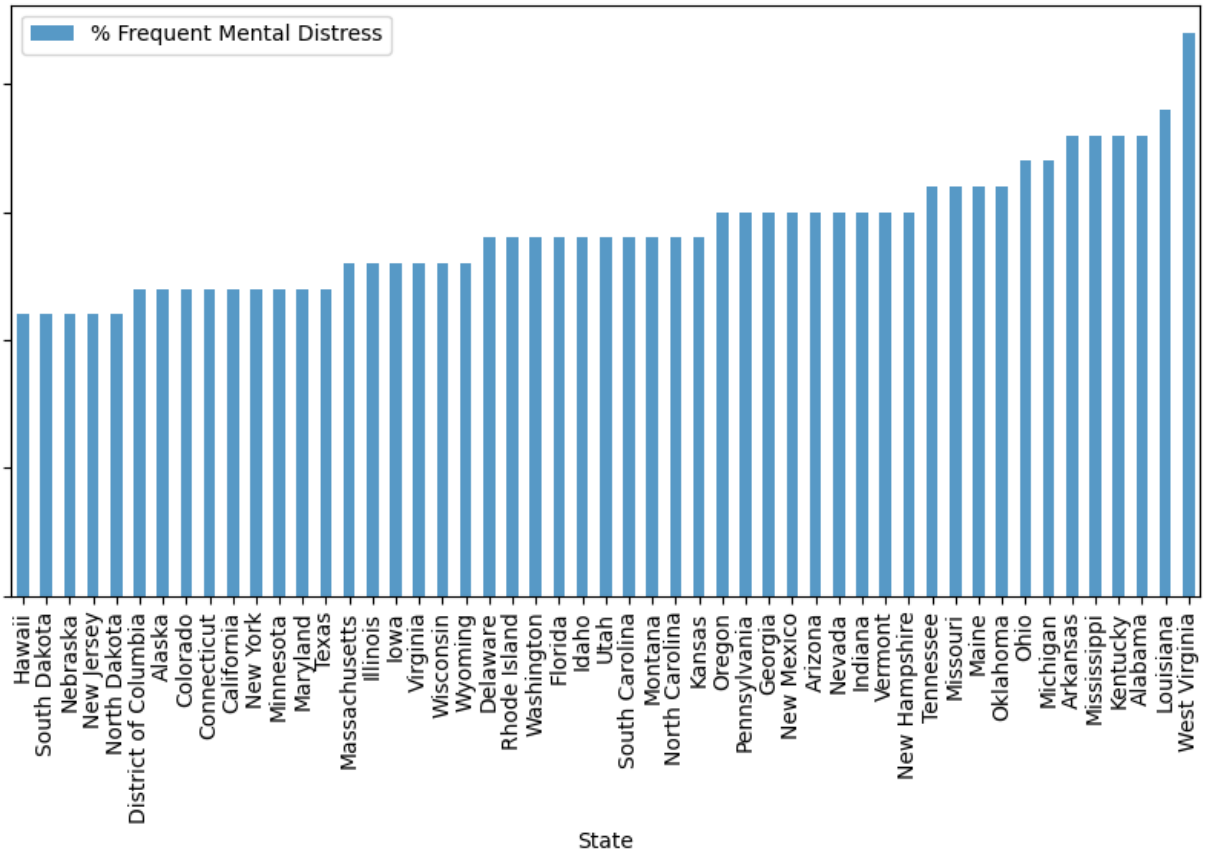



```
broadband_alpha.plot(x='State', kind='bar',alpha=0.75, figsize=(10,5))
```

```
<Axes: xlabel='State'>
```

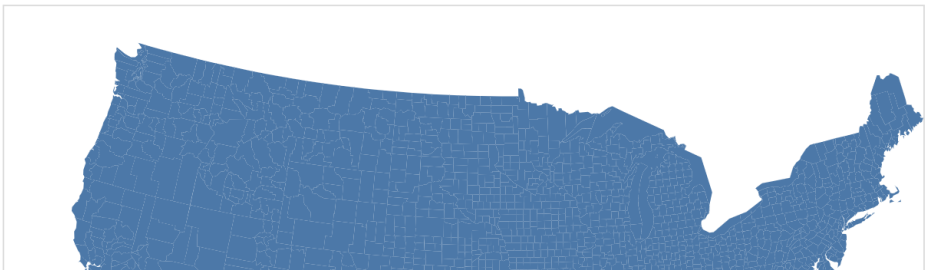
```
bad_mental_alpha.plot(x='State', kind='bar',alpha=0.75, figsize=(10,5))
```

```
<Axes: xlabel='State'>
```

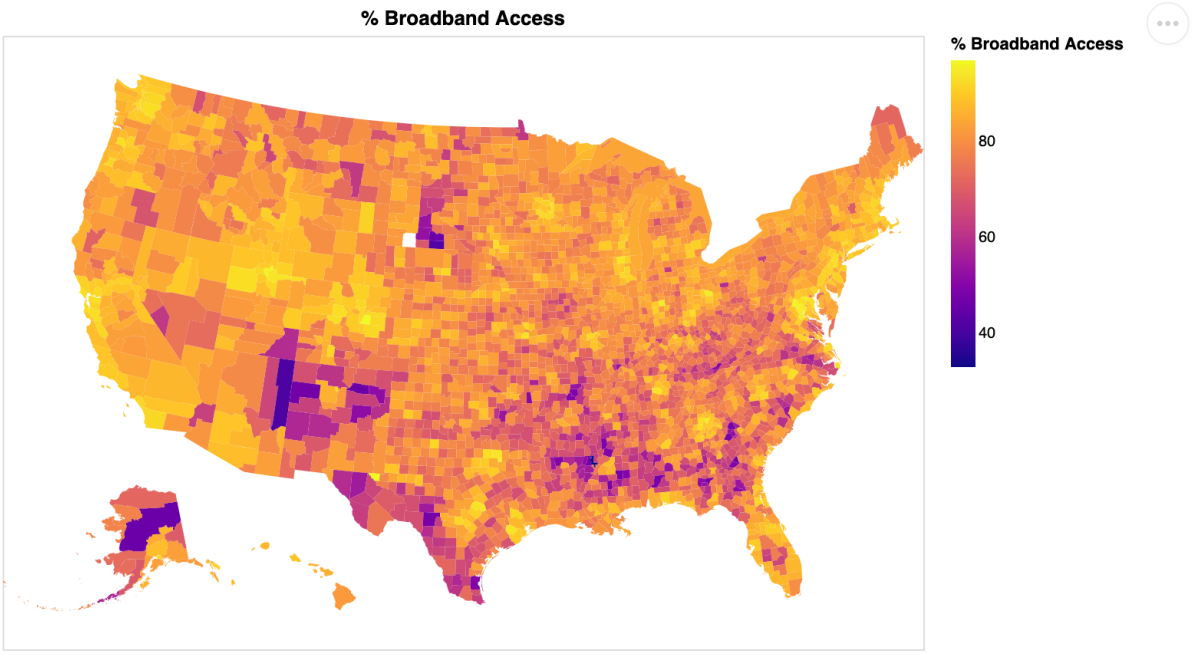


- Visualization for the % Broadband Access used in Q1

```
import pandas as pd
import altair as alt
from vega_datasets import data
counties = alt.topo_feature(data.us_10m.url, 'counties')


alt.Chart(counties).mark_geoshape().project(
    type='albersUsa').properties(
    width=600,
    height=400
)
```



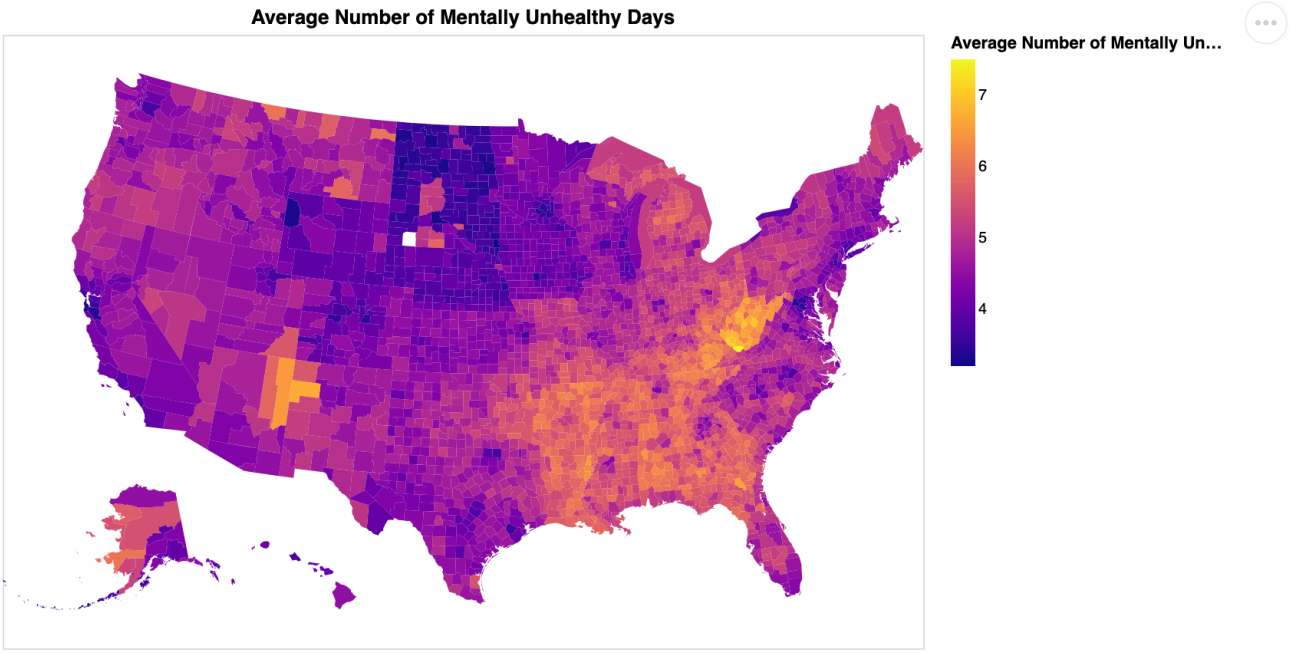
```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('% Broadband Access:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q1, 'FIPS', ['% Broadband Access'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Broadband Access'
)
```



Visualization for the Average Number of Mentally Unhealthy Days used in Q1

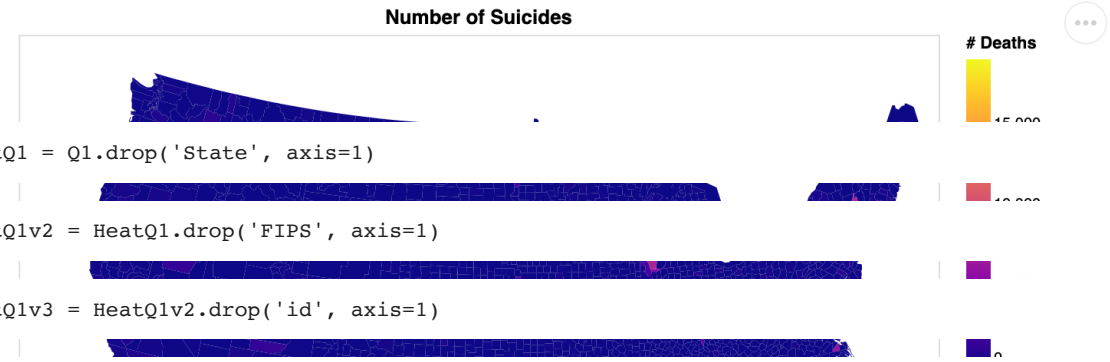
```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('Average Number of Mentally Unhealthy Days:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q1, 'FIPS', ['Average Number of Mentally Unhealthy Days'])
).project(
    type='albersUsa'
).properties(
    width=600,
```

```
    height=400,
    title='Average Number of Mentally Unhealthy Days'
)
```



- Visualization for the number of suicides used in Q1

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('# Deaths:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q1, 'FIPS', ['# Deaths'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='Number of Suicides'
)
```




```
HeatQ1 = Q1.drop('State', axis=1)
```

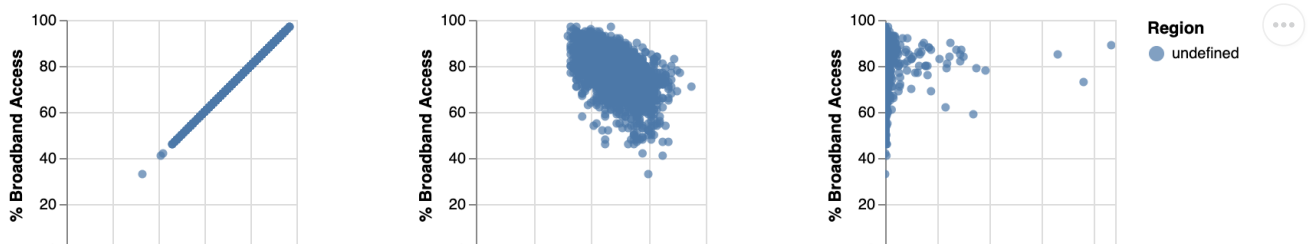
```
HeatQ1v2 = HeatQ1.drop('FIPS', axis=1)
```

```
HeatQ1v3 = HeatQ1v2.drop('id', axis=1)
```

```
corr = HeatQ1v3.astype('float64').corr()
corr.style.background_gradient(cmap='coolwarm')
```

	% Broadband Access	Average Number of Mentally Unhealthy Days	# Deaths
% Broadband Access	1.000000	-0.526111	0.062272
Average Number of Mentally Unhealthy Days	-0.526111	1.000000	-0.045780

```
import altair
alt.Chart(Q1).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='Region:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['% Broadband Access', 'Average Number of Mentally Unhealthy Days', '# Deaths'],
    column=['% Broadband Access','Average Number of Mentally Unhealthy Days', '# Deaths']
).interactive()
```



Q1: Majority of the States in the US have 80% or higher broadband access. So, when exploring for a correlation between suicide rates and the percentage of broadband access the amount of broadband access is too high to determine if that influences the suicide rate. If anything there's a negative correlation between % Broadband Access and the Average number of mentally unhealthy days.

Q2.) - Is there a connection between communities that have a higher number of single-parent households and the amount of teen pregnancy and or juvenile arrests?

- Single parent household frame building

```
States = pd.DataFrame()
States  = df1['State']
FIPS = pd.DataFrame()
FIPS = df1['FIPS']
SPH = pd.DataFrame()
SPH = df1['# Children in Single-Parent Households']
Juvenile_arrests = pd.DataFrame()
Juvenile_arrests = df2['Juvenile Arrest Rate']
Teen_pregnancy = pd.DataFrame()
Teen_pregnancy = df1['Teen Birth Rate']


Combine2 = pd.DataFrame()
Combine2 = [FIPS, States, SPH, Juvenile_arrests, Teen_pregnancy]


Q2 = pd.concat(Combine2, axis=1)


Q2
```

	FIPS	State	# Children in Single-Parent Households	Juvenile Arrest Rate	Teen Birth Rate
1	1000	Alabama	339392	NaN	28

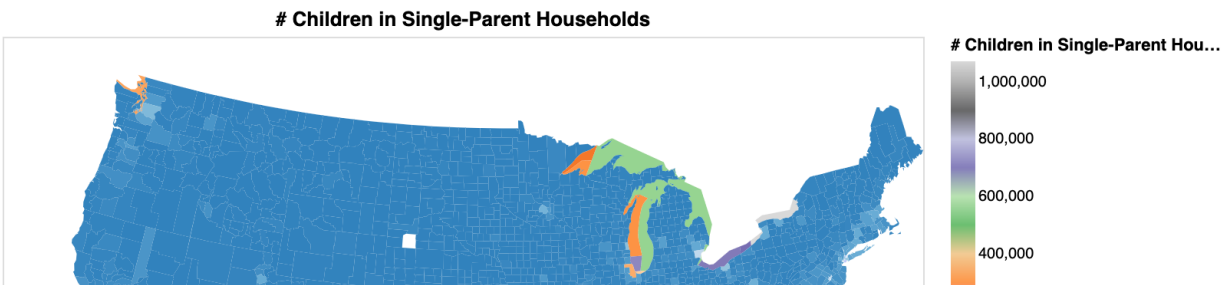
```
Q2 = Q2.fillna(0)

