```python
import pandas as pd
import seaborn as sns
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import sys
sys.path.append('.')
import null_analysis as na
import scoring
import analysis
```

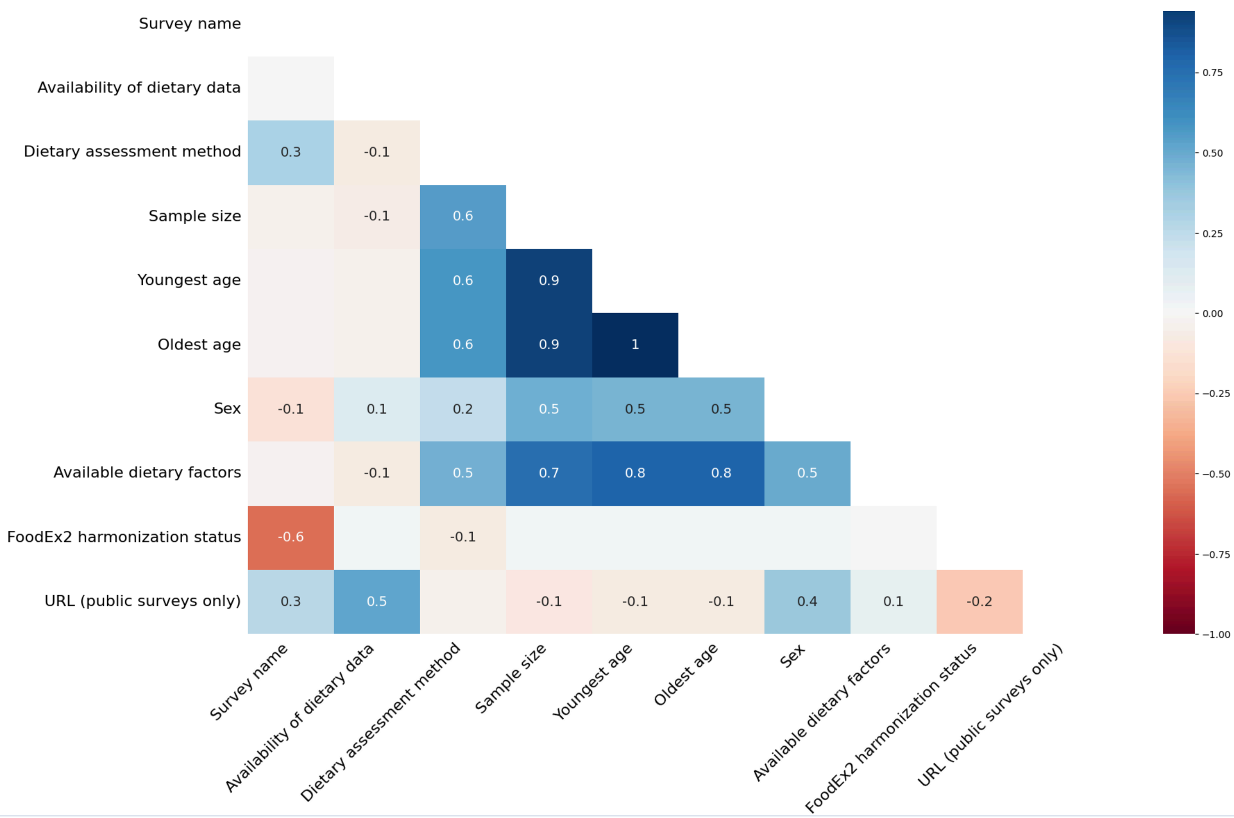
```python
# Load Global Dietary Database (GDD) dataset

gdd_df = pd.read_csv('GDD Dec2020 Survey Metadata 1611c.csv', low_memory=False)
print(f"Total number of rows: {gdd_df.size}")
print(gdd_df['Representativeness'].unique())
```

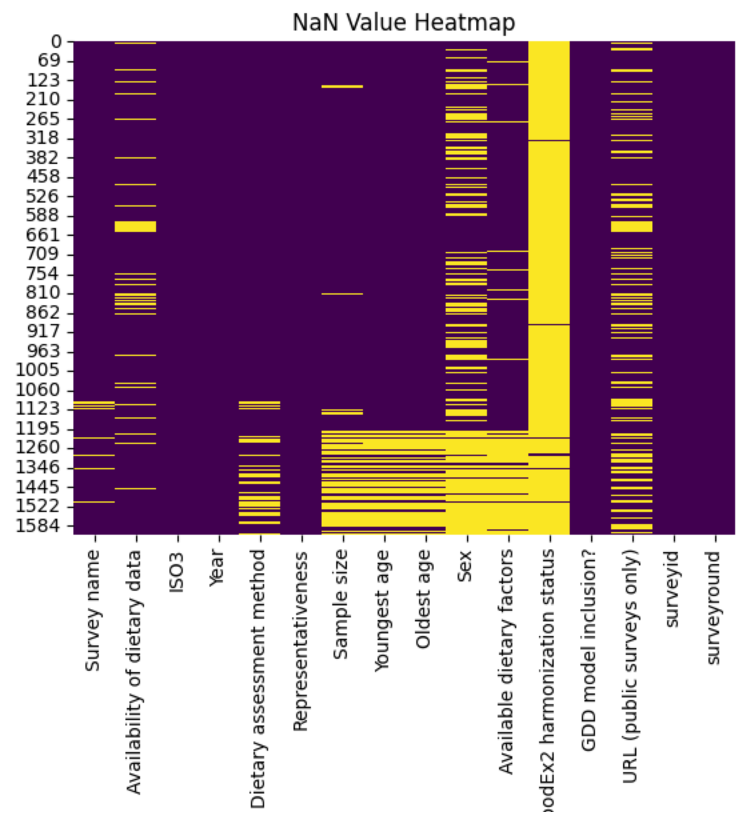
```
Total number of rows: 25776
['National' 'Local' 'Subnational' '9 MISSING']
```

```python
# Filter the dataframe based on the following conditions:
# - National level
# - Years between 1990 and 2017 (inclusive)
gdd_df = gdd_df[
    (gdd_df['Representativeness'] == 'National') &
    (gdd_df['Year'] >= 1990) &
    (gdd_df['Year'] <= 2017)
]
print(f"Total number of rows: {gdd_df.size}")
```

```
Total number of rows: 16288
```

```python
# Explore missing values in GDD dataset

# Heatmap of missing values correlation (Missingno)
msno.heatmap(gdd_df)
plt.show()

# Basic heatmap of NaN values (Seaborn)
sns.heatmap(gdd_df.isna(), cbar=False, cmap='viridis')
plt.title('NaN Value Heatmap')
plt.show()

# Count NaN values per column
print("Count of NaN values per column:")
print(gdd_df.isna().sum())

# Percentage of NaN values per column
print("\nPercentage of NaN values per column:")
print(gdd_df.isna().mean() * 100)
```






```
Representativeness              0
Sample size                   152
Youngest age                  139
Oldest age                    139
Sex                           423
Available dietary factors     203
FoodEx2 harmonization status  988
GDD model inclusion?            0
URL (public surveys only)     286
surveyid                        0
```

```python
# Inspect 'Sex' column in GDD
gdd_df.columns

# Print absolute counts of each value in the 'Sex' column (including NaN)
print(gdd_df['Sex'].value_counts(dropna=False))

# Print relative proportions (%) of each value in the 'Sex' column (including NaN)
print(gdd_df['Sex'].value_counts(normalize=True, dropna=False) * 100)
```

```
Sex
Male|Female    593
NaN            423
Male             1
Female           1
Name: count, dtype: int64
Sex
Male|Female    58.251473
NaN            41.552063
Male            0.098232
Female          0.098232
Name: proportion, dtype: float64
```

```python
# Process 'Available dietary factors' and clean GDD dataframe

# Drop rows with missing values
gdd_df = gdd_df.dropna(subset=['Available dietary factors'])

# Convert 'Year' to numeric, handling any non-numeric values
pd.to_numeric(gdd_df['Year'], errors='coerce')
# Drop rows where 'Year' is missing
gdd_df = gdd_df.dropna(subset=['Year'])
# Convert 'Year' to integer
gdd_df['Year'] = gdd_df['Year'].astype(int)

# Drop unnecessary columns
gdd_df = gdd_df.drop(
    columns=['Survey name', 'Availability of dietary data', 'Dietary assessment method', 'Representativeness',
    'Sample size', 'Sex','FoodEx2 harmonization status', 'GDD model inclusion?',
    'URL (public surveys only)', 'surveyid', 'surveyround']
)

gdd_df
```

