In [32]:
```
# Importing the pandas library
import pandas as pd

# Defining the file paths for the Excel files
file1_path = 'Variables.xlsx'
file2_path = 'Signal.xlsx'

# Loading the first Excel file into a Pandas DataFrame (Variable_df)
# Indexing the DataFrame by the "Sample.ID" column
Variable_df = pd.read_excel(file1_path, index_col="Sample.ID")

# Loading the second Excel file into a Pandas DataFrame (Signal_df)
Signal_df = pd.read_excel(file2_path)
```

In [33]:
```
# Filtering out rows in Variable_df where the index (Sample.ID) is not NaN
Variable_df = Variable_df[Variable_df.index.notna()]

# Displaying the first few rows of the modified Variable_df DataFrame
Variable_df.head()
```

Out[33]:

Sample.ID	Tissue	family	lipid	Carbons	Unsaturations	RT/s	Adduct	mz	ppm m/z adduct	MS/MS: m/z and fragment identification
Var_001	Liver	FA	FA(16:0)	16.0	0.0	248.01	[M-H]-	255.2311	-5.3	NaN
Var_002	Liver	FA	FA(16:1)	16.0	1.0	189.16	[M-H]-	253.2150	-7.0	NaN
Var_003	Liver	FA	FA(17:0)	17.0	0.0	291.59	[M-H]-	269.2463	-6.6	NaN
Var_004	Liver	FA	FA(18:0)	18.0	0.0	337.64	[M-H]-	283.2626	-3.7	NaN
Var_005	Liver	FA	FA(18:1)	18.0	1.0	263.97	[M-H]-	281.2468	-4.3	NaN

In [34]:
```
# Displaying the first few rows of the Signal_df DataFrame
Signal_df.head()
```

Out[34]:

	Sample.ID	Var_001	Var_002	Var_003	Var_004	Var_005	Var_006	Var_00
0	Control	3.743703e+06	124938.293443	34342.771479	3.237222e+06	2.276154e+06	1.839204e+06	34907.60723
1	Control	4.545332e+06	162651.056409	40449.353857	3.890073e+06	2.257932e+06	3.007269e+06	33246.50355
2	Control	3.142453e+06	103221.231656	29182.411423	2.627076e+06	1.540911e+06	2.041866e+06	25659.92652
3	Control	3.773138e+06	104932.822063	34026.158212	2.841866e+06	1.662993e+06	2.293310e+06	31054.23498
4	Control	3.685081e+06	174503.307487	29162.548885	2.856276e+06	2.873556e+06	2.286710e+06	26944.36636

5 rows × 283 columns

In [35]:
```
# Transposing the Signal_df DataFrame to swap rows and columns
transposed_signal_df = Signal_df.T

# Setting the columns of transposed_signal_df to the values in the first row
transposed_signal_df.columns = transposed_signal_df.iloc[0]

# Dropping the first row as it is now duplicated in the column headers
transposed_signal_df = transposed_signal_df[1:]
```

```
# Displaying the first few rows of the modified transposed_signal_df DataFrame
transposed_signal_df.head()
```

Out[35]:

Sample.ID	Control	Control	Control	Control	Control	Control	
Var_001	3743702.719371	4545331.53089	3142453.277634	3773137.834686	3685081.419616	3421208.186595	3626
Var_002	124938.293443	162651.056409	103221.231656	104932.822063	174503.307487	126512.768097	221
Var_003	34342.771479	40449.353857	29182.411423	34026.158212	29162.548885	27475.01862	29
Var_004	3237221.917914	3890073.086094	2627075.939279	2841866.205073	2856276.476992	2418878.431989	2400
Var_005	2276153.524206	2257931.672254	1540910.816543	1662993.437853	2873556.316456	2008735.84093	2196

5 rows × 38 columns

In [45]:
```
# Importing necessary libraries
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Extracting the features (Var_001 to Var_282) from Signal_df for PCA
features = Signal_df.iloc[:, 1:]

# Initializing a StandardScaler to standardize the features
scaler = StandardScaler()

# Standardizing the features by removing the mean and scaling to unit variance
features_standardized = scaler.fit_transform(features)
```

In [46]:
```
# Apply PCA with the desired number of components
num_components = 2  # You can change this to the number of components you want
pca = PCA(n_components=num_components)
principal_components = pca.fit_transform(features_standardized)

# Create a DataFrame with the principal components
principal_df = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

# Add the target variable to the principal DataFrame
principal_df['Sample.ID'] = Signal_df['Sample.ID']
principal_df.head()
```

