EDA: 500 Cities Current Asthma Prevelance

Benjamin Garnham, Monisola Jayeoba, Taylor Baker, Xamantha Laos, Yemi Kelani

This data was source from the CDC and cleaned using Open Refine and Python. \ Data Source: CDC

In [999…
```
import numpy as np
import pandas as pd
import seaborn as sns
from apyori import apriori
import matplotlib.pyplot as plt
from scipy.stats import f_oneway
from scipy.stats import chi2_contingency
from sklearn.preprocessing import PowerTransformer
```

In [100…
```
#preparing dataframes
asthma_df = pd.read_csv("Cleaned-500CitiesCurrentAsthma.csv")
asthma_df["PopulationCount"] = asthma_df["PopulationCount"].map(lambda pop: int

us_avg = asthma_df[asthma_df["StateAbbr"].str.contains("US")]
asthma_df = asthma_df.drop(us_avg.index)
asthma_df = asthma_df.reset_index()
```

In [100…
```
#utilities
def rip(match_str="", column="DataValueTypeID", ret=["StateAbbr", "Data_Value"]
    if match_str == "":
        ripped = asthma_df.copy()
    else:
        ripped = asthma_df[asthma_df[column].str.contains(match_str)]
    ripped = ripped[[*ret]]
    return ripped
```

In [100…
```
#data record example
asthma_df.head(1)
```

Out[1002]:

	index	Year	StateAbbr	StateDesc	CityName	GeographicLevel	DataSource	Category	U
0	0	2017	CA	California	Livermore	City	BRFSS	Health Outcomes	

1 rows × 27 columns

In [100…
```
mean = asthma_df["Data_Value"].mean()
std = asthma_df["Data_Value"].std()
med = asthma_df["Data_Value"].median()
print(f"Collective Metrics:\n\t Mean: {round(mean,2)}, Std: {round(std,2)}, Mec
```

```
Collective Metrics:
        Mean: 9.73, Std: 1.7, Median: 9.4
```

In [100…
```
ripped = rip("CrdPrv")
mean = ripped["Data_Value"].mean()
std = ripped["Data_Value"].std()
med = ripped["Data_Value"].median()
print(f"Collective Metrics for Data Value Type 'CrdPrv':\n\t Mean: {round(mean,
```

```
Collective Metrics for Data Value Type 'CrdPrv':
        Mean: 9.74, Std: 1.7, Median: 9.4
```

In [100…
```
#mean and std of average state data
ripped = rip("CrdPrv")
state_df = ripped.groupby("StateAbbr").agg(['count','mean', 'std'])
state_mean = state_df["Data_Value"]["mean"].mean()
state_std = state_df["Data_Value"]["mean"].std()
state_med = state_df["Data_Value"]["mean"].median()
print(f"Collective Metrics for State Aggregations on Data Value Type 'CrdPrv':\
```

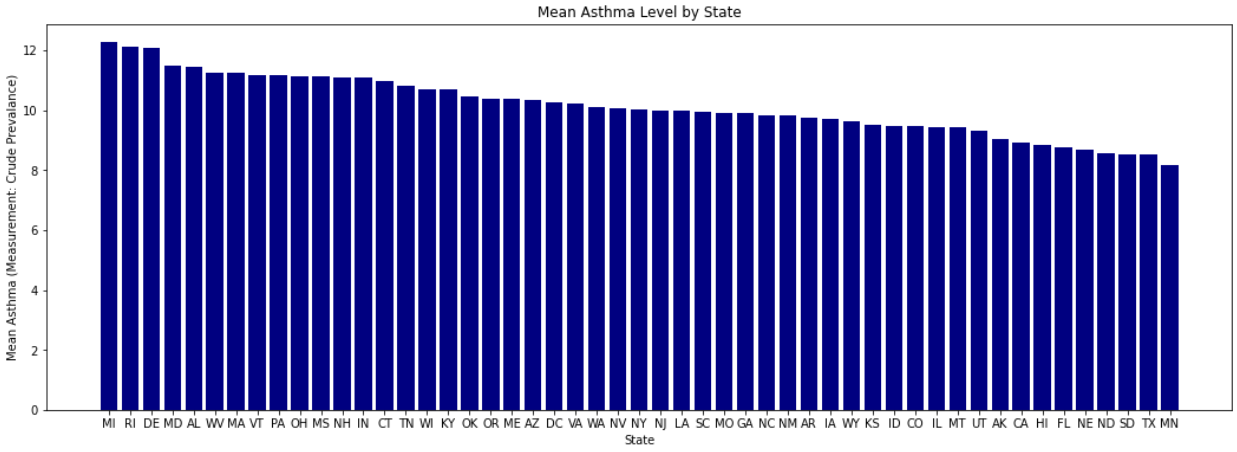
```
Collective Metrics for State Aggregations on Data Value Type 'CrdPrv':
        Mean: 10.14, Std: 1.0, Median: 10.04
```

