Analysis_4_2

April 9, 2025

Kathryn Mawhinney
Dr. McCarthy
Research in Health Economics
April 6, 2025

Homework 4-1

In this assignment, you'll again work with the Medicare Advantage data. These data are described in detail in the Medicare Advantage GitHub Repo. We worked with a subset of these data back in assignment 1; however, this assignment requires that you work with a more complete version of the Medicare Advantage data. We'll again focus on the years 2010-2015. Once you have the data downloaded and the code running, answer the following questions:

The due date for initial submission is 4/7, the revision due date is 4/9, and the final due date is Friday, 4/11.

[65]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import Markdown, display
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Load the final cleaned data
final_data = pd.read_csv("/Users/kathrynmawhinney/Documents/GitHub/Homework4/
↪data/output/final_ma_data.csv")
```

```
/var/folders/8p/wmnjrdd55rx2pn76f5j7m2tw0000gn/T/ipykernel_45977/3879530120.py:1
0: DtypeWarning: Columns (68,98,99,100,101,102,103,104,105,106) have mixed
types. Specify dtype option on import or set low_memory=False.
  final_data = pd.read_csv("/Users/kathrynmawhinney/Documents/GitHub/Homework4/d
ata/output/final_ma_data.csv")
```

0.0.1 Summarize the Data

1. Remove all SNPs, 800-series plans, and prescription drug only plans (i.e., plans that do not offer Part C benefits). Provide a box and whisker plot showing the distribution of plan counts by county over time. Do you think that the number of plans is sufficient, too few, or too many?

[66]:
```python
# Confirm that all SNPs are removed
print("Any SNPs remaining?", final_data["snp"].unique())

# Confirm that no planid is in the 800-899 range (inclusive)
print("Any 800-series plans?", final_data[(final_data["planid"] >= 800) &␣
 ↪(final_data["planid"] < 900)].shape[0])

# Confirm that all remaining plans offer Part C benefits (i.e., are not PDPs)
# If there's a `plan_type` or `partd` column, check for drug-only plans
if "plan_type" in final_data.columns:
    print("Plan types present:", final_data["plan_type"].unique())

if "partd" in final_data.columns:
    print("Any Part D only plans?", final_data["partd"].unique())
```

```
Any SNPs remaining? ['No']
Any 800-series plans? 0
Plan types present: ['Local PPO' 'National PACE' 'HMO/HMOPOS'
 'Continuing Care Retirement Community' '1876 Cost' 'PFFS' 'ESRD I'
 'PSO (State License)' 'MSA' 'Regional PPO'
 'Medicare-Medicaid Plan HMO/HMOPOS']
Any Part D only plans? ['Yes' 'No']
```

[67]:
```python
# Set seaborn style
sns.set(style="whitegrid")

# Plot
plt.figure(figsize=(10, 6))
sns.boxplot(
    data=plan_counts,
    x="year",
    y="num_plans",
    color="#007BFF",  # Bright blue
    width=0.6,
    fliersize=2  # smaller outlier dots
)

# Titles and labels
plt.title("MA Plan Availability by County (2010-2015)", fontsize=14,␣
 ↪weight='bold')
plt.xlabel("Year", fontsize=12)
plt.ylabel("Number of Unique MA Plans per County", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[67], line 7
      4 # Plot
      5 plt.figure(figsize=(10, 6))
      6 sns.boxplot(
----> 7     data=plan_counts,
      8     x="year",
      9     y="num_plans",
     10     color="#007BFF",  # Bright blue
     11     width=0.6,
     12     fliersize=2  # smaller outlier dots
     13 )
     15 # Titles and labels
     16 plt.title("MA Plan Availability by County (2010-2015)", fontsize=14,␣
 ↪weight='bold')