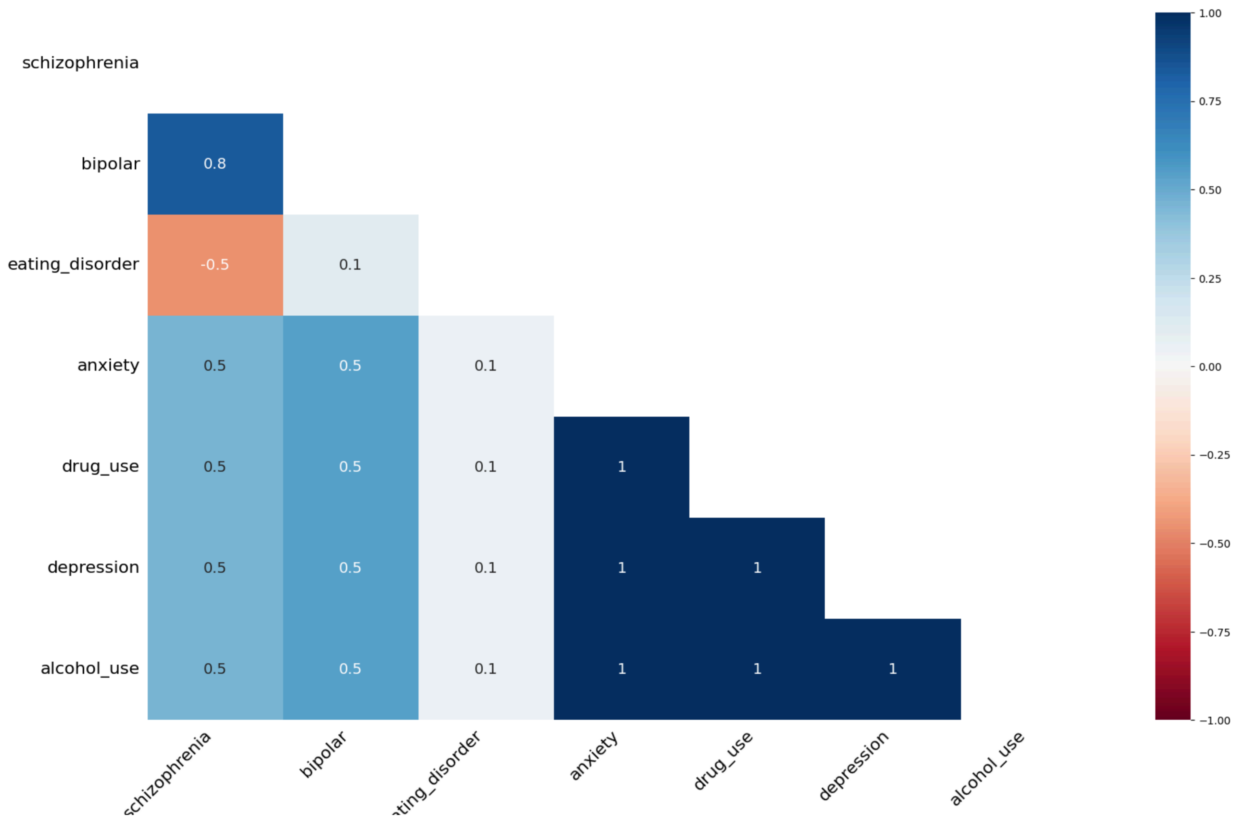
<table>
<tr><th></th><th>ISO3 object</th><th>Year int64</th><th>Youngest age flo...</th><th>Oldest age float64</th><th>Available dietary ...</th></tr>
<tr><td></td><td>USA 4.2%
JPN 2.8%
180 others 93%</td><td>1990 - 2017</td><td>0.0 - 70.0</td><td>1.0 - 120.0</td><td>Fruits|Non... 20.1%
Fruits|Non... 15.5%
198 others 64.4%</td></tr>
<tr><td>1</td><td>ALB</td><td>2008</td><td>0</td><td>98</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>2</td><td>ARE</td><td>2010</td><td>11</td><td>16</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>3</td><td>ARE</td><td>2009</td><td>6</td><td>50</td><td>Total Energy|Tota...</td></tr>
<tr><td>4</td><td>ARE</td><td>2005</td><td>11</td><td>16</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>5</td><td>ARE</td><td>2003</td><td>18</td><td>90</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>7</td><td>ARG</td><td>2012</td><td>11</td><td>16</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>11</td><td>ARG</td><td>2007</td><td>11</td><td>16</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>13</td><td>ARG</td><td>2004</td><td>20</td><td>49</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>15</td><td>ARM</td><td>2010</td><td>11</td><td>15</td><td>Fruits|Non-Starch...</td></tr>
<tr><td>16</td><td>ARM</td><td>2005</td><td>0</td><td>49</td><td>Fruits|Non-Starch...</td></tr>
</table>