In [100…
```
ripped = rip("AgeAdjPrv")
mean = ripped["Data_Value"].mean()
std = ripped["Data_Value"].std()
med = ripped["Data_Value"].median()
print(f"Collective Metrics for Data Value Type 'AgeAdjPrv':\n\t Mean: {round(me
```

```
Collective Metrics for Data Value Type 'AgeAdjPrv':
        Mean: 9.46, Std: 1.19, Median: 9.4
```

In [100…
```
ripped = rip("CrdPrv")
state_df = ripped.groupby("StateAbbr").agg(['count','mean', 'std'])
state_df.sort_values(by=('Data_Value', 'mean'), ascending=False, inplace=True,)

plt.figure(figsize=(18,6))
plt.bar(state_df.index, state_df["Data_Value"]["mean"], color ='navy')
plt.xlabel("State")
plt.ylabel("Mean Asthma (Measurement: Crude Prevalance)")
plt.title("Mean Asthma Level by State")
plt.show()
```



ANOVA: Analysis of Variation

Hypothesis: The mean values of the Crude Prevalence of Asthma depend on which State the individual is in? \ **Null Hypothesis:** The mean values of the Crude Prevalence of Asthma do NOT depend on which State the individual is in?

In [100…
```
#logarithmic transformation
pt = PowerTransformer()
ripped = rip("CrdPrv", ret=["StateAbbr", "Data_Value", "PopulationCount"])
ripped["Data_Value"] = pd.DataFrame(
    pt.fit_transform(ripped[["Data_Value"]]), columns=["Data_Value"]
)
log_state_df = ripped[~ripped["Data_Value"].isnull()]
state_count = len(log_state_df["StateAbbr"].unique())

#checking ANOVA assumptions
DV_std = ("Data_Value", "std")
DV_count = ("Data_Value", "count")
log_state_agg_df = log_state_df.groupby("StateAbbr").agg(['count','mean', 'std'
log_state_agg_df.head()

#  sub-sample standard: 100
N = 100
states_to_sample = list(log_state_agg_df[log_state_agg_df[DV_count] >= N].index
sub_sample_df = log_state_df[log_state_df["StateAbbr"].isin(states_to_sample)]
sub_sample_df = sub_sample_df.groupby("StateAbbr", as_index=False).apply(lambda
sub_sample_df = sub_sample_df.reset_index()[["StateAbbr", "Data_Value", "Popula
dropped_states = ", ".join([x for x in log_state_df["StateAbbr"].unique() if x
missing_count = state_count - len(states_to_sample)
print(f"{missing_count} states unaccounted for: {dropped_states}\n")

#  check variance and distribution
sub_sample_agg_df = sub_sample_df.groupby("StateAbbr").agg(['count','mean', 'st
set_min = sub_sample_agg_df[DV_std].min()
set_max = sub_sample_agg_df[DV_std].max()
ratio = set_max / set_min
print(f"The ratio of the maximum to minimum standard deviation is {ratio}.\n\t
```

```
12 states unaccounted for: ND, WI, AK, DE, ID, ME, MS, MT, NH, SD, WV, VT

The ratio of the maximum to minimum standard deviation is 1.8221445507906306.
        max: 1.0456837052059011, min: 0.5738752750171098
```

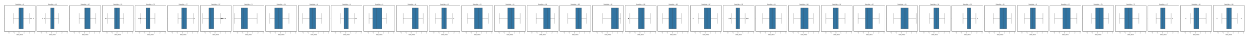
In [100…
```
#buckets used for ANOVA test
sub_sample_agg_df.head()
```