NameError: name 'plan_counts' is not defined
```

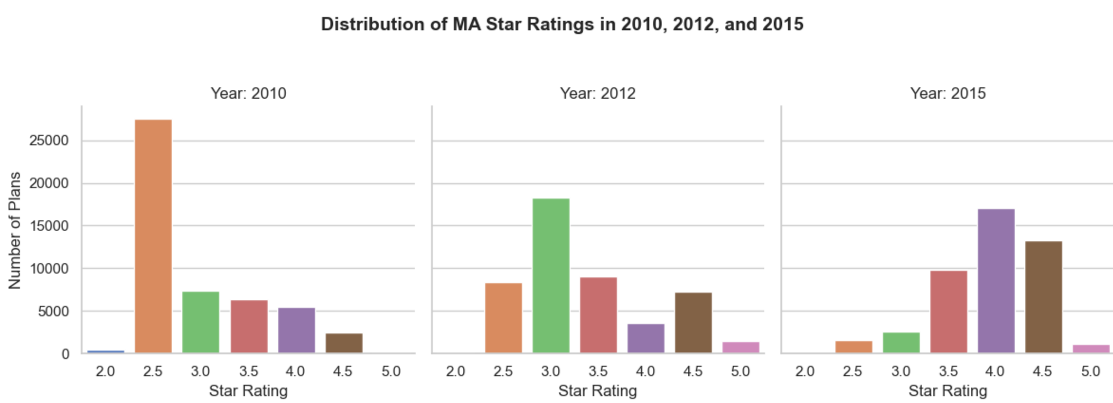
```
<Figure size 1000x600 with 0 Axes>
```

2. Provide bar graphs showing the distribution of star ratings in 2010, 2012, and 2015. How has this distribution changed over time?

[99]:
```
# Filter the data for only 2010, 2012, and 2015
selected_years = [2010, 2012, 2015]
rating_subset = final_data[final_data["year"].isin(selected_years)]

# Drop rows where Star_Rating is missing
rating_subset = rating_subset.dropna(subset=["Star_Rating"])

# Set plot style
sns.set(style="whitegrid")

# Create bar plots by year
g = sns.catplot(
    data=rating_subset,
    x="Star_Rating",
    col="year",
    kind="count",
    col_order=selected_years,
    height=4,
    aspect=1,
    palette="muted"
)
```

```
# Labeling
g.set_axis_labels("Star Rating", "Number of Plans")
g.set_titles("Year: {col_name}")
g.fig.suptitle("Distribution of MA Star Ratings in 2010, 2012, and 2015", y=1.
↪05, fontsize=14, weight='bold')

# Display
plt.tight_layout()
plt.show()
```

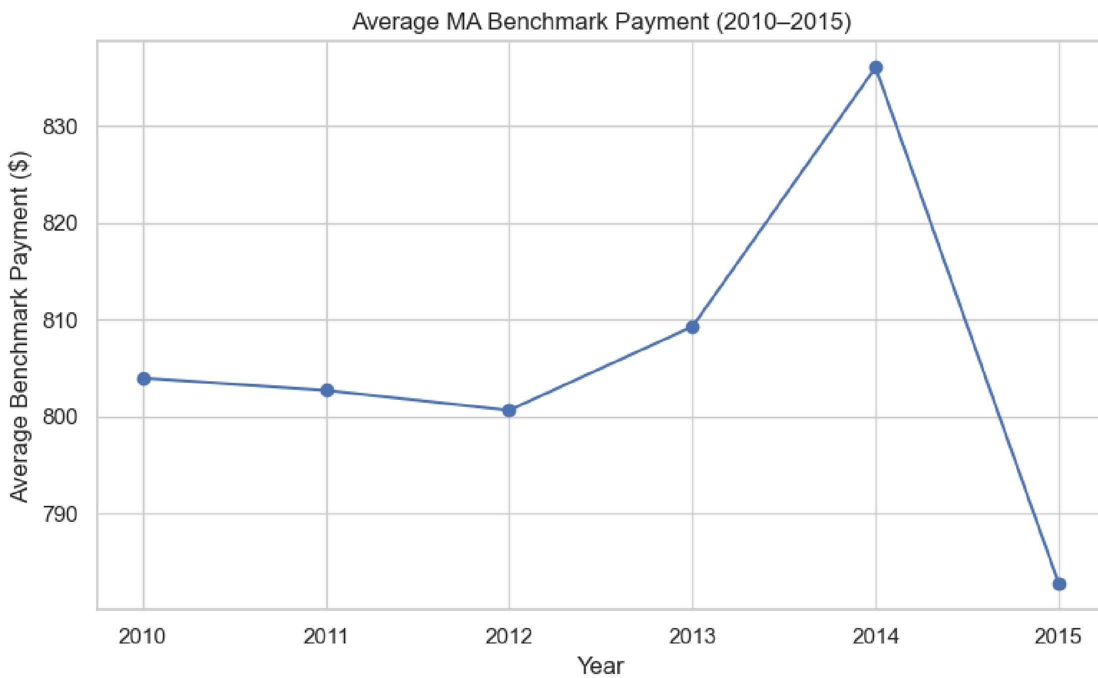
```
/var/folders/8p/wmnjrdd55rx2pn76f5j7m2tw0000gn/T/ipykernel_20352/1622397608.py:1
2: FutureWarning:

