Homework 2-2 Submission

```python
import pandas as pd
import numpy as np
from pathlib import Path
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
data2014= pd.read_csv(f'../hw2/data/output/data-2014.csv')
data2015= pd.read_csv(f'../hw2/data/output/data-2015.csv')
data2016= pd.read_csv(f'../hw2/data/output/data-2016.csv')
data2017= pd.read_csv(f'../hw2/data/output/data-2017.csv')
data2018= pd.read_csv(f'../hw2/data/output/data-2018.csv')
data2019= pd.read_csv(f'../hw2/data/output/data-2019.csv')
```

```
/tmp/ipykernel_1734625/1389888897.py:4: DtypeWarning: Columns (44,59) have mixed types. Speci
  data2017= pd.read_csv(f'../hw2/data/output/data-2017.csv')
```

```python
#merge years
years = range(2014, 2020)
dfs = []
for year in years:
    df = pd.read_csv(f'../hw2/data/output/data-{year}.csv')
    dfs.append(df)

# Concatenate into one DataFrame
data = pd.concat(dfs, ignore_index=True)
```

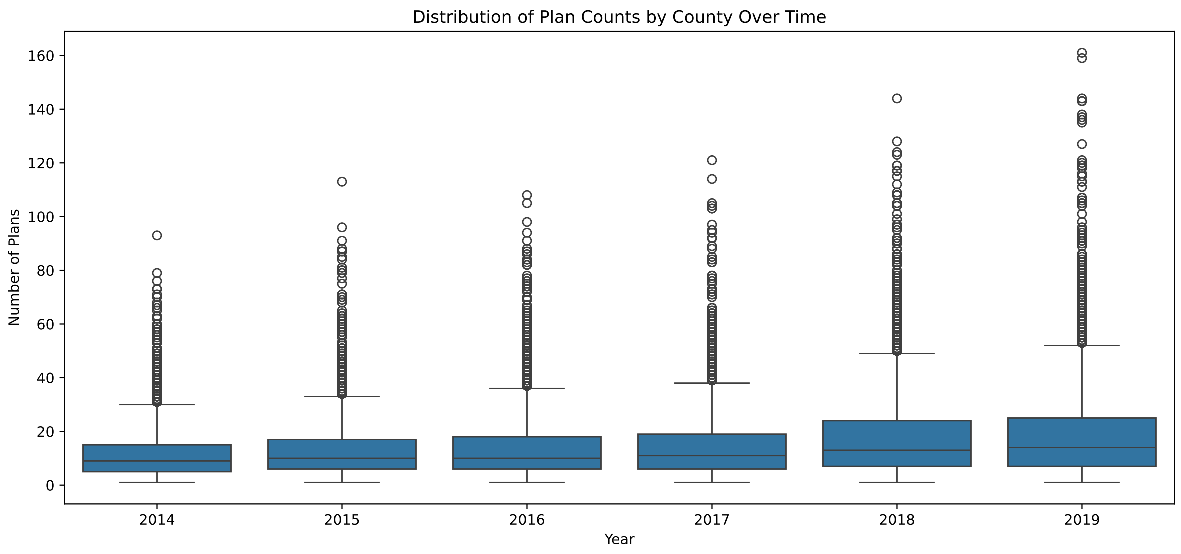
```
/tmp/ipykernel_1734625/2601088650.py:5: DtypeWarning: Columns (44,59) have mixed types. Speci
  df = pd.read_csv(f'../hw2/data/output/data-{year}.csv')
```

Question 1:

```
# question 1
q1 = data[data['snp'] == 'No']

# Remove 800-series plans (if planid is float)
q1 = q1[q1['planid'] < 800]

# Keep only plans with Part D (meaning they likely offer Part C in MA context)
q1 = q1[q1['partd'] == 'Yes']
```

```
plan_counts = q1.groupby(['year', 'county'])['planid'].nunique().reset_index()
```