Q2
```

	FIPS	State	# Children in Single-Parent Households	Juvenile Arrest Rate	Teen Birth Rate
1	1000	Alabama	339392	0	28
2	1001	Alabama	3628	8	23
3	1003	Alabama	8613	26	24
4	1005	Alabama	2822	24	35
5	1007	Alabama	1666	0	34
...	...	...	...	...	...
3189	56037	Wyoming	1774	13.0	27
3190	56039	Wyoming	773	6.0	9
3191	56041	Wyoming	628	6.0	26
3192	56043	Wyoming	292	11.0	21
3193	56045	Wyoming	151	8.0	19

3193 rows × 5 columns

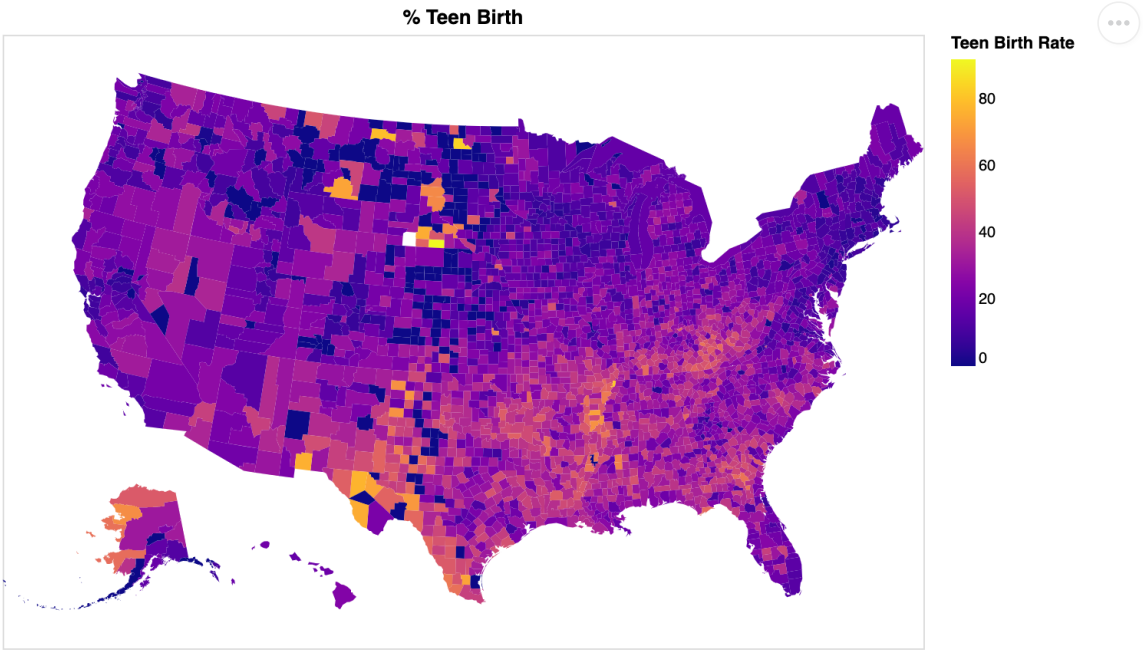
- Visualization for the number of Children living in a Single-Parent Households used in Q2

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('# Children in Single-Parent Households:Q', scale=alt.Scale(scheme="category20c"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q2, 'FIPS', ['# Children in Single-Parent Households'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='# Children in Single-Parent Households'
)
```



- Visualization for the teen birth rate used in Q2

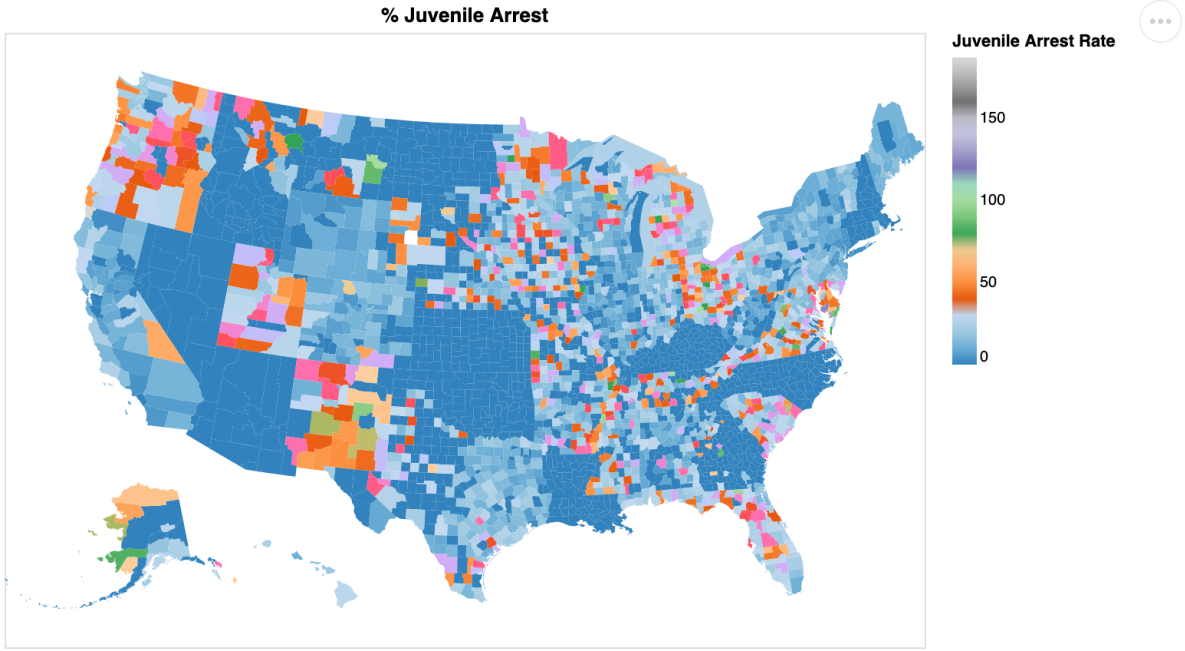
```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('Teen Birth Rate:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q2, 'FIPS', ['Teen Birth Rate'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Teen Birth'
)
```



- Visualization for the Juvenile Arrest rate used in Q2

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('Juvenile Arrest Rate:Q', scale=alt.Scale(scheme="category20c"))
).transform_lookup(
    lookup='id',
```

```
    from_=alt.LookupData(Q2, 'FIPS', ['Juvenile Arrest Rate'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Juvenile Arrest'
)
```



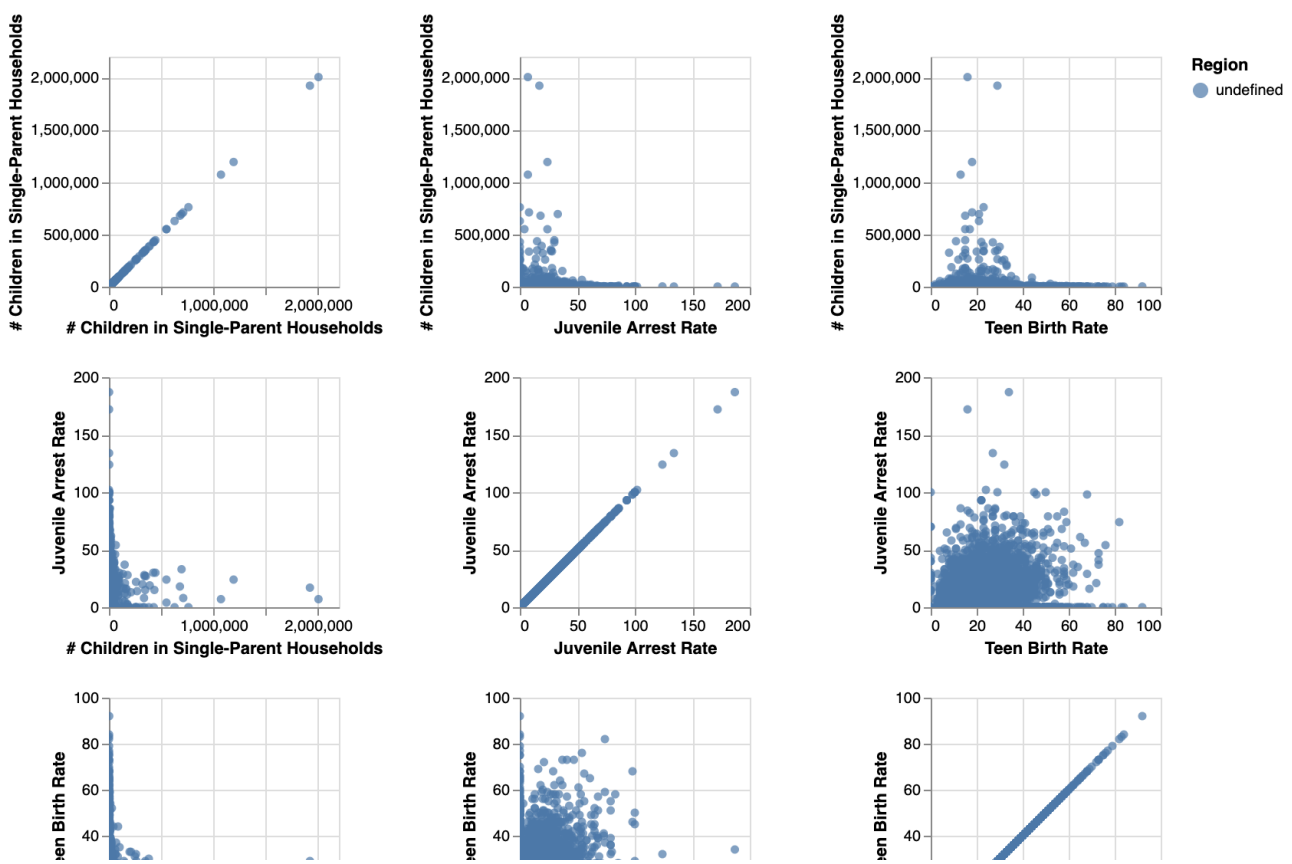

```
HeatQ2 = Q2.drop('State', axis=1)
```

```
HeatQ2v2 = HeatQ2.drop('FIPS', axis=1)
```

```
corr = HeatQ2v2.astype('float64').corr()
corr.style.background_gradient(cmap='coolwarm')
```

	# Children in Single-Parent Households	Juvenile Arrest Rate	Teen Birth Rate
# Children in Single-Parent Households	1.000000	-0.005805	-0.044308
Juvenile Arrest Rate	-0.005805	1.000000	0.067069

```
import altair
alt.Chart(Q2).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='Region:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['# Children in Single-Parent Households', 'Juvenile Arrest Rate', 'Teen Birth Rate'],
    column=['# Children in Single-Parent Households', 'Juvenile Arrest Rate', 'Teen Birth Rate']
).interactive()
```



Q2: When searching for a connection between the number of single parent households and the and the teen birth rate, there appears to be a connection. The top five states for teen birthrate also appear to have a high percentage of single parent households.

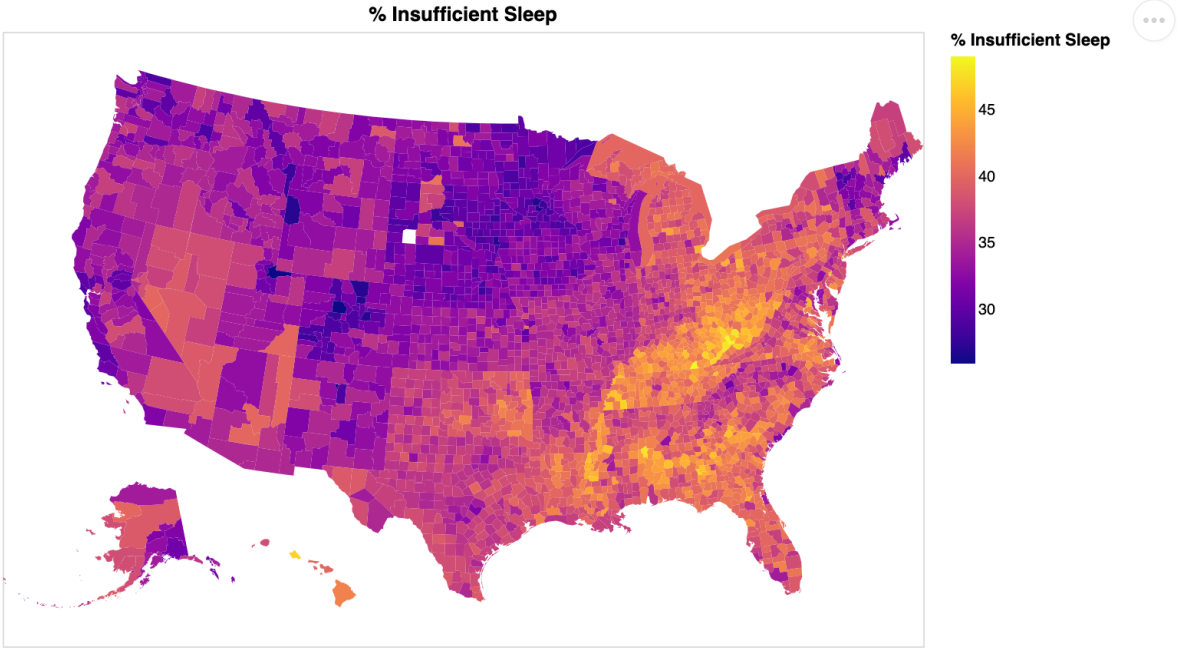
Q3.) Are there connections between areas that have higher rates of insufficient sleep and lower graduation rate/ Disconnected youth?