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

  g = sns.catplot(
```



3. Plot the average benchmark payment over time from 2010 through 2015. How much has the average benchmark payment risen over the years?

[100]:
```
# Calculate the average benchmark payment by year
benchmark_avg = (
    final_data
    .groupby("year", as_index=False)
    .agg(avg_benchmark=("ma_rate", "mean"))
)

# Plot
plt.figure(figsize=(8, 5))
plt.plot(benchmark_avg["year"], benchmark_avg["avg_benchmark"], marker='o')
plt.title("Average MA Benchmark Payment (2010-2015)")
```

```
plt.xlabel("Year")
plt.ylabel("Average Benchmark Payment ($)")
plt.grid(True)
plt.tight_layout()
plt.show()

# Display average changes
print(benchmark_avg)
print(f"\nIncrease from 2010 to 2015: ${benchmark_avg['avg_benchmark'].iloc[-1]
↪- benchmark_avg['avg_benchmark'].iloc[0]:.2f}")
```



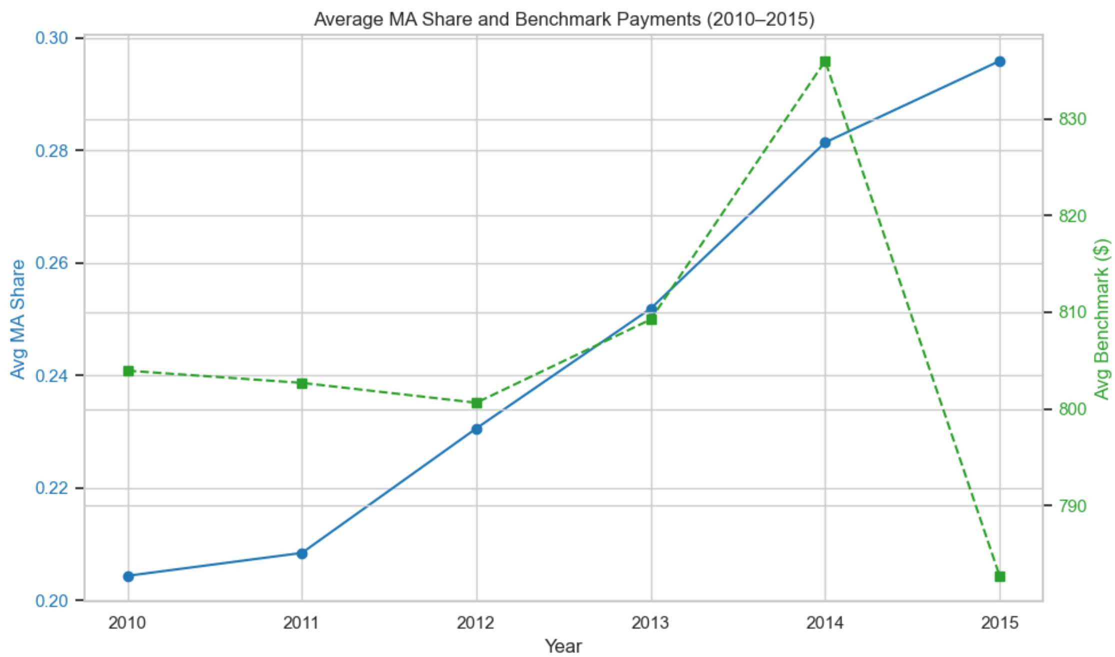

```
   year  avg_benchmark
0  2010     803.948611
1  2011     802.684419
2  2012     800.626973
3  2013     809.254803
4  2014     836.013611
5  2015     782.711741

Increase from 2010 to 2015: $-21.24
```

Over the years, the payment amounts have remained fairly consistent, although there was a noticeable peak in 2014 followed by a decline in 2015. From 2012 to 2014, benchmark payments included added incentives for quality improvement. However, these incentives were modified in 2015, which aligns with the rise in payments through 2014 and the drop that followed.

4. Plot the average share of Medicare Advantage (relative to all Medicare eligibles) over time from 2010 through 2015. Has Medicare Advantage increased or decreased in popularity? How does this share correlate with benchmark payments?

[101]:
```
# Step 1: Create MA share in the penetration data
ma_penetration_data["ma_share"] = (
    ma_penetration_data["avg_enrolled"] / ma_penetration_data["avg_eligibles"]
)