Out[46]:

	PC1	PC2	Sample.ID
0	9.413856	-1.022951	Control
1	-0.028738	-9.427465	Control
2	3.870471	-6.907008	Control
3	6.982983	-1.638149	Control
4	-0.087840	-6.539702	Control

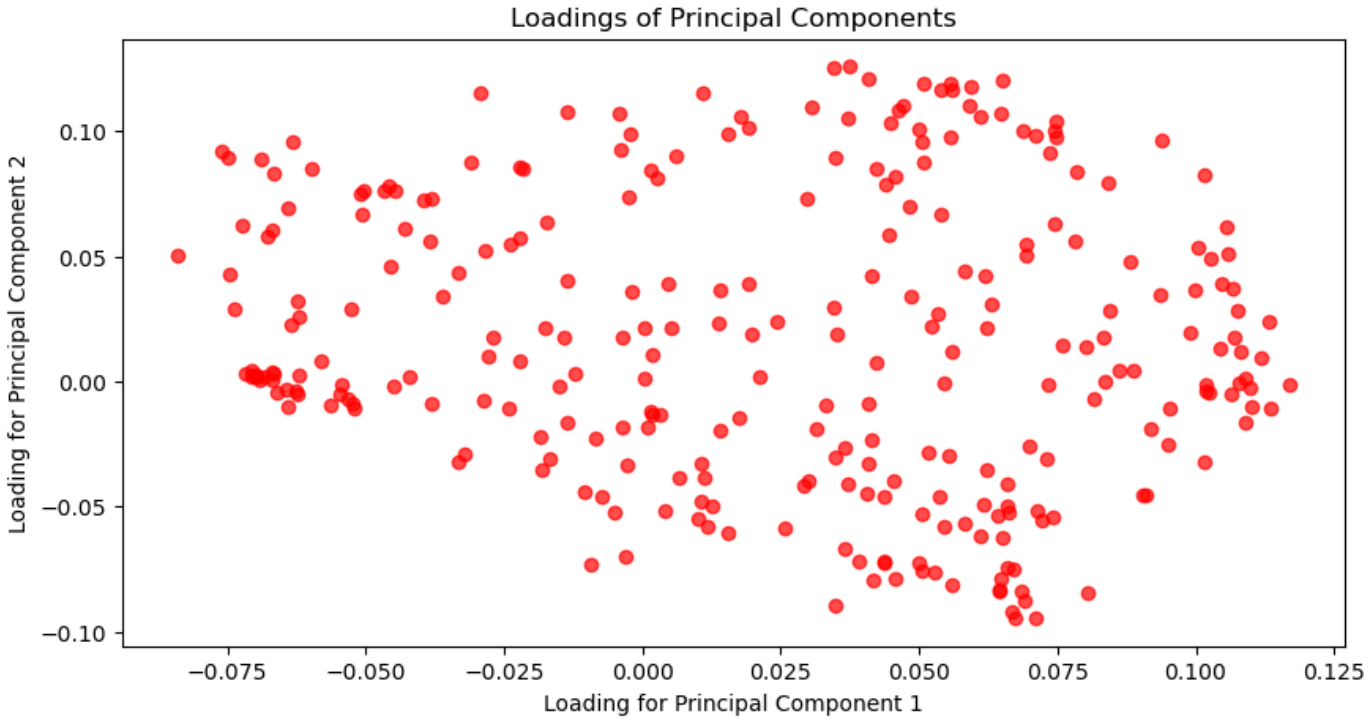
In [47]:
```
# Importing the NumPy library

import numpy as np

# Get the loadings from the PCA model
loadings = pca.components_.T  # Transpose to have loadings in rows

# Create a DataFrame with loadings
loadings_df = pd.DataFrame(data=loadings, columns=['PC1_loadings', 'PC2_loadings'], inde
```

In [48]:
```
# Displaying the first few rows of the loadings_df DataFrame

loadings_df.head()
```

Out[48]:

	PC1_loadings	PC2_loadings
Var_001	-0.005098	-0.052275
Var_002	0.043666	-0.071817
Var_003	-0.009278	-0.073087
Var_004	-0.045011	-0.001792
Var_005	0.039243	-0.071956

In [49]:
```
# Plot the loadings
plt.figure(figsize=(10, 5))
plt.scatter(loadings_df['PC1_loadings'], loadings_df['PC2_loadings'], marker='o', color=
plt.title('Loadings of Principal Components')
plt.xlabel('Loading for Principal Component 1')
plt.ylabel('Loading for Principal Component 2')


plt.show()
```



