{COVID-19 Case} Surveillance Public Use Data with Geography and {Air Quality Index (AQI) }

In [1]:
```
#importing libraries
import pandas as pd
import numpy as np
```

Checking the Dataframe

In [2]:
```
#Reading the COVID dataset, Also after reading the file and seeing its properti

covid_df = pd.read_csv('/Users/vishesh/Downloads/DA/cases.csv')
```

```
/var/folders/nr/x5htffh97tvfw1rt32pzkmc00000gn/T/ipykernel_51259/4244949551.p
y:3: DtypeWarning: Columns (3) have mixed types. Specify dtype option on impor
t or set low_memory=False.
  covid_df = pd.read_csv('/Users/vishesh/Downloads/DA/cases.csv')
```

In [3]:
```
# checking all the columns name and understading the column data
covid_df.columns
```

Out[3]:
```
Index(['case_month', 'res_state', 'state_fips_code', 'res_county',
       'county_fips_code', 'age_group', 'sex', 'race', 'ethnicity',
       'case_positive_specimen_interval', 'case_onset_interval', 'process',
       'exposure_yn', 'current_status', 'symptom_status', 'hosp_yn', 'icu_yn',
       'death_yn', 'underlying_conditions_yn'],
      dtype='object')
```

In [4]:
```
# Top 5 Rows of Covid_Dataframe
covid_df.head()
```

Out[4]:

	case_month	res_state	state_fips_code	res_county	county_fips_code	age_group	sex
0	2020-12	AR	5.0	BOONE	5009.0	0 - 17 years	NaN
1	2021-10	MO	29.0	PHELPS	29161.0	0 - 17 years	NaN
2	2020-10	MO	29.0	CASS	29037.0	0 - 17 years	Female
3	2021-08	TN	47.0	CLAIBORNE	47025.0	0 - 17 years	Female
4	2020-11	GA	13.0	EFFINGHAM	13103.0	0 - 17 years	Female

{ Data Cleaning and Wrangling }

In [5]:
```
# Summary of the Covid DataFrame, including the number of non-null values in ea
```

```
covid_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37532072 entries, 0 to 37532071
Data columns (total 19 columns):
 #   Column                            Dtype
---  ------                            -----
 0   case_month                        object
 1   res_state                         object
 2   state_fips_code                   float64
 3   res_county                        object
 4   county_fips_code                  float64
 5   age_group                         object
 6   sex                               object
 7   race                              object
 8   ethnicity                         object
 9   case_positive_specimen_interval   float64
 10  case_onset_interval               float64
 11  process                           object
 12  exposure_yn                       object
 13  current_status                    object
 14  symptom_status                    object
 15  hosp_yn                           object
 16  icu_yn                            object
 17  death_yn                          object
 18  underlying_conditions_yn          object
dtypes: float64(4), object(15)
memory usage: 5.3+ GB
```

In [6]:
```
## Check for missing values in each column of the covid_df DataFrame and retur
covid_df.isna().sum()
```

Out[6]:
```
case_month                            12
res_state                            920
state_fips_code                      920
res_county                       2561284
county_fips_code                 2561284
age_group                         406634
sex                              1244276
race                             6597416
ethnicity                        7837672
case_positive_specimen_interval  25113195
case_onset_interval              20221411
process                                0
exposure_yn                            0
current_status                         0
symptom_status                         0
hosp_yn                                0
icu_yn                                 0
death_yn                         1752914
underlying_conditions_yn        35206303
dtype: int64
```

Dealing with All the Missing Values now

In [7]:
```
covid_df.iloc[1300:1320]
```

Out[7]:

	case_month	res_state	state_fips_code	res_county	county_fips_code	age_grou
1300	2020-05	MD	24.0	CAROLINE	24011.0	18 to 4 year
1301	2021-08	AL	1.0	HOUSTON	13153.0	Missin
1302	2021-03	NE	31.0	DOUGLAS	20045.0	18 to 4 year
1303	2020-06	VA	51.0	BUCHANAN	51027.0	Nal
1304	2020-10	KY	21.0	LINCOLN	21137.0	0 - 1 year
1305	2020-12	MI	26.0	HURON	26063.0	18 to 4 year
1306	2020-06	UT	49.0	SANPETE	49039.0	18 to 4 year
1307	2021-10	OK	40.0	BECKHAM	40009.0	0 - 1 year
1308	2021-03	CO	8.0	BROOMFIELD	8014.0	0 - 1 year
1309	2021-08	AR	5.0	CRITTENDEN	5035.0	0 - 1 year
1310	2021-03	WI	55.0	DOUGLAS	55031.0	0 - 1 year
1311	2021-09	MO	29.0	DUNKLIN	29069.0	0 - 1 year
1312	2021-07	ID	16.0	ELMORE	16039.0	0 - 1 year
1313	2020-12	OH	39.0	FAYETTE	39047.0	0 - 1 year
1314	2021-10	KS	20.0	MONTGOMERY	20125.0	0 - 1 year
1315	2021-02	VA	51.0	NEW KENT	51127.0	0 - 1 year
1316	2020-12	PA	42.0	NORTHUMBERLAND	42097.0	0 - 1 year
1317	2021-09	MT	30.0	SILVER BOW	30093.0	0 - 1 year
1318	2021-08	NY	36.0	ST. LAWRENCE	36089.0	0 - 1 year
1319	2020-09	OH	39.0	AUGLAIZE	39011.0	18 to 4 year

After Analyzing the above cell: I observed that there are a large number of missing values in columns {Sex}, {Age_Group}. I also recognized that simply dropping these

rows with missing values would result in a biased and inaccurate dataset. Instead, I will be using a different approach, which is utilizing the backward fill method to fill in the missing values in the 'SEX' column. This approach is chosen because it uses the previous value to fill the missing values because there is a pattern of missing values in the Sex and Age_group columns, thus it will be more accurate and will not make the dataset biased. By doing this I am trying to improve the results of their analysis by ensuring that the 'SEX' column has a complete and accurate set of data.

In [8]:
```
# filling the missing values with Backword fill aproach
covid_df['sex'].fillna(method='bfill', inplace=True)
```

In [9]:
```
# checking weather the records are filled or not
covid_df.iloc[1300:1320]
```

Out[9]:

	case_month	res_state	state_fips_code	res_county	county_fips_code	age_grou
1300	2020-05	MD	24.0	CAROLINE	24011.0	18 to 4 year
1301	2021-08	AL	1.0	HOUSTON	13153.0	Missin
1302	2021-03	NE	31.0	DOUGLAS	20045.0	18 to 4 year
1303	2020-06	VA	51.0	BUCHANAN	51027.0	Nal
1304	2020-10	KY	21.0	LINCOLN	21137.0	0 - 1 year
1305	2020-12	MI	26.0	HURON	26063.0	18 to 4 year
1306	2020-06	UT	49.0	SANPETE	49039.0	18 to 4 year
1307	2021-10	OK	40.0	BECKHAM	40009.0	0 - 1 year
1308	2021-03	CO	8.0	BROOMFIELD	8014.0	0 - 1 year
1309	2021-08	AR	5.0	CRITTENDEN	5035.0	0 - 1 year
1310	2021-03	WI	55.0	DOUGLAS	55031.0	0 - 1 year
1311	2021-09	MO	29.0	DUNKLIN	29069.0	0 - 1 year
1312	2021-07	ID	16.0	ELMORE	16039.0	0 - 1 year
1313	2020-12	OH	39.0	FAYETTE	39047.0	0 - 1 year
1314	2021-10	KS	20.0	MONTGOMERY	20125.0	0 - 1 year
1315	2021-02	VA	51.0	NEW KENT	51127.0	0 - 1 year
1316	2020-12	PA	42.0	NORTHUMBERLAND	42097.0	0 - 1 year
1317	2021-09	MT	30.0	SILVER BOW	30093.0	0 - 1 year
1318	2021-08	NY	36.0	ST. LAWRENCE	36089.0	0 - 1 year
1319	2020-09	OH	39.0	AUGLAIZE	39011.0	18 to 4 year

In [10]:
```
# filling the missing values with Backword fill aproach
covid_df['age_group'].fillna(method='bfill', inplace=True)
```

In [11]:
```
covid_df.iloc[1300:1320]
```

Out[11]:

	case_month	res_state	state_fips_code	res_county	county_fips_code	age_grou
1300	2020-05	MD	24.0	CAROLINE	24011.0	18 to 4 year
1301	2021-08	AL	1.0	HOUSTON	13153.0	Missin
1302	2021-03	NE	31.0	DOUGLAS	20045.0	18 to 4 year
1303	2020-06	VA	51.0	BUCHANAN	51027.0	0 - 1 year
1304	2020-10	KY	21.0	LINCOLN	21137.0	0 - 1 year
1305	2020-12	MI	26.0	HURON	26063.0	18 to 4 year
1306	2020-06	UT	49.0	SANPETE	49039.0	18 to 4 year
1307	2021-10	OK	40.0	BECKHAM	40009.0	0 - 1 year
1308	2021-03	CO	8.0	BROOMFIELD	8014.0	0 - 1 year
1309	2021-08	AR	5.0	CRITTENDEN	5035.0	0 - 1 year
1310	2021-03	WI	55.0	DOUGLAS	55031.0	0 - 1 year
1311	2021-09	MO	29.0	DUNKLIN	29069.0	0 - 1 year
1312	2021-07	ID	16.0	ELMORE	16039.0	0 - 1 year
1313	2020-12	OH	39.0	FAYETTE	39047.0	0 - 1 year
1314	2021-10	KS	20.0	MONTGOMERY	20125.0	0 - 1 year
1315	2021-02	VA	51.0	NEW KENT	51127.0	0 - 1 year
1316	2020-12	PA	42.0	NORTHUMBERLAND	42097.0	0 - 1 year
1317	2021-09	MT	30.0	SILVER BOW	30093.0	0 - 1 year
1318	2021-08	NY	36.0	ST. LAWRENCE	36089.0	0 - 1 year
1319	2020-09	OH	39.0	AUGLAIZE	39011.0	18 to 4 year

After analyzing a random sample of rows and checking the 'Race', 'Ethnicity', 'Underling_Conditions_yn', and 'death_yn' columns of the DataFrame, it was determined that there were no patterns of missing values in those columns. Dropping rows with missing values in these columns would result in a biased and inaccurate analysis. As the dataframe does not have information about the race and ethnicity of the missing values, I choose an alternative approach to fill the missing values with the word 'Unknown'. This approach will provide better results as it will not compromise the integrity of the data and will not result in a biased analysis. This way I can have a more accurate understanding of the data and the missing values will not affect the analysis results.

In [12]:
```
# filling the missing values with "Unknown" word
covid_df['race'].fillna(value = 'Unknown', inplace = True)
```

In [13]:
```
# checking weather the records are filled or not
covid_df['race']
```

Out[13]:
```
0           Unknown
1           Unknown
2           Unknown
3           Missing
4           Unknown
             ...
37532067      White
37532068      White
37532069      White
37532070      White
37532071      White
Name: race, Length: 37532072, dtype: object
```

In [14]:
```
# filling the missing values with "Unknown" word
covid_df['ethnicity'].fillna(value = 'Unknown', inplace = True)
```

In [15]:
```
# checking weather the records are filled or not
covid_df['ethnicity'].head(50)
```

Out[15]:
```
0                 Unknown
1                 Unknown
2                 Unknown
3                 Missing
4                 Missing
5                 Unknown
6                 Unknown
7                 Missing
8                 Missing
9                 Missing
10                Missing
11                Missing
12                Missing
13                Unknown
14                Unknown
15                Unknown
16                Unknown
17                Unknown
18                Missing
19                Missing
20                Missing
21                Unknown
22                Unknown
23                Unknown
24                Unknown
25                Missing
26                Unknown
27                Missing
28                Unknown
29                Unknown
30        Hispanic/Latino
31        Hispanic/Latino
32        Hispanic/Latino
33        Hispanic/Latino
34        Hispanic/Latino
35        Hispanic/Latino
36        Hispanic/Latino
37        Hispanic/Latino
38        Hispanic/Latino
39        Hispanic/Latino
40        Hispanic/Latino
41        Hispanic/Latino
42        Hispanic/Latino
43        Hispanic/Latino
44        Hispanic/Latino
45        Hispanic/Latino
46        Hispanic/Latino
47        Hispanic/Latino
48        Hispanic/Latino
49    Non-Hispanic/Latino
Name: ethnicity, dtype: object
```

In [16]:
```
covid_df['underlying_conditions_yn'].fillna(value = 'Unknown', inplace = True)
```

In [17]:
```
covid_df['underlying_conditions_yn'].head(50)
```

Out[17]:
```
0      Unknown
1      Unknown
2      Unknown
3      Unknown
4      Unknown
5      Unknown
6      Unknown
7      Unknown
8      Unknown
9      Unknown
10     Unknown
11     Unknown
12     Unknown
13     Unknown
14     Unknown
15     Unknown
16     Unknown
17     Unknown
18     Unknown
19     Unknown
20     Unknown
21     Unknown
22     Unknown
23     Unknown
24     Unknown
25     Unknown
26     Unknown
27     Unknown
28     Unknown
29     Unknown
30     Unknown
31     Unknown
32     Unknown
33     Unknown
34     Unknown
35     Unknown
36     Unknown
37     Unknown
38     Unknown
39     Unknown
40     Unknown
41     Unknown
42     Unknown
43     Unknown
44     Unknown
45     Unknown
46     Unknown
47     Unknown
48         Yes
49     Unknown
Name: underlying_conditions_yn, dtype: object
```

In [18]:
```
covid_df['death_yn'].fillna(value = 'Unknown', inplace = True)
```

In [19]:
```
covid_df['death_yn'].head(50)
```

Out[19]:
```
0     Unknown
1     Unknown
2     Unknown
3     Missing
4     Missing
5          No
6          No
7          No
8     Missing
9     Missing
10    Missing
11    Missing
12    Missing
13    Unknown
14    Unknown
15    Unknown
16    Missing
17    Unknown
18    Missing
19    Missing
20    Missing
21    Unknown
22    Unknown
23    Unknown
24         No
25    Missing
26         No
27    Missing
28    Unknown
29    Unknown
30    Missing
31    Missing
32    Missing
33         No
34    Missing
35    Unknown
36    Missing
37    Missing
38    Unknown
39         No
40         No
41    Unknown
42         No
43         No
44         No
45         No
46         No
47         No
48    Unknown
49    Missing
Name: death_yn, dtype: object
```

After analyzing the 'case_positive_specimen_interval','case_onset_interval' columns of the DataFrame and observing that it has a relatively numerical values. I recognized that dropping the rows with missing values in this column will significantly impact the overall dataset. Therefore, I decided to take the mean of the whole column. This approach is chosen because the number of missing values in this column is large and randomly generated, so mean will surely affect the overall analysis.

In [20]:
```
# Using imputation techniques to fill the missing values with mean of the colu

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

# Fitting the imputer on the data
imputer.fit(covid_df[['case_positive_specimen_interval','case_onset_interval']]

# replacing missing values with the mean value
covid_df[['case_positive_specimen_interval','case_onset_interval']] = imputer.t
```

In [21]:
```
# checking both case_positive_specimen_interval and case_onset_interval records
covid_df[['case_positive_specimen_interval','case_onset_interval']].head(50)
```

Out[21]:

	case_positive_specimen_interval	case_onset_interval
0	0.000000	0.099948
1	0.000000	0.099948
2	0.000000	0.000000
3	0.000000	0.099948
4	0.200811	0.000000
5	0.200811	0.000000
6	0.200811	0.099948
7	0.000000	0.099948
8	0.200811	0.000000
9	0.200811	0.099948
10	0.200811	0.099948
11	0.200811	0.099948
12	0.200811	0.000000
13	0.000000	0.000000
14	0.200811	0.000000
15	0.200811	0.000000
16	0.200811	0.000000
17	0.200811	0.099948
18	0.000000	0.099948
19	0.200811	0.099948
20	0.200811	0.000000
21	0.200811	0.099948
22	0.200811	0.099948
23	0.000000	0.000000
24	0.200811	0.000000
25	0.200811	0.000000
26	0.200811	0.099948
27	0.200811	0.099948
28	0.000000	0.099948
29	0.200811	0.000000
30	0.200811	0.099948
31	0.200811	0.099948
32	0.200811	0.099948
33	0.200811	0.099948
34	0.200811	0.099948

	case_positive_specimen_interval	case_onset_interval
35	0.200811	0.099948
36	0.200811	0.099948
37	0.200811	0.000000
38	0.200811	0.099948
39	0.200811	0.000000
40	0.000000	0.099948
41	0.200811	0.099948
42	0.200811	0.099948
43	0.000000	0.099948
44	0.200811	0.099948
45	0.200811	0.000000
46	0.200811	0.000000
47	0.000000	0.000000
48	0.000000	0.000000
49	0.200811	0.000000

After analyzing the 'res_state' column of the DataFrame and observing that it has a relatively low number of missing values compared to other columns. I recognized that dropping the rows with missing values in this column will not significantly impact the overall dataset. Therefore, I decided to drop the rows with missing values in the 'res_state' column in order to maintain the integrity and accuracy of the overall dataset. This approach is chosen because the number of missing values in this column is not large enough to affect the overall analysis and by dropping the missing values the data will be more accurate.

In [22]:
```
# Dropping the Column 'res_county' and 'County_fips_code' based on the missing
covid_df.dropna(subset = ['res_state', 'county_fips_code'], inplace = True)
```

In [23]:
```
covid_df.isna().sum()
```

Out[23]:
```
case_month                          0
res_state                           0
state_fips_code                     0
res_county                          0
county_fips_code                    0
age_group                           0
sex                                 0
race                                0
ethnicity                           0
case_positive_specimen_interval     0
case_onset_interval                 0
process                             0
exposure_yn                         0
current_status                      0
symptom_status                      0
hosp_yn                             0
icu_yn                              0
death_yn                            0
underlying_conditions_yn            0
dtype: int64
```

All the missing values are done with processing now !!

Dealing with Relevant Columns now

In [24]:
```
covid_df.head()
```

Out[24]:

	case_month	res_state	state_fips_code	res_county	county_fips_code	age_group	sex
0	2020-12	AR	5.0	BOONE	5009.0	0 - 17 years	Female
1	2021-10	MO	29.0	PHELPS	29161.0	0 - 17 years	Female
2	2020-10	MO	29.0	CASS	29037.0	0 - 17 years	Female
3	2021-08	TN	47.0	CLAIBORNE	47025.0	0 - 17 years	Female
4	2020-11	GA	13.0	EFFINGHAM	13103.0	0 - 17 years	Female

In [25]:
```
# Checking the datatype of columns
covid_df.dtypes
```

Out[25]:
```
case_month                          object
res_state                           object
state_fips_code                    float64
res_county                          object
county_fips_code                   float64
age_group                           object
sex                                 object
race                                object
ethnicity                           object
case_positive_specimen_interval    float64
case_onset_interval                float64
process                             object
exposure_yn                         object
current_status                      object
symptom_status                      object
hosp_yn                             object
icu_yn                              object
death_yn                            object
underlying_conditions_yn            object
dtype: object
```

In [26]:
```
## Changing the datatype of column state_fips_code from float to 'Int'.
covid_df['state_fips_code'] = covid_df['state_fips_code'].astype(int)
```

I am renaming the 'state_fips_code' column of the current dataset to 'State Code'. The reason for this is that I noticed that there is a similar column with the same name in the dataset they were working on in task 1, which is the AQI dataset. By renaming the column, I am trying to make it easy to merge both the datasets on this column. This will enable them to join the data from both datasets and analyze it together.

In [27]:
```
# Renaming the column from state_fips_code' to 'State Code'
covid_df = covid_df.rename(columns = {'state_fips_code':'State Code'})
```

Taking relevant columns for further Analysis and dropping the rest of the uncessary columns for this Task-2

In [28]:
```
# Dropping Columns from dataframe

covid_df = covid_df.drop(columns = ['res_state', 'res_county', 'county_fips_coc
       'case_positive_specimen_interval', 'case_onset_interval', 'process',
       'exposure_yn', 'current_status', 'symptom_status'])
```

In [29]:
```
#Checking the dataframe
covid_df.head()
```

Out[29]:

	case_month	State Code	hosp_yn	icu_yn	death_yn	underlying_conditions_yn
0	2020-12	5	Missing	Missing	Unknown	Unknown
1	2021-10	29	Unknown	Missing	Unknown	Unknown
2	2020-10	29	Unknown	Missing	Unknown	Unknown
3	2021-08	47	Missing	Missing	Missing	Unknown
4	2020-11	13	Missing	Missing	Missing	Unknown

In [30]:
```
# Checking the dtypes
covid_df.dtypes
```

Out[30]:
```
case_month                  object
State Code                   int64
hosp_yn                     object
icu_yn                      object
death_yn                    object
underlying_conditions_yn    object
dtype: object
```

In [31]:
```
# Checking the size of dataframe
covid_df.shape
```

Out[31]:
```
(34970788, 6)
```

In [32]:
```
# converting the 'case_month' column of covid DataFrame into datetime datatype.
covid_df['case_month'] = pd.to_datetime(covid_df['case_month'])
covid_df['case_month'].head()
```

Out[32]:
```
0   2020-12-01
1   2021-10-01
2   2020-10-01
3   2021-08-01
4   2020-11-01
Name: case_month, dtype: datetime64[ns]
```

In [33]:
```
## Changing the datatype of column state_fips_code from float to 'Int'.
covid_df['State Code'] = covid_df['State Code'].astype(int)
```

In [34]:
```
covid_df.dtypes
```

Out[34]:
```
case_month                  datetime64[ns]
State Code                           int64
hosp_yn                             object
icu_yn                              object
death_yn                            object
underlying_conditions_yn            object
dtype: object
```

In [35]:
```
# filtering data for further analysis based on AQI_Datasets of year 2020 and 2(
filtered_df = covid_df[covid_df['case_month'].dt.year.isin([2020, 2021])]
```

In [36]:
```
filtered_df.shape
```

Out[36]:
```
(34970788, 6)
```

I identified that in order to calculate correlation and other factors related to the data I am working on. However, I realized that using the categorical values of the data will not be possible. So, I have decided to use an approach of converting the categorical values to numerical values. This will enable me to calculate factors such as hospital rate, death_yn, and underlying conditions based on the numerical values. This approach was chosen because numerical values can be used for mathematical calculations, making it possible to calculate correlation rates and other factors.

Additionally, having numerical values instead of categorical values will make it easier to perform good analysis.

In [37]:
```
# Converting the hosp_yn column categorical values to numerical
filtered_df['hosp_yn'].replace({'Yes':1, 'No':0, 'Unknown':0, 'Missing':0}, inp
```

In [38]:
```
# Converting the death_yn column categorical values to numerical
filtered_df['death_yn'].replace({'Yes':1, 'No':0, 'Unknown':0, 'Missing':0}, in
```

In [39]:
```
# Converting the icu_yn column categorical values to numerical
filtered_df['icu_yn'].replace({'Yes':1, 'No':0, 'Unknown':0, 'Missing':0}, inpl
```

In [40]:
```
# Converting the underlying_conditions_yn column categorical values to numerica
filtered_df['underlying_conditions_yn'].replace({'Yes':1, 'No':0, 'Unknown':0},
```

In [41]:
```
# checking the size of covid dataframe
filtered_df.shape
```

Out[41]:
```
(34970788, 6)
```

In [42]:
```
# Grouping the covid DataFrame by 'State_code' and 'case_month' columns and  ca

df_covid = filtered_df.groupby(['State Code', 'case_month'],as_index=False).sum
```

The Dataframe looks good with relevant columns now

In [43]:
```
df_covid.head(25)
```

Out[43]:

	State Code	case_month	hosp_yn	death_yn	underlying_conditions_yn
0	1	2020-03-01	405	11	22
1	1	2020-04-01	673	164	29
2	1	2020-05-01	671	74	70
3	1	2020-06-01	943	82	163
4	1	2020-07-01	1339	309	126
5	1	2020-08-01	1079	179	116
6	1	2020-09-01	700	0	84
7	1	2020-10-01	744	24	104
8	1	2020-11-01	673	234	113
9	1	2020-12-01	985	855	148
10	1	2021-01-01	1001	675	147
11	1	2021-02-01	664	23	100
12	1	2021-03-01	445	0	76
13	1	2021-04-01	418	0	83
14	1	2021-05-01	347	0	58
15	1	2021-06-01	240	0	45
16	1	2021-07-01	904	129	180
17	1	2021-08-01	1996	555	480
18	1	2021-09-01	1392	134	337
19	1	2021-10-01	384	0	98
20	2	2020-03-01	26	0	114
21	2	2020-04-01	11	0	65
22	2	2020-05-01	11	0	77
23	2	2020-06-01	28	0	246
24	2	2020-07-01	70	0	402

Highly Important Note below !!:

The AQI Concatenated dataset was imported and read separately to avoid kernel crashes due to the large size of the COVID dataset. This allows for more efficient processing and analysis of the data without running out of

memory. Now, the AQI dataset is being called directly in this notebook to merge with the COVID dataset for further analysis.

In [44]:
```
## Reading the Concatenated AQI Dataset of the relevant years

AQI_df = pd.read_csv('/Users/vishesh/Downloads/DA/f_AQI.csv')
```

In [45]:
```
AQI_df.head()
```

Out[45]:

	Unnamed: 0	State_Name	year_month	State Code	AQI
0	0	Alabama	2020-01	1.0	29.538732
1	1	Alabama	2020-02	1.0	28.063433
2	2	Alabama	2020-03	1.0	37.696078
3	3	Alabama	2020-04	1.0	41.858209
4	4	Alabama	2020-05	1.0	40.176329

In [46]:
```
# Dropping the irrelevant columns
AQI_df = AQI_df.drop(columns=['Unnamed: 0'])
```

In [47]:
```
df_covid
```

Out[47]:

	State Code	case_month	hosp_yn	death_yn	underlying_conditions_yn
0	1	2020-03-01	405	11	22
1	1	2020-04-01	673	164	29
2	1	2020-05-01	671	74	70
3	1	2020-06-01	943	82	163
4	1	2020-07-01	1339	309	126
...	...	...	...	...	...
998	56	2021-06-01	66	0	645
999	56	2021-07-01	104	0	846
1000	56	2021-08-01	165	0	2141
1001	56	2021-09-01	188	0	2348
1002	56	2021-10-01	162	0	2030

1003 rows × 5 columns

In [48]:
```
df_covid.dtypes
```

Out[48]:
```
State Code                          int64
case_month                 datetime64[ns]
hosp_yn                             int64
death_yn                            int64
underlying_conditions_yn            int64
dtype: object
```

By generating unique columns in both the covid and AQI dataframes, it allows for easy merging of the two datasets by creating a common identifier. This will prevent duplicate records and ensure accurate data analysis. The approach used here is to create a unique ID for each row in both datasets, making it possible to merge them with ease.

In [49]:
```
# Converting case_month to string format of '%Y-%m' which represents the year-m
df_covid['case_month'] = df_covid['case_month'].dt.strftime('%Y-%m')
```

In [50]:
```
# Generating Unique 'ID' Columns in covid dataframe
df_covid['ID'] = df_covid['State Code'].astype(str)+df_covid['case_month'].asty
```

In [51]:
```
# Generating Unique 'ID' Columns in AQI dataframe
AQI_df['ID'] = (AQI_df['State Code'].astype(int)).astype(str)+AQI_df['year_mont
```

In [52]:
```
# Checking the ID Column in AQI DataFrame
AQI_df
```

Out[52]:

	State_Name	year_month	State Code	AQI	ID
0	Alabama	2020-01	1.0	29.538732	12020-01
1	Alabama	2020-02	1.0	28.063433	12020-02
2	Alabama	2020-03	1.0	37.696078	12020-03
3	Alabama	2020-04	1.0	41.858209	12020-04
4	Alabama	2020-05	1.0	40.176329	12020-05
...	...	...	...	...	...
1143	Wyoming	2021-06	56.0	45.946939	562021-06
1144	Wyoming	2021-07	56.0	59.803191	562021-07
1145	Wyoming	2021-08	56.0	63.000000	562021-08
1146	Wyoming	2021-09	56.0	41.371622	562021-09
1147	Wyoming	2021-10	56.0	12.250000	562021-10

1148 rows × 5 columns

In [53]:
```
# Checking the ID Column in Covid DataFrame
df_covid
```

Out[53]:

	State Code	case_month	hosp_yn	death_yn	underlying_conditions_yn	ID
0	1	2020-03	405	11	22	12020-03
1	1	2020-04	673	164	29	12020-04
2	1	2020-05	671	74	70	12020-05
3	1	2020-06	943	82	163	12020-06
4	1	2020-07	1339	309	126	12020-07
...	...	...	...	...	...	...
998	56	2021-06	66	0	645	562021-06
999	56	2021-07	104	0	846	562021-07
1000	56	2021-08	165	0	2141	562021-08
1001	56	2021-09	188	0	2348	562021-09
1002	56	2021-10	162	0	2030	562021-10

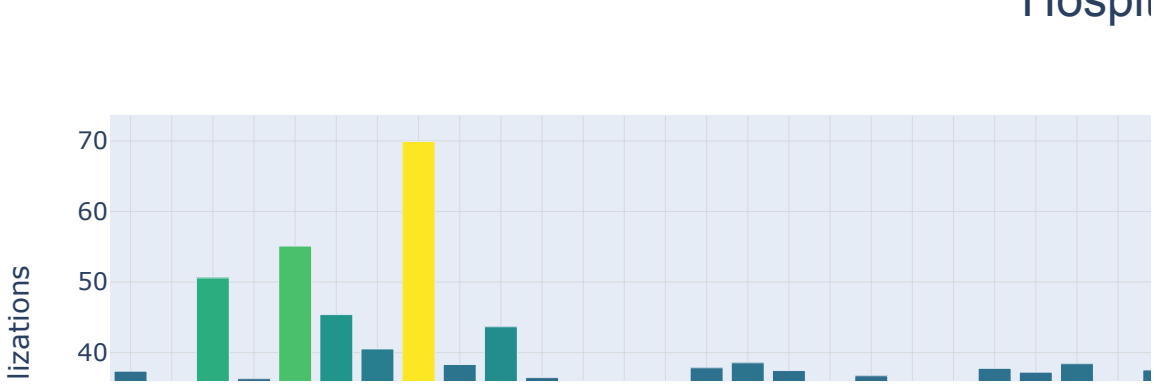
1003 rows × 6 columns

Performing Some Visualizations

Visualizations over State vs Hospitalization

In [54]:
```
fg = AQI_df.groupby(['State_Name'],as_index=False)['AQI'].mean()
```

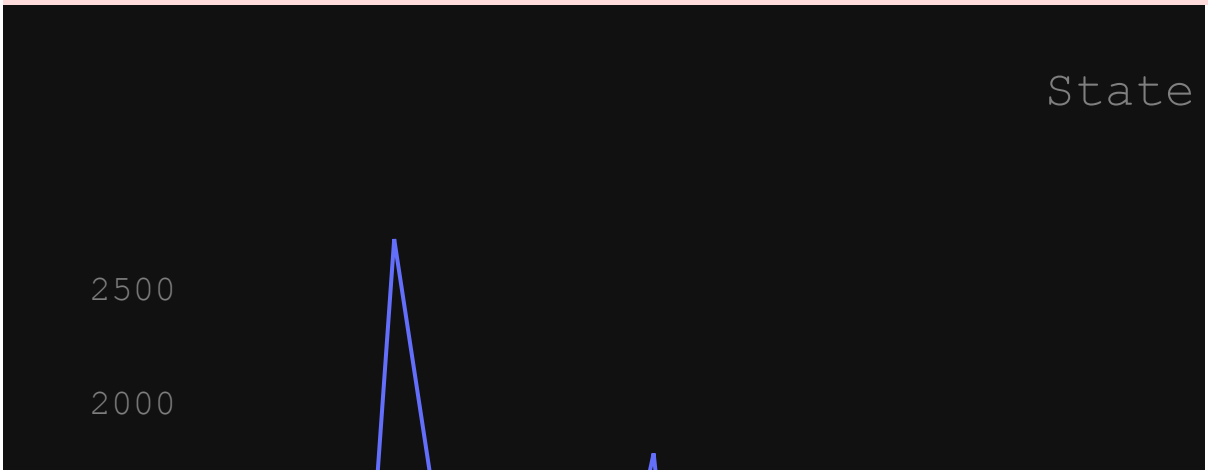
In [55]:
```
# Generating State_Name vs AQI charts

import plotly.express as px

fig = px.bar(fg, x='State_Name', y='AQI', title='Hospitalizations over time',
             color='AQI', color_continuous_scale='Viridis',
             labels={'State_Name':'State', 'AQI':'Hospitalizations'})
fig.update_layout(title={'x':0.5,'text':'Hospitalizations over State'},
                  xaxis_title='State', yaxis_title='Hospitalizations',
                  xaxis=dict(showgrid=True, gridcolor='lightgray', gridwidth=0.
                  yaxis=dict(showgrid=True, gridcolor='lightgray', gridwidth=0.
                  title_font=dict(size=24, family='Arial, sans-serif'),
                  paper_bgcolor='white')

fig.update_layout(
    yaxis=dict(
        tickformat=',d',
        showgrid=True,
        gridcolor='lightgray',
        gridwidth=0.5
    )
)



fig.show()
```



Visualizations over State vs Death

In [56]:
```
rg = df_covid.groupby(['State Code'],as_index=False)['hosp_yn', 'death_yn'].mea
import plotly.express as px

fig = px.line(rg, x='State Code', y='death_yn', title='State Code over Death')
fig.update_layout(title={'x':0.5,'text':'State Code over Death'},template='plot
fig.update_layout(
    yaxis=dict(
        tickformat='d'
    )
)

fig.update_layout(
    xaxis=dict(
        tickmode='linear'
    )
)

fig.show()
```

```
/var/folders/nr/x5htffh97tvfw1rt32pzkmc00000gn/T/ipykernel_51259/3166128952.py:1: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```



In [57]:
```
AQI_df.dtypes
```

Out[57]:
```
State_Name     object
year_month     object
State Code    float64
AQI           float64
ID             object
dtype: object
```

In [58]:
```
df_covid.dtypes
```

Out[58]:
```
State Code                  int64
case_month                 object
hosp_yn                     int64
death_yn                    int64
underlying_conditions_yn    int64
ID                         object
dtype: object
```

{ Merging AQI Dataframe and Covid Dataframe }

In [59]:
```
# Merging AQI and Covid dataframe
merged_df2 = pd.merge(AQI_df, df_covid, on='ID')
```

In [60]:
```
merged_df2.iloc[20:40]
```

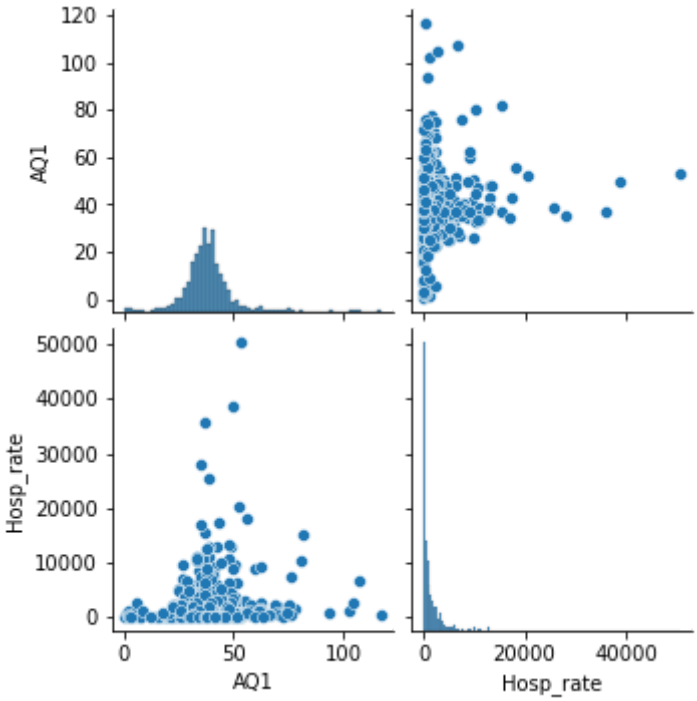
Out[60]:

	State_Name	year_month	State Code_x	AQI	ID	State Code_y	case_month	hosp_yn	dea
20	Alaska	2020-05	2.0	22.057416	22020-05	2	2020-05	11	
21	Alaska	2020-06	2.0	16.250000	22020-06	2	2020-06	28	
22	Alaska	2020-07	2.0	16.859459	22020-07	2	2020-07	70	
23	Alaska	2020-08	2.0	15.285714	22020-08	2	2020-08	80	
24	Alaska	2020-09	2.0	17.220339	22020-09	2	2020-09	62	
25	Alaska	2020-10	2.0	25.626374	22020-10	2	2020-10	169	
26	Alaska	2020-11	2.0	35.616667	22020-11	2	2020-11	283	
27	Alaska	2020-12	2.0	32.054945	22020-12	2	2020-12	196	
28	Alaska	2021-01	2.0	45.664516	22021-01	2	2021-01	74	
29	Alaska	2021-02	2.0	35.892857	22021-02	2	2021-02	55	
30	Alaska	2021-03	2.0	28.780645	22021-03	2	2021-03	88	
31	Alaska	2021-04	2.0	29.298611	22021-04	2	2021-04	115	
32	Alaska	2021-05	2.0	23.941935	22021-05	2	2021-05	44	
33	Alaska	2021-06	2.0	20.729167	22021-06	2	2021-06	35	
34	Alaska	2021-07	2.0	23.062500	22021-07	2	2021-07	142	
35	Alaska	2021-08	2.0	23.612903	22021-08	2	2021-08	286	
36	Alaska	2021-09	2.0	27.433333	22021-09	2	2021-09	364	
37	Arizona	2020-01	4.0	38.815920	42020-01	4	2020-01	0	
38	Arizona	2020-02	4.0	45.862069	42020-02	4	2020-02	10	
39	Arizona	2020-03	4.0	44.865672	42020-03	4	2020-03	769	

In [61]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
t1 = merged_df2[['AQI', 'hosp_yn']]
t1.columns = ['AQ1', 'Hosp_rate']
plt.figure(figsize=(12,7))
sns.pairplot(t1)
```

Out[61]:
```
<seaborn.axisgrid.PairGrid at 0x7f86c0c36040>

<Figure size 864x504 with 0 Axes>
```



In [62]:
```
# Checking the null values
merged_df2.isnull().sum()
```

Out[62]:
```
State_Name                0
year_month                0
State Code_x              0
AQI                       0
ID                        0
State Code_y              0
case_month                0
hosp_yn                   0
death_yn                  0
underlying_conditions_yn  0
dtype: int64
```

In [63]:
```
# Taking the unique State codes
merged_df2['State Code_y'].unique()
```

Out[63]:
```
array([ 1,  2,  4,  5,  6,  8,  9, 10, 12, 13, 15, 16, 17, 18, 19, 20, 21,
       22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
       39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56])
```

In [64]:
```
# Generating the list of Unique State Codes
sc = list([ 1,  2,  4,  5,  6,  8,  9, 10, 12, 13, 15, 16, 17, 18, 19, 20, 21,
       22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
       39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56])
```

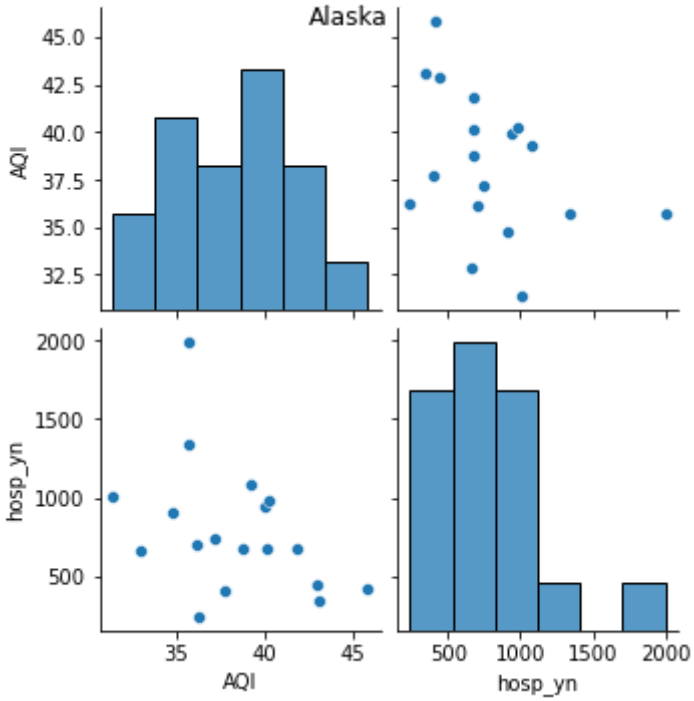
In [65]:
```
## prevent warnings from cluttering the output
import warnings
```

```
warnings.filterwarnings('ignore')

# Imoprting plotting libraries
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
```

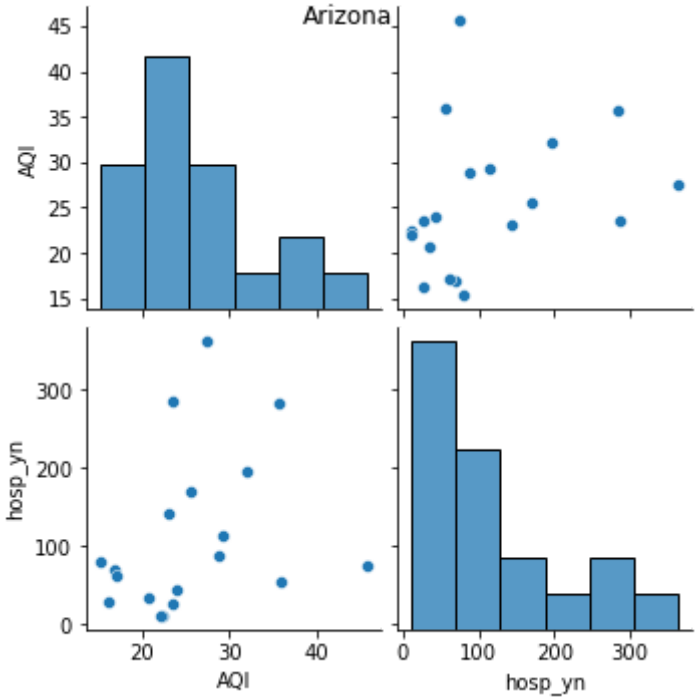
In [66]:
```
country_list=['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',
       'Colorado', 'Connecticut', 'Country Of Mexico', 'Delaware',
       'District Of Columbia', 'Florida', 'Georgia', 'Hawaii', 'Idaho',
       'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Louisiana',
       'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota',
       'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada',
       'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
       'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Oregon',
       'Pennsylvania', 'Puerto Rico', 'Rhode Island', 'South Carolina',
       'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Vermont',
       'Virgin Islands', 'Virginia', 'Washington', 'West Virginia',
       'Wisconsin', 'Wyoming']
```

In [67]:
```
for country in country_list:
    cn=[]
    cn.append(country)
    df_plotter = pd.DataFrame()
    df_plotter=merged_df2[merged_df2['State_Name'].isin(cn)]
    t1 = df_plotter[['AQI', 'hosp_yn']]
    plt.suptitle(country)
    plt.figure(figsize=(15,7))
    sns.pairplot(t1,diag_kind='hist')
```

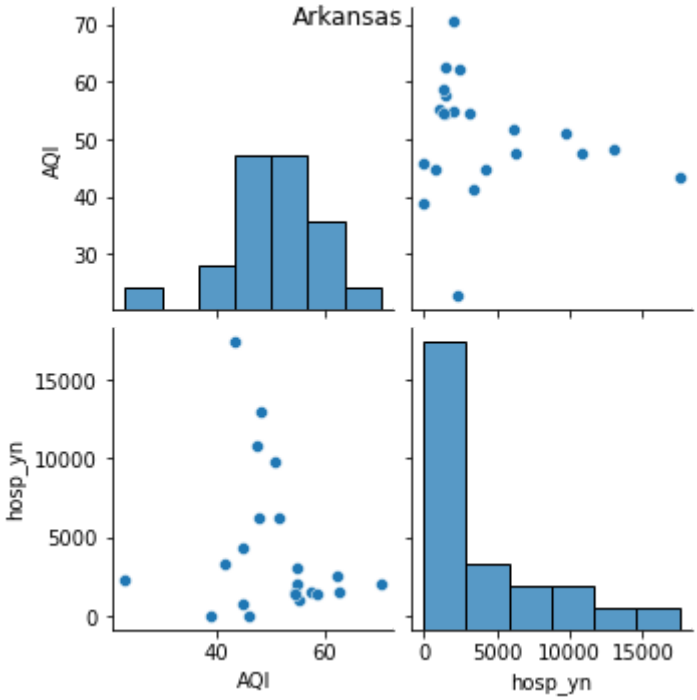
```
<Figure size 432x288 with 0 Axes>
<Figure size 1080x504 with 0 Axes>
```



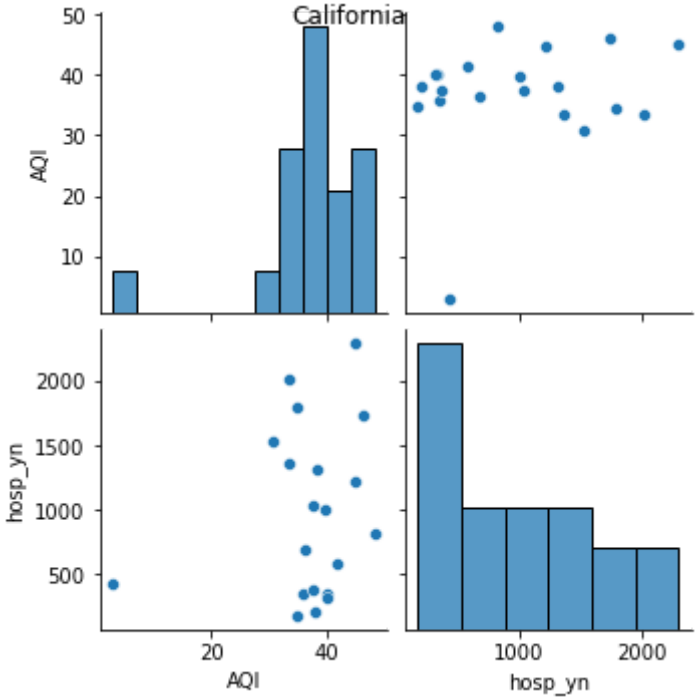

```
<Figure size 1080x504 with 0 Axes>
```



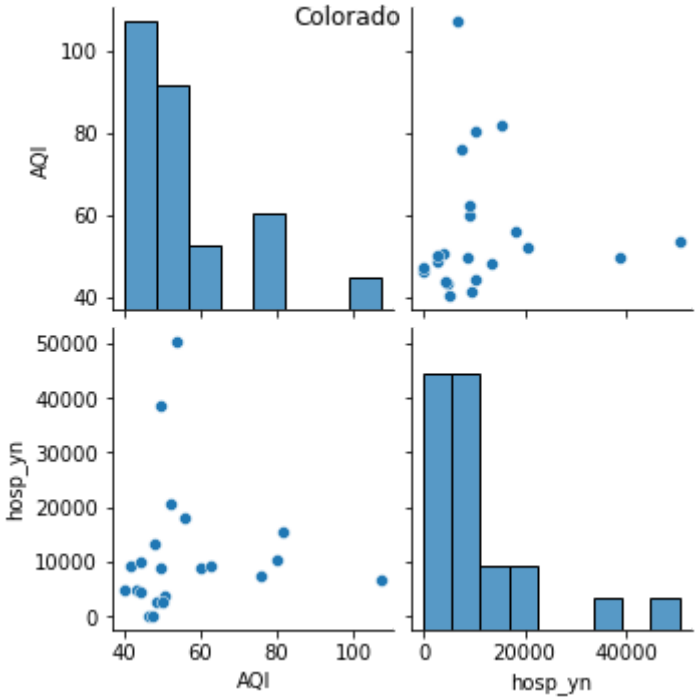

```
<Figure size 1080x504 with 0 Axes>
```



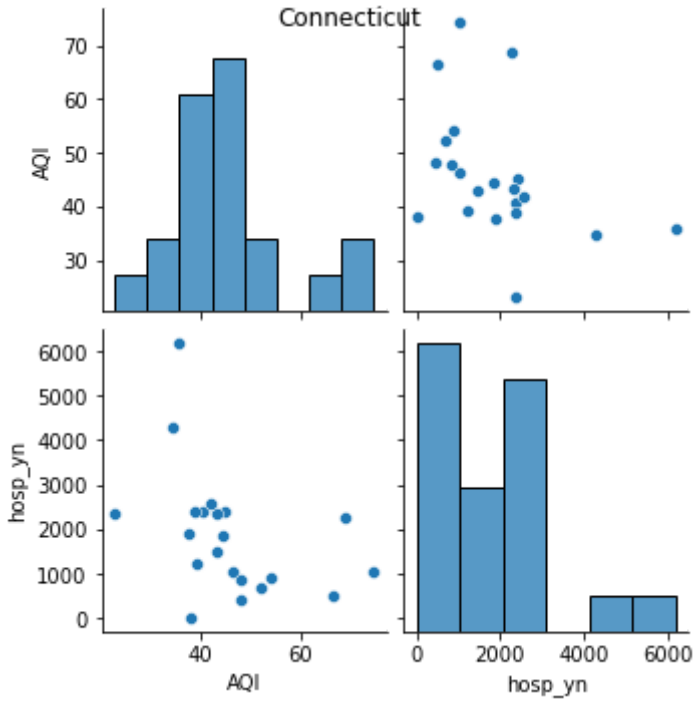

```
<Figure size 1080x504 with 0 Axes>
```



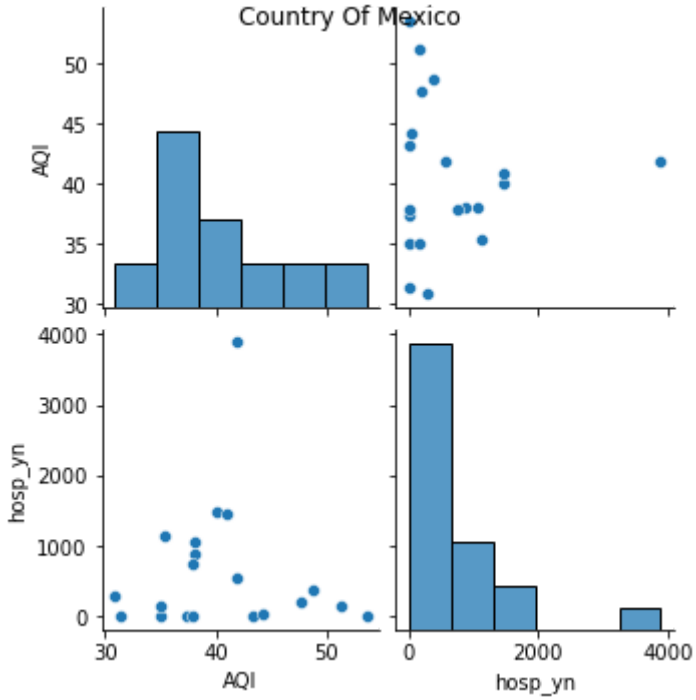

```
<Figure size 1080x504 with 0 Axes>
```



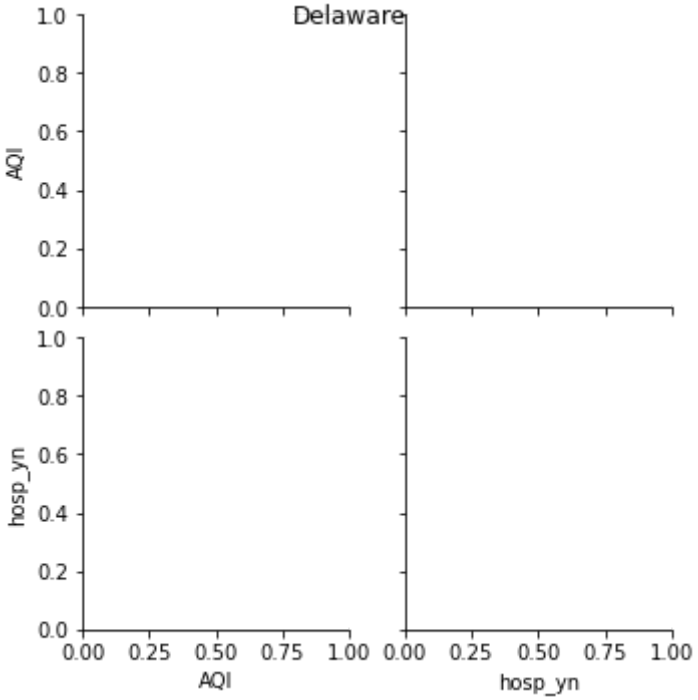

```
<Figure size 1080x504 with 0 Axes>
```



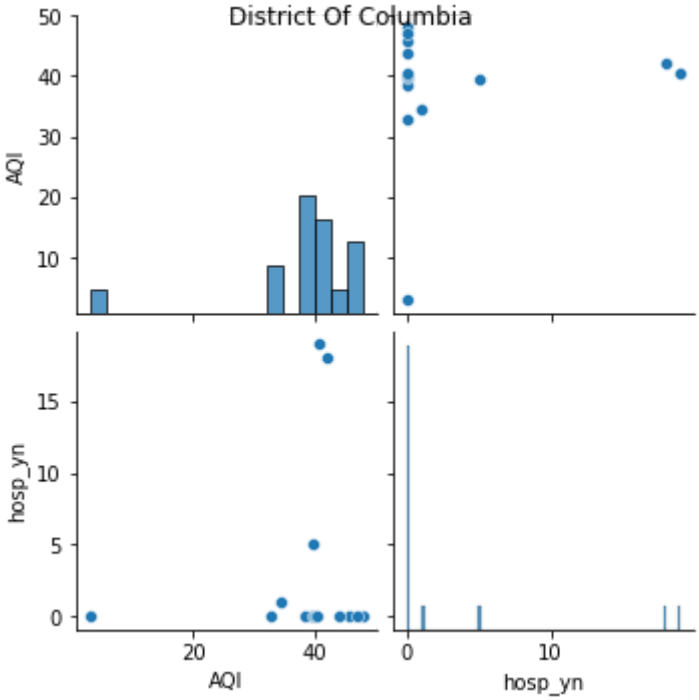

```
<Figure size 1080x504 with 0 Axes>
```



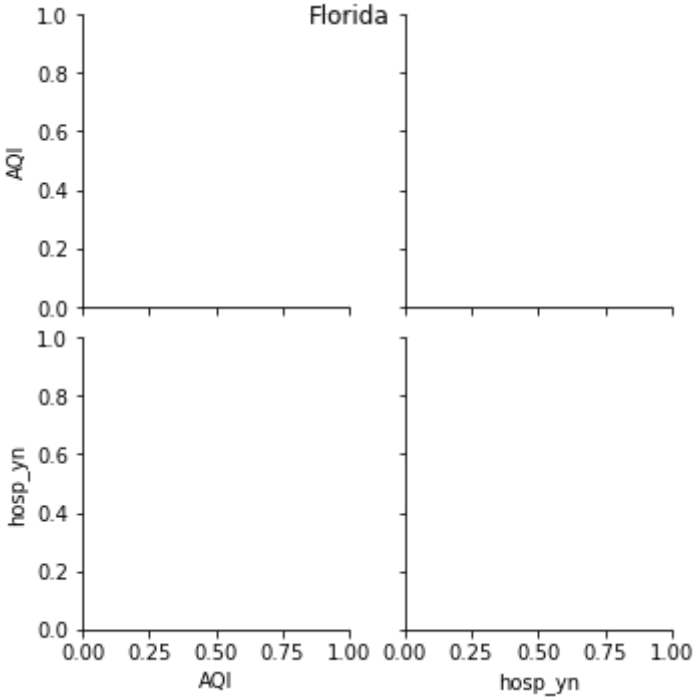

```
<Figure size 1080x504 with 0 Axes>
```



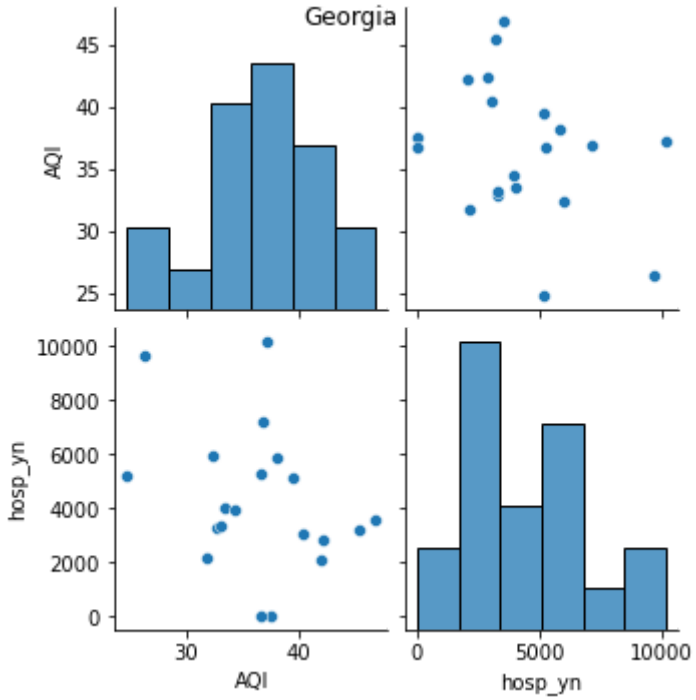

```
<Figure size 1080x504 with 0 Axes>
```



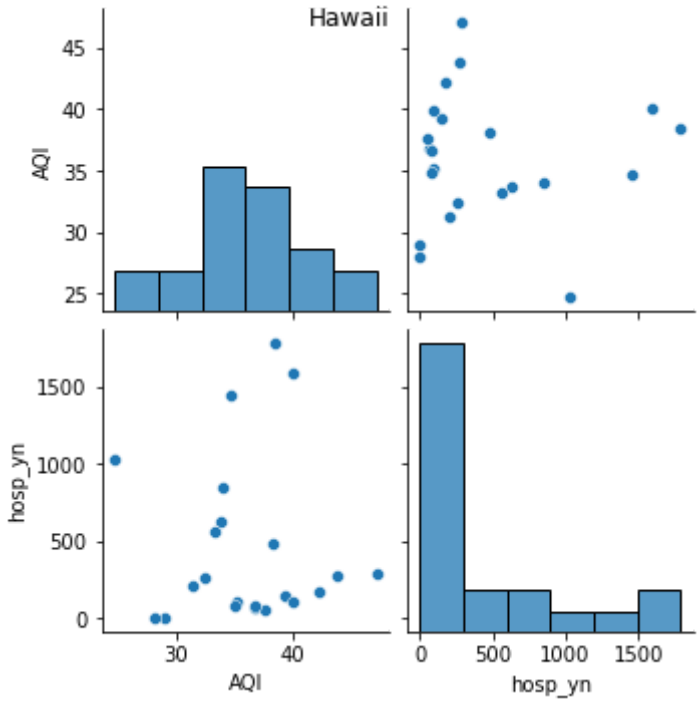

```
<Figure size 1080x504 with 0 Axes>
```



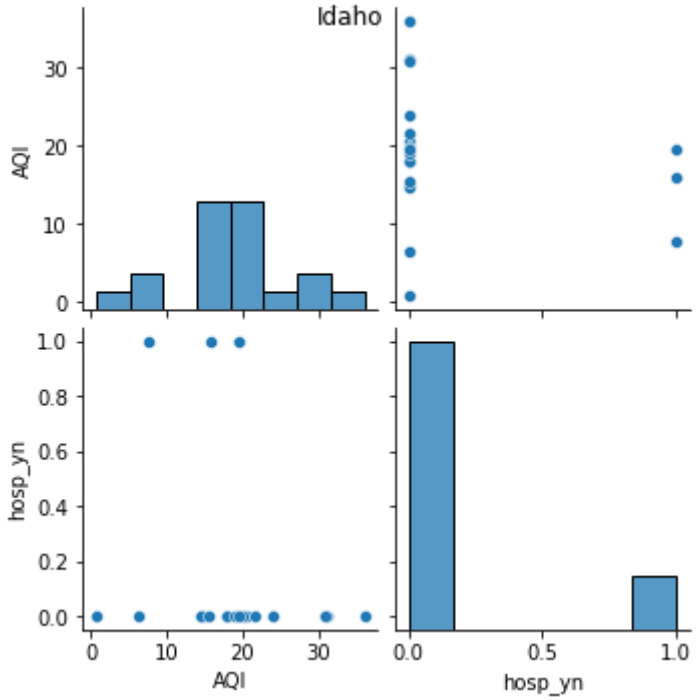

```
<Figure size 1080x504 with 0 Axes>
```



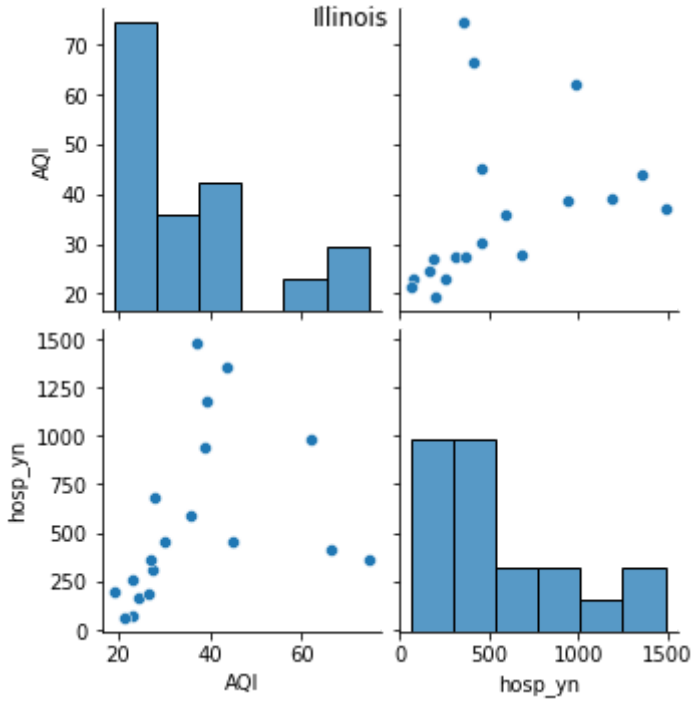

```
<Figure size 1080x504 with 0 Axes>
```



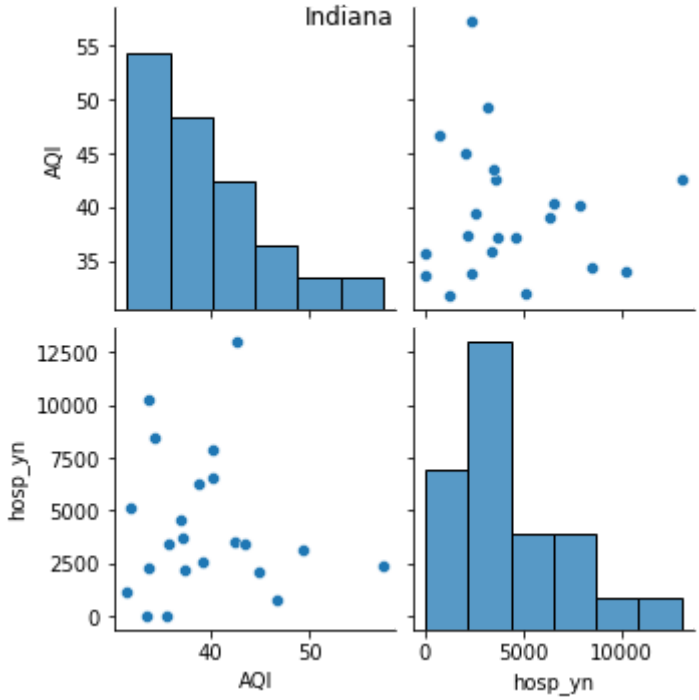
<Figure size 1080x504 with 0 Axes>



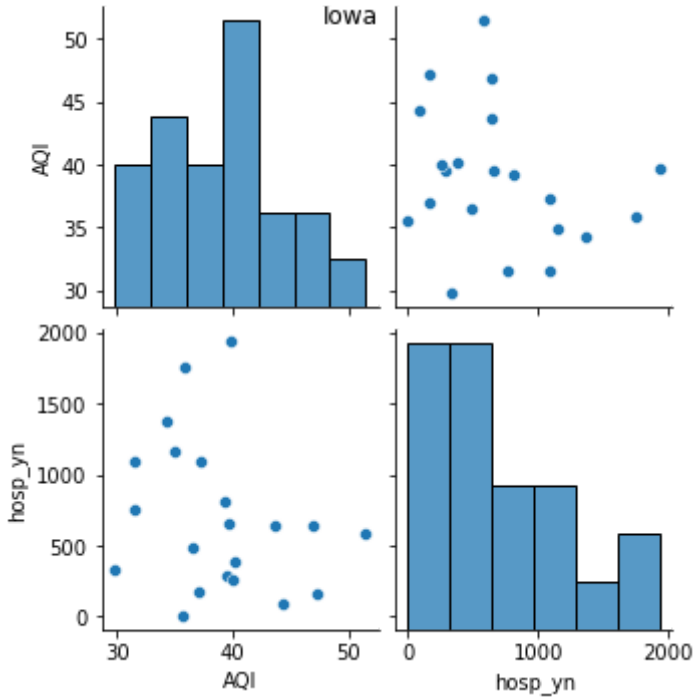
<Figure size 1080x504 with 0 Axes>



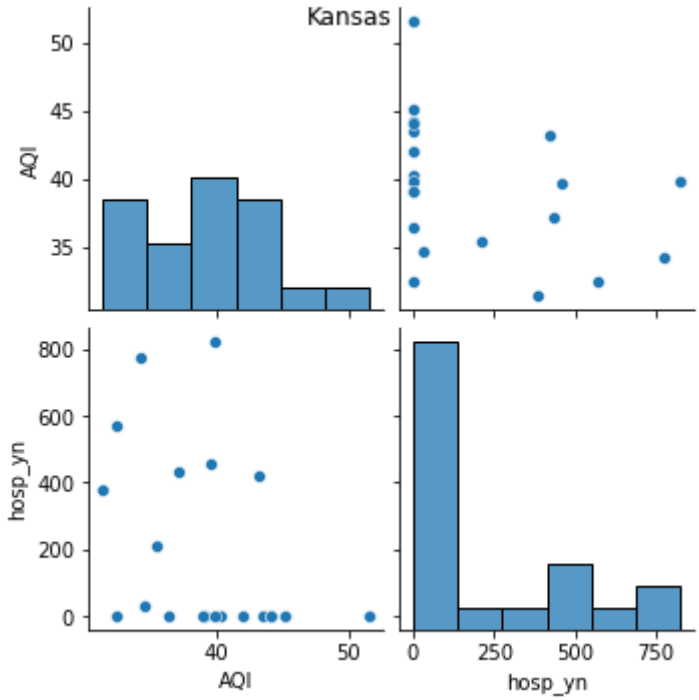

```
<Figure size 1080x504 with 0 Axes>
```



