In [114…]:
```
#MADE IN JUPYTER LAB - ANDREAS MAYR
#FOR UMD IC 2024

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [22]:
```
main_df = pd.read_csv("FF_SR_ data.csv")
categories_df = pd.read_csv("food_category_id table.csv")
nutrient_df = pd.read_csv("nutrient_id table.csv")
```

In [26]:
```
main_df.head()
```

Out[26]:

	FF_NDB	SR_NDB	food_category_id	FF Food description	SR Food description	Nutrient_id	rank	FF_Com
0	16158	16158	Legumes and Legume Products	Hummus, commercial	Hummus, commercial	Magnesium, Mg	5500	Magnesi
1	16158	16158	Legumes and Legume Products	Hummus, commercial	Hummus, commercial	Fatty acids, total saturated	9700	Fatt total sa
2	16158	16158	Legumes and Legume Products	Hummus, commercial	Hummus, commercial	Iron, Fe	5400	
3	16158	16158	Legumes and Legume Products	Hummus, commercial	Hummus, commercial	Water	100	
4	16158	16158	Legumes and Legume Products	Hummus, commercial	Hummus, commercial	Selenium, Se	6200	Selen

5 rows × 22 columns

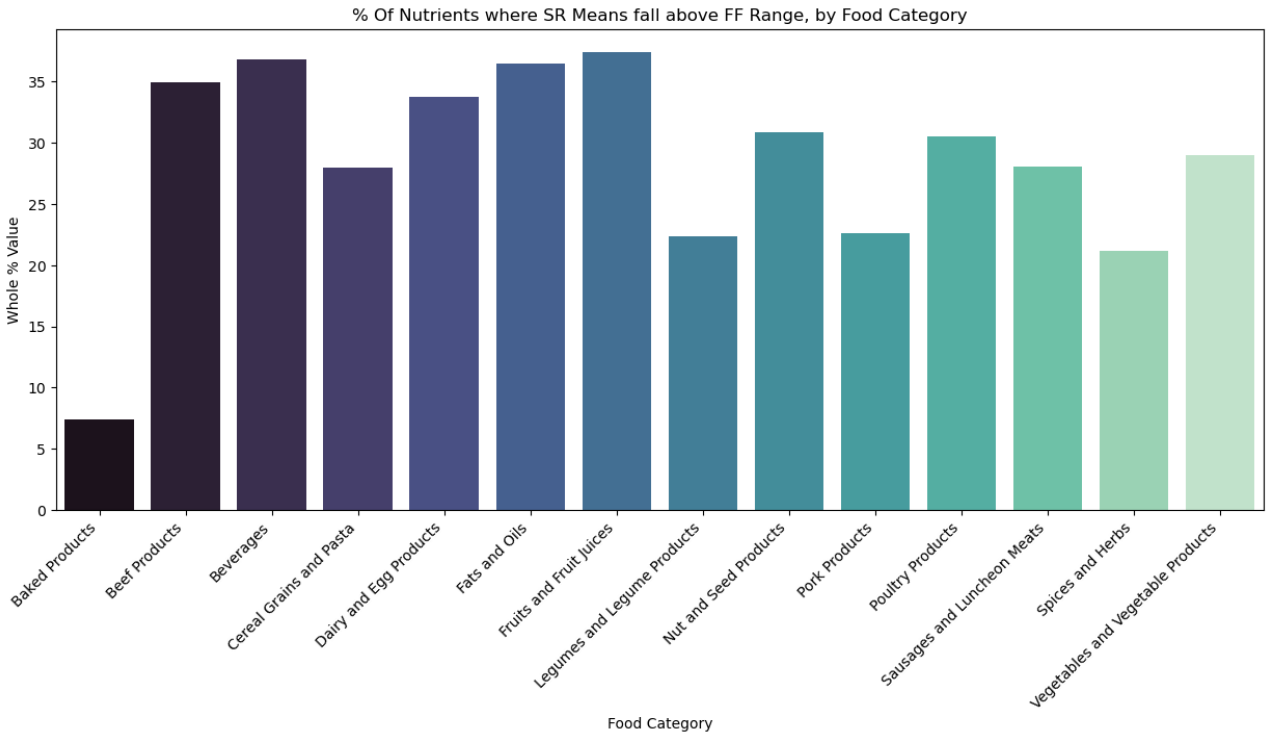
In [25]:
```
#Q1

#create new column for delta of mean per 100g
main_df['Mean per 100g percentage change'] = ((main_df['FF Mean per 100g'] -

# Replace food category IDs with names
food_id_to_desc = pd.Series(categories_df.description.values,index=categorie
main_df['food_category_id'] = main_df['food_category_id'].map(food_id_to_des