```
States = pd.DataFrame()
States  = df2['State']
FIPS = pd.DataFrame()
FIPS = df2['FIPS']
IS = pd.DataFrame()
IS = df2['% Insufficient Sleep']
HSG = pd.DataFrame()
HSG = df2['High School Graduation Rate']
Disconnected = pd.DataFrame()
Disconnected = df2['% Disconnected Youth']


Combine3 = pd.DataFrame()
Combine3 = [FIPS, States, IS, HSG, Disconnected]


Q3 = pd.concat(Combine3, axis=1)


Q3 = Q3.fillna(0)


Q3
```

	FIPS	State	% Insufficient Sleep	High School Graduation Rate	% Disconnected Youth
1	1000	Alabama	40	92	8
2	1001	Alabama	38	89	0
3	1003	Alabama	36	90	6
4	1005	Alabama	41	85	0
5	1007	Alabama	40	92	0
...	...	...	...	...	...
3189	56037	Wyoming	35	80.0	0
3190	56039	Wyoming	27	93.0	0
3191	56041	Wyoming	36	83.0	0
3192	56043	Wyoming	32	87.0	0
3193	56045	Wyoming	34	0	0

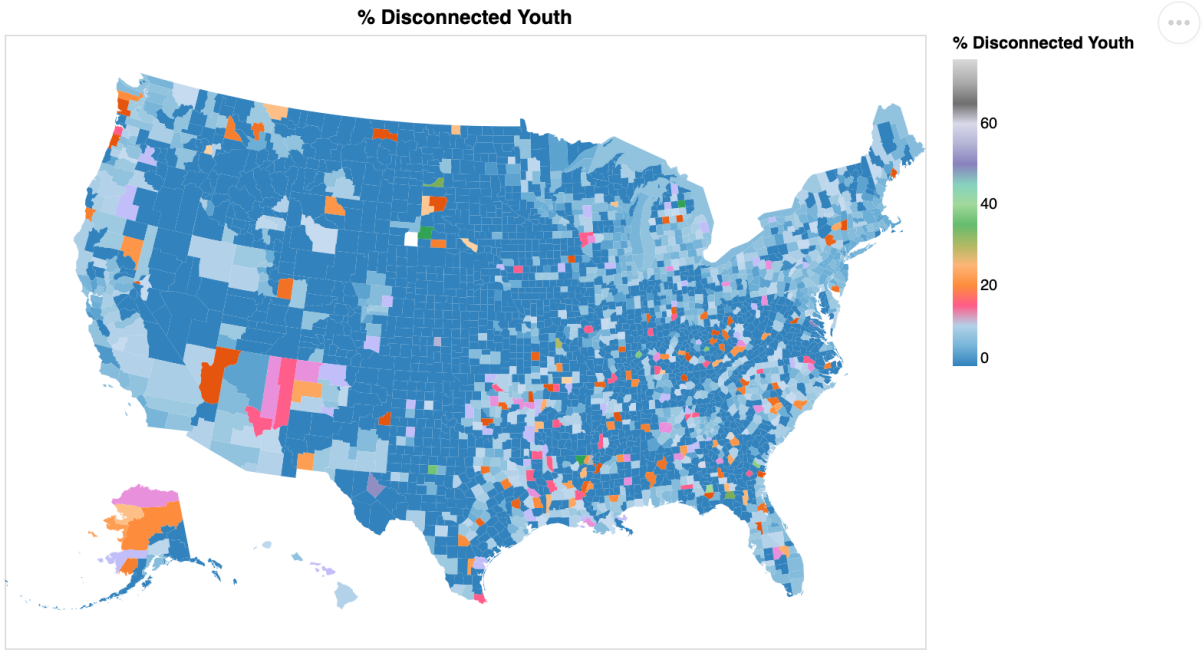
- Visualization for the % Insufficient Sleep used in Q3

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('% Insufficient Sleep:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q3, 'FIPS', ['% Insufficient Sleep'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Insufficient Sleep'
)
```



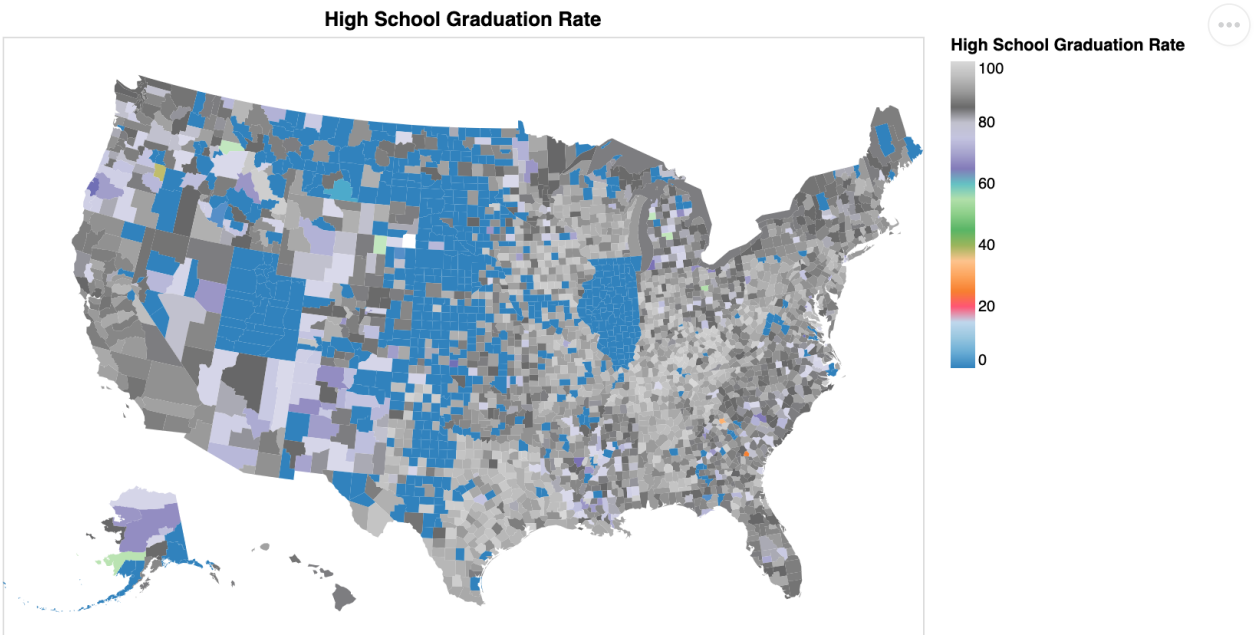
- Visualization for the % of disconnected youth used in Q3

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('% Disconnected Youth:Q', scale=alt.Scale(scheme="category20c"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q3, 'FIPS', ['% Disconnected Youth'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Disconnected Youth'
)
```



- Visualization for the Highschool graduation rate used in Q3

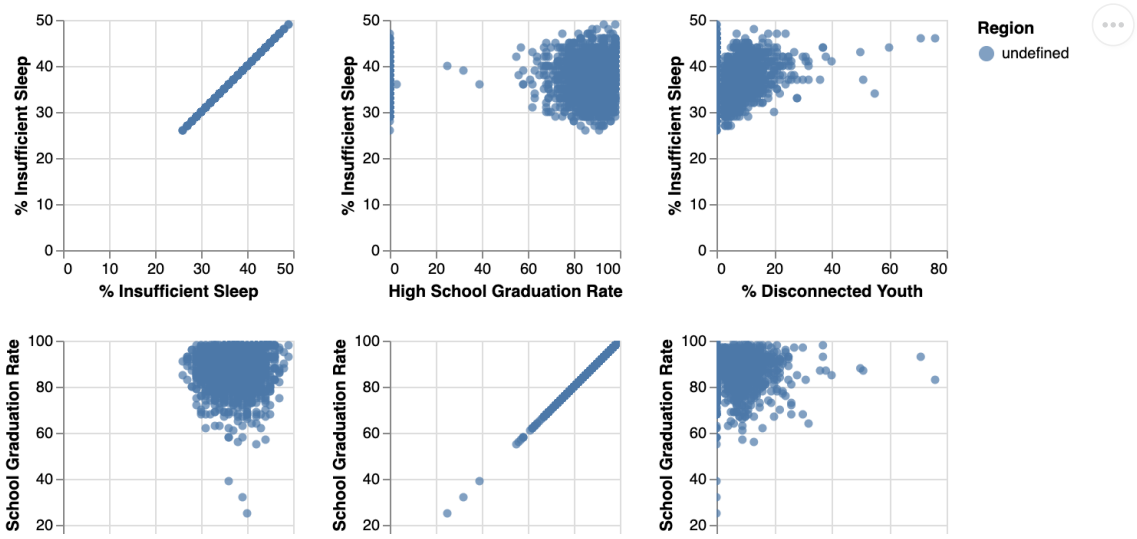
```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('High School Graduation Rate:Q', scale=alt.Scale(scheme="category20c"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q3, 'FIPS', ['High School Graduation Rate'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='High School Graduation Rate'
)
```




```
HeatQ3 = Q3.drop('State', axis=1)


HeatQ3v2 = HeatQ3.drop('FIPS', axis=1)


corr = HeatQ3v2.astype('float64').corr()
corr.style.background_gradient(cmap='coolwarm')
```

	% Insufficient Sleep	High School Graduation Rate	% Disconnected Youth
% Insufficient Sleep	1.000000	0.273422	0.183197
High School Graduation Rate	0.273422	1.000000	0.187159
% Disconnected Youth	0.183197	0.187159	1.000000

```
import altair
alt.Chart(Q3).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='Region:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['% Insufficient Sleep', 'High School Graduation Rate', '% Disconnected Youth'],
    column=['% Insufficient Sleep', 'High School Graduation Rate', '% Disconnected Youth']
).interactive()
```



Q3: It appears that there isn't any direct correlation between the percentage of insufficient sleep and the high school graduation rate because the areas that reported the highest level of sleep inefficiency, also have the highest graduation rate in the country, but the graduation rate is high for majority of the country. There does appear to be some correlation between sleep inefficiency and the percentage of disconnected youth. This can be seen in numerous counties in the states of New Mexico, Nevada, Oklahoma, Texas, Florida, and Michigan.

Q4.) Can a connection be made between the area with a higher number of physically inactive days and the limited access to healthy foods/ frequent physical distress?

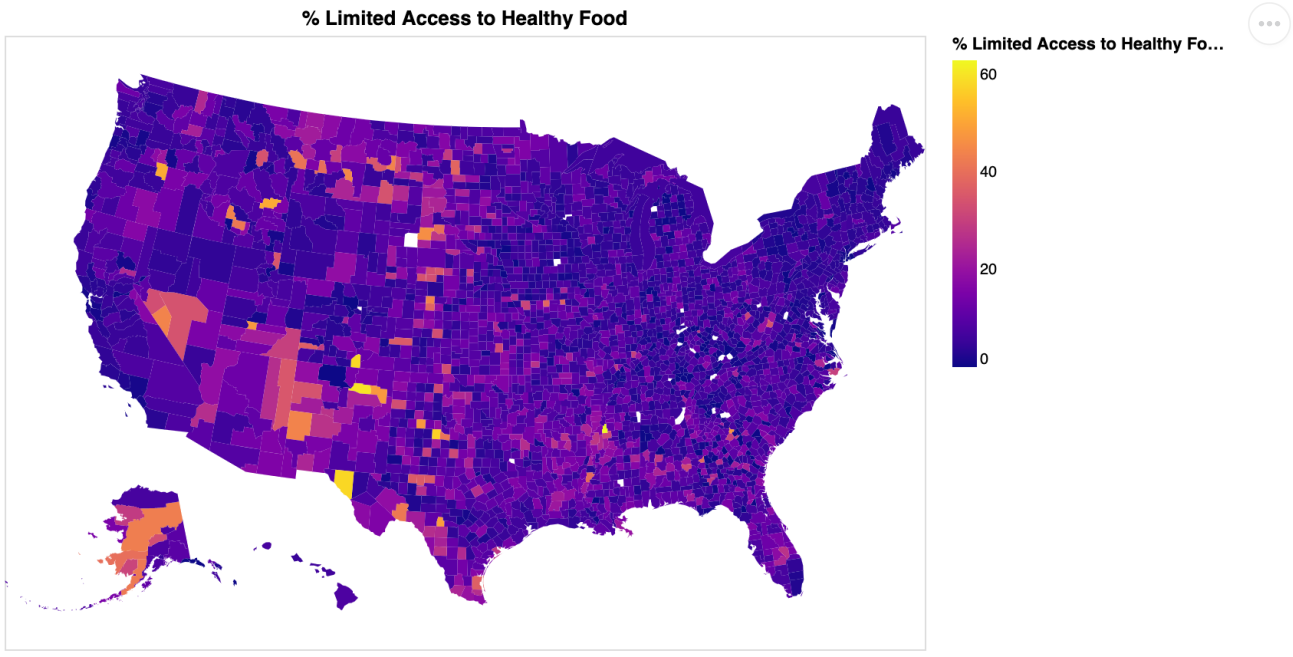
```
States = pd.DataFrame()
States  = df2['State']
FIPS = pd.DataFrame()
FIPS = df2['FIPS']
PI = pd.DataFrame()
PI = df1['% Physically Inactive']
LAHF = pd.DataFrame()
LAHF = df2['% Limited Access to Healthy Foods']
PD = pd.DataFrame()
PD = df2['% Frequent Physical Distress']


Combine4 = pd.DataFrame()
Combine4 = [FIPS, States, PI, LAHF, PD]


Q4 = pd.concat(Combine4, axis=1)


Q4.fillna(0)
```

	FIPS	State	% Physically Inactive	% Limited Access to Healthy Foods	% Frequent Physical Distress
1	1000	Alabama	31	9	15
2	1001	Alabama	32	13	14
3	1003	Alabama	28	8	13
4	1005	Alabama	42	10	19
5	1007	Alabama	38	0	16

- Visualization for the % of limited access to healthy foods used in Q4

3190	56039	Wyoming	19	4.0	9

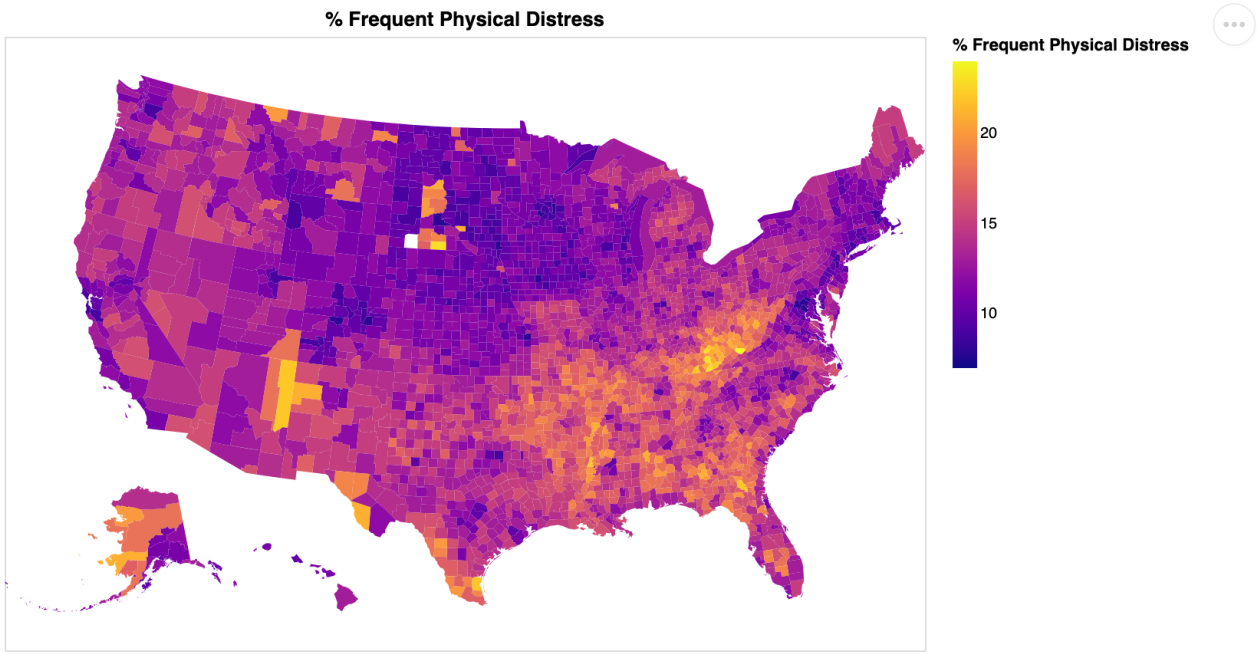
```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('% Limited Access to Healthy Foods:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q4, 'FIPS', ['% Limited Access to Healthy Foods'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Limited Access to Healthy Food'
)
```



- Visualization for the % Frequent Physical Distress used in Q4

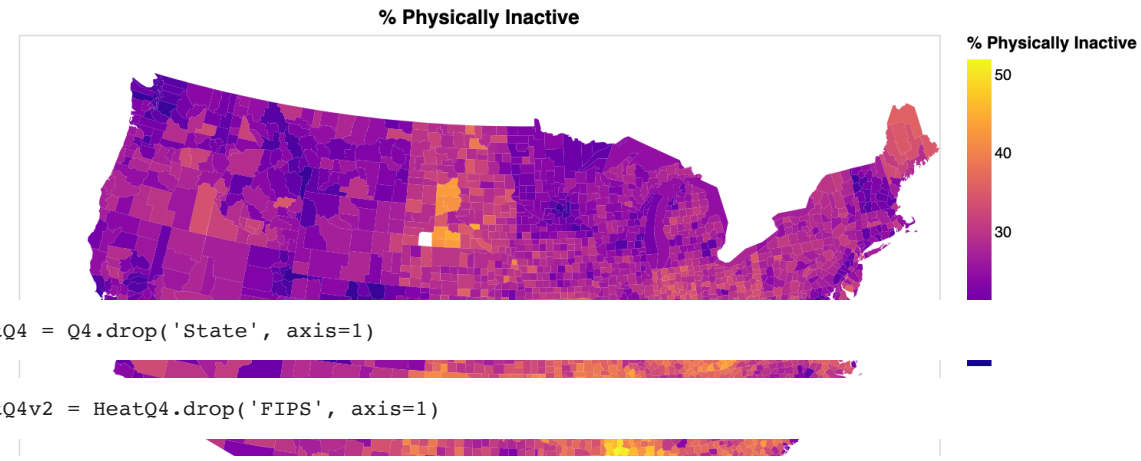
```
counties = alt.topo_feature(data.us_10m.url, feature='counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('% Frequent Physical Distress:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
```