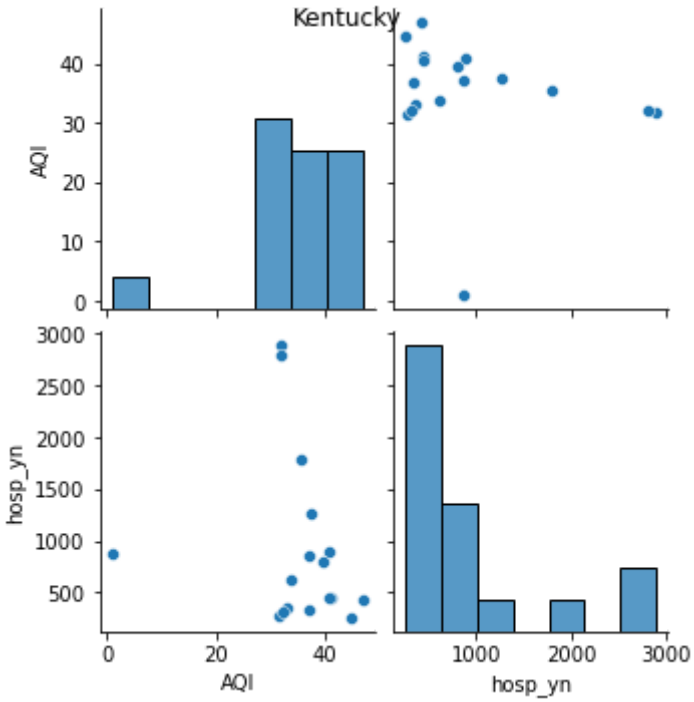

```
<Figure size 1080x504 with 0 Axes>
```



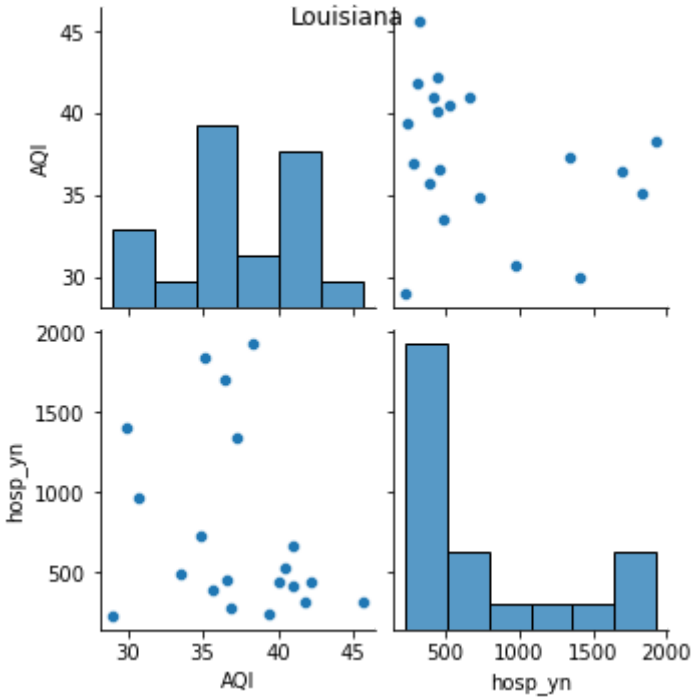

```
<Figure size 1080x504 with 0 Axes>
```



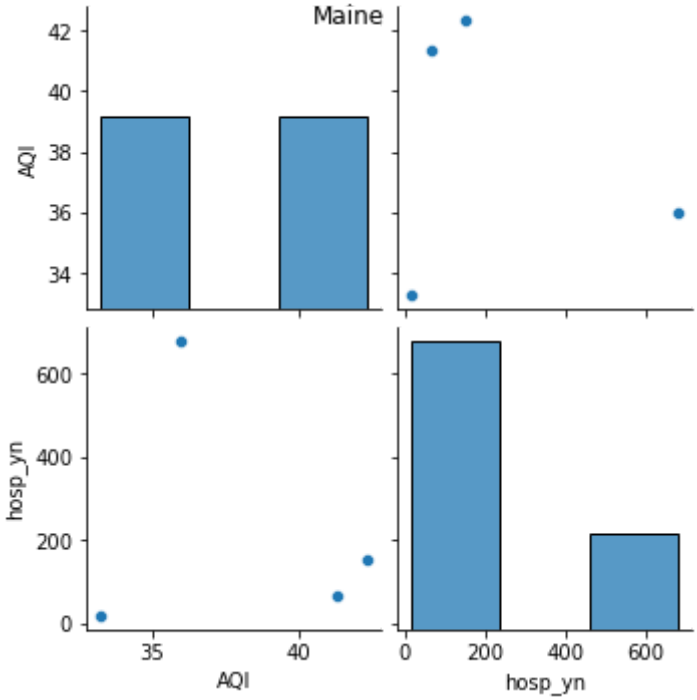

```
<Figure size 1080x504 with 0 Axes>
```



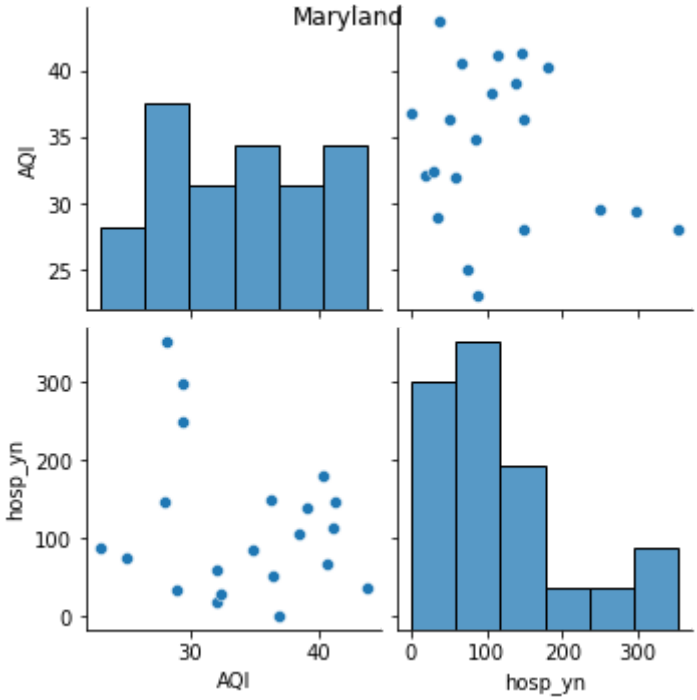

```
<Figure size 1080x504 with 0 Axes>
```



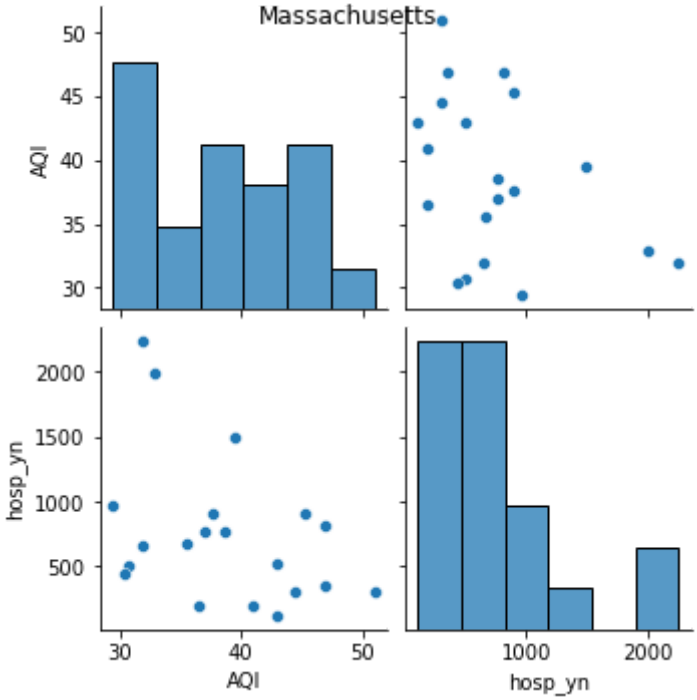

```
<Figure size 1080x504 with 0 Axes>
```



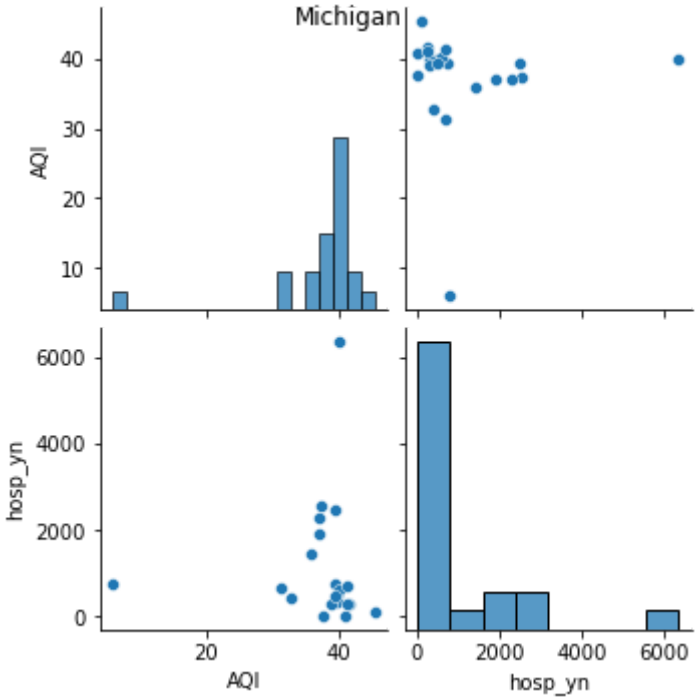

```
<Figure size 1080x504 with 0 Axes>
```



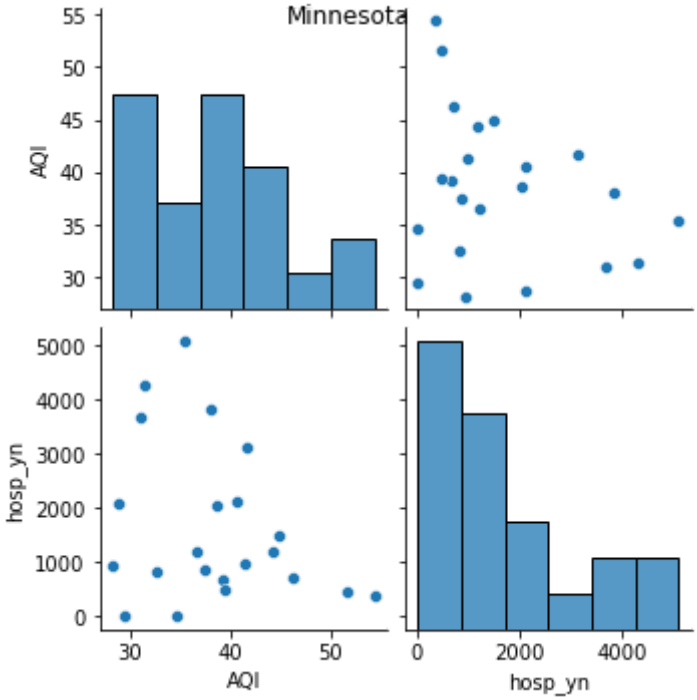

```
<Figure size 1080x504 with 0 Axes>
```



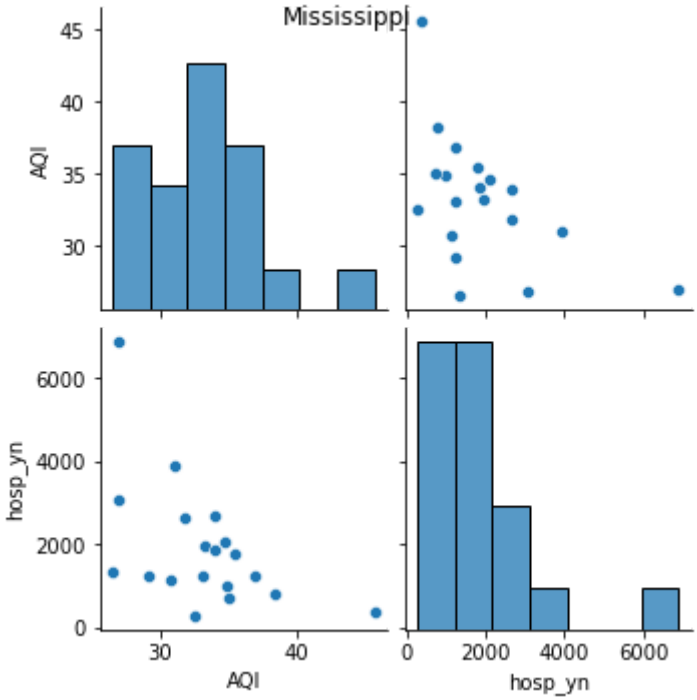

```
<Figure size 1080x504 with 0 Axes>
```



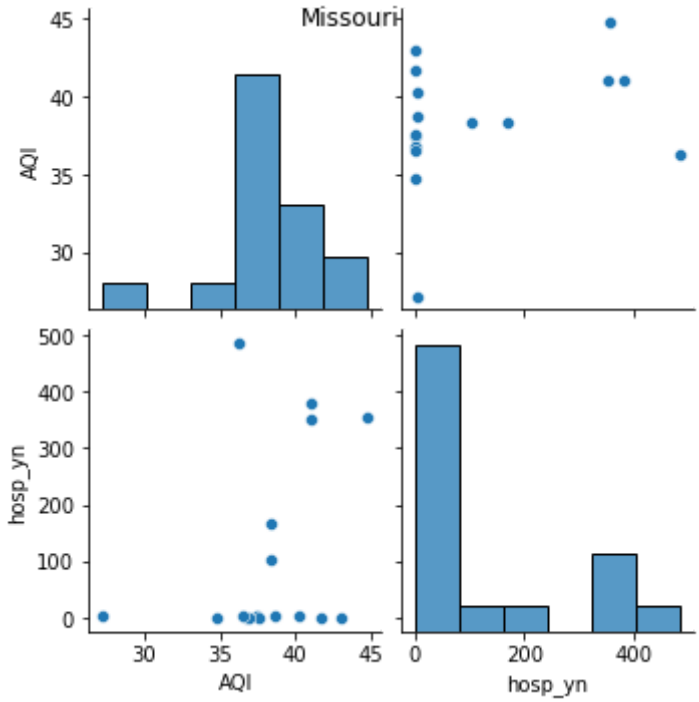

```
<Figure size 1080x504 with 0 Axes>
```



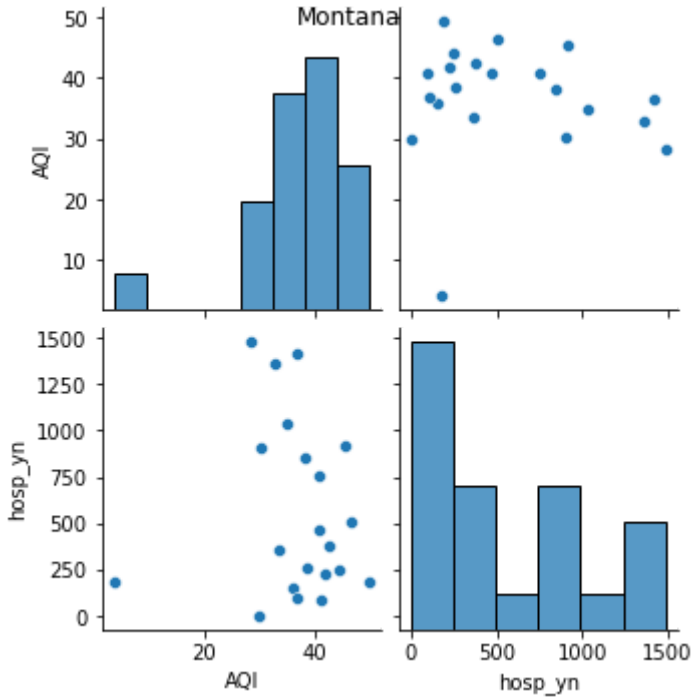

```
<Figure size 1080x504 with 0 Axes>
```



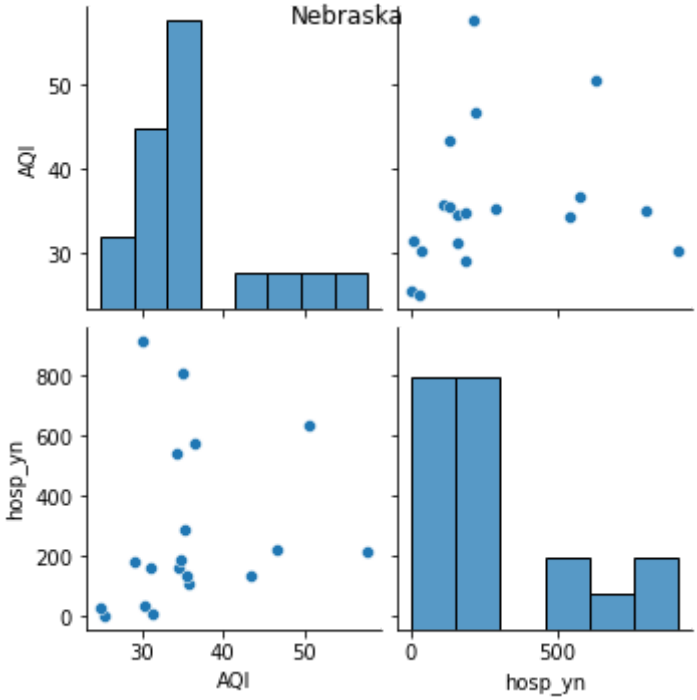

```
<Figure size 1080x504 with 0 Axes>
```



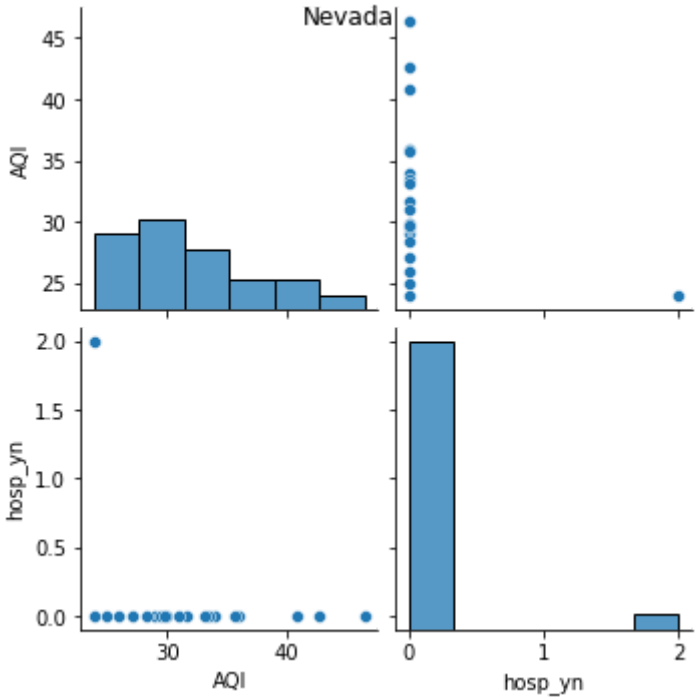

```
<Figure size 1080x504 with 0 Axes>
```



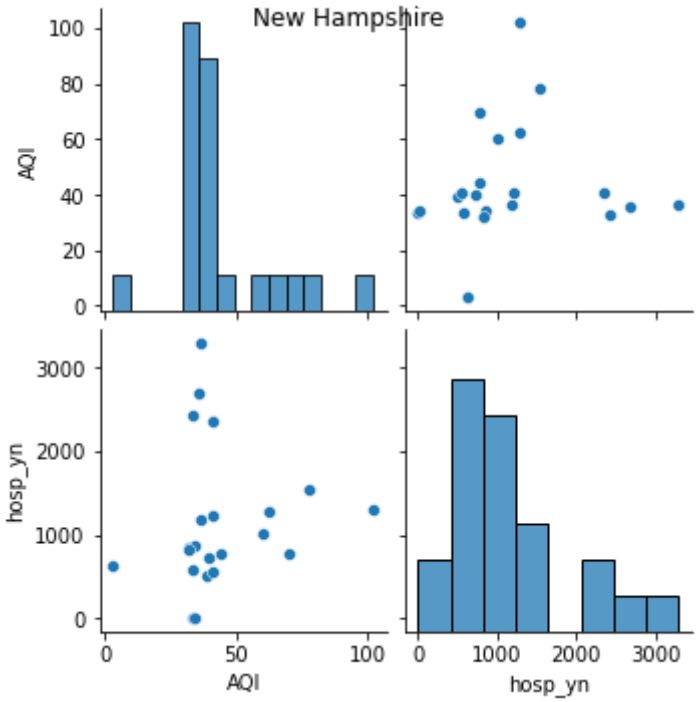

```
<Figure size 1080x504 with 0 Axes>
```



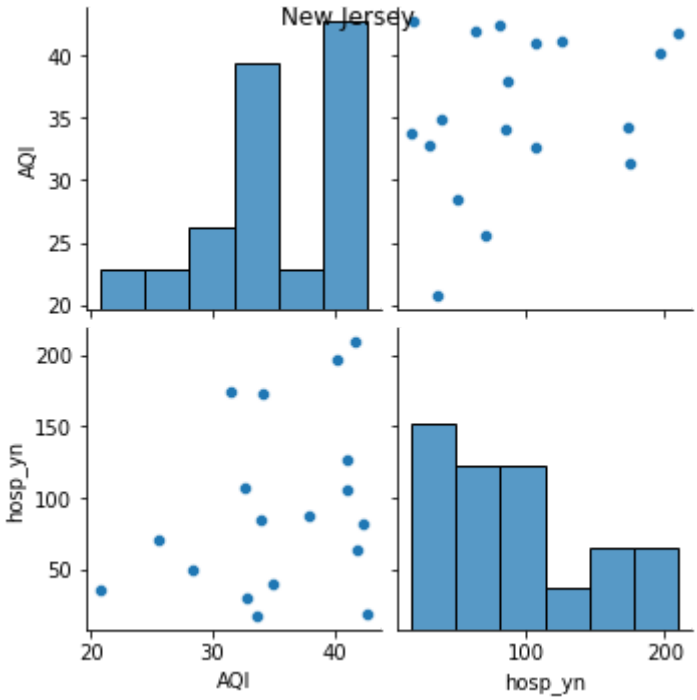

```
<Figure size 1080x504 with 0 Axes>
```



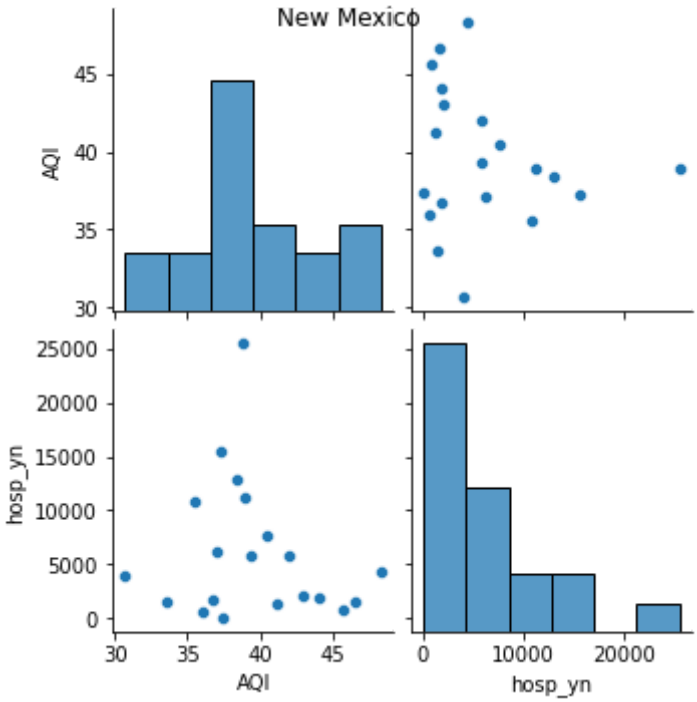

```
<Figure size 1080x504 with 0 Axes>
```



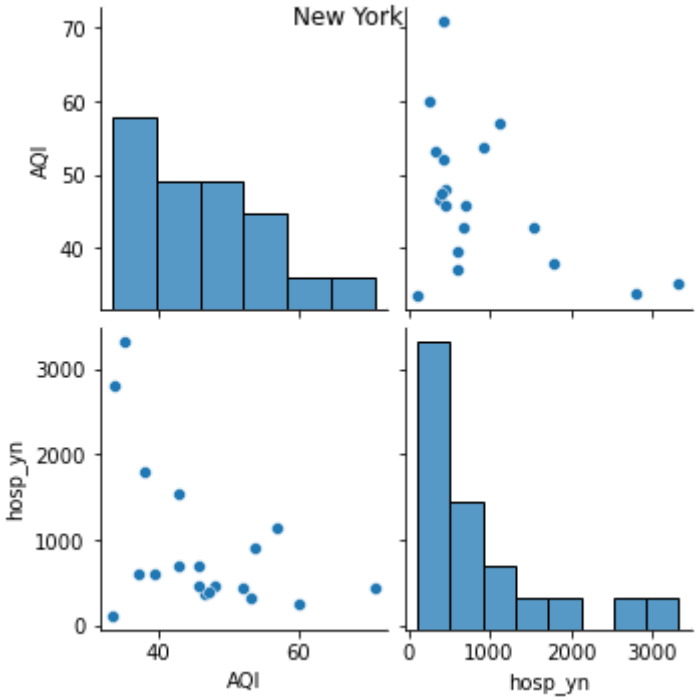

```
<Figure size 1080x504 with 0 Axes>
```



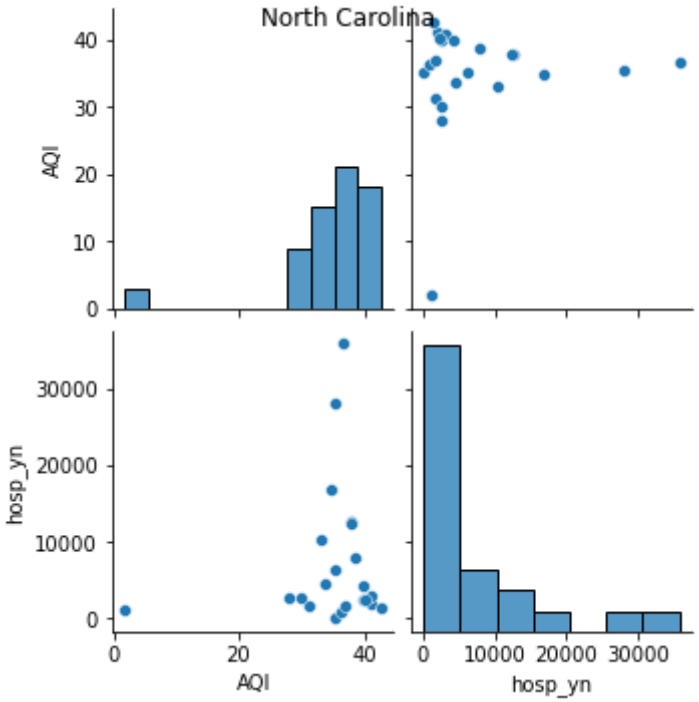

```
<Figure size 1080x504 with 0 Axes>
```



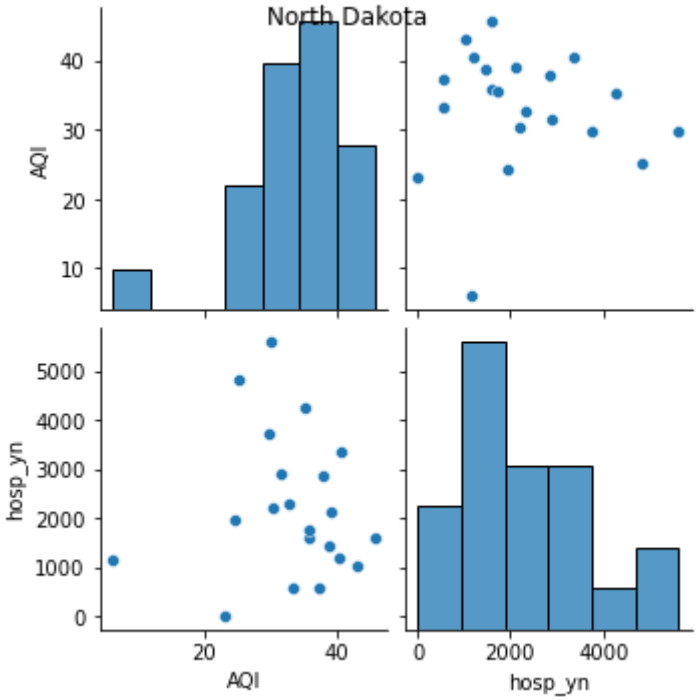

```
<Figure size 1080x504 with 0 Axes>
```



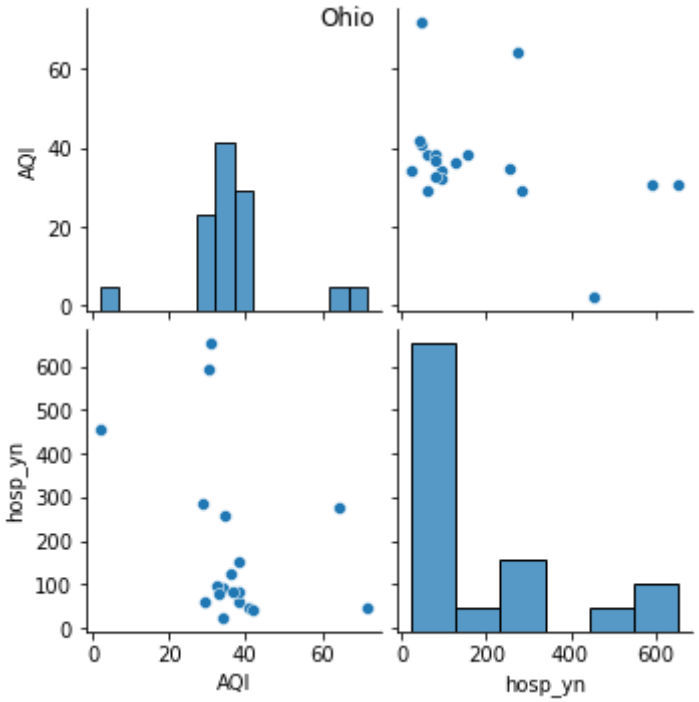

```
<Figure size 1080x504 with 0 Axes>
```



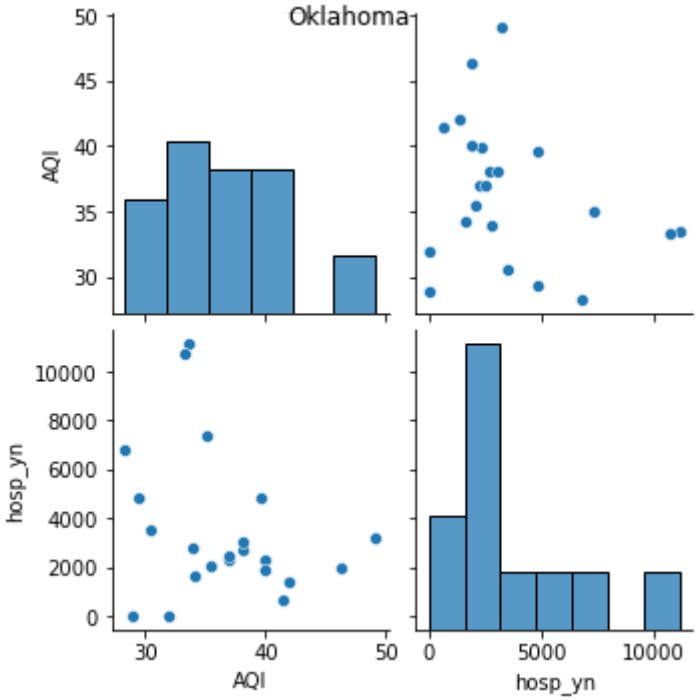

```
<Figure size 1080x504 with 0 Axes>
```



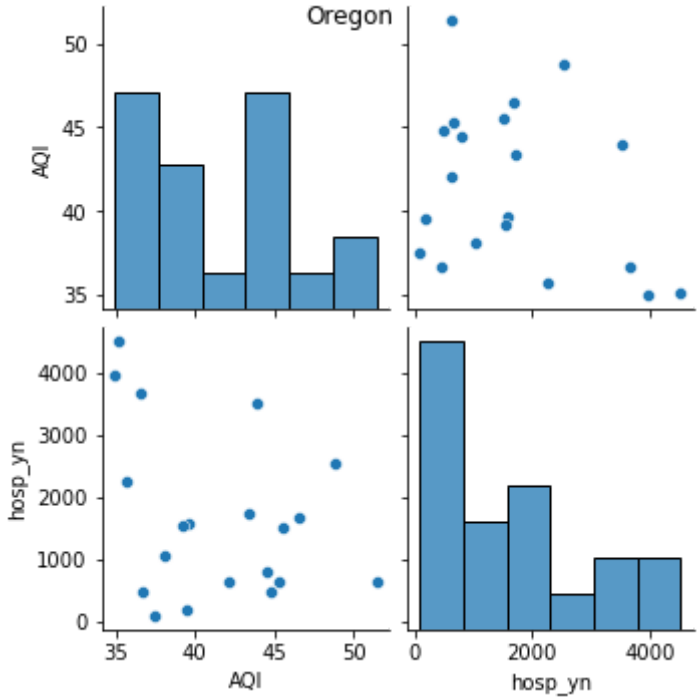

```
<Figure size 1080x504 with 0 Axes>
```



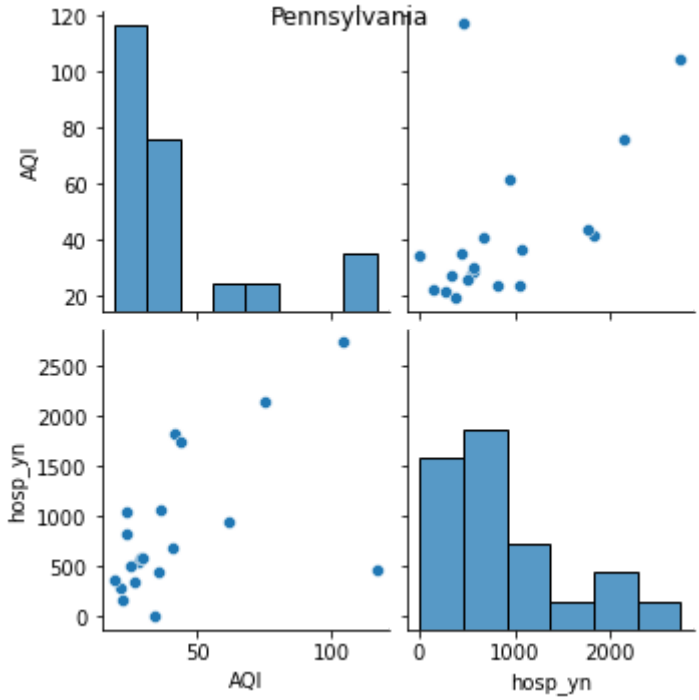

```
<Figure size 1080x504 with 0 Axes>
```



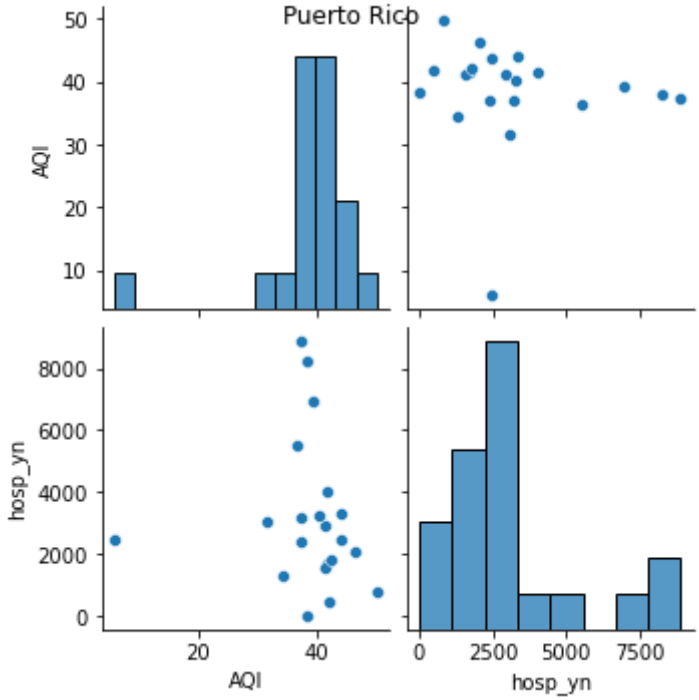

```
<Figure size 1080x504 with 0 Axes>
```



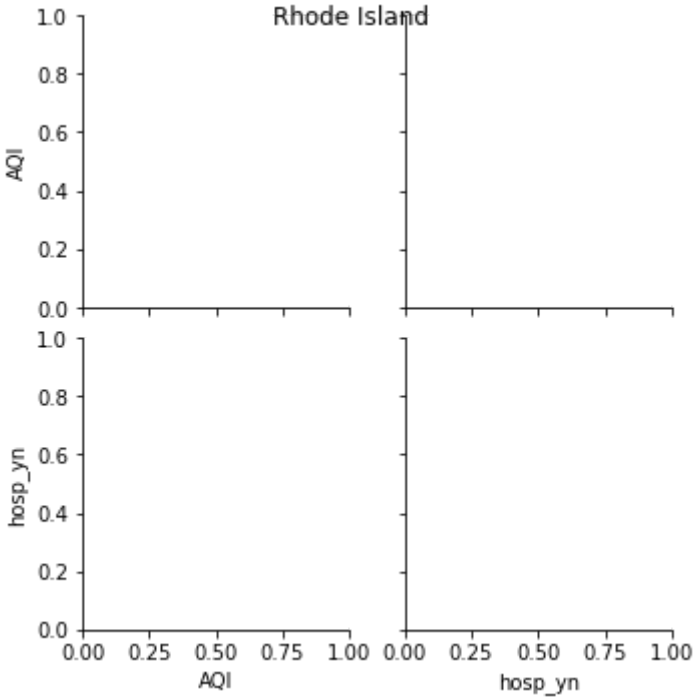

```
<Figure size 1080x504 with 0 Axes>
```



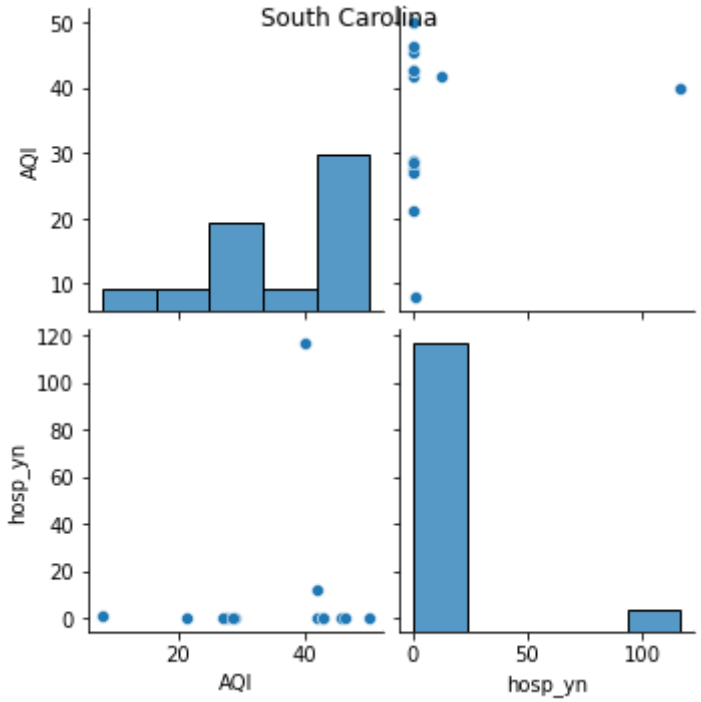

```
<Figure size 1080x504 with 0 Axes>
```



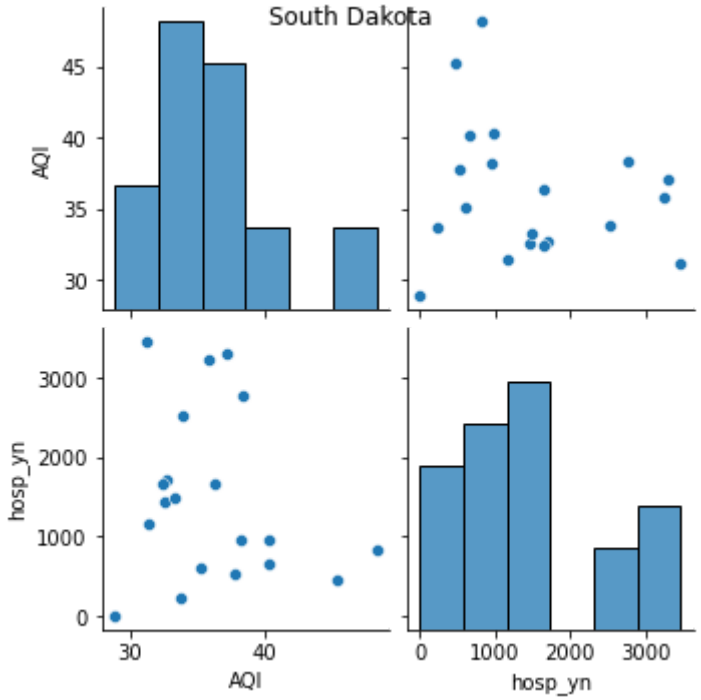

```
<Figure size 1080x504 with 0 Axes>
```



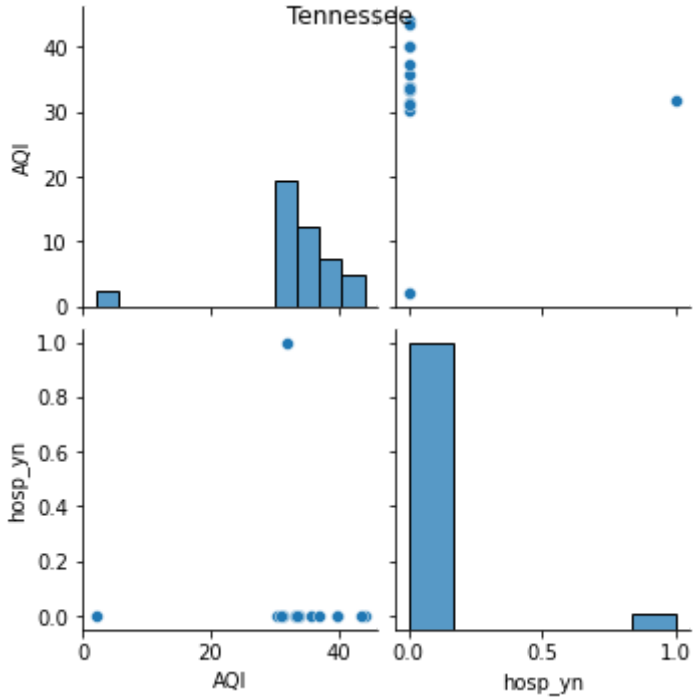

```
<Figure size 1080x504 with 0 Axes>
```



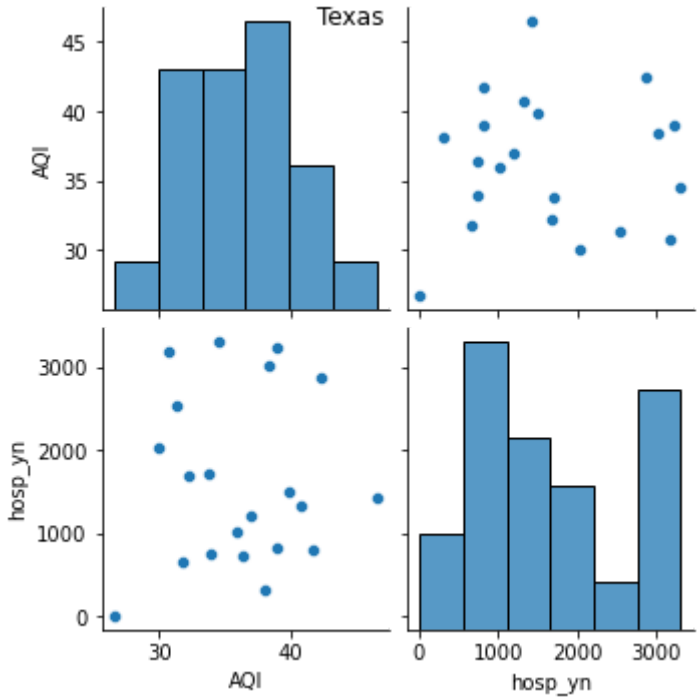

```
<Figure size 1080x504 with 0 Axes>
```



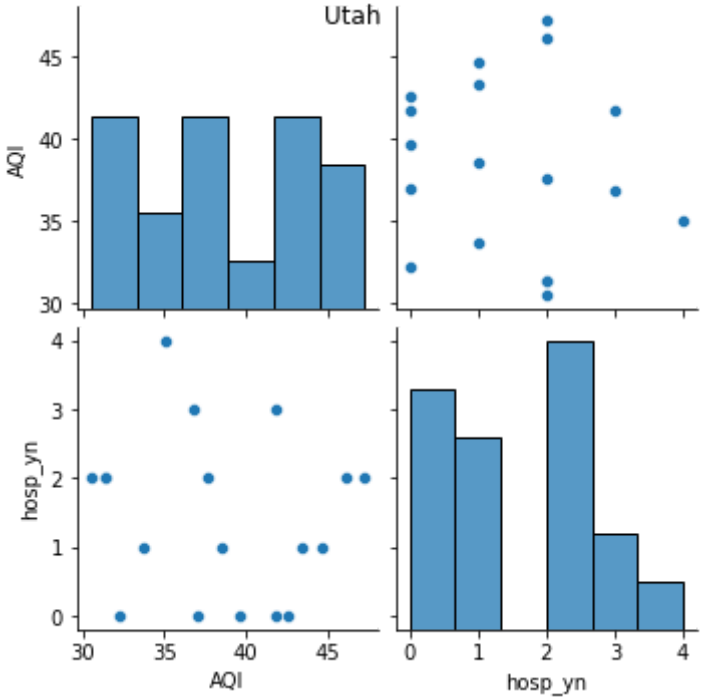

```
<Figure size 1080x504 with 0 Axes>
```



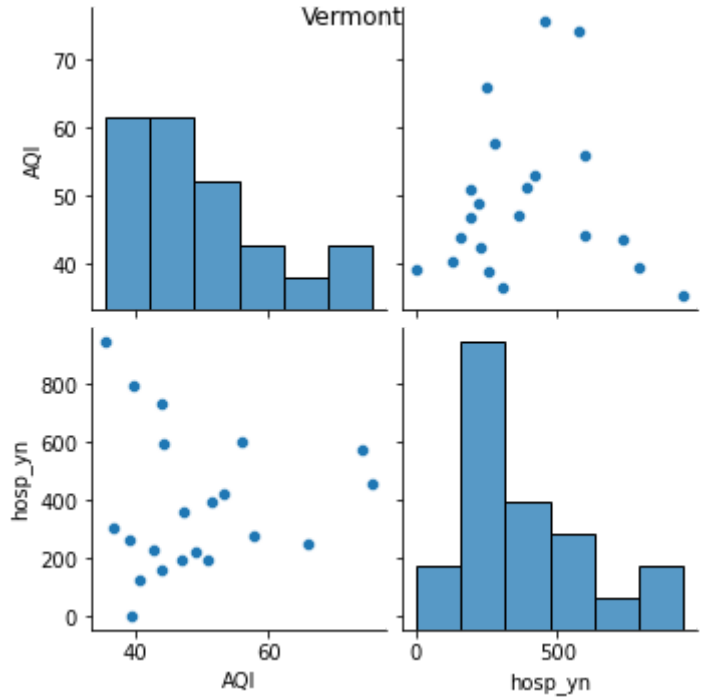

```
<Figure size 1080x504 with 0 Axes>
```



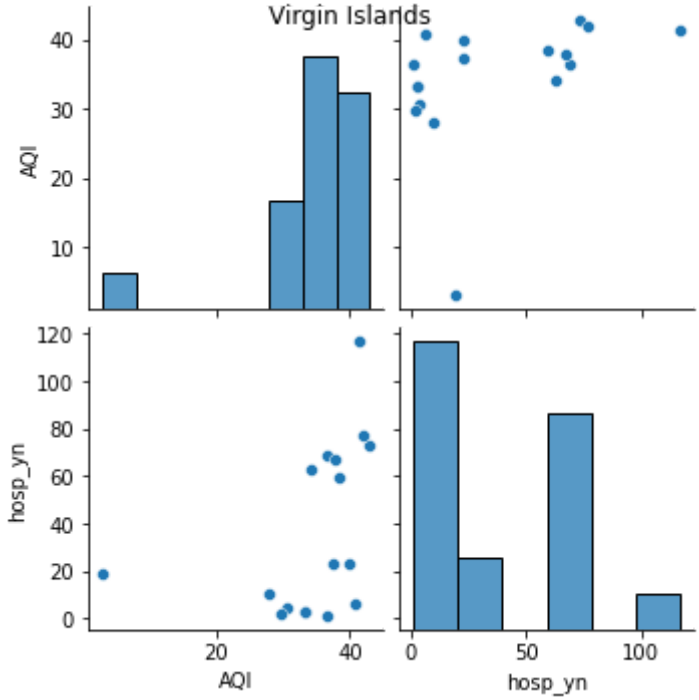

```
<Figure size 1080x504 with 0 Axes>
```



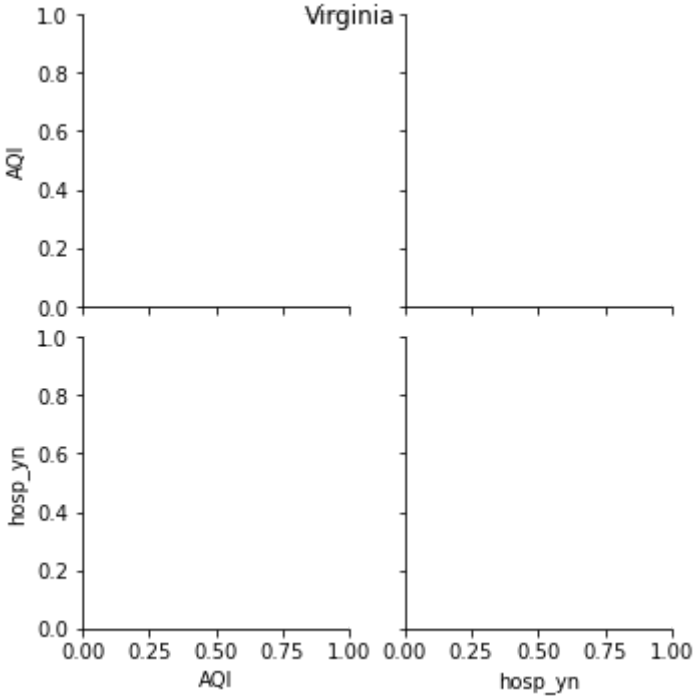

```
<Figure size 1080x504 with 0 Axes>
```



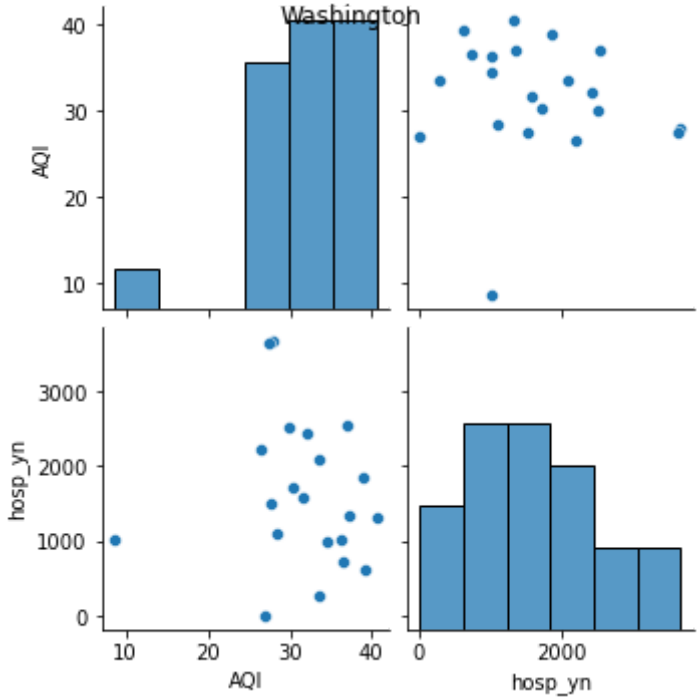

```
<Figure size 1080x504 with 0 Axes>
```



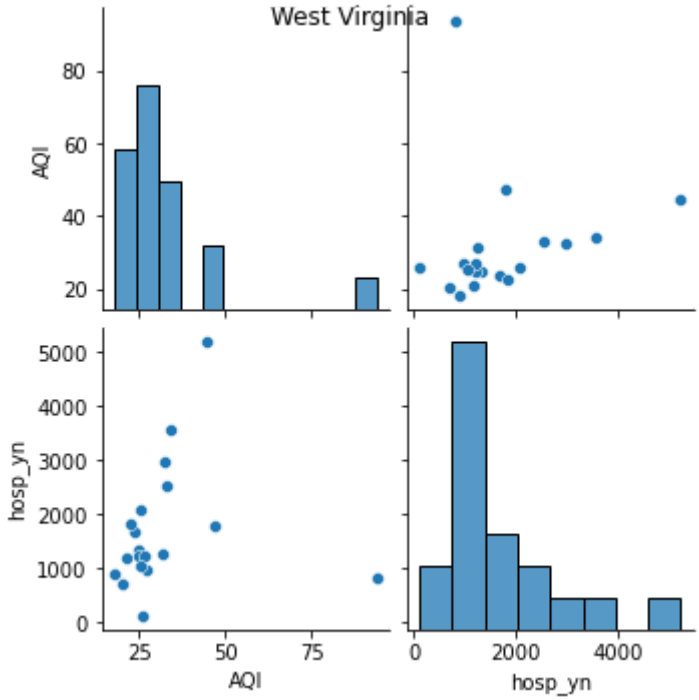

```
<Figure size 1080x504 with 0 Axes>
```



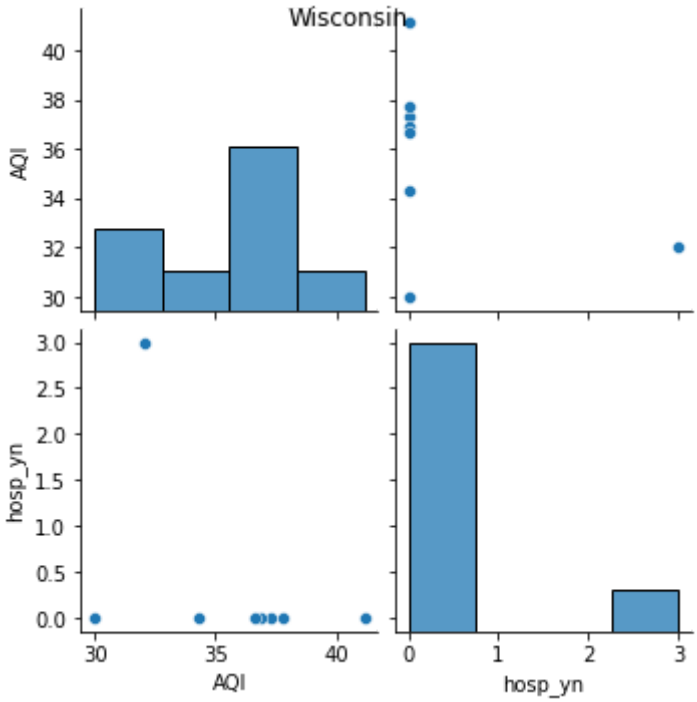

```
<Figure size 1080x504 with 0 Axes>
```



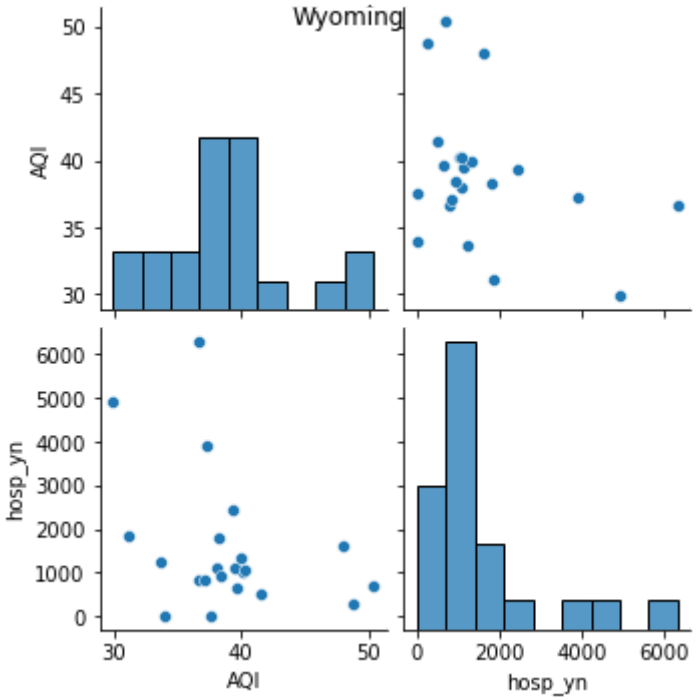

```
<Figure size 1080x504 with 0 Axes>
```



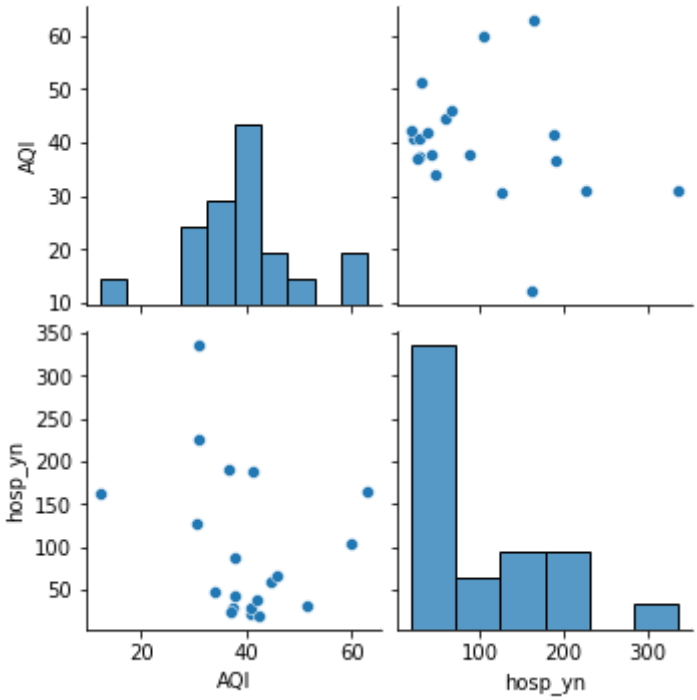

```
<Figure size 1080x504 with 0 Axes>
```




```
<Figure size 1080x504 with 0 Axes>
```




```
<Figure size 1080x504 with 0 Axes>
```



In [68]:
```
## For the above graphs Each graph has its own ""Title"" with State Name
```

Is there a correlation between Air Quality Index (AQI) and COVID-19 hospitalization rate?

Ans:

Yes there is a corrrelation between AQI and hosp_yn,

For State Code: {44, 41, 36, 25, 20, 21, 18, 8, 6, 5, 2} which represents State Name: {Rhode Island, Oregon, New York, Massachusetts, Kansas, Kentucky, Indiana, Colorado, California, } is directly correlated with AQI and Hospy_yn

For few States Code: {1, 5, 26, 27, 42, 39, 45, 48, 17, 19 } which represents State Name: {Alabama, Arkansas, Minesota, Michigan, Missouri, Montana, Nebraska, Ohio, Pennsylvania, South Carolina, Texas, Iova, Illinois } is not directly correleated with AQI. There could be several reasons why the correlation between AQI and hospitalization rate is not direct for certain states. Some possible explanations include:

1) Other factors, such as symptom_status, age distribution{Age_group}, and access to healthcare, may play a larger role in determining hospitalization rates in those states.

2) The data used in the analysis may not be comprehensive enough to accurately capture the relationship between AQI and hospitalization rate in those states.

3) There may be differences in how AQI is measured or reported in those states, leading to discrepancies in the data.

For Reference: The Above Answer is Based on the heat plots of AQI vs Hosp_yn for each state code|

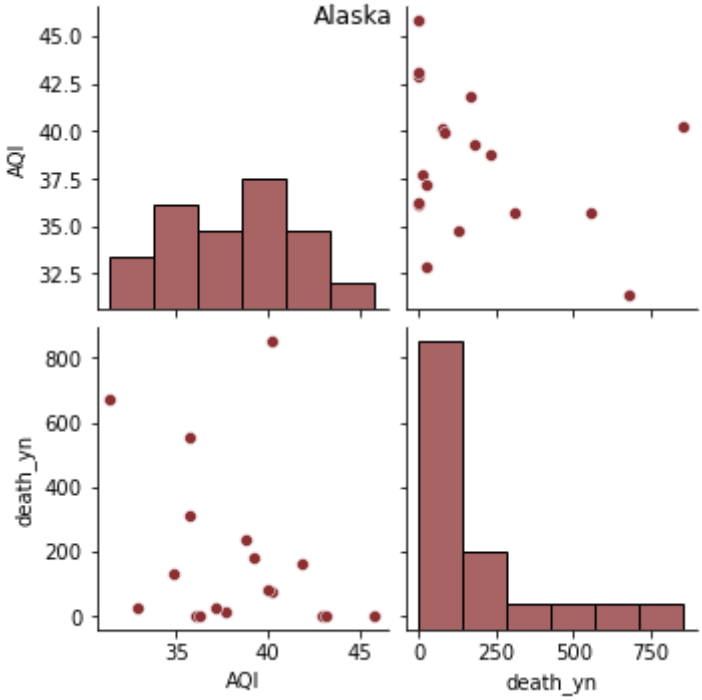
{Final Summary on Correlation}

The analysis of the heatmap graph suggests that there is a correlation between AQI and COVID-19 hospitalization rate. The data shows that for certain state codes, the AQI is directly correlated with the hospitalization rate. However, for other states, the correlation is not as clear. The Heat graph displays a total of 12 stacks, indicating that the data covers a 12 month period, or 8 months in the case of 8 stacks. Overall, it can be inferred that AQI may play a role in the hospitalization rate of COVID-19 patients in some states.