Out[1009]:

	Data_Value			PopulationCount		
	count	mean	std	count	mean	std
StateAbbr						
AL	100	0.484614	0.782956	100	3638.03	8052.735712
AR	100	0.099848	0.626228	100	8249.93	22511.304900
AZ	100	0.564120	0.819135	100	8427.01	30822.080258
CA	100	-0.070721	0.754054	100	12772.87	38228.802343
CO	100	-0.304118	0.650511	100	9174.26	42273.766801

In [101…
```
sns.catplot(data=sub_sample_df, kind="box", col="StateAbbr", x="Data_Value", hu
```

Out[1010]: <seaborn.axisgrid.FacetGrid at 0x2dc6bd370>



In [101…
```
data = []
for idx, s in enumerate(states_to_sample):
    if s == list(sub_sample_df[idx * 100 : (idx * 100) + 1]["StateAbbr"])[0]:
        data.append(list(sub_sample_df[idx * 100 : (idx + 1) * 100]["Data_Value
    else:
        print("Mistmatch of states, analysis may be invalid.")

#ANOVA
F, p = f_oneway(*data)
conclusions = f"""
               \033[1mANOVA Conclusions:\033[0m\n\t
               The \033[1mp-value is {p}\033[0m. Because the p-value is less t
               we can reject the null hypothesis proposed above.
              """
print(conclusions)
```

ANOVA Conclusions:

The **p-value is 7.46478902662807e-147.** Because the p-value is l
ess than the threshold (0.05)
we can reject the null hypothesis proposed above.

Investigate potential corrleation between asthma level and population count

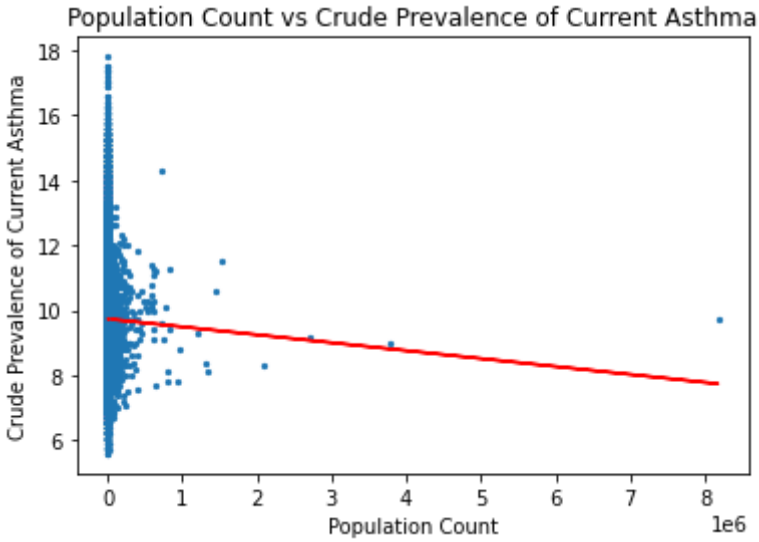
In [101…
```
corr_df = rip("CrdPrv", ret=["PopulationCount", "Data_Value"])
pearson_coef = corr_df.corr()["PopulationCount"]["Data_Value"]
spearman_coef = corr_df.corr(method="spearman")["PopulationCount"]["Data_Value'
print(f"PEARSON: {round(pearson_coef,5)}, SPEARMAN: {round(spearman_coef,5)}")
```

```
PEARSON: -0.00961, SPEARMAN: -0.12662
```

In [101…
```
plt.scatter(corr_df["PopulationCount"], corr_df["Data_Value"], s=5)
plt.title("Population Count vs Crude Prevalence of Current Asthma")
plt.ylabel("Crude Prevalence of Current Asthma")
plt.xlabel("Population Count")

#trendline
z = np.polyfit(corr_df["PopulationCount"], corr_df["Data_Value"], 1)
p = np.poly1d(z)
plt.plot(corr_df["PopulationCount"], p(corr_df["PopulationCount"]), color="red'
```

Out[1013]: [<matplotlib.lines.Line2D at 0x2de804fd0>]



