ECONOMIC_SANCTION_PROJECT DISCRIPTION

Globally Economic Sanctions is generally regarded as an important foreign policy used to punish regimes, persons and groups who actions deviate the globally accepted norms and values. As such powerful states, multilateral cooperation, and institutions in response to such situations imposes sanctions on such individuals, state and organizations. In this project I evaluate the effect of economic sanctions on Russia foreign direct investment. Using data from World Trade Organization (WTO), World Bank, and Global Sanction Database.

To estimate this effect, regression analysis was performed on obtained data. The data was divided into independent/exposure (binary) variables (Financial sanction containing treatment and control group) and the outcome/dependent variables (continuous) (FDI, Import and Export). Using inflation, GDP, and exchange rate as covariates variables. The Propensity Score was estimated which is the probabilities of receiving the treatment (Financial sanction) given the observed covariates (inflation, GDP, and exchange rate). Subsequently coarsened exact matching (CEM). Performed on the propensity scores. The CEM ensures a balance in the distribution of covariates between treated and control groups based on the already calculated the propensity scores by discretizing them into categories. After chi-square tests analysis was run to assess balance between the treated and control groups. Finally regression analysis was conducted to estimate the causal effect of the treatment variable (Financial sanction) on the outcome variables while controlling for the covariates.

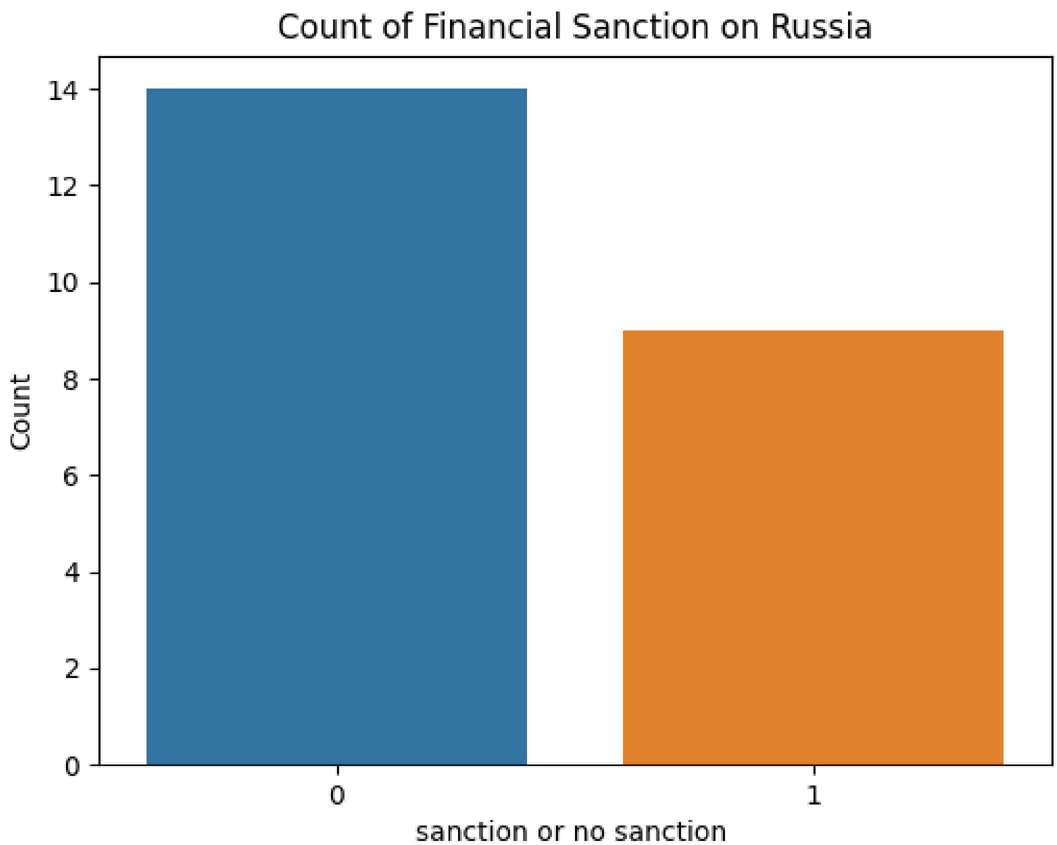
```
TERMINAL

   Country  Year        Export  Log_exp        Import  ...  Log_inflation  Exchange_rate  Log_exchange  GDP_per_capita    Log_GDP
0   Russia  2000  1.050000e+11     11.0  4.486200e+10  ...       1.318037      28.129167      1.449157    2.600000e+11  11.414973
1   Russia  2001  1.020000e+11     11.0  5.376400e+10  ...       1.331974      29.168525      1.464914    3.070000e+11  11.487138
2   Russia  2002  1.070000e+11     11.0  6.096600e+10  ...       1.198347      31.348483      1.496217    3.450000e+11  11.537819
3   Russia  2003  1.360000e+11     11.1  7.607000e+10  ...       1.135555      30.692025      1.487026    4.300000e+11  11.633468
4   Russia  2004  1.830000e+11     11.3  9.738200e+10  ...       1.036973      28.813742      1.459600    5.910000e+11  11.771587
5   Russia  2005  2.440000e+11     11.4  1.250000e+11  ...       1.103301      28.284442      1.451548    7.640000e+11  11.883093
6   Russia  2006  3.040000e+11     11.5  1.640000e+11  ...       0.985366      27.190958      1.434425    9.900000e+11  11.995635
7   Russia  2007  3.540000e+11     11.5  2.230000e+11  ...       0.954595      25.580845      1.407915    1.300000e+12  12.113943
8   Russia  2008  4.720000e+11     11.7  2.920000e+11  ...       1.149551      24.852875      1.395377    1.660000e+12  12.220108
9   Russia  2009  3.030000e+11     11.5  1.920000e+11  ...       1.066226      31.740358      1.501612    1.220000e+12  12.086360