# Step 2: Merge the MA share into final_data
final_data = final_data.merge(
    ma_penetration_data[["fips", "year", "ma_share"]],
    on=["fips", "year"],
    how="left"
)
```

[102]:
```
# Group and compute averages
penetration_trend = (
    final_data
    .groupby("year", as_index=False)
    .agg(
        avg_ma_share=("ma_share", "mean"),
        avg_benchmark=("ma_rate", "mean")
    )
)

fig, ax1 = plt.subplots(figsize=(10, 6))
color = 'tab:blue'
ax1.set_xlabel("Year")
ax1.set_ylabel("Avg MA Share", color=color)
ax1.plot(penetration_trend["year"], penetration_trend["avg_ma_share"],␣
↪marker="o", color=color)
ax1.tick_params(axis='y', labelcolor=color)
ax2 = ax1.twinx()  # second y-axis
color = 'tab:green'
ax2.set_ylabel("Avg Benchmark ($)", color=color)
ax2.plot(penetration_trend["year"], penetration_trend["avg_benchmark"],␣
↪marker="s", linestyle="--", color=color)
ax2.tick_params(axis='y', labelcolor=color)
plt.title("Average MA Share and Benchmark Payments (2010-2015)")
fig.tight_layout()
plt.show()
```



0.0.2 Estimate ATEs

For the rest of the assignment, we'll use a regression discontinuity design to estimate the average treatment effect from receiving a marginally higher rating. We'll focus only on 2010.

5. Calculate the running variable underlying the star rating. Provide a table showing the number of plans that are rounded up into a 3-star, 3.5-star, 4-star, 4.5-star, and 5-star rating.

[57]:
```
final_data = pd.read_csv("/Users/kathrynmawhinney/Documents/GitHub/Homework4/
 ↪data/output/final_ma_data.csv")
data_2010 = final_data[final_data["year"] == 2010].copy()

columns_to_average = [
    "breastcancer_screen", "rectalcancer_screen", "cv_cholscreen",␣
 ↪"diabetes_cholscreen",
    "monitoring", "flu_vaccine", "pn_vaccine", "physical_health",␣
 ↪"mental_health",
    "osteo_test", "physical_monitor", "primaryaccess", "hospital_followup",
    "depression_followup", "nodelays", "carequickly", "readmissions",␣
 ↪"coordination",
    "osteo_manage", "diabetes_eye", "diabetes_kidney", "diabetes_bloodsugar",
    "diabetes_chol", "copd_test", "bloodpressure", "ra_manage",
    "betablocker", "appeals_timely", "bladder", "falling"
]

# Subset data to 2010 and clean
```

```
data_2010 = (
    data_2010
    .query("year == 2010 and avg_enrollment.notna() and partc_score.notna()")
    .assign(
        # Calculate raw_rating as the average of selected columns
        raw_rating=lambda df: df[columns_to_average].mean(axis=1, skipna=True),
        mkt_share=lambda df: df["avg_enrollment"] / df["avg_eligibles"],
        HMO=lambda df: df["plan_type"].str.contains("HMO|HMO eligibles", na=False),
    )
    .loc[
    :,
    [
        "contractid", "planid", "fips", "avg_enrollment", "state", "county",
        "raw_rating", "partc_score", "avg_eligibles", "avg_enrolled",
        "risk_ab", "Star_Rating", "ma_rate",
        "plan_type", "partd", "mkt_share", "HMO"
    ]
]
)

ma_rounded = (
    data_2010
    .assign(
        rounded_30=lambda df: ((df["raw_rating"] >= 2.75) & (df["raw_rating"] < 3.00) & (df["Star_Rating"] == 3.0)).astype(int),
        rounded_35=lambda df: ((df["raw_rating"] >= 3.25) & (df["raw_rating"] < 3.50) & (df["Star_Rating"] == 3.5)).astype(int),
        rounded_40=lambda df: ((df["raw_rating"] >= 3.75) & (df["raw_rating"] < 4.00) & (df["Star_Rating"] == 4.0)).astype(int),
        rounded_45=lambda df: ((df["raw_rating"] >= 4.25) & (df["raw_rating"] < 4.50) & (df["Star_Rating"] == 4.5)).astype(int),
        rounded_50=lambda df: ((df["raw_rating"] >= 4.75) & (df["raw_rating"] < 5.00) & (df["Star_Rating"] == 5.0)).astype(int),
    )
    .query("Star_Rating in [3.0, 3.5, 4.0, 4.5, 5.0]")
    .groupby("Star_Rating", as_index=False)
    .agg({
        "rounded_30": "sum",
        "rounded_35": "sum",
        "rounded_40": "sum",
        "rounded_45": "sum",
        "rounded_50": "sum",
    })
    .assign(
        rounded=lambda df: df[
```