In [50]:
```
# Merging Variable_df and loadings_df based on their indices (Sample.ID)
# The left_index and right_index parameters specify that the merge should be based on th
merged_df = Variable_df.merge(loadings_df, left_index=True, right_index=True)

# Displaying the first few rows of the merged DataFrame
merged_df.head()
```

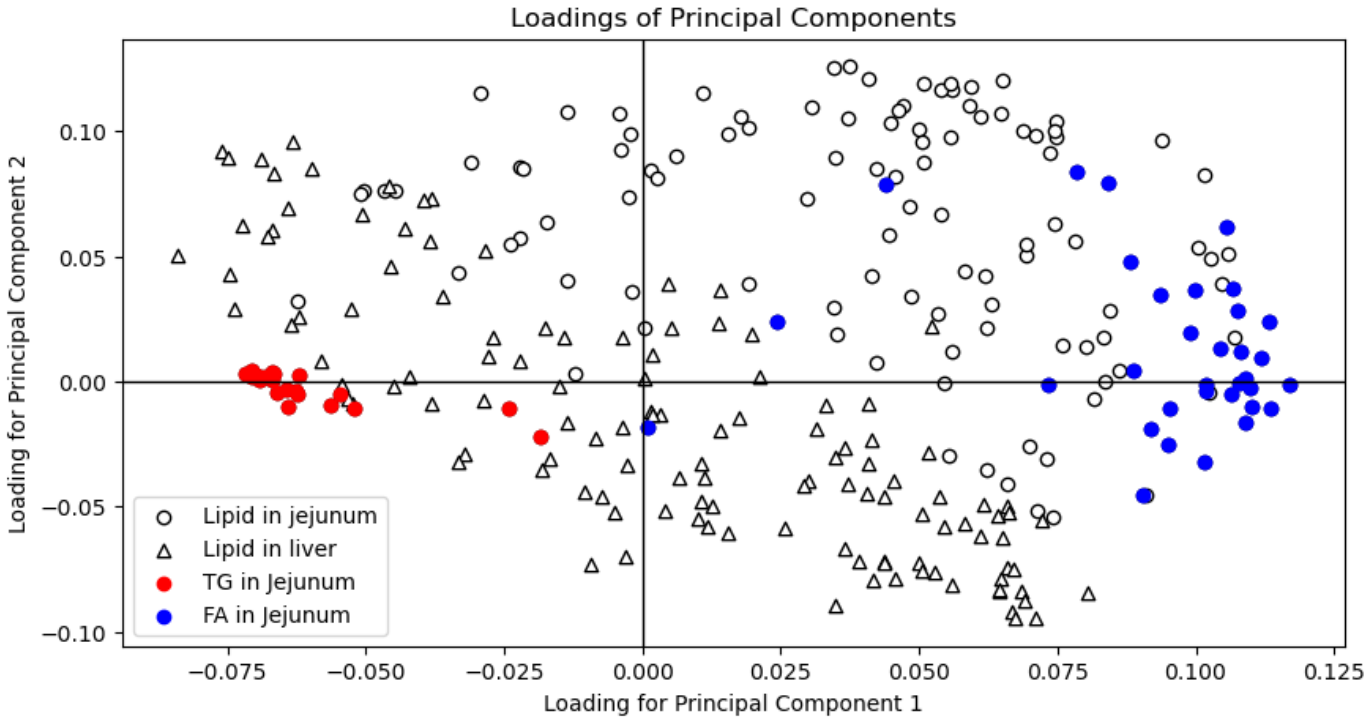
Out[50]:

	Tissue	family	lipid	Carbons	Unsaturations	RT/s	Adduct	mz	ppm m/z adduct	MS/MS: m/z and fragment identification	PC
Sample.ID											
Var_001	Liver	FA	FA(16:0)	16.0	0.0	248.01	[M-H]-	255.2311	-5.3	NaN	

Var_002	Liver	FA	FA(16:1)	16.0	1.0	189.16	[M-H]-	253.2150	-7.0	NaN
Var_003	Liver	FA	FA(17:0)	17.0	0.0	291.59	[M-H]-	269.2463	-6.6	NaN
Var_004	Liver	FA	FA(18:0)	18.0	0.0	337.64	[M-H]-	283.2626	-3.7	NaN
Var_005	Liver	FA	FA(18:1)	18.0	1.0	263.97	[M-H]-	281.2468	-4.3	NaN