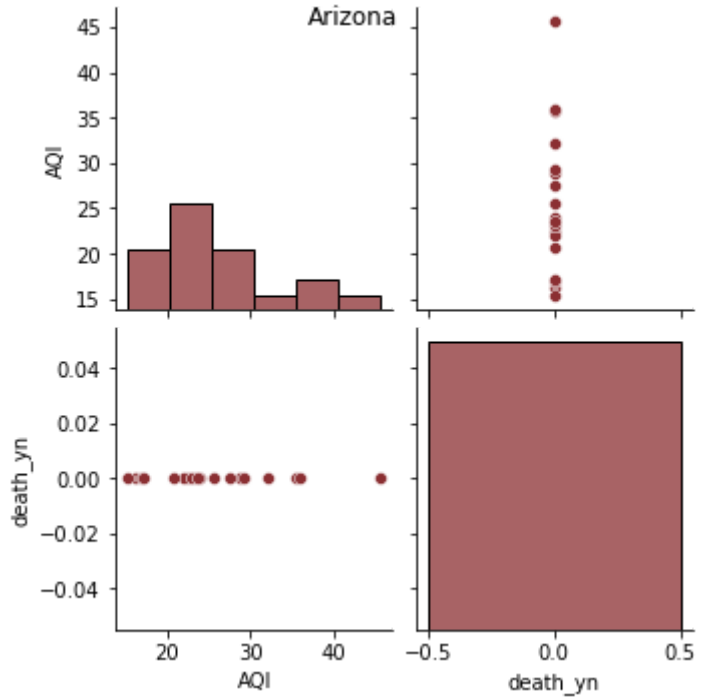
Analyzing other Co-founding Factors

In [69]:
```
for country in country_list:
    cn=[]
    cn.append(country)
    df_plotter = pd.DataFrame()
    df_plotter=merged_df2[merged_df2['State_Name'].isin(cn)]
    t1 = df_plotter[['AQI', 'death_yn']]
    plt.suptitle(country)
    sns.set_palette("Reds_r", desat=0.6)
    plt.figure(figsize=(15,7))
    sns.pairplot(t1,diag_kind='hist')
```

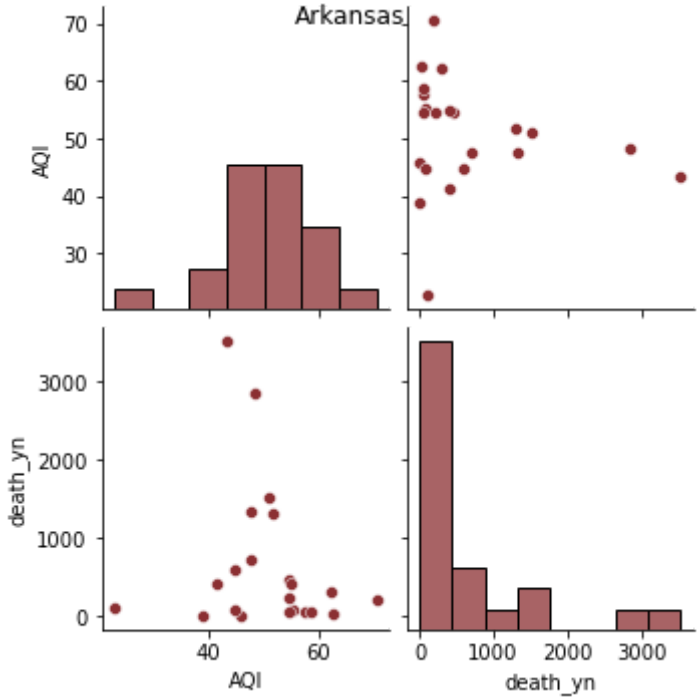
```
<Figure size 432x288 with 0 Axes>
<Figure size 1080x504 with 0 Axes>
```



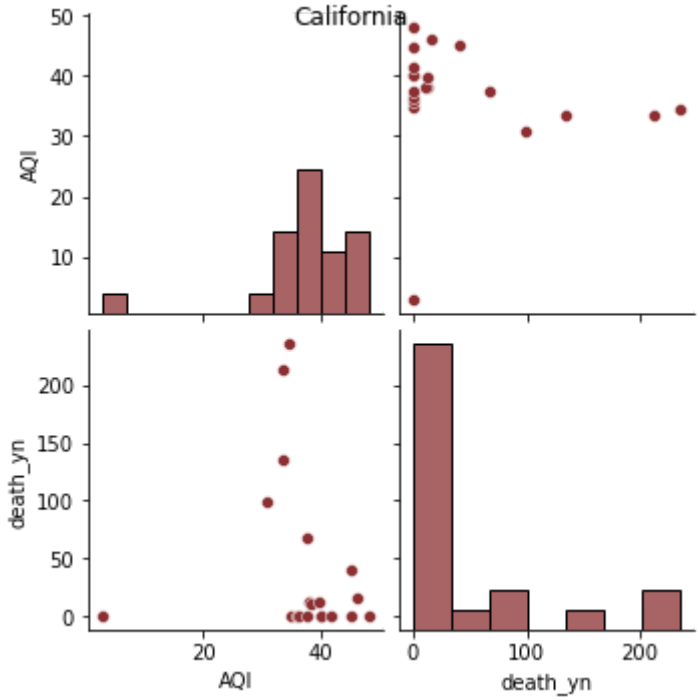

```
<Figure size 1080x504 with 0 Axes>
```



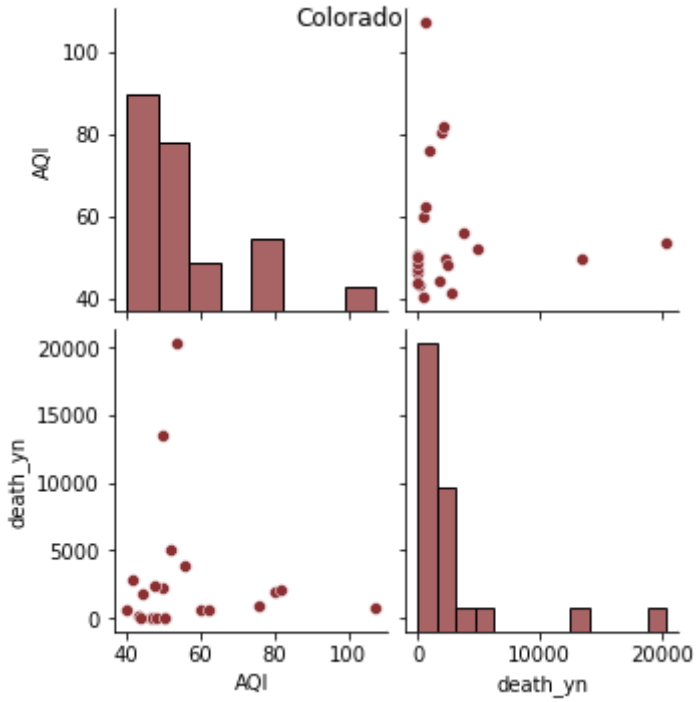

```
<Figure size 1080x504 with 0 Axes>
```



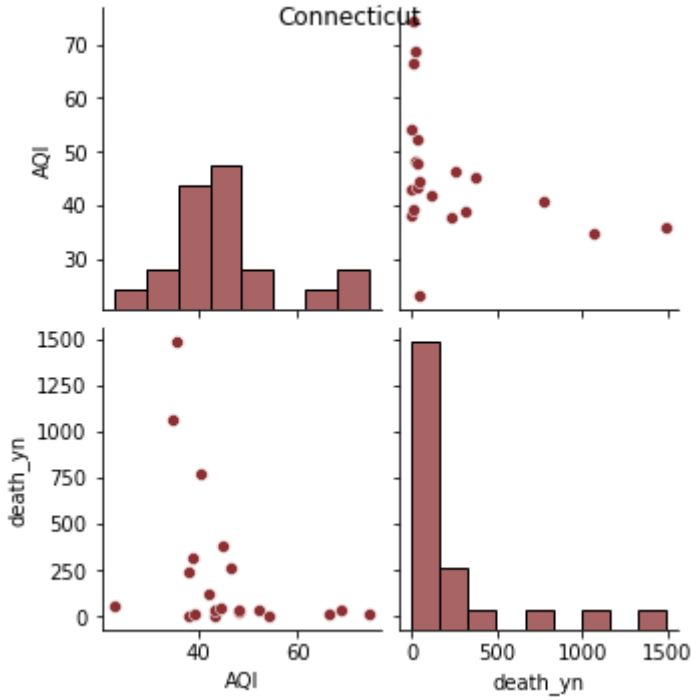

```
<Figure size 1080x504 with 0 Axes>
```



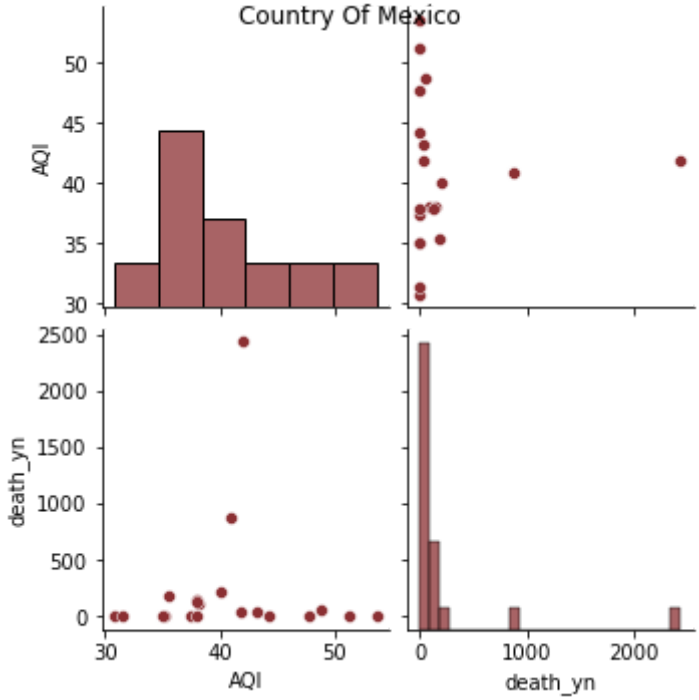

```
<Figure size 1080x504 with 0 Axes>
```



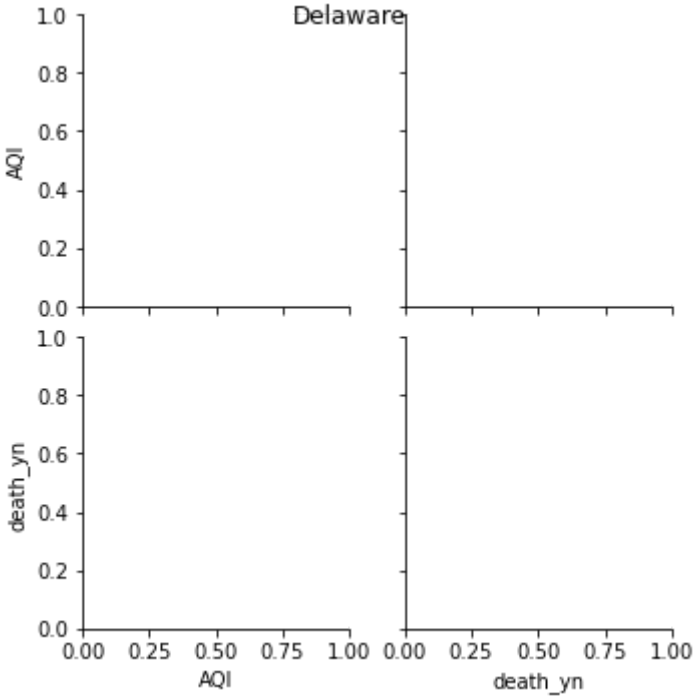

```
<Figure size 1080x504 with 0 Axes>
```



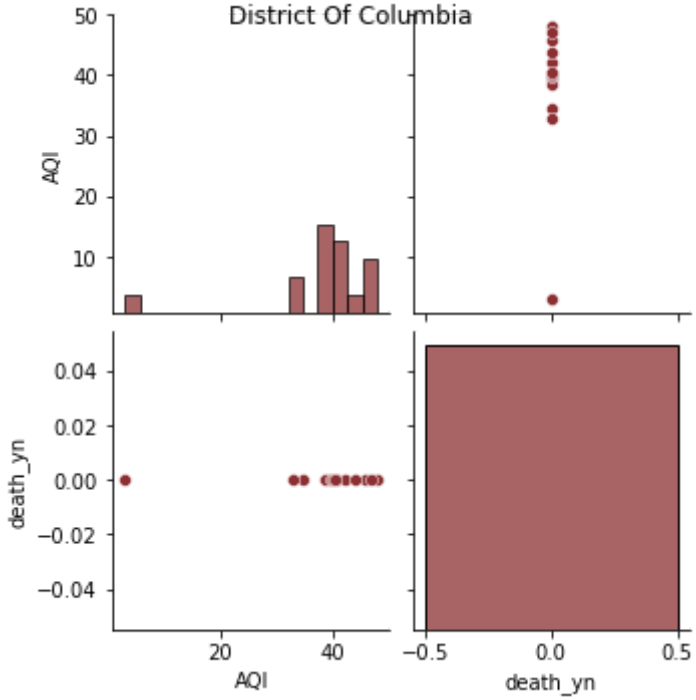

```
<Figure size 1080x504 with 0 Axes>
```



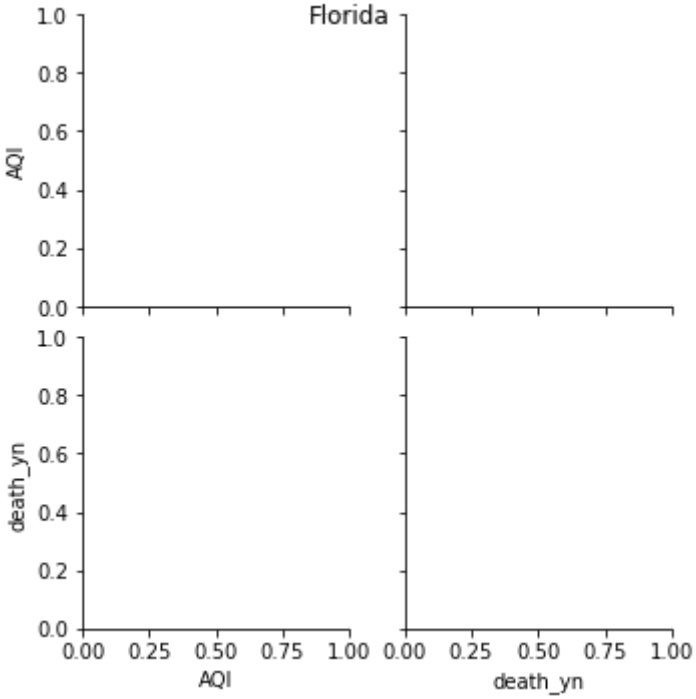

```
<Figure size 1080x504 with 0 Axes>
```



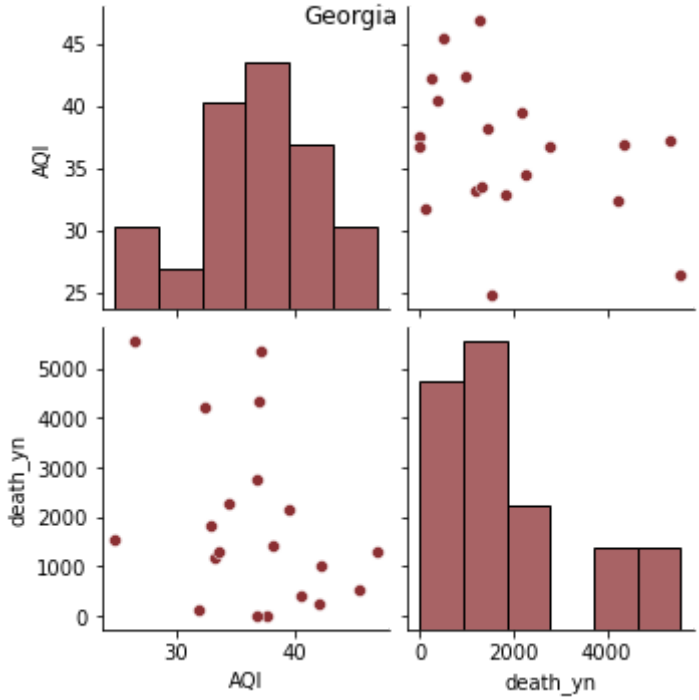

```
<Figure size 1080x504 with 0 Axes>
```



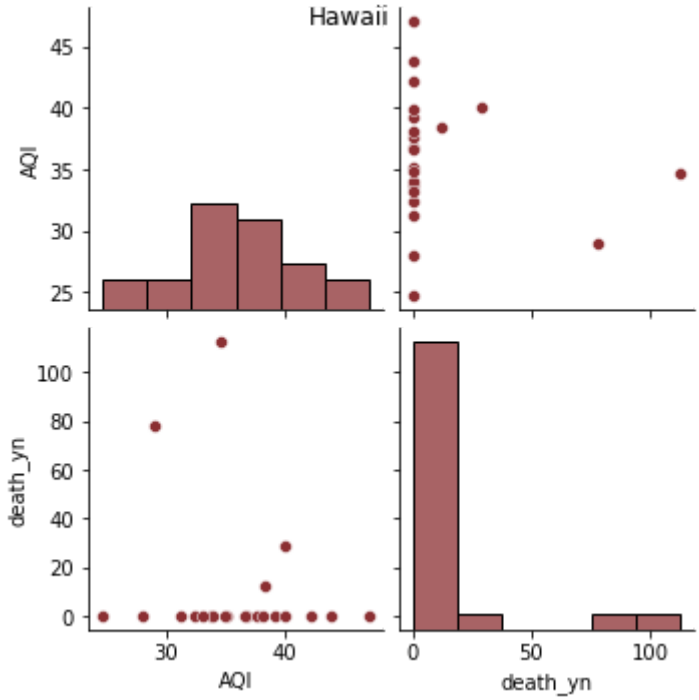

```
<Figure size 1080x504 with 0 Axes>
```



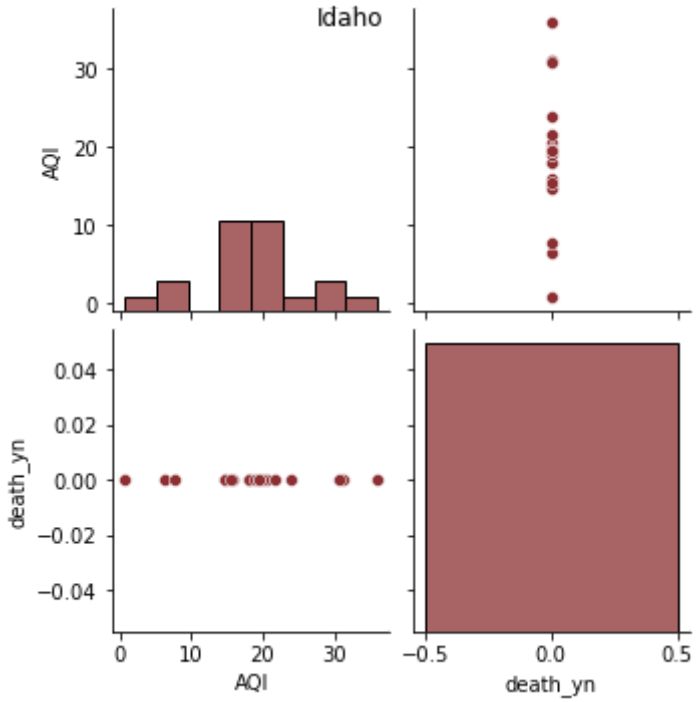

```
<Figure size 1080x504 with 0 Axes>
```



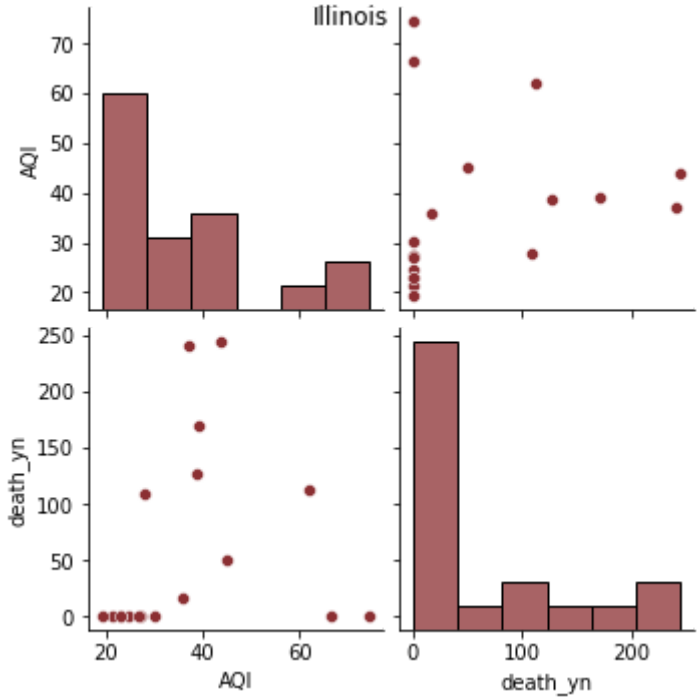

```
<Figure size 1080x504 with 0 Axes>
```



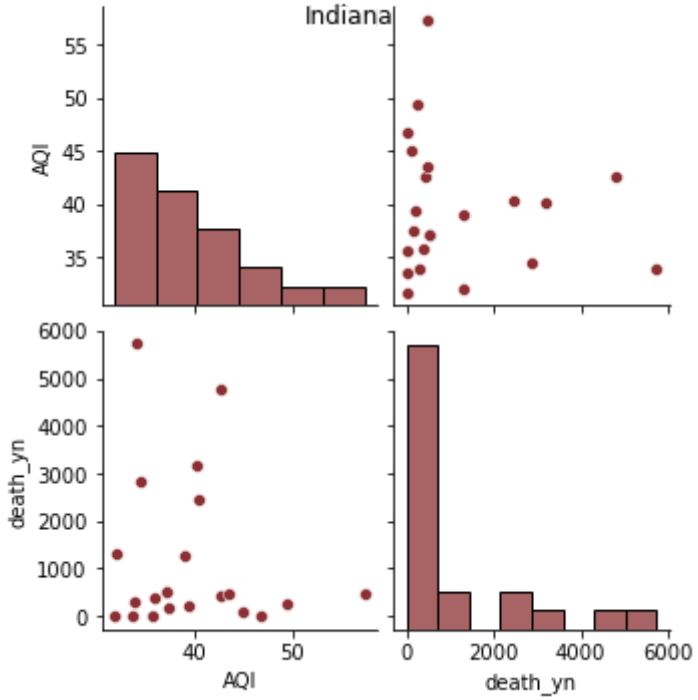

```
<Figure size 1080x504 with 0 Axes>
```



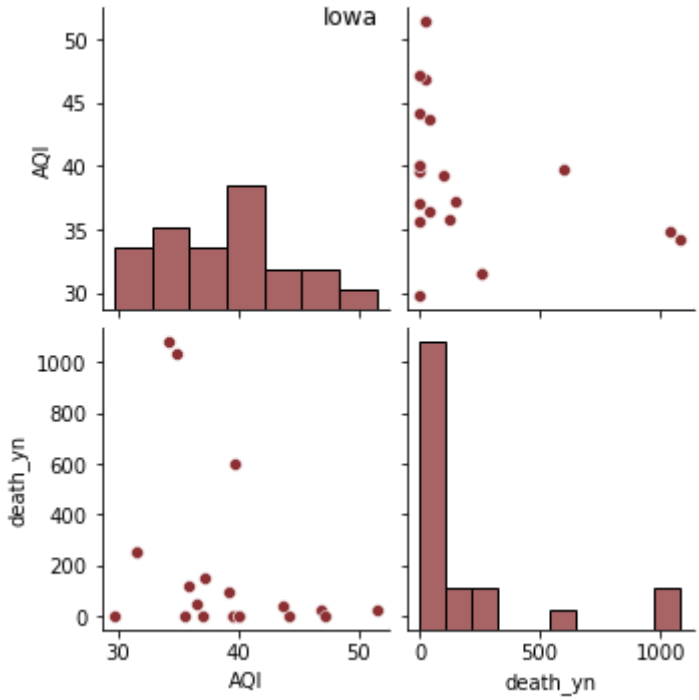

```
<Figure size 1080x504 with 0 Axes>
```



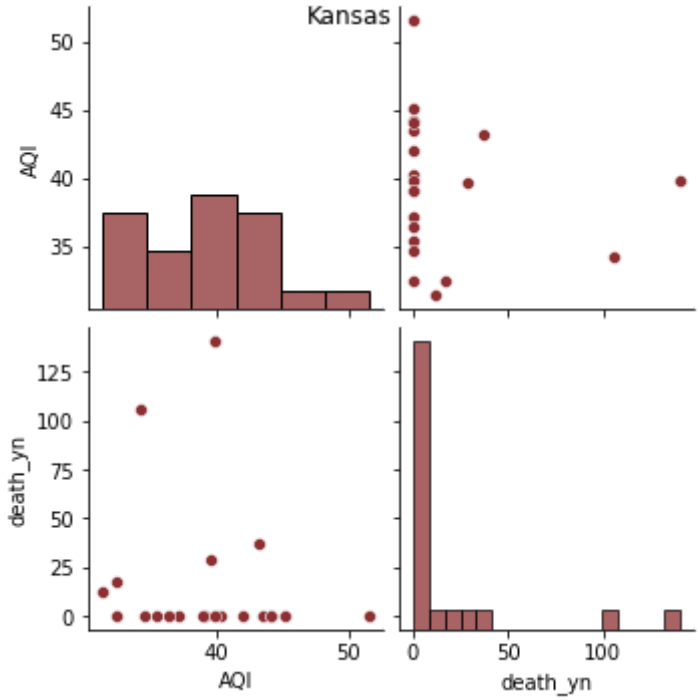

```
<Figure size 1080x504 with 0 Axes>
```



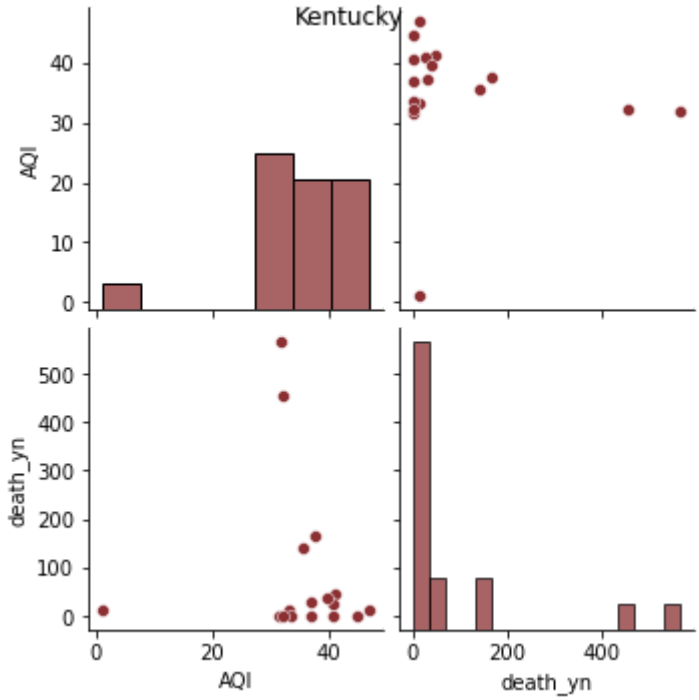

```
<Figure size 1080x504 with 0 Axes>
```



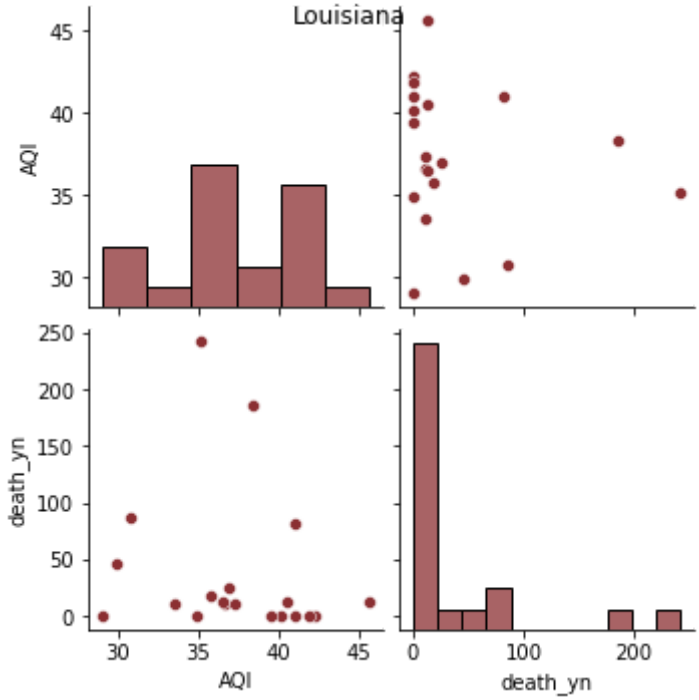

```
<Figure size 1080x504 with 0 Axes>
```



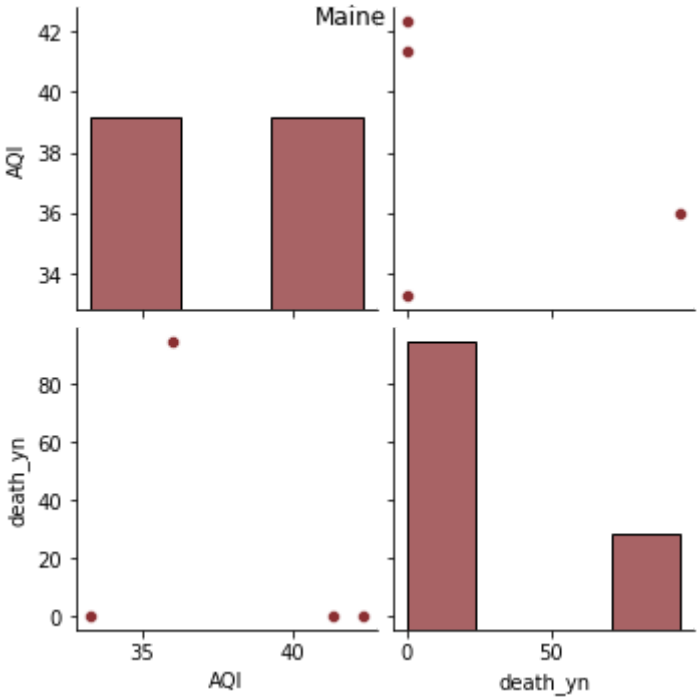

```
<Figure size 1080x504 with 0 Axes>
```



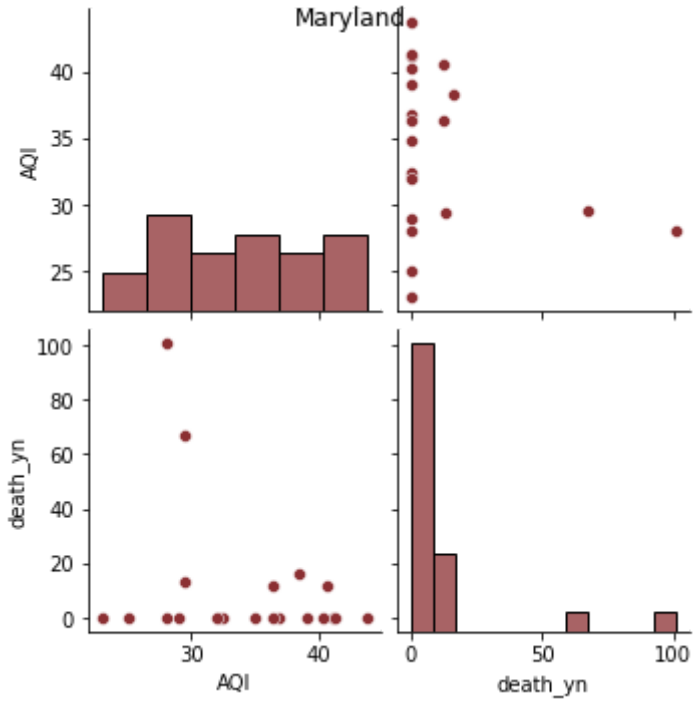

```
<Figure size 1080x504 with 0 Axes>
```



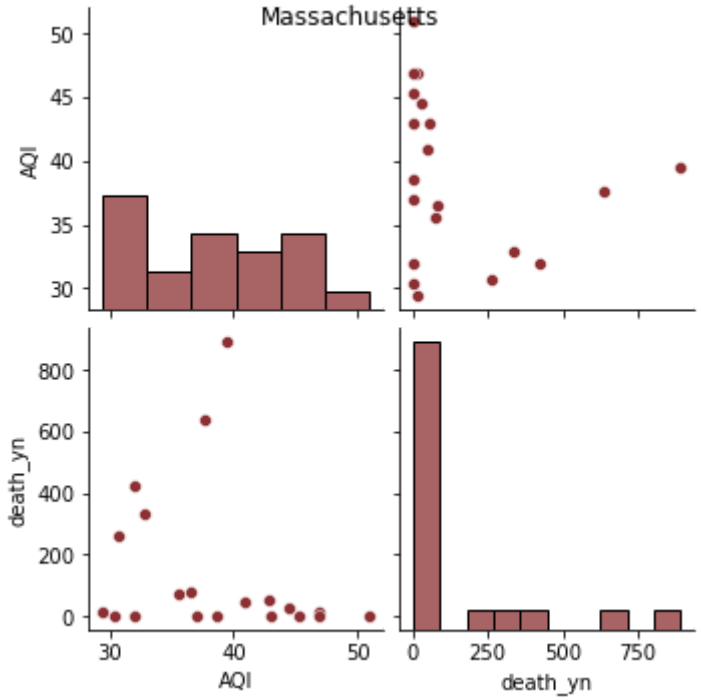

```
<Figure size 1080x504 with 0 Axes>
```



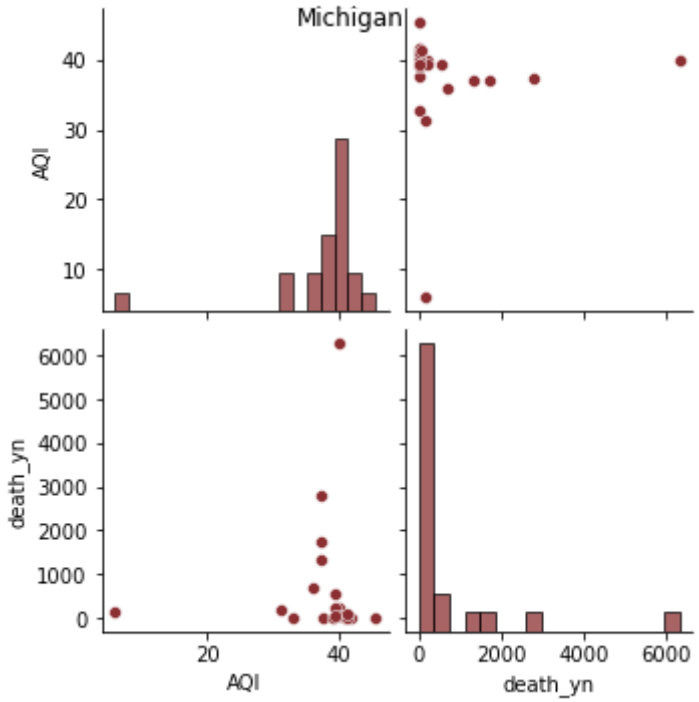

```
<Figure size 1080x504 with 0 Axes>
```



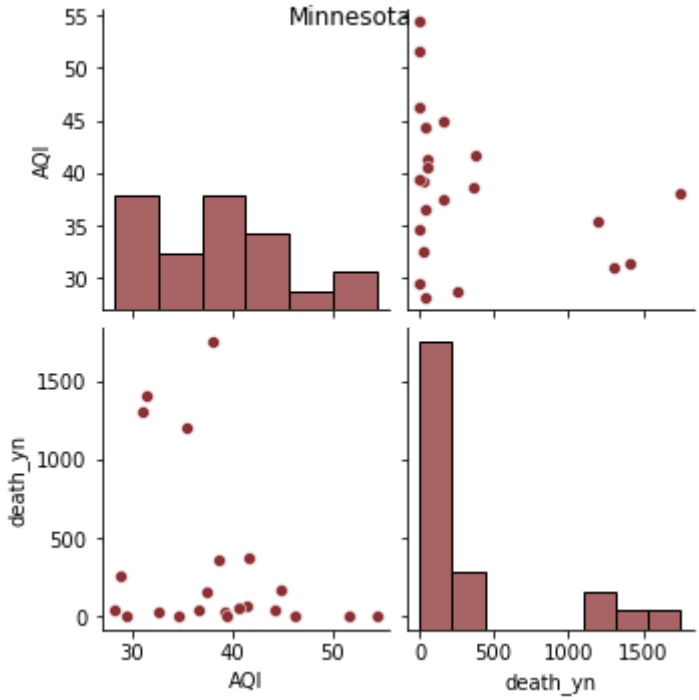

```
<Figure size 1080x504 with 0 Axes>
```



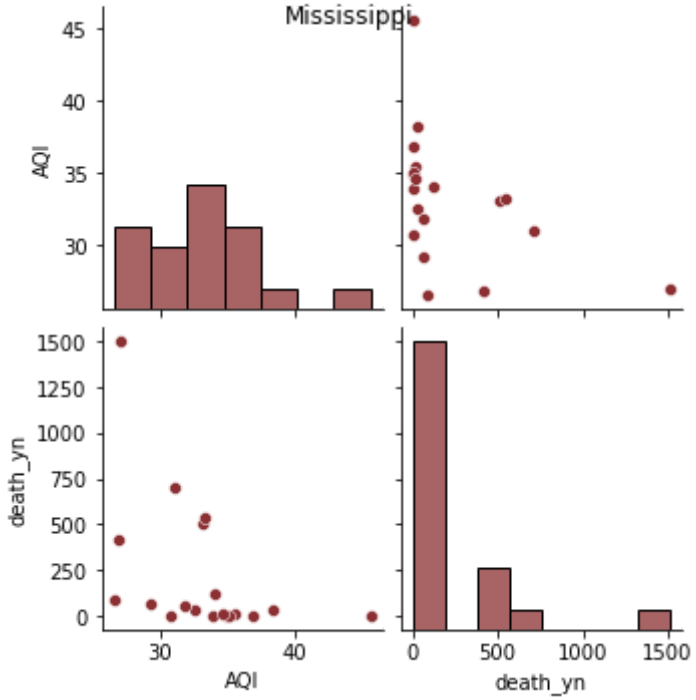

```
<Figure size 1080x504 with 0 Axes>
```



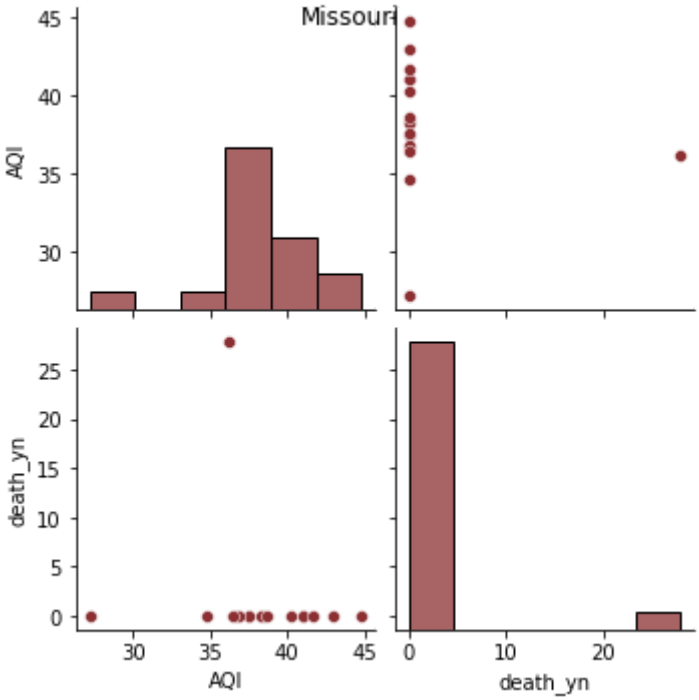

```
<Figure size 1080x504 with 0 Axes>
```



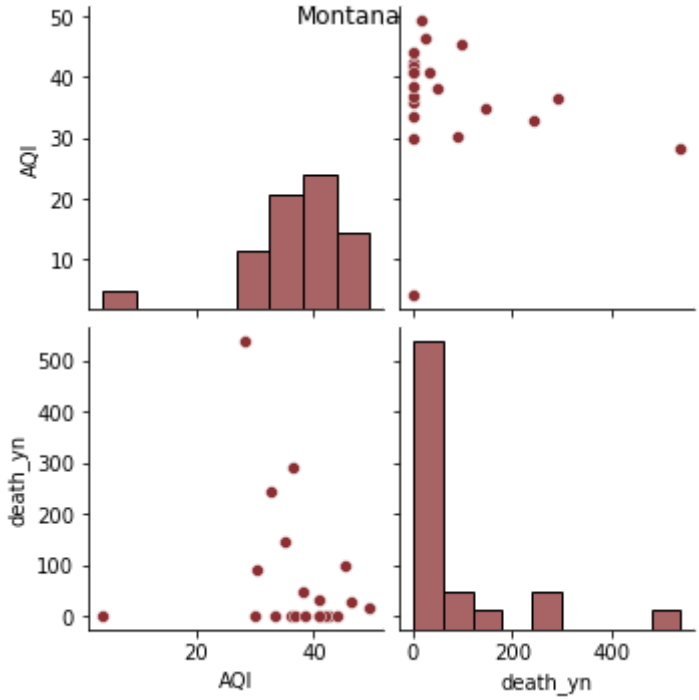

```
<Figure size 1080x504 with 0 Axes>
```



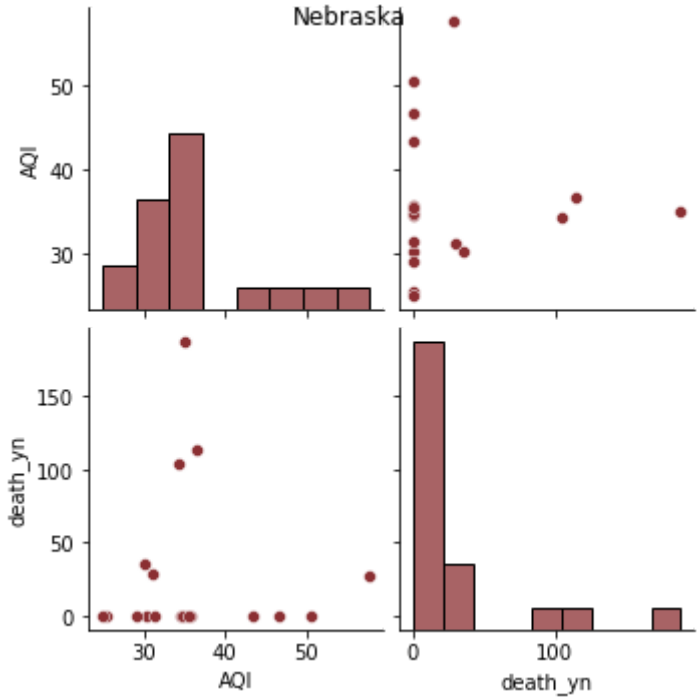
<Figure size 1080x504 with 0 Axes>



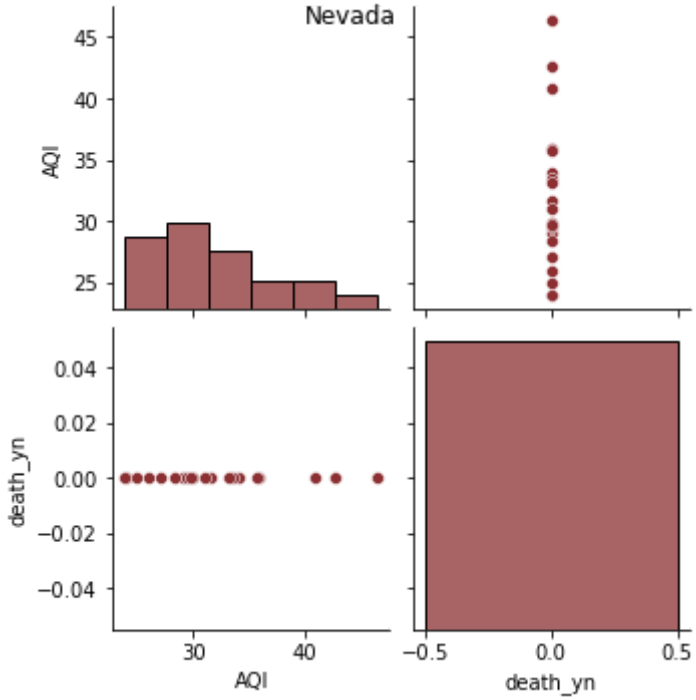
<Figure size 1080x504 with 0 Axes>



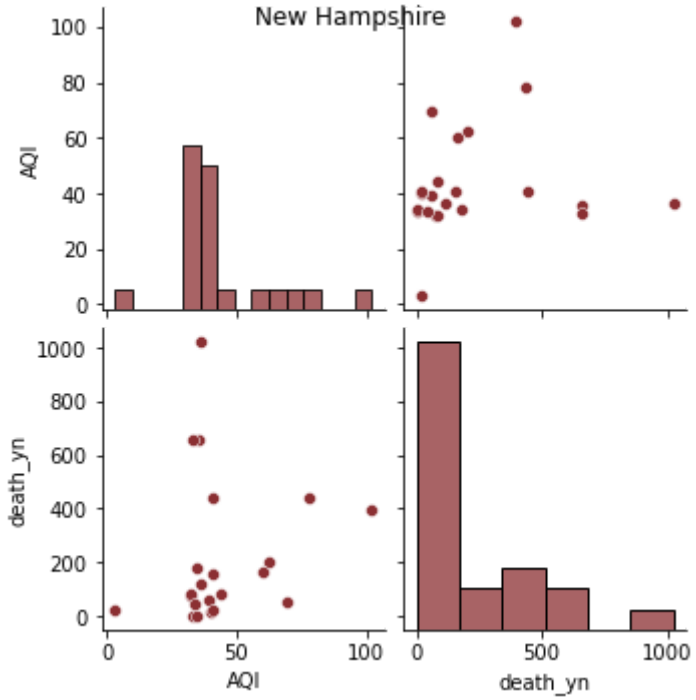

```
<Figure size 1080x504 with 0 Axes>
```



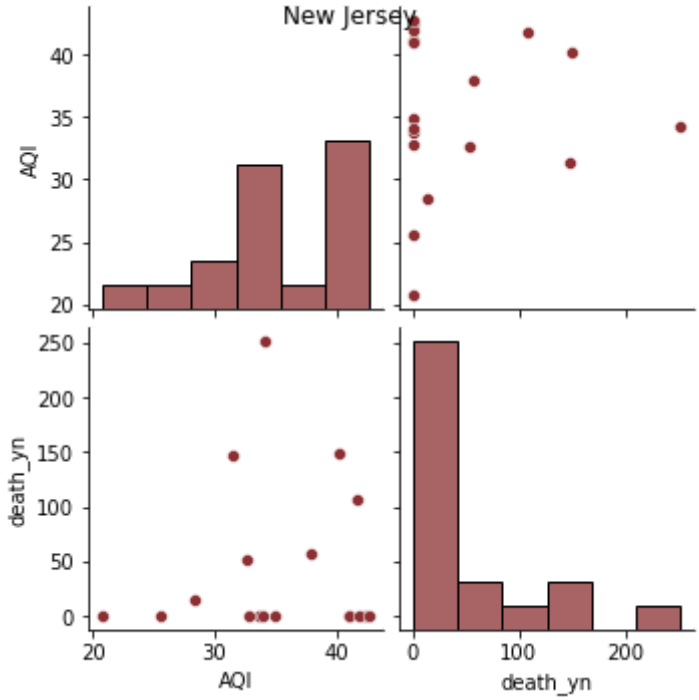

```
<Figure size 1080x504 with 0 Axes>
```



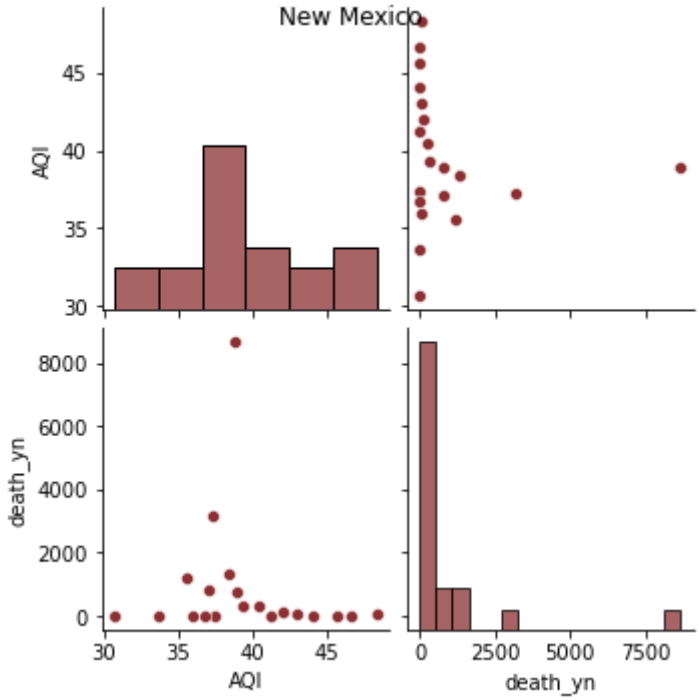

```
<Figure size 1080x504 with 0 Axes>
```



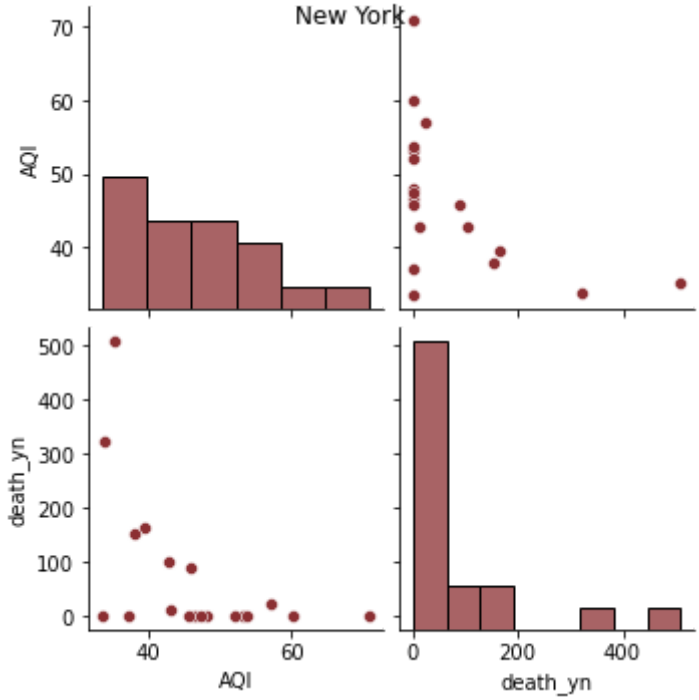

```
<Figure size 1080x504 with 0 Axes>
```



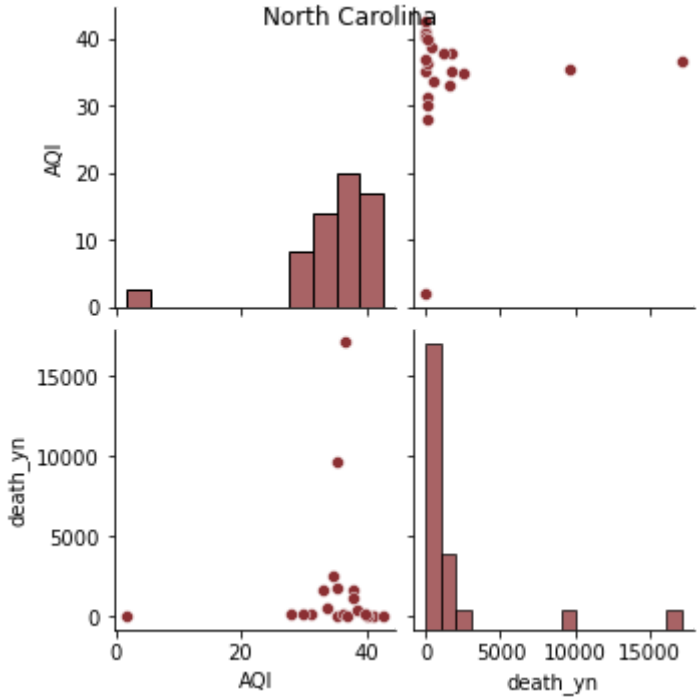

```
<Figure size 1080x504 with 0 Axes>
```



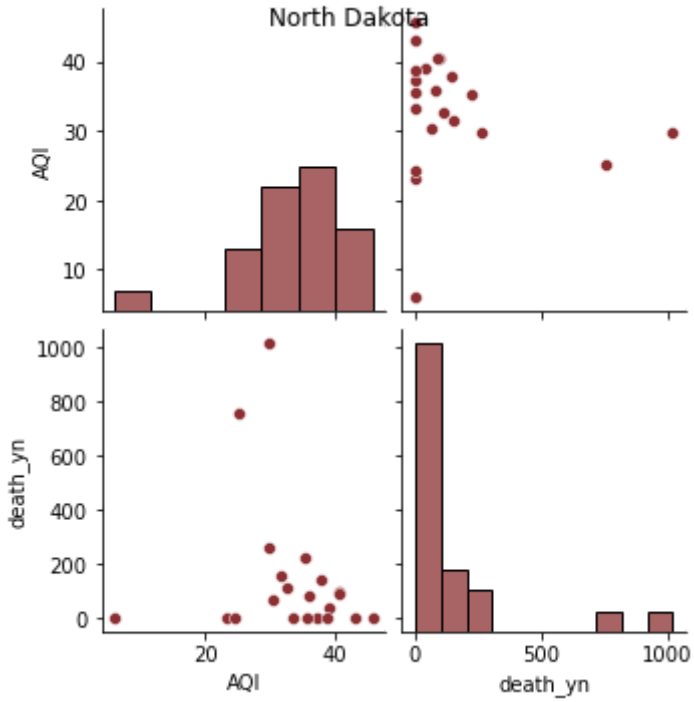

```
<Figure size 1080x504 with 0 Axes>
```



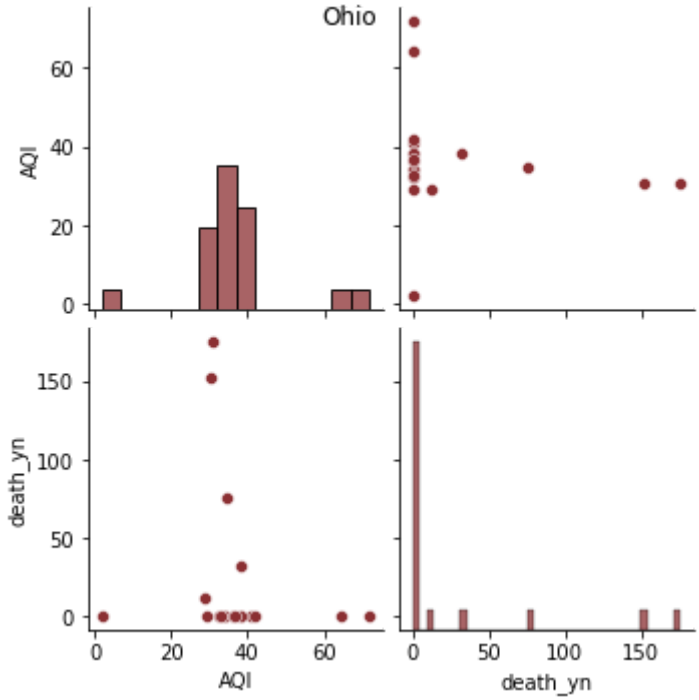

```
<Figure size 1080x504 with 0 Axes>
```



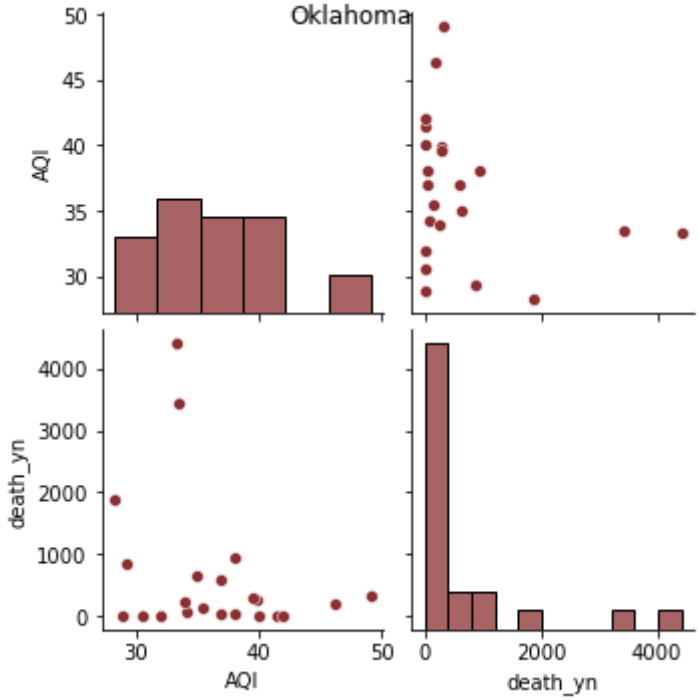

```
<Figure size 1080x504 with 0 Axes>
```



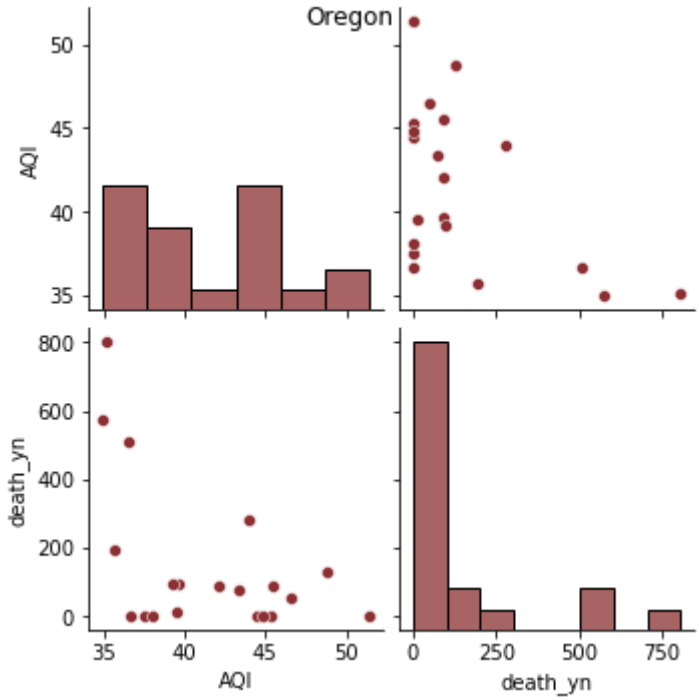

```
<Figure size 1080x504 with 0 Axes>
```



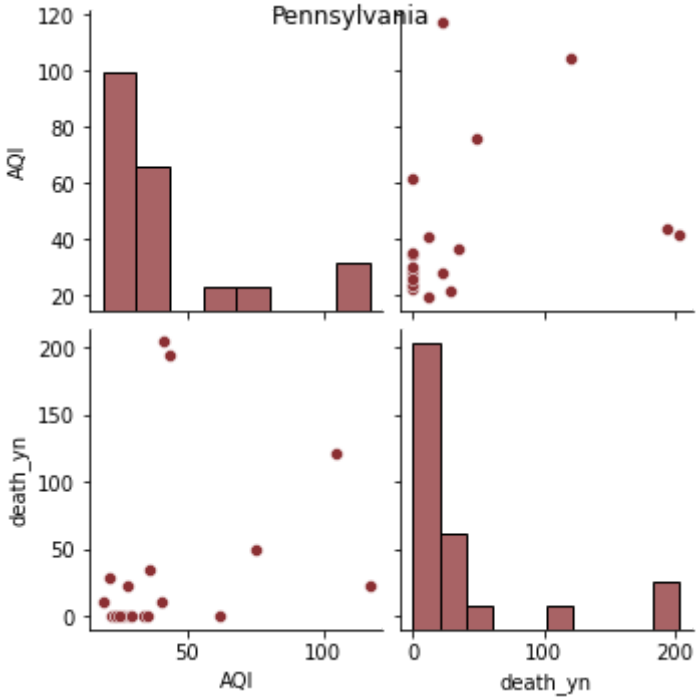

```
<Figure size 1080x504 with 0 Axes>
```



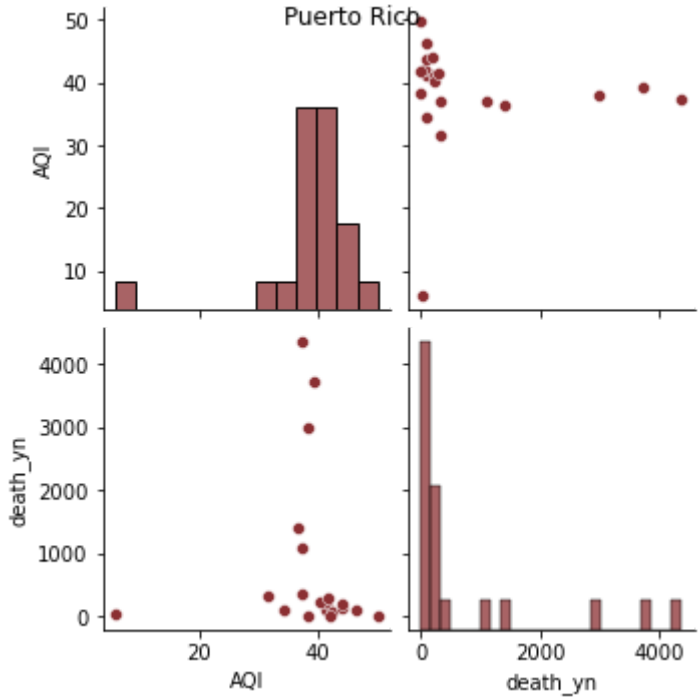

```
<Figure size 1080x504 with 0 Axes>
```



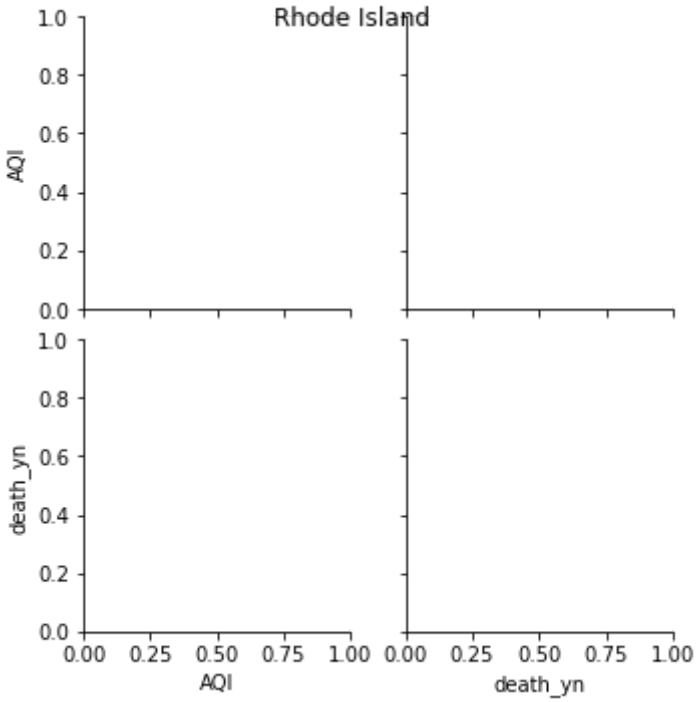

```
<Figure size 1080x504 with 0 Axes>
```



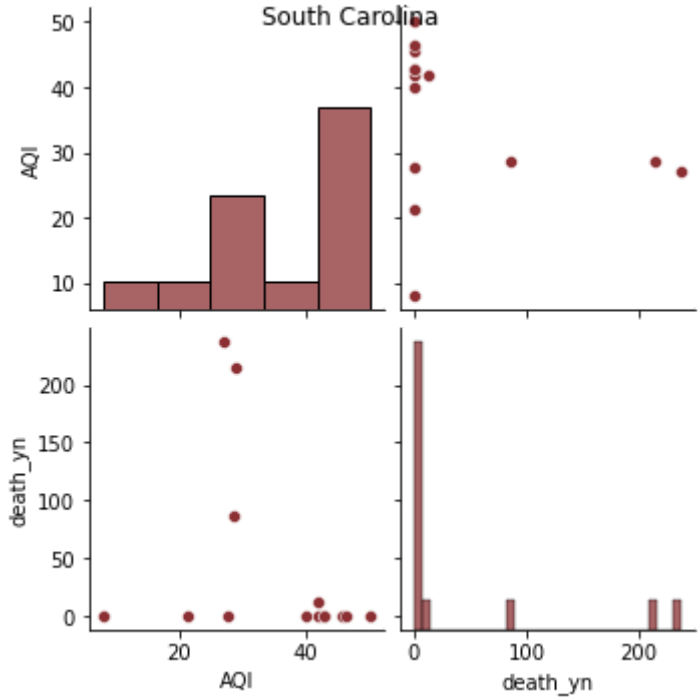

```
<Figure size 1080x504 with 0 Axes>
```



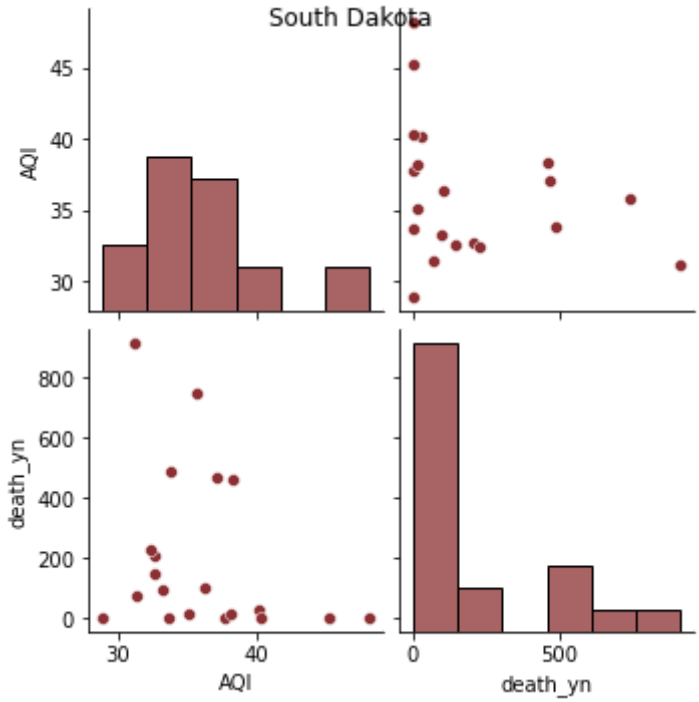

```
<Figure size 1080x504 with 0 Axes>
```



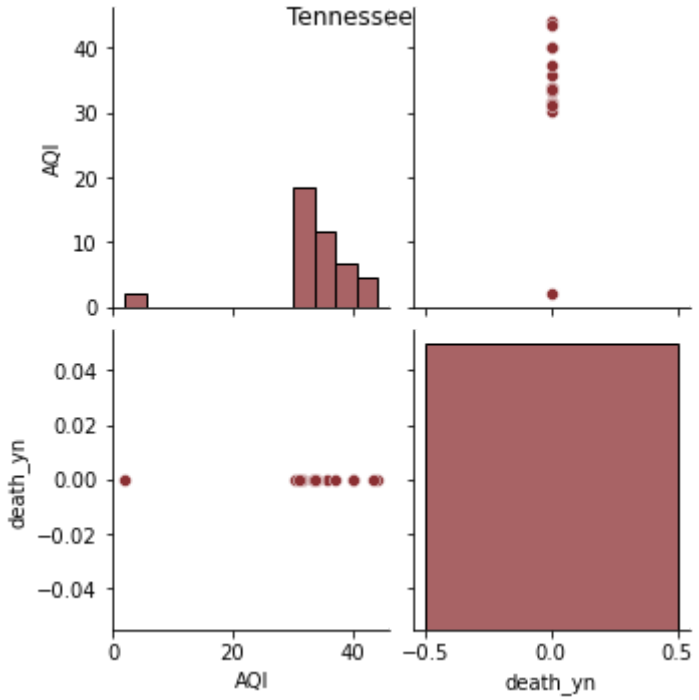

```
<Figure size 1080x504 with 0 Axes>
```