# Replace nutrient IDs with names
nutrient_id_to_name = pd.Series(nutrient_df.name.values,index=nutrient_df.nu
main_df['Nutrient_id'] = main_df['Nutrient_id'].map(nutrient_id_to_name)

#round values
main_df['Mean per 100g percentage change'] = main_df['Mean per 100g percenta

#show means of the change of the means grouped by category ID and nutrient I
mean_by_category_and_nutrient = main_df.groupby(['food_category_id', 'Nutrie

q1_df = mean_by_category_and_nutrient.dropna().copy()
```

In [145…
```
#Q3

#Create base DF for q3 from main_df
q3_df = main_df.copy()

#Drop Rows that contain NaN values for 'FF Min'/'FF Max'
q3_df.dropna(subset=['FF Min'], how='all', inplace=True)

# Create column indicating if SR mean per 100g is above ff min
q3_df['SR_above_FF_Min'] = q3_df['SR Mean per 100g'] >= q3_df['FF Min']

# Create column indicating if SR mean per 100g is below ff max
q3_df['SR_below_FF_Max'] = q3_df['SR Mean per 100g'] <= q3_df['FF Max']

# Create column indicating if SR is above ff max
q3_df['SR_above_FF_Max'] = q3_df['SR Mean per 100g'] >= q3_df['FF Max']

# Create column indicating if SR is above ff max
q3_df['SR_below_FF_Min'] = q3_df['SR Mean per 100g'] <= q3_df['FF Min']

# Create column indicating if SR mean is within range (True, True)
q3_df['SR_Mean_within_FF_Range'] = q3_df['SR_above_FF_Min'] & q3_df['SR_belo

#Create column if SR Mean is Above FF Range (indicating standards have gone


within_range_by_food_group = q3_df.groupby(['food_category_id'])['SR_Mean_wi
above_range_by_food_group = q3_df.groupby(['food_category_id'])['SR_above_FF
```

Out[145]:
```
food_category_id
Baked Products                          1.000000
Beef Products                           0.556604
Beverages                               0.736842
Cereal Grains and Pasta                 0.555944
Dairy and Egg Products                  0.719481
Fats and Oils                           0.428571
Fruits and Fruit Juices                 0.656489
Legumes and Legume Products             0.506289
Nut and Seed Products                   0.578534
Pork Products                           0.735849
Poultry Products                        0.597222
Sausages and Luncheon Meats             0.573171
Spices and Herbs                        0.732394
Vegetables and Vegetable Products       0.703759
Name: SR_Mean_within_FF_Range, dtype: float64
```

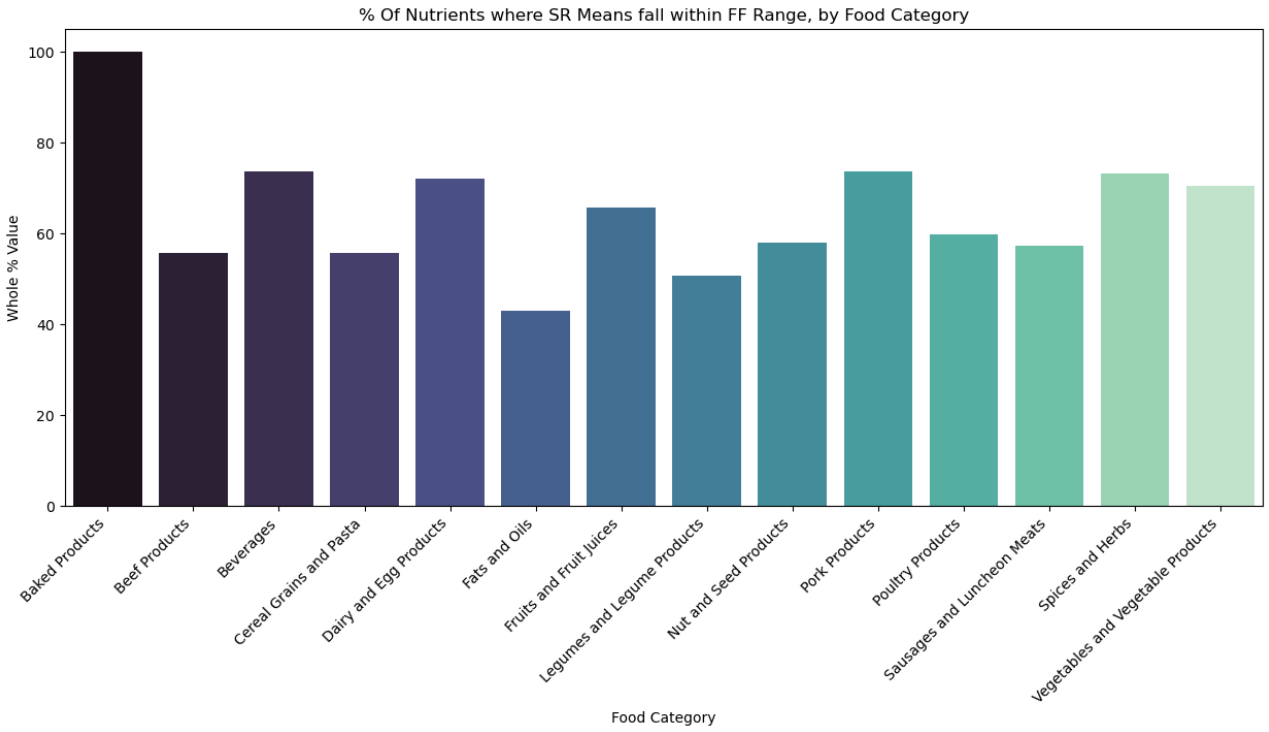
In [150…