```
    lookup='id',
    from_=alt.LookupData(Q4, 'FIPS', ['% Frequent Physical Distress'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Frequent Physical Distress'
)
```



- Visualization for the % Physically Inactive used in Q4

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('% Physically Inactive:Q', scale=alt.Scale(scheme="plasma"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q4, 'FIPS', ['% Physically Inactive'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='% Physically Inactive'
)
```



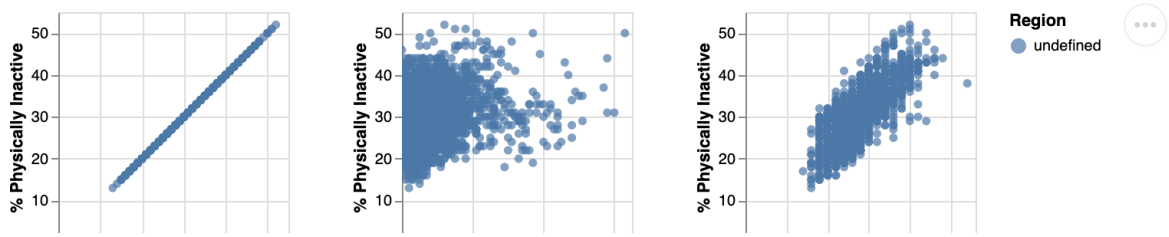

```
HeatQ4 = Q4.drop('State', axis=1)
```

```
HeatQ4v2 = HeatQ4.drop('FIPS', axis=1)
```

```
corr = HeatQ4v2.astype('float64').corr()
corr.style.background_gradient(cmap='coolwarm')
```

	% Physically Inactive	% Limited Access to Healthy Foods	% Frequent Physical Distress
% Physically Inactive	1.000000	0.179787	0.781401
% Limited Access to Healthy Foods	0.179787	1.000000	0.168335
% Frequent Physical Distress	0.781401	0.168335	1.000000

```
import altair
alt.Chart(Q4).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='Region:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['% Physically Inactive', '% Limited Access to Healthy Foods', '% Frequent Physical Distress'],
    column=['% Physically Inactive', '% Limited Access to Healthy Foods', '% Frequent Physical Distress']
).interactive()
```



Q4: There didn't appear to be a correlation between access to healthy food and either of the two categories. but there did appear to be a direct correlation between the percentage of physically inactive people and the percentage of frequent physical distress. To the point where the maps look nearly identical. This is most prominently shown in the states of Texas, Mississippi, Oklahoma, Georgia, and West Virginia.

Q5.) Are there any connections between higher suicide rates and the areas that score lower for home ownership?

```
States = pd.DataFrame()
States  = df2['State']
FIPS = pd.DataFrame()
FIPS = df2['FIPS']
PI = pd.DataFrame()
suicides = df3['# Deaths']
Homeowners = pd.DataFrame()
Homeowners = df2['# Homeowners']
```

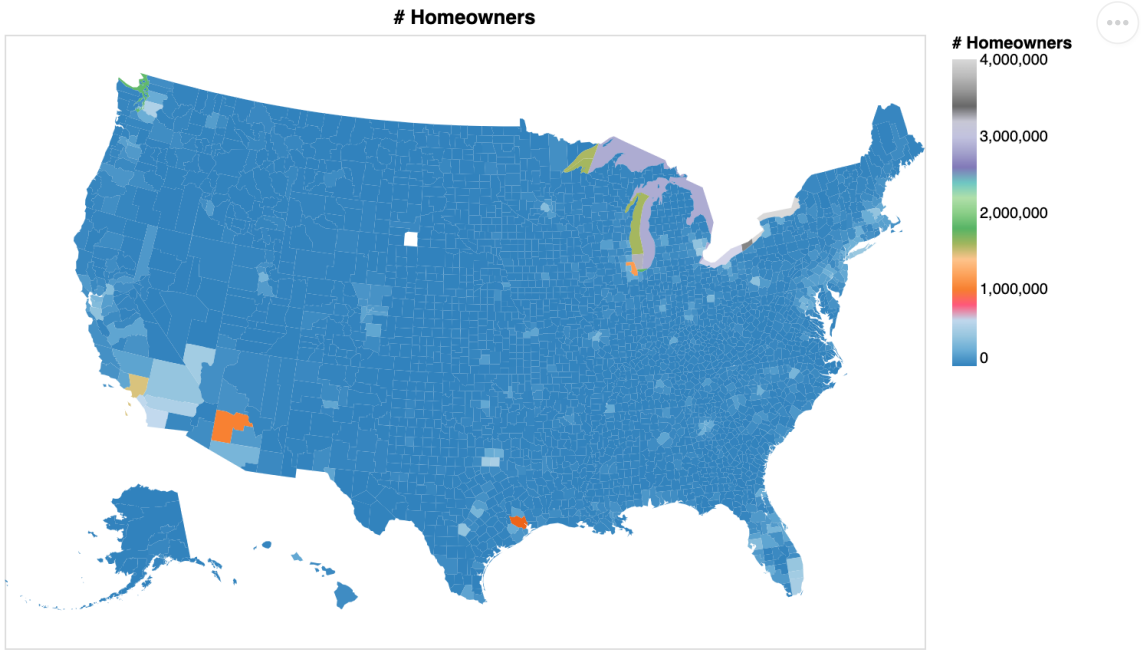
```
Combine5 = pd.DataFrame()
Combine5 = [FIPS, States, suicides, Homeowners]


Q5 = pd.concat(Combine5, axis=1)


Q5 = Q5.fillna(0)


Q5
```

	FIPS	State	# Deaths	# Homeowners
1	1000	Alabama	4043	1306505
2	1001	Alabama	52	16088
3	1003	Alabama	214	64716
4	1005	Alabama	20	5775
5	1007	Alabama	18	5428
...	...	...	...	...
3189	56037	Wyoming	0	11852
3190	56039	Wyoming	0	5609
3191	56041	Wyoming	0	5996
3192	56043	Wyoming	0	2510
3193	56045	Wyoming	0	2419

3193 rows × 4 columns

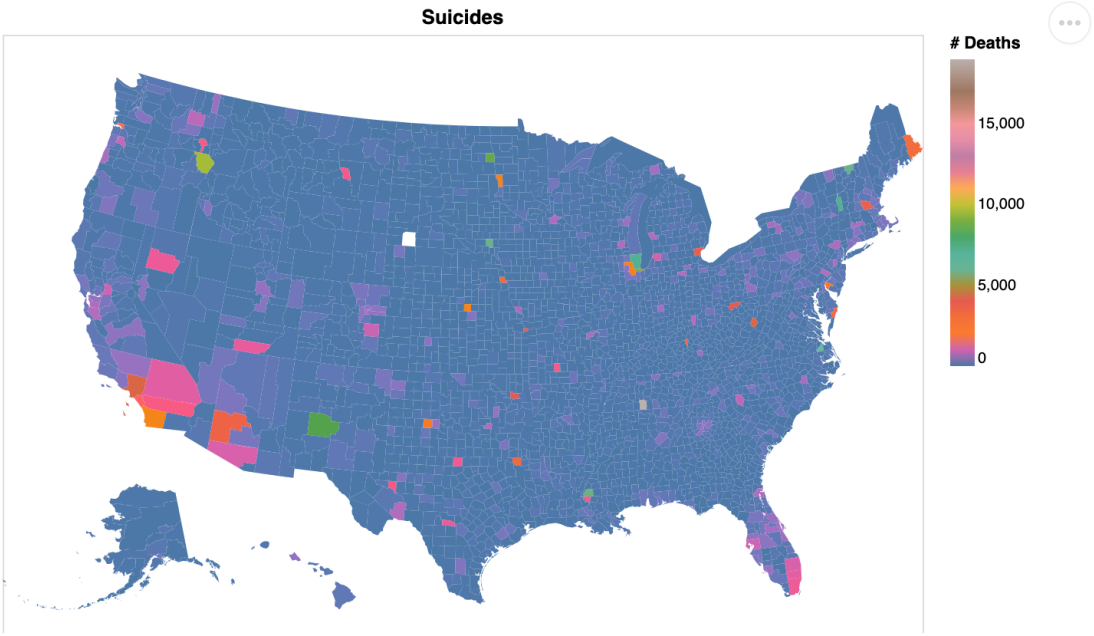
- Visualization for the number of Homeowners used in Q5

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('# Homeowners:Q', scale=alt.Scale(scheme="category20c"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q5, 'FIPS', ['# Homeowners'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='# Homeowners'
)
```



- Visualization for the number of suicides used in Q5

```
counties = alt.topo_feature(data.us_10m.url, 'counties')

alt.Chart(counties).mark_geoshape(
).encode(
    color=alt.Color('# Deaths:Q', scale=alt.Scale(scheme="tableau10"))
).transform_lookup(
    lookup='id',
    from_=alt.LookupData(Q5, 'FIPS', ['# Deaths'])
).project(
    type='albersUsa'
).properties(
    width=600,
    height=400,
    title='Suicides'
)
```



Q5: Similar to question one, where it might be assumed that lower homeowner rate would lead to higher suicide rates, based on the bar graphs, it appears that would be false. Looking at the lowest homeowner percent (DoC), they also have the lowest rate of suicide. and even places with much higher homeowner percentages like West Virginia or Louisiana still have a high suicide rate, almost twice so compared to District of Columbia. the states with the highest suicide rates (Wyoming, New Mexico, New York, and California) also have decent homeownership percentages.

```
HeatQ5 = Q5.drop('State', axis=1)
```

```
HeatQ5
```

	FIPS	# Deaths	# Homeowners
1	1000	4043	1306505
2	1001	52	16088
3	1003	214	64716
4	1005	20	5775
5	1007	18	5428
...	...	...	...
3189	56037	0	11852
3190	56039	0	5609
3191	56041	0	5996
3192	56043	0	2510
3193	56045	0	2419

3193 rows × 3 columns

```
HeatQ5v2 = HeatQ5.drop('FIPS', axis=1)
```

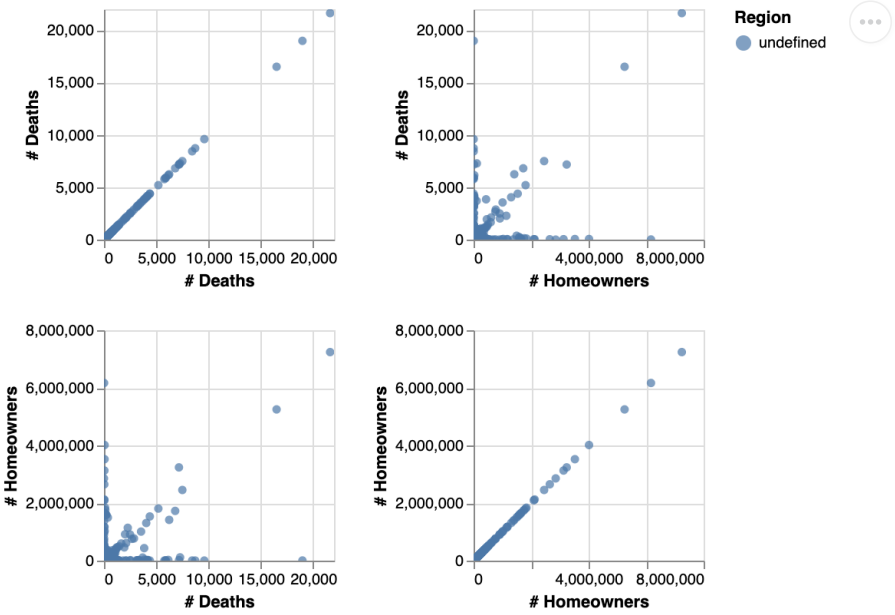
```
HeatQ5v2
```

	# Deaths	# Homeowners
1	4043	1306505
2	52	16088
3	214	64716
4	20	5775
5	18	5428
...	...	...
3189	0	11852
3190	0	5609
3191	0	5996
3192	0	2510

```
corr5 = HeatQ5v2.astype('float64').corr()
corr5.style.background_gradient(cmap='coolwarm')
```

	# Deaths	# Homeowners
# Deaths	1.000000	0.476863
# Homeowners	0.476863	1.000000

```
import altair
alt.Chart(Q5).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='Region:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['# Deaths', '# Homeowners'],
    column=['# Deaths', '# Homeowners']
).interactive()
```



Heat maps for Health behavior data.

% Broadband Access Average Number of Mentally Unhealthy Days : # Deaths : # Children in Single-Parent Households Juvenile Arrest Rate Teen Birth Rate % Insufficient Sleep High School Graduation Rate % Disconnected Youth % Physically Inactive % Limited Access to Healthy Foods % Frequent Physical Distress : # Homeowners

```
from google.colab import files
Ranked_measure = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Ranked measure sources and years.csv to Ranked measure sources and years.csv
```

```
PH = pd.DataFrame()
PH  = df1['% Fair or Poor Health']
PPH = pd.DataFrame()
PPH  = df1['Average Number of Physically Unhealthy Days']
PMHD = pd.DataFrame()
PMHD  = df1['Average Number of Mentally Unhealthy Days']
LBW = pd.DataFrame()
LBW = df1['% Low birthweight']
FPD = pd.DataFrame()
FDP = df2['% Frequent Physical Distress']
FMD = pd.DataFrame()
FMD = df2['% Frequent Mental Distress']
DP = pd.DataFrame()
DP = df2['% Adults with Diabetes']
HIV = pd.DataFrame()
HIV = df2['HIV Prevalence Rate']
```

```
FIPS = pd.DataFrame()
FIPS  = df1['FIPS']
smoking = pd.DataFrame()
smoking = df1['% Smokers']
obesity = pd.DataFrame()
obesity = df1['% Adults with Obesity']
FEI = pd.DataFrame()
FEI  = df1['Food Environment Index']
PI = pd.DataFrame()
PI  = df1['% Physically Inactive']
EO = pd.DataFrame()
EO  = df1['% With Access to Exercise Opportunities']
FI = pd.DataFrame()
FI = df2['% Food Insecure']
LAHF = pd.DataFrame()
LAHF = df2['% Limited Access to Healthy Foods']
drinking = pd.DataFrame()
drinking = df1['% Excessive Drinking']
DD = pd.DataFrame()
DD = df1['% Driving Deaths with Alcohol Involvement']
drug_mortality = pd.DataFrame()
drug_mortality = df2['Drug Overdose Mortality Rate']
STD = pd.DataFrame()
STD = df1['Chlamydia Rate']
TB = pd.DataFrame()
TB = df1['Teen Birth Rate']
```