10  Russia  2010  4.010000e+11     11.6  2.490000e+11  ...       0.835652      30.367915      1.482415    1.520000e+12  12.181844
11  Russia  2011  5.220000e+11     11.7  3.240000e+11  ...       0.926366      29.382341      1.468086    2.050000e+12  12.311754
12  Russia  2012  5.290000e+11     11.7  3.350000e+11  ...       0.705414      30.839831      1.489112    2.210000e+12  12.344392
13  Russia  2013  5.220000e+11     11.7  3.410000e+11  ...       0.829542      31.837144      1.502934    2.290000e+12  12.359835
14  Russia  2014  4.970000e+11     11.7  3.080000e+11  ...       0.893396      38.378207      1.584085    2.060000e+12  12.313867
15  Russia  2015  3.410000e+11     11.5  1.930000e+11  ...       1.191295      60.937650      1.784886    1.360000e+12  12.133539
16  Russia  2016  2.820000e+11     11.5  1.910000e+11  ...       0.847724      67.055933      1.826437    1.280000e+12  12.107210
17  Russia  2017  3.530000e+11     11.5  2.380000e+11  ...       0.566241      58.342801      1.765987    1.570000e+12  12.195900
18  Russia  2018  4.440000e+11     11.6  2.490000e+11  ...       0.459136      62.668133      1.797047    1.660000e+12  12.220108
19  Russia  2019  4.200000e+11     11.6  2.540000e+11  ...       0.650343      64.737658      1.811157    1.690000e+12  12.227887
20  Russia  2020  3.330000e+11     11.5  2.400000e+11  ...       0.529130      72.104908      1.857965    1.490000e+12  12.173186
21  Russia  2021  4.940000e+11     11.7  3.040000e+11  ...       0.825715      73.654350      1.867198    1.840000e+12  12.264818
22  Russia  2022  5.320000e+11     11.7  2.400000e+11  ...       1.138934      68.484942      1.835595    2.240000e+12  12.350248

[23 rows x 21 columns]
   Country  Year  Financial_sanction    Log_GDP  Log_inflation  Log_exchange    Log_FDI    Log_imp  Log_exp
0   Russia  2000                   0  11.414973       1.318037      1.449157   9.427815  10.651879     11.0
1   Russia  2001                   0  11.487138       1.331974      1.464914   9.454433  10.730492     11.0
2   Russia  2002                   0  11.537819       1.198347      1.496217   9.540809  10.785088     11.0
3   Russia  2003                   0  11.633468       1.135555      1.487026   9.899198  10.881213     11.1
4   Russia  2004                   0  11.771587       1.036973      1.459600  10.187605  10.988479     11.3
5   Russia  2005                   0  11.883093       1.103301      1.451548  10.190557  11.096910     11.4
6   Russia  2006                   0  11.995635       0.985366      1.434425  10.575127  11.214844     11.5
7   Russia  2007                   0  12.113943       0.954595      1.407915  10.747207  11.348305     11.5
8   Russia  2008                   0  12.220108       1.149551      1.395377  10.873802  11.465383     11.7
9   Russia  2009                   0  12.086360       1.066226      1.501612  10.563280  11.283301     11.5
10  Russia  2010                   0  12.181844       0.835652      1.482415  10.635160  11.396199     11.6
```