```python
#Q3

#create df to calculate percentage of nutrients within range by category
SR_above_range = above_range_by_food_group.reset_index().copy()

SR_above_range['SR_above_FF_Max'] = SR_above_range['SR_above_FF_Max']*100

plt.figure(figsize=(15, 6))
sns.barplot(x='food_category_id', y='SR_above_FF_Max', data=SR_above_range,
plt.xlabel('Food Category')
plt.ylabel('Whole % Value')
plt.title('% Of Nutrients where SR Means fall above FF Range, by Food Categc

plt.xticks(rotation=45, ha='right')

#plt.tight_layout()

plt.show()
```



In [149…

```
#Q3

#create df to calculate percentage of nutrients within range by category
SR_within_range = within_range_by_food_group.reset_index().copy()

SR_within_range['SR_Mean_within_FF_Range'] = SR_within_range['SR_Mean_within

plt.figure(figsize=(15, 6))
sns.barplot(x='food_category_id', y='SR_Mean_within_FF_Range', data=SR_withi
plt.xlabel('Food Category')
plt.ylabel('Whole % Value')
plt.title('% Of Nutrients where SR Means fall within FF Range, by Food Categ

plt.xticks(rotation=45, ha='right')

#plt.tight_layout()

plt.show()
```



In [129…

```
fats_and_oils_df[fats_and_oils_df['Nutrient_id'] == 'Tocopherol, beta']
```

Out[129]:

	food_category_id	Nutrient_id	FF Min	FF Max	SR Mean per 100g	gross_exceedance	normalized_exce
1648	Fats and Oils	Tocopherol, beta	1.57	2.20	0.90	-1.30	
1620	Fats and Oils	Tocopherol, beta	0.00	0.00	0.01	0.01	
2211	Fats and Oils	Tocopherol, beta	0.00	1.29	0.46	-0.83	

In [151…

```
#CHANGE AS NEEDED FOR EACH FOOD CATEGORY

#q3_df[q3_df['food_category_id'] == 'Fats and Oils'][['food_category_id','Nu

#Filter for food category
beef = q3_df[q3_df['food_category_id'] == 'Beef Products'][['food_category_i

# create gross exceedance column
beef['gross_exceedance'] = beef['SR Mean per 100g'] - beef['FF Max']

# create normalized exceedance column
beef['normalized_exceedance_pct'] = beef['gross_exceedance'] / (beef['FF Max

#mask negative numbers (not exceedance)
beef.loc[beef['normalized_exceedance_pct'] < 0, 'normalized_exceedance_pct']
#fats_and_oils_df