```
PMD = pd.DataFrame()
PMD  = df1['Years of Potential Life Lost Rate']
LE = pd.DataFrame()
LE  = df2['Life Expectancy']
PMAM = pd.DataFrame()
PMAM  = df2['Age-adjusted Death Rate']
CM = pd.DataFrame()
```

```
CM = df2['Child Mortality Rate']
IM = pd.DataFrame()
IM = df2['Infant Mortality Rate']

QoL = pd.DataFrame()
QoL = [PH, PPH, PMHD, LBW, FDP, FMD, DP, HIV]


Quality_of_Life = pd.concat(QoL, axis=1)


Quality_of_Life
```

	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% Low birthweight	% Frequent Physical Distress	% Frequent Mental Distress	% Adults with Diabetes	HIV Prevalence Rate
1	21	4.8	5.6	10	15	18	12	336
2	20	4.5	5.4	10	14	17	11	222
3	17	4.2	5.2	8	13	16	9	173
4	31	5.9	6.1	12	19	21	17	498
5	25	5.2	5.8	10	16	19	13	208
...	...	...	...	...	...	...	...	...
3189	16	3.5	3.9	10	11	13	8	67.0
3190	12	2.8	3.3	8	9	11	7	54.0
3191	17	3.8	4.3	10	12	14	8	NaN
3192	17	3.7	4.0	7	12	13	9	0.0

```
LoL = pd.DataFrame()
LoL = [PMD, LE, PMAM, CM, IM]

Length_of_Life = pd.concat(LoL, axis=1)

Length_of_Life
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate
1	10350	74.8	500	67	8
2	8027	76.6	427	51	7
3	8118	77.7	382	51	5
4	12877	72.9	590	83	NaN
5	11191	73.6	568	82	NaN
...	...	...	...	...	...
3189	8419	76.5	420.0	43.0	NaN
3190	3283	86.7	133.0	NaN	NaN
3191	9358	77.0	414.0	52.0	NaN
3192	7074	78.8	370.0	NaN	NaN
3193	5149	80.7	263.0	NaN	NaN

3193 rows × 5 columns

```
hdf = pd.DataFrame()
hdf = [FIPS, smoking, obesity, FEI, PI, EO, FI, LAHF, drinking, DD, drug_mortality, STD, TB]


Health = pd.concat(hdf, axis=1)


Health
```

	FIPS	% Smokers	% Adults with Obesity	Food Environment Index	% Physically Inactive	% With Access to Exercise Opportunities	% Food Insecure	% Limited Access to Healthy Foods	% Excessive Drinking	% Drivi Deaths wi Alcoh Involvem
1	1000	21	36	5.3	31	57	16	9	15	
2	1001	20	35	6.5	32	63	16	13	16	
3	1003	20	30	7.4	28	75	13	8	22	
4	1005	28	40	5.7	42	50	21	10	14	
5	1007	25	41	7.6	38	11	16	0	16	
...	...	...	...	...	...	...	...	...	...	
3189	56037	17	34	8.1	25	83	11	5.0	18	
3190	56039	11	22	8.7	19	96	9	4.0	19	
3191	56041	19	30	8.3	28	79	13	1.0	18	
3192	56043	18	29	8.3	26	76	12	3.0	19	
3193	56045	19	33	7.6	27	28	12	9.0	19	

3193 rows × 13 columns

```
frames = pd.DataFrame()
frames = [Health, Quality_of_Life, Length_of_Life]


frames = pd.concat(frames, axis=1)
```

Show how health behaviors relate to the Length of life and Quality of life. >>>

Health behaviors:

1.) Tobacco use:

- Adult smoking (RMD)

2.) Diet & Excercise:

- Adult obesity (RMD)
- Food environement index (RMD)
- Physical inactivity (RMD)
- Access to excesise opportunities (RMD)
- Food insecurity (ADD)
- Limited access to health foods (ADD)

3.) Alcohol and drug use:

- Excessive drinking (RMD)
- Alcohol impaired driving deaths (RMD)
- Drug overdose death (ADD)

4.) Sexual activity:

- Sexually transmitted infections (RMD)
- Teen Births (RMD)

Length of Life:

- Premature Death (RMD)
- Life expectancy (ADD)
- Premature Age adjusted mortality (ADD)
- child mortality (ADD)
- infant mortality (ADD)

quality of life:

- Poor or fair health (RMD)
- Poor physical health days (RMD)
- Poor mental health days (RMD)
- Low birthweight (RMD)
- Frequent physical distress (ADD)
- Frequent mental distress (ADD)
- Diabetes prevalence (ADD)
- HIV prevalence (ADD)

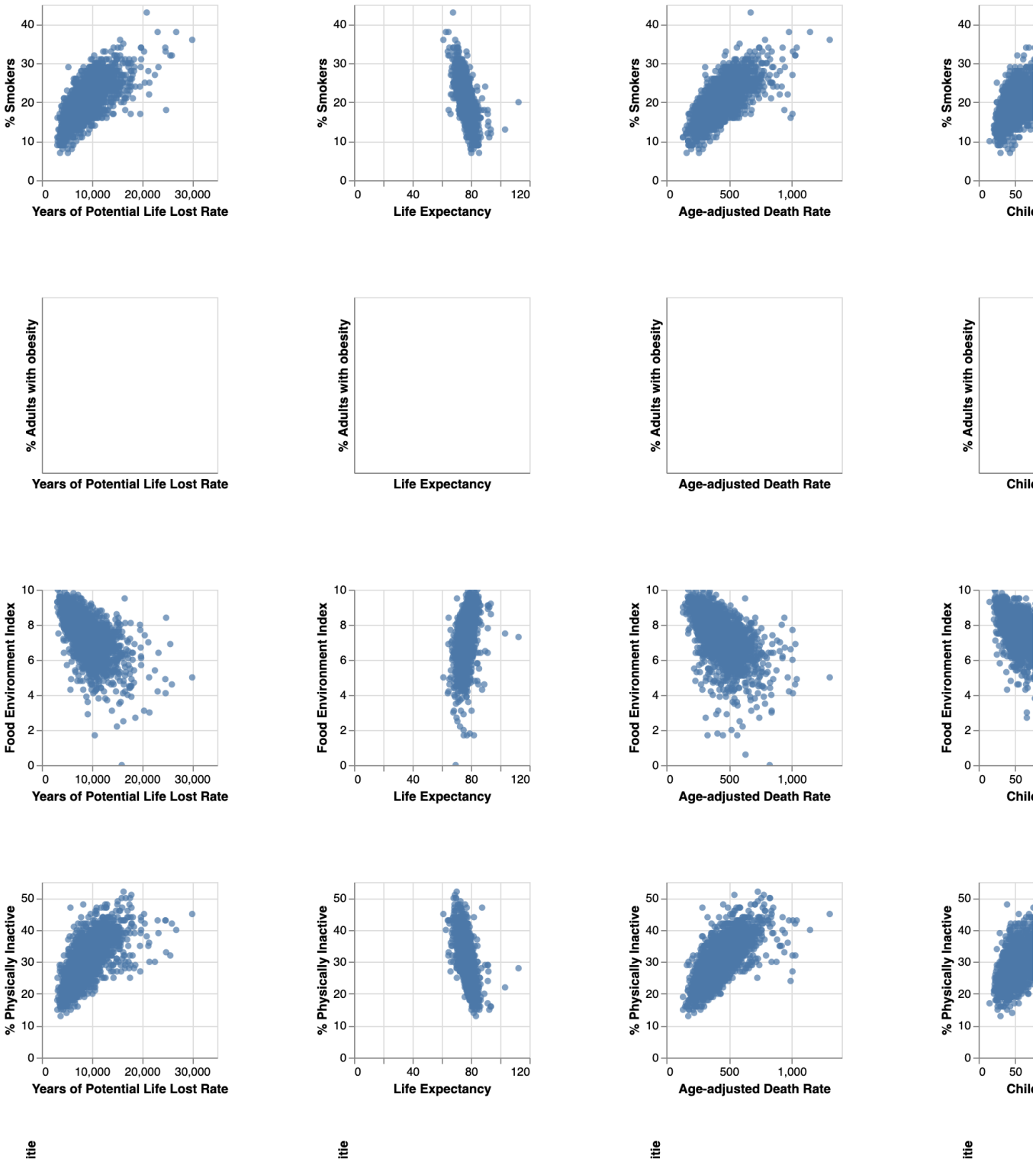
```
frames
```

	FIPS	% Smokers	% Adults with Obesity	Food Environment Index	% Physically Inactive	% With Access to Exercise Opportunities	% Food Insecure	% Limited Access to Healthy Foods	% Excessive Drinking	% Drivi Deaths wi Alcoh Involvem
1	1000	21	36	5.3	31	57	16	9	15	2
2	1001	20	35	6.5	32	63	16	13	16	3
3	1003	20	30	7.4	28	75	13	8	22	3
4	1005	28	40	5.7	42	50	21	10	14	3
5	1007	25	41	7.6	38	11	16	0	16	2
...	...	...	...	...	...	...	...	...	...	
3189	56037	17	34	8.1	25	83	11	5.0	18	
3190	56039	11	22	8.7	19	96	9	4.0	19	
3191	56041	19	30	8.3	28	79	13	1.0	18	
3192	56043	18	29	8.3	26	76	12	3.0	19	
3193	56045	19	33	7.6	27	28	12	9.0	19	2

3193 rows × 26 columns

```
import altair
alt.Chart(frames).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='Region:N'
).properties(
    width=150,
    height=150
).repeat(
```

```
        row=['% Smokers', '% Adults with obesity', 'Food Environment Index', '% Physically Inactive', '% With Access
        column=['Years of Potential Life Lost Rate', 'Life Expectancy', 'Age-adjusted Death Rate', 'Child Mortality R
    ).interactive()
```


% Smokers
% Adults with obesity
Food Environment Index
% Physically Inactive
Years of Potential Life Lost Rate
Life Expectancy
Age-adjusted Death Rate
Chil
itie

```
correlation = frames.astype('float64').corr()
correlation.style.background_gradient(cmap='plasma')
```

	FIPS	% Smokers	% Adults with Obesity	Food Environment Index	% Physically Inactive	% With Access to Exercise Opportunities	% Food Insecure	% Limited Access to Healthy Foods	% Excessive Drinking
FIPS	1.000000	-0.069257	0.084473	0.072706	-0.028312	-0.008461	-0.099242	-0.013708	0.04214(
% Smokers	-0.069257	1.000000	0.657633	-0.476277	0.715990	-0.453057	0.675482	0.076112	-0.39423
% Adults with Obesity	0.084473	0.657633	1.000000	-0.468492	0.789756	-0.410438	0.562581	0.170422	-0.43064
Food Environment Index	0.072706	-0.476277	-0.468492	1.000000	-0.558536	0.327792	-0.774284	-0.775558	0.44979
% Physically Inactive	-0.028312	0.715990	0.789756	-0.558536	1.000000	-0.412819	0.696945	0.179787	-0.60841
% With Access to Exercise Opportunities	-0.008461	-0.453057	-0.410438	0.327792	-0.412819	1.000000	-0.292389	-0.224916	0.16997
% Food Insecure	-0.099242	0.675482	0.562581	-0.774284	0.696945	-0.292389	1.000000	0.216171	-0.59694
% Limited Access to Healthy Foods	-0.013708	0.076112	0.170422	-0.775558	0.179787	-0.224916	0.216171	1.000000	-0.10468

social and economic factors>>>

% Driving

```
Education = pd.DataFrame()
FIPS  = df1['FIPS']
HSC = pd.DataFrame()
HSC = df1['% Completed High School']
Some_College = pd.DataFrame()
Some_College = df1['% Some College']
Grad = pd.DataFrame()
Grad = df2['High School Graduation Rate']
DY = pd.DataFrame()
DY  = df2['% Disconnected Youth']
RS = pd.DataFrame()
RS = df2['Average Grade Performance']
MS = pd.DataFrame()
MS = df2['Average Grade Performance']
SI = pd.DataFrame()
SI = df2['Segregation index']
SF = pd.DataFrame()
SF = df2['Spending per-pupil']
Employment = pd.DataFrame()
Employment = df1['% Unemployed']
CIP = pd.DataFrame()
CIP = df1['% Children in Poverty']
GPG = pd.DataFrame()
GPG = df2['Gender Pay Gap']
MHI = pd.DataFrame()
MHI = df2['Median Household Income']
RL = pd.DataFrame()
RL = df2['% Enrolled in Free or Reduced Lunch']
CSPH = pd.DataFrame()
CSPH = df1['% Children in Single-Parent Households']
SA = pd.DataFrame()
SA = df1['Social Association Rate']
CCC = pd.DataFrame()
```

```
CCC = df2['% household income required for childcare expenses']
ID = pd.DataFrame()
ID = df1['Injury Death Rate']
HR = pd.DataFrame()
HR = df2['Homicide Rate']
SR = pd.DataFrame()
SR = df2['Suicide Rate (Age-Adjusted)']
FF = pd.DataFrame()
FF = df2['Firearm Fatalities Rate']
MVF = pd.DataFrame()
MVF = df2['Motor Vehicle Mortality Rate']
JA = pd.DataFrame()
JA = df2['Juvenile Arrest Rate']
```

Social & Economic Factors:

- % Completed High School
- % Some College
- High School Graduation Rate
- % Disconnected Youth
- Average Grade Performance
- Segregation index
- Spending per-pupil
- % Unemployed
- % Children in Poverty
- Gender Pay Gap
- Median Household Income
- % Enrolled in Free or Reduced Lunch
- % Children in Single-Parent Households
- Social Association Rate
- % household income required for childcare expenses
- Injury Death Rate
- Homicide Rate
- Suicide Rate (Age-Adjusted)
- Firearm Fatalities Rate
- Motor Vehicle Mortality Rate
- Juvenile Arrest Rate

```
Education = [FIPS, HSC, Some_College, Grad, DY, RS, MS, SI, SF, Employment, CIP, GPG, MHI, RL, CSPH, SA, RS, CCC,
```

```
Education = pd.concat(Education, axis=1)
```

```
Educationv2 = Education.drop('FIPS', axis=1)
```

```
Educationv2 = Educationv2.fillna(0)
```

```
Educationv2
```

	% Completed High School	% Some College	High School Graduation Rate	% Disconnected Youth	Average Grade Performance	Average Grade Performance	Average Grade Performance	Average Grade Performance	Segreg
1	87	62	92	8	2.9	2.7	2.9	2.7	
2	89	61	89	0	3.1	2.9	3.1	2.9	
3	90	66	90	6	3.2	3.0	3.2	3.0	
4	75	38	85	0	2.4	2.0	2.4	2.0	
5	81	38	92	0	2.7	2.6	2.7	2.6	
...	...	...	...	...	...	...	...	...	
3189	93	62	80.0	0	3.3	3.4	3.3	3.4	
3190	95	76	93.0	0	3.5	3.6	3.5	3.6	

Length of life DataFrame >>>

- Years of Potential Life Lost Rate
- Life Expectancy
- Age-adjusted Death Rate
- Child Mortality Rate
- Infant Mortality Rate

```
LoL = pd.DataFrame()
premature_death = pd.DataFrame()
premature_death = df1['Years of Potential Life Lost Rate']
LE = pd.DataFrame()
LE = df2['Life Expectancy']
PMAAM = pd.DataFrame()
PMAAM  = df2['Age-adjusted Death Rate']
CMR = pd.DataFrame()
CMR = df2['Child Mortality Rate']
IMR = pd.DataFrame()
IMR = df2['Infant Mortality Rate']


LoL = [premature_death, LE, PMAAM, CMR, IMR]


LoL = pd.concat(LoL, axis=1)


LoL = LoL.fillna(0)


LoL
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate
1	10350	74.8	500	67	8
2	8027	76.6	427	51	7
3	8118	77.7	382	51	5
4	12877	72.9	590	83	0
5	11191	73.6	568	82	0
...	...	...	...	...	...

Quality of Life DataFrame >>>

- % Fair or Poor Health
- Average Number of Physically Unhealthy Days
- Average Number of Mentally Unhealthy Days
- % Low birthweight
- % Frequent Physical Distress
- % Frequent Mental Distress
- % Adults with Diabetes
- HIV Prevalence Rate