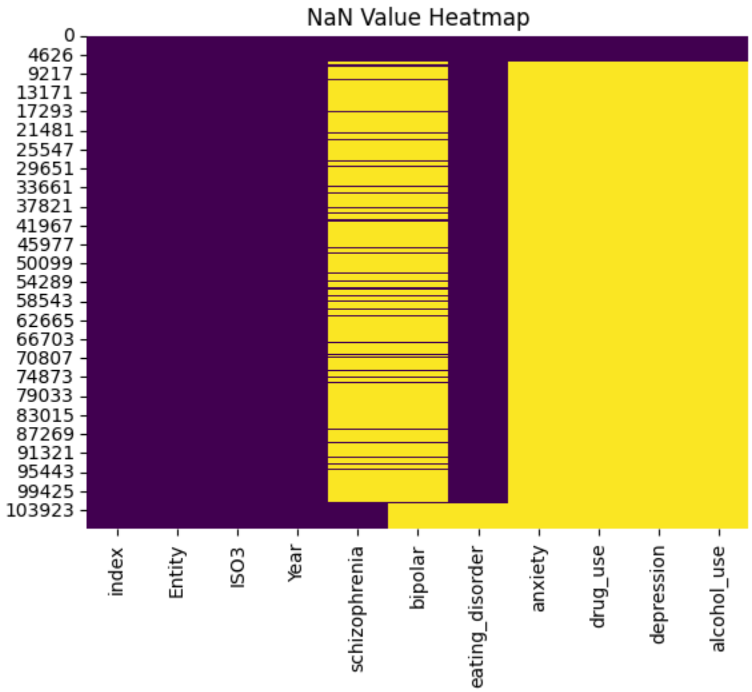
815 rows, 5 cols 10 / page Page 1 of 82

```python
# Load and clean Global Mental Health Disorders dataset

mental_health_df = pd.read_csv('mental_health.csv', low_memory=False)

# Convert 'Year' to numeric, handling any non-numeric values
mental_health_df['Year'] = pd.to_numeric(mental_health_df['Year'], errors='coerce')
# Drop rows where 'Year' is missing
mental_health_df = mental_health_df.dropna(subset=['Year'])
# Convert 'Year' to integer
mental_health_df['Year'] = mental_health_df['Year'].astype(int)

# Rename 'Code' column to 'ISO3'
mental_health_df = mental_health_df.rename(columns={'Code': 'ISO3'})

# Drop rows where 'ISO3' is missing or is 'OWID_WRL'
# NOTE: 'OWID_WRL' stands for 'Our World In Data - World' (global aggregate row)
mental_health_df = mental_health_df.dropna(subset=['ISO3'])
mental_health_df = mental_health_df[mental_health_df['ISO3'] != 'OWID_WRL']

# Rename columns to make it easier to work with
rename_map = {
    'Schizophrenia (%)': 'schizophrenia',
    'Bipolar disorder (%)': 'bipolar',
    'Eating disorders (%)': 'eating_disorder',
    'Anxiety disorders (%)': 'anxiety',
    'Drug use disorders (%)': 'drug_use',
    'Depression (%)': 'depression',
    'Alcohol use disorders (%)': 'alcohol_use'
}
mental_health_df = mental_health_df.rename(columns=rename_map)

print(f"Total number of rows: {mental_health_df.size}")
```

```
Total number of rows: 1130580
```

```python
# Visualize missing data patterns

# Heatmap of missing values correlation (Missingno)
msno.heatmap(mental_health_df)
plt.show()

# Basic heatmap of NaN values (Seaborn)
sns.heatmap(mental_health_df.isna(), cbar=False, cmap='viridis')
plt.title('NaN Value Heatmap')
plt.show()

# Count NaN values per column
print("Count of NaN values per column:")
print(mental_health_df.isna().sum())

# Percentage of NaN values per column
print("\nPercentage of NaN values per column:")
print(mental_health_df.isna().mean() * 100)
```






```
Count of NaN values per column:
index                 0
Entity                0
ISO3                  0
Year                  0
schizophrenia     80940
bipolar           86400
eating_disorder    5460
anxiety           97320
drug_use          97320
depression        97320
alcohol_use       97320
dtype: int64

Percentage of NaN values per column:
index               0.000000
Entity              0.000000
ISO3                0.000000
Year                0.000000
schizophrenia      78.750730
bipolar            84.063047
eating_disorder     5.312318
anxiety            94.687682
drug_use           94.687682
depression         94.687682
alcohol_use        94.687682
dtype: float64
```

```python
# Additional analysis on missing data for eating_disorder, schizophrenia, and
# dataset excluding eating_disorder and schizophrenia

# eating_disorder: had the least amount of NaN values in the dataset
ed_clean_df = mental_health_df.dropna(subset=['eating_disorder'])
ed_clean_df = ed_clean_df[['ISO3', 'Year', 'eating_disorder']]
print(f"eating_disorder clean dataframe shape: {ed_clean_df.shape}")
print(ed_clean_df.isnull().sum())

# schizophrenia: spitting from the ends and excluding middle values to prevent skews
schiz_df = mental_health_df.dropna(subset=['schizophrenia']).sort_values(by='schizophrenia')
schiz_df = schiz_df[['ISO3', 'Year', 'schizophrenia']]
schiz_first_5k_df = schiz_df.head(5000)
schiz_last_5k_df = schiz_df.tail(5000)
schiz_cleaned_df = pd.concat([schiz_first_5k_df, schiz_last_5k_df], axis=0).reset_index(drop=True)
print(f"schizophrenia first/last 5k dataframe shape: {schiz_cleaned_df.shape}")
print(schiz_cleaned_df.isnull().sum())

# chunk of data frame with no missing values, excluding eating_disorder and schizophrenia
mh_columns_to_keep = ['ISO3', 'Year', 'bipolar', 'anxiety', 'drug_use', 'depression', 'alcohol_use']
mh_clean_df = mental_health_df.drop(columns=['schizophrenia', 'eating_disorder'])
mh_clean_df = mh_clean_df.head(5000)
mh_clean_df = mh_clean_df[mh_columns_to_keep]
print(f"5k rows, mental health disorders (excluding schizophrenia and ED) dataframe shape: {mh_clean_df.shape}")
print(mh_clean_df.isnull().sum())
```

```
eating_disorder clean dataframe shape: (97320, 3)
ISO3               0
Year               0
eating_disorder    0
dtype: int64
schizophrenia first/last 5k dataframe shape: (10000, 3)
ISO3             0
Year             0
schizophrenia    0
dtype: int64
5k rows, mental health disorders (excluding schizophrenia and ED) dataframe shape: (5000, 7)
ISO3           0
Year           0
bipolar        0
anxiety        0
drug_use       0
depression     0
alcohol_use    0
dtype: int64
```

```python
# Analyze missing values in Mental Health Disorders by country
# Explore NaN values by country (ISO3)

# Mental health disorder list
disorders = ['schizophrenia', 'bipolar', 'eating_disorder', 'anxiety', 'drug_use', 'depression', 'alcohol_use']

# Group by country (ISO3) and calculate the number of missing values (NaN) per disorder
# [Count]
nan_by_iso3 = mental_health_df[disorders].isna().groupby(mental_health_df['ISO3']).sum()
print(nan_by_iso3)
# [Percentage]
nan_pct_by_iso3 = mental_health_df[disorders].isna().groupby(mental_health_df['ISO3']).mean() * 100
print(nan_pct_by_iso3)
```

```
      schizophrenia  bipolar  eating_disorder  anxiety  drug_use  depression  \
ISO3
ABW             140      140                0      140       140         140
AFG             384      412               28      468       468         468
AGO             384      412               28      468       468         468
AIA             140      140                0      140       140         140
ALB             384      412               28      468       468         468
...             ...      ...              ...      ...       ...         ...
WSM             384      412               28      468       468         468
YEM             384      412               28      468       468         468
ZAF             384      412               28      468       468         468
ZMB             384      412               28      468       468         468
ZWE             384      412               28      468       468         468

      alcohol_use
ISO3
ABW           140
AFG           468
AGO           468
AIA           140
ALB           468
...           ...
WSM           468
YEM           468
ZAF           468
ZMB           468
ZWE           468

[234 rows x 7 columns]
      schizophrenia     bipolar  eating_disorder     anxiety    drug_use  \
```

```python
# Drop rows with missing values in any of the disorder columns
print(f"Rows BEFORE dropping missing values: {len(mental_health_df)}")

# Drop rows where any of the disorder columns are NaN
nonnull_mental_health_df = mental_health_df.dropna(subset=disorders)
print(f"Rows AFTER dropping missing values: {len(nonnull_mental_health_df)}")
```

```
Rows BEFORE dropping missing values: 102780
Rows AFTER dropping missing values: 5460
```

```python
# Clean Mental Health dataframe
# Convert column types from object to numeric
for column in disorders:
    mental_health_df[column] = pd.to_numeric(mental_health_df[column], errors='coerce')

# Drop unnecessary columns
mental_health_df = mental_health_df.drop(columns=['index'])
mental_health_df
```

	Entity object	ISO3 object	Year int64	schizophrenia flo...	bipolar float64	eating_disorder f...	anxiety float64	d
0	Afghanistan	AFG	1990	0.16056	0.697779	0.101855	4.82883	
1	Afghanistan	AFG	1991	0.160312	0.697961	0.099313	4.82974	
2	Afghanistan	AFG	1992	0.160135	0.698107	0.096692	4.831108	
3	Afghanistan	AFG	1993	0.160037	0.698257	0.094336	4.830864	
4	Afghanistan	AFG	1994	0.160022	0.698469	0.092439	4.829423	

102780 rows, 10 cols 5 / page Page 1 of 20556

```python
# Process Global Dietary Diversity dataset

# Convert string of factors to a set per row
def parse_factors(factor_str):
    if pd.isna(factor_str):
        return set()
    return set(f.strip() for f in factor_str.strip('{}').split('|'))

# Apply parsing to create per-row sets
gdd_df['factor_set'] = gdd_df['Available dietary factors'].apply(parse_factors)

# Group by ISO3 and Year to union all sets per country-year
factor_union_df = gdd_df.groupby(['ISO3', 'Year'])['factor_set'].apply(lambda sets: set().union(*sets)).reset_index()

# Merge back to original gdd_df and drop the per-row factor_set to avoid conflict
gdd_df = gdd_df.drop(columns=['factor_set', 'Available dietary factors'])

# Merge in the unified factor_set and Diet Category
diet_df = pd.merge(gdd_df, factor_union_df, on=['ISO3', 'Year'], how='left')

diet_df.sample(20)
```

	ISO3 object	Year int64	Youngest age flo...	Oldest age float64	factor_set object
	HUN 10% ARE 5% 17 others 85%	1991 - 2014	0.0 - 25.0	4.0 - 100.0	{'Sugar-Sw... 15% {'Non-Starc... 10% 14 others 75%
3	ARE	2005	11	16	{'Non-Starchy Ve...
78	BRN	2014	7	16	{'Sugar-Sweeten...
34	BEN	2011	0	49	{'Yogurt', 'Beans a...
154	DMA	2009	11	16	{'Sugar-Sweeten...
176	EGY	2011	11	16	{'Sugar-Sweeten...
287	HUN	2003	18	95	{'Dietary Sodium',...
44	BFA	1993	0	4	{'Fruit Juice', 'Tot...
272	HND	2011	0	49	{'Yogurt', 'Beans a...
576	POL	2000	4	100	{'Yogurt', 'Dietary ...
295	HUN	1991	0	100	{'Beans and Legu...