beef_exceedance = beef.groupby('Nutrient_id')[['Nutrient_id','normalized_exc
beef_exceedance
```

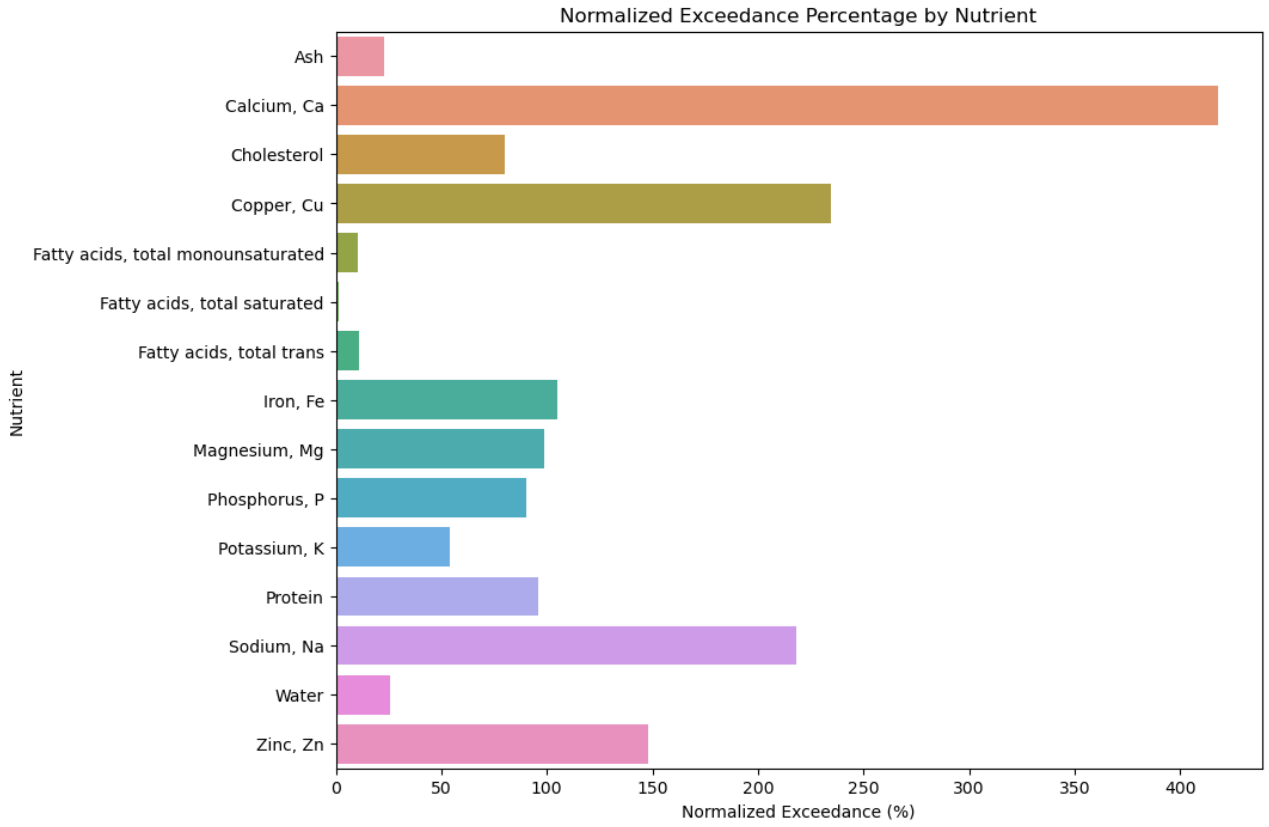
```
/var/folders/hd/zrgq78qs0jndrkwy7f9gnzfr0000gn/T/ipykernel_64119/1882247073.
py:16: FutureWarning: The default value of numeric_only in DataFrameGroupBy.
mean is deprecated. In a future version, numeric_only will default to False.
Either specify numeric_only or select only columns which should be valid for
the function.
  beef_exceedance = beef.groupby('Nutrient_id')[['Nutrient_id','normalized_e
xceedance_pct']].mean()
```

Out[151]:

Nutrient_id	normalized_exceedance_pct
Ash	22.727273
Calcium, Ca	417.669665
Cholesterol	80.246914
Copper, Cu	234.330736
Fatty acids, total monounsaturated	10.643777
Fatty acids, total polyunsaturated	NaN
Fatty acids, total saturated	1.272727
Fatty acids, total trans	11.111111
Iron, Fe	104.918033
Magnesium, Mg	98.632479
Manganese, Mn	NaN
Phosphorus, P	90.270445
Potassium, K	53.958066
Protein	96.050000
Sodium, Na	217.864467
Total lipid (fat)	NaN
Water	25.654450
Zinc, Zn	148.026316

In [152…

```
#CHANGE AS NEEDED FOR FOOD CATEGORY, MUST USE CELL ABOVE AS WELL

beef_exceedance_reset = beef_exceedance[beef_exceedance['normalized_exceedan
# Create the barplot
plt.figure(figsize=(10, 8))  # Adjust the size as needed
sns.barplot(data=beef_exceedance_reset, x='normalized_exceedance_pct', y='Nu

# Set the title and labels
plt.title('Normalized Exceedance Percentage by Nutrient')
plt.xlabel('Normalized Exceedance (%)')
plt.ylabel('Nutrient')

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



In [ ]:
```
## Export to CSV file
q1_df.to_csv('/Users/andreasmayr/Desktop/MeanByCategoryAndNutrient.csv', ind
```

In [11]:
```
#Export to CSV file
q3_df.to_csv('/Users/andreasmayr/Desktop/Q3_data.csv', index=False)
```

In [ ]:
```
#locate field and then by index

q3_df[q3_df['food_category_id'] == 'Beverages'][['Nutrient_id','SR Mean per
#main_df.loc[INDEX]
```