Columns to analyse in the dataset

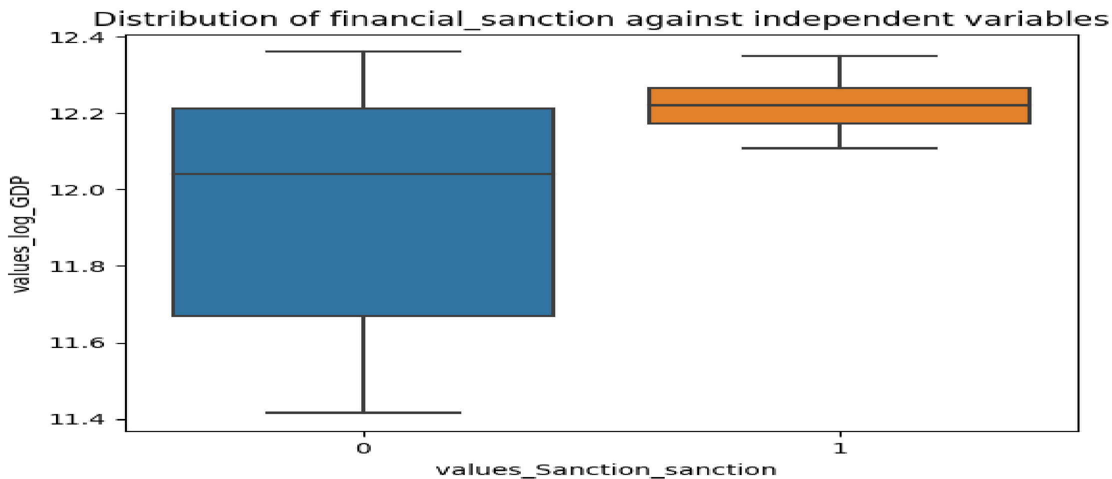
Check for missing values

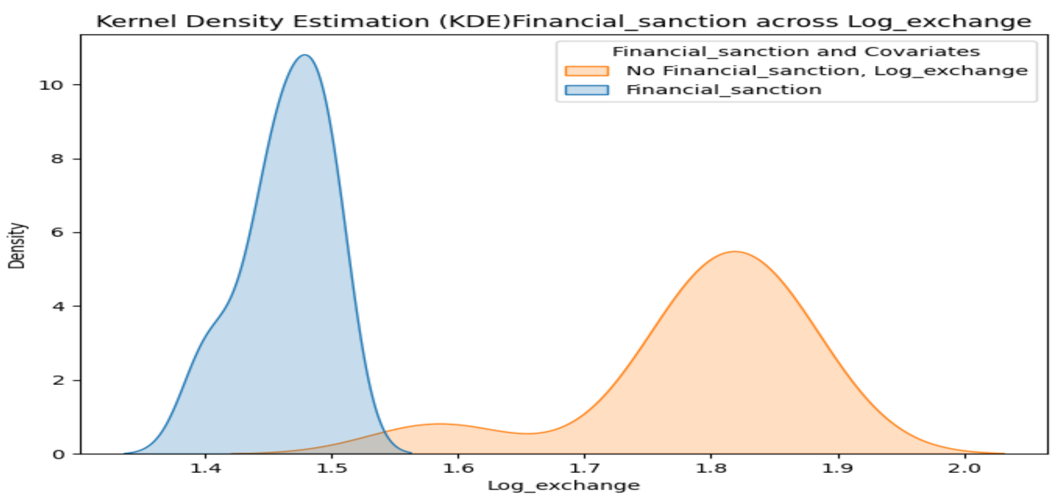
```
    Country   Year  Export  Log_exp  Import  ...  Log_inflation  Exchange_rate  Log_exchange  GDP_per_capita  Log_GDP
0     False  False   False    False   False  ...          False          False         False           False    False
1     False  False   False    False   False  ...          False          False         False           False    False
2     False  False   False    False   False  ...          False          False         False           False    False
3     False  False   False    False   False  ...          False          False         False           False    False
4     False  False   False    False   False  ...          False          False         False           False    False
5     False  False   False    False   False  ...          False          False         False           False    False
6     False  False   False    False   False  ...          False          False         False           False    False
7     False  False   False    False   False  ...          False          False         False           False    False
8     False  False   False    False   False  ...          False          False         False           False    False
9     False  False   False    False   False  ...          False          False         False           False    False
10    False  False   False    False   False  ...          False          False         False           False    False
11    False  False   False    False   False  ...          False          False         False           False    False
12    False  False   False    False   False  ...          False          False         False           False    False
13    False  False   False    False   False  ...          False          False         False           False    False
14    False  False   False    False   False  ...          False          False         False           False    False
15    False  False   False    False   False  ...          False          False         False           False    False
16    False  False   False    False   False  ...          False          False         False           False    False
17    False  False   False    False   False  ...          False          False         False           False    False
18    False  False   False    False   False  ...          False          False         False           False    False
19    False  False   False    False   False  ...          False          False         False           False    False
20    False  False   False    False   False  ...          False          False         False           False    False
21    False  False   False    False   False  ...          False          False         False           False    False
20    False  False   False    False   False  ...          False          False         False           False    False
21    False  False   False    False   False  ...          False          False         False           False    False
22    False  False   False    False   False  ...          False          False         False           False    False
```

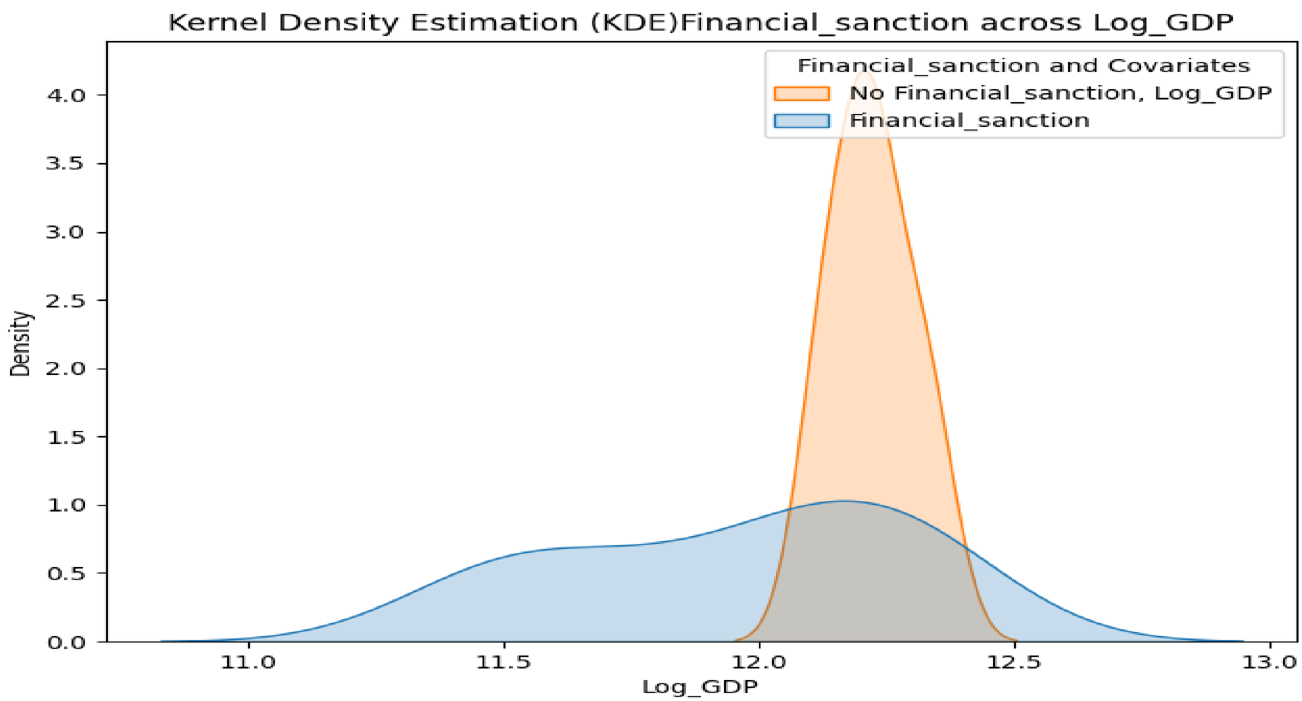
Sanction or no sanction

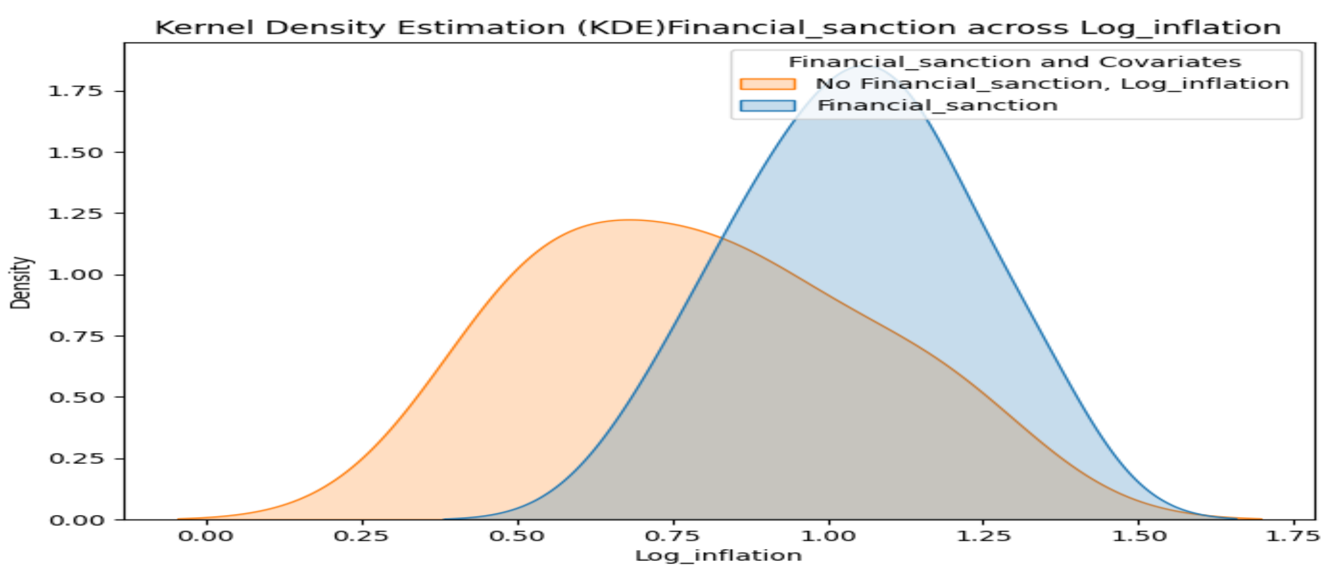


The figure above shows count of Financial sanction in the data using outcome=sns.countplot(x="Financial_sanction", data=df_cleaned) function.


Distribution of financial_sanction against independent variables
values_log_GDP
12.4
12.2
12.0
11.8
11.6
11.4
0
1
values_Sanction_sanction


Kernel Density Estimation (KDE)Financial_sanction across Log_exchange
Financial_sanction and Covariates
No Financial_sanction, Log_exchange
Financial_sanction
Density
10
8
6
4
2
0
1.4
1.5
1.6
1.7
1.8
1.9
2.0
Log_exchange


Kernel Density Estimation (KDE)Financial_sanction across Log_GDP
Financial_sanction and Covariates
No Financial_sanction, Log_GDP
Financial_sanction
Density
Log_GDP
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
11.0
11.5
12.0
12.5
13.0


Kernel Density Estimation (KDE)Financial_sanction across Log_inflation
Financial_sanction and Covariates
No Financial_sanction, Log_inflation
Financial_sanction
Density
Log_inflation
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

The screenshot below is a result obtained from a propensity test score analysis. The scores shows the probability of each observation in the dataset (FDI) receiving the treatment (Financial sanction) based on the various covariates.

	0
0	4.06181e-07
1	4.06181e-07
2	4.06181e-07
3	4.06181e-07
4	4.06181e-07
5	4.06181e-07
6	4.06181e-07
7	4.06181e-07
8	4.06181e-07
9	4.06181e-07
10	4.06181e-07
11	4.06181e-07
12	4.06181e-07
13	4.06181e-07
14	1
15	1
16	1

A screenshot from the matched data of treatment variable/exposure (Financial_sanction and covariates)

	Financial_sanction	Log_GDP	Log_inflation	Log_exchange	Log_FDI
0	0	Low	High	Low	9.427815438
1	0	Low	High	Low	9.454433228
2	0	Low	High	Low	9.540808561
3	0	Low	High	Low	9.899198151
4	0	Medium	Medium	Low	10.18760503
5	0	Medium	High	Low	10.19055719
6	0	Medium	Medium	Low	10.57512743
7	0	High	Medium	Low	10.74720728
8	0	High	High	Low	10.87380236
9	0	High	High	Low	10.5632805
10	0	High	Medium	Low	10.63515971
11	0	High	Medium	Low	10.74102255
12	0	High	Low	Low	10.70404373
13	0	High	Medium	Low	10.84022463
14	1	High	Medium	Medium	10.34304091
15	1	High	High	High	9.835878831
16	1	High	Medium	High	10.51240287
17	1	High	Low	High	10.45571927
18	1	High	Low	High	9.94373435
19	1	High	Low	High	10.50480743
20	1	High	Low	High	9.976753818
21	1	High	Medium	High	10.60691842
22	1	High	High	High	10.6

Perform chi-square test for Log_inflation, log_gdp, log_exchange (Covariates)

Inflation		GDP		Exchange	
P-value	Chi-square stat	P-value	Chi-square stat	P-value	Chi-square stat
0.00001013	23.000000	0.000010130	23.00000000	0.0000101	23.00000

After performing the Coarsen Exact Matching a chi-square test was run in other to evaluate the balance between the matched data. From the table above since the P-Values are less than the significance alpha (α), value of 0.05 hence we can conclude that there is an association between the matched covariates.

	coef	std err	t	P>|t|	[0.025	0.975]
Financial_sanction	-0.4384	0.424	-1.034	0.315	-1.329	0.452
Log_GDP	1.6008	0.202	7.923	0.000	1.176	2.025
Log_inflation	0.1119	0.411	0.272	0.788	-0.751	0.975
Log_exchange	0.1033	1.199	0.086	0.932	-2.415	2.621
const	-9.0892	3.360	-2.705	0.014	-16.149	-2.030

The screenshot above shows result from a regression analysis of the effect of financial sanction on export.

	coef	std err	t	P>|t|	[0.025	0.975]
Financial_sanction	0.0181	0.052	0.345	0.734	-0.092	0.128
Log_GDP	0.8602	0.046	18.893	0.000	0.765	0.956
Log_inflation	0.0612	0.057	1.071	0.298	-0.059	0.181
Log_exchange	-0.1559	0.176	-0.884	0.388	-0.526	0.214
const	1.2892	0.694	1.857	0.080	-0.169	2.748

The screenshot above shows result from a regression analysis of the effect of financial sanction on export.

	coef	std err	t	P>|t|	[0.025	0.975]
Financial_sanction	0.0042	0.042	0.100	0.921	-0.085	0.093
Log_GDP	0.8916	0.060	14.913	0.000	0.766	1.017
Log_inflation	-0.1009	0.096	-1.047	0.309	-0.303	0.102
Log_exchange	-0.1654	0.197	-0.840	0.412	-0.579	0.248
const	0.8614	1.030	0.836	0.414	-1.302	3.025

The screenshot above shows result from a regression analysis of the effect of financial sanction on import.

CODE LINE FOR THE ANALYSIS

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import statsmodels.api as sm
from causalinference import CausalModel
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from scipy.stats import ttest_ind
from statsmodels.stats.diagnostic import het_white
from reportlab.lib.pagesizes import letter, landscape
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
from reportlab.lib import colors
from fpdf import FPDF

from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import letter
from tabulate import tabulate

df=pd.read_csv(r"C:\Users\isrea\OneDrive\Desktop\python
regression\CSV's\RUSSIA_DATA_CSV.csv")
print(df)
print(df.isnull())

### Data_Wrangling
print(df.isnull().sum())
plt.show()
null_values= sns.heatmap(df.isnull(), yticklabels=False)
print(null_values)
plt.show()
null_values_1= sns.heatmap(df.isnull(), yticklabels=False, cmap='viridis')
print(null_values_1)
plt.show()

# Create a copy of the DataFrame for cleaning
df_cleaned = df.copy()

# Check and print the count of NaN values in each colum
```

```
print(df_cleaned.isnull().sum())

print(df_cleaned)
#column in concern

# Define the columns to use
columns_to_analyse_df = [
'Country',  'Year',   'Financial_sanction',    'Log_GDP',     'Log_inflation',
   'Log_exchange',    'Log_FDI',       'Log_imp',     'Log_exp']

# Create the new DataFrame with selected columns
column_to_use = df_cleaned[columns_to_analyse_df].copy()

# Display the new DataFrame
print(column_to_use)


column_to_analyse =pd.DataFrame(column_to_use)

df_cleaned_df = pd.DataFrame(df_cleaned)
column_to_analyse.to_csv('column_to_analyse_df.txt', sep='\t', index=False)

print(column_to_analyse)
# Save the DataFrame to a text file
df_cleaned_df.to_csv('df_cleaned_df.txt', sep='\t', index=False)

# Drop rows with NaN values (this will modify df_cleaned in place)
df_cleaned.dropna(inplace=True)
print(df_cleaned.isnull())
# Create a heatmap to visualize NaN values
sns.heatmap(df_cleaned.isnull(), yticklabels=False, cbar=False)

plt.show()
# Create a countplot for the "Financial_sanction" column
outcome= sns.countplot(x="Financial_sanction", data=df_cleaned)
# Set labels and title
plt.xlabel("sanction or no sanction ")
plt.ylabel("Count")
plt.title("Count of Financial Sanction on Russia")

# Show the plot
plt.show()

# distribution of financial sanction
```

```
effect = sns.boxplot(x='Financial_sanction', y= 'Log_GDP',
data=column_to_analyse)

plt.xlabel(" values_Sanction_sanction")
plt.ylabel("values_log_GDP")
plt.title("Distribution of financial_sanction against independent variables")
print(effect)
plt.show()
# Define the variables

# Define the variables
treatment_variable = 'Financial_sanction'
covariates = ['Log_GDP', 'Log_inflation', 'Log_exchange']

# Plot KDE for each covariate
for covariate in covariates:
    plt.figure(figsize=(8, 6))
    sns.kdeplot(data=df_cleaned, x=covariate, hue=treatment_variable, fill=True,
common_norm=False)
    plt.title(f'Kernel Density Estimation (KDE){treatment_variable} across
{covariate}')
    plt.xlabel(covariate)
    plt.ylabel('Density')