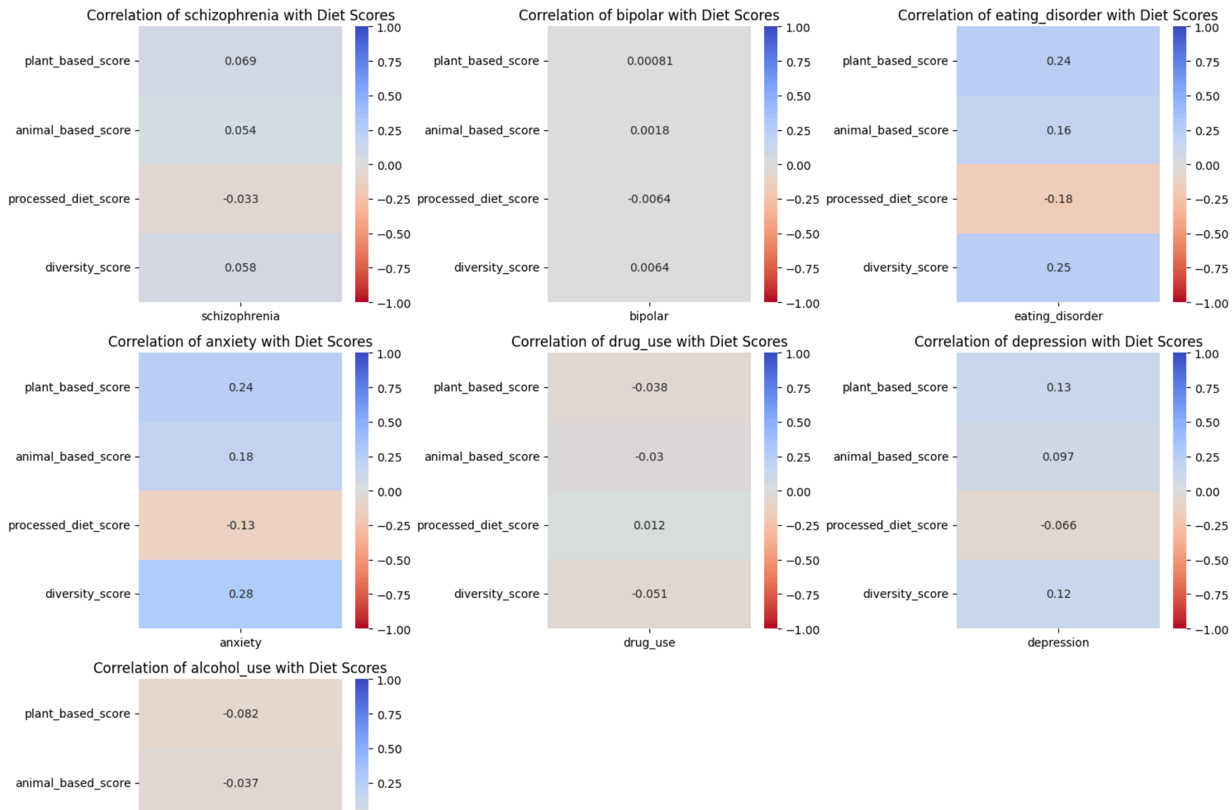
20 rows, 5 cols 10 / page Page 1 of 2

```python
# Keeping the exploration for historal purposes.


# vitamins = {'Vitamin A with Supplements', 'Vitamin A without Supplements',
#     'Vitamin B1', 'Vitamin B2', 'Vitamin B3', 'Vitamin B6', 'Vitamin B9',
#     'Vitamin C', 'Vitamin D', 'Vitamin E', 'Calcium', 'Magnesium', 'Potassium',
#     'Iodine', 'Selenium', 'Zinc'}

# # Function to compute vitamin scores
# # vitamin_count = number of distinct vitamins
# # vitamin_coverage = percentage of distinct vitamins, normalized score so we can compare across countries
# def vitamin_score(factors):
#     if not factors:
#         return pd.Series({'vitamin_count': 0,
#                         'vitamin_coverage': 0,
#                         'has_vitamin': 0})

#     vitamin_count = sum(f in vitamins for f in factors)
#     max_vitamin_count = len(vitamins)
#     vitamin_coverage = vitamin_count / max_vitamin_count if max_vitamin_count > 0 else 0.0
#     has_vitamin = 1 if vitamin_count > 0 else 0

#     return pd.Series({'vitamin_count': vitamin_count,
#                     'vitamin_coverage': vitamin_coverage,
#                     'has_vitamin': has_vitamin})

# diet_df[['vitamin_count', 'vitamin_coverage', 'has_vitamin']] = diet_df['factor_set'].apply(vitamin_score)
# # Apply counting function to your factor sets
# diet_df[['ISO3', 'Year', 'vitamin_count', 'vitamin_coverage', 'has_vitamin']]
```

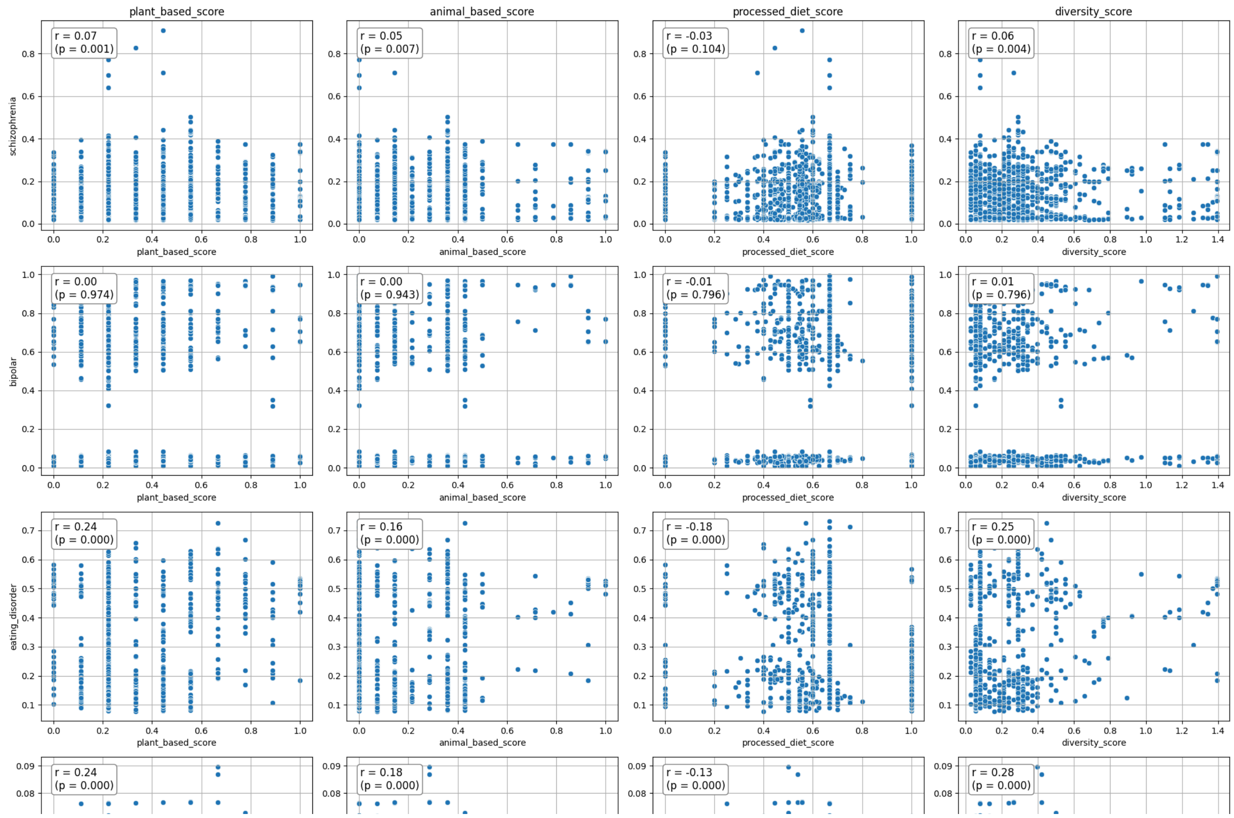
```python
# Merge grouped diet data with mental health dataset on ISO3 and Year
diet_mental_health_df = pd.merge(diet_df, mental_health_df, on=['ISO3', 'Year'])
```

```python
# Groupings of dietary factors for analysis
plant_based = {
    'Fruits', 'Non-Starchy Vegetables', 'Beans and Legumes', 'Nuts and Seeds',
    'Fruit Juice', 'Dietary Fiber', 'Whole Grains', 'Plant Omega-3 Fat',
    'Plant Protein'
}

animal_based = {
    'Unprocessed Red Meats', 'Total Seafoods', 'Seafood Omega-3 Fat', 'Dietary Cholesterol',
    'Total Milk', 'Cheese', 'Yogurt', 'Eggs', 'Whole Fat Milk', 'Reduced Fat Milk',
    'Total Processed Meats', 'Dairy Protein', 'Animal Protein', 'Total Animal Protein'
}

processed_items = {
    'Added Sugars', 'Sugar-Sweetened Beverages', 'Fruit Juice', 'Refined Grains',
    'Unprocessed Red Meats', 'Total Processed Meats', 'Total Animal Protein',
    'Total Energy', 'Saturated Fat', 'Trans Fatty Acid', 'Dietary Cholesterol',
    'Total Carbohydrates', 'Dairy Protein', 'Whole Fat Milk', 'Reduced Fat Milk',
    'Cheese', 'Glycemic Index', 'Glycemic Load', 'Other Starchy Vegetables',
    'Potatoes', 'Monounsaturated Fat', 'Animal Protein'
}

unprocessed_items = {
    'Fruits', 'Non-Starchy Vegetables', 'Beans and Legumes', 'Nuts and Seeds',
    'Whole Grains', 'Plant Protein', 'Plant Omega-3 Fat', 'Seafood Omega-3 Fat',
    'Dietary Fiber', 'Dietary Sodium',
    'Total Seafoods', 'Yogurt', 'Eggs', 'Total Protein', 'Coffee', 'Tea'
}

# To score diet diversity
all_items = all_items = processed_items.union(unprocessed_items)

feature_cols = ['plant_based_score', 'animal_based_score', 'processed_diet_score', 'diversity_score']
outcome_cols = ['schizophrenia', 'bipolar', 'eating_disorder', 'anxiety', 'drug_use', 'depression', 'alcohol_use']
```

```
# Correlation matrix for diet_mental_health_df

diet_mental_health_df = scoring.add_diet_scores(diet_mental_health_df, plant_based, animal_based, 'factor_set')
diet_mental_health_df = scoring.compute_processed_diet_score(diet_mental_health_df, 'factor_set', processed_items, unproc
diet_mental_health_df = scoring.add_diversity_score(diet_mental_health_df, 'factor_set', all_items)
diet_mental_health_df = analysis.normalize_outcome_columns(diet_mental_health_df)
analysis.plot_outcome_correlations(diet_mental_health_df, outcome_cols, feature_cols)
```