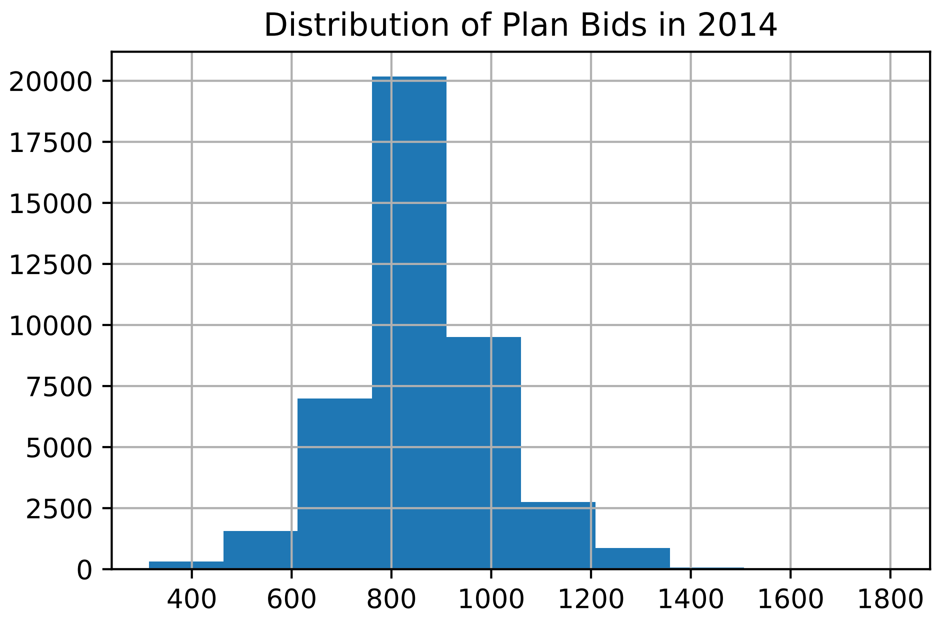
```
plt.figure(figsize=(14, 6))
sns.boxplot(data=plan_counts, x='year', y='planid')
plt.title('Distribution of Plan Counts by County Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Plans')
plt.show()
```



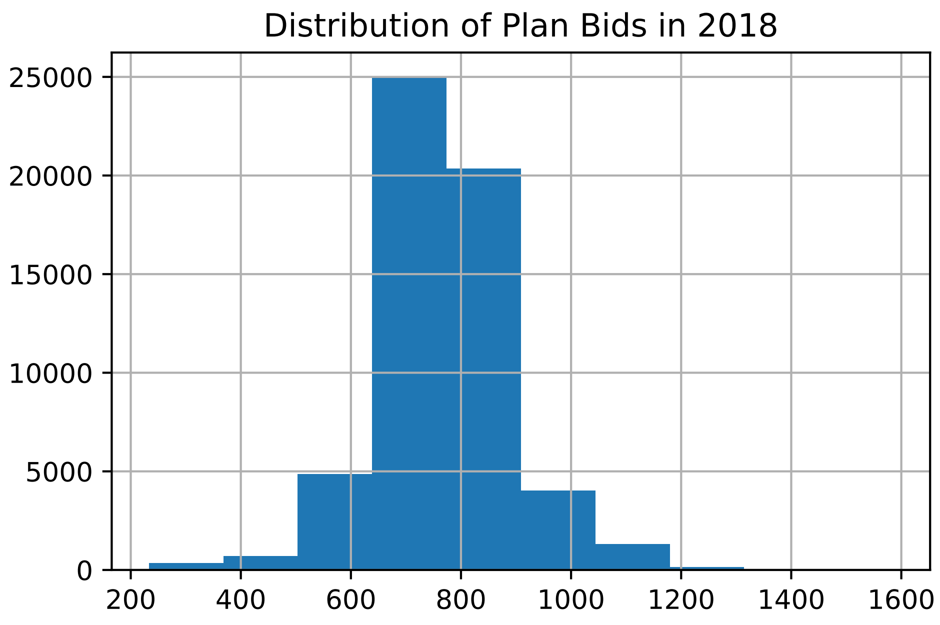
Interpretation: Over time, the number of plans seems to be increasing. The median hovers around 10-15 plans, but there are seemingly more and more extreme outliers as time goes on, with some counties having as much as 160 plan counts in 2019. 10-15 plan types seems to be a reasonable amount of plans, but with there being a visually large amount of outliers across 2014-2019 ranging from 30-160 plans, it seems like there could be too many plans.