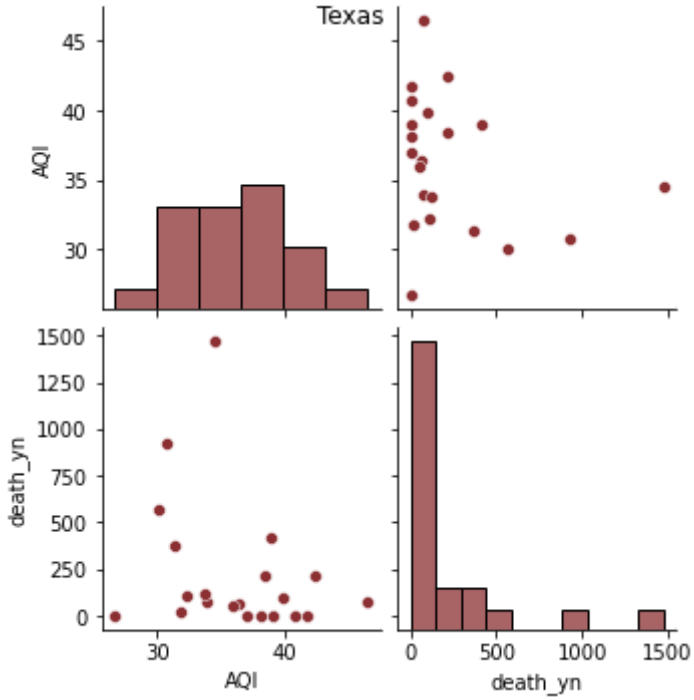

```
<Figure size 1080x504 with 0 Axes>
```



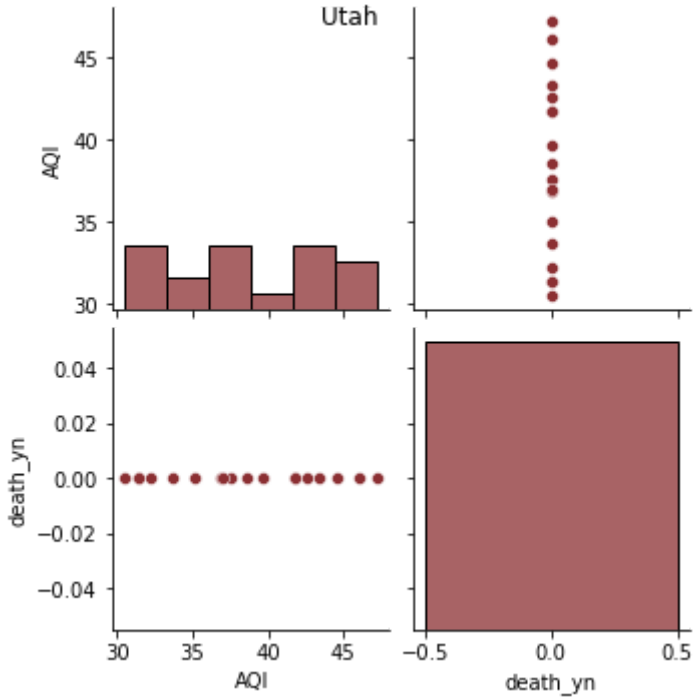

```
<Figure size 1080x504 with 0 Axes>
```



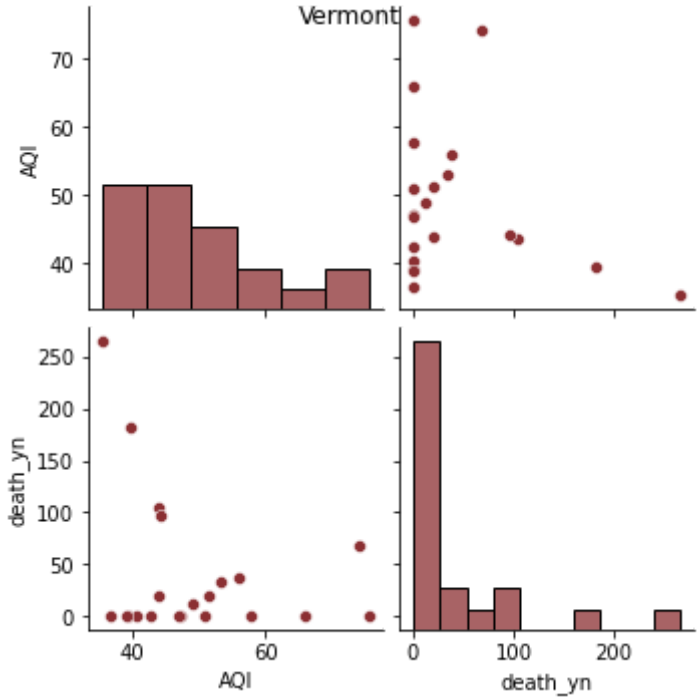

```
<Figure size 1080x504 with 0 Axes>
```



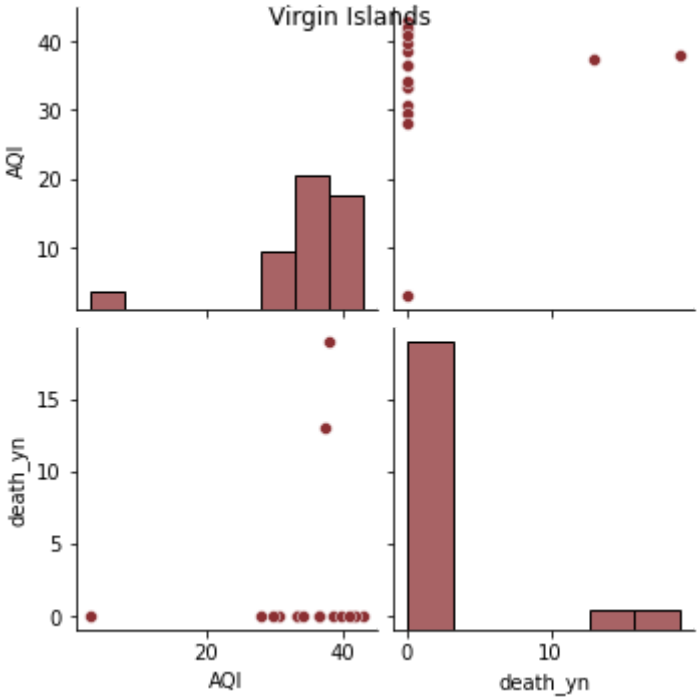
<Figure size 1080x504 with 0 Axes>



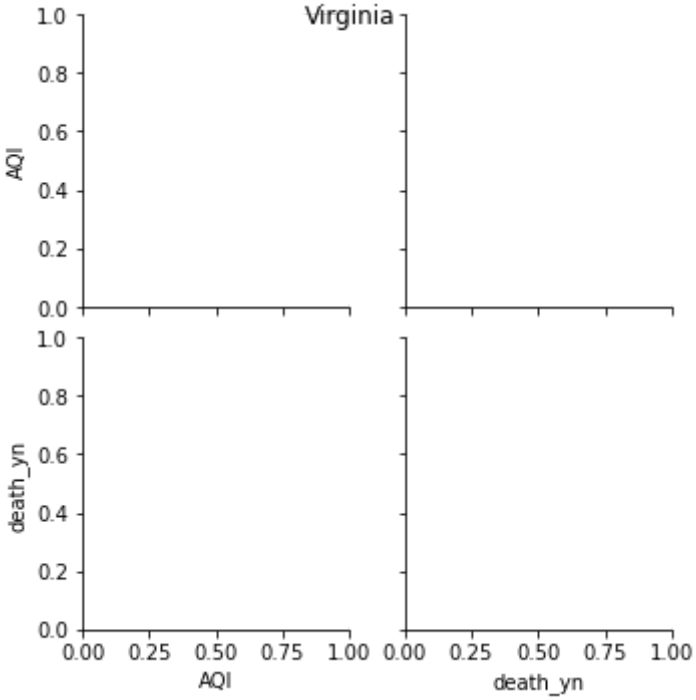
<Figure size 1080x504 with 0 Axes>



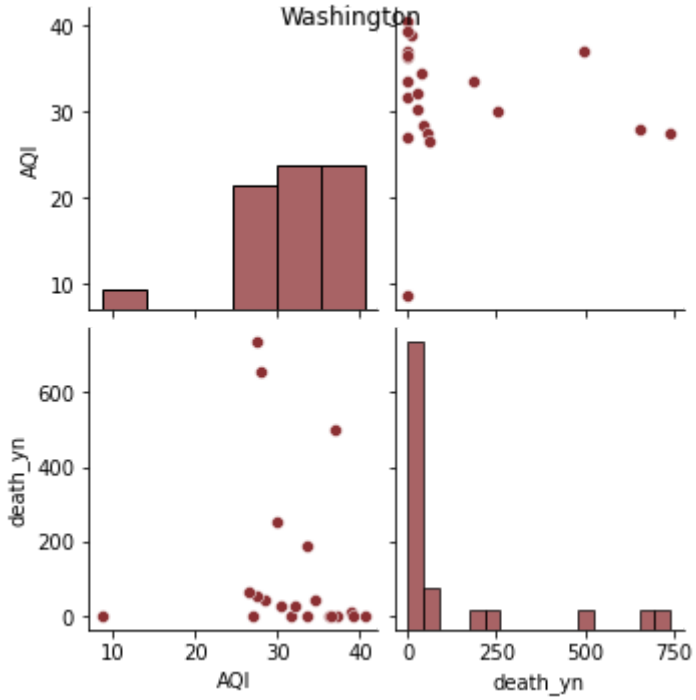

```
<Figure size 1080x504 with 0 Axes>
```



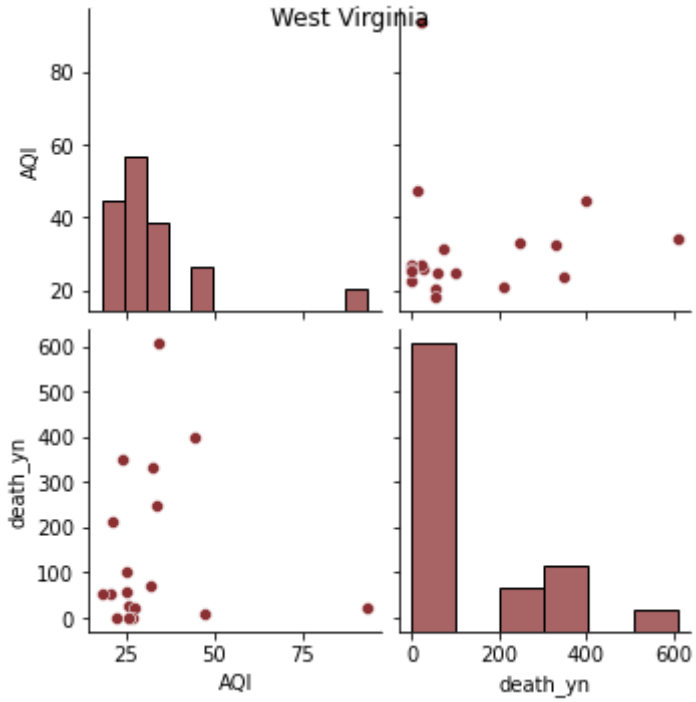

```
<Figure size 1080x504 with 0 Axes>
```



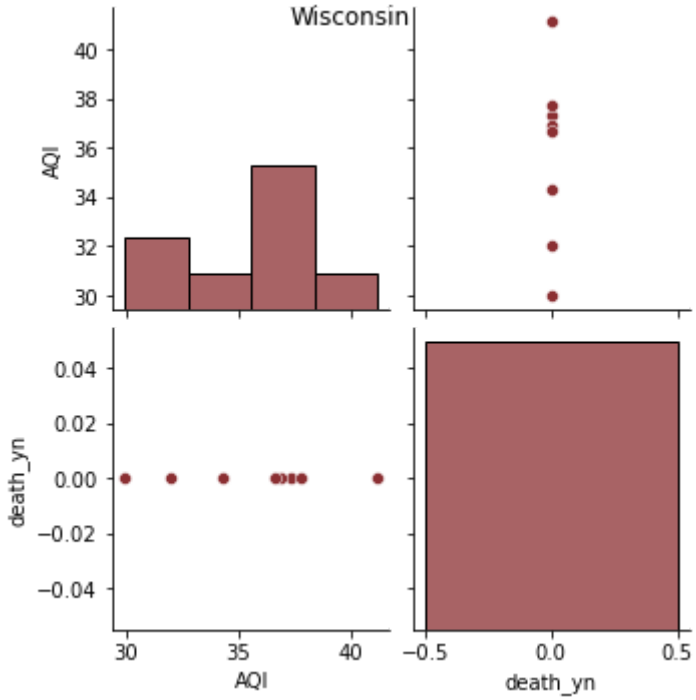

```
<Figure size 1080x504 with 0 Axes>
```



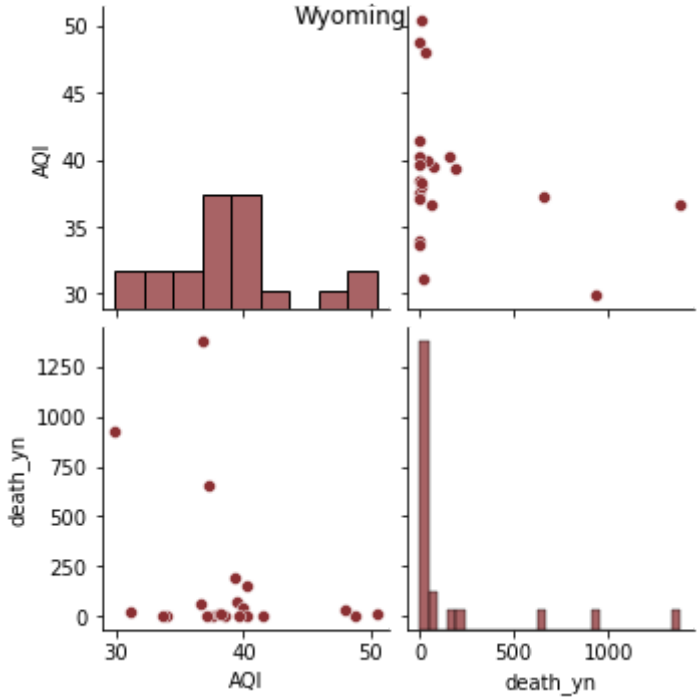

```
<Figure size 1080x504 with 0 Axes>
```



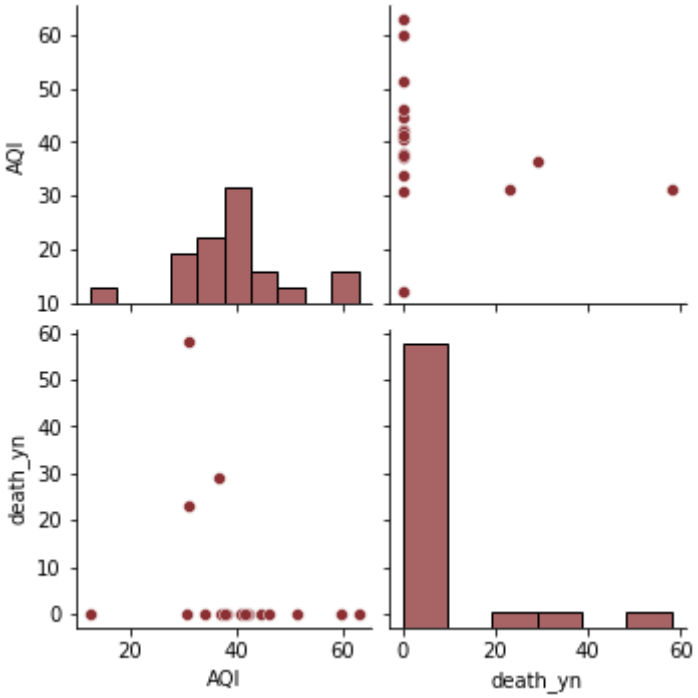

```
<Figure size 1080x504 with 0 Axes>
```




```
<Figure size 1080x504 with 0 Axes>
```

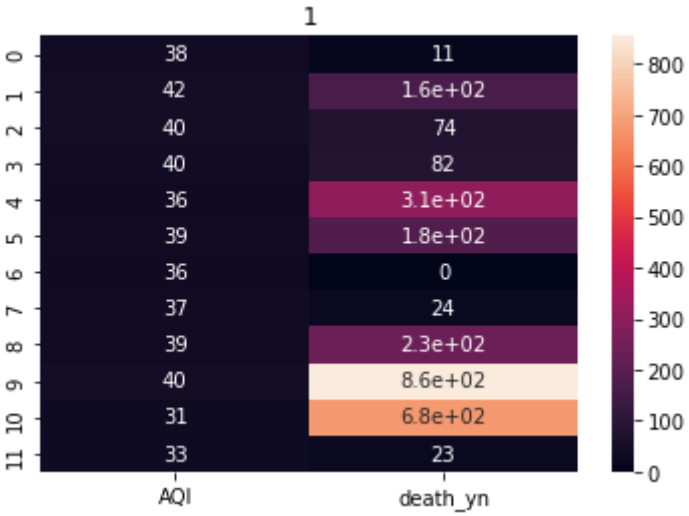



```
<Figure size 1080x504 with 0 Axes>
```

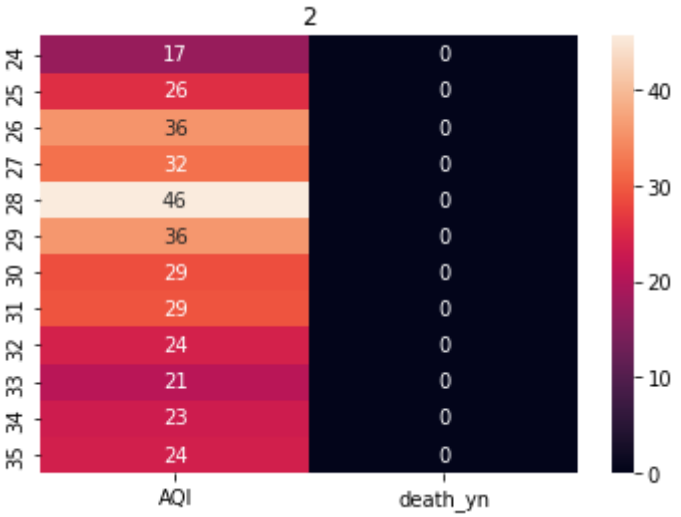


In [70]:
```
##Generating Correlation Heat Map AQI and Death_yn

i=0
j=24
for code in sc:

  df_plotter = pd.DataFrame()
  df_plotter=merged_df2.iloc[i:j,:]
  df_plotter = df_plotter.drop(['State_Name','year_month','State Code_x','ID','
  df_plotter_2020=pd.DataFrame()
  df_plotter_2019=pd.DataFrame()
  df_plotter_2019=df_plotter.iloc[:12,]
  df_plotter_2020=df_plotter.iloc[12:,]
```

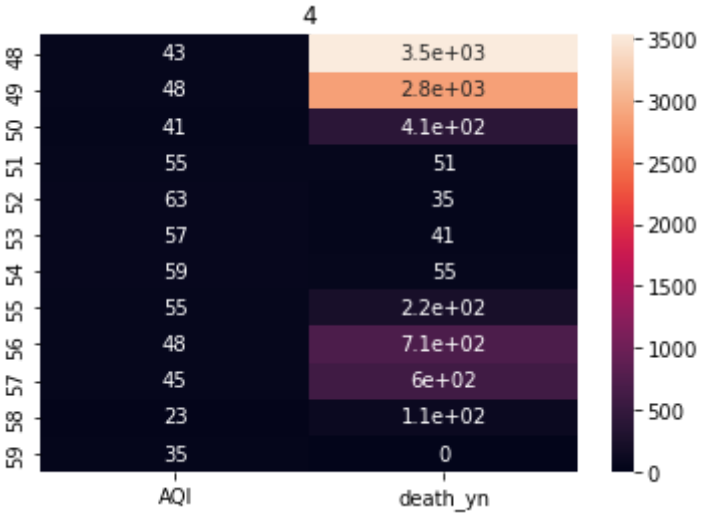
```
if not df_plotter.empty:
    try:
        ax = plt.axes()
        sns.heatmap(df_plotter_2019, annot=True, ax=ax)
        ax.set_title(code)
        plt.show()
        print('----------------------')
        p2 = sns.heatmap(df_plotter_2020, annot=True, ax=ax)
        ax.set_title(code)
        plt.show()
        print('----------------------')
    except ValueError:
        print("Warning: zero-size array")


i+=24
j+=24
```

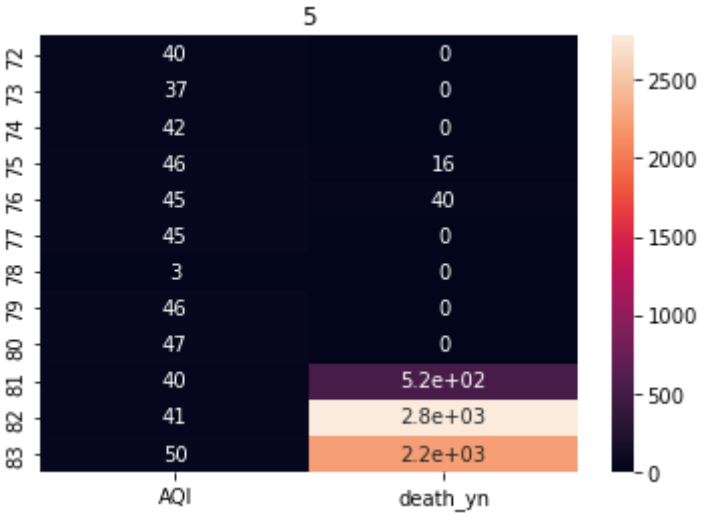


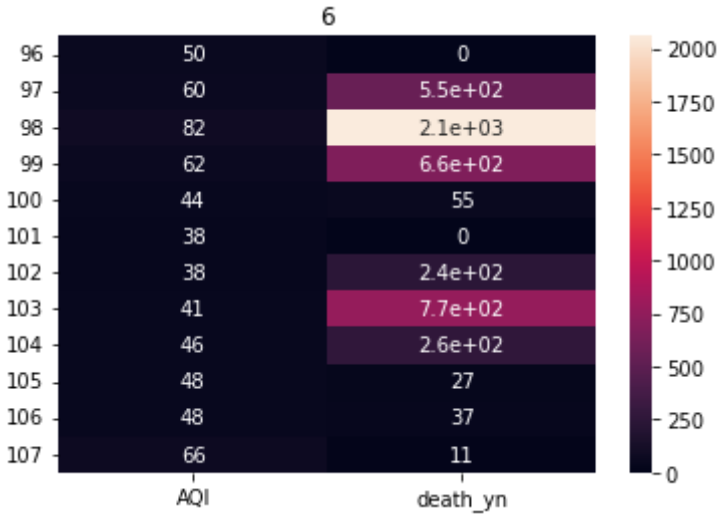

```
----------------------
----------------------
```

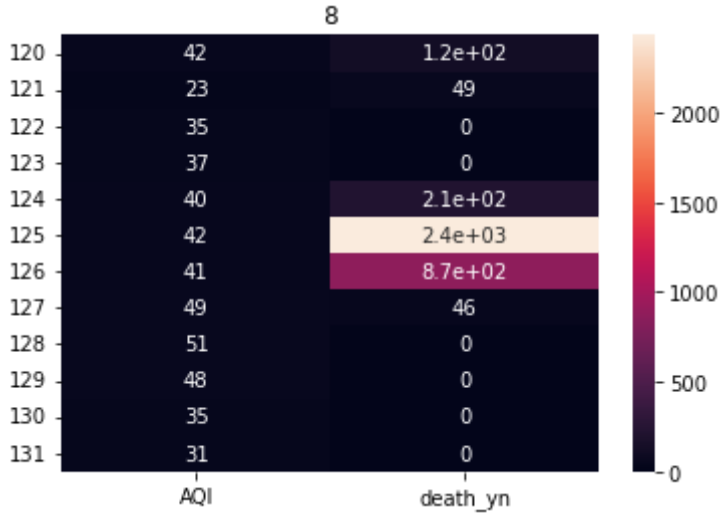


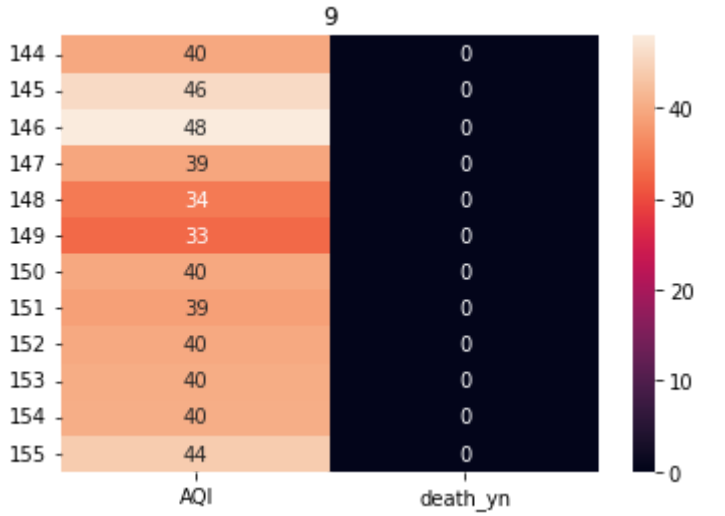

```
----------------------
----------------------
```

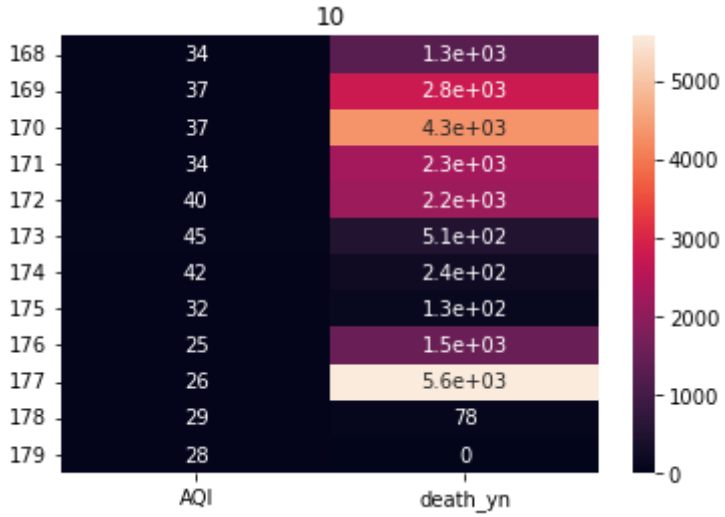

4
48 43 3.5e+03
49 48 2.8e+03
50 41 4.1e+02
51 55 51
52 63 35
53 57 41
54 59 55
55 55 2.2e+02
56 48 7.1e+02
57 45 6e+02
58 23 1.1e+02
59 35 0
AQI
death_yn
3500
3000
2500
2000
1500
1000
500
0

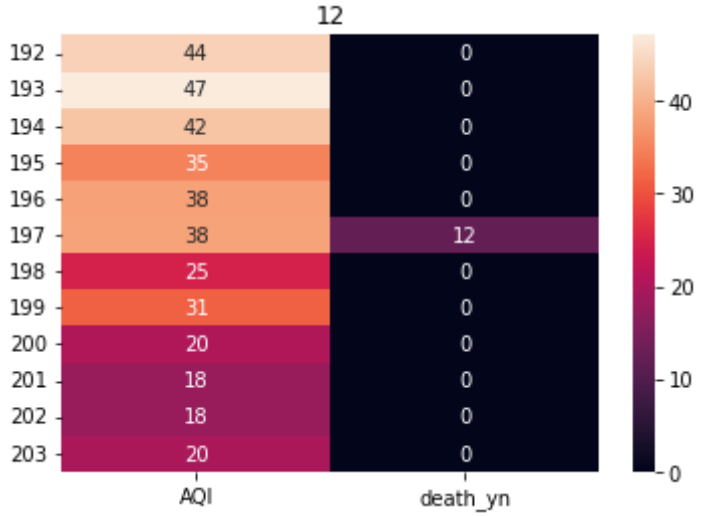

5
72 40 0
73 37 0
74 42 0
75 46 16
76 45 40
77 45 0
78 3 0
79 46 0
80 47 0
81 40 5.2e+02
82 41 2.8e+03
83 50 2.2e+03
AQI
death_yn
2500
2000
1500
1000
500
0

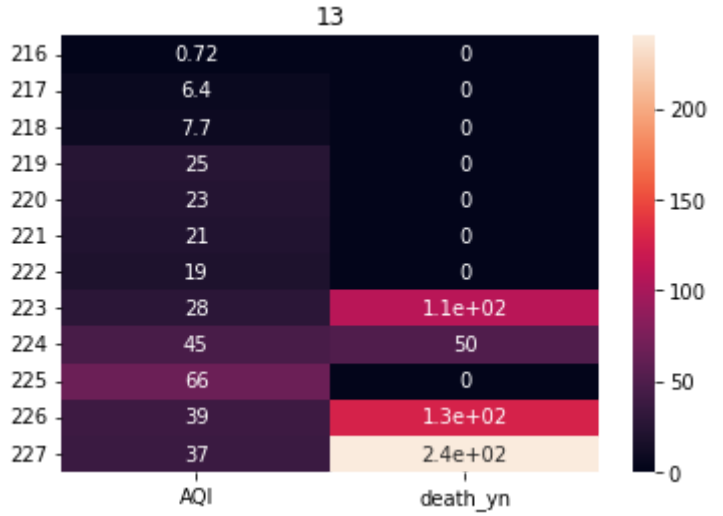

6
96 50 0
97 60 5.5e+02
98 82 2.1e+03
99 62 6.6e+02
100 44 55
101 38 0
102 38 2.4e+02
103 41 7.7e+02
104 46 2.6e+02
105 48 27
106 48 37
107 66 11
AQI
death_yn
2000
1750
1500
1250
1000
750
500
250
0

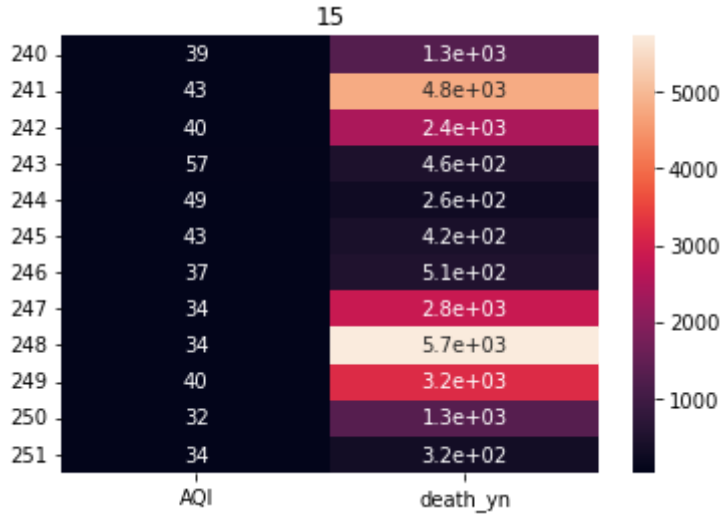

8
120 42 1.2e+02
121 23 49
122 35 0
123 37 0
124 40 2.1e+02
125 42 2.4e+03
126 41 8.7e+02
127 49 46
128 51 0
129 48 0
130 35 0
131 31 0
AQI
death_yn
2000
1500
1000
500
0

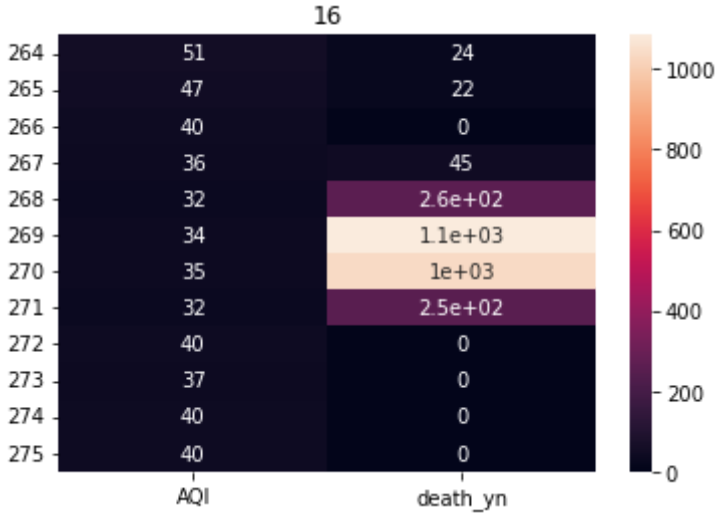

9
144 40 0
145 46 0
146 48 0
147 39 0
148 34 0
149 33 0
150 40 0
151 39 0
152 40 0
153 40 0
154 40 0
155 44 0
AQI
death_yn
40
30
20
10
0

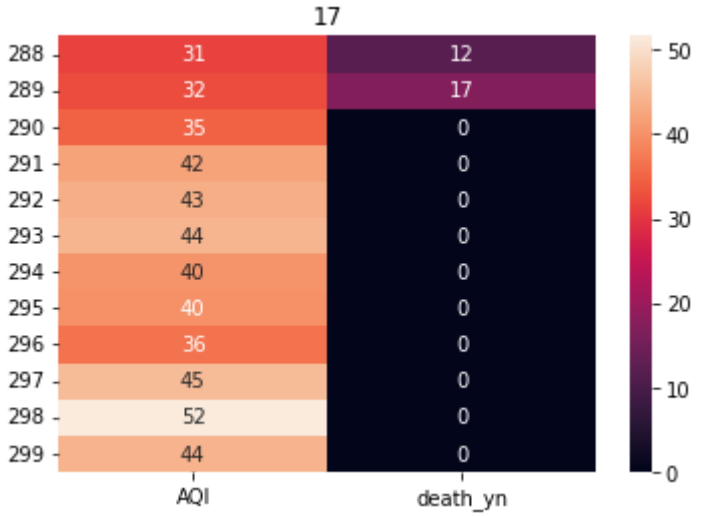

10
168 34 1.3e+03
169 37 2.8e+03
170 37 4.3e+03
171 34 2.3e+03
172 40 2.2e+03
173 45 5.1e+02
174 42 2.4e+02
175 32 1.3e+02
176 25 1.5e+03
177 26 5.6e+03
178 29 78
179 28 0
AQI
death_yn
5000
4000
3000
2000
1000
0


12
192 44 0
193 47 0
194 42 0
195 35 0
196 38 0
197 38 12
198 25 0
199 31 0
200 20 0
201 18 0
202 18 0
203 20 0
AQI
death_yn
40
30
20
10
0

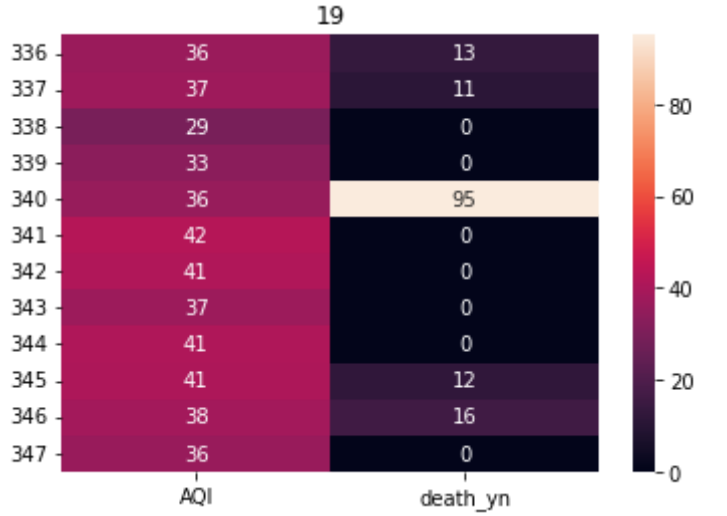

13
216 0.72 0
217 6.4 0
218 7.7 0
219 25 0
220 23 0
221 21 0
222 19 0
223 28 1.1e+02
224 45 50
225 66 0
226 39 1.3e+02
227 37 2.4e+02
AQI
death_yn
200
150
100
50
0

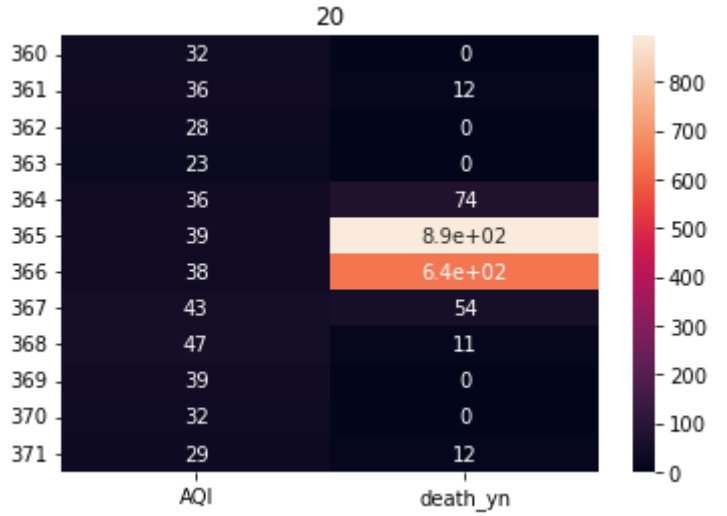

15
240 39 1.3e+03
241 43 4.8e+03
242 40 2.4e+03
243 57 4.6e+02
244 49 2.6e+02
245 43 4.2e+02
246 37 5.1e+02
247 34 2.8e+03
248 34 5.7e+03
249 40 3.2e+03
250 32 1.3e+03
251 34 3.2e+02
AQI
death_yn
5000
4000
3000
2000
1000

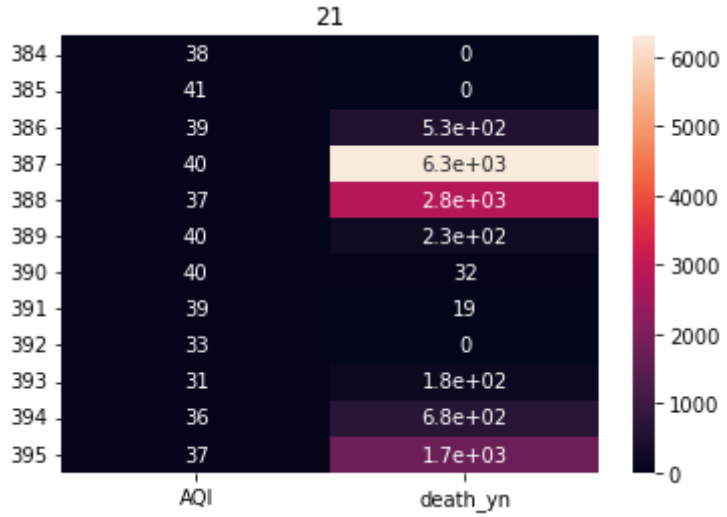

16
264 51 24
265 47 22
266 40 0
267 36 45
268 32 2.6e+02
269 34 1.1e+03
270 35 1e+03
271 32 2.5e+02
272 40 0
273 37 0
274 40 0
275 40 0
AQI
death_yn
1000
800
600
400
200
0

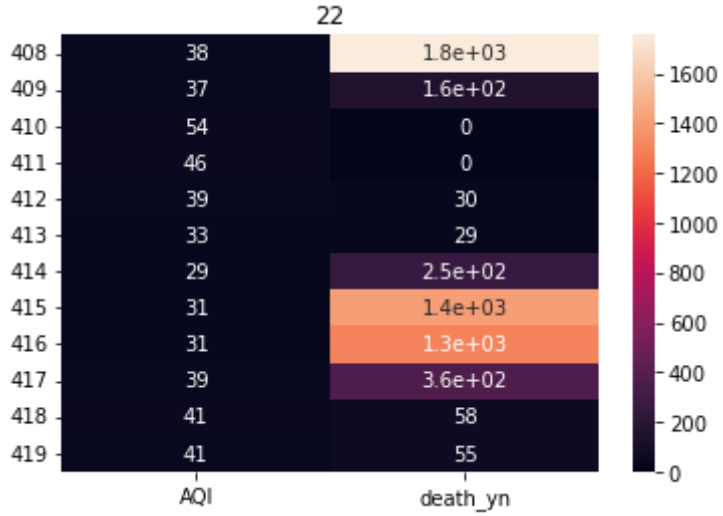

17
288 31 12
289 32 17
290 35 0
291 42 0
292 43 0
293 44 0
294 40 0
295 40 0
296 36 0
297 45 0
298 52 0
299 44 0
AQI
death_yn
50
40
30
20
10
0

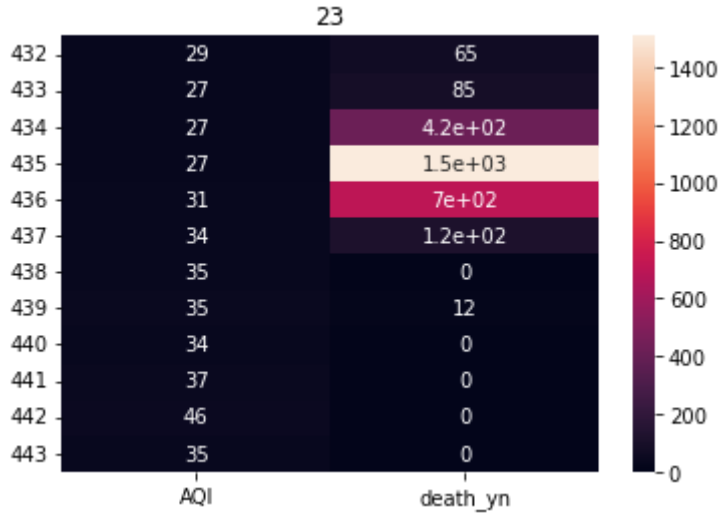

18
312 36 1.4e+02
313 34 0
314 37 0
315 41 0
316 32 0
317 45 0
318 1 12
319 36 18
320 41 81
321 37 25
322 46 12
323 40 13
AQI
death_yn
140
120
100
80
60
40
20
0

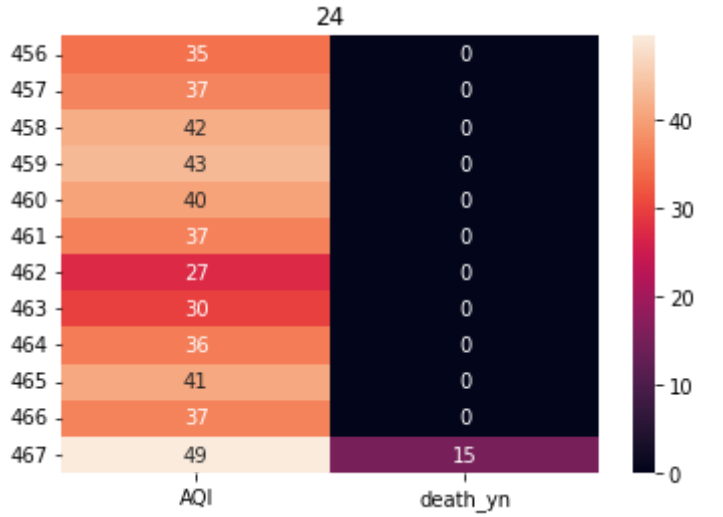

19
336 36 13
337 37 11
338 29 0
339 33 0
340 36 95
341 42 0
342 41 0
343 37 0
344 41 0
345 41 12
346 38 16
347 36 0
AQI
death_yn
80
60
40
20
0

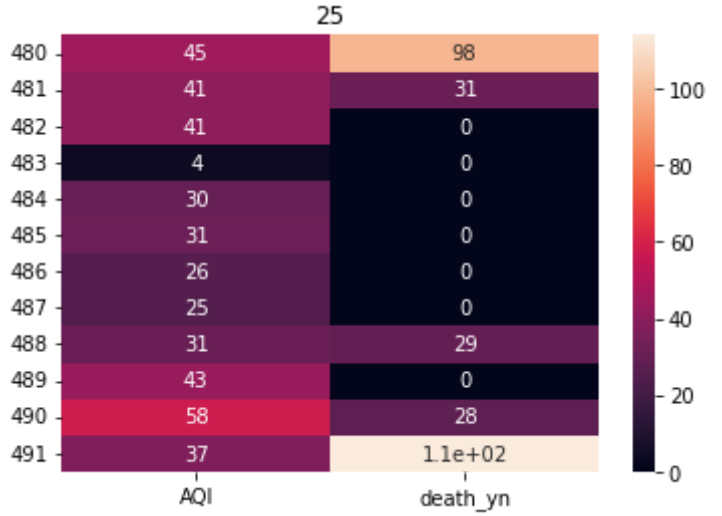

20
360 32 0
361 36 12
362 28 0
363 23 0
364 36 74
365 39 8.9e+02
366 38 6.4e+02
367 43 54
368 47 11
369 39 0
370 32 0
371 29 12
AQI
death_yn
800
700
600
500
400
300
200
100
0

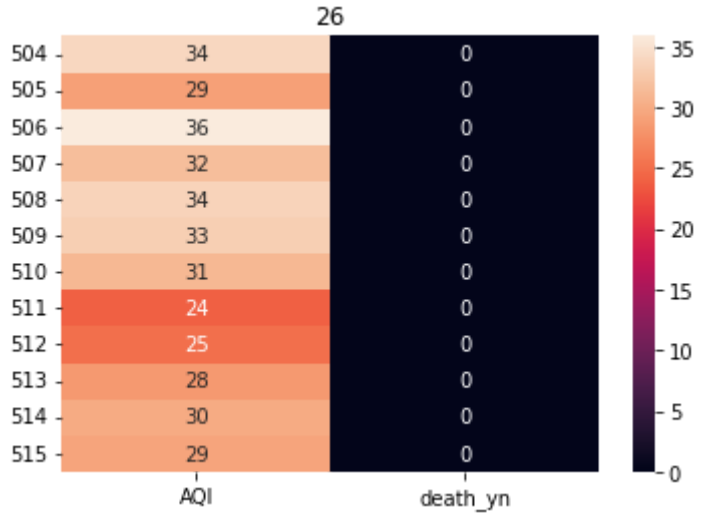

21
384 38 0
385 41 0
386 39 5.3e+02
387 40 6.3e+03
388 37 2.8e+03
389 40 2.3e+02
390 40 32
391 39 19
392 33 0
393 31 1.8e+02
394 36 6.8e+02
395 37 1.7e+03
AQI
death_yn
6000
5000
4000
3000
2000
1000
0

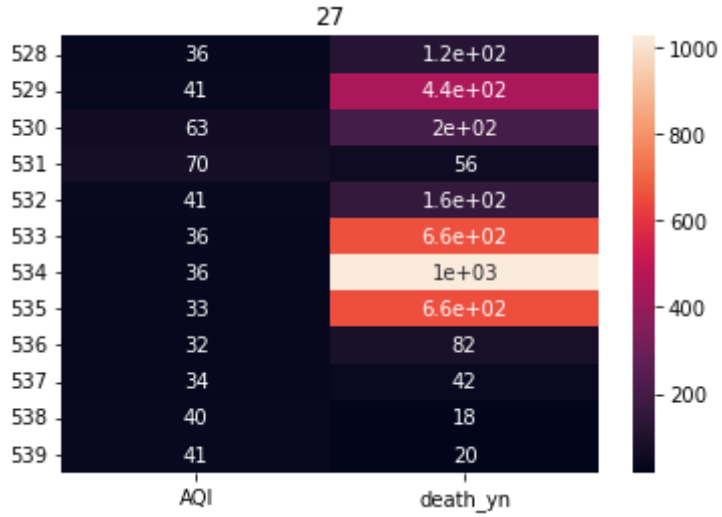

22
408 38 1.8e+03
409 37 1.6e+02
410 54 0
411 46 0
412 39 30
413 33 29
414 29 2.5e+02
415 31 1.4e+03
416 31 1.3e+03
417 39 3.6e+02
418 41 58
419 41 55
AQI
death_yn
1600
1400
1200
1000
800
600
400
200
0

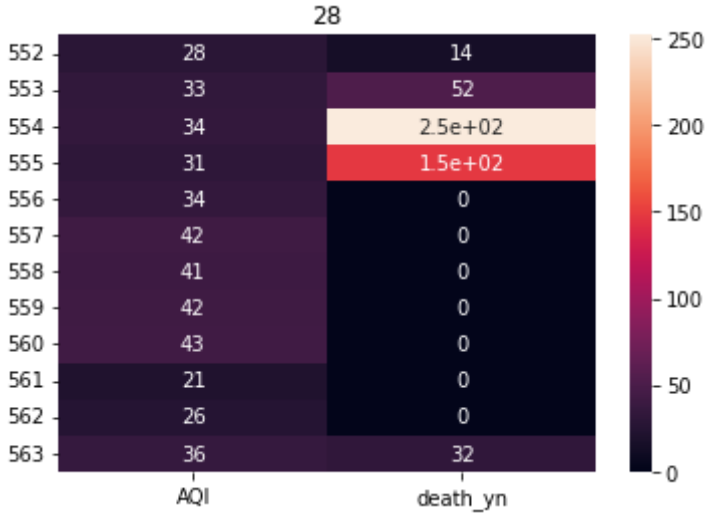

23
432 29 65
433 27 85
434 27 4.2e+02
435 27 1.5e+03
436 31 7e+02
437 34 1.2e+02
438 35 0
439 35 12
440 34 0
441 37 0
442 46 0
443 35 0
AQI
death_yn
1400
1200
1000
800
600
400
200
0

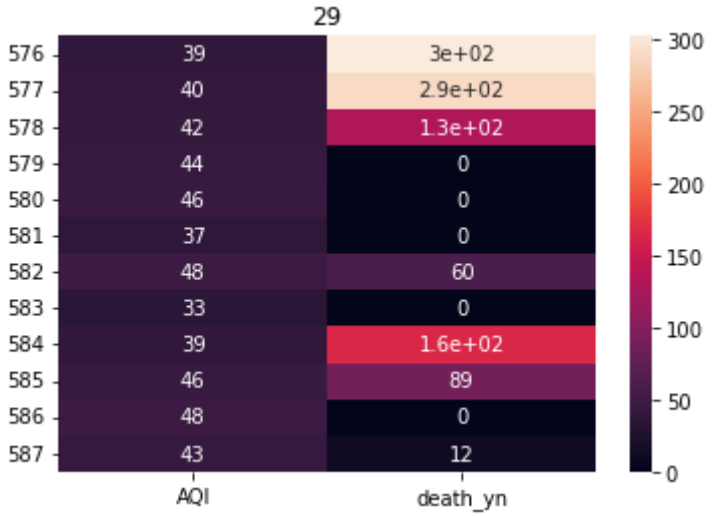

24
456 35 0
457 37 0
458 42 0
459 43 0
460 40 0
461 37 0
462 27 0
463 30 0
464 36 0
465 41 0
466 37 0
467 49 15
AQI
death_yn
40
30
20
10
0

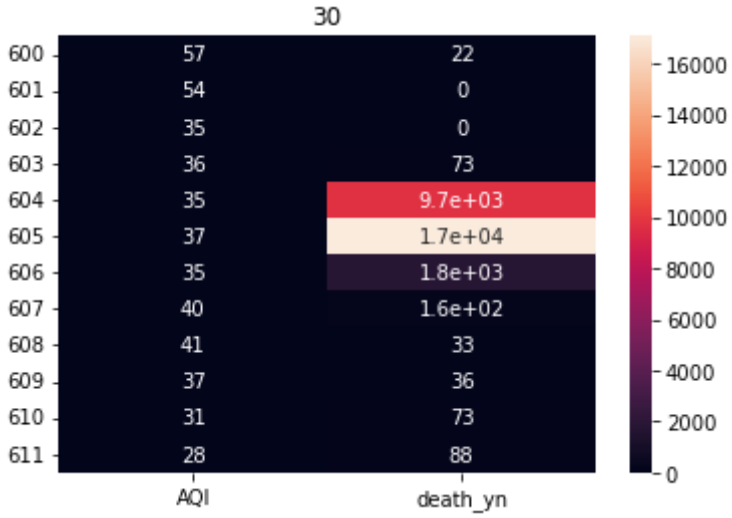

25
480 45 98
481 41 31
482 41 0
483 4 0
484 30 0
485 31 0
486 26 0
487 25 0
488 31 29
489 43 0
490 58 28
491 37 1.1e+02
AQI
death_yn
100
80
60
40
20
0

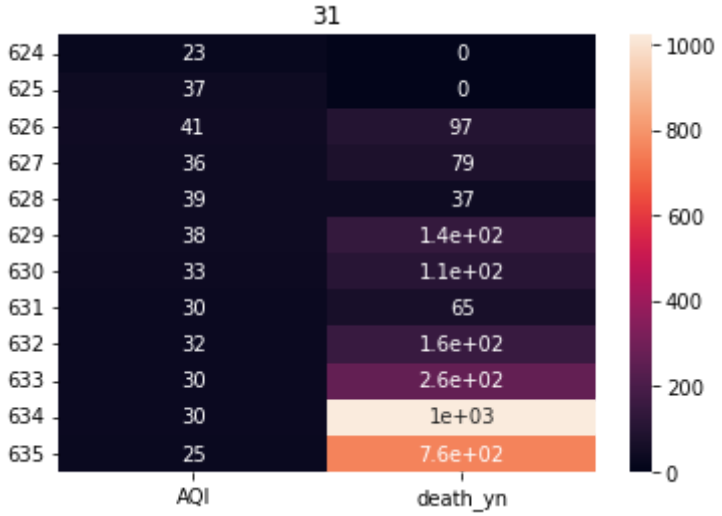

26
504 34 0
505 29 0
506 36 0
507 32 0
508 34 0
509 33 0
510 31 0
511 24 0
512 25 0
513 28 0
514 30 0
515 29 0
AQI
death_yn
35
30
25
20
15
10
5
0

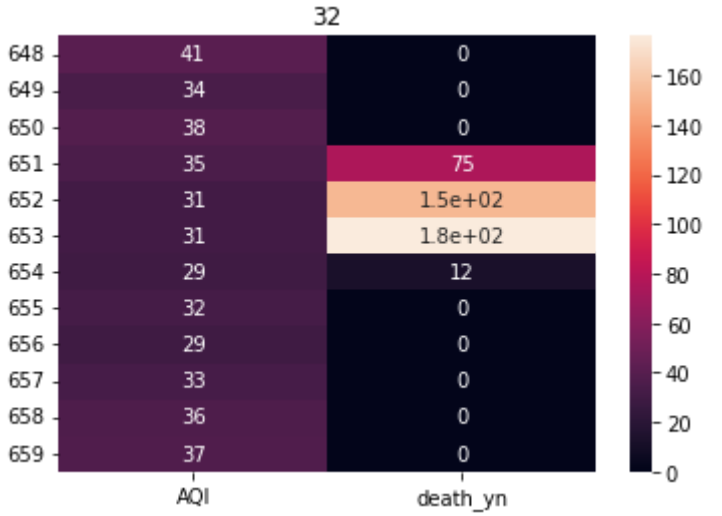

27
528 36 1.2e+02
529 41 4.4e+02
530 63 2e+02
531 70 56
532 41 1.6e+02
533 36 6.6e+02
534 36 1e+03
535 33 6.6e+02
536 32 82
537 34 42
538 40 18
539 41 20
AQI
death_yn
1000
800
600
400
200

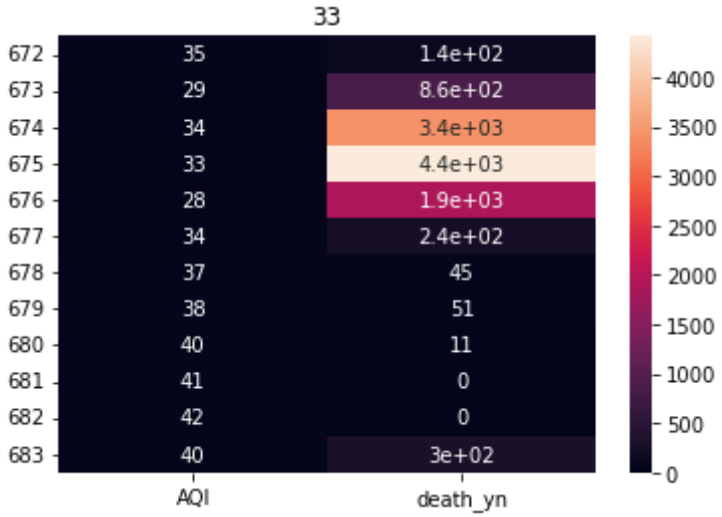

28
552 28 14
553 33 52
554 34 2.5e+02
555 31 1.5e+02
556 34 0
557 42 0
558 41 0
559 42 0
560 43 0
561 21 0
562 26 0
563 36 32
AQI
death_yn
250
200
150
100
50
0

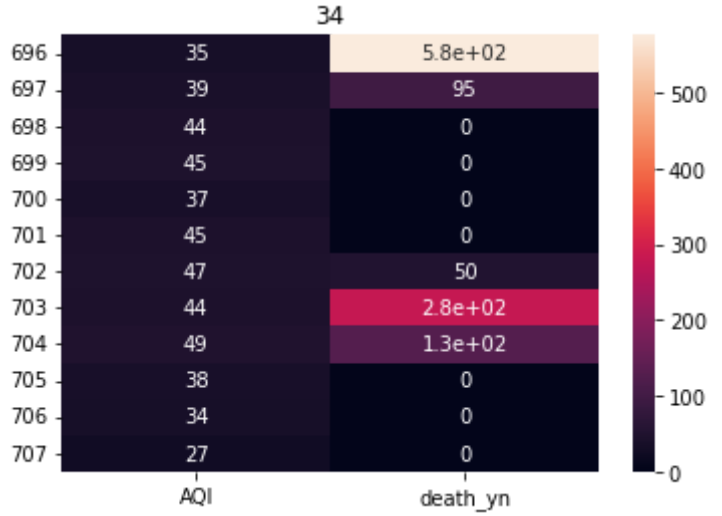

29
576 39 3e+02
577 40 2.9e+02
578 42 1.3e+02
579 44 0
580 46 0
581 37 0
582 48 60
583 33 0
584 39 1.6e+02
585 46 89
586 48 0
587 43 12
AQI
death_yn
300
250
200
150
100
50
0

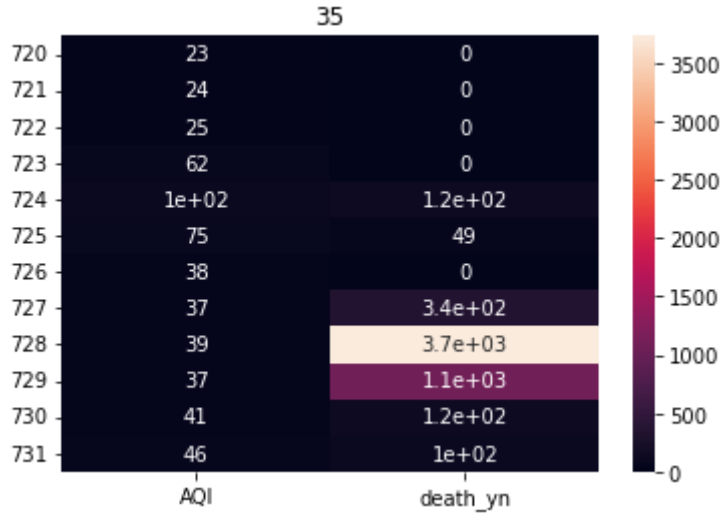

30
600 57 22
601 54 0
602 35 0
603 36 73
604 35 9.7e+03
605 37 1.7e+04
606 35 1.8e+03
607 40 1.6e+02
608 41 33
609 37 36
610 31 73
611 28 88
AQI
death_yn
16000
14000
12000
10000
8000
6000
4000
2000
0

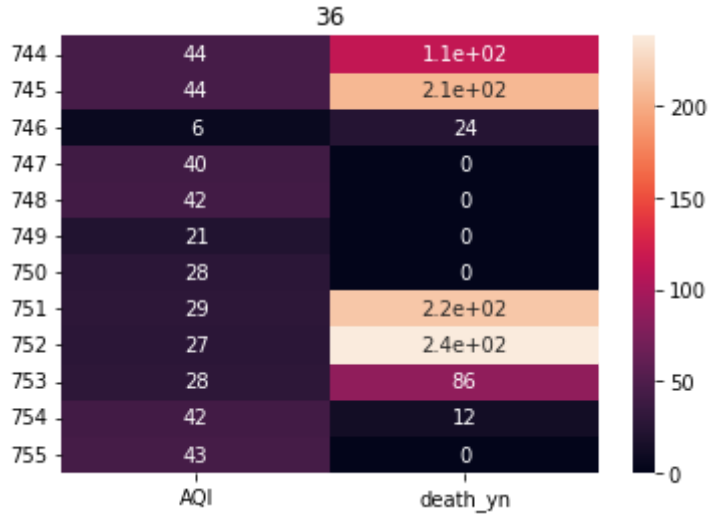

31
624 23 0
625 37 0
626 41 97
627 36 79
628 39 37
629 38 1.4e+02
630 33 1.1e+02
631 30 65
632 32 1.6e+02
633 30 2.6e+02
634 30 1e+03
635 25 7.6e+02
AQI
death_yn
1000
800
600
400
200
0

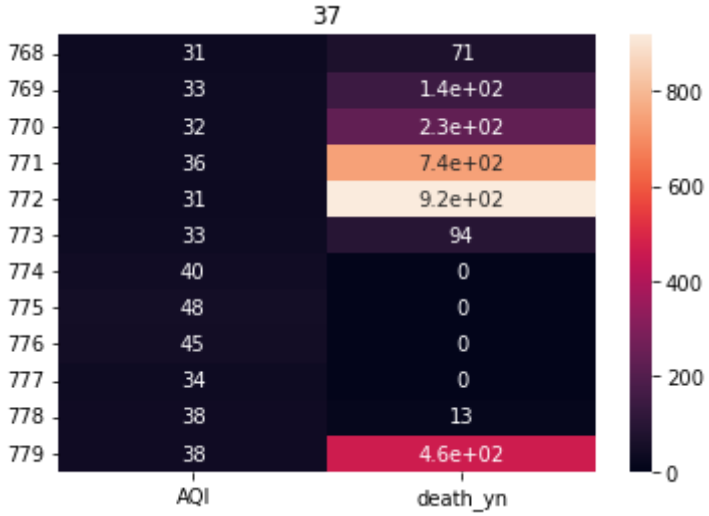

32
648 41 0
649 34 0
650 38 0
651 35 75
652 31 1.5e+02
653 31 1.8e+02
654 29 12
655 32 0
656 29 0
657 33 0
658 36 0
659 37 0
AQI
death_yn
160
140
120
100
80
60
40
20
0

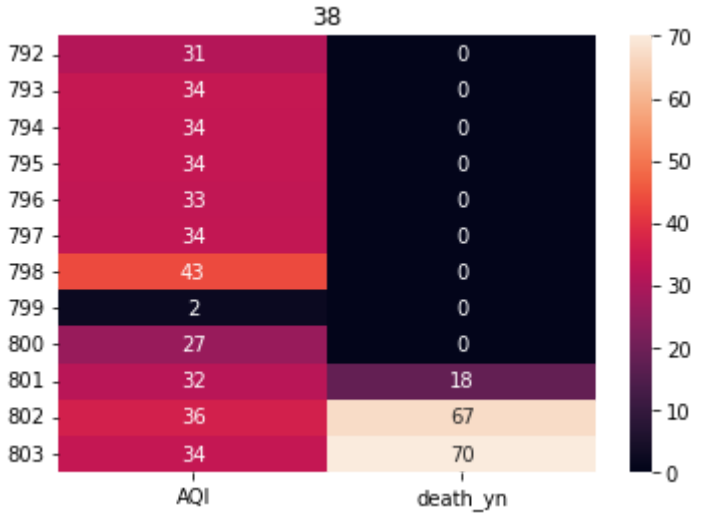

33
672 35 1.4e+02
673 29 8.6e+02
674 34 3.4e+03
675 33 4.4e+03
676 28 1.9e+03
677 34 2.4e+02
678 37 45
679 38 51
680 40 11
681 41 0
682 42 0
683 40 3e+02
AQI
death_yn
4000
3500
3000
2500
2000
1500
1000
500
0

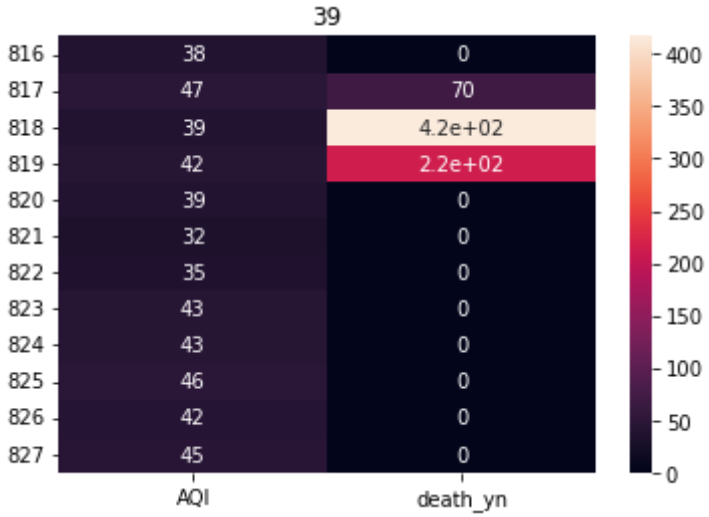

34
696 35 5.8e+02
697 39 95
698 44 0
699 45 0
700 37 0
701 45 0
702 47 50
703 44 2.8e+02
704 49 1.3e+02
705 38 0
706 34 0
707 27 0
AQI
death_yn
500
400
300
200
100
0