```
QoL = pd.DataFrame()
POFH = pd.DataFrame()
POFH = df1['% Fair or Poor Health']
PPHD = pd.DataFrame()
PPHD = df1['Average Number of Physically Unhealthy Days']
PMHD = pd.DataFrame()
PMHD  = df1['Average Number of Mentally Unhealthy Days']
LBW = pd.DataFrame()
LBW = df1['% Low birthweight']
FPD = pd.DataFrame()
FPD = df2['% Frequent Physical Distress']
FMD = pd.DataFrame()
FMD = df2['% Frequent Mental Distress']
DP = pd.DataFrame()
DP = df2['% Adults with Diabetes']
HIVP = pd.DataFrame()
HIVP = df2['HIV Prevalence Rate']


QoL = [POFH, PPHD, PMHD, LBW, FPD, FMD, DP, HIVP]


QoL = pd.concat(QoL, axis=1)


QoL = QoL.fillna(0)


QoL
```

	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% Low birthweight	% Frequent Physical Distress	% Frequent Mental Distress	% Adults with Diabetes	HIV Prevalence Rate
1	21	4.8	5.6	10	15	18	12	336
2	20	4.5	5.4	10	14	17	11	222
3	17	4.2	5.2	8	13	16	9	173
4	31	5.9	6.1	12	19	21	17	498
5	25	5.2	5.8	10	16	19	13	208
...	...	...	...	...	...	...	...	...

Compare Social & Economic factors with Health outcomes >>>

```
Health_outcomes = pd.DataFrame()

Health_outcomes = [LoL, QoL]

Health_outcomes = pd.concat(Health_outcomes, axis=1)

Health_outcomes
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% Low birthweight	Frequ Physi Distr
1	10350	74.8	500	67	8	21	4.8	5.6	10	
2	8027	76.6	427	51	7	20	4.5	5.4	10	
3	8118	77.7	382	51	5	17	4.2	5.2	8	
4	12877	72.9	590	83	0	31	5.9	6.1	12	
5	11191	73.6	568	82	0	25	5.2	5.8	10	
...	...	...	...	...	...	...	...	...	...	
3189	8419	76.5	420.0	43.0	0	16	3.5	3.9	10	
3190	3283	86.7	133.0	0	0	12	2.8	3.3	8	
3191	9358	77.0	414.0	52.0	0	17	3.8	4.3	10	
3192	7074	78.8	370.0	0	0	17	3.7	4.0	7	
3193	5149	80.7	263.0	0	0	16	3.7	4.2	7	

3193 rows × 13 columns

```
comparison = pd.DataFrame()

comparison = [Health_outcomes, Educationv2]

comparison = pd.concat(comparison, axis=1)

comparison
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% Low birthweight	Frequ Physi Distr
1	10350	74.8	500	67	8	21	4.8	5.6	10	
2	8027	76.6	427	51	7	20	4.5	5.4	10	
3	8118	77.7	382	51	5	17	4.2	5.2	8	
4	12877	72.9	590	83	0	31	5.9	6.1	12	
5	11191	73.6	568	82	0	25	5.2	5.8	10	
...	...	...	...	...	...	...	...	...	...	
3189	8419	76.5	420.0	43.0	0	16	3.5	3.9	10	
3190	3283	86.7	133.0	0	0	12	2.8	3.3	8	
3191	9358	77.0	414.0	52.0	0	17	3.8	4.3	10	
3192	7074	78.8	370.0	0	0	17	3.7	4.0	7	
3193	5149	80.7	263.0	0	0	16	3.7	4.2	7	

3193 rows × 40 columns

Remove some of the duplicated columns here... May have dropped the Math GPA scores and one of the segregation indexes but all other factors are retained. >>>

```
comparison = comparison.loc[:,~comparison.columns.duplicated(keep='first')]
```

```
comparison
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% Low birthweight	Frequ Physi Distr
1	10350	74.8	500	67	8	21	4.8	5.6	10	
2	8027	76.6	427	51	7	20	4.5	5.4	10	
3	8118	77.7	382	51	5	17	4.2	5.2	8	
4	12877	72.9	590	83	0	31	5.9	6.1	12	
5	11191	73.6	568	82	0	25	5.2	5.8	10	
...	...	...	...	...	...	...	...	...	...	
3189	8419	76.5	420.0	43.0	0	16	3.5	3.9	10	
3190	3283	86.7	133.0	0	0	12	2.8	3.3	8	
3191	9358	77.0	414.0	52.0	0	17	3.8	4.3	10	
3192	7074	78.8	370.0	0	0	17	3.7	4.0	7	
3193	5149	80.7	263.0	0	0	16	3.7	4.2	7	

3193 rows × 34 columns

```
comparison.columns
```

```
Index(['Years of Potential Life Lost Rate', 'Life Expectancy',
       'Age-adjusted Death Rate', 'Child Mortality Rate',
       'Infant Mortality Rate', '% Fair or Poor Health',
       'Average Number of Physically Unhealthy Days',
       'Average Number of Mentally Unhealthy Days', '% Low birthweight',
       '% Frequent Physical Distress', '% Frequent Mental Distress',
       '% Adults with Diabetes', 'HIV Prevalence Rate',
       '% Completed High School', '% Some College',
       'High School Graduation Rate', '% Disconnected Youth',
       'Average Grade Performance', 'Segregation index', 'Spending per-pupil',
       '% Unemployed', '% Children in Poverty', 'Gender Pay Gap',
       'Median Household Income', '% Enrolled in Free or Reduced Lunch',
       '% Children in Single-Parent Households', 'Social Association Rate',
       '% household income required for childcare expenses',
       'Injury Death Rate', 'Homicide Rate', 'Suicide Rate (Age-Adjusted)',
       'Firearm Fatalities Rate', 'Motor Vehicle Mortality Rate',
       'Juvenile Arrest Rate'],
      dtype='object')
```

Comparison for individual Social & Economic factors alongside Health outcomes

```
correlation = comparison.astype('float64').corr()
correlation.style.background_gradient(cmap='plasma')
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% L birthweig
Years of Potential Life Lost Rate	1.000000	0.107594	0.820175	0.432530	0.205612	0.595631	0.635405	0.613982	0.5826
Life Expectancy	0.107594	1.000000	0.072992	0.061262	0.072285	-0.127414	-0.112605	-0.075248	0.2742
Age-adjusted Death Rate	0.820175	0.072992	1.000000	0.314978	0.118939	0.713201	0.725927	0.672344	0.5896
Child Mortality Rate	0.432530	0.061262	0.314978	1.000000	0.536158	0.241308	0.253624	0.269453	0.3429
Infant Mortality Rate	0.205612	0.072285	0.118939	0.536158	1.000000	0.053306	0.045120	0.084340	0.2579
% Fair or Poor Health	0.595631	-0.127414	0.713201	0.241308	0.053306	1.000000	0.900129	0.727508	0.4950
Average Number of Physically Unhealthy Days	0.635405	-0.112605	0.725927	0.253624	0.045120	0.900129	1.000000	0.912782	0.4646
Average Number of Mentally Unhealthy Days	0.613982	-0.075248	0.672344	0.269453	0.084340	0.727508	0.912782	1.000000	0.4677
% Low birthweight	0.582698	0.274219	0.589611	0.342942	0.257906	0.495054	0.464687	0.467723	1.0000
% Frequent Physical Distress	0.636944	-0.117036	0.734901	0.251850	0.042414	0.919389	0.987620	0.889029	0.4722
% Frequent Mental Distress	0.622119	-0.096289	0.703120	0.252075	0.056256	0.765070	0.927604	0.979581	0.4619
% Adults with Diabetes	0.569732	-0.120104	0.679171	0.264167	0.122143	0.929416	0.775598	0.595705	0.5356
HIV Prevalence Rate	0.248725	0.034988	0.209060	0.294839	0.318705	0.243559	0.148496	0.116389	0.4217
% Completed High School	-0.418995	0.084217	-0.520139	-0.140934	0.010472	-0.853422	-0.698225	-0.503439	-0.3673
% Some College	-0.458983	0.067136	-0.556450	-0.114603	0.105584	-0.766441	-0.734277	-0.618321	-0.3249
High School Graduation Rate	0.345220	0.251779	0.148420	0.349903	0.267229	0.129394	0.162020	0.223555	0.2793
% Disconnected Youth	0.208708	0.028786	0.157417	0.305768	0.344764	0.159987	0.169404	0.145551	0.1512
Average Grade Performance	0.069372	0.354407	0.050500	-0.020726	0.001957	-0.081827	-0.039679	-0.011626	0.0993

Segregation

Social & Economic factors

- % Completed High School
- % Some College
- High School Graduation Rate
- % Disconnected Youth
- Average Grade Performance (Reading)
- Segregation index
- Spending per-pupil
- % Unemployed
- % Children in Poverty
- Gender Pay Gap
- Median Household Income
- % Enrolled in Free or Reduced Lunch
- % Children in Single-Parent Households
- Social Association Rate
- % household income required for childcare expenses
- Injury Death Rate
- Homicide Rate
- Suicide Rate (Age-Adjusted)
- Firearm Fatalities Rate
- Motor Vehicle Mortality Rate
- Juvenile Arrest Rate

---

isolate the highest scoring correlations here (>= .8) >>>

Rate

```
filtered = comparison.astype('float64').corr()
filtered = filtered[((filtered >= .9) & (filtered != 1.0))]
filtered.style.background_gradient(cmap='plasma')
```

```
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/style.py:3931: RuntimeWarning: All-NaN slice encou
  smin = np.nanmin(gmap) if vmin is None else vmin
/usr/local/lib/python3.10/dist-packages/pandas/io/formats/style.py:3932: RuntimeWarning: All-NaN slice encou
  smax = np.nanmax(gmap) if vmax is None else vmax
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% Lo birthweigh
Years of Potential Life Lost Rate	nan	nan	nan	nan	nan	nan	nan	nan	na
Life Expectancy	nan	nan	nan	nan	nan	nan	nan	nan	na
Age-adjusted Death Rate	nan	nan	nan	nan	nan	nan	nan	nan	na
Child Mortality Rate	nan	nan	nan	nan	nan	nan	nan	nan	na
Infant Mortality Rate	nan	nan	nan	nan	nan	nan	nan	nan	na
% Fair or Poor Health	nan	nan	nan	nan	nan	nan	0.900129	nan	na
Average Number of Physically Unhealthy Days	nan	nan	nan	nan	nan	0.900129	nan	0.912782	na
Average Number of Mentally Unhealthy Days	nan	nan	nan	nan	nan	nan	0.912782	nan	na
% Low birthweight	nan	nan	nan	nan	nan	nan	nan	nan	na
% Frequent Physical Distress	nan	nan	nan	nan	nan	0.919389	0.987620	nan	na
% Frequent Mental Distress	nan	nan	nan	nan	nan	nan	0.927604	0.979581	na
% Adults with Diabetes	nan	nan	nan	nan	nan	0.929416	nan	nan	na

Does A higher level of poverty and

isolate lowest negative correlations >>>

```
filtered2 = comparison.astype('float64').corr()
filtered2 = filtered2[((filtered2 <= -0.6) & (filtered2 != 1.0))]
filtered2.style.background_gradient(cmap='plasma')
```

	Years of Potential Life Lost Rate	Life Expectancy	Age-adjusted Death Rate	Child Mortality Rate	Infant Mortality Rate	% Fair or Poor Health	Average Number of Physically Unhealthy Days	Average Number of Mentally Unhealthy Days	% L birthweig
Years of Potential Life Lost Rate	nan	nan	nan	nan	nan	nan	nan	nan	n
Life Expectancy	nan	nan	nan	nan	nan	nan	nan	nan	n
Age-adjusted Death Rate	nan	nan	nan	nan	nan	nan	nan	nan	n
Child Mortality Rate	nan	nan	nan	nan	nan	nan	nan	nan	n
Infant Mortality Rate	nan	nan	nan	nan	nan	nan	nan	nan	n
% Fair or Poor Health	nan	nan	nan	nan	nan	nan	nan	nan	n
Average Number of Physically Unhealthy Days	nan	nan	nan	nan	nan	nan	nan	nan	n
Average Number of Mentally Unhealthy Days	nan	nan	nan	nan	nan	nan	nan	nan	n
% Low birthweight	nan	nan	nan	nan	nan	nan	nan	nan	n
% Frequent Physical Distress	nan	nan	nan	nan	nan	nan	nan	nan	n
% Frequent Mental Distress	nan	nan	nan	nan	nan	nan	nan	nan	n
% Adults with Diabetes	nan	nan	nan	nan	nan	nan	nan	nan	n
HIV Prevalence Rate	nan	nan	nan	nan	nan	nan	nan	nan	n
% Completed									

Above are the highest scoring Social & Economic factors that correlate to health outcomes

Quality of Life factors:

- % Fair or Poor Health
- Average Number of Physically Unhealthy Days
- Average Number of Mentally Unhealthy Days
- % Low birthweight
- % Frequent Physical Distress
- % Frequent Mental Distress
- % Adults with Diabetes
- HIV Prevalence Rate

Length of Life factors:

- Years of Potential Life Lost Rate
- Life Expectancy
- Age-adjusted Death Rate
- Child Mortality Rate
- Infant Mortality Rate

Social & Economic factors

- % Completed High School
- % Some College
- High School Graduation Rate
- % Disconnected Youth
- Average Grade Performance (Reading)
- Segregation index
- Spending per-pupil
- % Unemployed
- % Children in Poverty
- Gender Pay Gap
- Median Household Income
- % Enrolled in Free or Reduced Lunch
- % Children in Single-Parent Households
- Social Association Rate
- % household income required for childcare expenses
- Injury Death Rate
- Homicide Rate
- Suicide Rate (Age-Adjusted)
- Firearm Fatalities Rate
- Motor Vehicle Mortality Rate
- Juvenile Arrest Rate

Predicitve analysis questions

- Does a higher rate of children in poverty predict higher levels of frequent physical distress resulting in negative health behaviors?