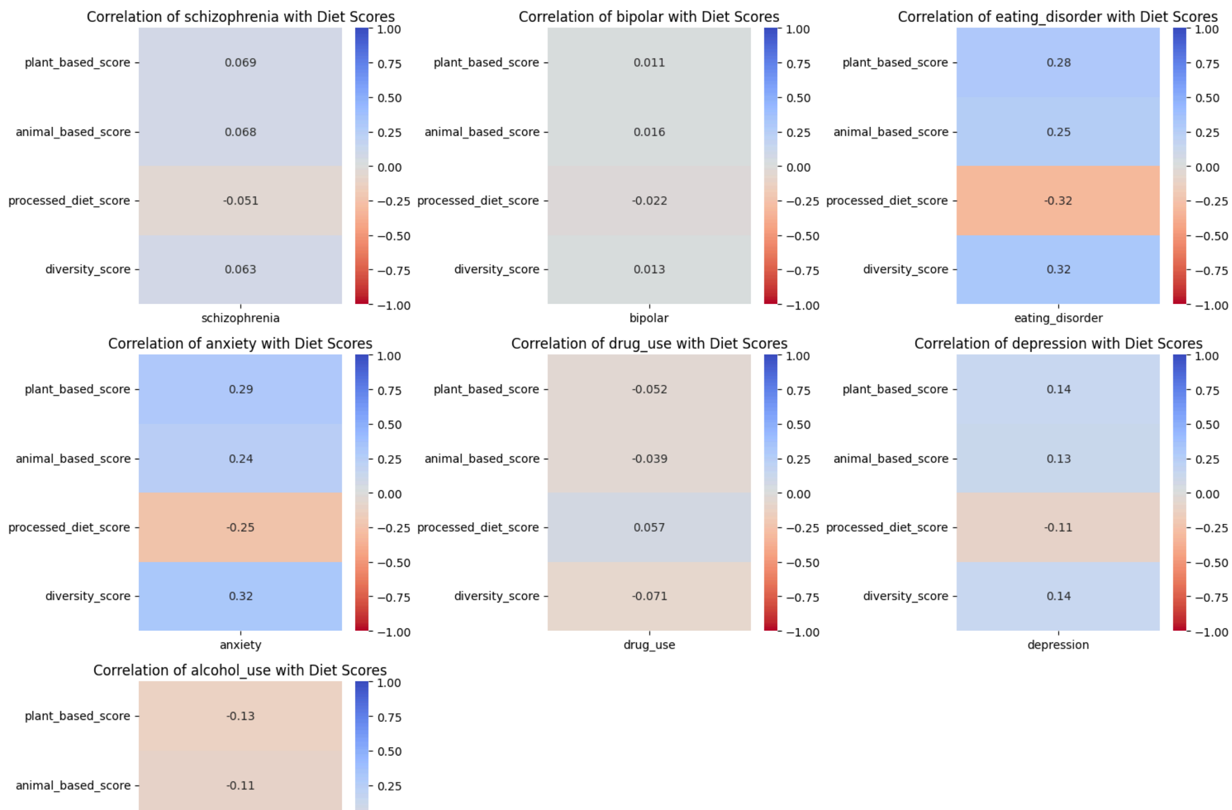
```python
# Scatter plot with correlation for diet_mental_health_df

analysis.plot_scatter_with_correlation(diet_mental_health_df, feature_cols, outcome_cols)
```




```python
# Splitting up adults and kids

adults_df = diet_mental_health_df[diet_mental_health_df['Oldest age'] >= 18]
kids_df = diet_mental_health_df[diet_mental_health_df['Oldest age'] < 18]
```

```python
# Correlation matrix for adult_df

adults_df = scoring.add_diet_scores(adults_df, plant_based, animal_based, 'factor_set')
adults_df = scoring.compute_processed_diet_score(adults_df, 'factor_set', processed_items, unprocessed_items)
adults_df = scoring.add_diversity_score(adults_df, 'factor_set', all_items)
adults_df = analysis.normalize_outcome_columns(adults_df)
analysis.plot_outcome_correlations(adults_df, outcome_cols, feature_cols)
```

```
/root/work/scoring.py:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
  df[['plant_based_score', 'animal_based_score']] = df[factor_col].apply(compute_scores)
```



```python
# Scatter plot with correlation for adults_df

analysis.plot_scatter_with_correlation(adults_df, feature_cols, outcome_cols)
```


```python
# Merging adult dataframe with mental health outcome dataframes

ed_merged_df = pd.merge(diet_df, ed_clean_df, on=['ISO3', 'Year'])
ed_adults_df = ed_merged_df[ed_merged_df['Oldest age'] >= 18]
print(ed_adults_df.columns)

schiz_merged_df = pd.merge(diet_df, schiz_cleaned_df, on=['ISO3', 'Year'])
schiz_adults_df = schiz_merged_df[schiz_merged_df['Oldest age'] >= 18]
print(schiz_adults_df.columns)

mh_clean_merged_df = pd.merge(diet_df, mh_clean_df, on=['ISO3', 'Year'])
mh_adults_df = mh_clean_merged_df[mh_clean_merged_df['Oldest age'] >= 18]
print(mh_adults_df.columns)
```

```
Index(['ISO3', 'Year', 'Youngest age', 'Oldest age', 'factor_set',
       'eating_disorder'],
      dtype='object')
Index(['ISO3', 'Year', 'Youngest age', 'Oldest age', 'factor_set',
       'schizophrenia'],
      dtype='object')
Index(['ISO3', 'Year', 'Youngest age', 'Oldest age', 'factor_set', 'bipolar',
       'anxiety', 'drug_use', 'depression', 'alcohol_use'],
      dtype='object')
```

```python
# Normalizing scores, maintaing rows between 0 to 1
# Skipping columns for dataframes that don't exist

ed_adults_df = analysis.normalize_outcome_columns(ed_adults_df)
schiz_adults_df = analysis.normalize_outcome_columns(schiz_adults_df)
mh_adults_df = analysis.normalize_outcome_columns(mh_adults_df)
```

```
schizophrenia not found in dataframe, skipping.
bipolar not found in dataframe, skipping.
anxiety not found in dataframe, skipping.
drug_use not found in dataframe, skipping.
depression not found in dataframe, skipping.
alcohol_use not found in dataframe, skipping.
bipolar not found in dataframe, skipping.
eating_disorder not found in dataframe, skipping.
anxiety not found in dataframe, skipping.
drug_use not found in dataframe, skipping.
depression not found in dataframe, skipping.
alcohol_use not found in dataframe, skipping.
schizophrenia not found in dataframe, skipping.
eating_disorder not found in dataframe, skipping.
```

```python
# Scorings for eating disorder adult dataframe; dropping duplicate rows

ed_adults_df = scoring.add_diet_scores(ed_adults_df, plant_based, animal_based, 'factor_set')
ed_adults_df = scoring.compute_processed_diet_score(ed_adults_df, 'factor_set', processed_items, unprocessed_items)
ed_adults_df = scoring.add_diversity_score(ed_adults_df, 'factor_set', all_items)
ed_adults_df.drop_duplicates(["ISO3", "Year"], inplace = True)
ed_adults_df
```

	ISO3 object JPN 5.1% USA 4.1% 147 others 90.8%	Year int64 1990 - 2017	Youngest age flo... 0.0 - 70.0	Oldest age float64 18.0 - 120.0	factor_set object {'Non-Star... 16.3% {'Beans and... 8% 153 others 75.7%	eating_disorder f... 0.079669 - 0.725...	plant_based_score 0.0 - 1.0	a 0
0	ALB	2008	0	98	{'Beans and Legu...	0.154945	0.3333333333	
6	ARE	2009	6	50	{'Dietary Sodium',...	0.262274	0.1111111111	
12	ARE	2003	18	90	{'Non-Starchy Ve...	0.289831	0.2222222222	
21	ARG	2004	20	49	{'Dietary Sodium',...	0.362547	0.6666666667	
27	ARM	2005	0	49	{'Beans and Legu...	0.135408	0.5555555556	
30	ARM	2001	18	99	{'Unprocessed Re...	0.122517	0.2222222222	
36	AUS	1995	0	100	{'Beans and Legu...	0.725294	0.6666666667	
42	AUT	2010	6	81	{'Dietary Sodium',...	0.667758	0.7777777778	
48	AUT	2007	7	100	{'Dietary Sodium',...	0.65167	0.1111111111	
54	AUT	2005	19	64	{'Total Milk', 'Fruit...	0.638828	0.3333333333	