```
            ["rounded_30", "rounded_35", "rounded_40", "rounded_45",␣
↪"rounded_50"]
        ].sum(axis=1)
    )
    .loc[:, ["Star_Rating", "rounded"]]
)

# Display the table
from IPython.display import Markdown, display
display(Markdown(ma_rounded.to_markdown()))
```

```
/var/folders/8p/wmnjrdd55rx2pn76f5j7m2tw0000gn/T/ipykernel_45977/4205015293.py:1
: DtypeWarning: Columns (68,98,99,100,101,102,103,104,105,106) have mixed types.
Specify dtype option on import or set low_memory=False.
  final_data = pd.read_csv("/Users/kathrynmawhinney/Documents/GitHub/Homework4/d
ata/output/final_ma_data.csv")
```

	Star_Rating	rounded
0	3	1297
1	3.5	1252
2	4	645
3	4.5	24
4	5	0

6. Using the RD estimator with a bandwidth of 0.125, provide an estimate of the effect of receiving a 3-star versus a 2.5 star rating on enrollments. Repeat the exercise to estimate the effects at 3.5 stars, and summarize your results in a table.

[80]:
```
import statsmodels.formula.api as smf
data_2010 = final_data[final_data["year"] == 2010].copy()

# Subset data to 2010 and clean
data_2010 = (
    data_2010
    .query("year == 2010 and avg_enrollment.notna() and partc_score.notna()")
    .assign(
        # Calculate raw_rating as the average of selected columns
        raw_rating=lambda df: df[columns_to_average].mean(axis=1, skipna=True),
        mkt_share=lambda df: df["avg_enrollment"] / df["avg_eligibles"],
        HMO=lambda df: df["plan_type"].str.contains("HMO|HMO eligibles",␣
↪na=False),
    )
    .loc[
    :,
    [
        "contractid", "planid", "fips", "avg_enrollment", "state", "county",
        "raw_rating", "partc_score", "avg_eligibles", "avg_enrolled",
```

```
        "risk_ab", "Star_Rating", "ma_rate",
        "plan_type", "partd", "mkt_share", "HMO"
    ]
]
)


# Star 3.0 vs 2.5 (cutoff at 2.75)
df_30 = (
    data_2010[
        (data_2010["raw_rating"] >= (2.75 - 0.125)) &
        (data_2010["raw_rating"] <= (2.75 + 0.125)) &
        (data_2010["Star_Rating"].isin([2.5, 3.0]))
    ]
    .copy()
)

df_30["treat"] = (df_30["Star_Rating"] == 3.0).astype(int)
df_30["score"] = df_30["raw_rating"] - 2.75

star30 = smf.ols("mkt_share ~ treat + score", data=df_30).fit()

# Star 3.5 vs 3.0 (cutoff at 3.25)
df_35 = (
    data_2010[
        (data_2010["raw_rating"] >= (3.25 - 0.125)) &
        (data_2010["raw_rating"] <= (3.25 + 0.125)) &
        (data_2010["Star_Rating"].isin([3.0, 3.5]))
    ]
    .copy()
)

df_35["treat"] = (df_35["Star_Rating"] == 3.5).astype(int)
df_35["score"] = df_35["raw_rating"] - 3.25

star30 = smf.ols("mkt_share ~ treat + score", data=df_30).fit()
star35 = smf.ols("mkt_share ~ treat + score", data=df_35).fit()
```

[81]:
```
import pandas as pd
from IPython.display import Markdown, display

# Build results table manually
results_data = {
    "3 vs 2.5 Stars": [star30.params['Intercept'], star30.params['score'],␣
 ↪star30.params['treat'],
                       int(star30.nobs), round(star30.rsquared, 3)],
```

```
    "3.5 vs 3 Stars": [star35.params['Intercept'], star35.params['score'],␣
↪star35.params['treat'],
                       int(star35.nobs), round(star35.rsquared, 3)]
}

results_table = pd.DataFrame(results_data,
                             index=['Intercept', 'Running Score', 'Treatment',␣
↪'N', 'R²'])