Predict the Alcohol-impaired Driving Deaths & Drug overdose Deaths based on the % of Children in Poverty

imports for predicitve modeling >>>

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import numpy as np
```

```
from google.colab import files
rankings = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving rankings.csv to rankings.csv
```

```
import io
df4 = pd.read_csv(io.BytesIO(rankings['rankings.csv']))
```

```
social_df = pd.DataFrame({
    '% Children in Poverty': comparison['% Children in Poverty'],
    'Excessive Drinking': frames['% Excessive Drinking'],
    '% Frequent Physical Distress': comparison['% Frequent Physical Distress'],
    'Alcohol-impaired Driving Deaths': frames['% Driving Deaths with Alcohol Involvement'],
    'Drug overdose Deaths': frames['Drug Overdose Mortality Rate'],
    'Quality of Life': df4['Quality of Life'],
    'Length of Life': df4['Length of Life']

})
```

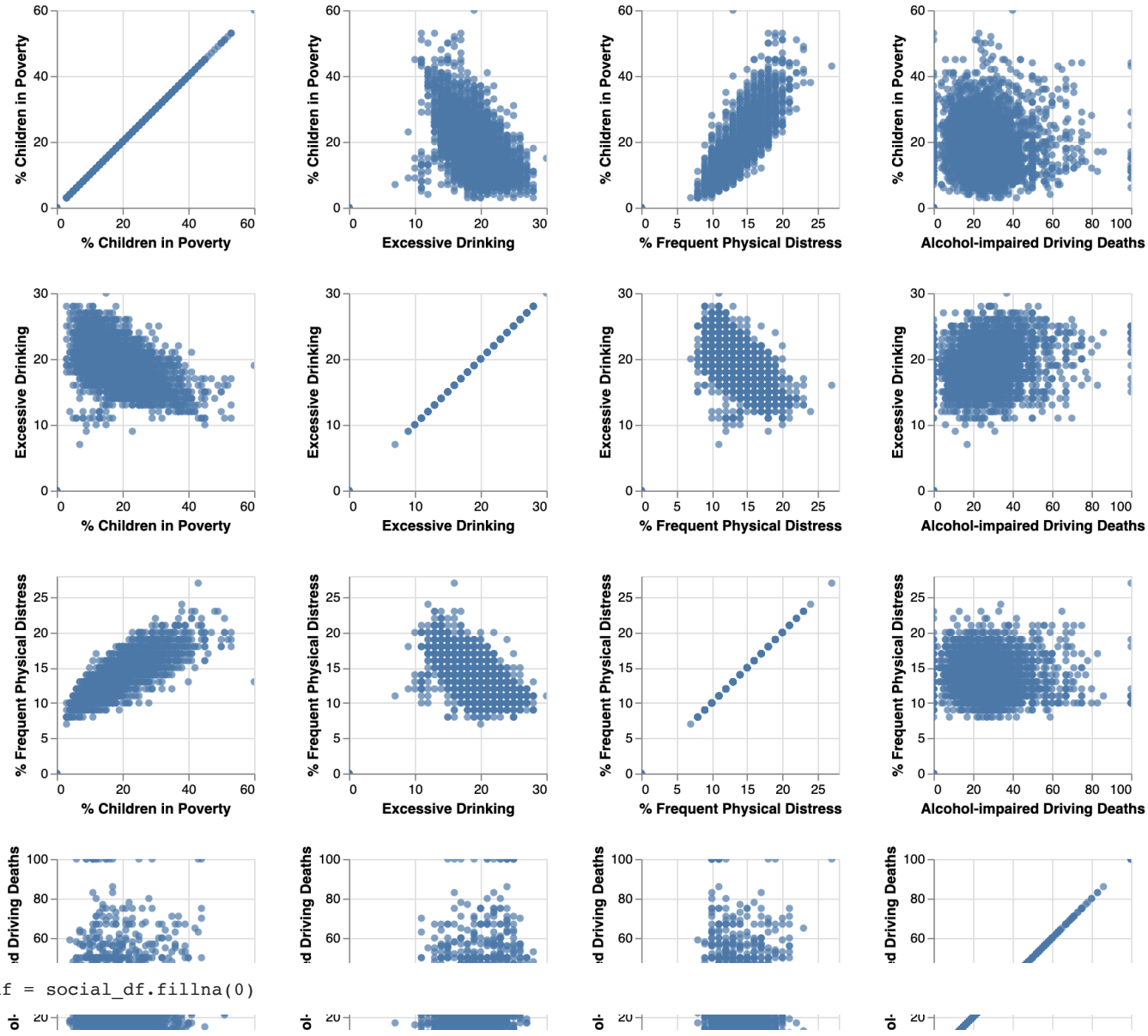
```
social_df = social_df.drop(labels=0, axis=0)
```

```
social_df.loc[social_df['% Children in Poverty'] == 'NR'] = 0
social_df.loc[social_df['% Frequent Physical Distress'] == 'NR'] = 0
social_df.loc[social_df['Excessive Drinking'] == 'NR'] = 0
social_df.loc[social_df['Alcohol-impaired Driving Deaths'] == 'NR'] = 0
social_df.loc[social_df['Drug overdose Deaths'] == 'NR'] = 0
social_df.loc[social_df['Quality of Life'] == 'NR'] = 0
social_df.loc[social_df['Length of Life'] == 'NR'] = 0
```

```
correlation = social_df.astype('float64').corr()
correlation.style.background_gradient(cmap='plasma')
```

	% Children in Poverty	Excessive Drinking	% Frequent Physical Distress	Alcohol-impaired Driving Deaths	Drug overdose Deaths	Quality of Life	Length of Life
% Children in Poverty	1.000000	-0.213769	0.797436	0.028882	0.318019	0.533977	0.470571
Excessive Drinking	-0.213769	1.000000	-0.062544	0.295877	0.096194	-0.094073	-0.049363
% Frequent Physical Distress	0.797436	-0.062544	1.000000	0.088185	0.418536	0.461563	0.389409

```
import altair
alt.Chart(social_df).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
    color='Region:N'
).properties(
    width=150,
    height=150
).repeat(
    row=['% Children in Poverty', 'Excessive Drinking', '% Frequent Physical Distress', 'Alcohol-impaired Driving
    column=['% Children in Poverty', 'Excessive Drinking', '% Frequent Physical Distress', 'Alcohol-impaired Driv
).interactive()
```




```
soc_df = social_df.fillna(0)
```

```
soc_df
```

	% Children in Poverty	Excessive Drinking	% Frequent Physical Distress	Alcohol-impaired Driving Deaths	Drug overdose Deaths	Quality of Life	Length of Life
1	21	15	15	26	17	0	0
2	15	16	14	32	8	8	4
3	12	22	13	32	18	2	5
4	38	14	19	38	0	60	52
5	22	16	16	21	24	35	31
...	...	...	...	...	...	...	...
3189	8	18	11	18	22.0	18	17
3190	4	19	9	18	0	1	1
3191	10	18	12	17	0	22	20
3192	11	19	12	0	0	6	8
3193	13	19	11	22	0	8	3

3193 rows × 7 columns

Predicitve modeling question >>>

- With a higher rate of children in poverty relating to levels of frequent physical distress, does this connection predict heightened negative health behaviors and outcomes?

Predict the Alcohol-impaired Driving Deaths & Drug overdose Deaths based on the % of Children in Poverty

Linear Regression Model for the % Children in Poverty and Alcohol-impaired Driving Deaths >>>

```
soc_df.loc[soc_df['% Children in Poverty'] == 'NR'] = 0
soc_df.loc[soc_df['% Frequent Physical Distress'] == 'NR'] = 0
soc_df.loc[soc_df['Excessive Drinking'] == 'NR'] = 0
soc_df.loc[soc_df['Alcohol-impaired Driving Deaths'] == 'NR'] = 0
soc_df.loc[soc_df['Drug overdose Deaths'] == 'NR'] = 0
soc_df.loc[soc_df['Quality of Life'] == 'NR'] = 0
soc_df.loc[soc_df['Length of Life'] == 'NR'] = 0


scaler = preprocessing.MinMaxScaler()
scaled_df = scaler.fit_transform(soc_df.to_numpy())


scaled_df = pd.DataFrame(scaled_df, columns=[
    '% Children in Poverty', 'Excessive Drinking', '% Frequent Physical Distress', 'Alcohol-impaired Driving Deat
])


soc_df = scaled_df


soc_df_X = soc_df['% Children in Poverty']
soc_df_y = soc_df['Alcohol-impaired Driving Deaths']


# Split the data into training/testing sets
X_train = soc_df_X[:80]
X_test = soc_df_X[80:]

# Split the targets into training/testing sets
y_train = soc_df_y[:80]
y_test = soc_df_y[80:]


train_test_split(X_train, y_train, test_size=0.20, random_state=3000)

    [43    0.633333
     56    0.450000
     66    0.483333
     50    0.500000
     77    0.300000
             ...
     26    0.283333
     53    0.733333
     46    0.500000
     1     0.250000
     51    0.466667
     Name: % Children in Poverty, Length: 64, dtype: float64,
     58    0.250000
     14    0.400000
     19    0.483333
     38    0.350000
     8     0.283333
```

```
67    0.383333
24    0.700000
70    0.100000
64    0.383333
31    0.533333
52    0.333333
63    0.283333
55    0.466667
4     0.366667
33    0.500000
74    0.000000
Name: % Children in Poverty, dtype: float64,
43    0.30
56    0.56
66    0.41
50    0.20
77    0.67
      ...
26    0.36
53    0.70
46    0.25
1     0.32
51    0.30
Name: Alcohol-impaired Driving Deaths, Length: 64, dtype: float64,
58    0.17
14    0.39
19    0.30
38    0.24
8     0.14
67    0.19
24    0.39
70    1.00
64    0.22
31    0.26
52    0.19
63    0.28
55    0.30
4     0.21
33    0.43
74    0.00
Name: Alcohol-impaired Driving Deaths, dtype: float64]
```

```
X_train = X_train.values.reshape(-1,1)
y_train = y_train.values.reshape(-1,1)
X_test = X_test.values.reshape(-1,1)
y_test = y_test.values.reshape(-1,1)


model = linear_model.LinearRegression()
model.fit(X_train, y_train)
```

▾ LinearRegression

LinearRegression()

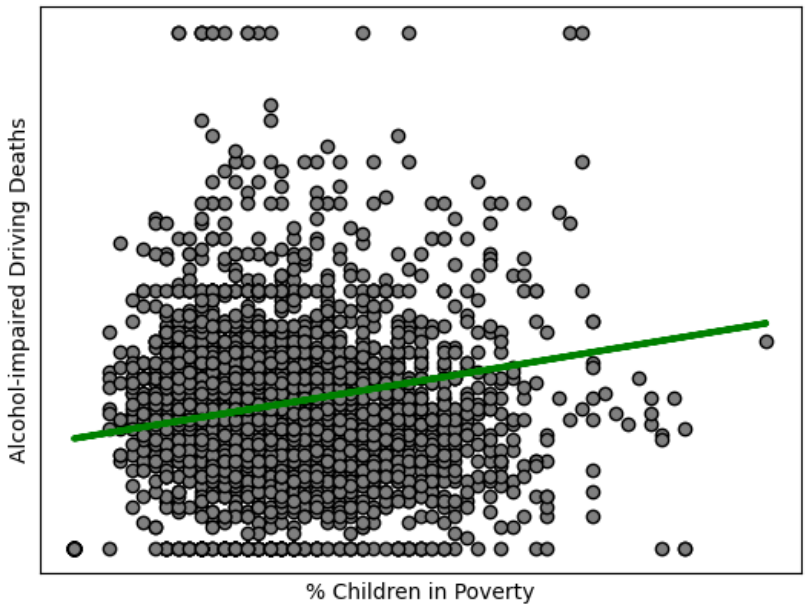
```
y_predict = model.predict(X_test)


mean_squared_error(y_test, model.predict(X_test))
```

0.023332103917226815

```
print("Coefficients: \n", model.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_predict))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test, y_predict))
plt.scatter(X_test, y_test, color='gray', edgecolors='black')
plt.plot(X_test, y_predict, color='green', linewidth=3)
plt.axis()
```

```
plt.xticks(())
plt.yticks(())
plt.xlabel('% Children in Poverty')
plt.ylabel('Alcohol-impaired Driving Deaths')
plt.show()
```

```
Coefficients:
 [[0.22338626]]
Mean squared error: 0.02
Coefficient of determination: -0.04
```



Linear regression model for the % Children in Poverty in relation to the amount of Drug Overdose Deaths

```
soc_df_X_2 = soc_df['% Children in Poverty']
soc_df_y_2 = soc_df['Drug overdose Deaths']


# Split the data into training/testing sets
X_train_2 = soc_df_X_2[:80]
X_test_2 = soc_df_X_2[80:]

# Split the targets into training/testing sets
y_train_2 = soc_df_y_2[:80]
y_test_2 = soc_df_y_2[80:]


train_test_split(X_train_2, y_train_2, test_size=0.20, random_state=3000)
```

```
[43    0.633333
 56    0.450000
 66    0.483333
 50    0.500000
 77    0.300000
         ...
 26    0.283333
 53    0.733333
 46    0.500000
 1     0.250000
 51    0.466667
 Name: % Children in Poverty, Length: 64, dtype: float64,
 58    0.250000
 14    0.400000
 19    0.483333
```

```
38    0.350000
8     0.283333
67    0.383333
24    0.700000
70    0.100000
64    0.383333
31    0.533333
52    0.333333
63    0.283333
55    0.466667
4     0.366667
33    0.500000
74    0.000000
Name: % Children in Poverty, dtype: float64,
43    0.000000
56    0.000000
66    0.000000
50    0.000000
77    0.000000
        ...
26    0.088710
53    0.000000
46    0.000000
1     0.064516
51    0.072581
Name: Drug overdose Deaths, Length: 64, dtype: float64,
58    0.225806
14    0.000000
19    0.000000
38    0.000000
8     0.153226
67    0.145161
24    0.000000
70    0.000000
64    0.217742
31    0.153226
52    0.145161
63    0.120968
55    0.000000
4     0.193548
33    0.000000
74    0.000000
Name: Drug overdose Deaths, dtype: float64]
```

```
X_train_2 = X_train_2.values.reshape(-1,1)
y_train_2 = y_train_2.values.reshape(-1,1)
X_test_2 = X_test_2.values.reshape(-1,1)
y_test_2 = y_test_2.values.reshape(-1,1)
```

```
model_2 = linear_model.LinearRegression()
model_2.fit(X_train_2, y_train_2)
```

▾ LinearRegression

LinearRegression()

```
y_predict_2 = model_2.predict(X_test_2)
```

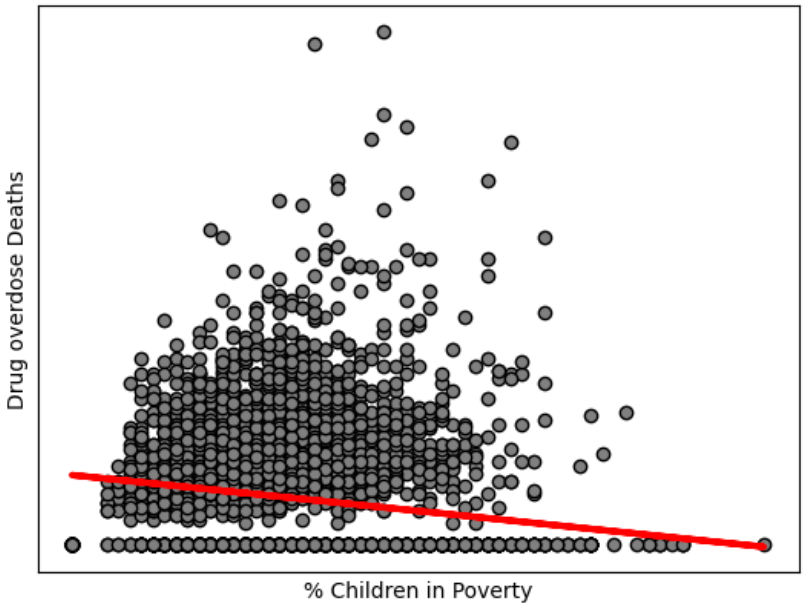
```
mean_squared_error(y_test_2, model.predict(X_test_2))
```

```
0.04413140489448729
```

```
print("Coefficients: \n", model_2.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test_2, y_predict_2))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test_2, y_predict_2))
plt.scatter(X_test_2, y_test_2, color='gray', edgecolors='black')
```

```
plt.plot(X_test_2, y_predict_2, color='red', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.xlabel('% Children in Poverty')
plt.ylabel('Drug overdose Deaths')
plt.show()
```

```
Coefficients:
 [[-0.1399004]]
Mean squared error: 0.02
Coefficient of determination: -0.07
```

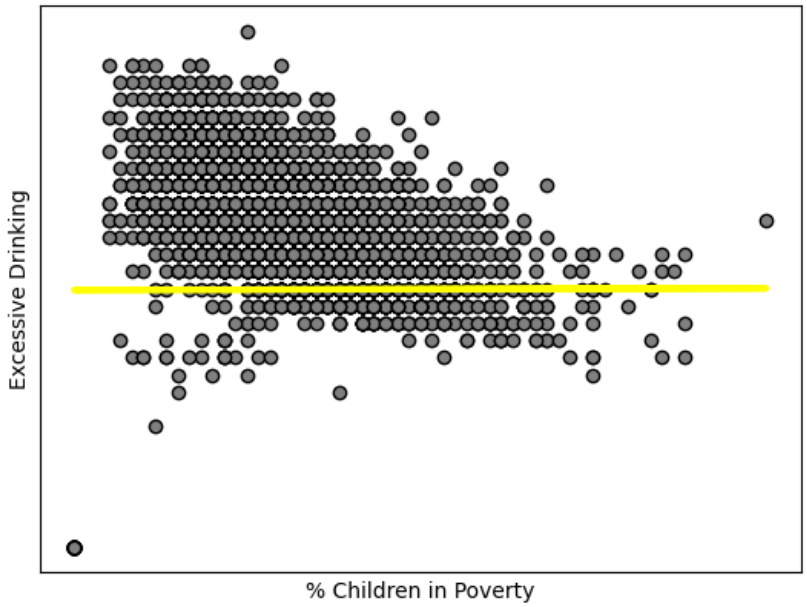


Are areas scoring higher in Frequent Physical Distress able to predict higher negative health behaviors? >>>

```
soc_df_X_3 = soc_df['% Children in Poverty']
soc_df_y_3 = soc_df['Excessive Drinking']

# Split the data into training/testing sets
X_train_3 = soc_df_X_3[:80]
X_test_3 = soc_df_X_3[80:]

# Split the targets into training/testing sets
y_train_3 = soc_df_y_3[:80]
y_test_3 = soc_df_y_3[80:]

train_test_split(X_train_3, y_train_3, test_size=0.20, random_state=3000)

# Train test split
X_train_3 = X_train_3.values.reshape(-1,1)
y_train_3 = y_train_3.values.reshape(-1,1)
X_test_3 = X_test_3.values.reshape(-1,1)
y_test_3 = y_test_3.values.reshape(-1,1)

# Model set up
model_3 = linear_model.LinearRegression()
model_3.fit(X_train_3, y_train_3)

# prediciton
y_predict_3 = model_3.predict(X_test_3)

# MSE
```

```
mean_squared_error(y_test_3, model_3.predict(X_test_3))

# Plot the model
print("Coefficients: \n", model_3.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test_3, y_predict_3))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test_3, y_predict_3))
plt.scatter(X_test_3, y_test_3, color='gray', edgecolors='black')
plt.plot(X_test_3, y_predict_3, color='yellow', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.xlabel('% Children in Poverty')
plt.ylabel('Excessive Drinking')
plt.show()
```

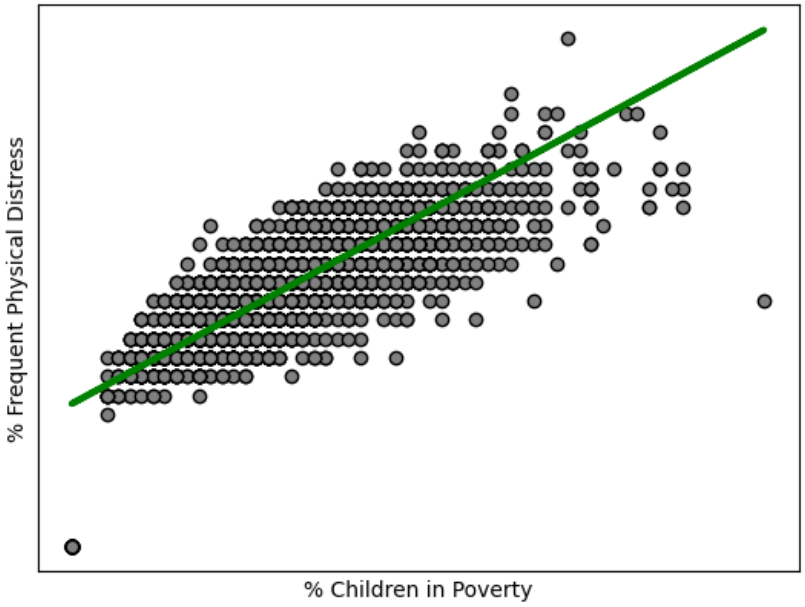
```
Coefficients:
 [[0.0032994]]
Mean squared error: 0.04
Coefficient of determination: -0.83
```




```
soc_df_X_4 = soc_df['% Children in Poverty']
soc_df_y_4 = soc_df['% Frequent Physical Distress']

# Split the data into training/testing sets
X_train_4 = soc_df_X_4[:80]
X_test_4 = soc_df_X_4[80:]

# Split the targets into training/testing sets
y_train_4 = soc_df_y_4[:80]
y_test_4 = soc_df_y_4[80:]

train_test_split(X_train_4, y_train_4, test_size=0.20, random_state=3000)

# Train test split
X_train_4 = X_train_4.values.reshape(-1,1)
y_train_4 = y_train_4.values.reshape(-1,1)
X_test_4 = X_test_4.values.reshape(-1,1)
y_test_4 = y_test_4.values.reshape(-1,1)

# Model set up
model_4 = linear_model.LinearRegression()
model_4.fit(X_train_4, y_train_4)
```

```
# prediciton
y_predict_4 = model_4.predict(X_test_4)

# MSE
mean_squared_error(y_test_4, model_4.predict(X_test_4))

# Plot the model
print("Coefficients: \n", model_4.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test_4, y_predict_4))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test_4, y_predict_4))
plt.scatter(X_test_4, y_test_4, color='gray', edgecolors='black')
plt.plot(X_test_4, y_predict_4, color='green', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.xlabel('% Children in Poverty')
plt.ylabel('% Frequent Physical Distress')
plt.show()
```

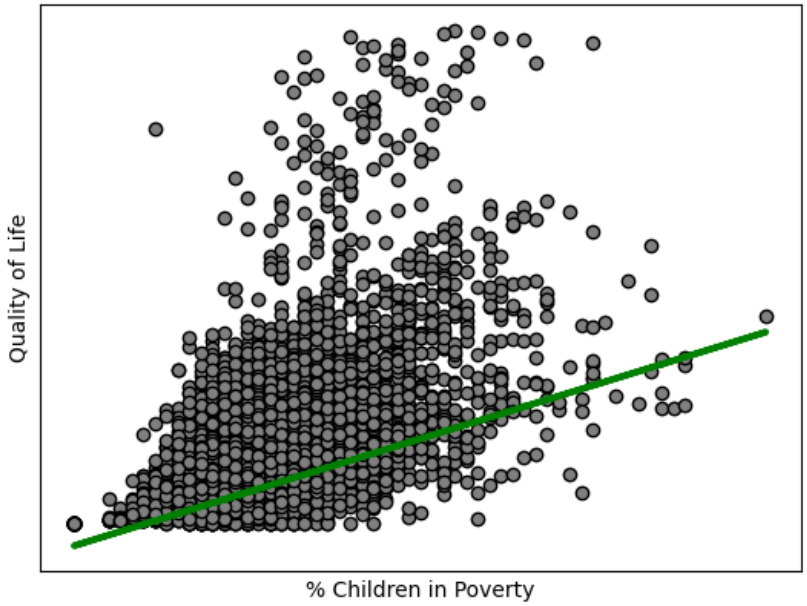
```
Coefficients:
 [[0.73517385]]
Mean squared error: 0.01
Coefficient of determination: 0.63
```




```
soc_df_X_9 = soc_df['% Children in Poverty']
soc_df_y_9 = soc_df['Quality of Life']

# Split the data into training/testing sets
X_train_9 = soc_df_X_9[:80]
X_test_9 = soc_df_X_9[80:]

# Split the targets into training/testing sets
y_train_9 = soc_df_y_9[:80]
y_test_9 = soc_df_y_9[80:]

train_test_split(X_train_9, y_train_9, test_size=0.20, random_state=1001)

# Train test split
X_train_9 = X_train_9.values.reshape(-1,1)
y_train_9 = y_train_9.values.reshape(-1,1)
X_test_9 = X_test_9.values.reshape(-1,1)
y_test_9 = y_test_9.values.reshape(-1,1)

# Model set up
model_9 = linear_model.LinearRegression()
model_9.fit(X_train_9, y_train_9)
```

```
model_9.fit(X_train_9, y_train_9)

# prediciton
y_predict_9 = model_9.predict(X_test_9)

# MSE
mean_squared_error(y_test_9, model_9.predict(X_test_9))

# Plot the model
print("Coefficients: \n", model_9.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test_9, y_predict_9))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test_9, y_predict_9))
plt.scatter(X_test_9, y_test_9, color='gray', edgecolors='black')
plt.plot(X_test_9, y_predict_9, color='green', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.xlabel('% Children in Poverty')
plt.ylabel('Quality of Life')
plt.show()
```

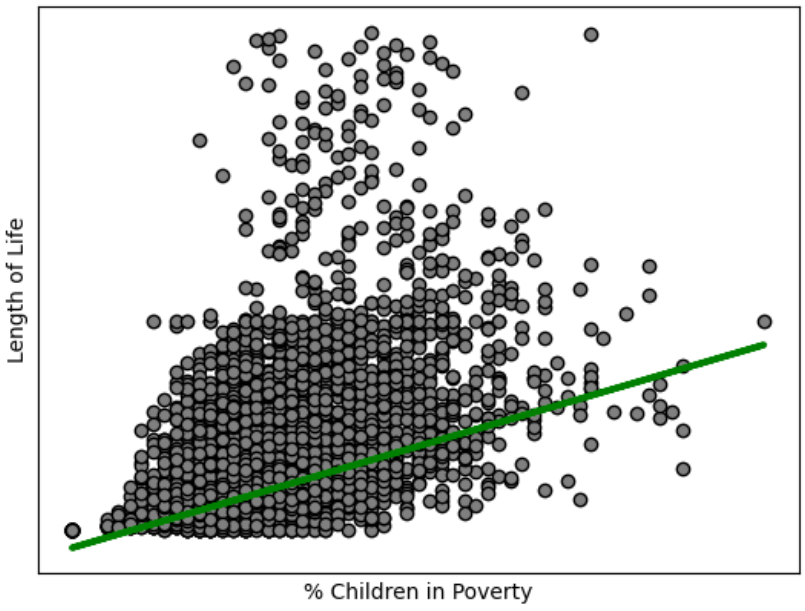
```
Coefficients:
 [[0.43382975]]
Mean squared error: 0.03
Coefficient of determination: -0.12
```




```
soc_df_X_10 = soc_df['% Children in Poverty']
soc_df_y_10 = soc_df['Length of Life']

# Split the data into training/testing sets
X_train_10 = soc_df_X_10[:80]
X_test_10 = soc_df_X_10[80:]

# Split the targets into training/testing sets
y_train_10 = soc_df_y_10[:80]
y_test_10 = soc_df_y_10[80:]

train_test_split(X_train_10, y_train_10, test_size=0.20, random_state=1001)

# Train test split
X_train_10 = X_train_10.values.reshape(-1,1)
y_train_10 = y_train_10.values.reshape(-1,1)
X_test_10 = X_test_10.values.reshape(-1,1)
y_test_10 = y_test_10.values.reshape(-1,1)
```

```
# Model set up
model_10 = linear_model.LinearRegression()
model_10.fit(X_train_10, y_train_10)

# prediciton
y_predict_10 = model_10.predict(X_test_10)

# MSE
mean_squared_error(y_test_10, model_10.predict(X_test_10))

# Plot the model
print("Coefficients: \n", model_10.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test_10, y_predict_10))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test_10, y_predict_10))
plt.scatter(X_test_10, y_test_10, color='gray', edgecolors='black')
plt.plot(X_test_10, y_predict_10, color='green', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.xlabel('% Children in Poverty')
plt.ylabel('Length of Life')
plt.show()
```

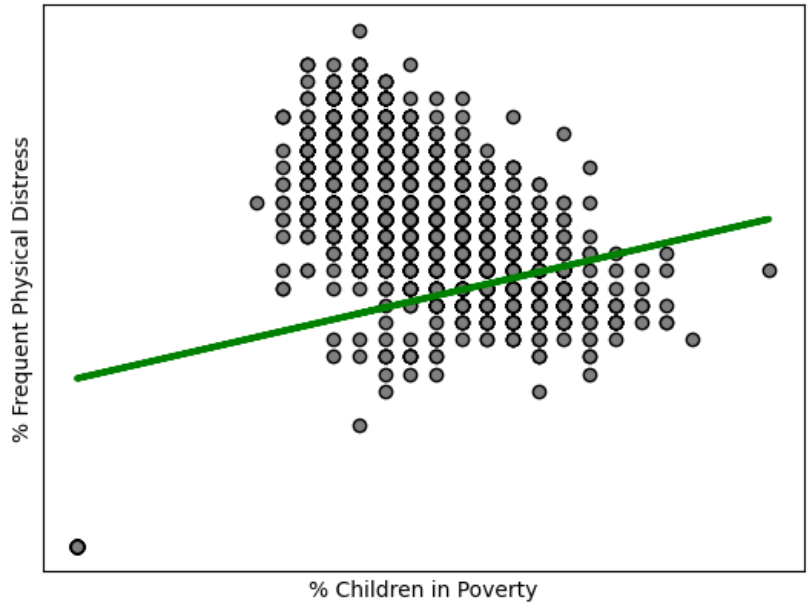
```
Coefficients:
 [[0.40804756]]
Mean squared error: 0.03
Coefficient of determination: -0.14
```




```
soc_df_X_5 = soc_df['% Frequent Physical Distress']
soc_df_y_5 = soc_df['Excessive Drinking']

# Split the data into training/testing sets
X_train_5 = soc_df_X_5[:80]
X_test_5 = soc_df_X_5[80:]

# Split the targets into training/testing sets
y_train_5 = soc_df_y_5[:80]
y_test_5 = soc_df_y_5[80:]

train_test_split(X_train_5, y_train_5, test_size=0.20, random_state=2480)

# Train test split
X_train_5 = X_train_5.values.reshape(-1,1)
y_train_5 = y_train_5.values.reshape(-1,1)
```

```
X_test_5 = X_test_5.values.reshape(-1,1)
y_test_5 = y_test_5.values.reshape(-1,1)

# Model set up
model_5 = linear_model.LinearRegression()
model_5.fit(X_train_5, y_train_5)

# prediciton
y_predict_5 = model_5.predict(X_test_5)

# MSE
mean_squared_error(y_test_5, model_5.predict(X_test_5))

# Plot the model
print("Coefficients: \n", model_5.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test_5, y_predict_5))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test_5, y_predict_5))
plt.scatter(X_test_5, y_test_5, color='gray', edgecolors='black')
plt.plot(X_test_5, y_predict_5, color='green', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.xlabel('% Children in Poverty')
plt.ylabel('% Frequent Physical Distress')
plt.show()
```

```
Coefficients:
 [[0.30889782]]
Mean squared error: 0.04
Coefficient of determination: -1.22
```




```
soc_df_X_6 = soc_df['% Frequent Physical Distress']
soc_df_y_6 = soc_df['Alcohol-impaired Driving Deaths']

# Split the data into training/testing sets
X_train_6 = soc_df_X_6[:80]
X_test_6 = soc_df_X_6[80:]

# Split the targets into training/testing sets
y_train_6 = soc_df_y_6[:80]
y_test_6 = soc_df_y_6[80:]

train_test_split(X_train_6, y_train_6, test_size=0.20, random_state=1001)

# Train test split
```