Question 2:

```
#question 2
data2014['bid'].hist()
plt.title('Distribution of Plan Bids in 2014')
plt.show()
```



```
data2018['bid'].hist()
plt.title('Distribution of Plan Bids in 2018')
plt.show()
```



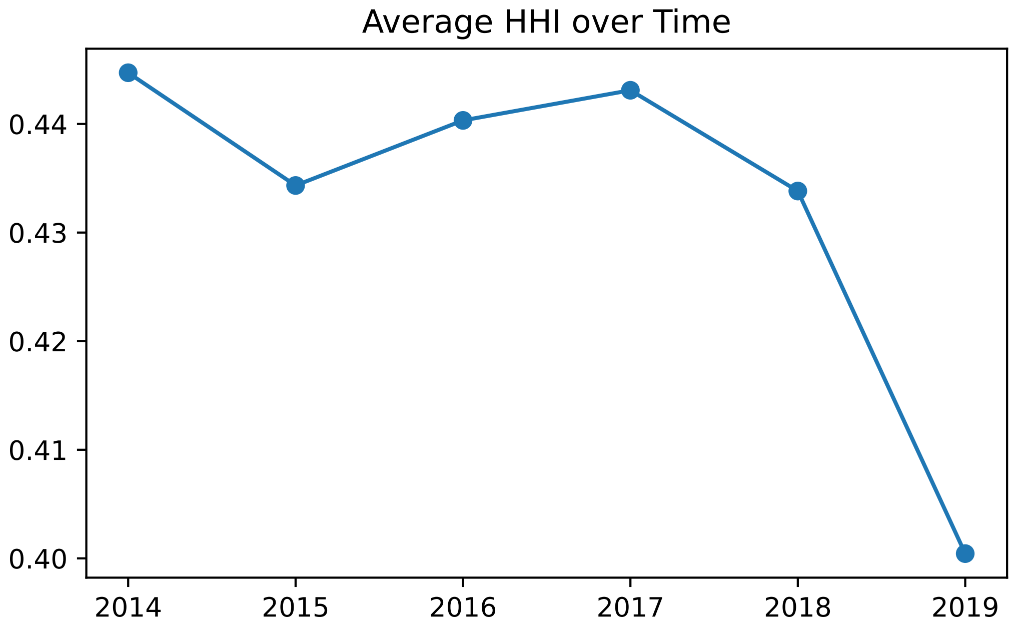
Over time, the distribution of plan bids has shifted left, with the highest concentration being around 900 in 2014. In 2018, the distribution becomes much narrower and is mainly concentrated around 700.

Question 3:

```
#question 3
data["avg_enrollment"] = pd.to_numeric(data["avg_enrollment"], errors="coerce")
data["avg_enrollment"].notna()

# Compute contract-level enrollment per county-year
contractenrollment = (
    data.groupby(["year", "fips", "contractid"])["avg_enrollment"]
    .sum()
    .reset_index(name="contract_enroll")
)

# Compute total county enrollment
countyenroll = (
    contractenrollment.groupby(["year", "fips"])["contract_enroll"]
    .sum()
    .reset_index(name="county_total")
)

contractenrollment = contractenrollment.merge(
    countyenroll,
    on=["year", "fips"],
    how="left"
)

# Compute market shares and HHI
contractenrollment["share"] = (
    contractenrollment["contract_enroll"]
    / contractenrollment["county_total"]
)

contractenrollment["share_sq"] = contractenrollment["share"] ** 2
county_hhi = (
    contractenrollment.groupby(["year", "fips"])["share_sq"]
    .sum()
    .reset_index(name="hhi")
)

# Average HHI by year
avg_hhi = (
    county_hhi.groupby("year")["hhi"]
    .mean()
    .reset_index(name="avg_hhi")
)

print(avg_hhi)
```

```
   year   avg_hhi
0  2014  0.444715
1  2015  0.434334
2  2016  0.440327
3  2017  0.443102
4  2018  0.433823
5  2019  0.400439
```