# Display table
display(Markdown(results_table.to_markdown()))
```

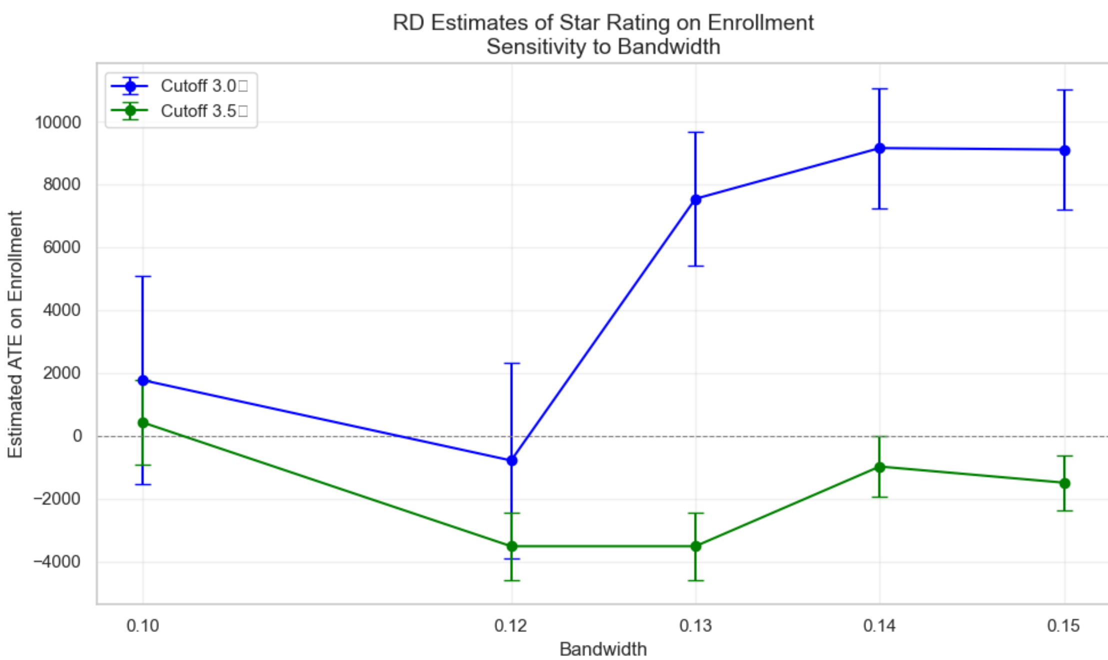
	3 vs 2.5 Stars	3.5 vs 3 Stars
Intercept	0.00999412	0.00874213
Running Score	-0.00124801	-0.0665126
Treatment	0.00562603	0.0115924
N	4383	1517
R^2	0.022	0.032

7. Repeat your results for bandwidhts of 0.1, 0.12, 0.13, 0.14, and 0.15 (again for 3 and 3.5 stars). Show all of the results in a graph. How sensitive are your findings to the choice of bandwidth?

[82]:
```
bandwidths = [0.1, 0.12, 0.13, 0.14, 0.15]
results = []

for bw in bandwidths:
    for cutoff in [3.0, 3.5]:
        ate, se = rd_estimate(data_2010, cutoff=cutoff, bandwidth=bw)
        results.append({
            "Cutoff": cutoff,
            "Bandwidth": bw,
            "ATE": ate,
            "SE": se
        })

# Convert to DataFrame
rd_results = pd.DataFrame(results)

# Plot setup
fig, ax = plt.subplots(figsize=(10, 6))
for cutoff, color in zip([3.0, 3.5], ['blue', 'green']):
    subset = rd_results[rd_results["Cutoff"] == cutoff]
    ax.errorbar(subset["Bandwidth"], subset["ATE"], yerr=subset["SE"],
                label=f'Cutoff {cutoff} ', marker='o', linestyle='-',
                color=color, capsize=5)
```

```
# Styling
ax.set_title("RD Estimates of Star Rating on Enrollment\nSensitivity to␣
 ↪Bandwidth", fontsize=14)
ax.set_xlabel("Bandwidth")
ax.set_ylabel("Estimated ATE on Enrollment")
ax.axhline(0, color='gray', linestyle='--', linewidth=0.8)
ax.legend()
plt.xticks(bandwidths)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