In [51]:

```
# Plot the loadings
plt.figure(figsize=(10, 5))

# Scatter plot for Jejunum
plt.scatter(merged_df[merged_df['Tissue'] == 'Jejunum']['PC1_loadings'],
                              merged_df[merged_df['Tissue'] == 'Jejunum']['PC2_loadings'
                              marker='o', facecolors='white', edgecolors='black', label=

# Scatter plot for Liver
plt.scatter(merged_df[merged_df['Tissue'] == 'Liver']['PC1_loadings'],
                            merged_df[merged_df['Tissue'] == 'Liver']['PC2_loadings'],
                            marker='^', facecolors='white', edgecolors='black', label='L

# Scatter plot for Jejunum TG (red)
plt.scatter(merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['family'] == 'TG')
                        merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['famil
                        marker='o', facecolors='red', label='TG in Jejunum ')

# Scatter plot for Jejunum FA (blue)
plt.scatter(merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['family'] == 'FA')
                        merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['famil
                        marker='o', facecolors='blue', label='FA in Jejunum ')

# Set labels and title
plt.title('Loadings of Principal Components')
plt.xlabel('Loading for Principal Component 1')
plt.ylabel('Loading for Principal Component 2')

# Add lines at x and y = 0
plt.axhline(0, color='black', linestyle='-', linewidth=1)
plt.axvline(0, color='black', linestyle='-', linewidth=1)

# Add legend
plt.legend(loc='lower left')

# Show the plot
plt.show()
```

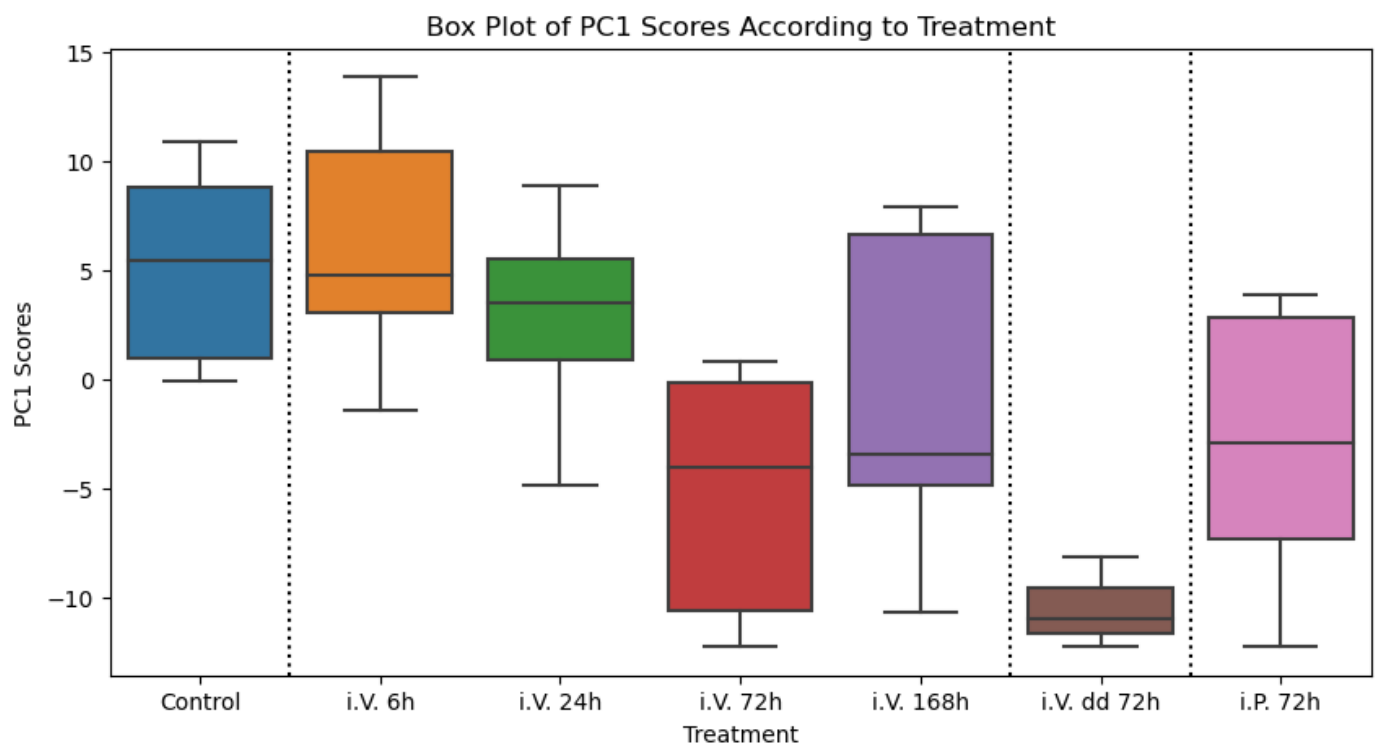


In [67]:
```
# Displaying the first few rows of the principal_df dataframe

principal_df.head()
```