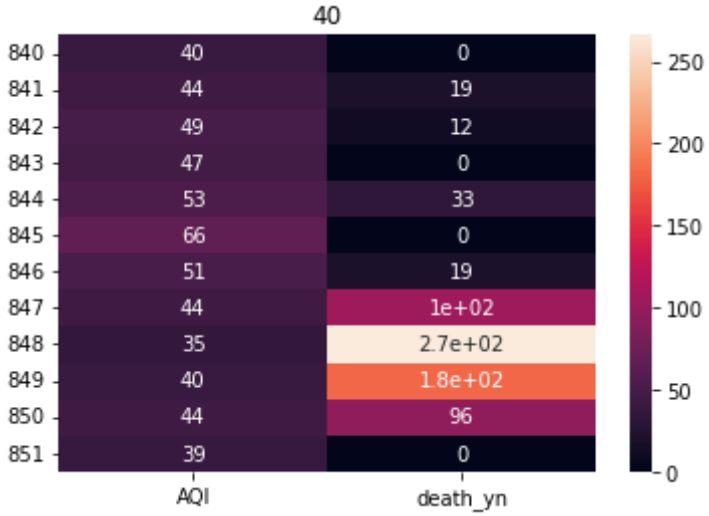

35
720 23 0
721 24 0
722 25 0
723 62 0
724 1e+02 1.2e+02
725 75 49
726 38 0
727 37 3.4e+02
728 39 3.7e+03
729 37 1.1e+03
730 41 1.2e+02
731 46 1e+02
AQI
death_yn
3500
3000
2500
2000
1500
1000
500
0

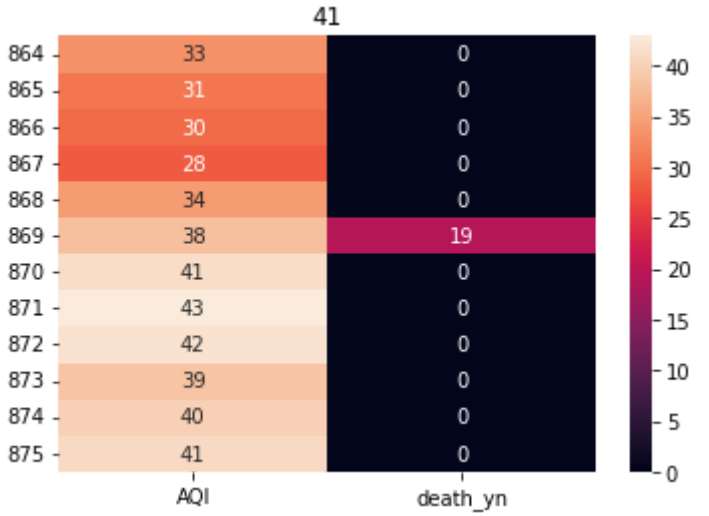

36
744 44 1.1e+02
745 44 2.1e+02
746 6 24
747 40 0
748 42 0
749 21 0
750 28 0
751 29 2.2e+02
752 27 2.4e+02
753 28 86
754 42 12
755 43 0
AQI
death_yn
200
150
100
50
0

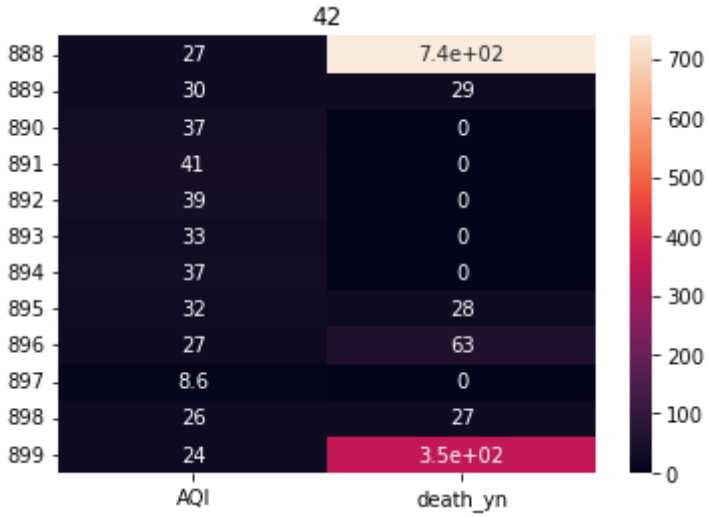

37
768 31 71
769 33 1.4e+02
770 32 2.3e+02
771 36 7.4e+02
772 31 9.2e+02
773 33 94
774 40 0
775 48 0
776 45 0
777 34 0
778 38 13
779 38 4.6e+02
AQI
death_yn
800
600
400
200
0

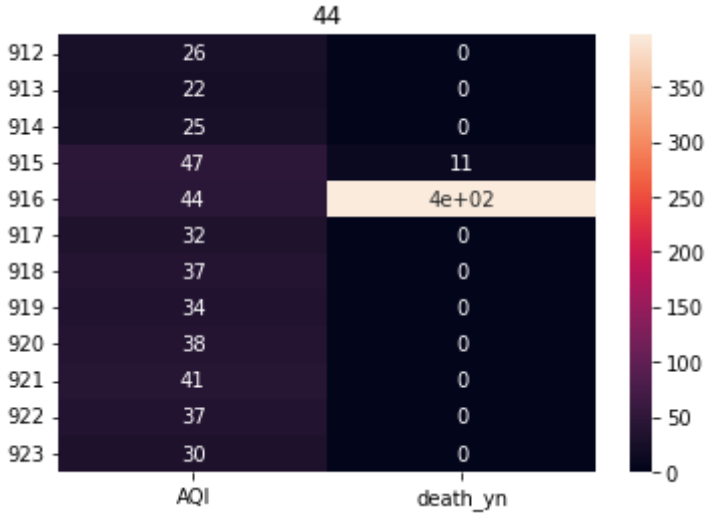

38
792 31 0
793 34 0
794 34 0
795 34 0
796 33 0
797 34 0
798 43 0
799 2 0
800 27 0
801 32 18
802 36 67
803 34 70
AQI
death_yn
70
60
50
40
30
20
10
0

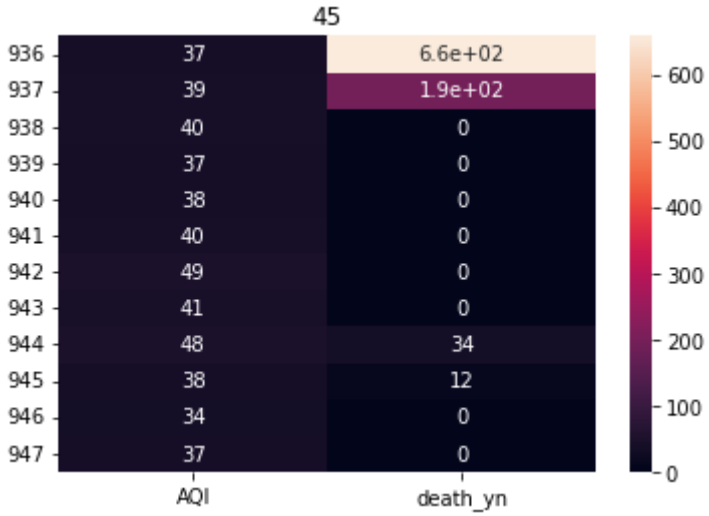

39
816 38 0
817 47 70
818 39 4.2e+02
819 42 2.2e+02
820 39 0
821 32 0
822 35 0
823 43 0
824 43 0
825 46 0
826 42 0
827 45 0
AQI
death_yn
400
350
300
250
200
150
100
50
0

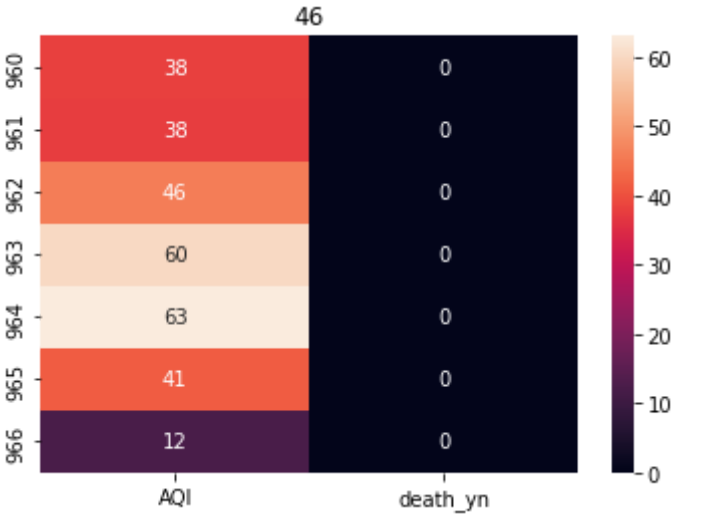

40
840 40 0
841 44 19
842 49 12
843 47 0
844 53 33
845 66 0
846 51 19
847 44 1e+02
848 35 2.7e+02
849 40 1.8e+02
850 44 96
851 39 0
AQI
death_yn
250
200
150
100
50
0


41
864 33 0
865 31 0
866 30 0
867 28 0
868 34 0
869 38 19
870 41 0
871 43 0
872 42 0
873 39 0
874 40 0
875 41 0
AQI
death_yn
40
35
30
25
20
15
10
5
0


42
888 27 7.4e+02
889 30 29
890 37 0
891 41 0
892 39 0
893 33 0
894 37 0
895 32 28
896 27 63
897 8.6 0
898 26 27
899 24 3.5e+02
AQI
death_yn
700
600
500
400
300
200
100
0







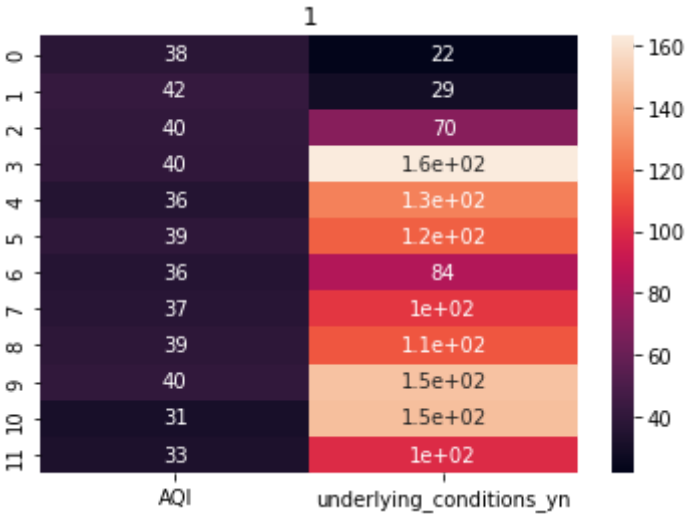
```
Warning: zero-size array
```

After Analysing the death conditions of the Dataframe: death_yn states that the the patient die as a result of this illness, which is

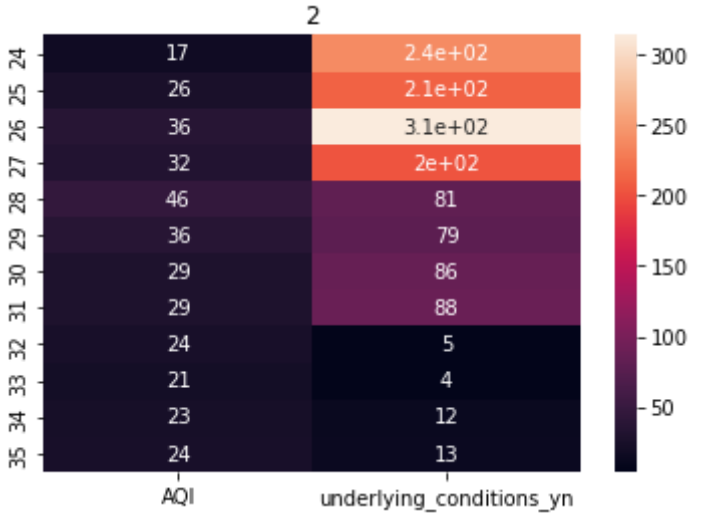
specified in terms of [Yes; No; Unknown; Missing] .

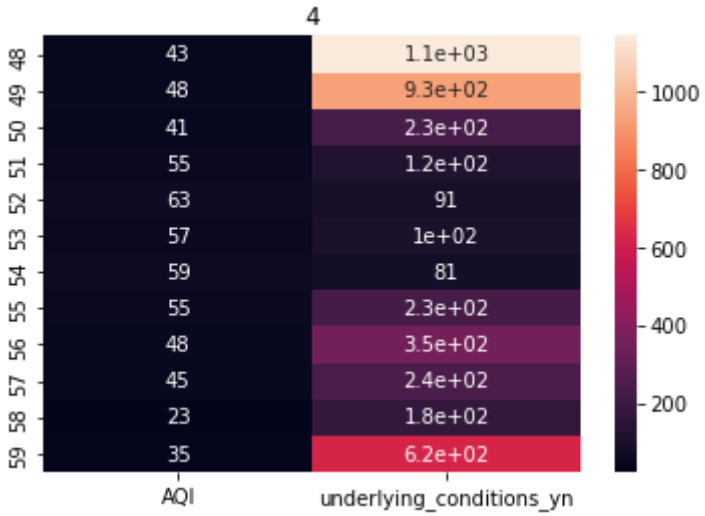
Based on the Above heat maps, the AQI has no correlation with the death_yn Heatmaps as many States of them has 0 death rates between the AQI .

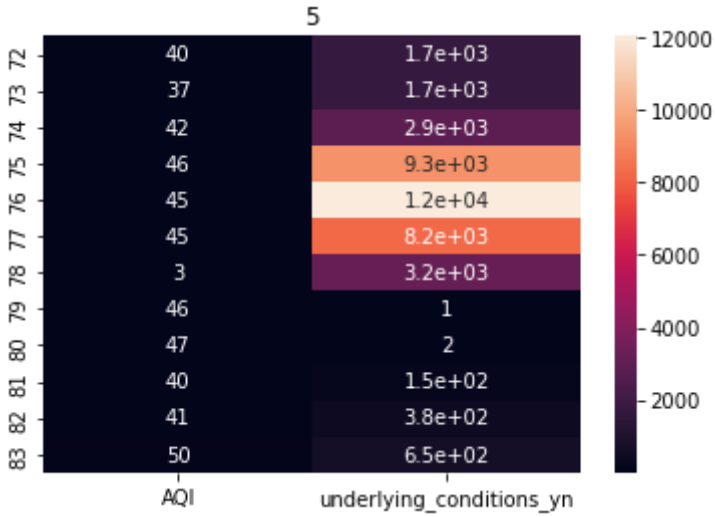
In [71]:
```
##Generating Correlation Heat Map of AQI and Underlying_Conditions

i=0
j=24
for code in sc:

  df_plotter = pd.DataFrame()
  df_plotter=merged_df2.iloc[i:j,:]
  df_plotter = df_plotter.drop(['State_Name','year_month','State Code_x','ID','
  df_plotter_2020=pd.DataFrame()
  df_plotter_2019=pd.DataFrame()
  df_plotter_2019=df_plotter.iloc[:12,]
  df_plotter_2020=df_plotter.iloc[12:,]

  if not df_plotter.empty:
      try:
          ax = plt.axes()
          sns.heatmap(df_plotter_2019, annot=True, ax=ax)
          ax.set_title(code)
          plt.show()
          print('----------------------')
          p2 = sns.heatmap(df_plotter_2020, annot=True, ax=ax)
          ax.set_title(code)
          plt.show()
          print('----------------------')
      except ValueError:
          print("Warning: zero-size array")


  i+=24
  j+=24
```



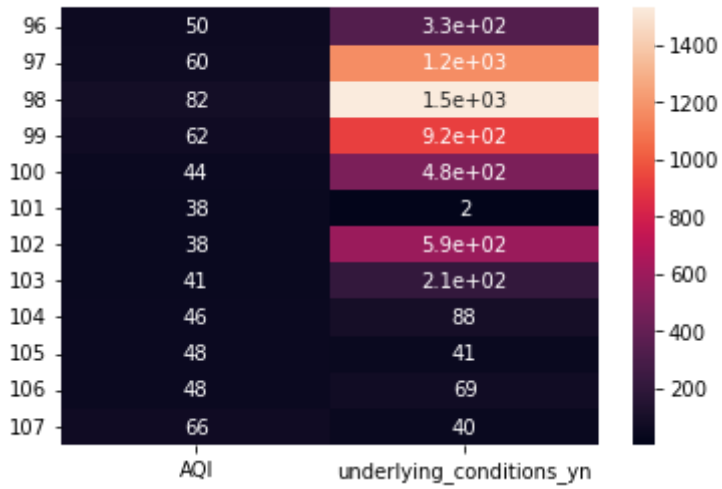

```
----------------------
----------------------
```


2
24 17 2.4e+02
25 26 2.1e+02
26 36 3.1e+02
27 32 2e+02
28 46 81
29 36 79
30 29 86
31 29 88
32 24 5
33 21 4
34 23 12
35 24 13
AQI
underlying_conditions_yn
300
250
200
150
100
50

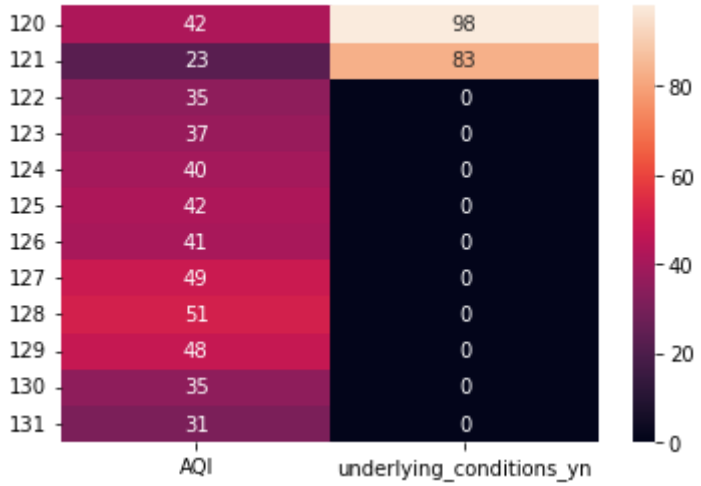

4
48 43 1.1e+03
49 48 9.3e+02
50 41 2.3e+02
51 55 1.2e+02
52 63 91
53 57 1e+02
54 59 81
55 55 2.3e+02
56 48 3.5e+02
57 45 2.4e+02
58 23 1.8e+02
59 35 6.2e+02
AQI
underlying_conditions_yn
1000
800
600
400
200


5
72 40 1.7e+03
73 37 1.7e+03
74 42 2.9e+03
75 46 9.3e+03
76 45 1.2e+04
77 45 8.2e+03
78 3 3.2e+03
79 46 1
80 47 2
81 40 1.5e+02
82 41 3.8e+02
83 50 6.5e+02
AQI
underlying_conditions_yn
12000
10000
8000
6000
4000
2000

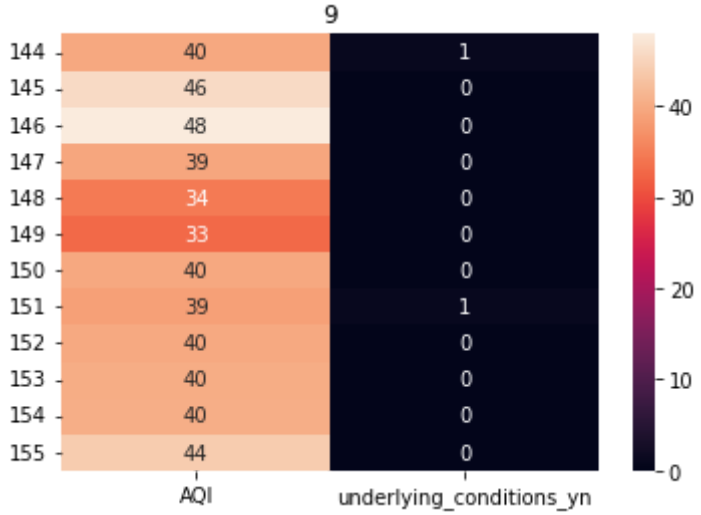
6

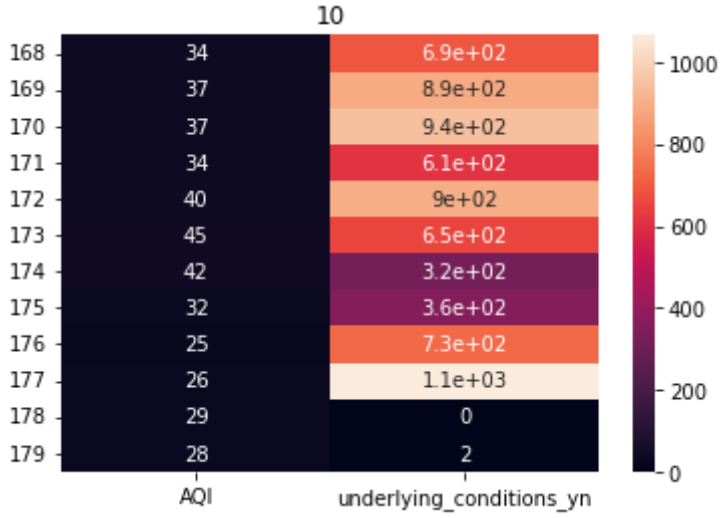

96 50 3.3e+02
97 60 1.2e+03
98 82 1.5e+03
99 62 9.2e+02
100 44 4.8e+02
101 38 2
102 38 5.9e+02
103 41 2.1e+02
104 46 88
105 48 41
106 48 69
107 66 40
AQI
underlying_conditions_yn
1400
1200
1000
800
600
400
200

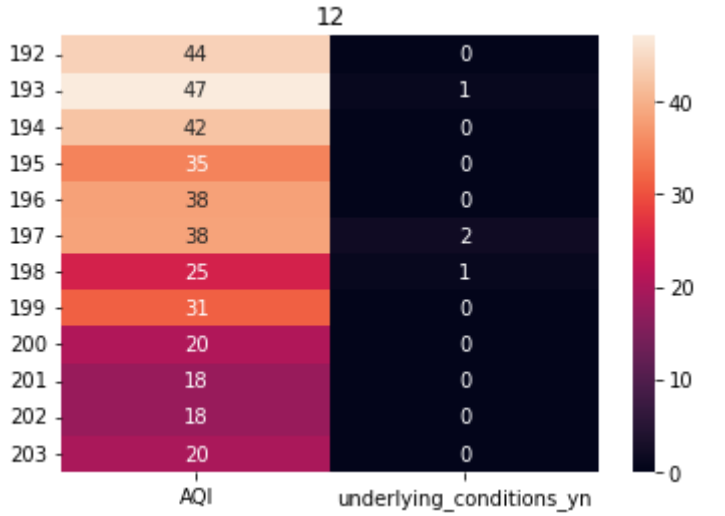
8

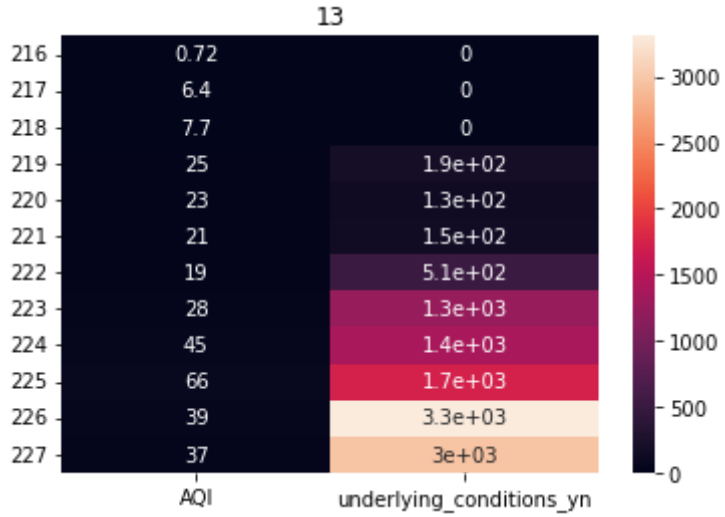

120 42 98
121 23 83
122 35 0
123 37 0
124 40 0
125 42 0
126 41 0
127 49 0
128 51 0
129 48 0
130 35 0
131 31 0
AQI
underlying_conditions_yn
80
60
40
20
0

9

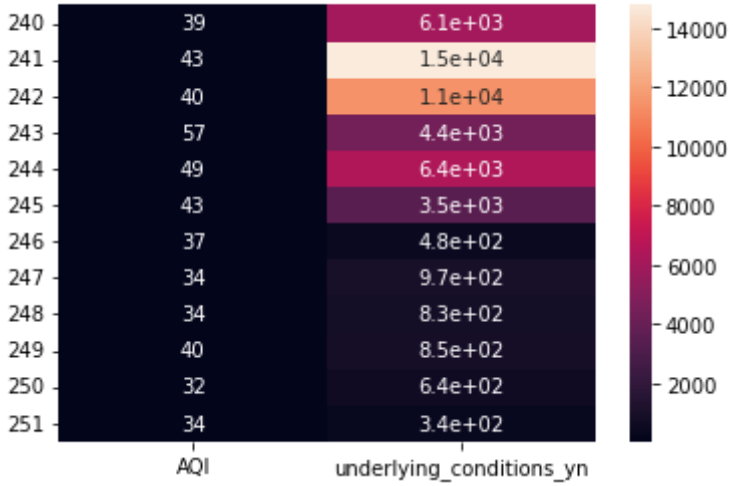

144 40 1
145 46 0
146 48 0
147 39 0
148 34 0
149 33 0
150 40 0
151 39 1
152 40 0
153 40 0
154 40 0
155 44 0
AQI
underlying_conditions_yn
40
30
20
10
0


10
168 34 6.9e+02
169 37 8.9e+02
170 37 9.4e+02
171 34 6.1e+02
172 40 9e+02
173 45 6.5e+02
174 42 3.2e+02
175 32 3.6e+02
176 25 7.3e+02
177 26 1.1e+03
178 29 0
179 28 2
AQI
underlying_conditions_yn
1000
800
600
400
200
0

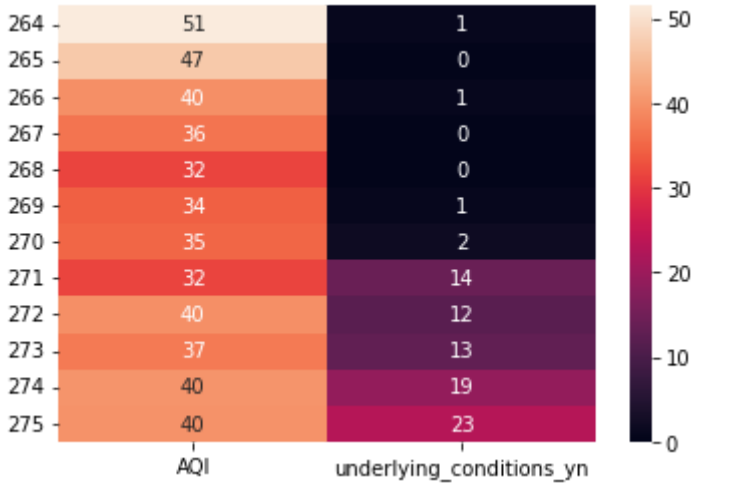

12
192 44 0
193 47 1
194 42 0
195 35 0
196 38 0
197 38 2
198 25 1
199 31 0
200 20 0
201 18 0
202 18 0
203 20 0
AQI
underlying_conditions_yn
40
30
20
10
0


13
216 0.72 0
217 6.4 0
218 7.7 0
219 25 1.9e+02
220 23 1.3e+02
221 21 1.5e+02
222 19 5.1e+02
223 28 1.3e+03
224 45 1.4e+03
225 66 1.7e+03
226 39 3.3e+03
227 37 3e+03
AQI
underlying_conditions_yn
3000
2500
2000
1500
1000
500
0

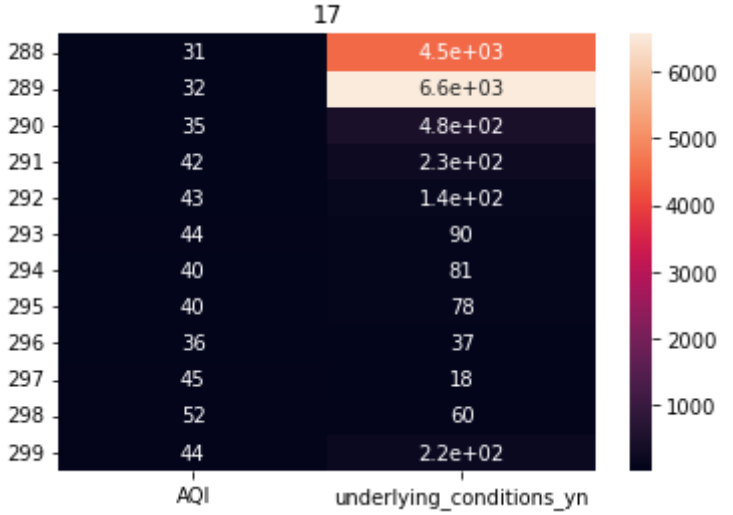
15

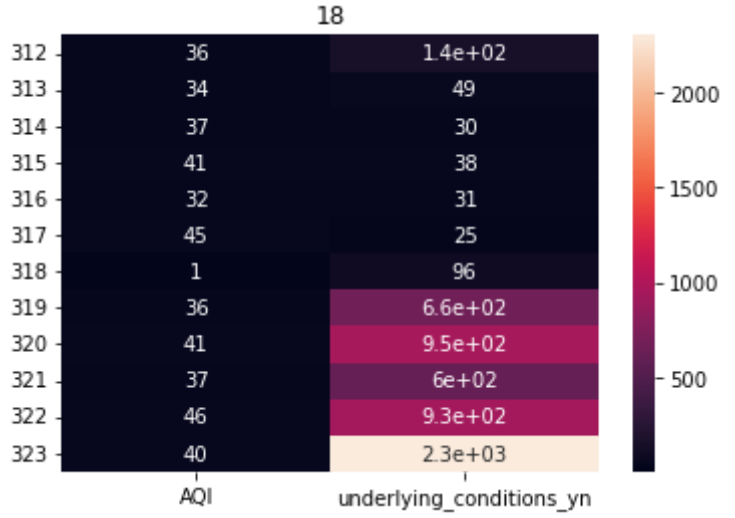

240 39 6.1e+03
241 43 1.5e+04
242 40 1.1e+04
243 57 4.4e+03
244 49 6.4e+03
245 43 3.5e+03
246 37 4.8e+02
247 34 9.7e+02
248 34 8.3e+02
249 40 8.5e+02
250 32 6.4e+02
251 34 3.4e+02
AQI
underlying_conditions_yn
14000
12000
10000
8000
6000
4000
2000

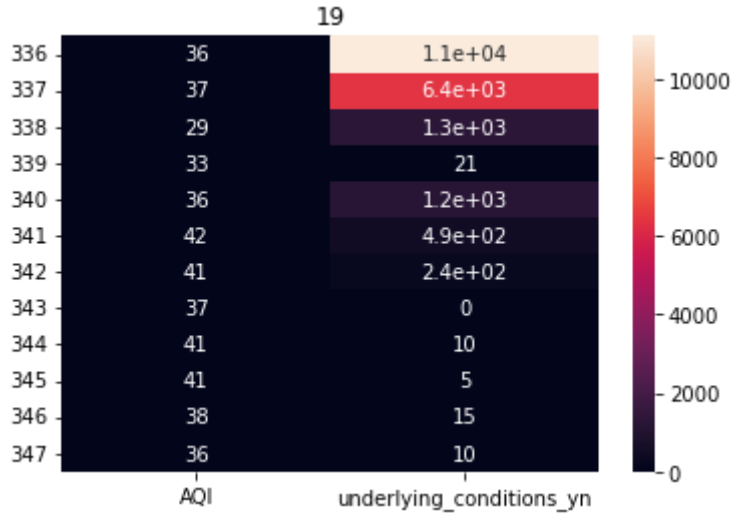
16

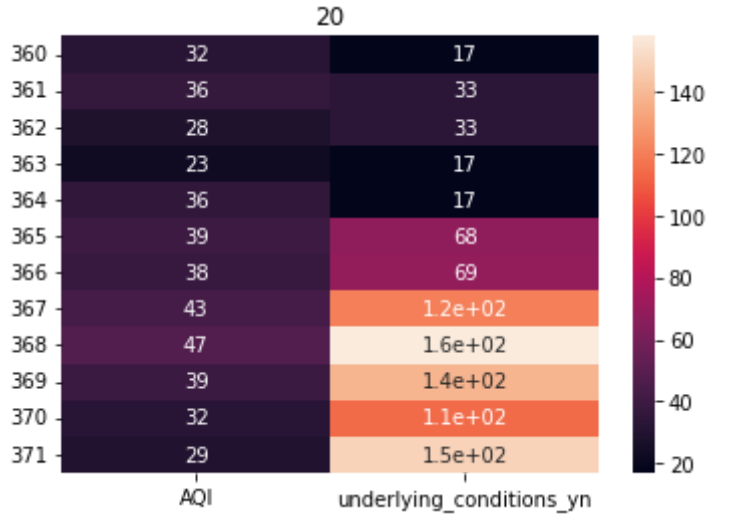

264 51 1
265 47 0
266 40 1
267 36 0
268 32 0
269 34 1
270 35 2
271 32 14
272 40 12
273 37 13
274 40 19
275 40 23
AQI
underlying_conditions_yn
50
40
30
20
10
0

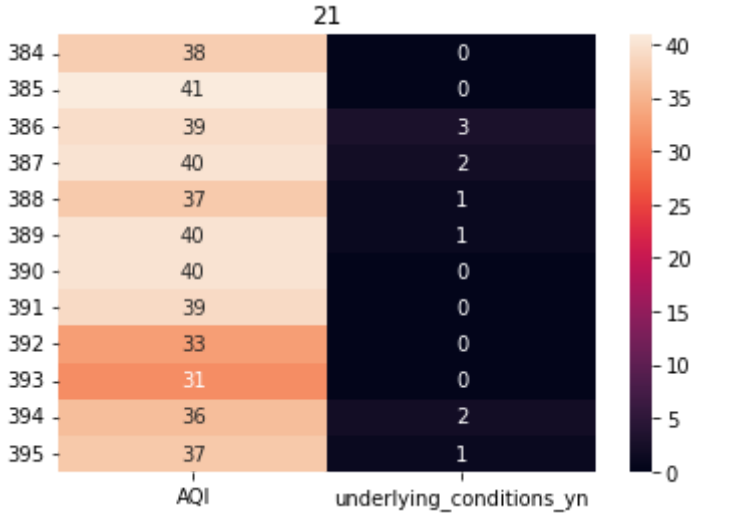
17

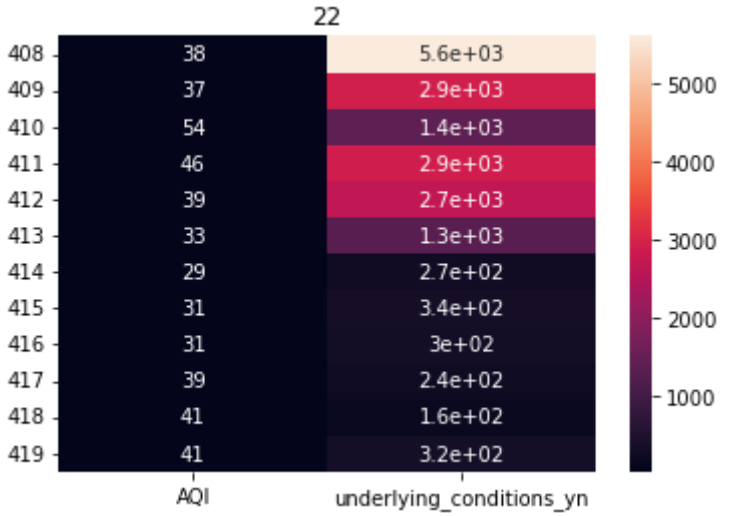

288 31 4.5e+03
289 32 6.6e+03
290 35 4.8e+02
291 42 2.3e+02
292 43 1.4e+02
293 44 90
294 40 81
295 40 78
296 36 37
297 45 18
298 52 60
299 44 2.2e+02
AQI
underlying_conditions_yn
6000
5000
4000
3000
2000
1000

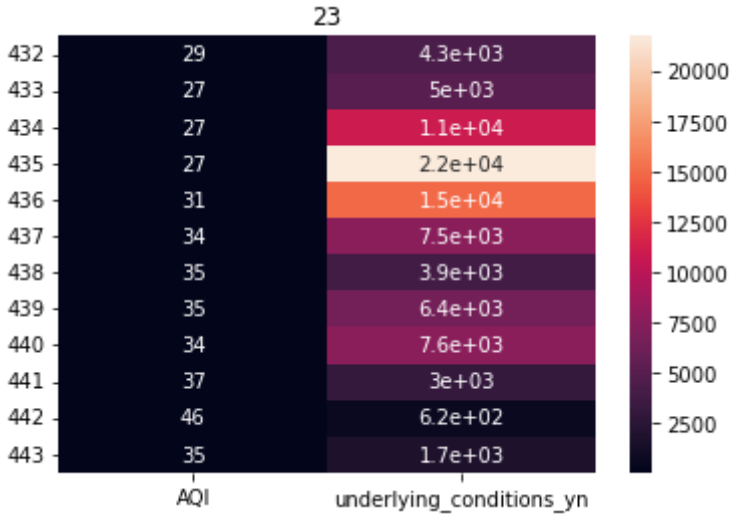

18
312
313
314
315
316
317
318
319
320
321
322
323
36
34
37
41
32
45
1
36
41
37
46
40
1.4e+02
49
30
38
31
25
96
6.6e+02
9.5e+02
6e+02
9.3e+02
2.3e+03
2000
1500
1000
500
AQI
underlying_conditions_yn


19
336
337
338
339
340
341
342
343
344
345
346
347
36
37
29
33
36
42
41
37
41
41
38
36
1.1e+04
6.4e+03
1.3e+03
21
1.2e+03
4.9e+02
2.4e+02
0
10
5
15
10
10000
8000
6000
4000
2000
0
AQI
underlying_conditions_yn

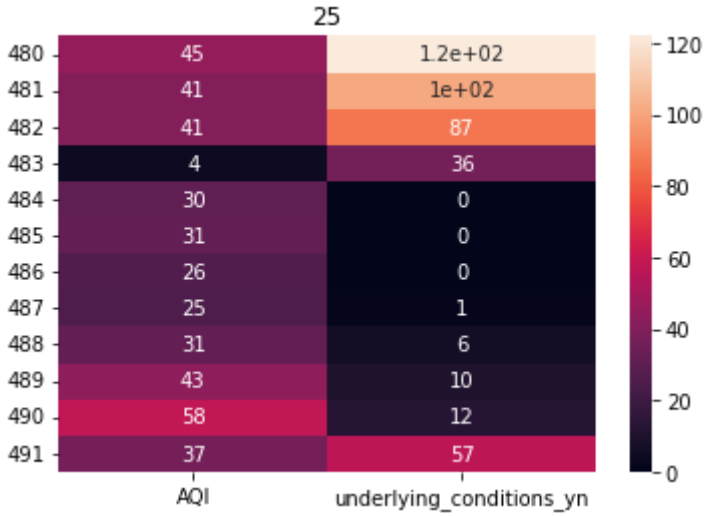

20
360
361
362
363
364
365
366
367
368
369
370
371
32
36
28
23
36
39
38
43
47
39
32
29
17
33
33
17
17
68
69
1.2e+02
1.6e+02
1.4e+02
1.1e+02
1.5e+02
140
120
100
80
60
40
20
AQI
underlying_conditions_yn

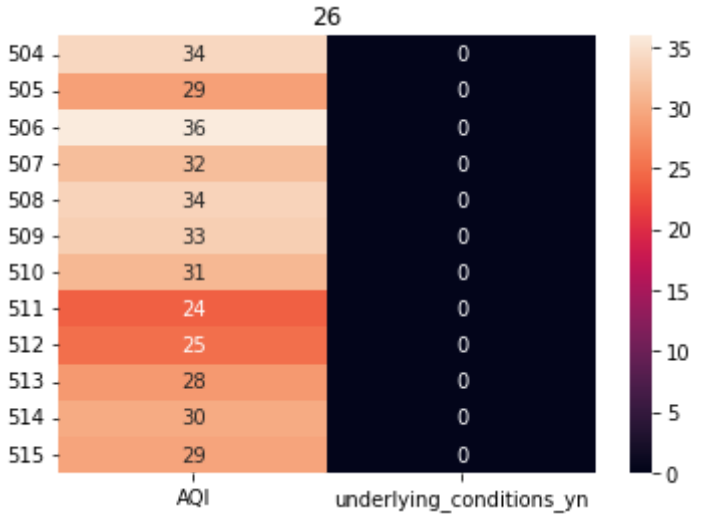

21
384 38 0
385 41 0
386 39 3
387 40 2
388 37 1
389 40 1
390 40 0
391 39 0
392 33 0
393 31 0
394 36 2
395 37 1
AQI
underlying_conditions_yn
40
35
30
25
20
15
10
5
0

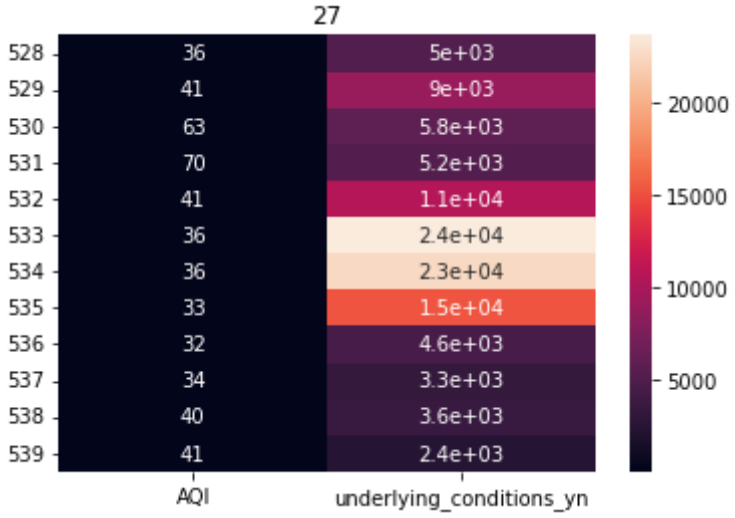

22
408 38 5.6e+03
409 37 2.9e+03
410 54 1.4e+03
411 46 2.9e+03
412 39 2.7e+03
413 33 1.3e+03
414 29 2.7e+02
415 31 3.4e+02
416 31 3e+02
417 39 2.4e+02
418 41 1.6e+02
419 41 3.2e+02
AQI
underlying_conditions_yn
5000
4000
3000
2000
1000

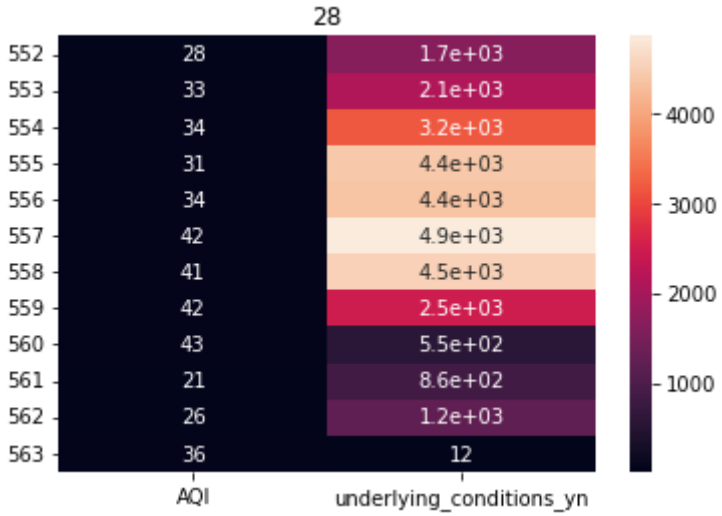

23
432 29 4.3e+03
433 27 5e+03
434 27 1.1e+04
435 27 2.2e+04
436 31 1.5e+04
437 34 7.5e+03
438 35 3.9e+03
439 35 6.4e+03
440 34 7.6e+03
441 37 3e+03
442 46 6.2e+02
443 35 1.7e+03
AQI
underlying_conditions_yn
20000
17500
15000
12500
10000
7500
5000
2500

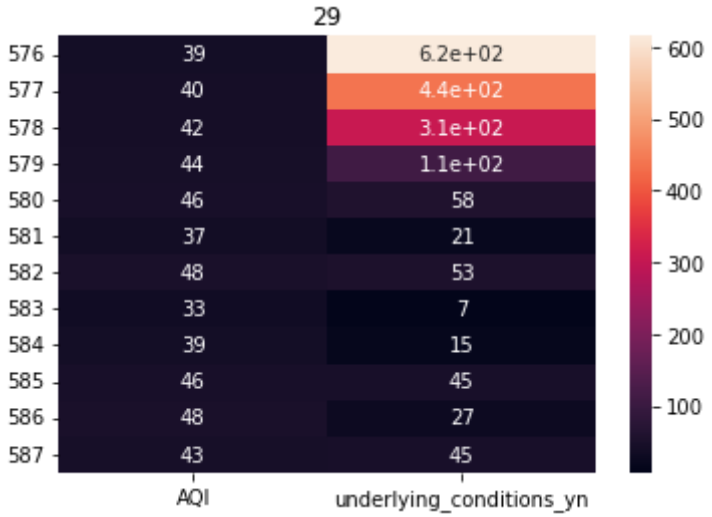

24
456 35 0
457 37 1
458 42 0
459 43 0
460 40 0
461 37 0
462 27 0
463 30 0
464 36 11
465 41 0
466 37 2
467 49 3
AQI
underlying_conditions_yn
40
30
20
10
0

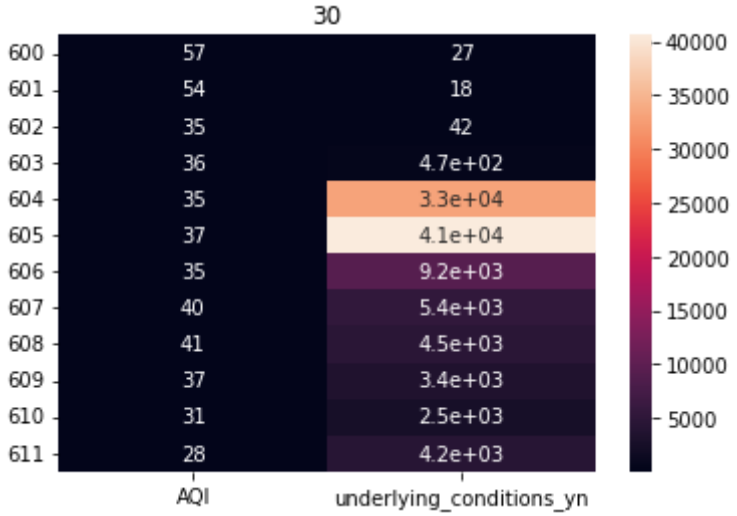

25
480 45 1.2e+02
481 41 1e+02
482 41 87
483 4 36
484 30 0
485 31 0
486 26 0
487 25 1
488 31 6
489 43 10
490 58 12
491 37 57
AQI
underlying_conditions_yn
120
100
80
60
40
20
0

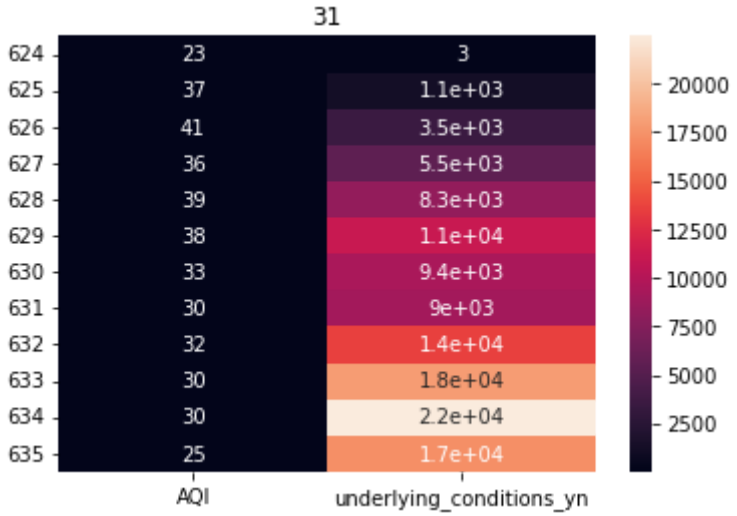

26
504 34 0
505 29 0
506 36 0
507 32 0
508 34 0
509 33 0
510 31 0
511 24 0
512 25 0
513 28 0
514 30 0
515 29 0
AQI
underlying_conditions_yn
35
30
25
20
15
10
5
0


27
528 36 5e+03
529 41 9e+03
530 63 5.8e+03
531 70 5.2e+03
532 41 1.1e+04
533 36 2.4e+04
534 36 2.3e+04
535 33 1.5e+04
536 32 4.6e+03
537 34 3.3e+03
538 40 3.6e+03
539 41 2.4e+03
AQI
underlying_conditions_yn
20000
15000
10000
5000


28
552 28 1.7e+03
553 33 2.1e+03
554 34 3.2e+03
555 31 4.4e+03
556 34 4.4e+03
557 42 4.9e+03
558 41 4.5e+03
559 42 2.5e+03
560 43 5.5e+02
561 21 8.6e+02
562 26 1.2e+03
563 36 12
AQI
underlying_conditions_yn
4000
3000
2000
1000

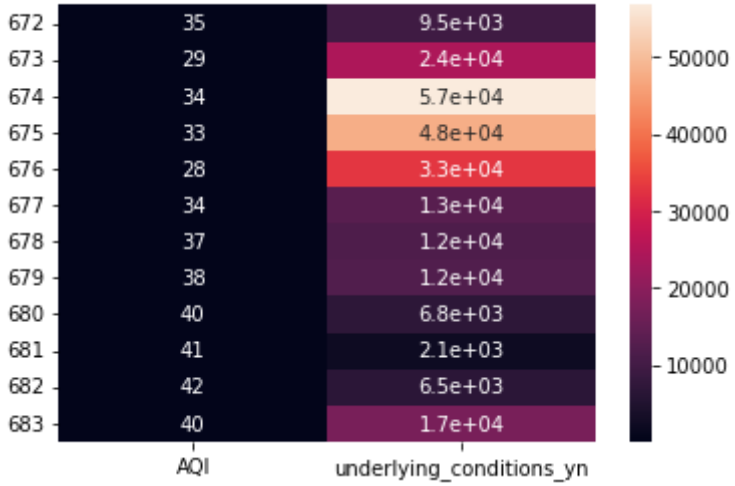

29
576 39 6.2e+02
577 40 4.4e+02
578 42 3.1e+02
579 44 1.1e+02
580 46 58
581 37 21
582 48 53
583 33 7
584 39 15
585 46 45
586 48 27
587 43 45
AQI
underlying_conditions_yn
600
500
400
300
200
100


30
600 57 27
601 54 18
602 35 42
603 36 4.7e+02
604 35 3.3e+04
605 37 4.1e+04
606 35 9.2e+03
607 40 5.4e+03
608 41 4.5e+03
609 37 3.4e+03
610 31 2.5e+03
611 28 4.2e+03
AQI
underlying_conditions_yn
40000
35000
30000
25000
20000
15000
10000
5000

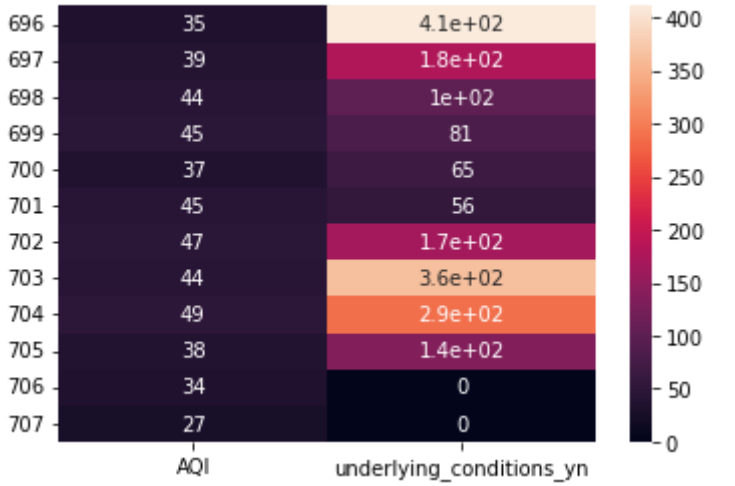

31
624 23 3
625 37 1.1e+03
626 41 3.5e+03
627 36 5.5e+03
628 39 8.3e+03
629 38 1.1e+04
630 33 9.4e+03
631 30 9e+03
632 32 1.4e+04
633 30 1.8e+04
634 30 2.2e+04
635 25 1.7e+04
AQI
underlying_conditions_yn
20000
17500
15000
12500
10000
7500
5000
2500


32
648 41 0
649 34 0
650 38 0
651 35 0
652 31 0
653 31 0
654 29 0
655 32 0
656 29 0
657 33 0
658 36 0
659 37 0
AQI
underlying_conditions_yn
40
35
30
25
20
15
10
5
0

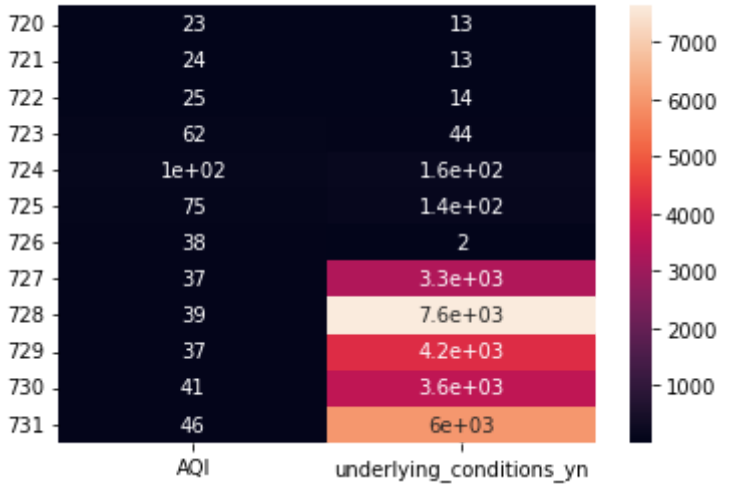
33

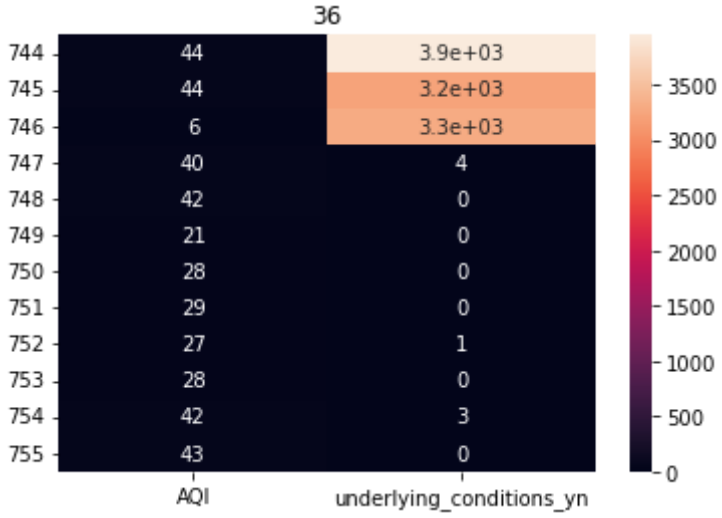

672 35 9.5e+03
673 29 2.4e+04
674 34 5.7e+04
675 33 4.8e+04
676 28 3.3e+04
677 34 1.3e+04
678 37 1.2e+04
679 38 1.2e+04
680 40 6.8e+03
681 41 2.1e+03
682 42 6.5e+03
683 40 1.7e+04
AQI
underlying_conditions_yn
50000
40000
30000
20000
10000

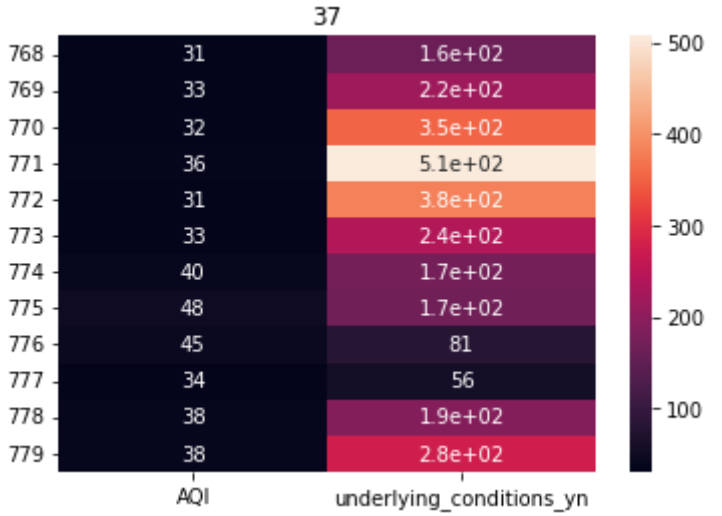
34

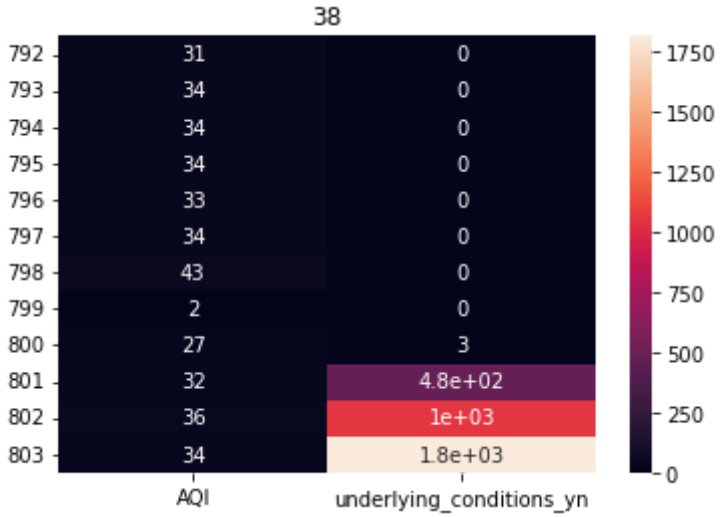

696 35 4.1e+02
697 39 1.8e+02
698 44 1e+02
699 45 81
700 37 65
701 45 56
702 47 1.7e+02
703 44 3.6e+02
704 49 2.9e+02
705 38 1.4e+02
706 34 0
707 27 0
AQI
underlying_conditions_yn
400
350
300
250
200
150
100
50
0

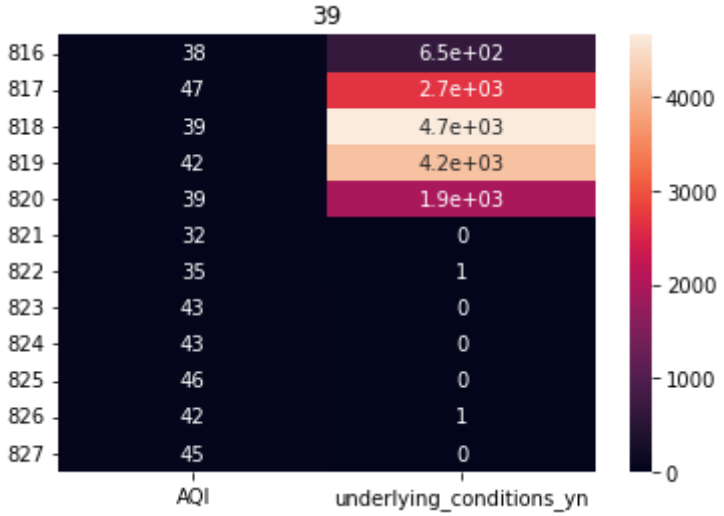
35

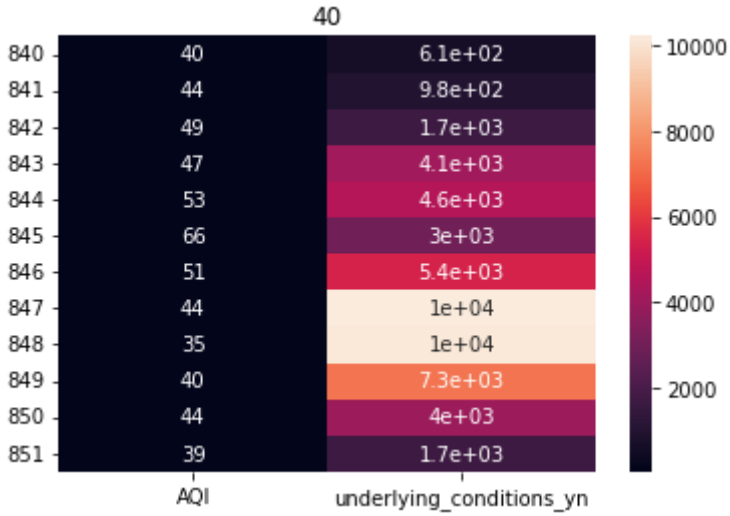

720 23 13
721 24 13
722 25 14
723 62 44
724 1e+02 1.6e+02
725 75 1.4e+02
726 38 2
727 37 3.3e+03
728 39 7.6e+03
729 37 4.2e+03
730 41 3.6e+03
731 46 6e+03
AQI
underlying_conditions_yn
7000
6000
5000
4000
3000
2000
1000

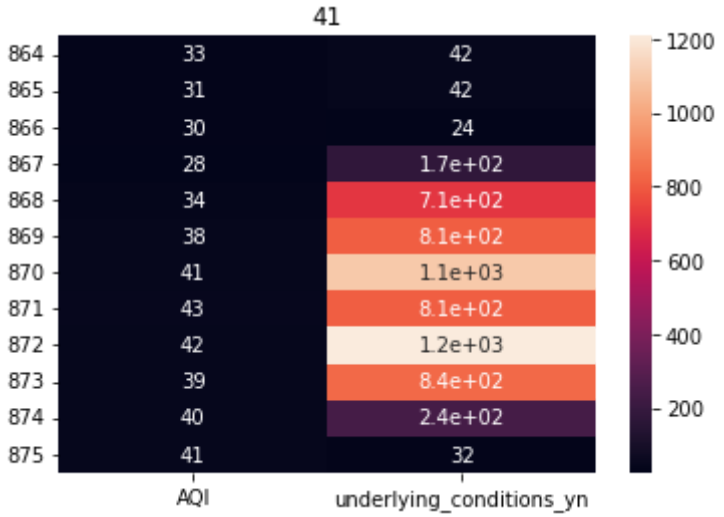

36
744 44 3.9e+03
745 44 3.2e+03
746 6 3.3e+03
747 40 4
748 42 0
749 21 0
750 28 0
751 29 0
752 27 1
753 28 0
754 42 3
755 43 0
AQI
underlying_conditions_yn
3500
3000
2500
2000
1500
1000
500
0

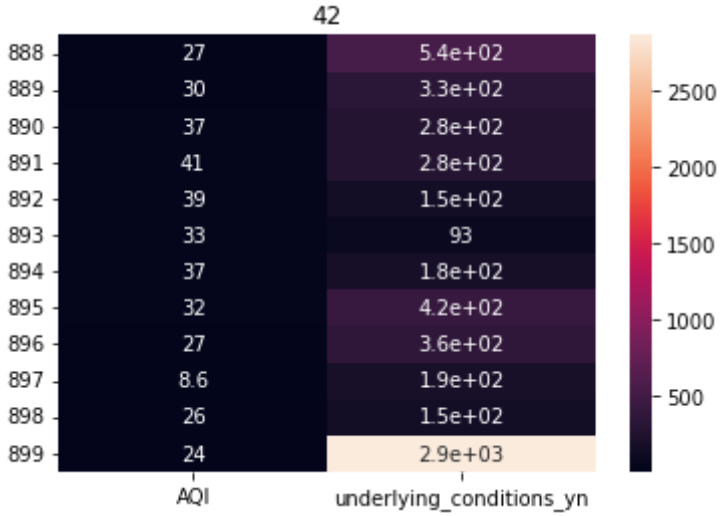

37
768 31 1.6e+02
769 33 2.2e+02
770 32 3.5e+02
771 36 5.1e+02
772 31 3.8e+02
773 33 2.4e+02
774 40 1.7e+02
775 48 1.7e+02
776 45 81
777 34 56
778 38 1.9e+02
779 38 2.8e+02
AQI
underlying_conditions_yn
500
400
300
200
100

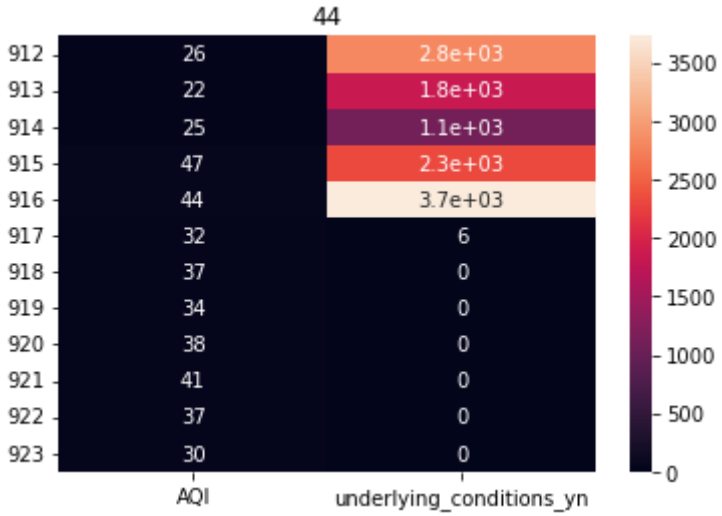

38
792 31 0
793 34 0
794 34 0
795 34 0
796 33 0
797 34 0
798 43 0
799 2 0
800 27 3
801 32 4.8e+02
802 36 1e+03
803 34 1.8e+03
AQI
underlying_conditions_yn
1750
1500
1250
1000
750
500
250
0