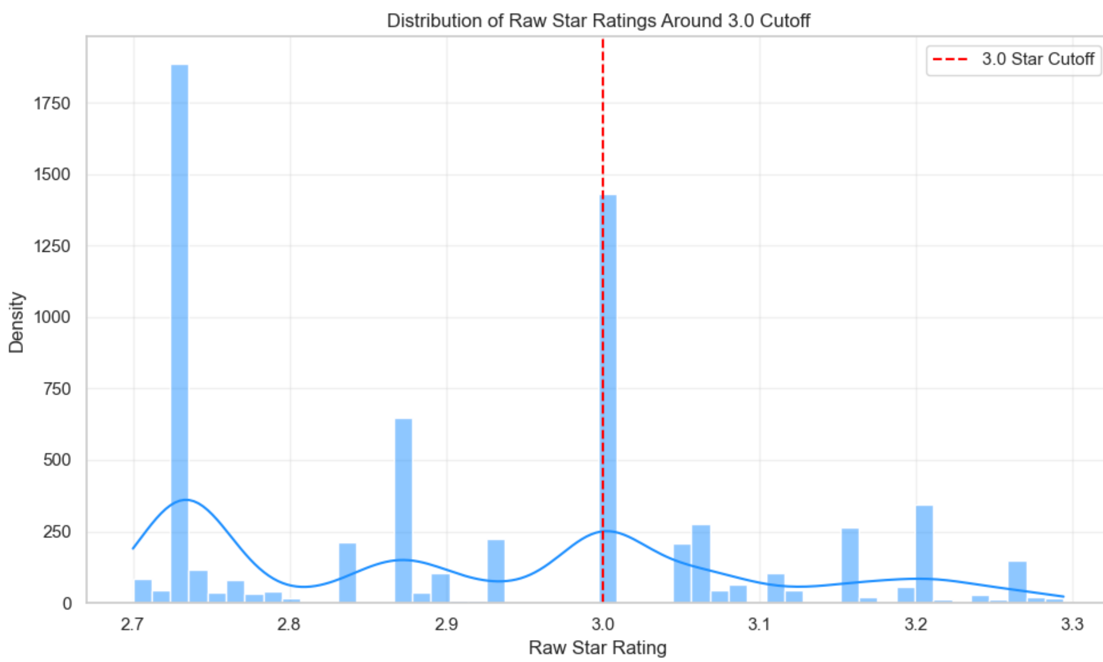
```
/var/folders/8p/wmnjrdd55rx2pn76f5j7m2tw0000gn/T/ipykernel_45977/920963729.py:33
: UserWarning: Glyph 9733 (\N{BLACK STAR}) missing from font(s) Arial.
  plt.tight_layout()
/opt/anaconda3/lib/python3.12/site-packages/IPython/core/pylabtools.py:170:
UserWarning: Glyph 9733 (\N{BLACK STAR}) missing from font(s) Arial.
  fig.canvas.print_figure(bytes_io, **kw)
```



8. Examine (graphically) whether contracts appear to manipulate the running variable. In other words, look at the distribution of the running variable before and after the relevent threshold values. What do you find?