412 rows, 12 cols 10 / page Page 1 of 42

```python
# Scorings for schizophrenia adult dataframe; dropping duplicate rows

schiz_adults_df = scoring.add_diet_scores(schiz_adults_df, plant_based, animal_based, 'factor_set')
schiz_adults_df = scoring.compute_processed_diet_score(schiz_adults_df, 'factor_set', processed_items, unprocessed_items)
schiz_adults_df = scoring.add_diversity_score(schiz_adults_df, 'factor_set', all_items)
schiz_adults_df.drop_duplicates(["ISO3", "Year"], inplace = True)
schiz_adults_df
```

	ISO3 object	Year int64	Youngest age flo...	Oldest age float64	factor_set object	schizophrenia flo...	plant_based_score	a
	JPN 5.6% RUS 3.4% 142 others 91%	1990 - 2016	0.0 - 70.0	18.0 - 120.0	{'Non-Star... 17.7% {'Beans an... 8.5% 138 others 73.8%	0.041837429999...	0.0 - 1.0	0
0	ALB	2008	0	98	{'Beans and Legu...	0.198306	0.3333333333	
5	ARE	2009	6	50	{'Dietary Sodium',...	0.213088	0.1111111111	
9	ARE	2003	18	90	{'Non-Starchy Ve...	0.210353	0.2222222222	
13	ARG	2004	20	49	{'Dietary Sodium',...	0.196537	0.6666666667	
17	ARM	2005	0	49	{'Beans and Legu...	0.19195	0.5555555556	
20	ARM	2001	18	99	{'Unprocessed Re...	0.189543	0.2222222222	
23	AUS	1995	0	100	{'Beans and Legu...	nan	0.6666666667	
25	AUT	2010	6	81	{'Dietary Sodium',...	0.256201	0.7777777778	
27	AUT	2007	7	100	{'Dietary Sodium',...	0.25644	0.1111111111	
29	AUT	2005	19	64	{'Total Milk', 'Fruit...	0.256589	0.3333333333	

378 rows, 12 cols 10 / page Page 1 of 38

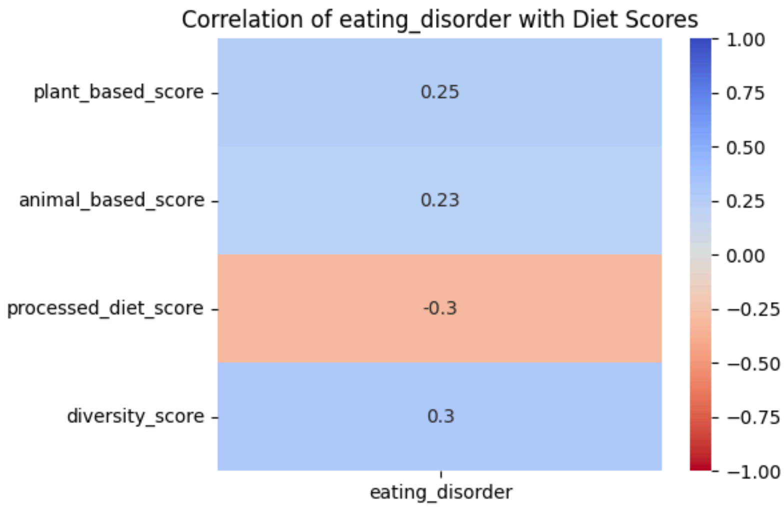
```python
# Scorings for all mental health adult dataframe; dropping duplicate rows

mh_adults_df = scoring.add_diet_scores(mh_adults_df, plant_based, animal_based, 'factor_set')
mh_adults_df = scoring.compute_processed_diet_score(mh_adults_df, 'factor_set', processed_items, unprocessed_items)
mh_adults_df = scoring.add_diversity_score(mh_adults_df, 'factor_set', all_items)
mh_adults_df.drop_duplicates(["ISO3", "Year"], inplace = True)
mh_adults_df
```

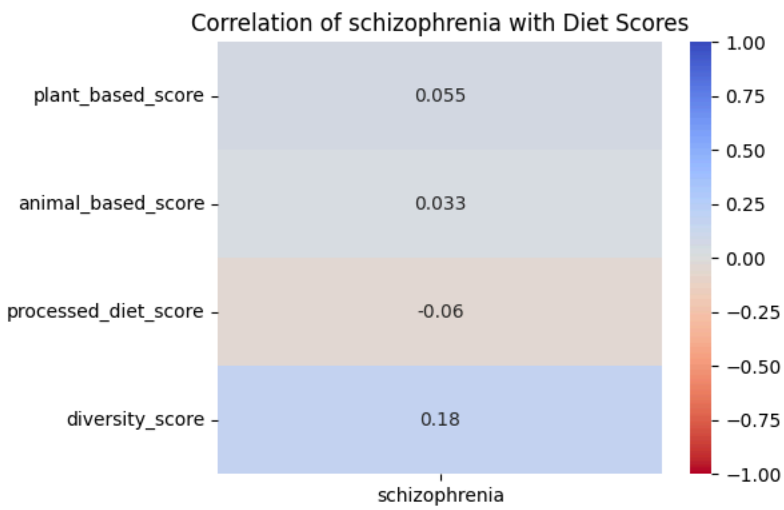
	ISO3 object	Year int64	Youngest age flo...	Oldest age float64	factor_set object	bipolar float64	anxiety float64	d
	JPN 5.7% RUS 3.6% 135 others 90.7%	1990 - 2017	0.0 - 70.0	18.0 - 120.0	{'Non-Star... 16.1% {'Beans and... 9% 139 others 74.9%	0.0100413 - 0.99...	0.025033029999...	0
0	ALB	2008	0	98	{'Beans and Legu...	0.7024	0.03390168	
3	ARG	2004	20	49	{'Dietary Sodium',...	0.769343	0.06256249	
5	ARM	2005	0	49	{'Beans and Legu...	0.714894	0.02591655	
6	ARM	2001	18	99	{'Unprocessed Re...	0.713487	0.02587215	
8	AUS	1995	0	100	{'Beans and Legu...	0.01142137	0.06478469	
10	AUT	2010	6	81	{'Dietary Sodium',...	0.939911	0.05352376	
12	AUT	2007	7	100	{'Dietary Sodium',...	0.94009	0.0535486	
14	AUT	2005	19	64	{'Total Milk', 'Fruit...	0.940236	0.05356742	
16	AUT	1999	0	100	{'Total Milk', 'Suga...	0.94735	0.05361666	
19	AZE	2006	0	49	{'Yogurt', 'Beans a...	0.683228	0.02561173	

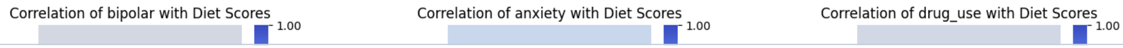
366 rows, 16 cols 10 / page Page 1 of 37

```
# Keeping for historical purposes. Code was used to analyze data and decide how to normalize the datasets.

# def print_column_ranges(df):

#     outcome_cols = ['schizophrenia', 'bipolar', 'eating_disorder', 'anxiety', 'drug_use', 'depression', 'alcohol_use']
#     for col in outcome_cols:
#         try:
#             # Attempt to convert column to numeric (ignore errors for non-numeric columns)
#             col_data = pd.to_numeric(df[col], errors='coerce')
#             max_val = col_data.max()
#             min_val = col_data.min()
#             print(f"{col}: {max_val} to {min_val}")
#         except Exception as e:
#             # In case anything unexpected happens
#             print(f"{col}: Error processing column ({e})")

# # mental_health_df
# print_column_ranges(ed_adults_df)
# print_column_ranges(schiz_adults_df)
# print_column_ranges(mh_adults_df)
```