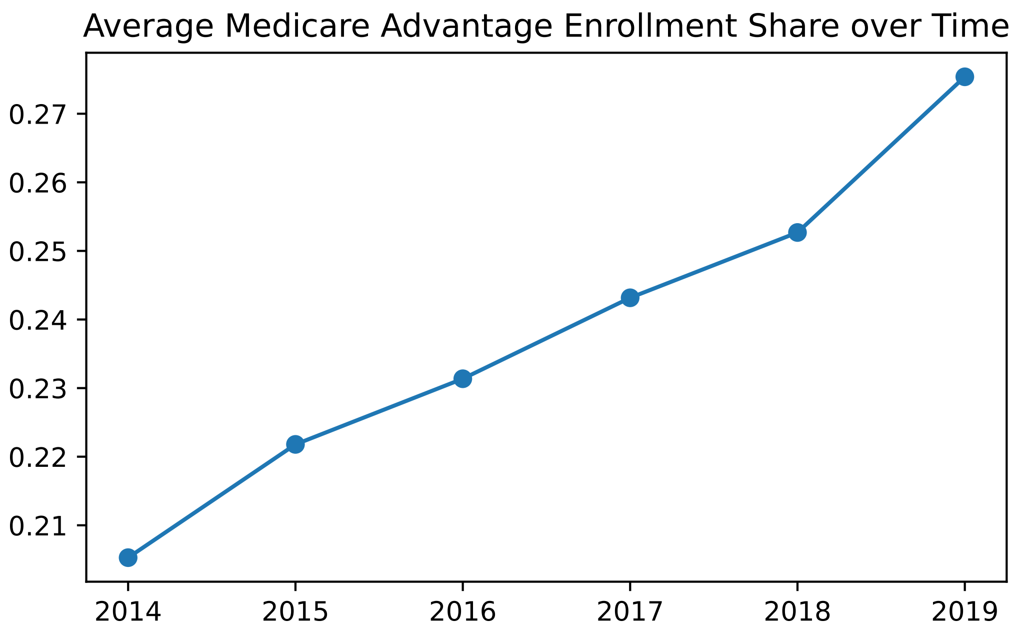
```
plt.figure()
plt.plot(avg_hhi["year"], avg_hhi["avg_hhi"], marker="o")
plt.title("Average HHI over Time")
plt.tight_layout()
plt.show()
```



Average HHI staayed mostly constant around .43 to .44 up until 2018. In 2019, average HHI dropped significantly to .4, telling us the market has become more competitive.

Question 4:

```
#question 4
pos = (
    data.groupby(["year", "fips"], as_index=False)
       .agg({
           "avg_enrolled": "first",
           "avg_eligibles": "first"
       })
)
pos["ma_share"] = (
    pos["avg_enrolled"] / pos["avg_eligibles"]
)

mashares = (
    pos.groupby("year", as_index=False)["ma_share"]
    .mean()
    .rename(columns={"ma_share": "avg_ma_share"})
)

plt.figure()
plt.plot(mashares["year"], mashares["avg_ma_share"], marker="o")
plt.title("Average Medicare Advantage Enrollment Share over Time")
plt.tight_layout()
plt.show()
```



Average Medicare Advantage Enrollment compared to eligibles has steadily increased from 2014 to 2019, from around 20% enrollment rate to around 28%.