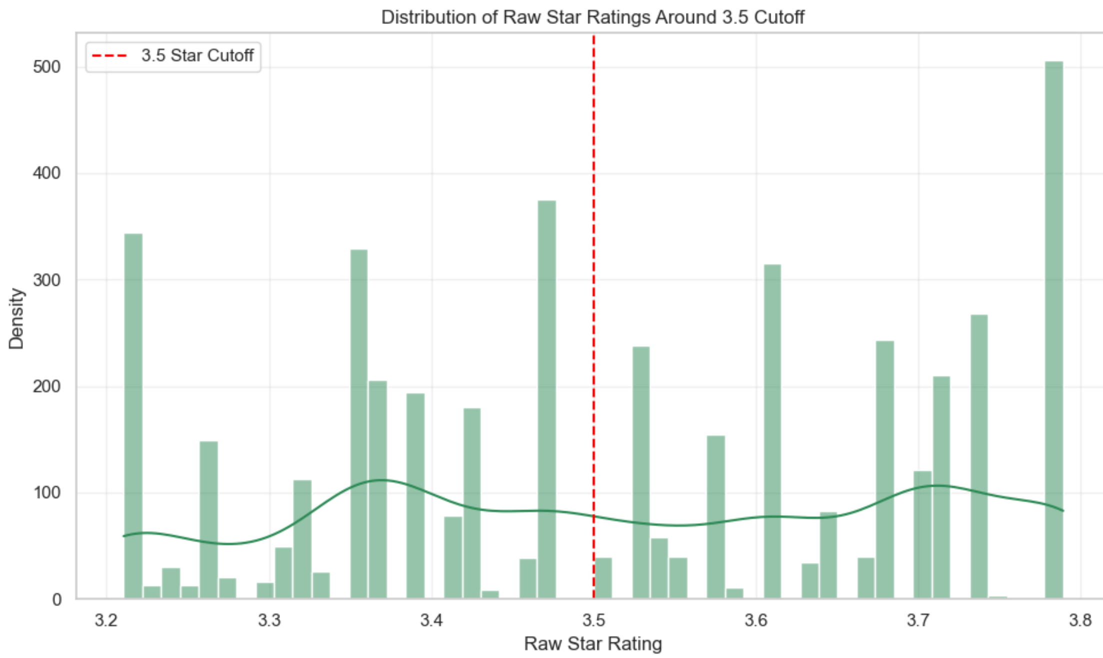
[83]:
```
# Filter data around the 3.0 cutoff
cutoff = 3.0
```

```
bandwidth = 0.3
subset = data_2010[(data_2010["raw_rating"] >= cutoff - bandwidth) &
↪(data_2010["raw_rating"] <= cutoff + bandwidth)]

# Plot
plt.figure(figsize=(10, 6))
sns.histplot(subset["raw_rating"], bins=50, kde=True, color='dodgerblue',
↪edgecolor='white')
plt.axvline(x=cutoff, color='red', linestyle='--', label='3.0 Star Cutoff')

plt.title("Distribution of Raw Star Ratings Around 3.0 Cutoff")
plt.xlabel("Raw Star Rating")
plt.ylabel("Density")
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



[84]:
```
# Filter data around the 3.5 cutoff
cutoff = 3.5
subset = data_2010[(data_2010["raw_rating"] >= cutoff - bandwidth) &
↪(data_2010["raw_rating"] <= cutoff + bandwidth)]

# Plot
plt.figure(figsize=(10, 6))
```

```
sns.histplot(subset["raw_rating"], bins=50, kde=True, color='seagreen',␣
    ↪edgecolor='white')
plt.axvline(x=cutoff, color='red', linestyle='--', label='3.5 Star Cutoff')

plt.title("Distribution of Raw Star Ratings Around 3.5 Cutoff")
plt.xlabel("Raw Star Rating")
plt.ylabel("Density")
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



9. Similar to question 4, examine whether plans just above the threshold values have different characteristics than contracts just below the threshold values. Use HMO and Part D status as your plan characteristics.

[85]:
```
# Focus on 2010 and create raw_rating if not already there
data_2010 = final_data[final_data["year"] == 2010].copy()
data_2010["raw_rating"] = data_2010["Star_Rating"]

# Function to summarize covariates above/below RD cutoff
def covariate_check(data, cutoff, bandwidth):
    subset = data[(data["raw_rating"] >= cutoff - bandwidth) &␣
    ↪(data["raw_rating"] <= cutoff + bandwidth)].copy()
    subset["above"] = (subset["raw_rating"] >= cutoff).astype(int)
```

```
    summary = subset.groupby("above").agg(
        share_hmo=("plan_type", lambda x: (x == "HMO").mean()),
        share_partd=("partd", lambda x: (x == "Yes").mean()),
        n_plans=("contractid", "count")
    ).reset_index()

    summary["group"] = summary["above"].map({0: f"Below {cutoff}", 1: f"Above␣
 ↪{cutoff}"})
    return summary[["group", "share_hmo", "share_partd", "n_plans"]]

# Run for both thresholds
cov_3 = covariate_check(data_2010, cutoff=3.0, bandwidth=0.125)
cov_35 = covariate_check(data_2010, cutoff=3.5, bandwidth=0.125)

# Combine results
covariate_table = pd.concat([cov_3, cov_35], ignore_index=True)
covariate_table.columns = ["Group", "Share HMO", "Share Part D", "Number of␣
 ↪Plans"]

# Display the final table
print("\nCovariate Balance Around RD Thresholds (Bandwidth = 0.125)")
display(covariate_table)
```

```
Covariate Balance Around RD Thresholds (Bandwidth = 0.125)
```

	Group	Share HMO	Share Part D	Number of Plans
0	Above 3.0	0.0	0.893651	7419
1	Above 3.5	0.0	0.853317	6347

10. Summarize your findings from 5-9. What is the effect of increasing a star rating on enrollments? Briefly explain your results.