Out[67]:

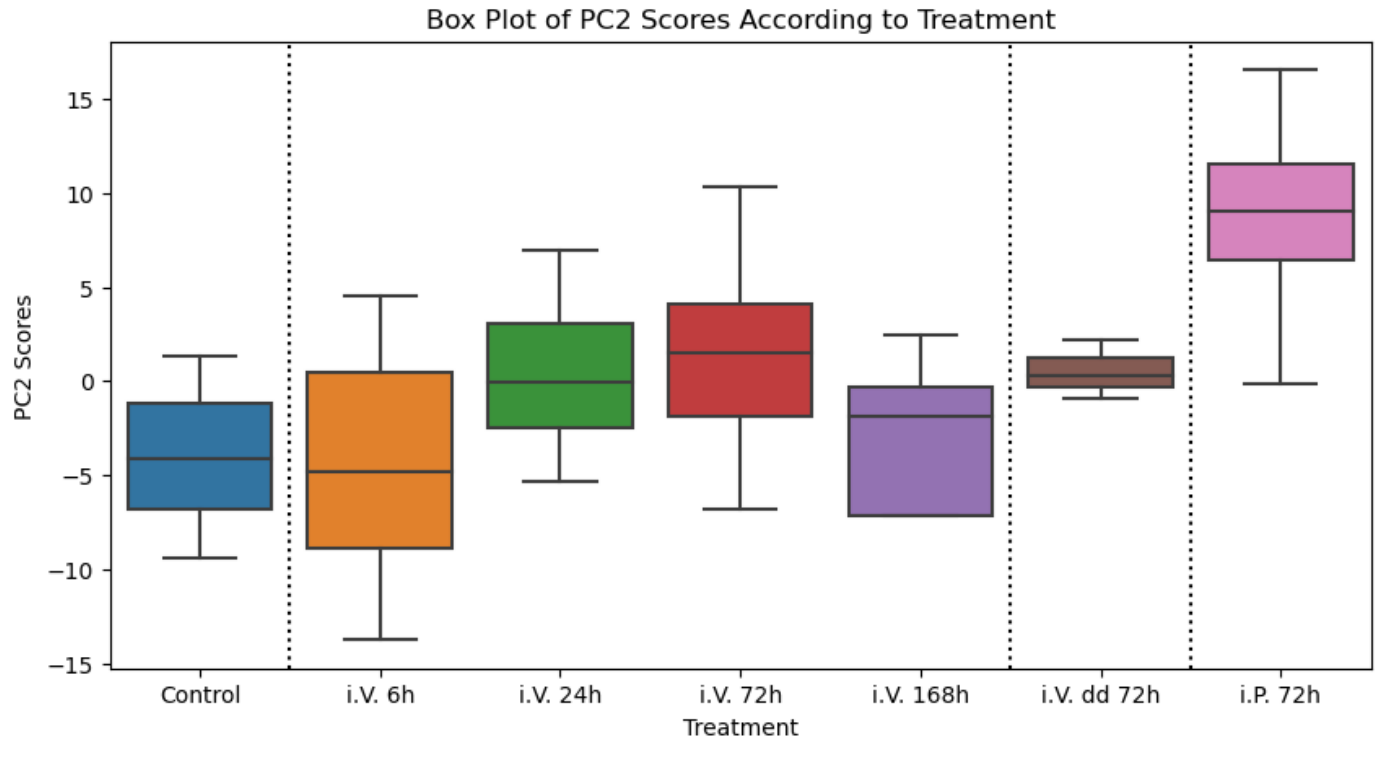
	PC1	PC2	Sample.ID
0	9.413856	-1.022951	Control
1	-0.028738	-9.427465	Control
2	3.870471	-6.907008	Control
3	6.982983	-1.638149	Control
4	-0.087840	-6.539702	Control