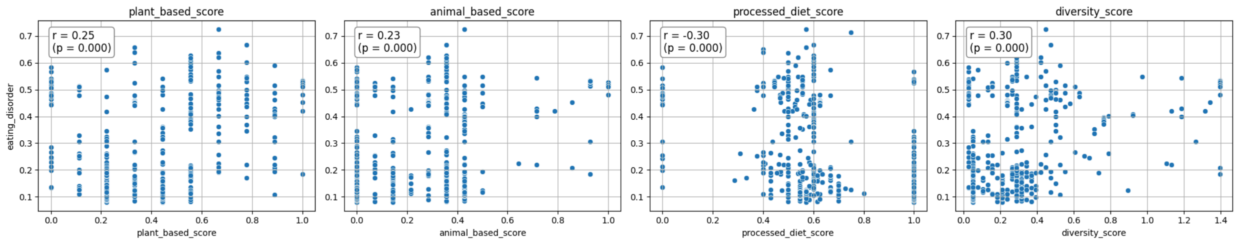
```python
# Correlation matrices
analysis.plot_outcome_correlations(ed_adults_df, outcome_cols, feature_cols)
analysis.plot_outcome_correlations(schiz_adults_df, outcome_cols, feature_cols)
analysis.plot_outcome_correlations(mh_adults_df, outcome_cols, feature_cols)
```

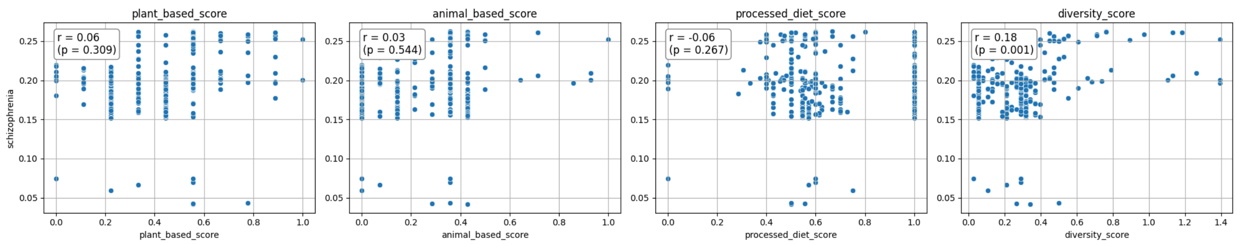


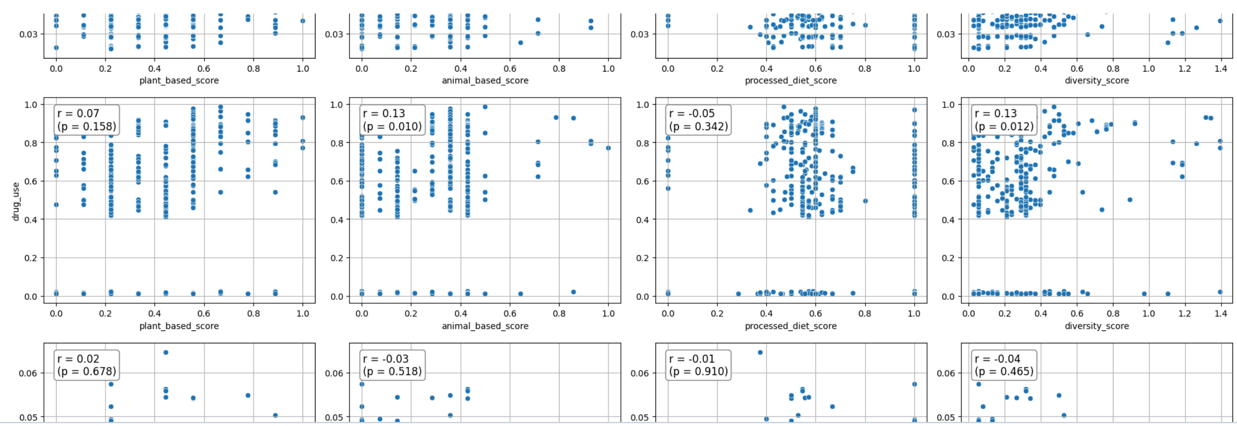





```python
# Scatter plot with correlation
analysis.plot_scatter_with_correlation(ed_adults_df, feature_cols, outcome_cols)
analysis.plot_scatter_with_correlation(schiz_adults_df, feature_cols, outcome_cols)
analysis.plot_scatter_with_correlation(mh_adults_df, feature_cols, outcome_cols)
```

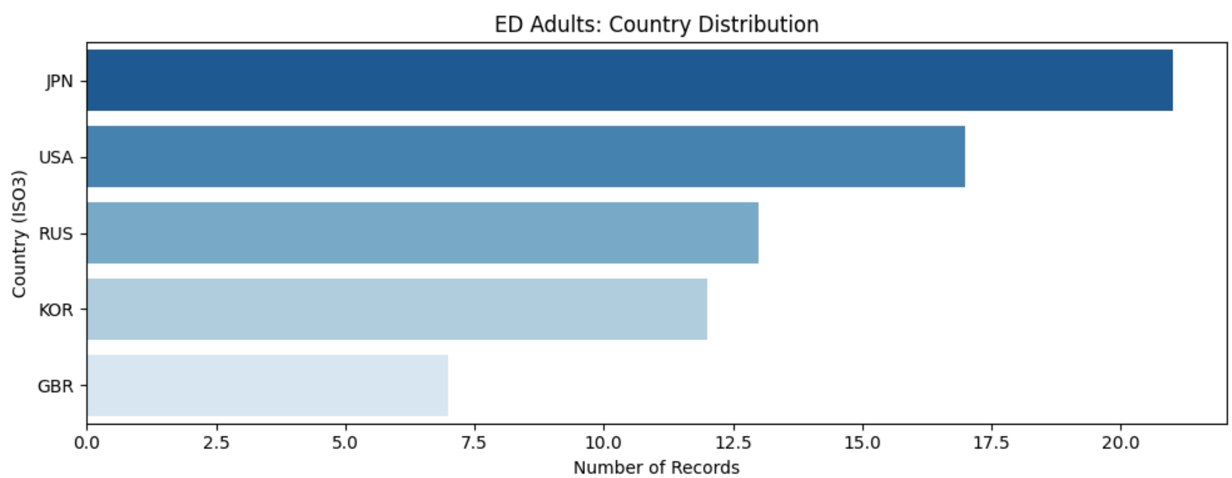




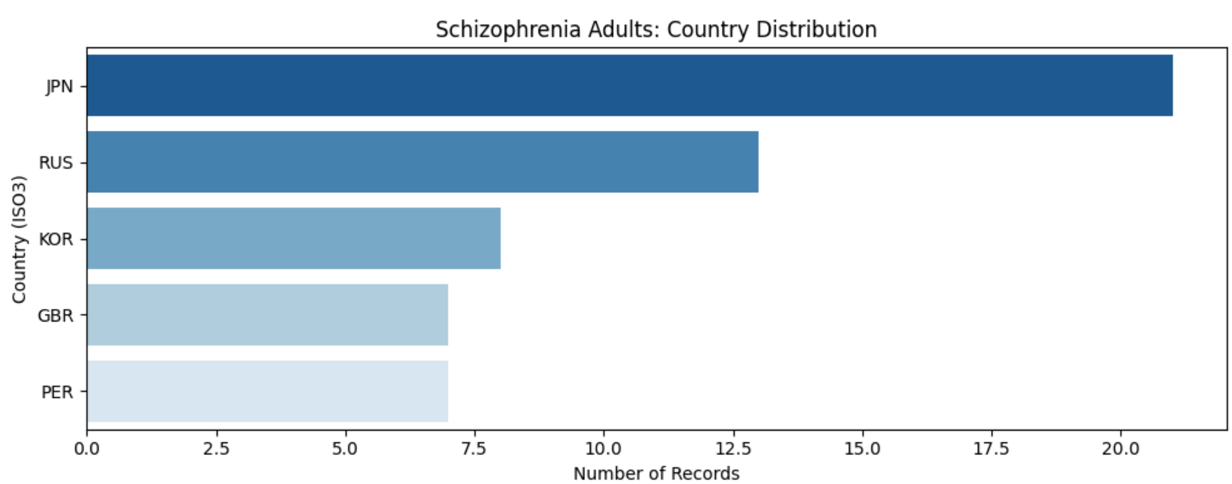


```python
# Top 5 country row distribution, with a minimum of 5 entries, for each dataframe
analysis.plot_country_distribution(ed_adults_df, title="ED Adults: Country Distribution", top_n=5)
analysis.plot_country_distribution(schiz_adults_df, title="Schizophrenia Adults: Country Distribution", top_n=5)
analysis.plot_country_distribution(mh_adults_df, title="Mental Health Adults: Country Distribution", top_n=5)
```