Question 5:

```
#question 5
countytotals = (
    data2018.groupby("fips")["avg_enrollment"]
      .sum()
      .reset_index(name="total_enrollment")
```

```python
)

data2018 = data2018.merge(countytotals, on="fips", how="inner")
```

```python
data2018["market_share"] = data2018["avg_enrollment"] / data2018["total_enrollment"]

county_hhi = (
    data2018.groupby("fips")["market_share"]
      .apply(lambda x: np.sum(x ** 2))
      .reset_index(name="hhi")
)

# Define competitive vs uncompetitive
comp = county_hhi["hhi"].quantile(0.33)
unc = county_hhi["hhi"].quantile(0.66)

county_hhi["market_type"] = np.where(
    county_hhi["hhi"] <= comp, "competitive",
    np.where(county_hhi["hhi"] >= unc, "uncompetitive", "mid")
)
county_hhi = county_hhi[county_hhi["market_type"].isin(["competitive", "uncompetitive"])]


data2018 = data2018.merge(county_hhi[["fips", "market_type"]], on="fips", how="inner")
```

```python
avgbid = (
    data2018.groupby("market_type")["bid"]
      .mean()
      .reset_index(name="avg_bid")
)
print(avgbid)
```

```
     market_type     avg_bid
0    competitive  766.228886
1  uncompetitive  772.622669
```

Question 6:

```
#question 6
data2018 = data2018.dropna(
    subset=["bid", "payment_partc", "total_enrollment", "market_type"]
)

# Create quartiles
data2018["ffs_quartile"] = pd.qcut(
    data2018["payment_partc"],
    4,
    labels=False
)

for q in range(4):
    data2018[f"q{q}"] = (data2018["ffs_quartile"] == q).astype(int)
#checking qs
print(data2018[[f"q{i}" for i in range(4)]].sum())
```

```
q0    9856
q1    9884
q2    9938
q3    9743
dtype: int64
```

```
table = (
    data2018
    .groupby(["ffs_quartile", "market_type"])["bid"]
    .mean()
    .unstack()
)

table.columns = ["Control Mean Bid", "Treated Mean Bid"]
print(table)
```

```
              Control Mean Bid  Treated Mean Bid
ffs_quartile
0                   691.199596        718.357386
1                   781.694478        798.910212
2                   786.459810        783.615659
3                   807.128882        790.141730
```