In [52]:
```
import seaborn as sns

# Create a box plot using Seaborn
plt.figure(figsize=(10, 5))
sns.boxplot(x='Sample.ID', y='PC1', data=principal_df)

plt.axvline(x=0.5, linestyle='dotted', color='black')
plt.axvline(x=4.5, linestyle='dotted', color='black')
plt.axvline(x=5.5, linestyle='dotted', color='black')

# Customize the plot
plt.title('Box Plot of PC1 Scores According to Treatment')
plt.xlabel('Treatment')
plt.ylabel('PC1 Scores')

# Show the plot
plt.show()
```



In [53]:

```
# Create a box plot using Seaborn
plt.figure(figsize=(10, 5))
sns.boxplot(x='Sample.ID', y='PC2', data=principal_df)

plt.axvline(x=0.5, linestyle='dotted', color='black')
plt.axvline(x=4.5, linestyle='dotted', color='black')
plt.axvline(x=5.5, linestyle='dotted', color='black')


# Customize the plot
plt.title('Box Plot of PC2 Scores According to Treatment')
plt.xlabel('Treatment')
plt.ylabel('PC2 Scores')

# Show the plot
plt.show()
```



In [68]:
```
# Filtering rows where 'Tissue' is 'Jejunum' and 'family' is 'TG'
filtered_TG_df = merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['family'] ==

# Get the number of rows in the filtered DataFrame for TG family
number_TG = len(filtered_TG_df)

# Assuming merged_df is your DataFrame
# Filtering rows where 'Tissue' is 'Jejunum' and 'family' is 'FA'
filtered_FA_df = merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['family'] ==

# Get the number of rows in the filtered DataFrame for FA family
number_FA = len(filtered_FA_df)

# Displaying the number of rows where Tissue is 'Jejunum' and Family is 'TG'
print("Number of rows where Tissue is 'Jejunum' and Family is 'TG':", number_TG)

# Displaying the number of rows where Tissue is 'Jejunum' and Family is 'FA'
print("Number of rows where Tissue is 'Jejunum' and Family is 'FA':", number_FA)
```

```
Number of rows where Tissue is 'Jejunum' and Family is 'TG': 22
Number of rows where Tissue is 'Jejunum' and Family is 'FA': 33
```

In [55]:
```
# Filtering rows in merged_df where 'Tissue' is 'Jejunum' and 'family' is 'TG'
tg_rows = merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['family'] == 'TG')]

# Extracting the index names of rows with TG family
tg_row_names = tg_rows.index.tolist()

# Filtering rows in merged_df where 'Tissue' is 'Jejunum' and 'family' is 'FA'
fa_rows = merged_df[(merged_df['Tissue'] == 'Jejunum') & (merged_df['family'] == 'FA')]

# Extracting the index names of rows with FA family
fa_row_names = fa_rows.index.tolist()
```

In [56]:
```
# Displaying the row names for TGs
print("Row Names for TGs:")
print(tg_row_names)

# Displaying the row names for FAs
print("\nRow Names for FAs:")
print(fa_row_names)
```

```
Row Names for TGs:
['Var_242', 'Var_243', 'Var_244', 'Var_245', 'Var_246', 'Var_247', 'Var_248', 'Var_249',
'Var_250', 'Var_251', 'Var_252', 'Var_253', 'Var_254', 'Var_255', 'Var_256', 'Var_257',
'Var_258', 'Var_259', 'Var_260', 'Var_261', 'Var_262', 'Var_263']

Row Names for FAs:
['Var_125', 'Var_126', 'Var_127', 'Var_128', 'Var_129', 'Var_130', 'Var_131', 'Var_132',
'Var_133', 'Var_134', 'Var_135', 'Var_136', 'Var_137', 'Var_138', 'Var_139', 'Var_140',
'Var_141', 'Var_142', 'Var_143', 'Var_144', 'Var_145', 'Var_146', 'Var_147', 'Var_148',
'Var_149', 'Var_150', 'Var_151', 'Var_152', 'Var_153', 'Var_154', 'Var_155', 'Var_156',
'Var_157']
```

In [57]:
```
# Displaying the first few rows of the transposed_signal_df DataFrame

transposed_signal_df.head()
```

Out[57]:

Sample.ID	Control	Control	Control	Control	Control	Control	
Var_001	3743702.719371	4545331.53089	3142453.277634	3773137.834686	3685081.419616	3421208.186595	3626
Var_002	124938.293443	162651.056409	103221.231656	104932.822063	174503.307487	126512.768097	221
Var_003	34342.771479	40449.353857	29182.411423	34026.158212	29162.548885	27475.01862	29