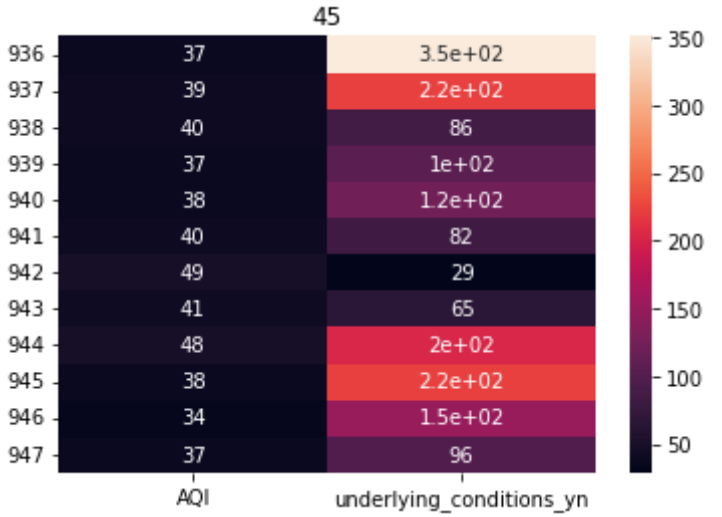

39
816 38 6.5e+02
817 47 2.7e+03
818 39 4.7e+03
819 42 4.2e+03
820 39 1.9e+03
821 32 0
822 35 1
823 43 0
824 43 0
825 46 0
826 42 1
827 45 0
AQI
underlying_conditions_yn
4000
3000
2000
1000
0

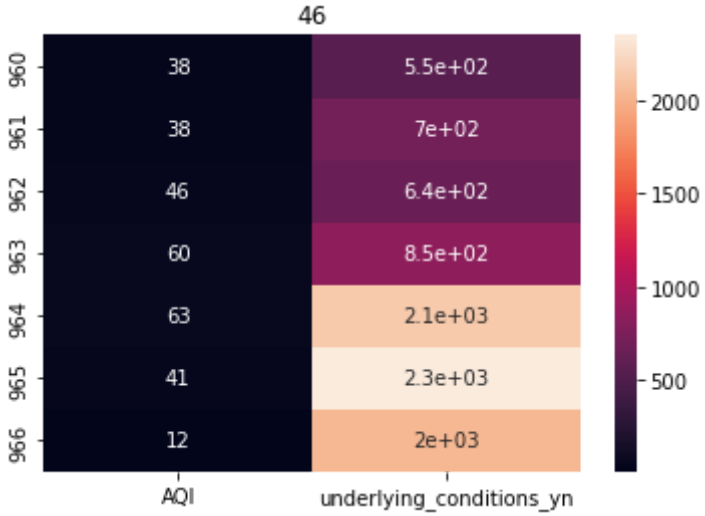

40
840 40 6.1e+02
841 44 9.8e+02
842 49 1.7e+03
843 47 4.1e+03
844 53 4.6e+03
845 66 3e+03
846 51 5.4e+03
847 44 1e+04
848 35 1e+04
849 40 7.3e+03
850 44 4e+03
851 39 1.7e+03
AQI
underlying_conditions_yn
10000
8000
6000
4000
2000


41
864 33 42
865 31 42
866 30 24
867 28 1.7e+02
868 34 7.1e+02
869 38 8.1e+02
870 41 1.1e+03
871 43 8.1e+02
872 42 1.2e+03
873 39 8.4e+02
874 40 2.4e+02
875 41 32
AQI
underlying_conditions_yn
1200
1000
800
600
400
200


42
888 27 5.4e+02
889 30 3.3e+02
890 37 2.8e+02
891 41 2.8e+02
892 39 1.5e+02
893 33 93
894 37 1.8e+02
895 32 4.2e+02
896 27 3.6e+02
897 8.6 1.9e+02
898 26 1.5e+02
899 24 2.9e+03
AQI
underlying_conditions_yn
2500
2000
1500
1000
500


44
912 26 2.8e+03
913 22 1.8e+03
914 25 1.1e+03
915 47 2.3e+03
916 44 3.7e+03
917 32 6
918 37 0
919 34 0
920 38 0
921 41 0
922 37 0
923 30 0
AQI
underlying_conditions_yn
3500
3000
2500
2000
1500
1000
500
0


45
936 37 3.5e+02
937 39 2.2e+02
938 40 86
939 37 1e+02
940 38 1.2e+02
941 40 82
942 49 29
943 41 65
944 48 2e+02
945 38 2.2e+02
946 34 1.5e+02
947 37 96
AQI
underlying_conditions_yn
350
300
250
200
150
100
50



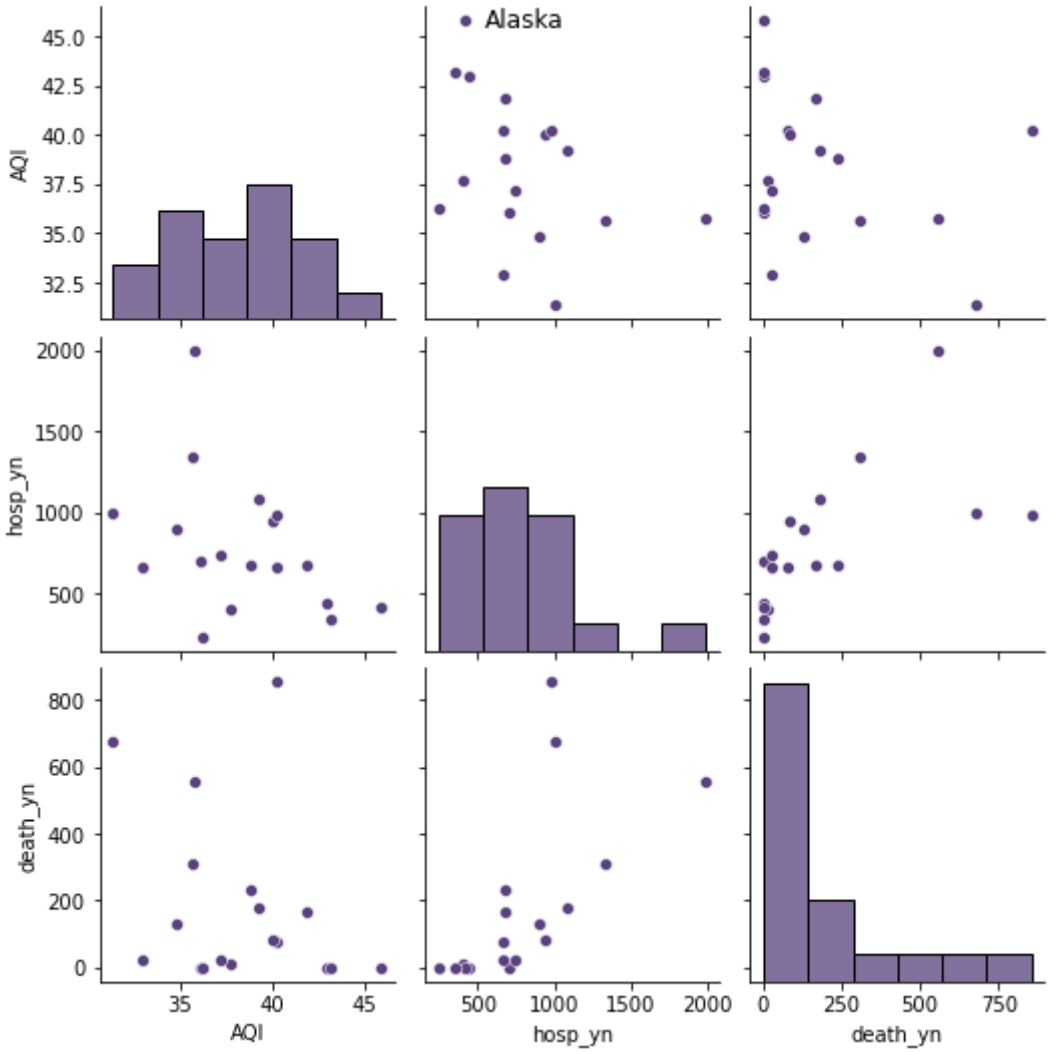

```
----------------------
Warning: zero-size array
```

After Analysing the Underlying conditions of the Dataframe: Underlying_Conditions states that the patient have one or more of the underlying medical conditions and risk behaviors: diabetes mellitus, hypertension, severe obesity (BMI>40), cardiovascular disease, chronic renal disease, chronic liver disease, chronic lung disease, other chronic diseases, immunosuppressive condition, autoimmune condition, current smoker, former smoker, substance abuse or misuse, disability, psychological/psychiatric, pregnancy, other. Which is specified in terms of [Yes, No, Unknown]

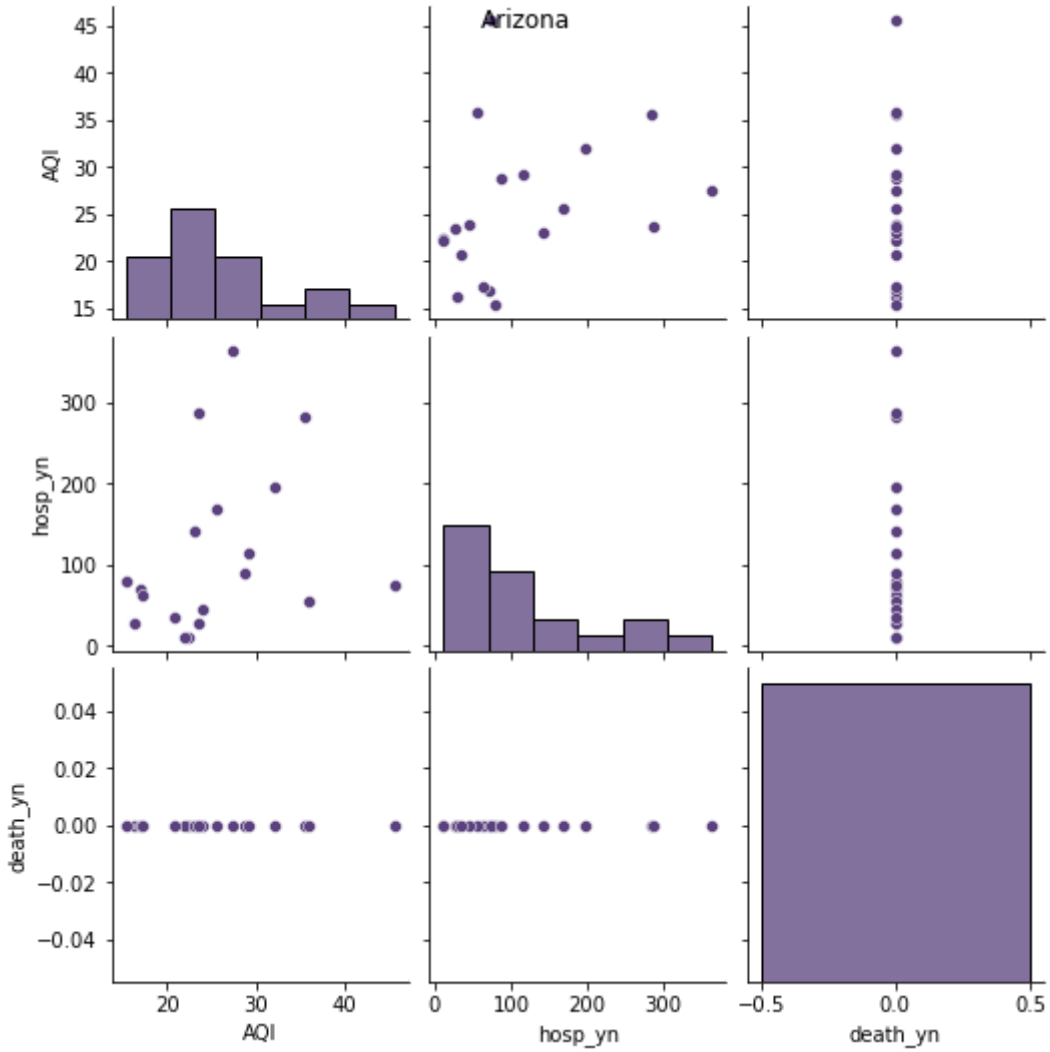
Based on the Above heat maps, the AQI has no correlation with the Underlying_Heatmaps.

In [72]:
```
for country in country_list:
    cn=[]
    cn.append(country)
    df_plotter = pd.DataFrame()
    df_plotter=merged_df2[merged_df2['State_Name'].isin(cn)]
    t1 = df_plotter[['AQI', 'hosp_yn', 'death_yn']]
    plt.suptitle(country)
    sns.set_palette("Purples_r", desat=0.6)
    plt.figure(figsize=(15,7))
    sns.pairplot(t1,diag_kind='hist')
```

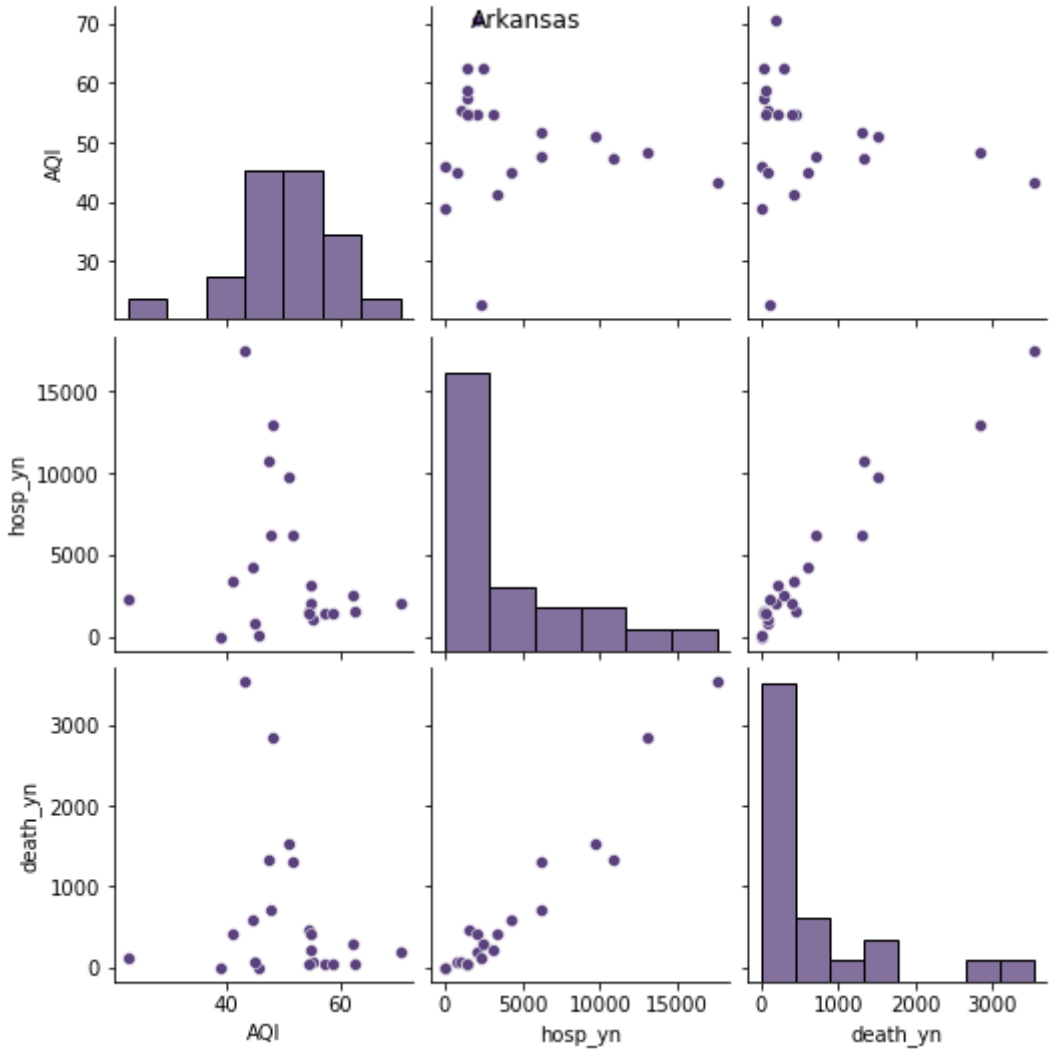
```
<Figure size 432x288 with 0 Axes>
<Figure size 1080x504 with 0 Axes>
```



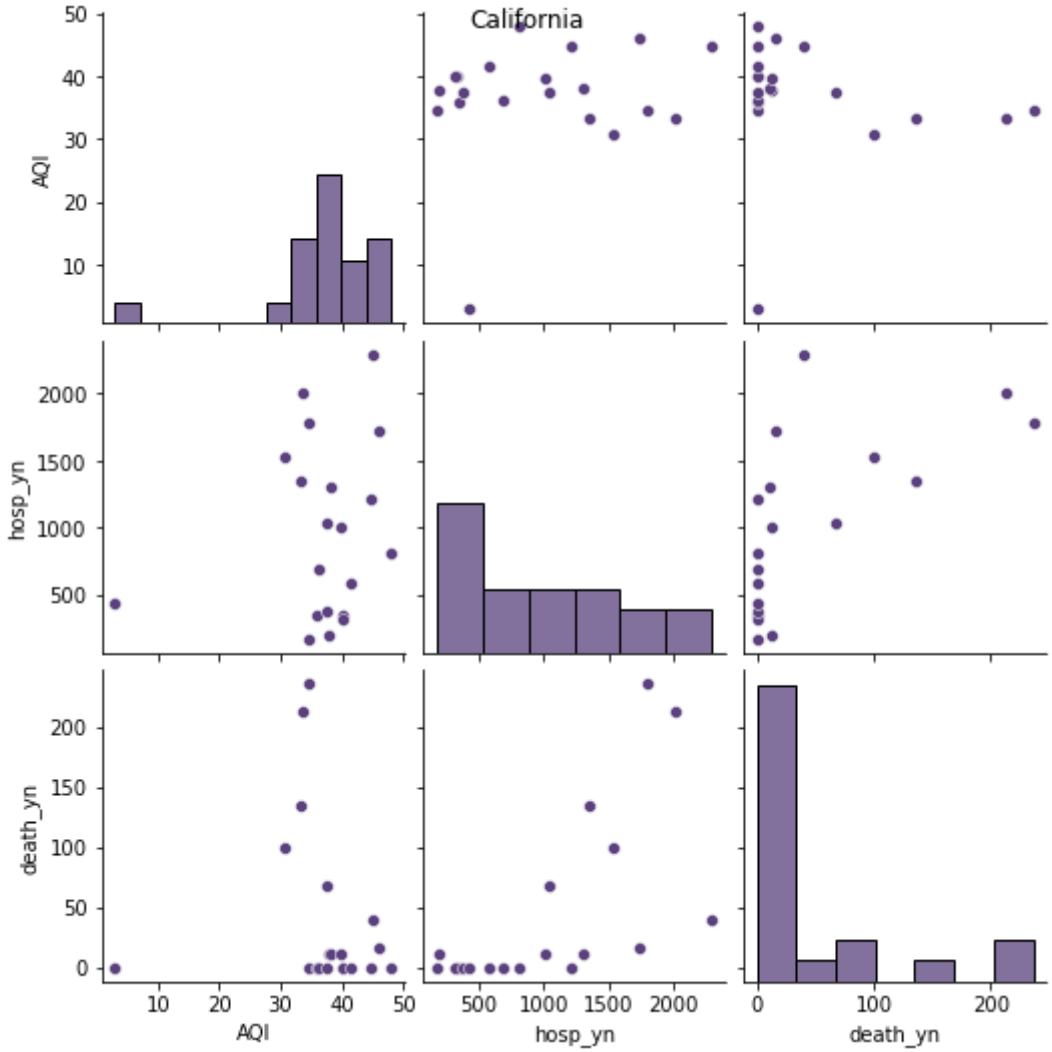

```
<Figure size 1080x504 with 0 Axes>
```



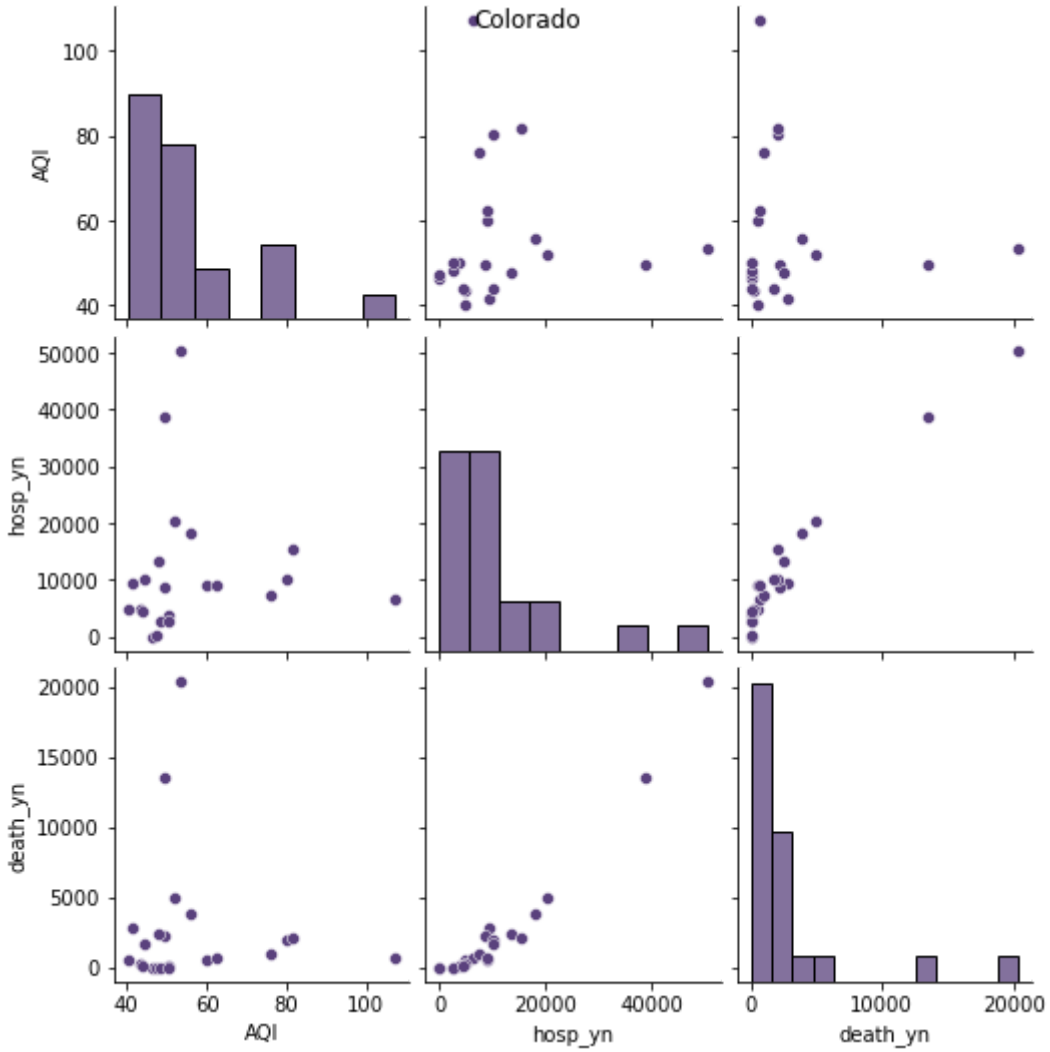

```
<Figure size 1080x504 with 0 Axes>
```



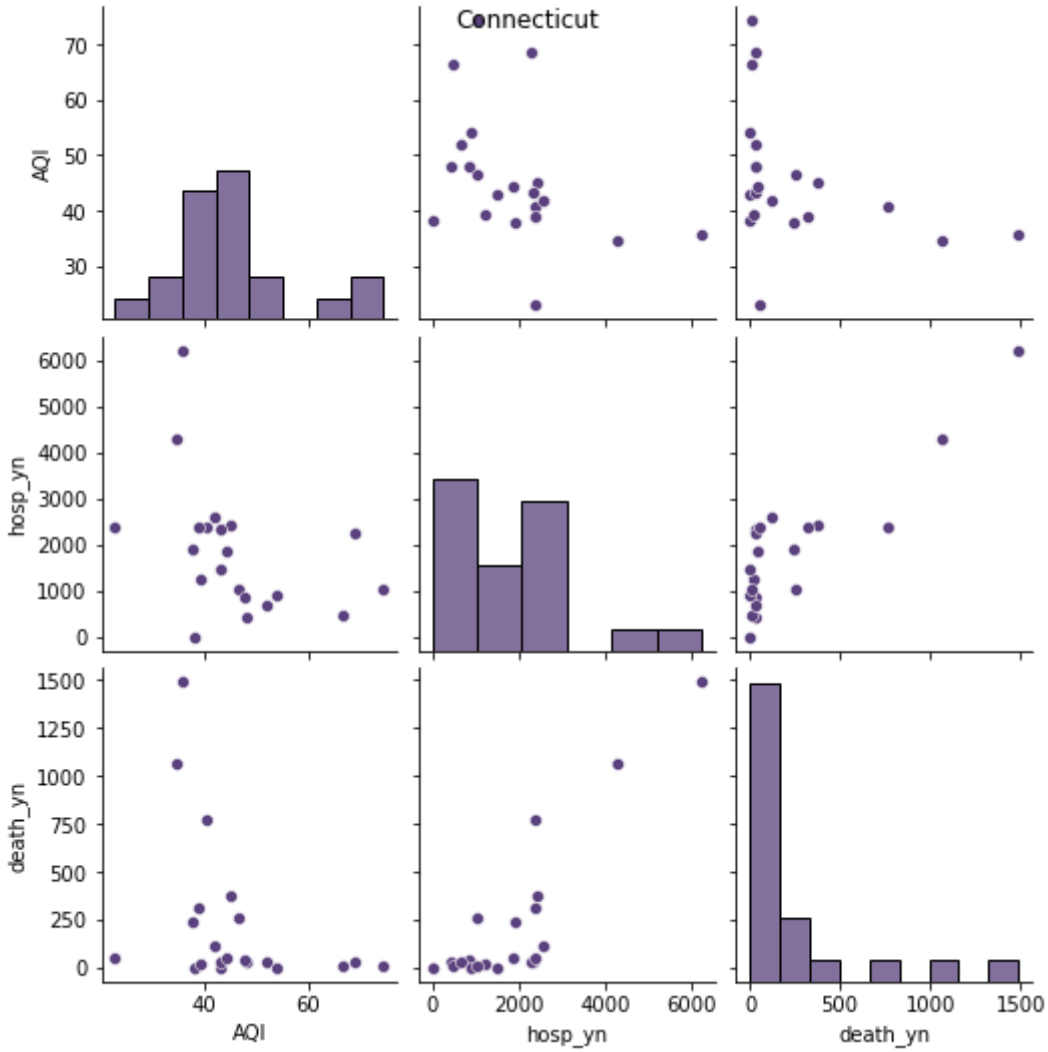

```
<Figure size 1080x504 with 0 Axes>
```



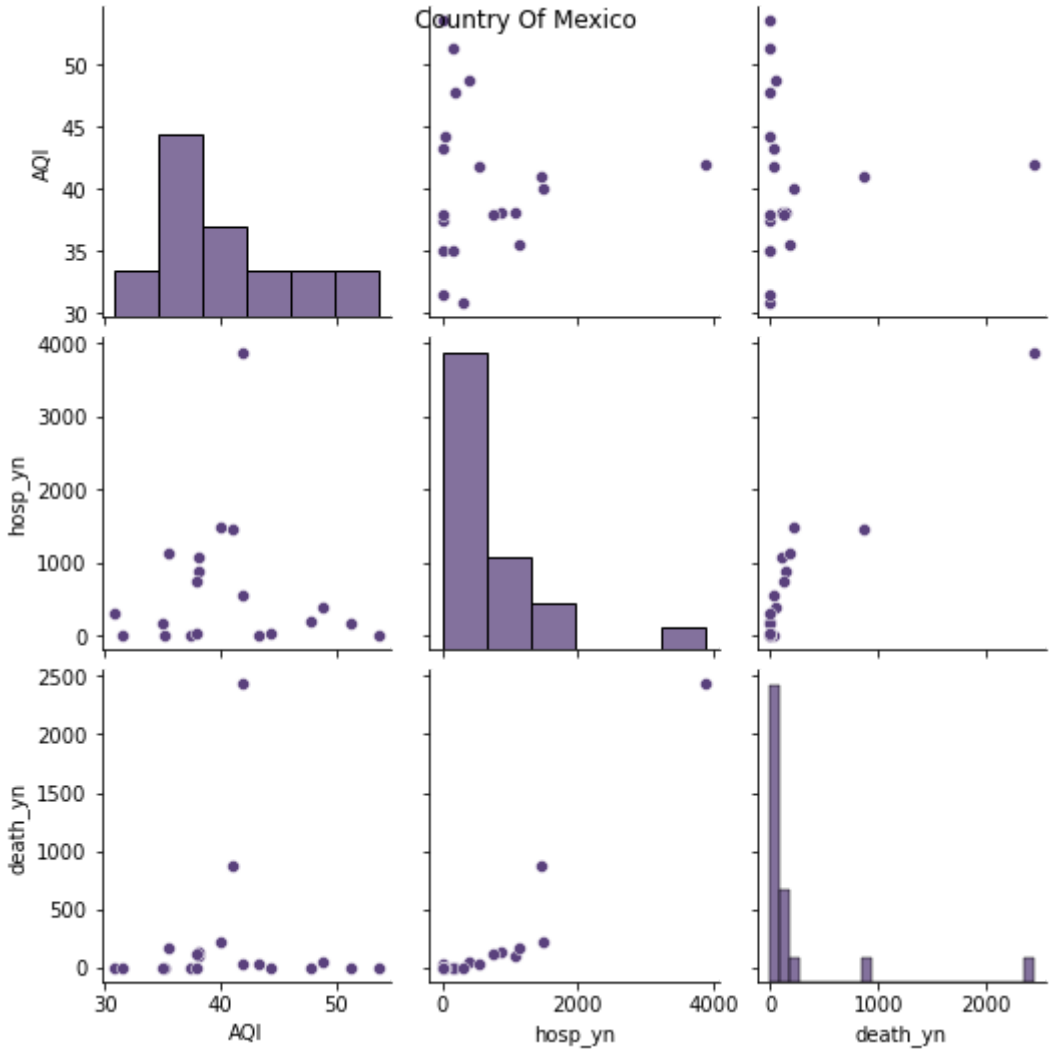

```
<Figure size 1080x504 with 0 Axes>
```



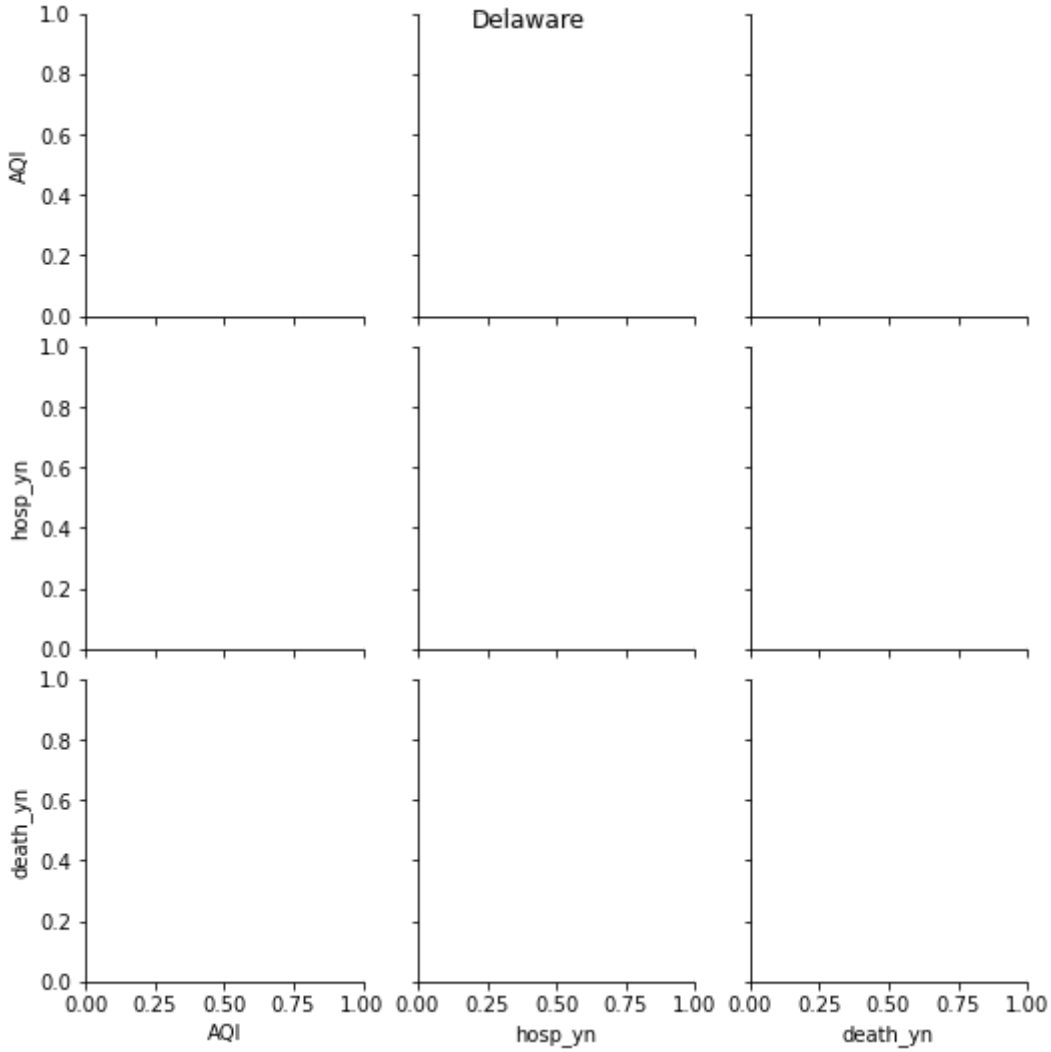
<Figure size 1080x504 with 0 Axes>



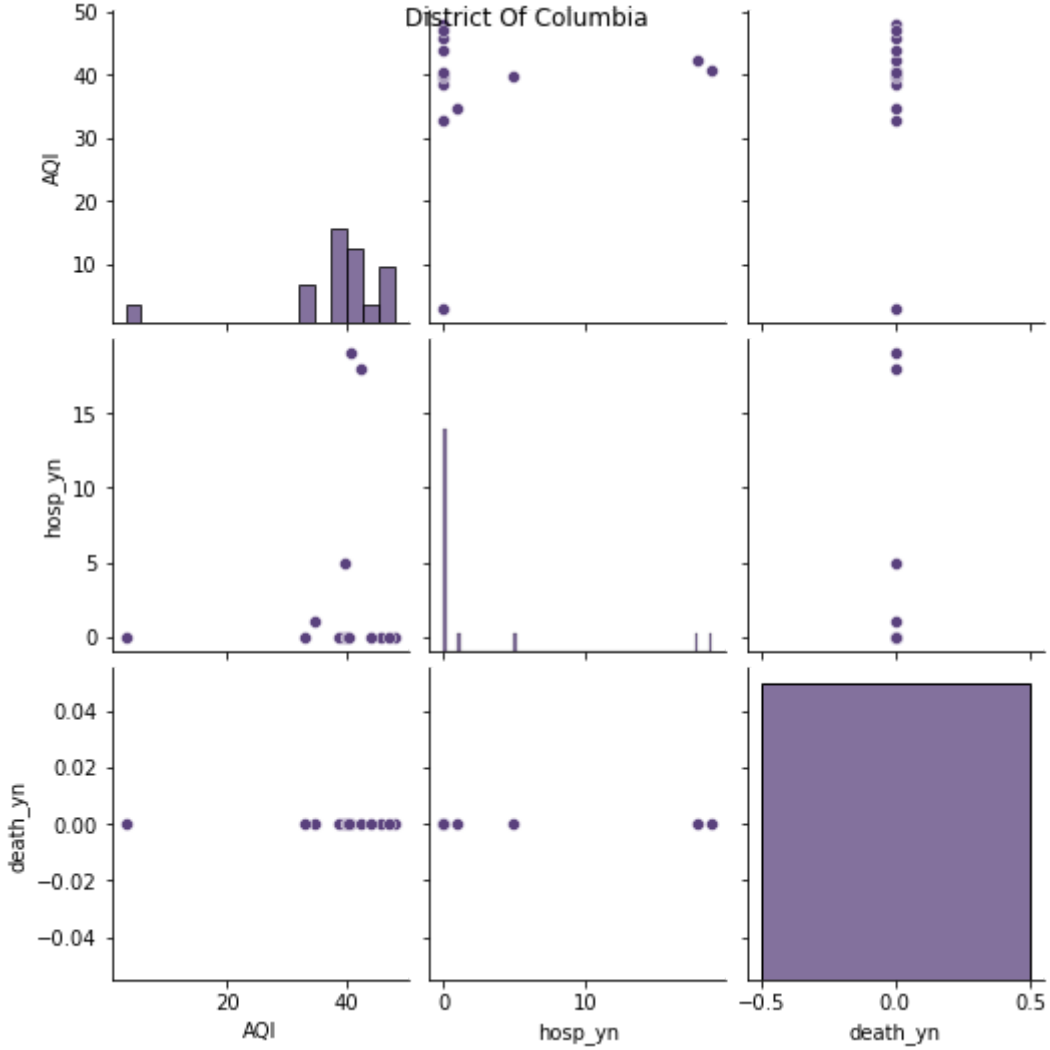

```
<Figure size 1080x504 with 0 Axes>
```



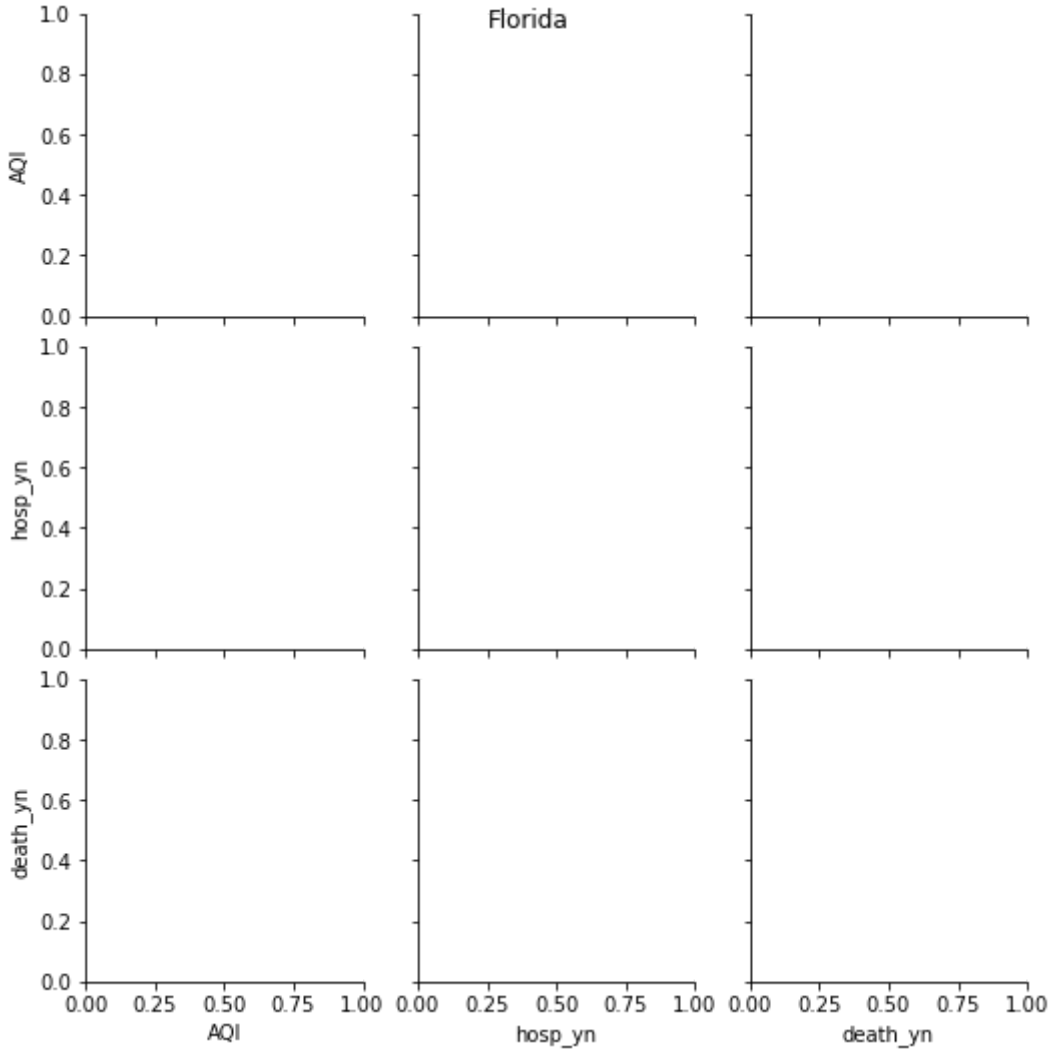

```
<Figure size 1080x504 with 0 Axes>
```



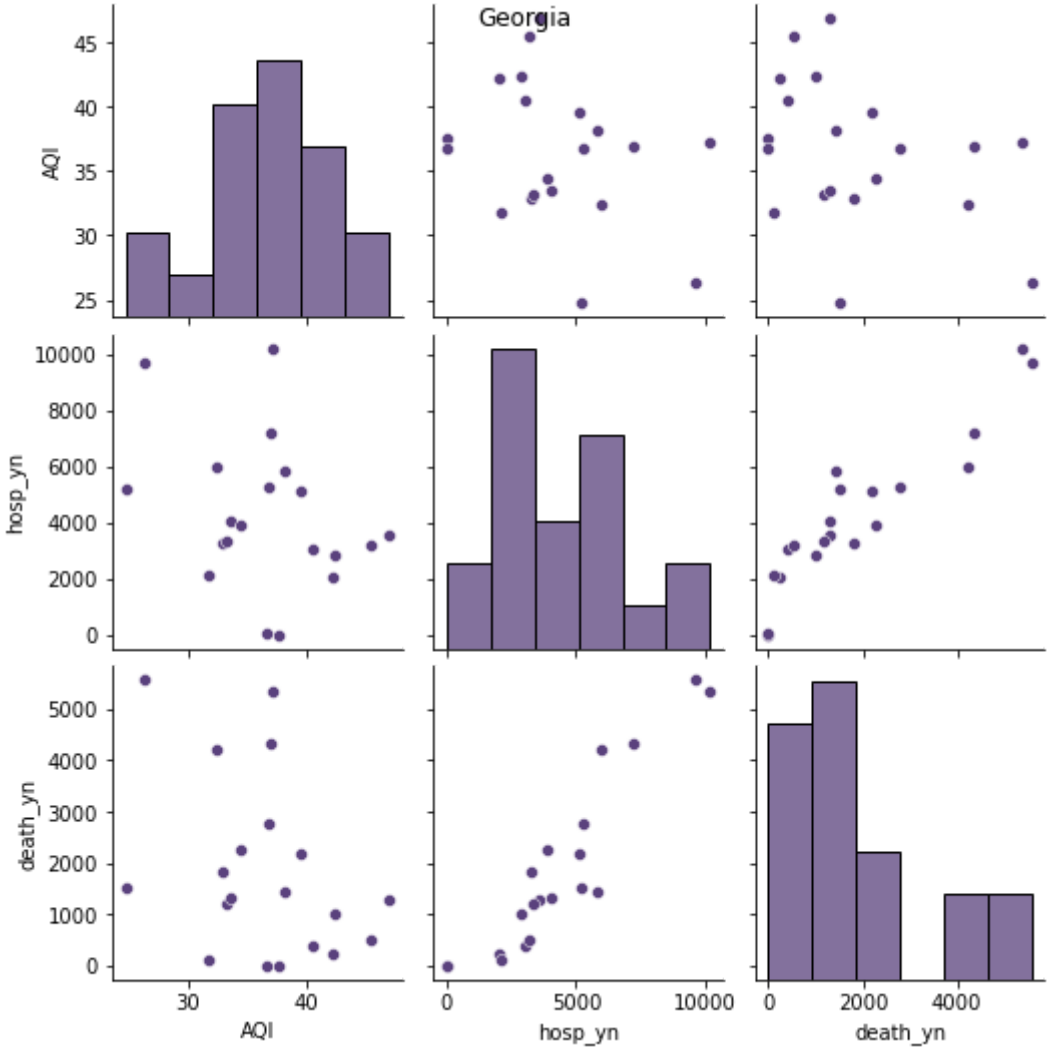

```
<Figure size 1080x504 with 0 Axes>
```



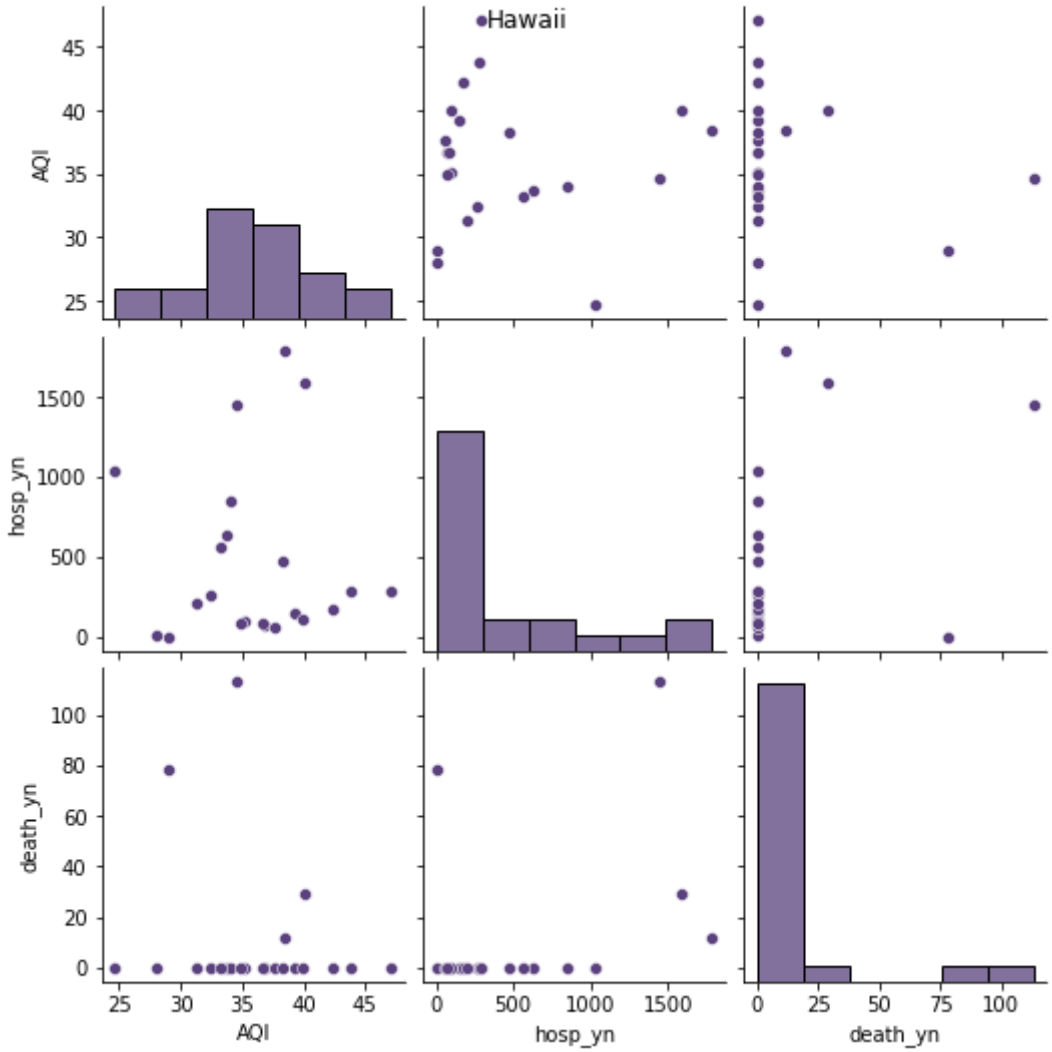

```
<Figure size 1080x504 with 0 Axes>
```



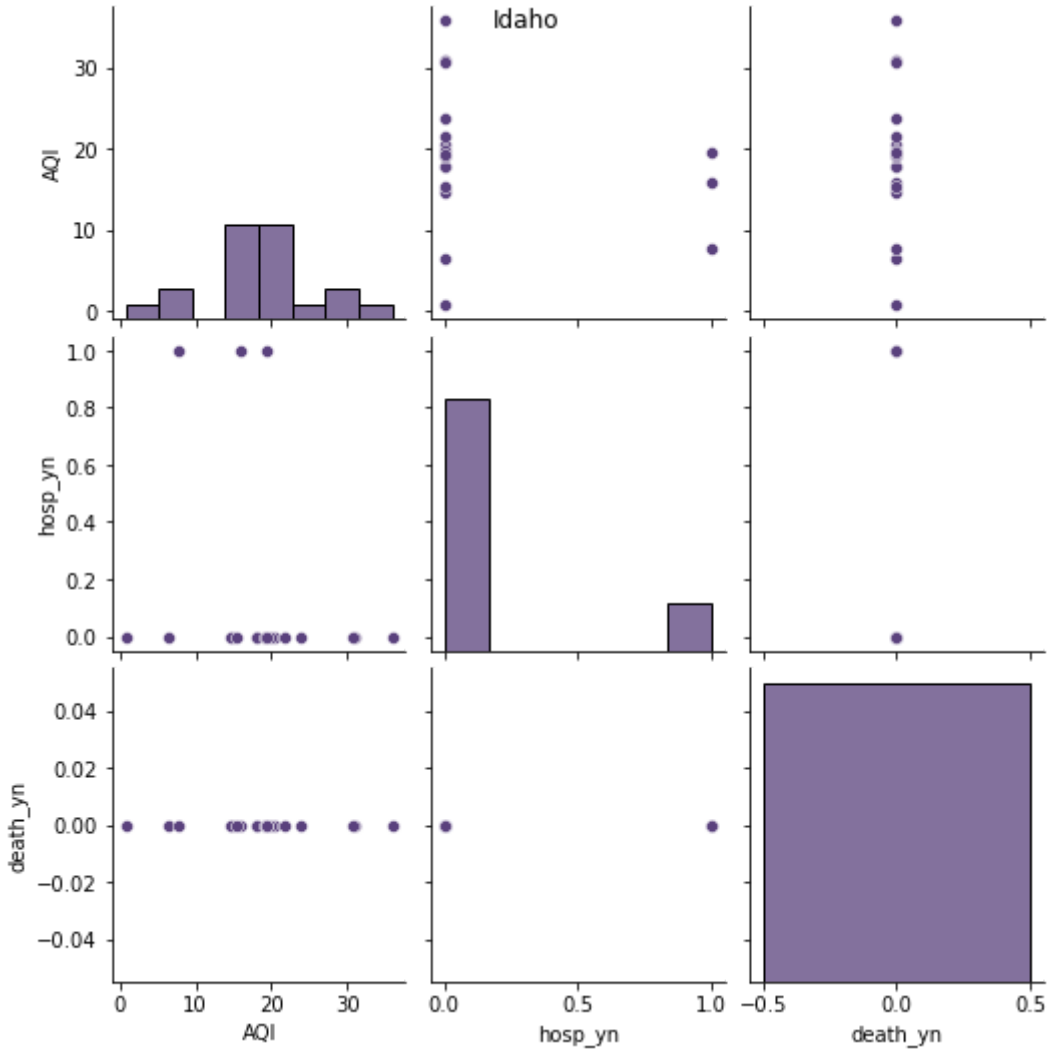

```
<Figure size 1080x504 with 0 Axes>
```



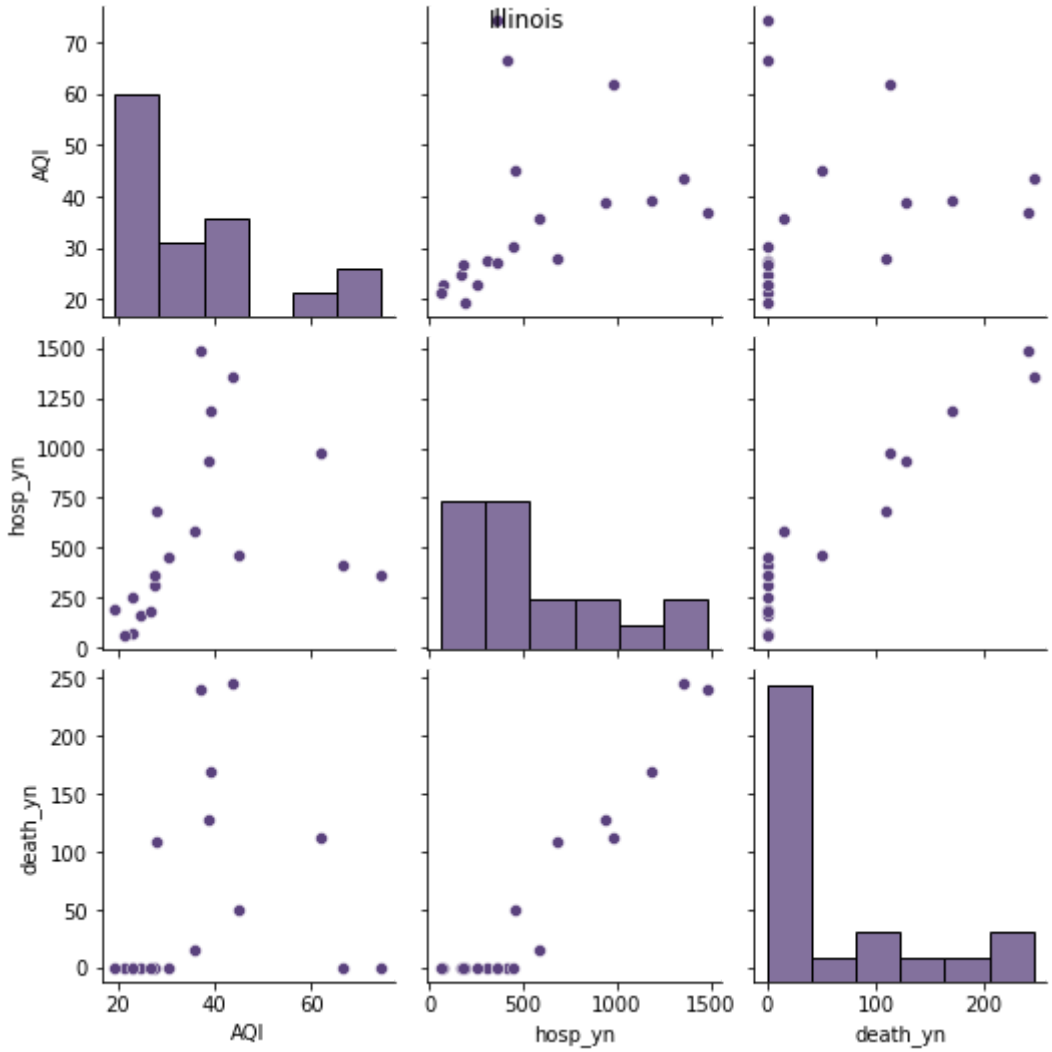

```
<Figure size 1080x504 with 0 Axes>
```



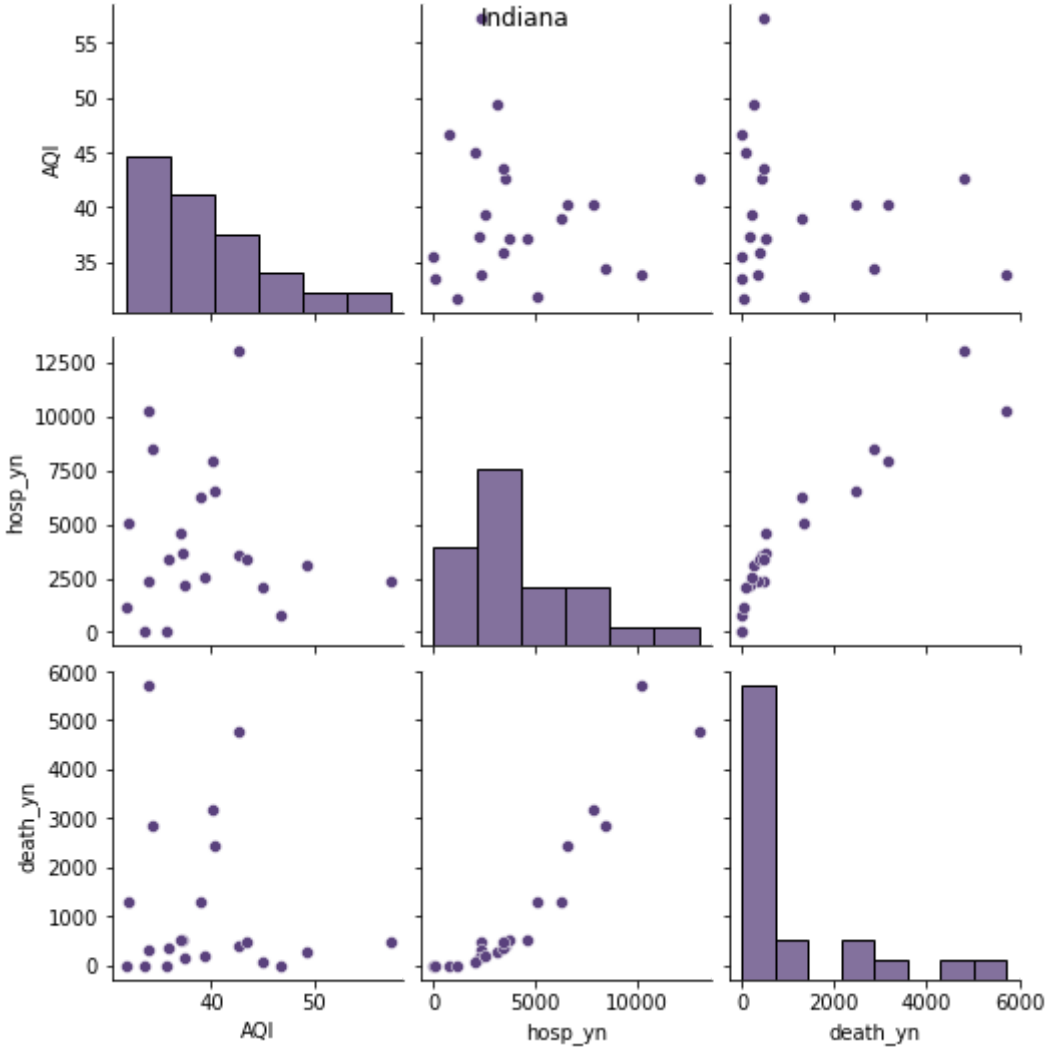

```
<Figure size 1080x504 with 0 Axes>
```



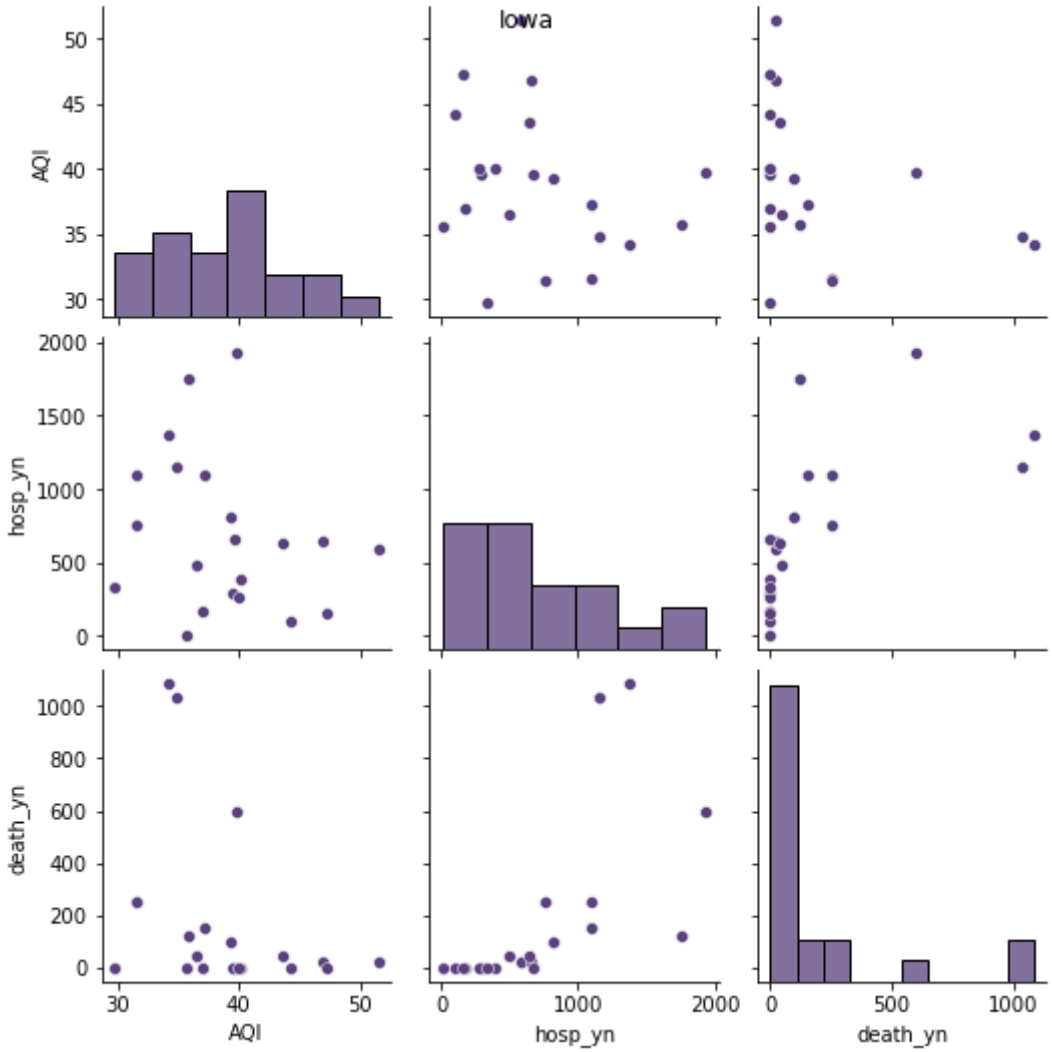
<Figure size 1080x504 with 0 Axes>



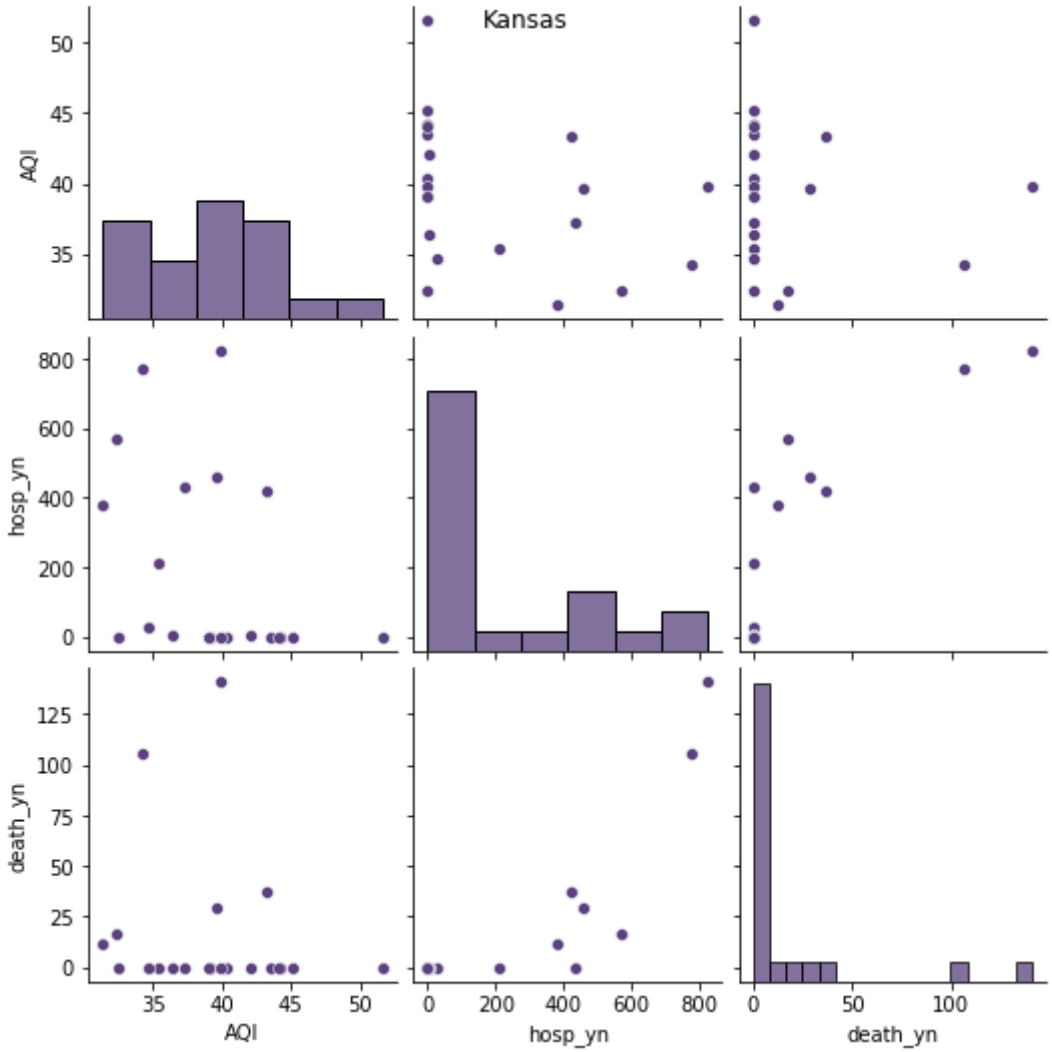

```
<Figure size 1080x504 with 0 Axes>
```