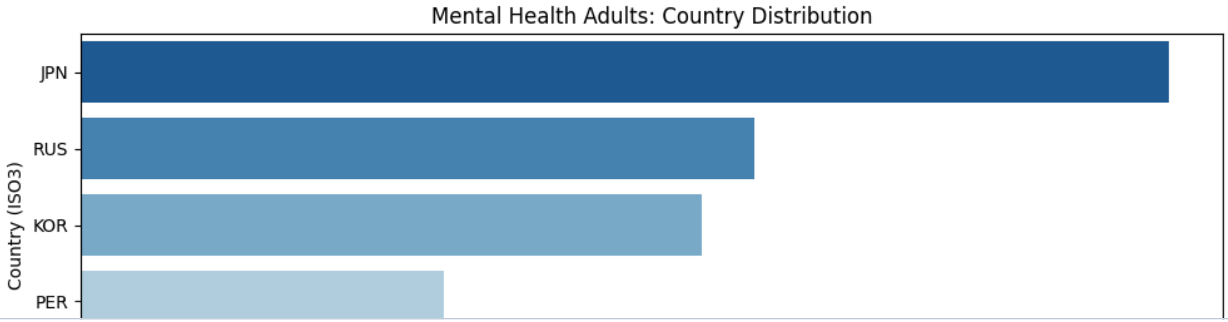
```
/root/work/analysis.py:143: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(x=country_counts.values, y=country_counts.index, palette='Blues_r')
```




```
/root/work/analysis.py:143: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(x=country_counts.values, y=country_counts.index, palette='Blues_r')
```



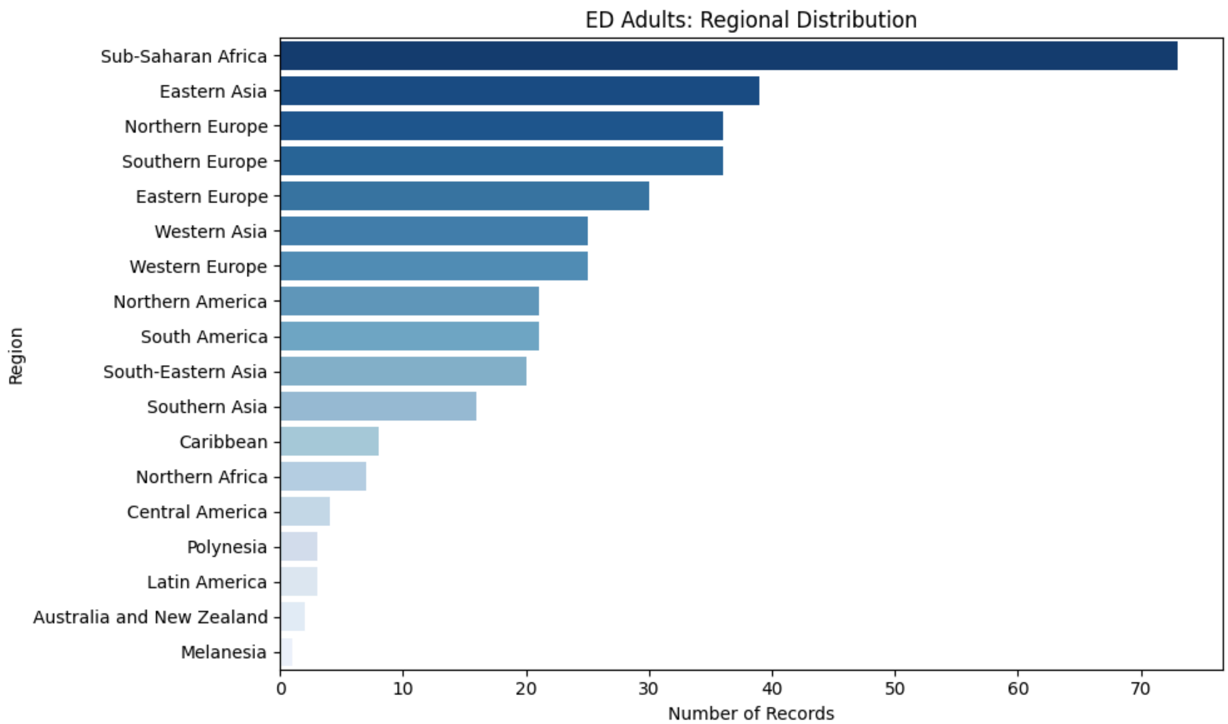

```
/root/work/analysis.py:143: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(x=country_counts.values, y=country_counts.index, palette='Blues_r')
```




```
# Grouping ISO3 codes into regions
iso3_to_region = {
    # Africa
    'DZA':'Northern Africa', 'EGY':'Northern Africa','LBY':'Northern Africa','MOR':'Northern Africa','SDN':'Northern Afri
    'AGO':'Sub-Saharan Africa','BWA':'Sub-Saharan Africa','BDI':'Sub-Saharan Africa','CMR':'Sub-Saharan Africa','CAF':'Su
    'TCD':'Sub-Saharan Africa','COG':'Sub-Saharan Africa','COD':'Sub-Saharan Africa','SWZ':'Sub-Saharan Africa','ETH':'Su
    'GAB':'Sub-Saharan Africa','GHA':'Sub-Saharan Africa','GIN':'Sub-Saharan Africa','CIV':'Sub-Saharan Africa','KEN':'Su
    'LSO':'Sub-Saharan Africa','LBR':'Sub-Saharan Africa','MDG':'Sub-Saharan Africa','MWI':'Sub-Saharan Africa','MLI':'Su
    'MOZ':'Sub-Saharan Africa','NAM':'Sub-Saharan Africa','NER':'Sub-Saharan Africa','NGA':'Sub-Saharan Africa','RWA':'Su
    'SEN':'Sub-Saharan Africa','SLE':'Sub-Saharan Africa','SOM':'Sub-Saharan Africa','ZAF':'Sub-Saharan Africa','TZA':'Su
    'UGA':'Sub-Saharan Africa','ZMB':'Sub-Saharan Africa','ZWE':'Sub-Saharan Africa',
    # Americas
    'CAN':'Northern America','USA':'Northern America',
    'BLZ':'Central America','CRI':'Central America','SLV':'Central America','GTM':'Central America','HND':'Central Americ
    'PAN':'Central America',
    'ARG':'South America','BOL':'South America','BRA':'South America','CHL':'South America','COL':'South America','ECU':'
    'PRY':'South America','PER':'South America','URY':'South America','VEN':'South America',
    'ATG':'Caribbean','BHS':'Caribbean','BRB':'Caribbean','CUB':'Caribbean','DOM':'Caribbean','GRD':'Caribbean','HTI':'Ca
    'JAM':'Caribbean','KNA':'Caribbean','LCA':'Caribbean','VCT':'Caribbean',
    'MEX':'Latin America',  # Mexico sometimes grouped in Latin America overall
    # Asia
    'CHN':'Eastern Asia','JPN':'Eastern Asia','KOR':'Eastern Asia','MNG':'Eastern Asia',
    'IND':'Southern Asia','PAK':'Southern Asia','BGD':'Southern Asia','NPL':'Southern Asia','LKA':'Southern Asia','BTN':'
    'IDN':'South-Eastern Asia','MYS':'South-Eastern Asia','PHL':'South-Eastern Asia','SGP':'South-Eastern Asia','THA':'So
    'AZE':'Western Asia','ARM':'Western Asia','IRN':'Western Asia','IRQ':'Western Asia','ISR':'Western Asia','JOR':'Weste
    # Europe
    'ALB':'Southern Europe','AND':'Southern Europe','BIH':'Southern Europe','HRV':'Southern Europe','GRC':'Southern Europ
    'AUT':'Western Europe','BEL':'Western Europe','FRA':'Western Europe','DEU':'Western Europe','IRL':'Western Europe','L
    'BLR':'Northern Europe','DNK':'Northern Europe','EST':'Northern Europe','FIN':'Northern Europe','ISL':'Northern Europ
    'CZE':'Eastern Europe','HUN':'Eastern Europe','POL':'Eastern Europe','SVK':'Eastern Europe','UKR':'Eastern Europe','B
    # Oceania
    'AUS':'Australia and New Zealand','NZL':'Australia and New Zealand','FJI':'Melanesia','PNG':'Melanesia','SLB':'Melane
    'KIR':'Micronesia','NRU':'Micronesia','TKL':'Micronesia','WSM':'Polynesia','TON':'Polynesia','TUV':'Polynesia','NIU':
    # Antarctica (optional)
    'ATA':'Antarctica'
}
```

```python
# Where most of the regional data is coming from
analysis.plot_region_distribution(ed_adults_df, region_map=iso3_to_region, title="ED Adults: Regional Distribution")
analysis.plot_region_distribution(schiz_adults_df, region_map=iso3_to_region, title="Schizophrenia: Regional Distribution
analysis.plot_region_distribution(mh_adults_df, region_map=iso3_to_region, title="Mental Health: Regional Distribution")
```

```
/root/work/analysis.py:181: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(x=region_counts.values, y=region_counts.index, palette='Blues_r')
```




```
/root/work/analysis.py:181: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

  sns.barplot(x=region_counts.values, y=region_counts.index, palette='Blues_r')
```

Schizophrenia: Regional Distribution