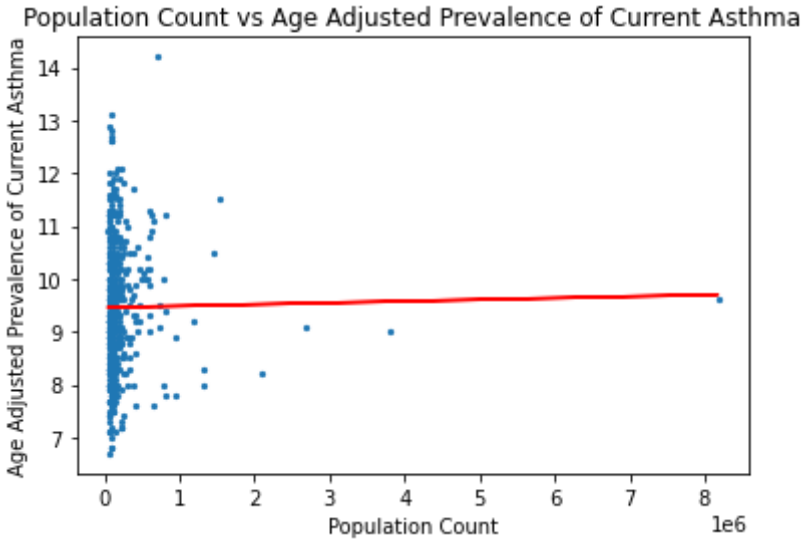
In [101…
```
corr_df = rip("AgeAdjPrv", ret=["PopulationCount", "Data_Value"])
pearson_coef = corr_df.corr()["PopulationCount"]["Data_Value"]
spearman_coef = corr_df.corr(method="spearman")["PopulationCount"]["Data_Value"
print(f"PEARSON: {round(pearson_coef,5)}, SPEARMAN: {round(spearman_coef,5)}")
```

```
PEARSON: 0.01156, SPEARMAN: 0.06083
```

In [101…
```
plt.scatter(corr_df["PopulationCount"], corr_df["Data_Value"], s=5)
plt.title("Population Count vs Age Adjusted Prevalence of Current Asthma")
plt.ylabel("Age Adjusted Prevalence of Current Asthma")
plt.xlabel("Population Count")

#trendline
z = np.polyfit(corr_df["PopulationCount"], corr_df["Data_Value"], 1)
p = np.poly1d(z)
plt.plot(corr_df["PopulationCount"], p(corr_df["PopulationCount"]), color="red"
```

Out[1015]:
```
[<matplotlib.lines.Line2D at 0x2df42fd30>]
```



Chi-Square Test of Independence

Hypothesis: The Asthma Prevelance Level is dependent on the Population for a given city. \
Null Hypothesis: The Asthma Prevelance Level is NOT dependent on the Population of a

given city.

Basis: **Delineating Metropolitan and Micropolitan Statistical Areas** - "Each metropolitan statistical area must have at least one urbanized area of 50,000 or more inhabitants. Each micropolitan statistical area must have at least one urban cluster of at least 10,000 but less than 50,000 population."

In [101…
```
ripped = rip("CrdPrv", ret=["PopulationCount", "StateAbbr", "Data_Value"])
asthma_prev_above_median = ripped["Data_Value"].map(lambda prev: prev >= state_
micropolitan = ripped["PopulationCount"].map(lambda pop: pop > 10000 and pop <
metropolitan = ripped["PopulationCount"].map(lambda pop: pop >= 50000)
micropolitan_contingency = pd.crosstab(micropolitan, asthma_prev_above_median)
micropolitan_contingency
```

Out[1016]:

Data_Value	False	True
PopulationCount		
False	17364	10159
True	169	18

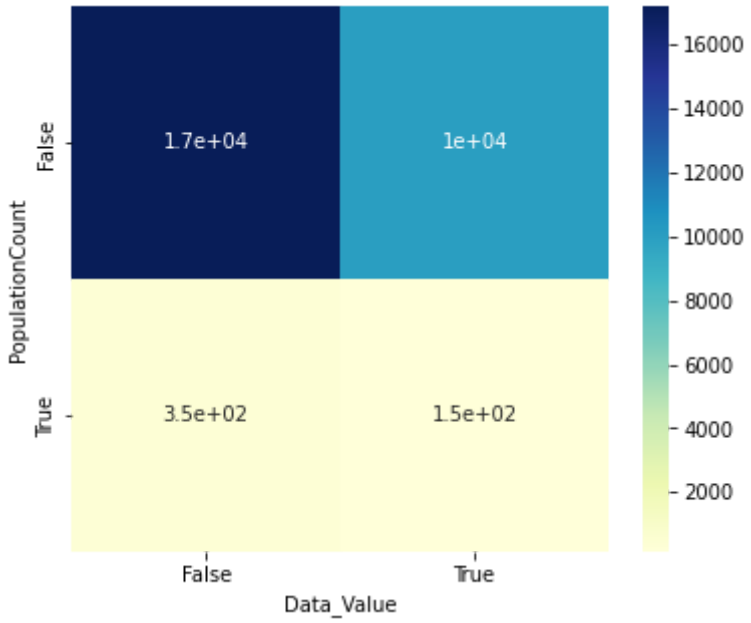
In [101…
```
metropolitan_contingency = pd.crosstab(metropolitan, asthma_prev_above_median)
metropolitan_contingency
```

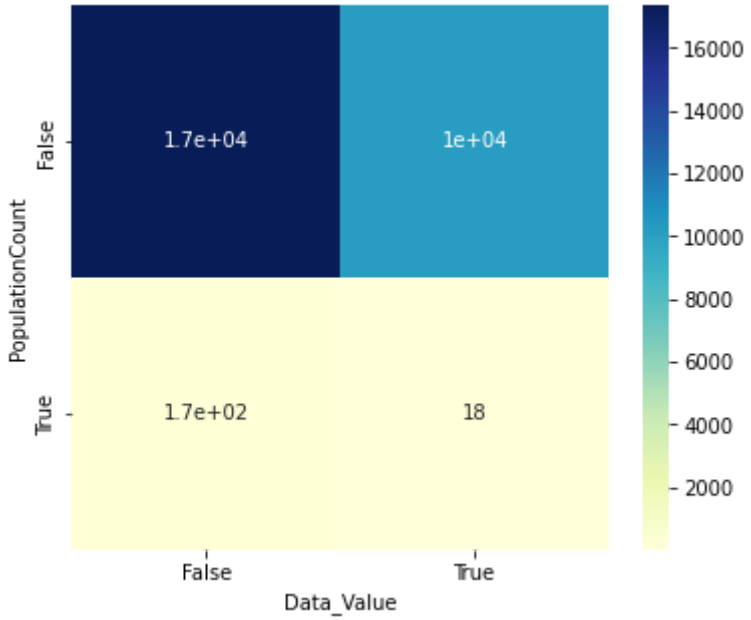
Out[1017]:

Data_Value	False	True
PopulationCount		
False	17183	10028
True	350	149

In [101…
```
plt.figure(figsize=(6,5))
sns.heatmap(metropolitan_contingency, annot=True, cmap="YlGnBu")
plt.figure(figsize=(6,5))
sns.heatmap(micropolitan_contingency, annot=True, cmap="YlGnBu")
```

Out[1018]:
```
<AxesSubplot:xlabel='Data_Value', ylabel='PopulationCount'>
```





In [101…

```
chi2_mi, p_mi, dof_mi, expected_mi = chi2_contingency(micropolitan_contingency)
chi2_me, p_me, dof_me, expected_me = chi2_contingency(metropolitan_contingency)
print(f"Associated P-values:\n\t \033[1mMicropolitan p-value\033[0m: {p_mi}, \(
conclusions = f"""
            The Chi-Square test of independence presented a \033[1mp-value of {
            \033[1mp-value of {p_me}\033[0m for Metropolitan areas. Because bot
            We can reject the null hypothesis proposed above at a confidence le
              """
print(conclusions)
```

Associated P-values:
Micropolitan p-value: 2.208730870072854e-14, **Metropolitan p-value:** 0.0015543115885484806

The Chi-Square test of independence presented a **p-value of 2.208730870072854e-14** for Micropolitan areas and a
p-value of 0.0015543115885484806 for Metropolitan areas. Because both calulated p-values are less than the threshold (0.05)
We can reject the null hypothesis proposed above at a confidence level of 95%.