Question 7:

```
#question 7
from scipy.spatial.distance import cdist

X = data2018[["q0", "q1", "q2"]].values
T = data2018["market_type"].values
Y = data2018["bid"].values

X_treated = X[T == 'uncompetitive']
X_control = X[T == 'competitive']

Y_treated = Y[T == 'uncompetitive']
Y_control = Y[T == 'competitive']
```

```
# Nearest neighbor matching (1-to-1) with inverse variance distance based on quartiles of FF
var = np.var(X, axis=0)
inv_var = 1 / var

X_treated_scaled = X_treated * inv_var
X_control_scaled = X_control * inv_var

dist_tc = cdist(X_treated_scaled, X_control_scaled)
match_tc = dist_tc.argmin(axis=1)
effect_tc = Y_treated - Y_control[match_tc]

dist_ct = cdist(X_control_scaled, X_treated_scaled)
match_ct = dist_ct.argmin(axis=1)
effect_ct = Y_treated[match_ct] - Y_control

ate1 = np.mean(np.concatenate([effect_tc, effect_ct]))

print("ATE (Inverse Variance Matching):", ate1)
```

```
ATE (Inverse Variance Matching): 20.593875395145883
```

```
#Nearest neighbor matching (1-to-1) with Mahalanobis distance based on quartiles of FFS cost

# Covariance matrix
S = np.cov(X, rowvar=False)
```

```python
S_inv = np.linalg.inv(S)

X_treated = X[T == 'uncompetitive']
X_control = X[T == 'competitive']

Y_treated = Y[T == 'uncompetitive']
Y_control = Y[T == 'competitive']

dist_tc = cdist(X_treated, X_control, metric='mahalanobis', VI=S_inv)
match_tc = dist_tc.argmin(axis=1)
effect_tc = Y_treated - Y_control[match_tc]

dist_ct = cdist(X_control, X_treated, metric='mahalanobis', VI=S_inv)
match_ct = dist_ct.argmin(axis=1)
effect_ct = Y_treated[match_ct] - Y_control

ate2 = np.mean(np.concatenate([effect_tc, effect_ct]))

print("ATE (Mahalanobis Matching):", ate2)
```

```
ATE (Mahalanobis Matching): 20.593875395145883
```

```python
#Inverse propensity weighting, where the propensity scores are based on quartiles of FFS cos
import statsmodels.api as sm
Tnum = np.where(T == "competitive", 0, 1)

X_ps = sm.add_constant(data2018[["q0","q1","q2"]])
logit = sm.Logit(Tnum, X_ps).fit(disp=0)

ps = logit.predict(X_ps)
data2018["ps"] = ps

weights = np.where(Tnum == 1, 1/ps, 1/(1-ps))

ate3 = (
    np.sum(weights * Tnum * Y) / np.sum(weights * Tnum)
    - np.sum(weights * (1-Tnum) * Y) / np.sum(weights * (1-Tnum))
)

print("ATE (Inverse propensity weighting):", ate3)
```

```
ATE (Inverse propensity weighting): 6.191027628639517
```

```python
#Simple linear regression, adjusting for quartiles of FFS costs using dummy variables and ap
data2018["T"] = Tnum

for q in range(3):
    data2018[f"T_q{q}"] = data2018["T"] * data2018[f"q{q}"]

X_reg = data2018[["T","q0","q1","q2","T_q0","T_q1","T_q2"]]
X_reg = sm.add_constant(X_reg)

model = sm.OLS(Y, X_reg).fit()

# ATE = average predicted treatment effect
beta_T = model.params["T"]
beta_interactions = np.array([
    model.params["T_q0"],
    model.params["T_q1"],
    model.params["T_q2"],
    0  # reference quartile (q3)
])

quartile_probs = data2018["ffs_quartile"].value_counts(normalize=True).sort_index().values

ate4 = beta_T + np.sum(beta_interactions * quartile_probs)


print("ATE (Regression):", ate4)
```

```
ATE (Regression): 6.191027628638736
```

```python
results = pd.DataFrame({
    "Estimator": [
        "NN Matching (Inverse Variance)",
        "NN Matching (Mahalanobis)",
        "Inverse Propensity Weighting",
        "Regression with Quartile Interactions"
    ],
    "ATE": [
        ate1,
        ate2,
        ate3,
        ate4
    ]
})

print(results)
```

```
                             Estimator        ATE
0        NN Matching (Inverse Variance)  20.593875
1             NN Matching (Mahalanobis)  20.593875
2           Inverse Propensity Weighting   6.191028
3  Regression with Quartile Interactions   6.191028
```

Question 8:

The Nearest Neighbors Matching with Inverse Variance and Mahalanobis were identical, while the Inverse Propensity Weighting and Regression were identical, but these two groups were very different compared to each other.

Question 9:

```
#Question 9
#Estimator of choice: regression

X_cont = data2018[["T","payment_partc","total_enrollment"]]
X_cont = sm.add_constant(X_cont)

model_cont = sm.OLS(Y, X_cont).fit()

ate9 = model_cont.params["T"]

print("ATE (Regression - Continuous Controls):", ate9)
```

```
ATE (Regression - Continuous Controls): -2.622298718964715
```

The ATE calculated here is very different than what we saw in question 7 using the quartiles. The ATE is negative as well, telling us the treatment (in this case is an uncompetitive market) has a negative effect on bids.

Question 10:

Overall, I was definitely challenged but was still able to understand what was going on for this homework. I ran into many issues reading the raw data and cleaning it up, but eventually worked through it. I found the first part of this assignment to not be too challenging, as I have had experience before with simple data visualization in python. However, the second part (question 5 and onward) definitely was quite confusing. I definitely had to use outside resources to help figure out what code and additional libraries/functions I needed in order to calculate ATEs as this was new to me. One thing that was challenging/aggrevating is realizing I was missing something that needed to be downloaded or having a small error or typo that would require me to restart everything, but it eventually worked out.