Var_004	3237221.917914	3890073.086094	2627075.939279	2841866.205073	2856276.476992	2418878.431989	2400
Var_005	2276153.524206	2257931.672254	1540910.816543	1662993.437853	2873556.316456	2008735.84093	2196

5 rows × 38 columns

In [58]:
```
# Extracting rows from transposed_signal_df for TG family using tg_row_names
signals_TG_df = transposed_signal_df.loc[tg_row_names]

# Displaying the first few rows of the signals_TG_df DataFrame
signals_TG_df.head()
```

Out[58]:

Sample.ID	Control	Control	Control	Control	Control	Control	
Var_242	224986.518962	222824.98726	235527.918659	219986.71889	3556897.717756	449533.986091	2
Var_243	2465863.721814	2486565.005852	2322272.665719	2596337.820758	8020593.097473	3391335.631506	24
Var_244	974038.817938	642457.264766	639433.724035	826171.913619	38996956.765054	9495680.645246	6
Var_245	413997.375163	246269.993958	183435.240879	352493.210978	42083919.56983	11178733.136763	2
Var_246	226411.898338	148348.604138	137516.2237	265019.947969	39621626.829111	6240109.872047	1

5 rows × 38 columns

In [59]:
```
# Extracting rows from transposed_signal_df for FA family using fa_row_names
signals_FA_df = transposed_signal_df.loc[fa_row_names]

# Displaying the first few rows of the signals_FA_df DataFrame
signals_FA_df.head()
```

Out[59]:

Sample.ID	Control	Control	Control	Control	Control	Contro
Var_125	112579.479075	64133.071423	93970.538561	105756.943518	109647.579152	147615.78176
Var_126	33360.132273	23173.916541	29010.741517	30797.833533	35343.188933	39604.07267
Var_127	244701.259073	164652.950698	251926.831617	326431.483196	269966.689348	365584.86222
Var_128	20065720.777074	15223593.948577	19821818.518761	26144621.504733	17880296.747787	26220104.99482
Var_129	2243236.846569	1373942.486798	1852082.394956	3809647.300398	2691969.289577	3046330.86321

5 rows × 38 columns

In [60]:
```
# Calculating the sum of signals for TG family and adding a new column 'Sum_TG'
Signal_df['Sum_TG'] = Signal_df[tg_row_names].sum(axis=1)

# Calculating the sum of signals for FA family and adding a new column 'Sum_FA'
Signal_df['Sum_FA'] = Signal_df[fa_row_names].sum(axis=1)

# Displaying the first few rows of the modified Signal_df DataFrame
Signal_df.head()
```

Out[60]:

	Sample.ID	Var_001	Var_002	Var_003	Var_004	Var_005	Var_006	Var_00
0	Control	3.743703e+06	124938.293443	34342.771479	3.237222e+06	2.276154e+06	1.839204e+06	34907.60723
1	Control	4.545332e+06	162651.056409	40449.353857	3.890073e+06	2.257932e+06	3.007269e+06	33246.50355
2	Control	3.142453e+06	103221.231656	29182.411423	2.627076e+06	1.540911e+06	2.041866e+06	25659.92652
3	Control	3.773138e+06	104932.822063	34026.158212	2.841866e+06	1.662993e+06	2.293310e+06	31054.23498