    # Customize legend
    plt.legend(title=f'{treatment_variable} and Covariates',
               labels=[f'No {treatment_variable}, {covariate}',
f'{treatment_variable}', f'{covariate}'])

    plt.show()
#treatment variable and covariates in df
# combined_data = df_cleaned[['Financial_sanction', 'Trade_sanction',
'Military_sanction', 'Multilateral_sender_sanction', 'Log_GDP', 'Log_inflation',
'Log_exchange', 'Log_FDI', 'Log_imp', 'Log_exp']]

# Defined treatment and outcome variables

combined_data = df_cleaned[['Financial_sanction', 'Trade_sanction',
'Military_sanction', 'Multilateral_sender_sanction', 'Log_GDP', 'Log_inflation',
'Log_exchange', 'Log_FDI', 'Log_imp', 'Log_exp']]

# Defined treatment and outcome variables
exposure_column = 'Financial_sanction'
outcome_column = 'Log_FDI'
```

```python
# initializing CausalModel with combined DataFrame
causal = CausalModel(
    Y=combined_data[outcome_column].values,
    D=combined_data[exposure_column].values,
    X=combined_data.drop([outcome_column, exposure_column], axis=1).values)

# Estimate propensity scores
causal.est_propensity_s()
propensity_scores = causal.propensity['fitted']
print(propensity_scores)
#create dataframe for propensity score
propensity_scores_df=pd.DataFrame(propensity_scores)
#tabulation for propensity score
propensity_scores_df_str= tabulate(propensity_scores_df, headers='keys',
tablefmt='grid')
print(propensity_scores_df_str)
# save the tabulated result to a text file
with open('save propensity_score_str.txt', 'w') as f:
    f.write(propensity_scores_df_str)
# Define coarsening schemas for each continuous covariate
coarsening_schemas = {
    'Log_GDP': {'num_bins': 3, 'labels': ['Low', 'Medium', 'High']},
    'Log_inflation': {'num_bins': 3, 'labels': ['Low', 'Medium', 'High']},
    'Log_exchange': {'num_bins': 3, 'labels': ['Low', 'Medium', 'High']},
}

# Initialize variables for matching
matched_data = combined_data.copy()

# Perform matching based on propensity scores and coarsening schemas
for covariate, schema in coarsening_schemas.items():
    bins = pd.cut(combined_data[covariate], bins=schema['num_bins'],
labels=schema['labels'])
    matched_data[covariate] = bins
# Check the results
print(matched_data)
print(matched_data.columns)
# Creating a dataframe to saved the data
matched_data_save= pd.DataFrame(matched_data)
#save the dataframe
save_matched_data_str = tabulate(matched_data_save, headers='keys',
tablefmt='grid')
print(save_matched_data_str)
# save the tabulated result to a text file
with open('save_matched_data_str.txt', 'w') as f:
```

```
    f.write(save_matched_data_str)
from scipy.stats import chi2_contingency

# Create contingency table for Log_inflation, log_gdp, log_exchange
contingency_table = pd.crosstab(column_to_analyse['Financial_sanction'],
matched_data['Log_inflation'])
alpha= 0.5
contingency_table = pd.crosstab(matched_data['Financial_sanction'],
matched_data['Log_GDP'])
contingency_table = pd.crosstab(matched_data['Financial_sanction'],
matched_data['Log_exchange'])
# Perform chi-square test for Log_inflation, log_gdp, log_exchange

chi2, p_value, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-square statistic for Log_inflation: {chi2}")
print(f"P-value for Log_inflation: {p_value}")
# Organize chi-square statistic and p-value into a dictionary
chi2_results = {'Chi-square statistic': chi2, 'P-value': p_value}

# Create a DataFrame from the dictionary
chi2_dataframe = pd.DataFrame(chi2_results, index=[0])

# Save the DataFrame to a text file
with open('chi2_results.txt', 'w') as f:
    f.write(chi2_dataframe.to_string(index=False))
if p_value <= alpha:
    print("Reject the null hypothesis")
    print("There is evidence to suggest an association.")
else:
    print("Fail to reject the null hypothesis")
    print("There isn't enough evidence to suggest an association.")
#other categorical variables (Log_GDP and Log_exchange)

chi2_2, p_value_2, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-square statistic for Log_GDP: {chi2_2}")
print(f"P-value for Log_GDP: {p_value_2}")

# Organize chi-square statistic and p-value into a dictionary
chi2_2_results = {'Chi-square statistic': chi2_2, 'P-value': p_value_2}

# Create a DataFrame from the dictionary
chi2_2_dataframe = pd.DataFrame(chi2_2_results, index=[0])

# Save the DataFrame to a text file
with open('chi2_results.txt', 'w') as f:
```

```
    f.write(chi2_2_dataframe.to_string(index=False))

if p_value_2 <= alpha:
    print("Reject the null hypothesis")
    print("There is evidence to suggest an association.")
else:
    print("Fail to reject the null hypothesis")
    print("There isn't enough evidence to suggest an association.")

chi2_3, p_value_3, dof, expected = chi2_contingency(contingency_table)
print(f"Chi-square statistic for Log_exchange: {chi2_3}")
print(f"P-value for Log_exchange: {p_value_3}")

# Organize chi-square statistic and p-value into a dictionary
chi2_3_results = {'Chi-square statistic': chi2_3, 'P-value': p_value_3}

# Create a DataFrame from the dictionary
chi2_3_dataframe = pd.DataFrame(chi2_3_results, index=[0])

# Save the DataFrame to a text file
with open('chi2_results.txt', 'w') as f:
    f.write(chi2_3_dataframe.to_string(index=False))

if p_value <= alpha:
    print("Reject the null hypothesis")
    print("There is evidence to suggest an association.")
else:
    print("Fail to reject the null hypothesis")
    print("There isn't enough evidence to suggest an association.")

# Convert 'Financial_sanction' to a binary variable (0 or 1)

matched_data['Financial_sanction'] =
matched_data['Financial_sanction'].astype('category').cat.codes
# Define the covariates used in CEM (the coarsened covariates)
coarsened_covariates = ['Log_GDP', 'Log_inflation', 'Log_exchange']

# Create a DataFrame with the matched data
matched_data = df_cleaned[['Financial_sanction'] + coarsened_covariates +
['Log_FDI']]
matched_data_log_imp = df_cleaned[['Financial_sanction'] + coarsened_covariates +
['Log_imp']]
matched_data_log_exp = df_cleaned[['Financial_sanction'] + coarsened_covariates +
['Log_exp']]
```

```python
# Add a constant term to the regression model (intercept)
matched_data = sm.add_constant(matched_data)
matched_data_log_imp = sm.add_constant(matched_data_log_imp)
matched_data_log_exp = sm.add_constant(matched_data_log_exp)
# Perform linear regression
X = matched_data[['Financial_sanction'] + coarsened_covariates + ['const']]
y = matched_data[outcome_column]

model = sm.OLS(y, X).fit(alpha=0.05)

outcome_column_import = 'Log_imp'
X = matched_data_log_imp[['Financial_sanction'] + coarsened_covariates +
['const']]
y = matched_data_log_imp[outcome_column_import]

model_log_imp = sm.OLS(y, X).fit(alpha=0.05)
outcome_column_export = 'Log_exp'
X = matched_data_log_exp[['Financial_sanction'] + coarsened_covariates +
['const']]
y = matched_data_log_exp[outcome_column_export]

model_log_exp = sm.OLS(y, X).fit(alpha=0.05)

results = het_white(model.resid, X)
print("White test p-value for the main model:", results[1])
p_value = results[1]

# Save the p-value to a text file
with open('white_test_p_value.txt', 'w') as f:
    f.write(f"White test p-value for the main model: {p_value}")


results_log_imp = het_white(model_log_imp.resid, X)
print("White test p-value for the model with Log_imp:", results_log_imp[1])

results_log_exp = het_white(model_log_exp.resid, X)
print("White test p-value for the model with Log_exp:", results_log_exp[1])

# Get robust standard errors
# Get the summary of the regression results

regression_result_4_FDI = model.get_robustcov_results(cov_type='HC3')
regression_result_4_FDI_summary = regression_result_4_FDI.summary()

# Convert the summary to a DataFrame
```

```
regression_result_4_FDI_df =
pd.DataFrame(regression_result_4_FDI_summary.tables[1].data)

# Save the DataFrame to a text file
regression_result_4_FDI_df.to_csv('regression_result_4_FDI.txt', sep='\t',
index=False)

regression_result_4_export = model_log_exp.get_robustcov_results(cov_type='HC3')

# Get the summary of the regression results
regression_result_4_export_summary = regression_result_4_export.summary()

# Convert the summary to a DataFrame
regression_result_4_export_df =
pd.DataFrame(regression_result_4_export_summary.tables[1].data)

# Save the DataFrame to a text file
regression_result_4_export_df.to_csv('regression_result_4_export.txt', sep='\t',
index=False)


regression_result_4_import = model_log_imp.get_robustcov_results(cov_type='HC3')
regression_result_4_import_summary = regression_result_4_import.summary()


# Convert the summary to a DataFrame
regression_result_4_import_df =
pd.DataFrame(regression_result_4_import_summary.tables[1].data)

# Save the DataFrame to a text file
regression_result_4_import_df.to_csv('regression_result_4_import.txt', sep='\t',
index=False)

print(regression_result_4_FDI.summary())
print(regression_result_4_export.summary())
print(regression_result_4_import.summary())
```

THANK YOU!