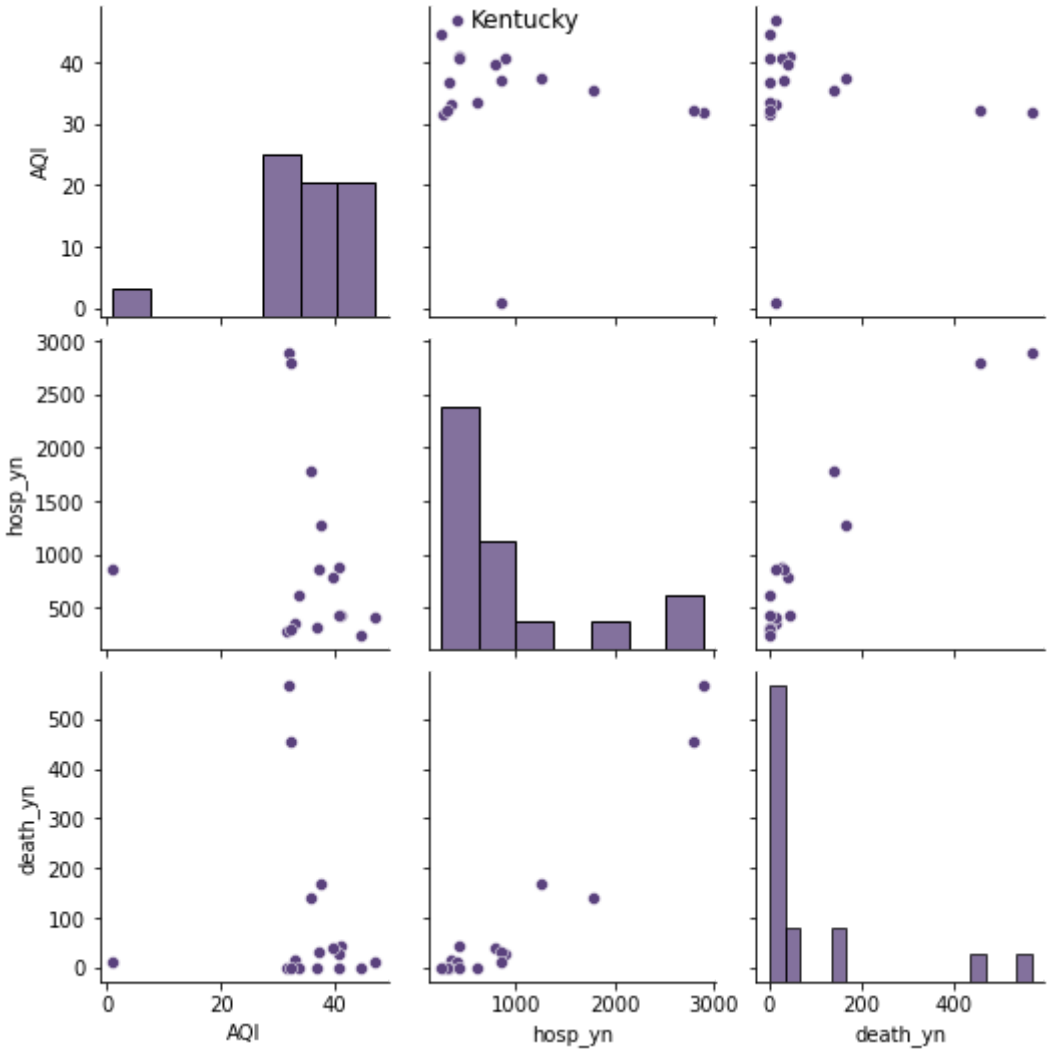

```
<Figure size 1080x504 with 0 Axes>
```



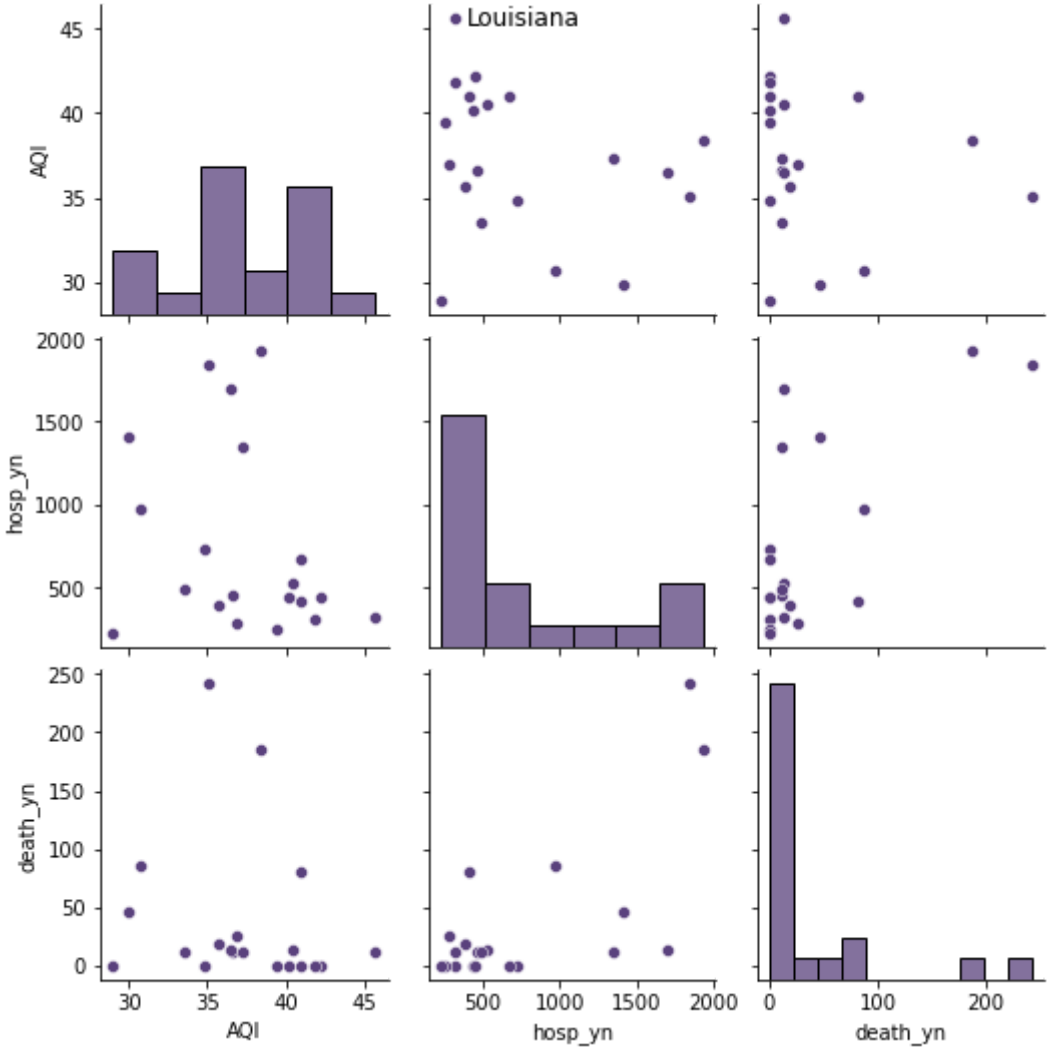

```
<Figure size 1080x504 with 0 Axes>
```



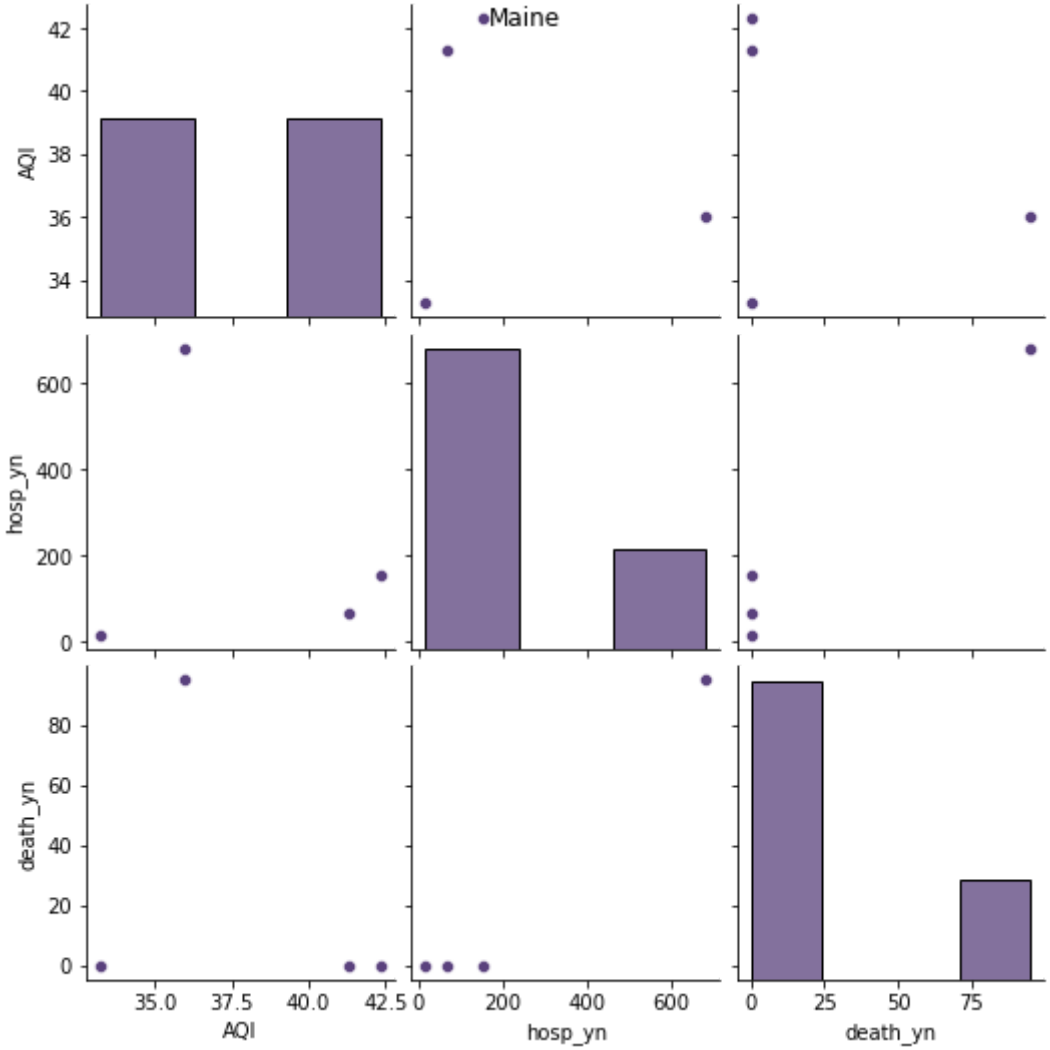
<Figure size 1080x504 with 0 Axes>



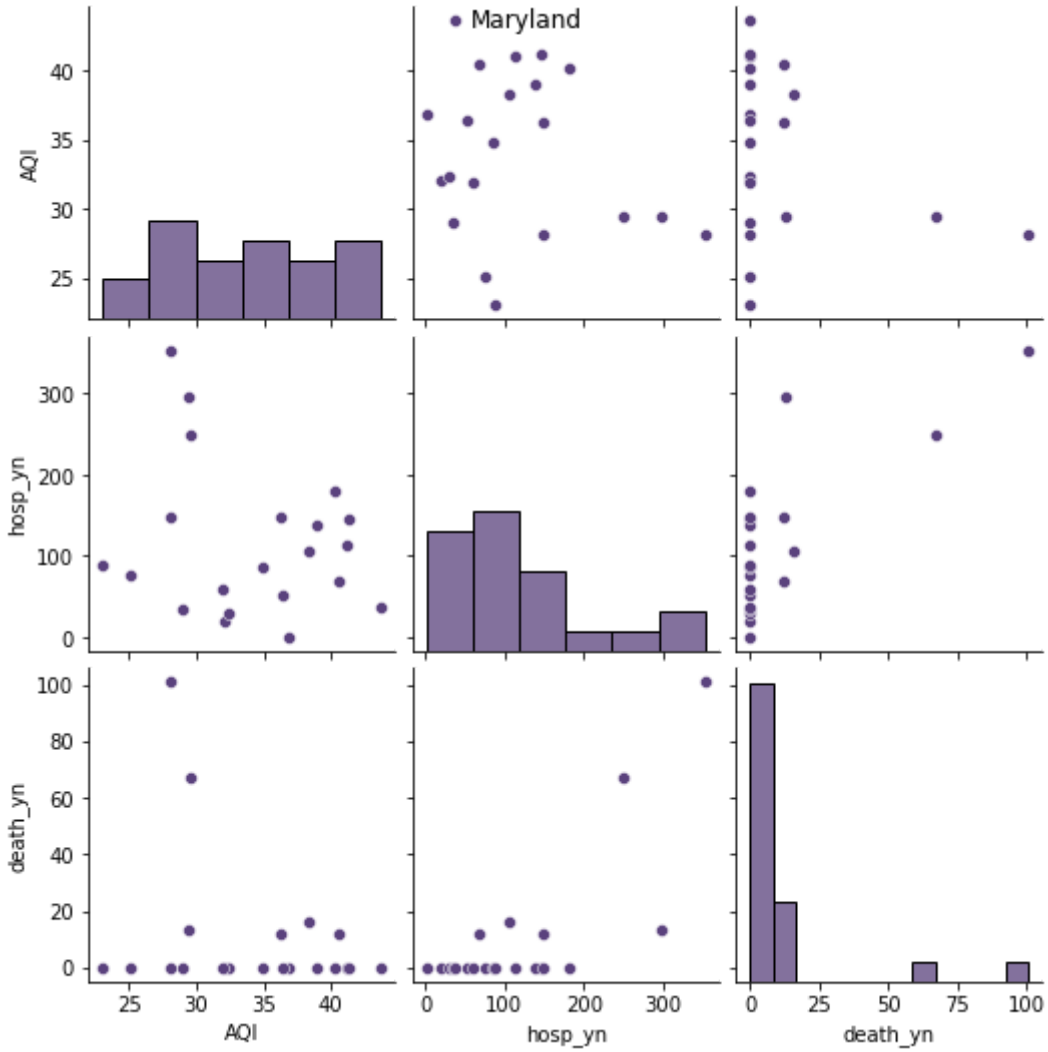

```
<Figure size 1080x504 with 0 Axes>
```



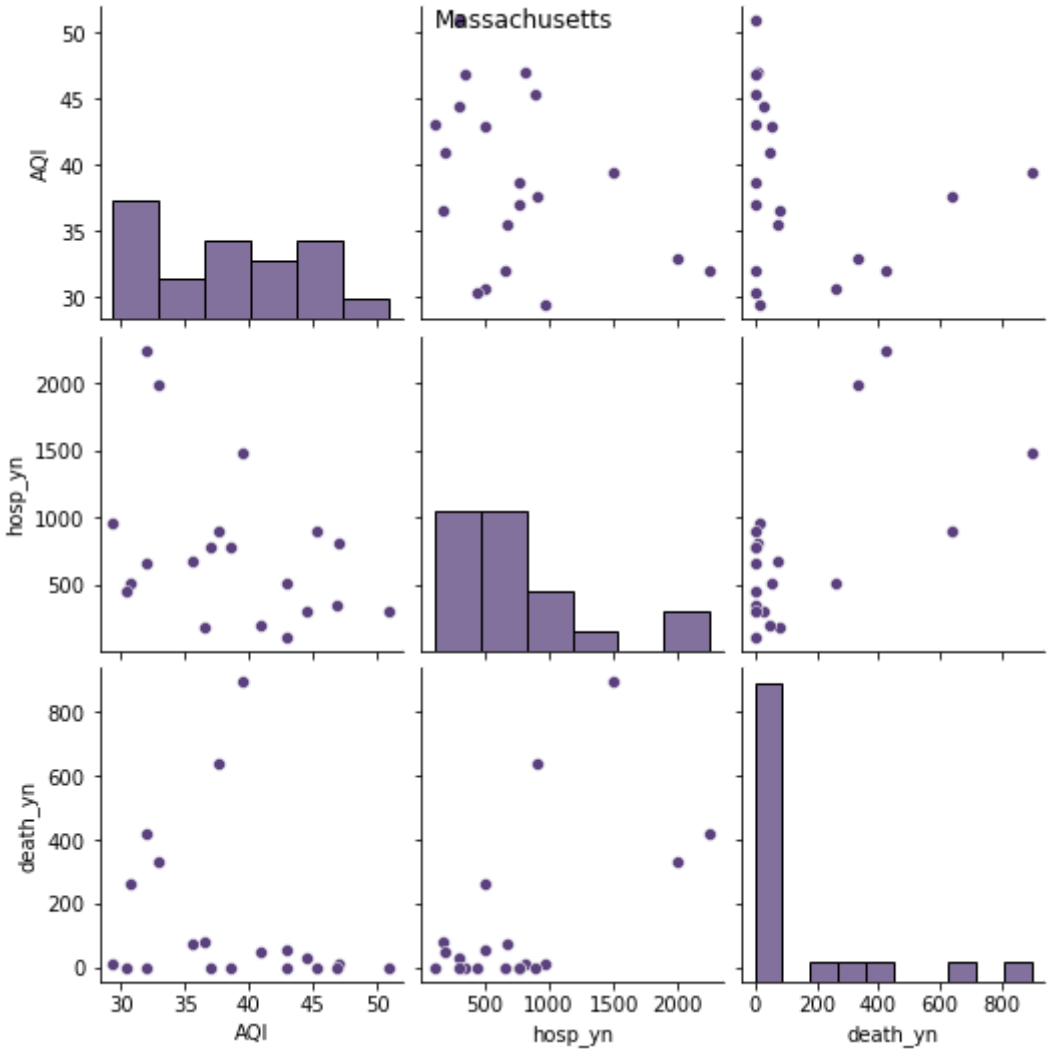

```
<Figure size 1080x504 with 0 Axes>
```



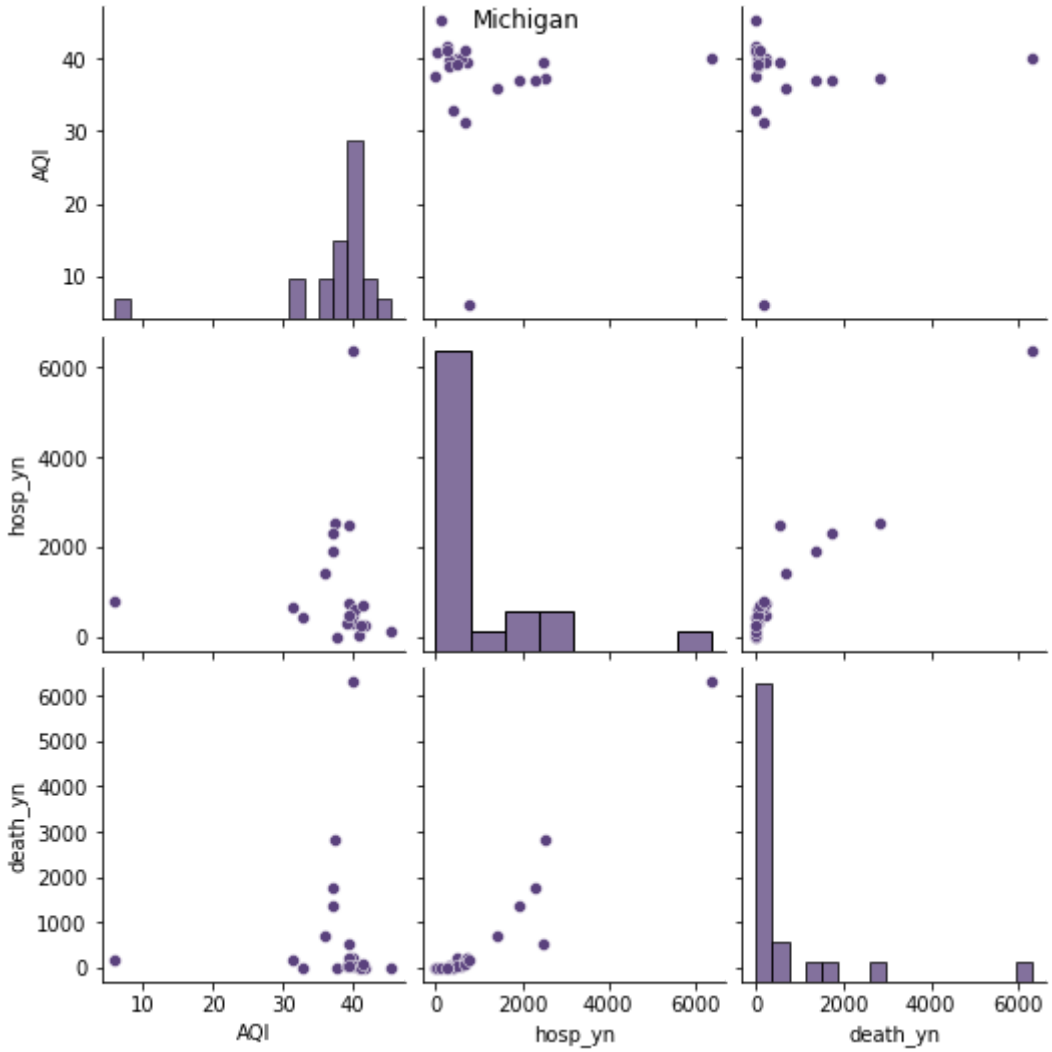

```
<Figure size 1080x504 with 0 Axes>
```



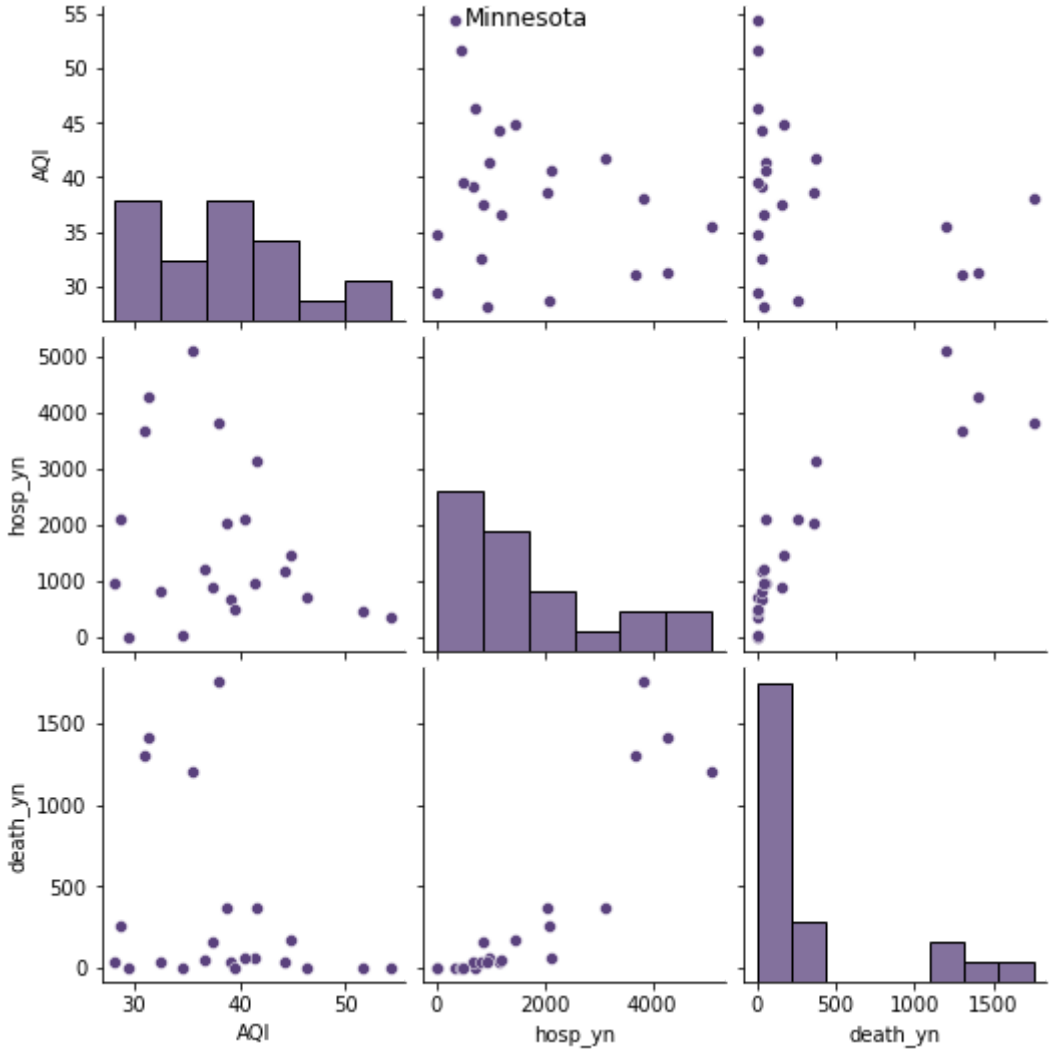

```
<Figure size 1080x504 with 0 Axes>
```



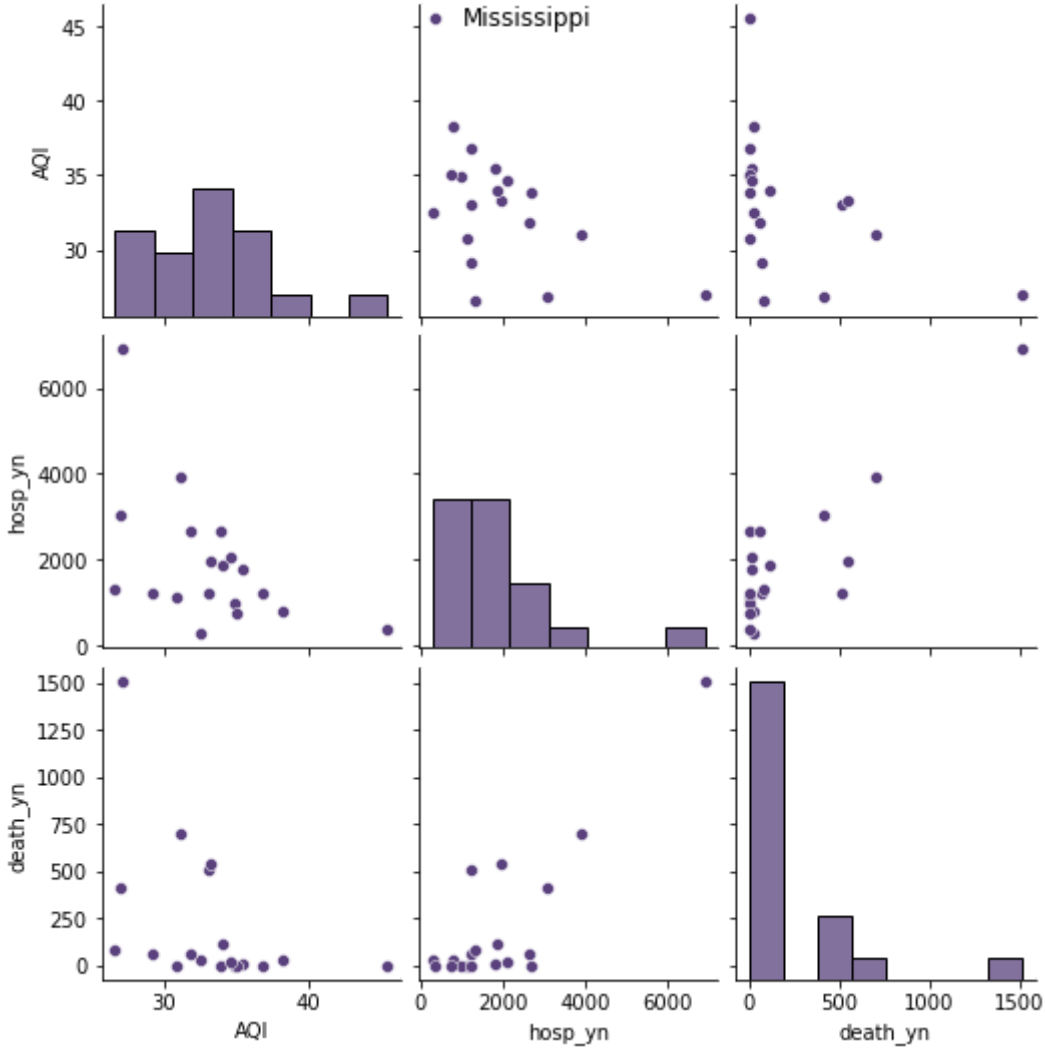

```
<Figure size 1080x504 with 0 Axes>
```



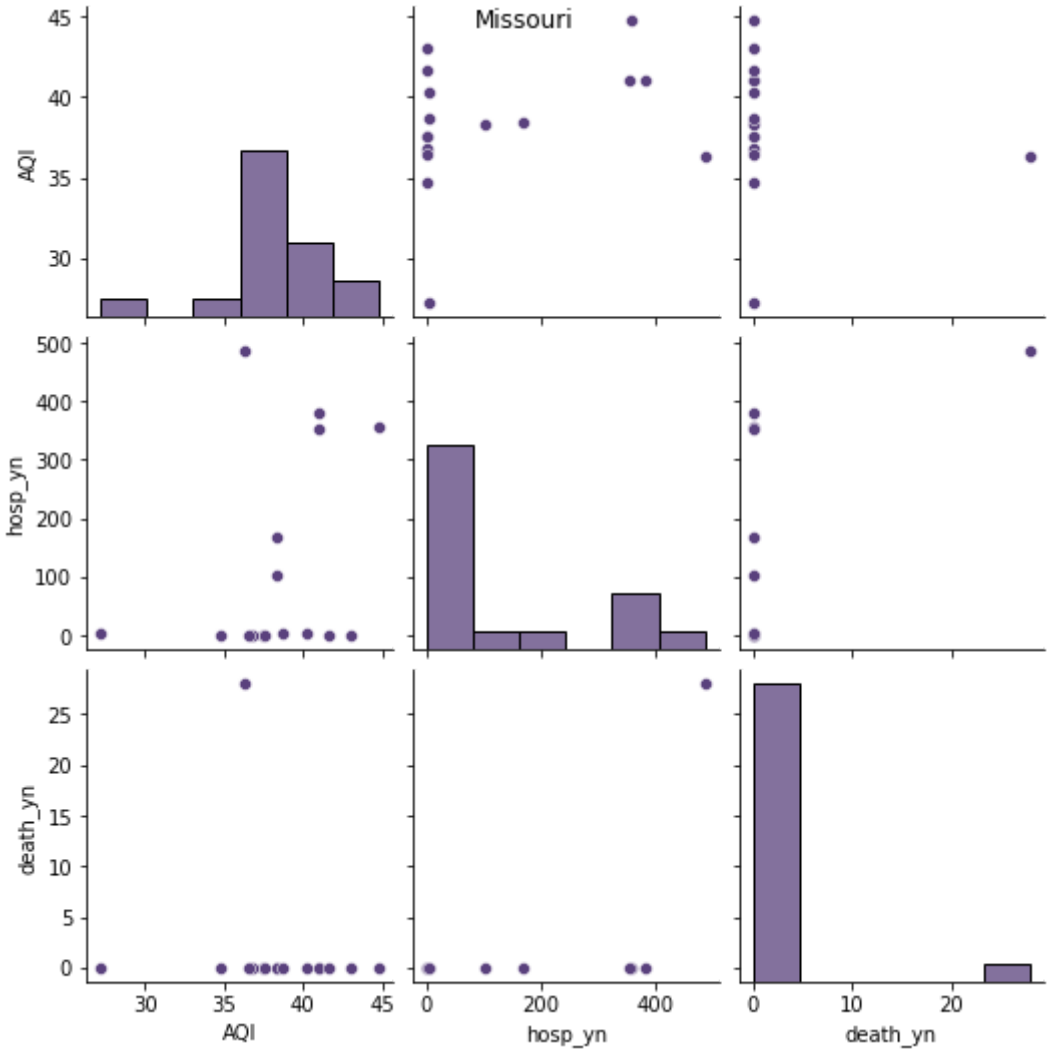

```
<Figure size 1080x504 with 0 Axes>
```



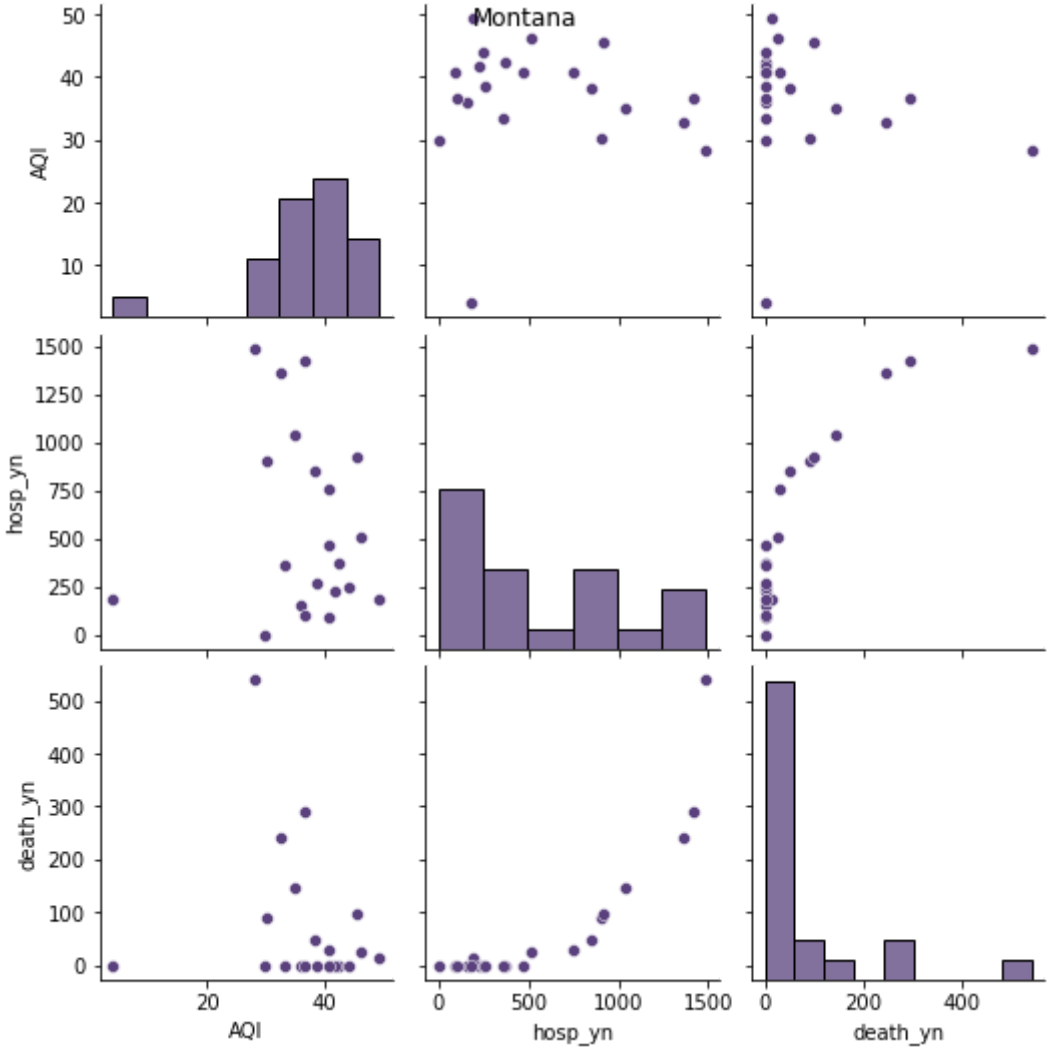

```
<Figure size 1080x504 with 0 Axes>
```



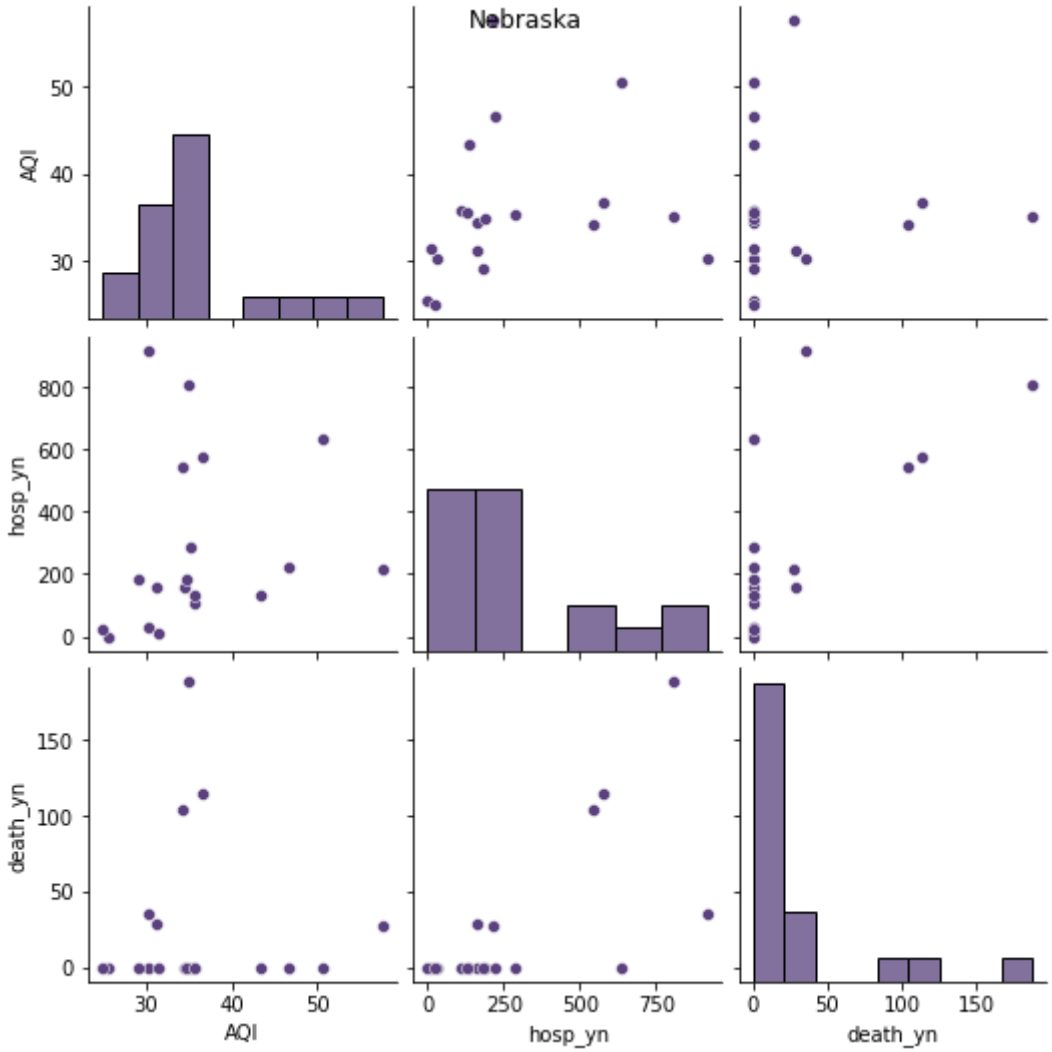

```
<Figure size 1080x504 with 0 Axes>
```



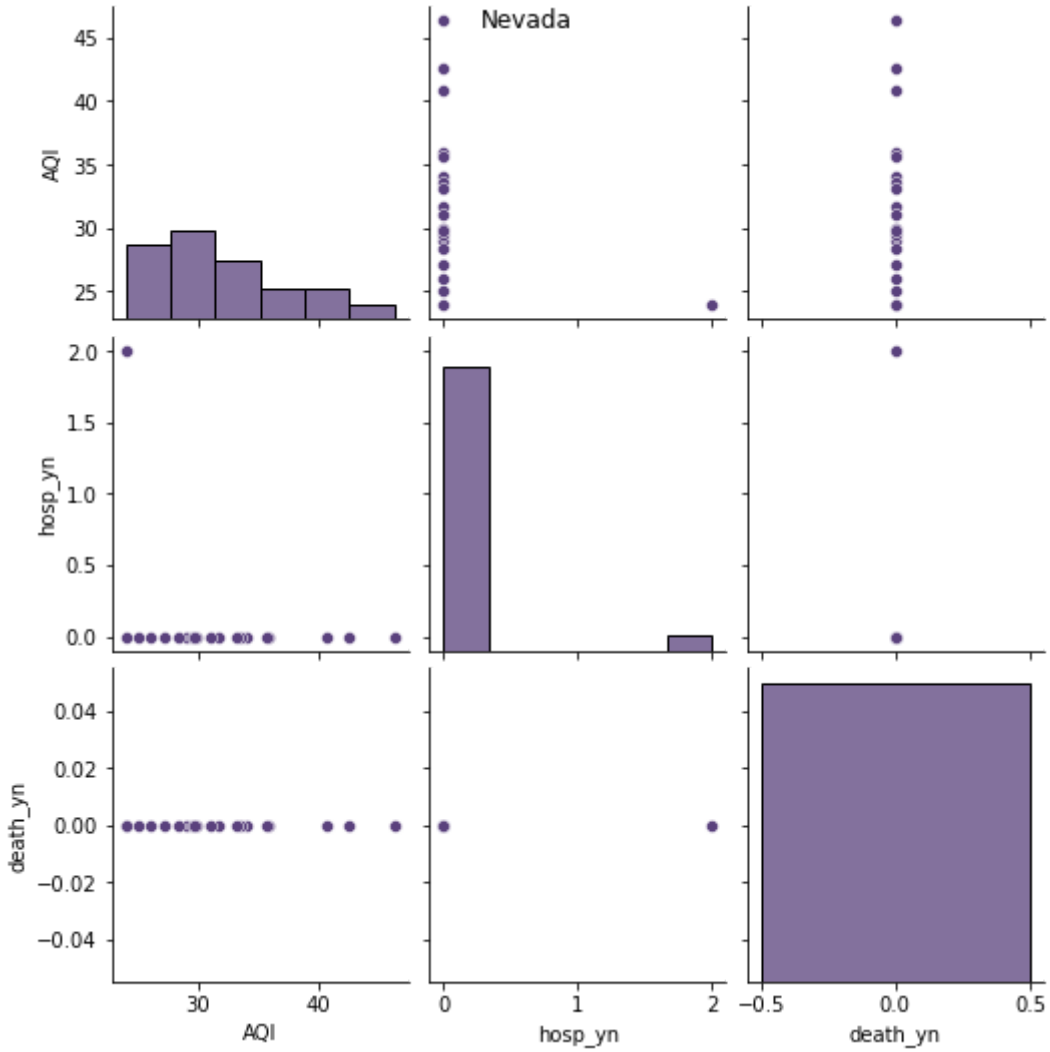
<Figure size 1080x504 with 0 Axes>



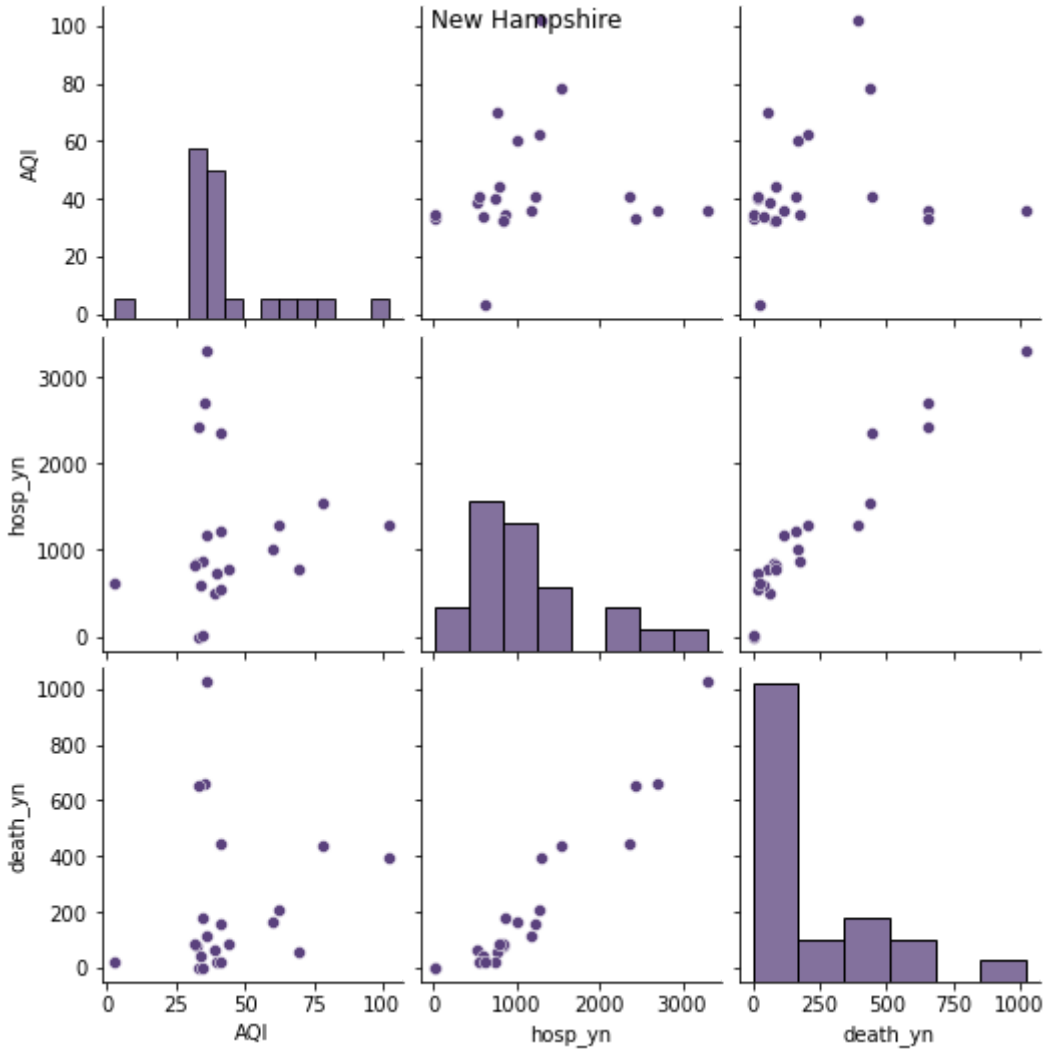

```
<Figure size 1080x504 with 0 Axes>
```



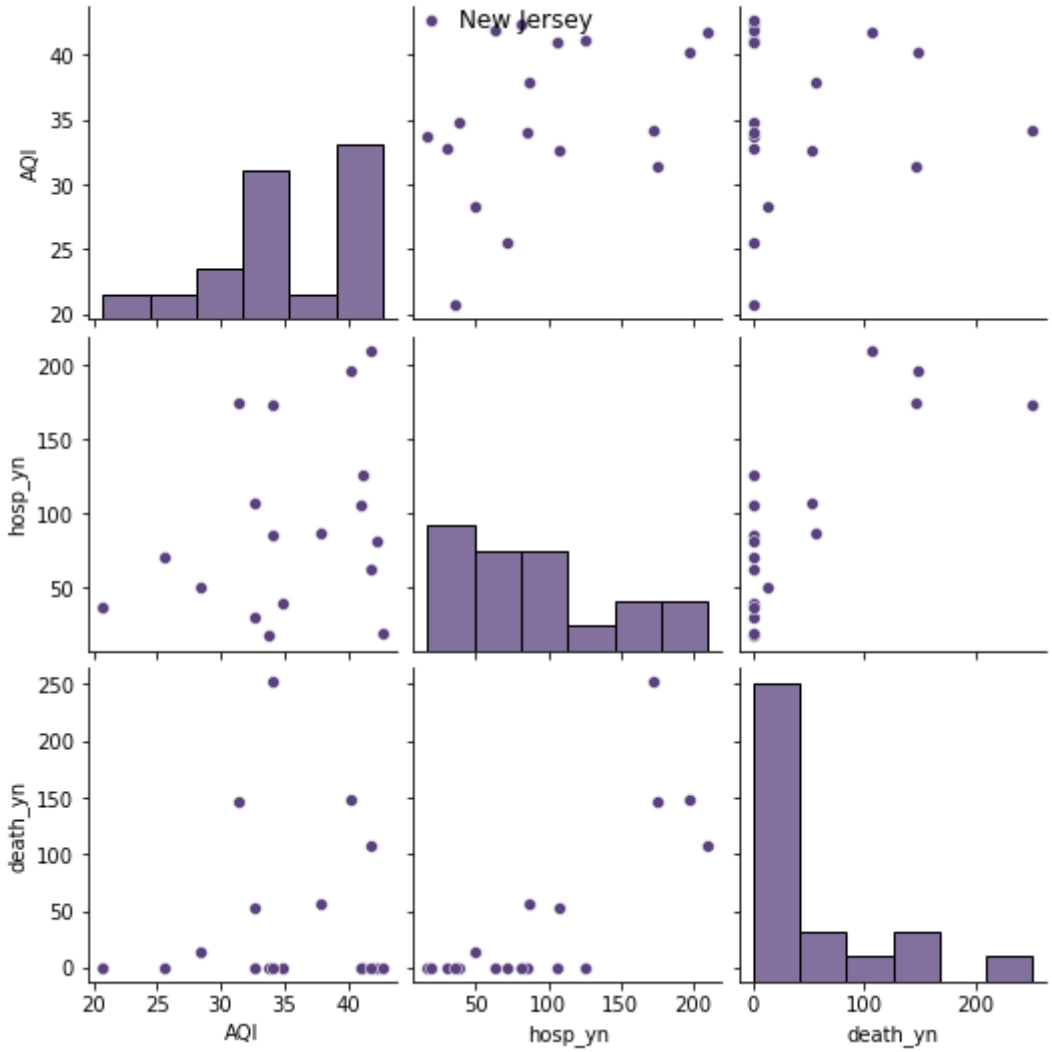
<Figure size 1080x504 with 0 Axes>



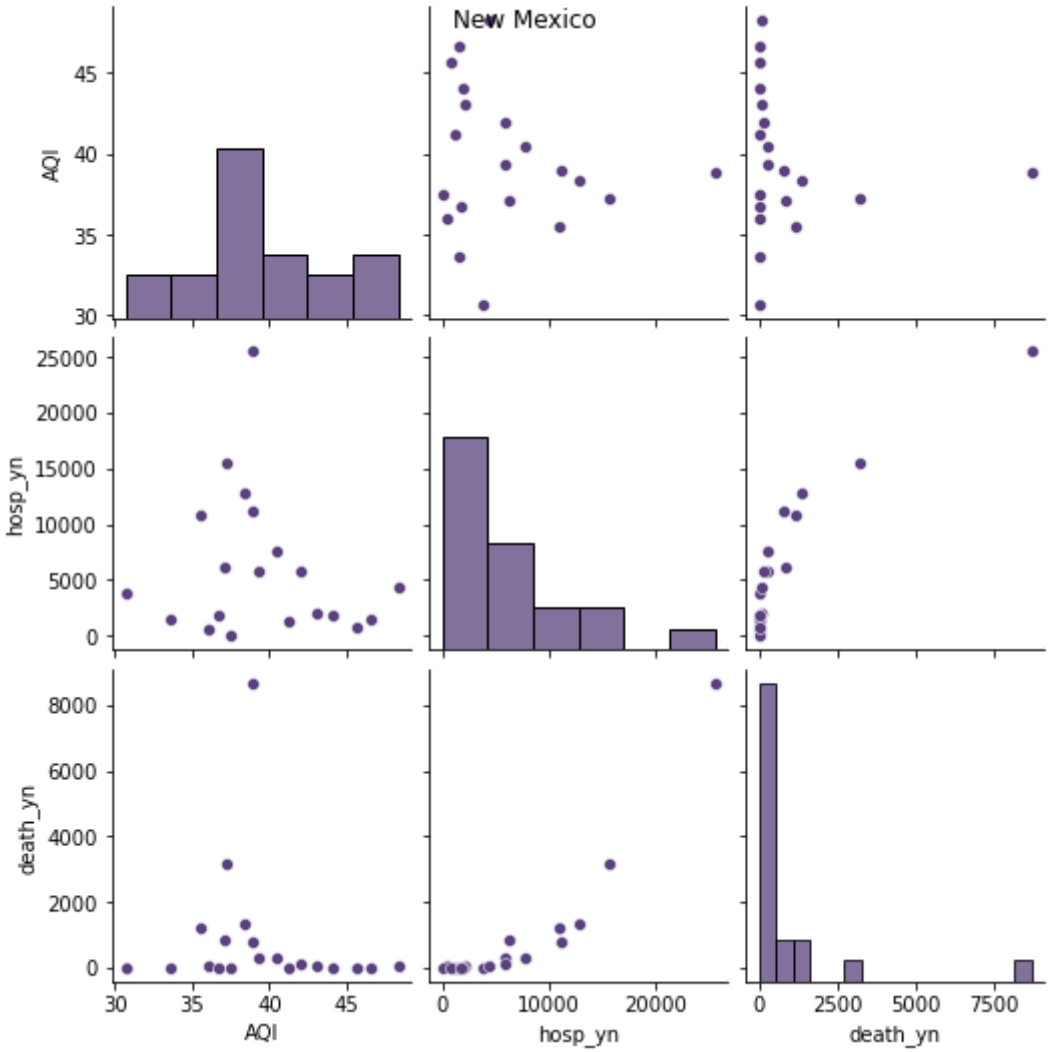

```
<Figure size 1080x504 with 0 Axes>
```



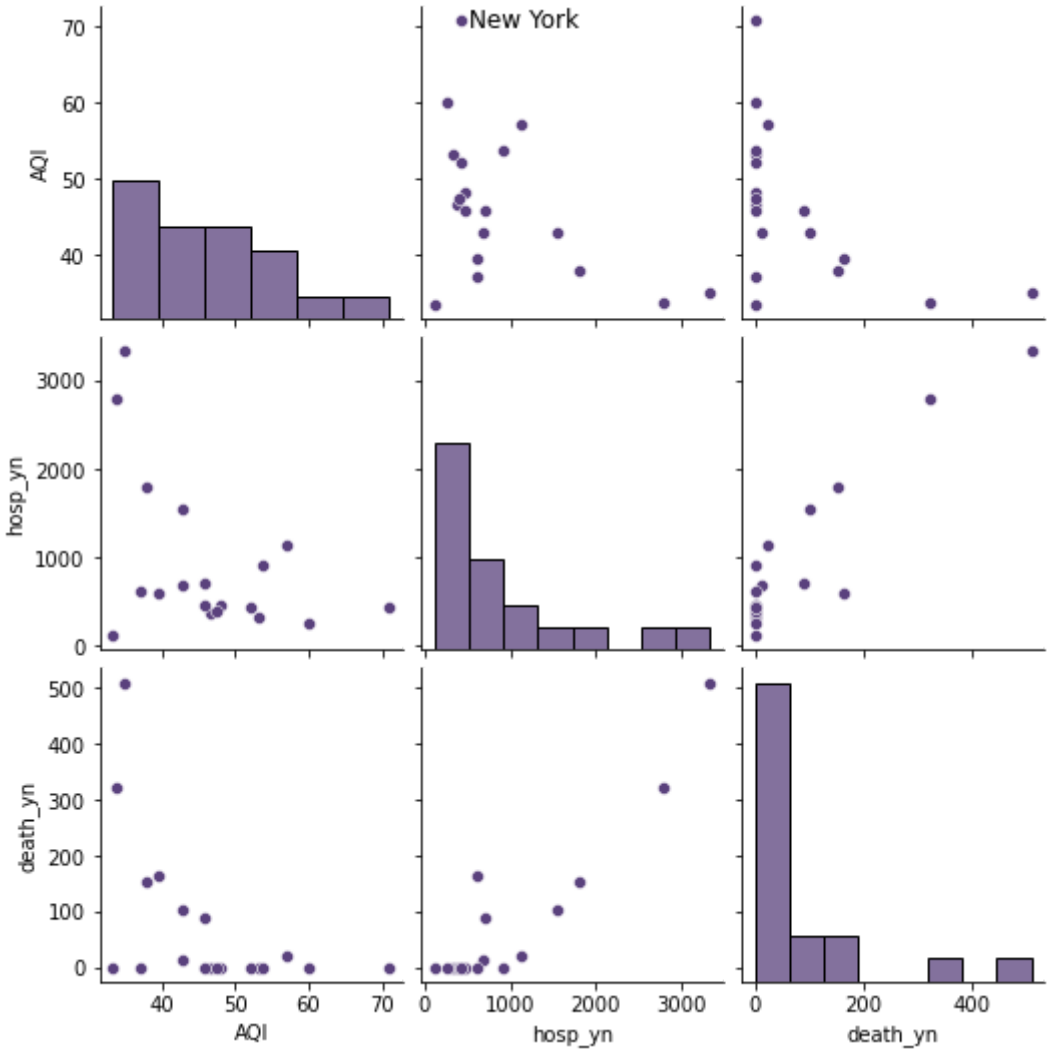

```
<Figure size 1080x504 with 0 Axes>
```



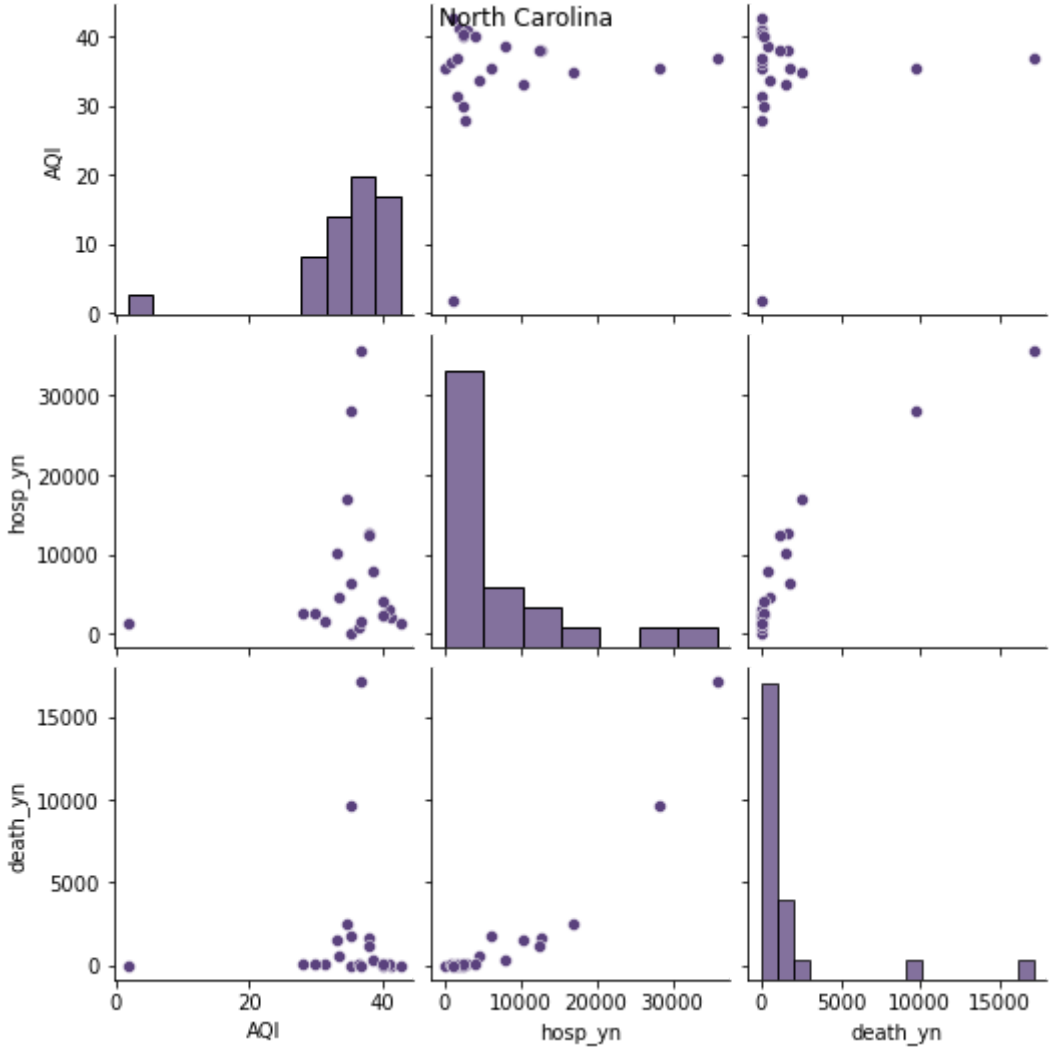

```
<Figure size 1080x504 with 0 Axes>
```



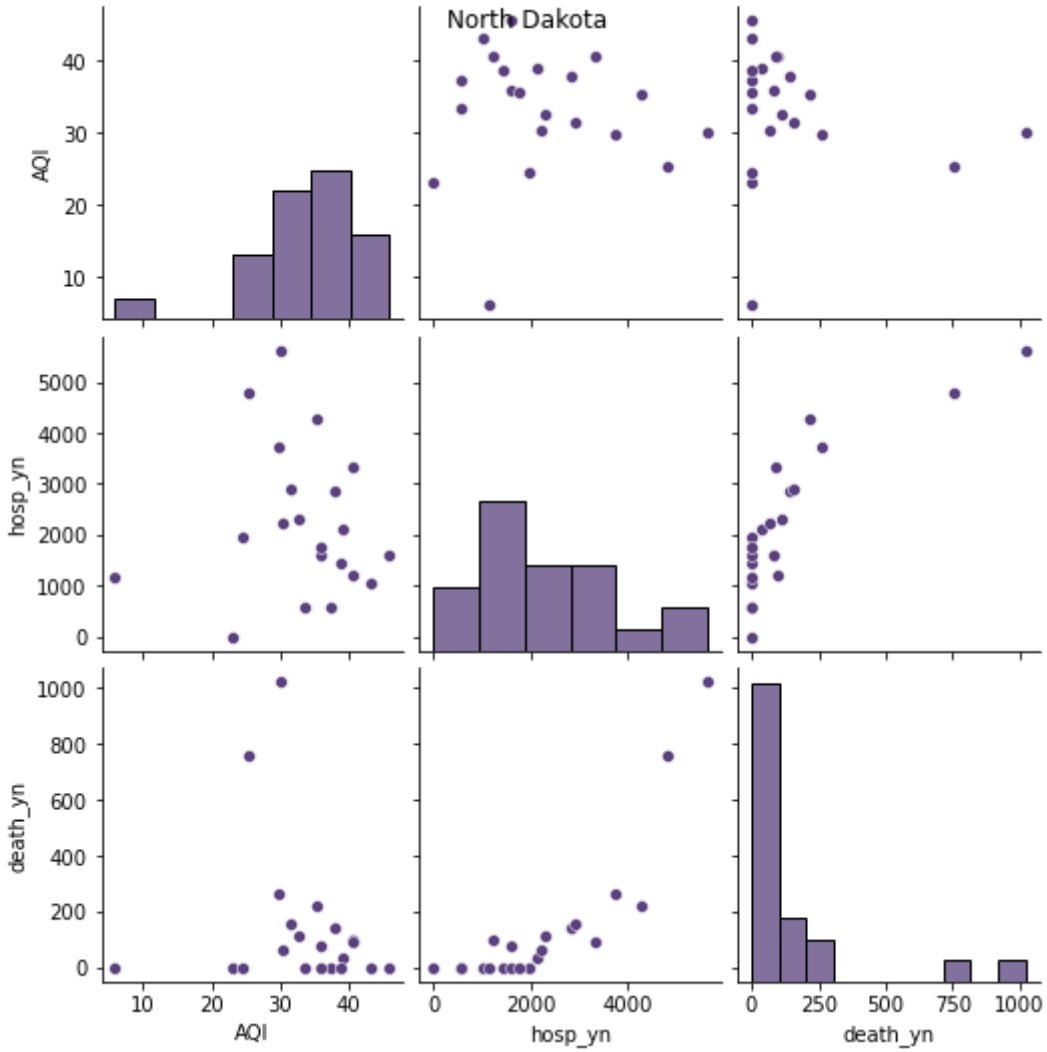

```
<Figure size 1080x504 with 0 Axes>
```



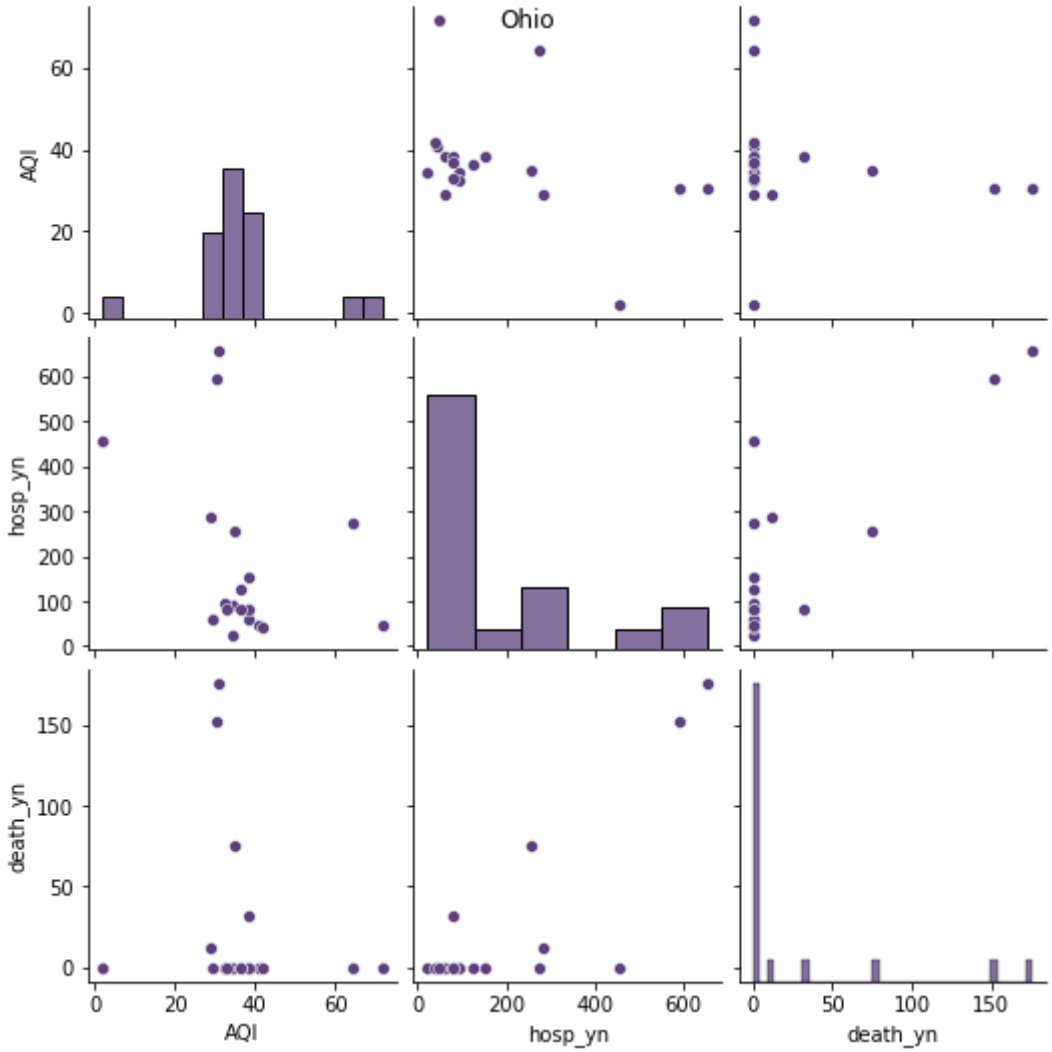

```
<Figure size 1080x504 with 0 Axes>
```



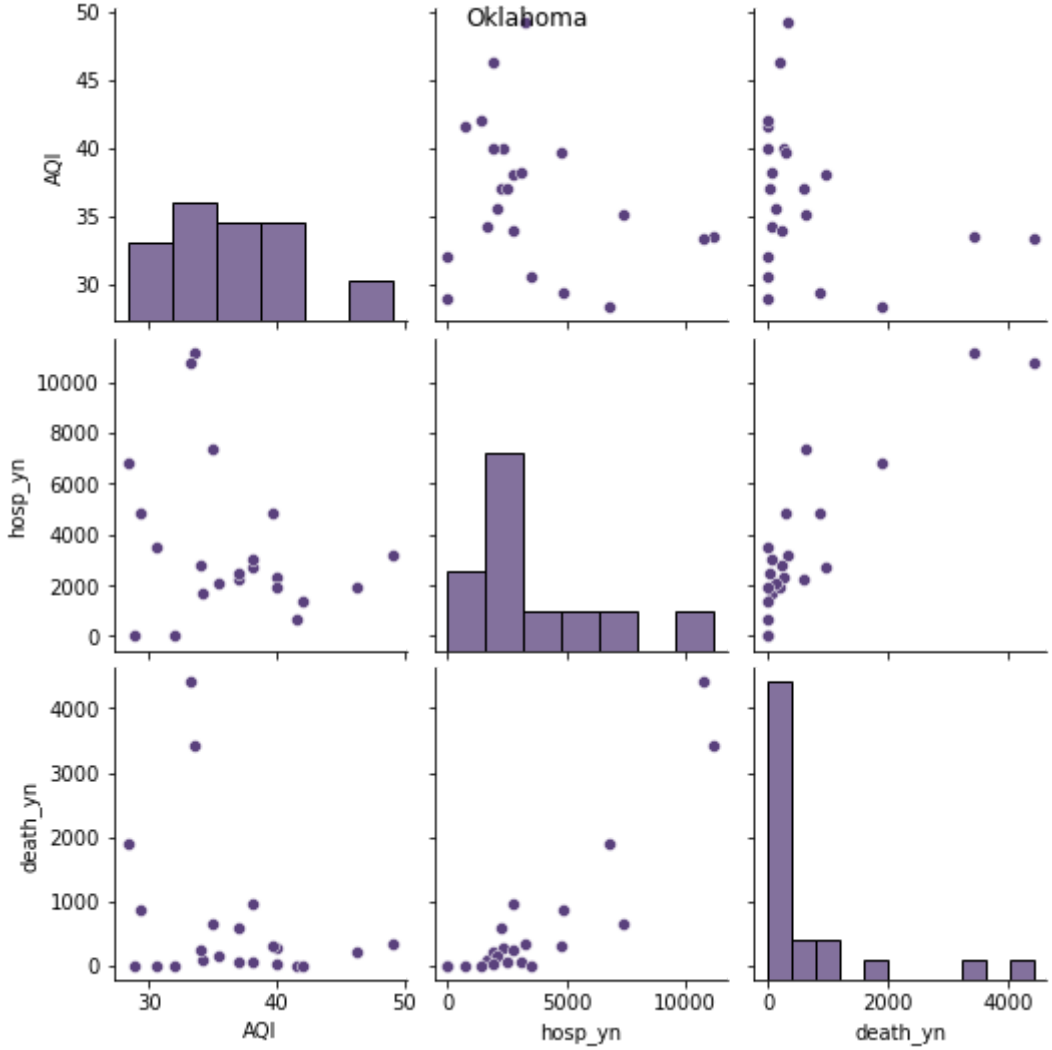

```
<Figure size 1080x504 with 0 Axes>
```



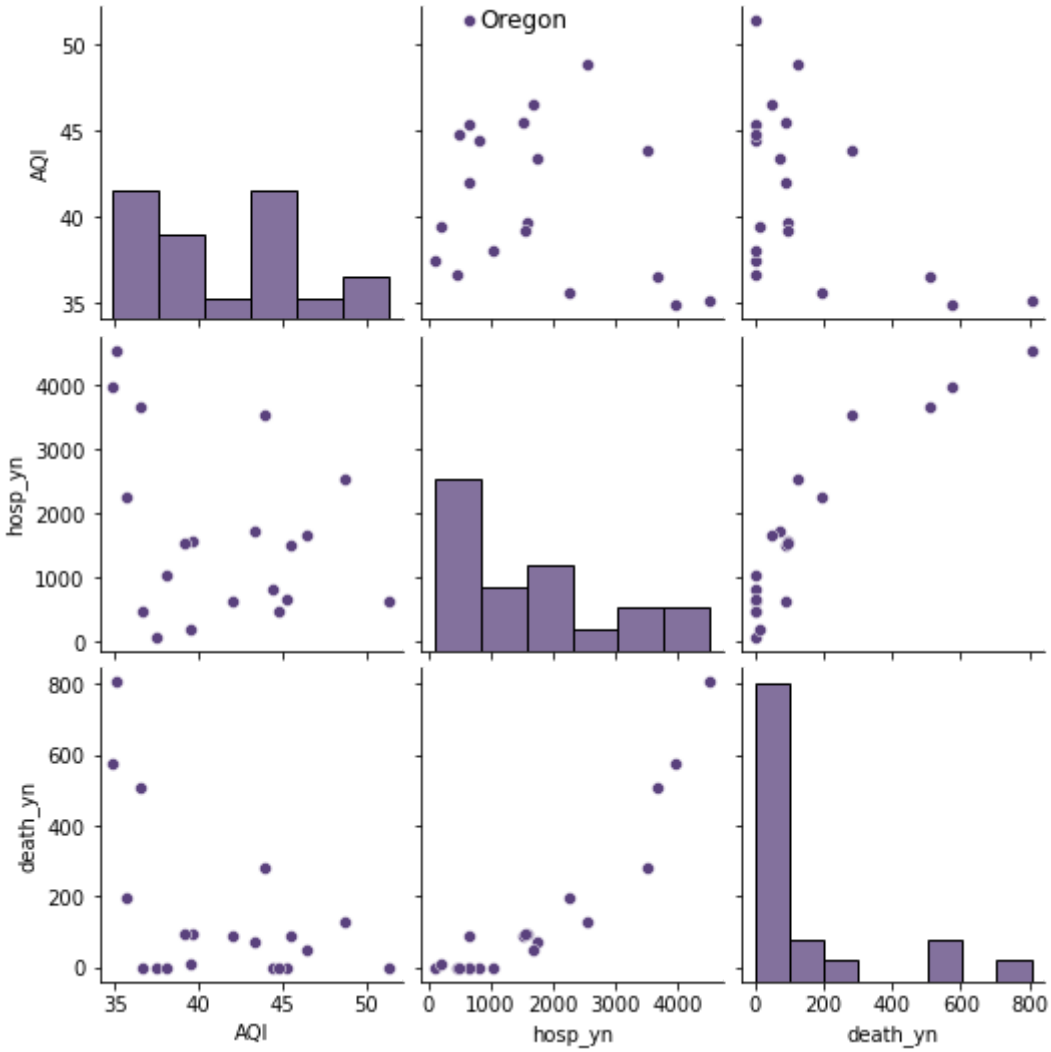

```
<Figure size 1080x504 with 0 Axes>
```



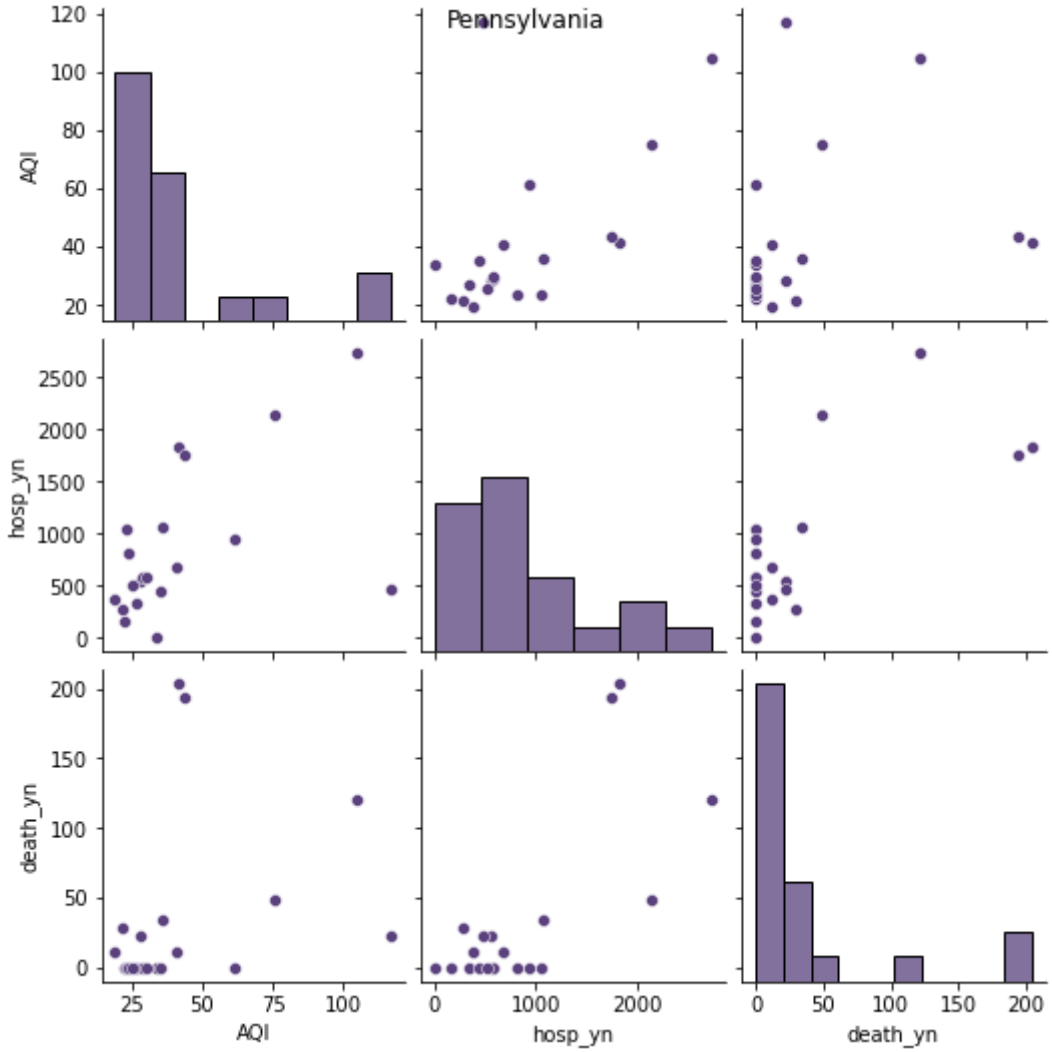
<Figure size 1080x504 with 0 Axes>



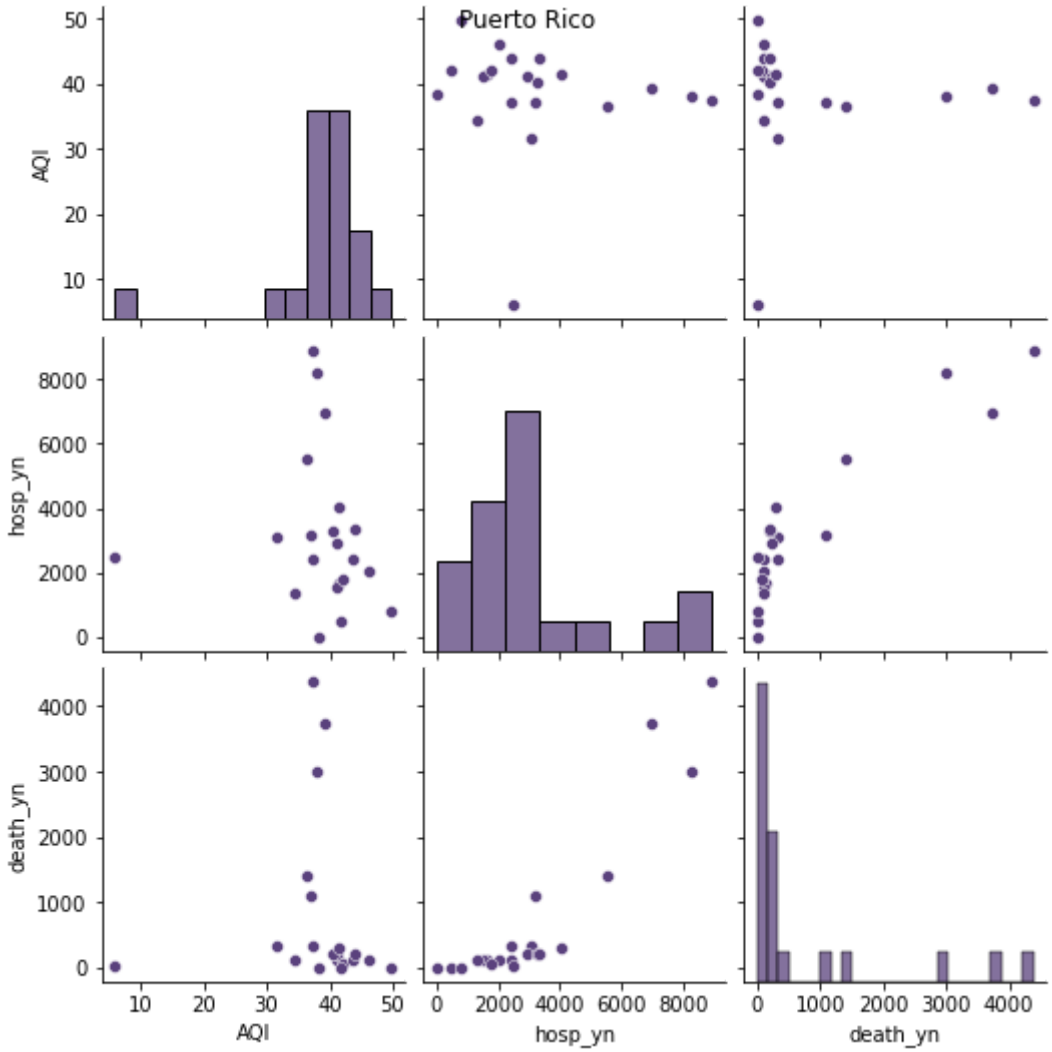

```
<Figure size 1080x504 with 0 Axes>
```



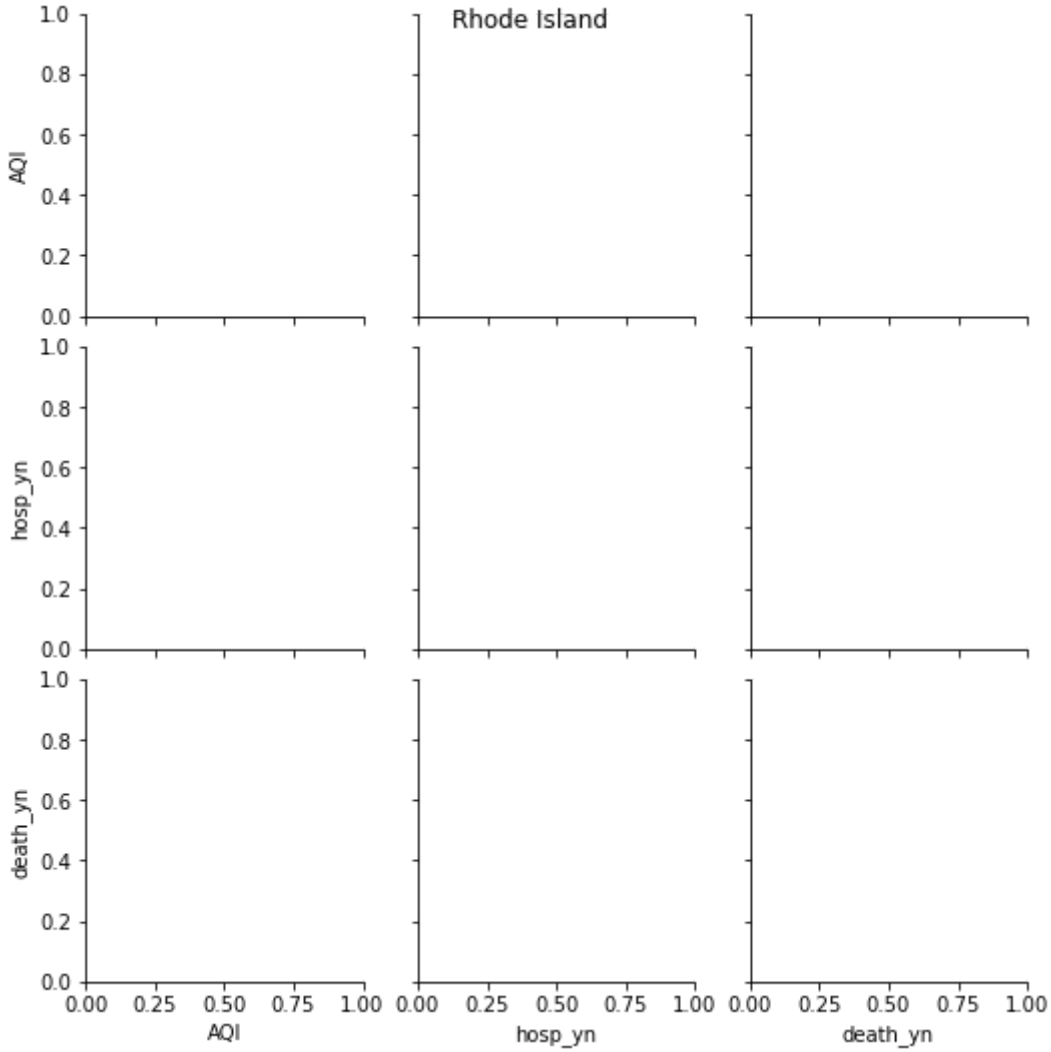

```
<Figure size 1080x504 with 0 Axes>
```



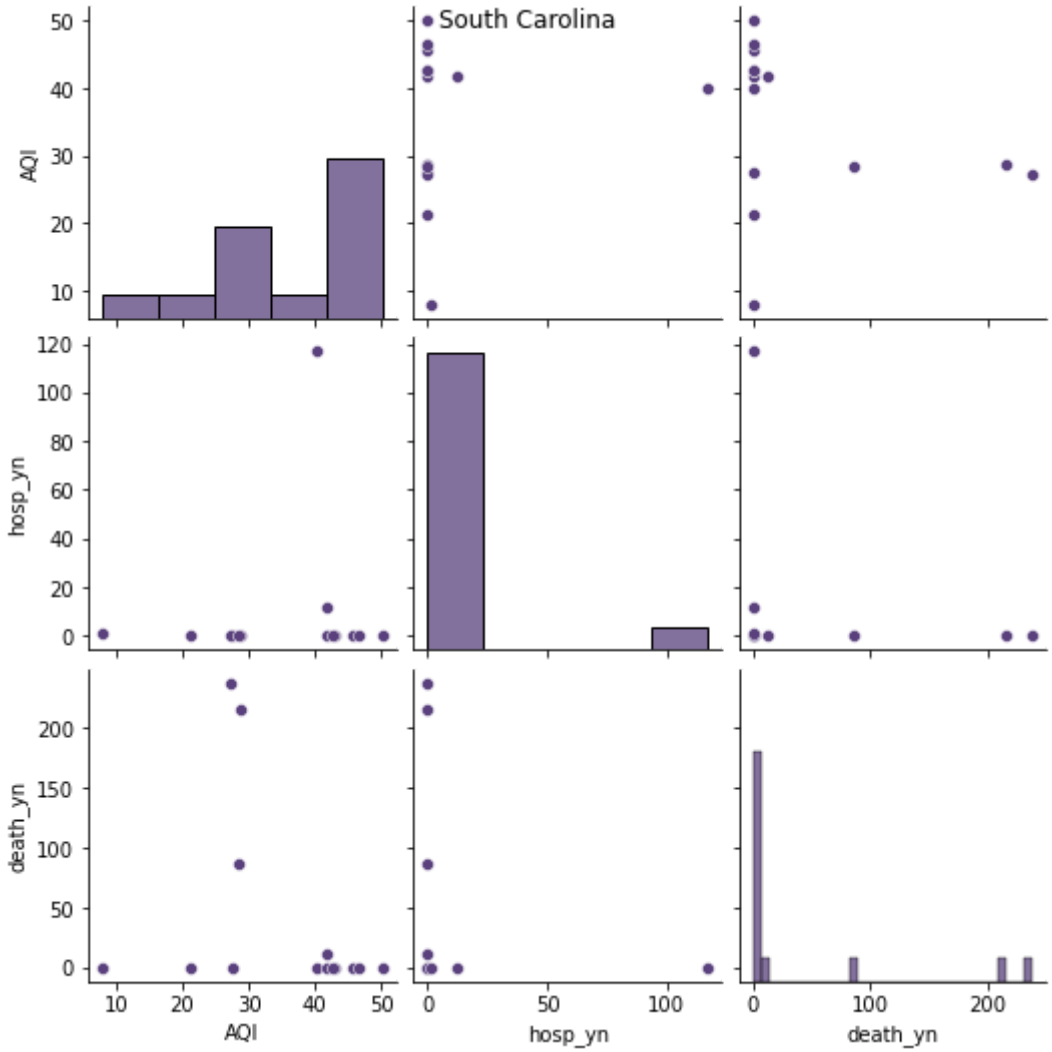

```
<Figure size 1080x504 with 0 Axes>
```



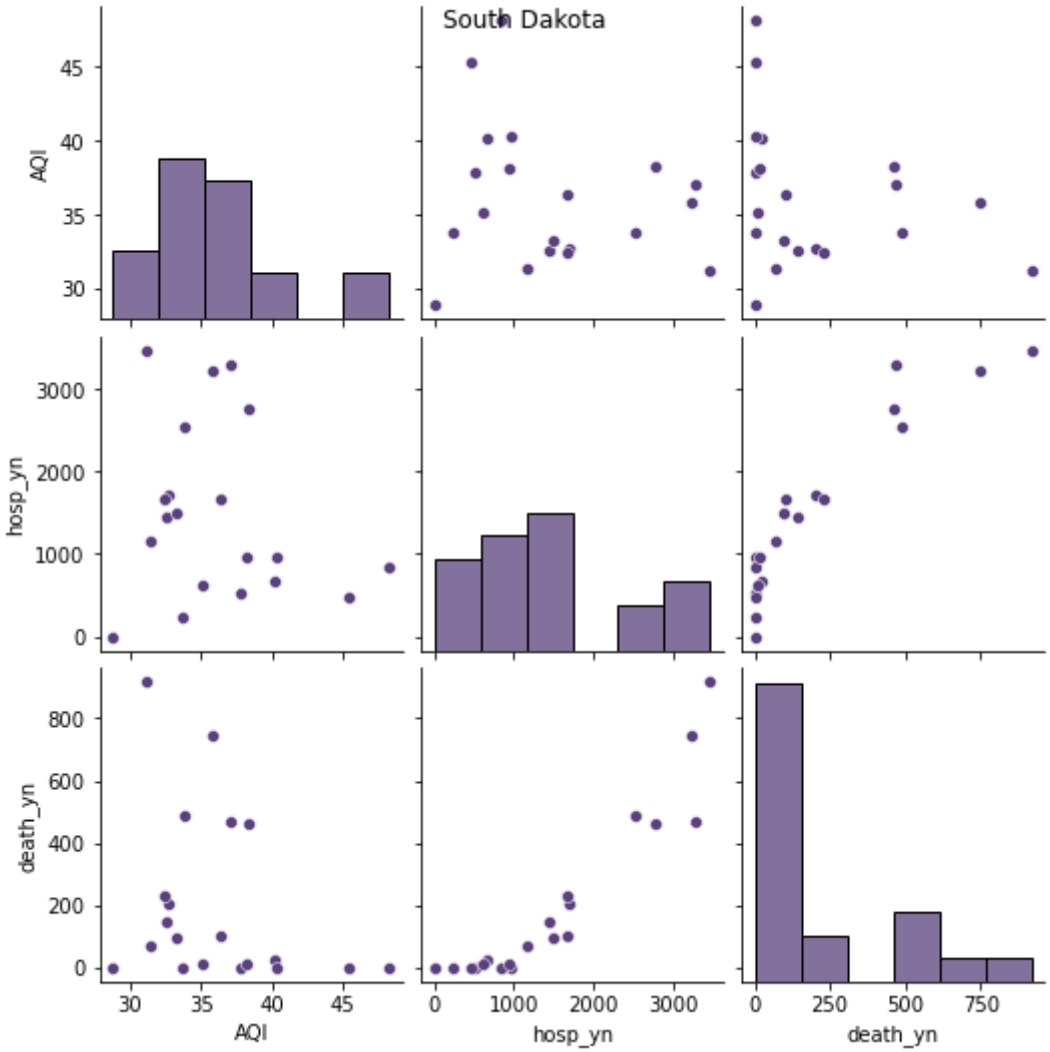

```
<Figure size 1080x504 with 0 Axes>
```



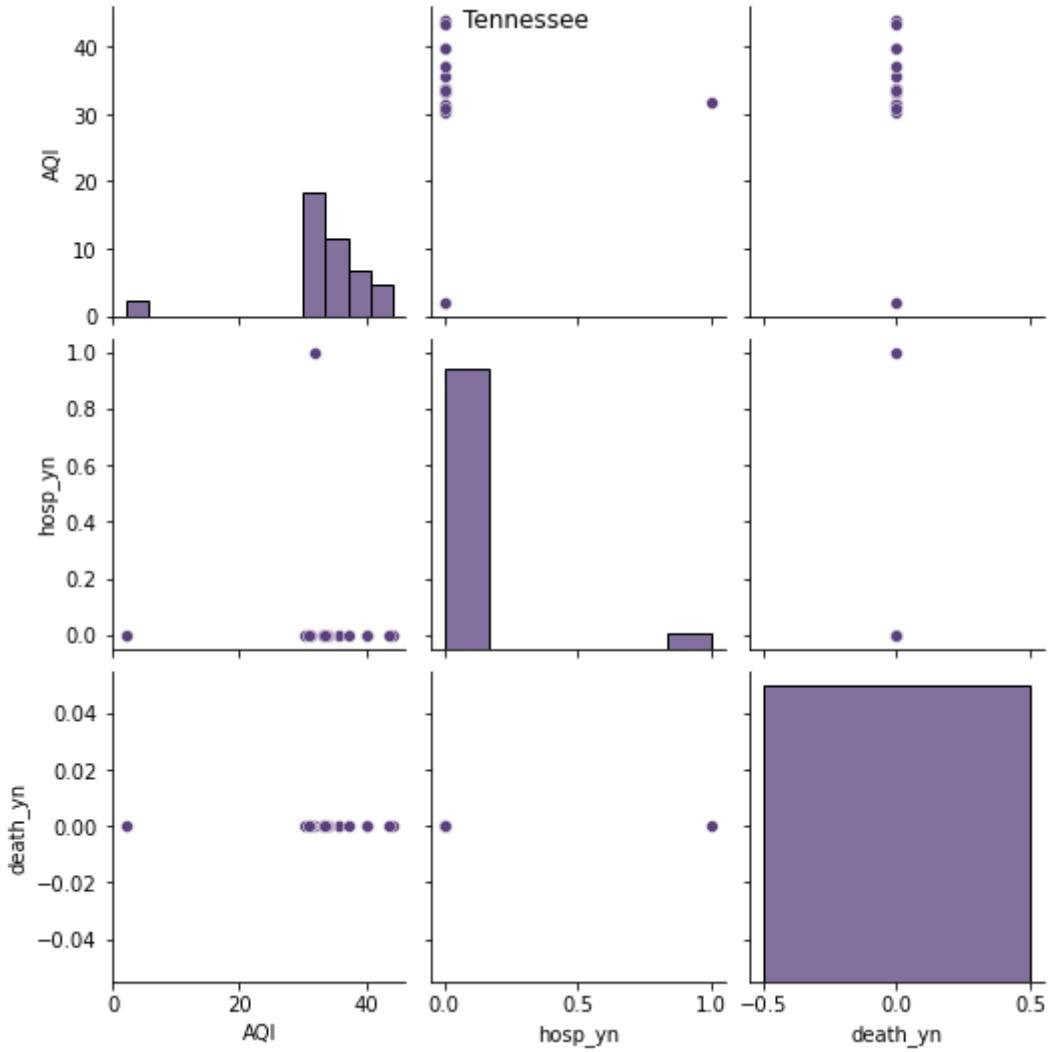
<Figure size 1080x504 with 0 Axes>



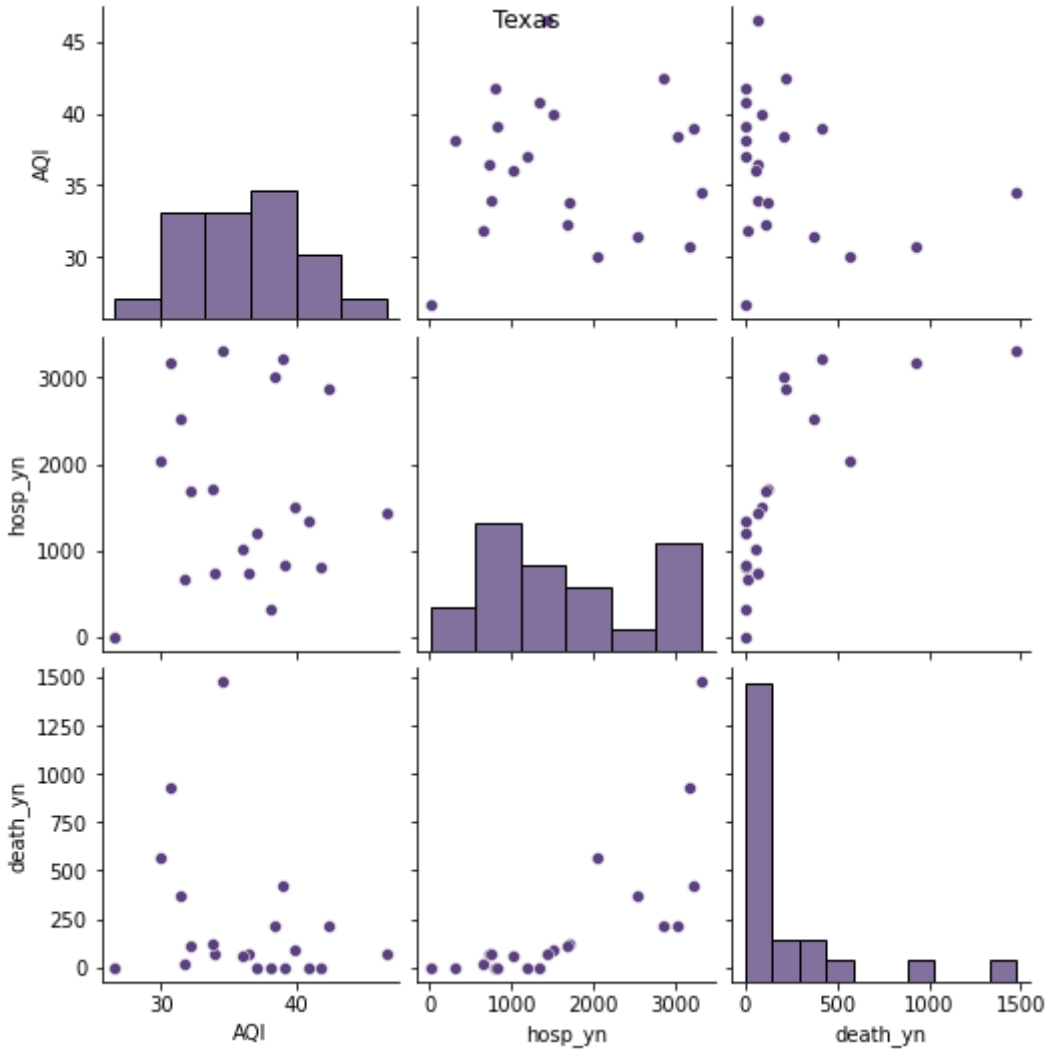
<Figure size 1080x504 with 0 Axes>



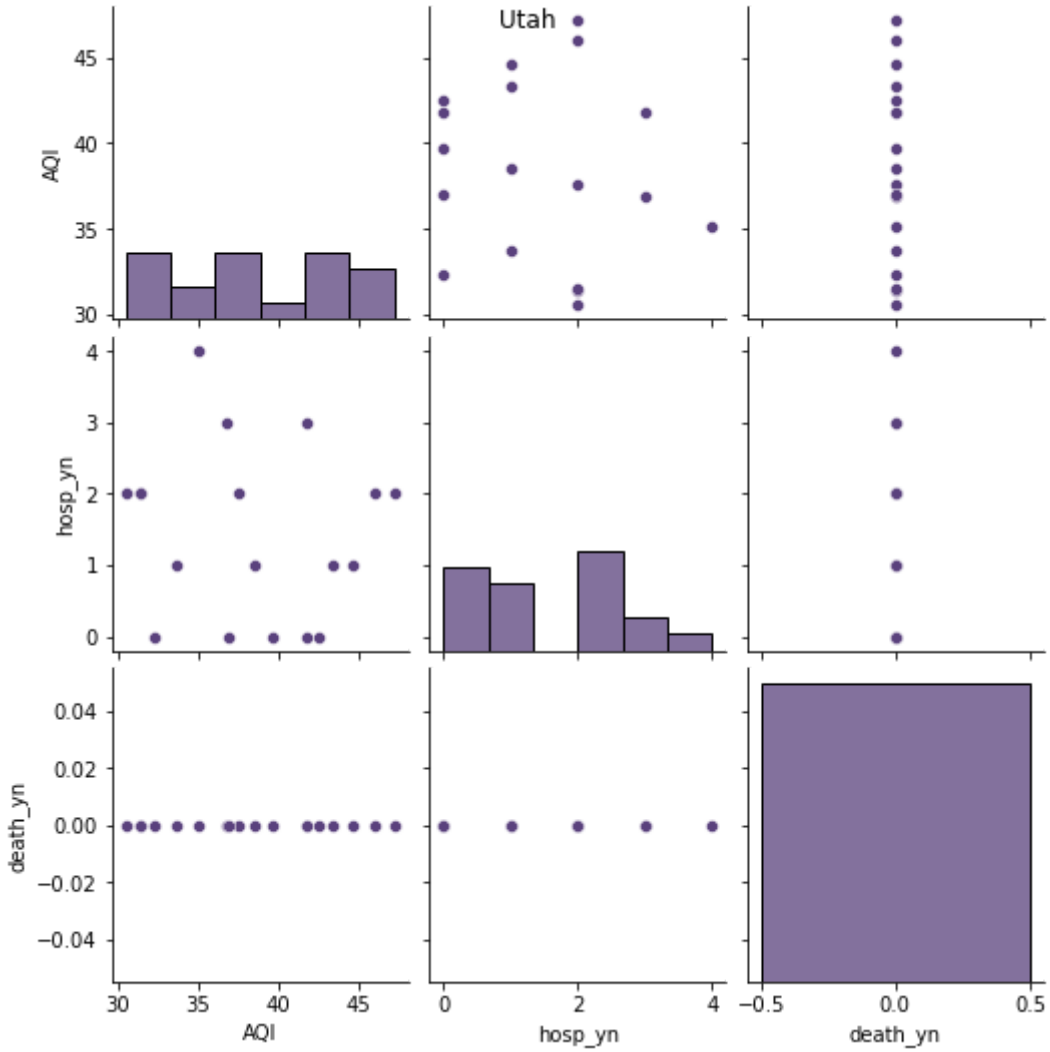

```
<Figure size 1080x504 with 0 Axes>
```



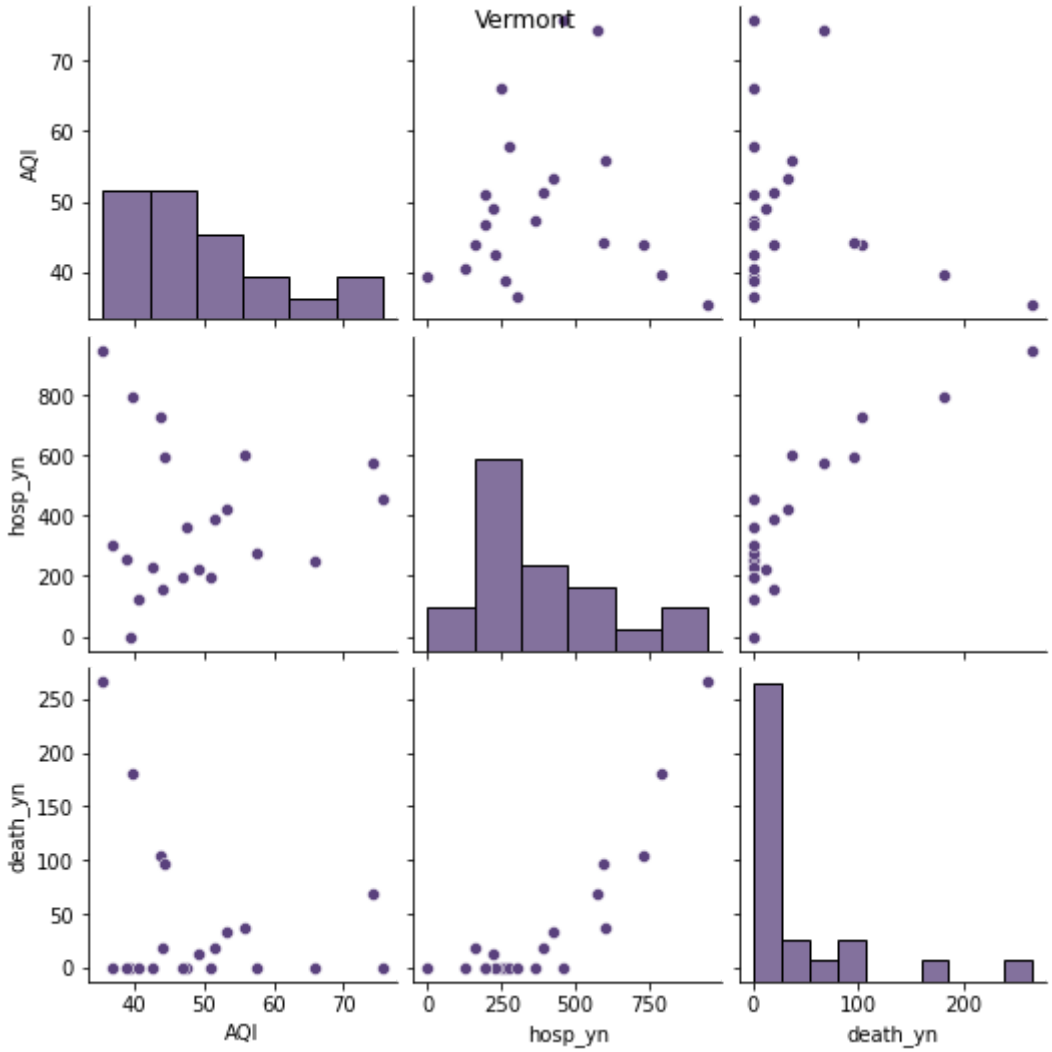

```
<Figure size 1080x504 with 0 Axes>
```



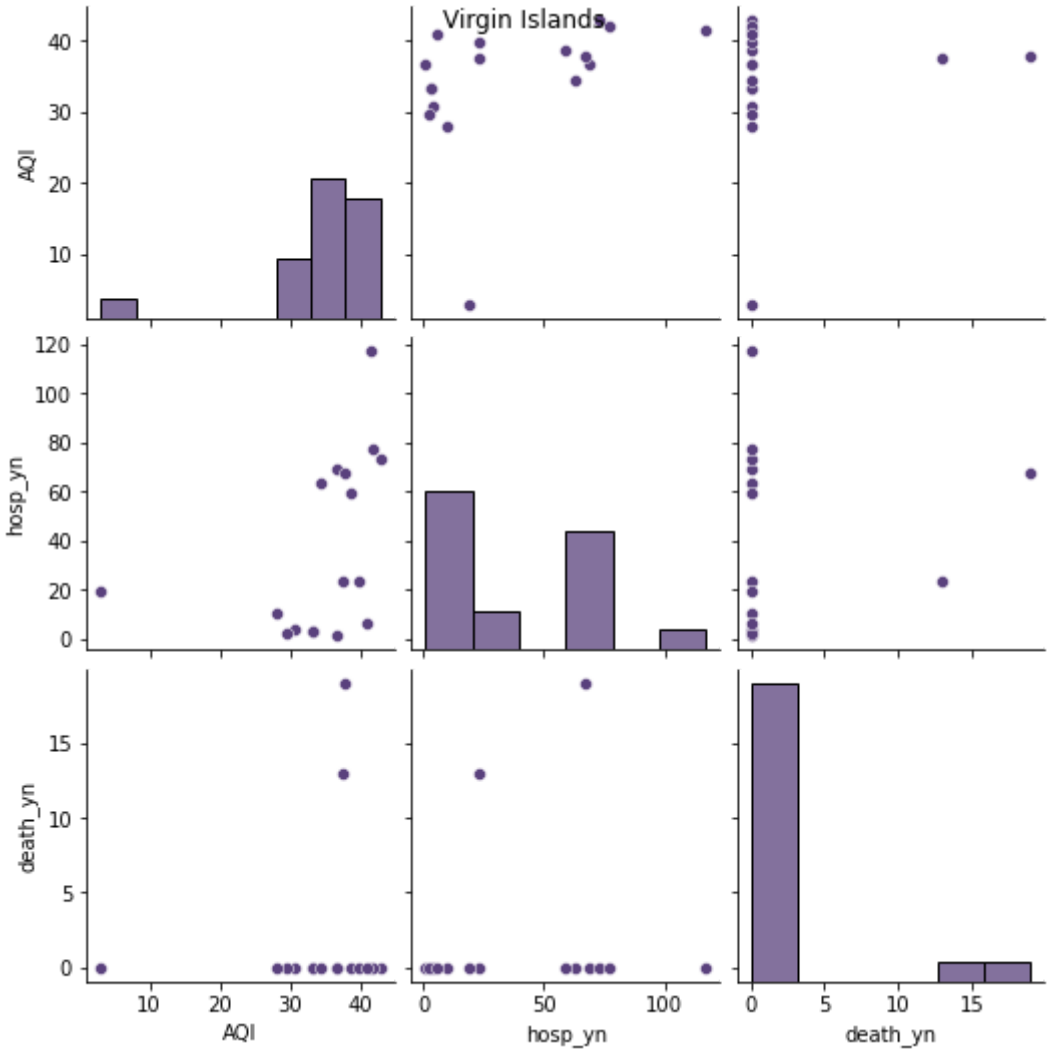

```
<Figure size 1080x504 with 0 Axes>
```



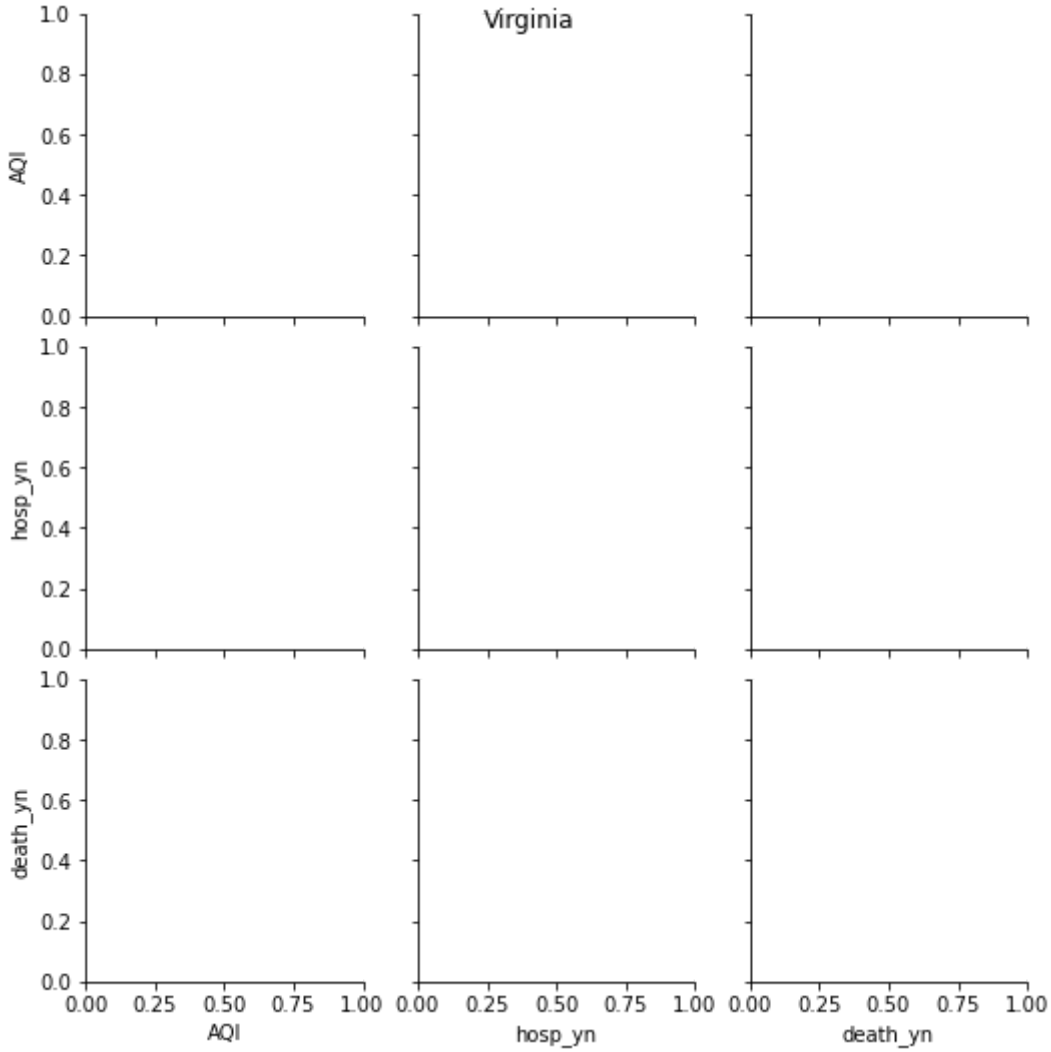

```
<Figure size 1080x504 with 0 Axes>
```



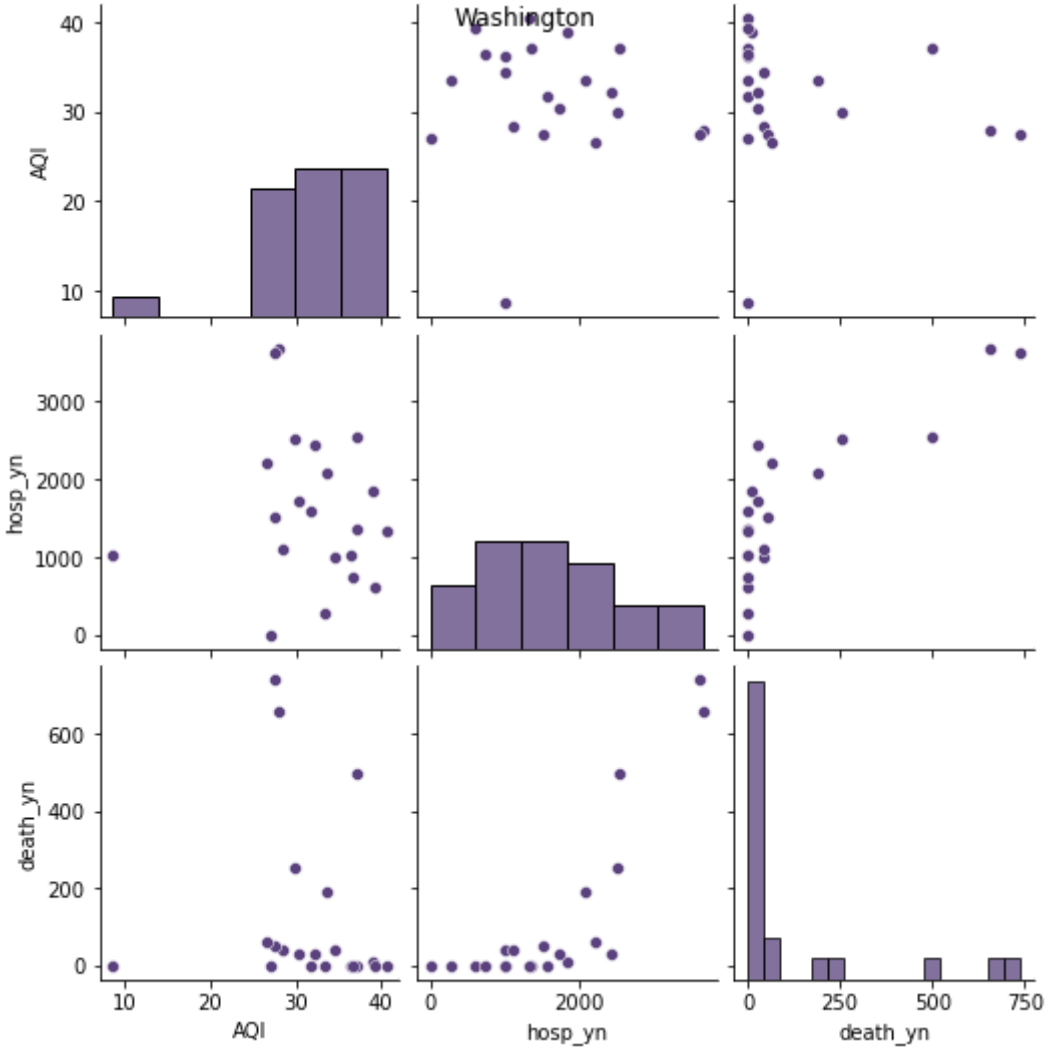

```
<Figure size 1080x504 with 0 Axes>
```



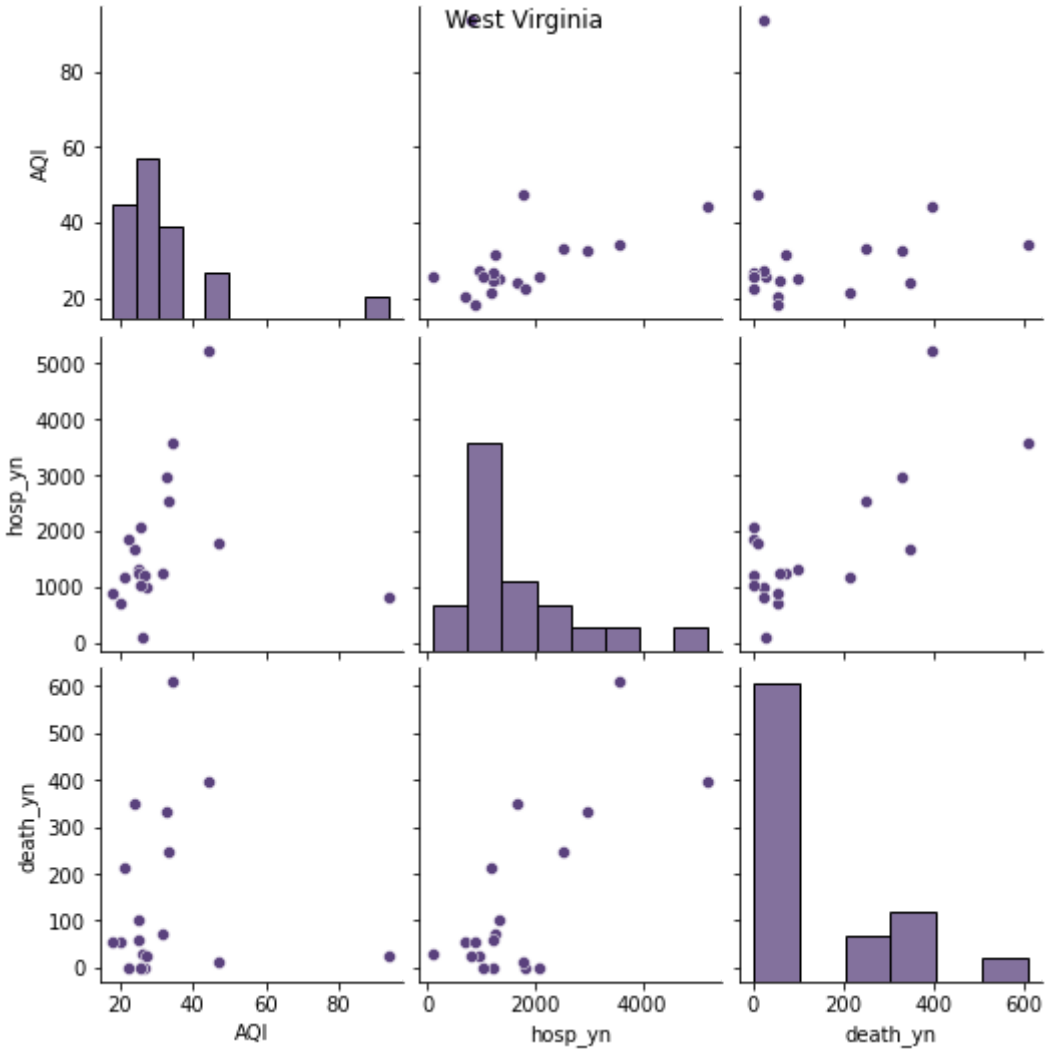

```
<Figure size 1080x504 with 0 Axes>
```



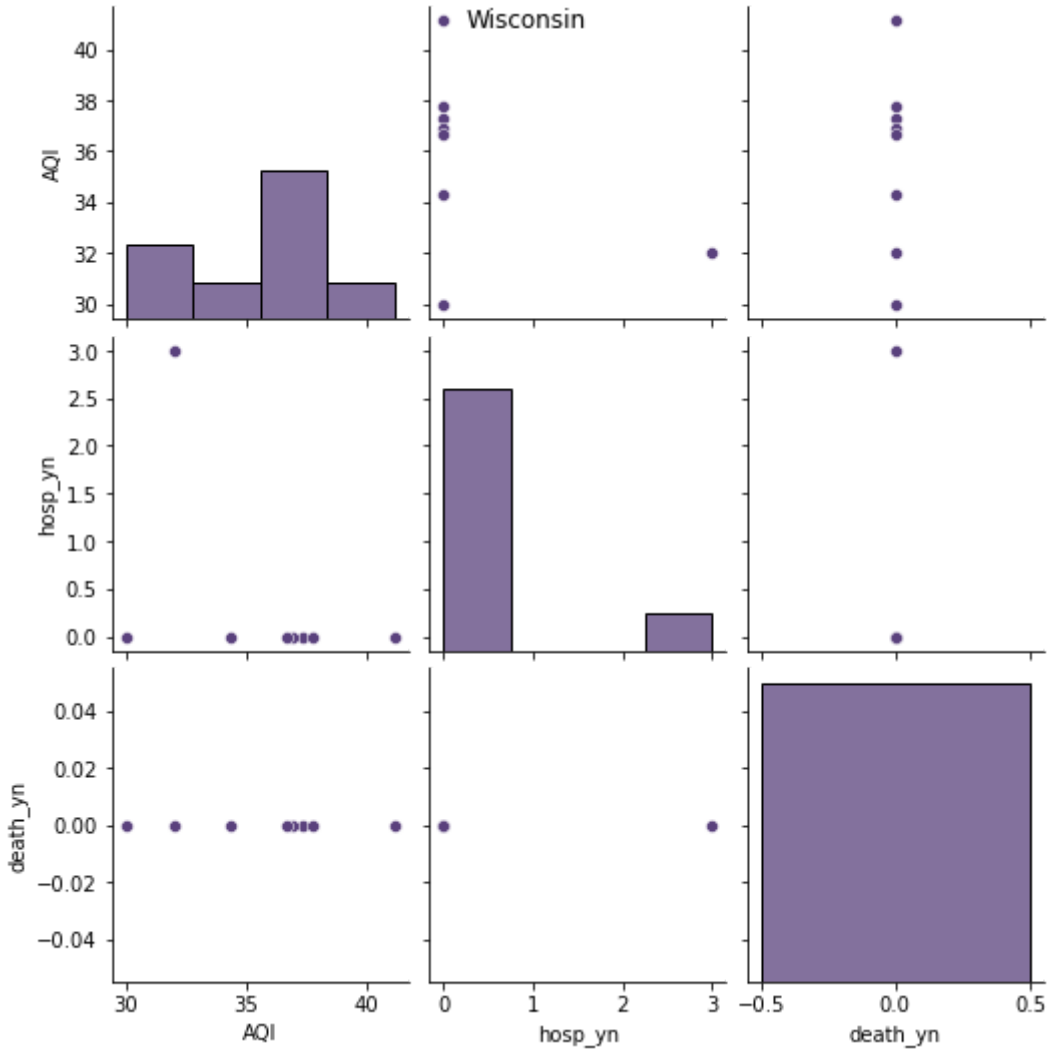

```
<Figure size 1080x504 with 0 Axes>
```



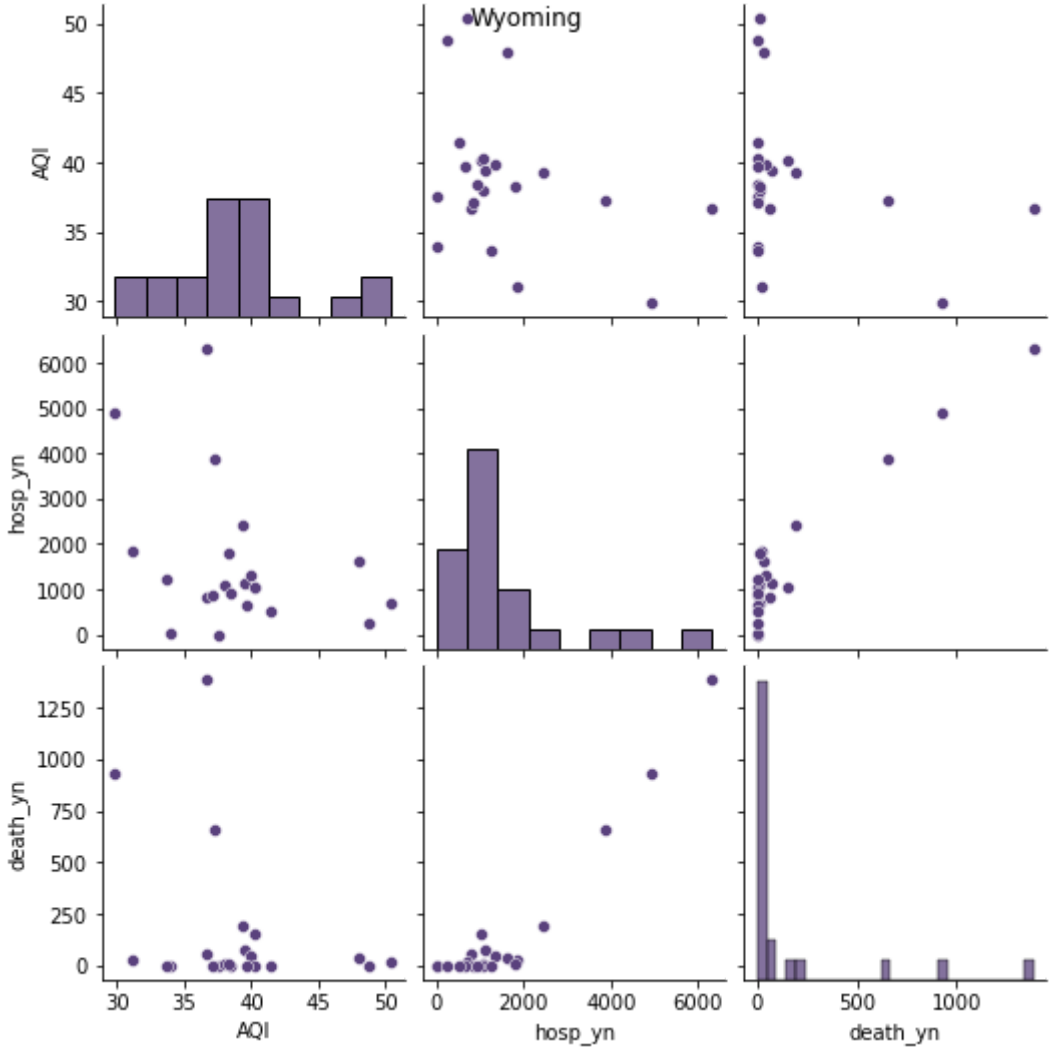

```
<Figure size 1080x504 with 0 Axes>
```



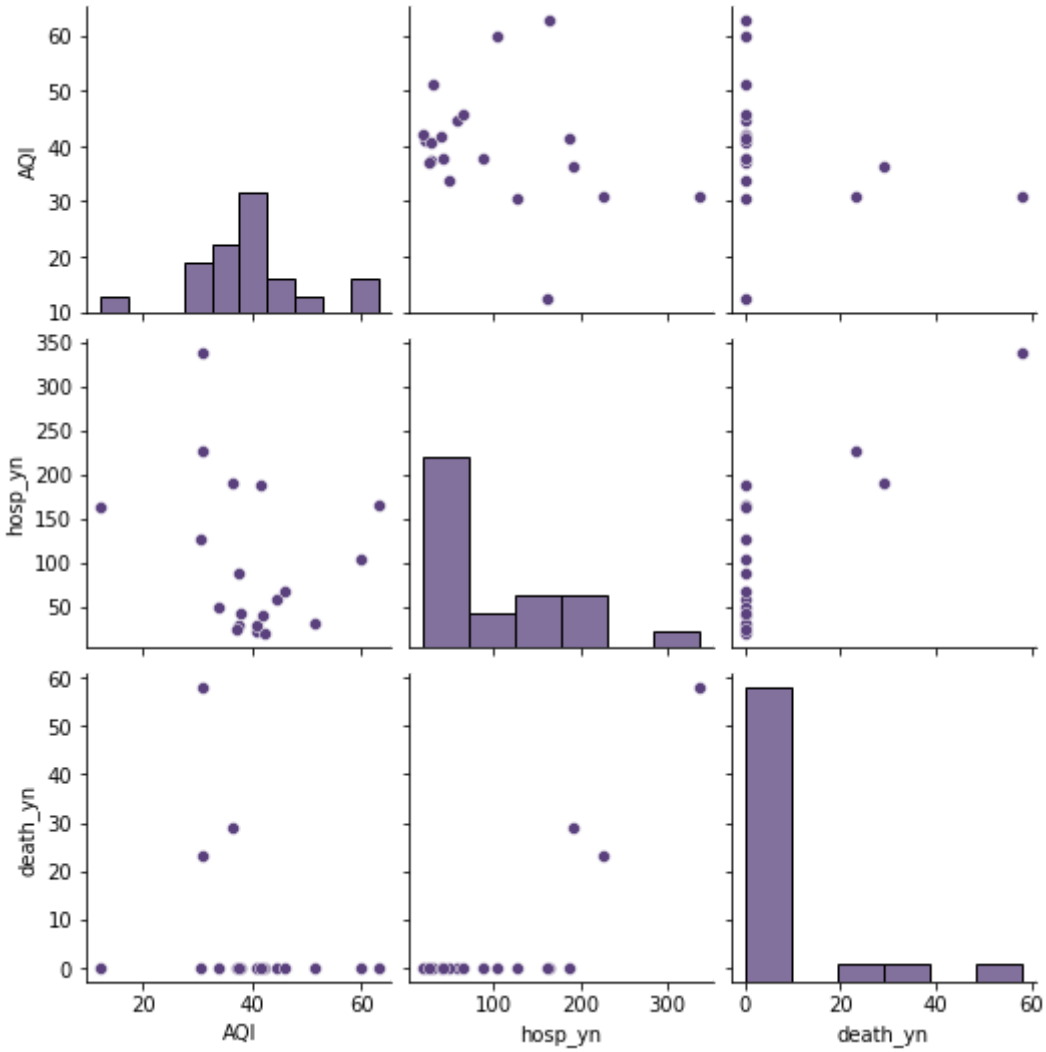

```
<Figure size 1080x504 with 0 Axes>
```




```
<Figure size 1080x504 with 0 Axes>
```




```
<Figure size 1080x504 with 0 Axes>
```



Final Summary

Confounding factors are variables that are associated with both the exposure and the outcome, and can therefore distort the relationship between the exposure and outcome. In this case according to the heat graph of Death and underlying conditions, they are not directly associated with AQI: Hence, they are not considered confounding factors in the relationship between AQI and death or underlying conditions. Confounding factors should be controlled for in order to accurately assess the relationship between the exposure and outcome.