4	Control	3.685081e+06	174503.307487	29162.548885	2.856276e+06	2.873556e+06	2.286710e+06	26944.36636

5 rows × 285 columns

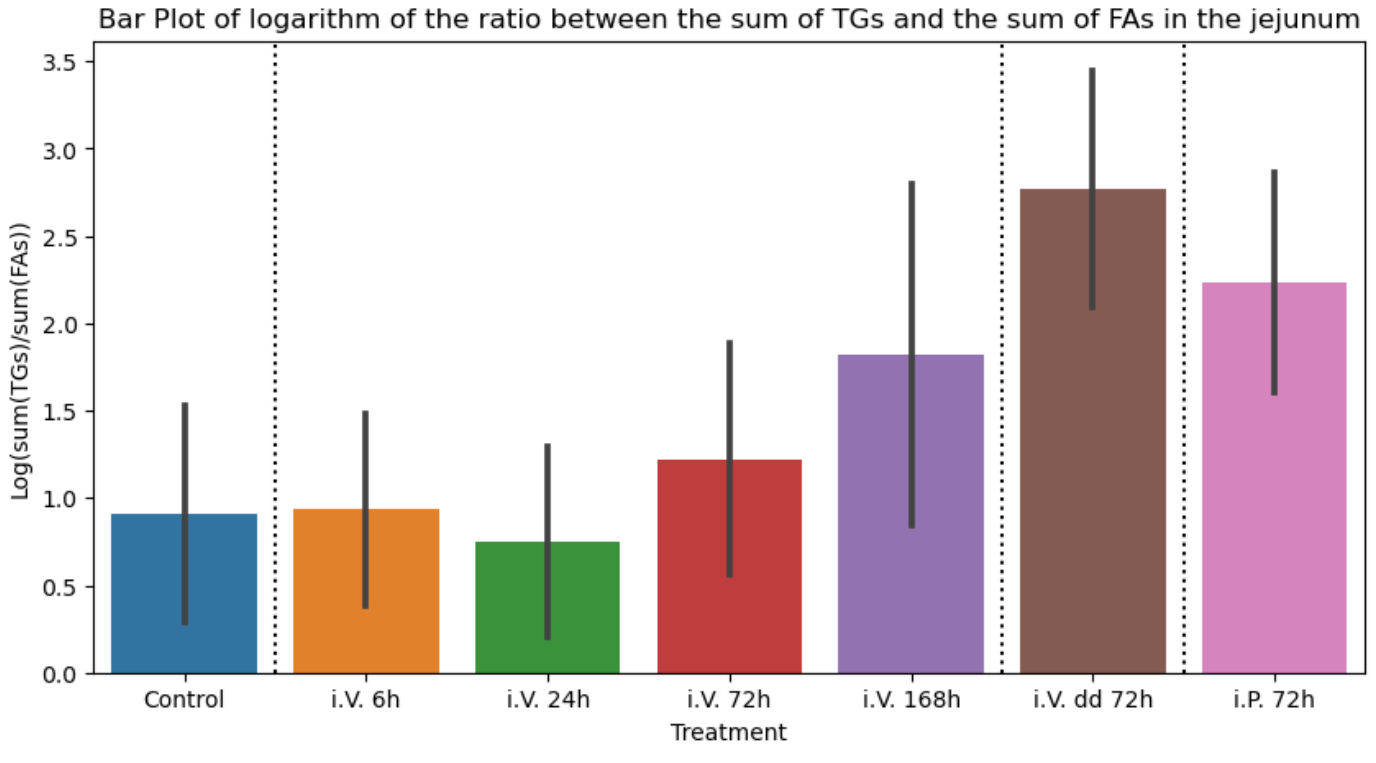
In [61]:
```
# Calculating the log ratio of the sum of signals for TG and FA families and adding a ne
Signal_df['Log_Ratio'] = np.log(Signal_df['Sum_TG'] / Signal_df['Sum_FA'])

# Displaying the first few rows of the modified Signal_df DataFrame
Signal_df.head()
```

Out[61]:

	Sample.ID	Var_001	Var_002	Var_003	Var_004	Var_005	Var_006	Var_00
0	Control	3.743703e+06	124938.293443	34342.771479	3.237222e+06	2.276154e+06	1.839204e+06	34907.60723
1	Control	4.545332e+06	162651.056409	40449.353857	3.890073e+06	2.257932e+06	3.007269e+06	33246.50355
2	Control	3.142453e+06	103221.231656	29182.411423	2.627076e+06	1.540911e+06	2.041866e+06	25659.92652
3	Control	3.773138e+06	104932.822063	34026.158212	2.841866e+06	1.662993e+06	2.293310e+06	31054.23498
4	Control	3.685081e+06	174503.307487	29162.548885	2.856276e+06	2.873556e+06	2.286710e+06	26944.36636

5 rows × 286 columns

In [62]:
```
# Create a box plot using Seaborn
plt.figure(figsize=(10, 5))
sns.barplot(x='Sample.ID', y='Log_Ratio', data=Signal_df, errorbar='se')

plt.axvline(x=0.5, linestyle='dotted', color='black')
plt.axvline(x=4.5, linestyle='dotted', color='black')
plt.axvline(x=5.5, linestyle='dotted', color='black')

# Customize the plot
plt.title('Bar Plot of logarithm of the ratio between the sum of TGs and the sum of FAs
plt.xlabel('Treatment')
plt.ylabel('Log(sum(TGs)/sum(FAs))')

# Show the plot
plt.show()
```



In [ ]:

In [ ]: