Nutrition data analysis of common foods: Exploring the viability of common foods for both protein and fiber requirements.

1. Research background and Objectives and Aims:

1.1 Introduction

Fiber content in protein-rich foods

Protein is an important macronutrient that is required in our daily diet. Fiber is also an important, but often neglected part of a balanced diet. It is also a macronutrient with an alarmingly low amount of consumption [1]. It has been found that excess protein consumption without including fiber in the diet has several negative health consequences. This research is inspired by the work of Xu et. al. (2016) [2] where it is reported that a high intake of protein relative to fiber leads to an increased risk of cardiovascular and kidney diseases.

The question that arises (and is the research question for this project) is the suitability of a particular food item for an appropriate amount of consumption with respect to both protein and fiber. Protein-rich foods, while important as the source of required protein, do not always contain adequate amount of fiber. Thus, we want to identify foods that are rich in both protein and fiber. In other words, we want to find foods that will most efficiently provide not only the protein requirements of our diets, but also the required fiber.

To answer the research question, we first obtain a database of common food items with their macronutrient contents. Among these food items, we select the most protein-rich foods and analyse the fiber-content of each of these food items to arrive at a conclusion on their suitability as the source of fiber in our daily diet.

Additionally, the price of food to satisfy the daily requirements is of interest since we would like to identify food items that will satisfy the requirements for the lowest dollar value. In this study, we look at the price of both protein and fiber contained in protein rich foods.

The final objective of the study is to recommend a short list of food items. To my knowledge, a study specifically looking at the fiber content of protein-rich foods is yet to be conducted.

The following questions are addressed in this study:

- What are the top sources of protein among common foods?
- Which protein-rich foods, are also rich in fiber?

- What is the most economical food for simultaneous consumption of both protein and fiber daily requirements?
- Recommend a list of food items that satisfy 3 conditions (high in protein, high in fiber, economical to purchase)

Main Research Question- Create a short list of recommended top food items that are not only rich in fiber and protein but are also cost-effective for daily consumption.

1.2 Aims and objectives:

OBJECTIVES:

- Explore the relationship between food-groups and the macronutrient profile(fiber and protein specifically).
  - For example, answer questions like "which food group to choose for a protein rich profile.
- Look for relationship between macronutrients.
  - Relationships like, 'Are food rich in protein also rich in fiber?'
- Create a numerical macronutrient profile of each food.
- Determine the characterisitics of food-groups.
  - Categorise data calculated for single foods into respective food groups, and then determine the characterisitics of that food group.
- Analyse individual food-item and determine its characteristics and see if it fits to our solution to research question.
- Calculate macronutrient proportion of each food.
- Determine if they fit in our final list of recommended foods based on the data and calculations. -Carry out a price analysis on each food to test their viability as a macronutrient source(fiber and protein specifically).

AIMS:

- Determine the type of raw data that we would need to fulfill the above objectives and also find a reliable source for these raw data.
- Use webscraping to acquire identified data and then store it in reusable formats.
- Organise and standardise the data, to make it analysis-ready.
- Create functions that we can use on these datas for further analysis.
- Create data filtering systems to filter out specific data.
- Explore the data through statistical analysis.
- Carry out pattern analysis and visualise the data.

1.3 Data:

1.3.1 Data requirements:

We needed a data that gave us a list of food-items. the food-item's macronutrient profile (protein content, carbohydrate content, fiber content,and fat content), and its categorization into food-

group and food-subgroup.

Food-item list: We are only concerned with the most basic form of foods with very minimal processing available and found in all of north america.

- Level of processing: None. Example (apples are included but apple derived products such as apple-sauce, candied apples, dried apples, apple-juice are not.)
- Geography: Easily found in North america.Example: (basic fruits like bananas and apple are included but fruits like mangosteen found mostly in the tropics arent included.)
- Available: Easily available in the markets.

Macronutrient profile

- Macronutrient proportion (written in forms such as: 20 mg of protein per 100 grams of the food)
- Proportions in appropriate and convertible units such as miligrams, grams and so on.
- Macronutrient profile data sourced from chemical analysis done by reliable sources such as national labs.

Classification of the food-item

- Information about the category and subcategory that the food-item belongs to.
- Precise and informative categories such as "fruits" and "vegetables" rather than vague or descriptive categories.

Price of food-items

- This will be sourced from north-america relevant national bodies.
- In appropriate units such as dollars and per-pound, per-kg.

1.3.2 Websites considered for sourcing the data:

1. U.S. DEPARTMENT OF AGRICULTURE:

   pros:

   - Highest standard for nutrition data.

   cons:

   - Organization of the data was such that, web-scraping to acquire the information was excessively time-consuming.
   - Too much information: Only a very low percentage of data was useful for this study.

1. Australian Food Composition Database

   pros:

   - Reliable nutrition data.

   cons:

- Incorrect geographical region

1. www.foodb.ca

   pros:

   - Uses peer reviewed literature for its data.
   - FooDB is offered to the public as a freely available resource.
   - Government of Canada funded project.
   - Focused on the correct geographical region: North America.

   cons:

   - Elaborated further down on subheading "1.3.4 Limitations"

1.3.3 Final decision for the source chosen and Why?:

Source: www.foodb.ca. This was selected as our final source for data. All the data in this study has been acquired through web-scraping this website.

Why is this data suitable for our project?

This data contains all the information we need for our analysis(As listed in 1.3.1). It is from a reliable source. It is free for public use. This data is for foods available in the Northern America, which is our geographical area of interest for this project. Time calculation for web-scraping this website is not excessive. It is already a somewhat filtered form of data which makes our study easier.We aim to provide a short list of foods after looking at them through 3 lenses: protein content, fiber content, and cost-effectiveness. The data we get from this source, helps us form 2 of these lenses very well, as this data source gives us numerical values pertaining to each foods such as- protein in mg per 100 gram of food, fiber in mg per 100 gram of food, category of food and so on.

Furthermore, this particular source of data has results from lab studies without human subjects and we can assume that the data has been collected in ethical manner and has no personal data.

1.3.4 Limitations:

- Lack of information on the state of food, such as ripeness of fruits, maturity of seeds, shelled/unshelled, variety and species of food.
- Only the average macronutrient data was taken and the upper and lower bounds were ignored. There could have been a huge deviation from the average but this was considered beyond the scope of our study.
- This project aims to recommend a small list of foods with good protein and fiber ratios and which are also most cost-effective to meet this macronutrient need. However, this recommendation does not include the implications of cooking and processing techniques that will be further utilised by the consumer.
- A singular average price data has been used (acquired through usda grocery reports). The variation of price points across the north american continent has not been considered. This

allows for only a general recommendation and not local area specific recommendation.
- Fluidity of food prices have not been considered: Prices of food-items change all the time, but we are using a constant value for this, for now. This can be incorporated as a future project where we write an algorithm that periodically scrapes the web for updated prices of food.

1.3.5 Ethical considerations:

- Modern agricultural processes can often be harmful to the environment and biodiversity and this harm is not equal across different food-items. Recommendation of food without this consideration may be unethical in terms of environmental damage. Example: Almonds farming needs 15 gallons of water to produce just 16 almonds, making almonds one of the most water-intensive crops in the california state of USA.[3]
- The data source used in this study is freely available to the public and is open for non-commercial use such as this study. Thus, no ethical violation has occured in obtaining and using the data.
- Additionally, no personal information has been used in this study.
- Also, I, as an author of this study, have no conflict of interest.

2. Web-scraping to fetch data

- library used : selenium
- website that was scraped : https://fooddb.ca/

Steps involved in the webscraping:

1. Open the url.
2. Interact the url to display 100 items in the list in place of just 10.
3. Extract all the numerical data in the page.
4. Extract a list of useful urls that will be opened for further data acquisition.
5. Interact with the page to do "next frame" action and repeat 2- 4 until all the data is exhausted.

Methods used for handling errors that occur while scraping and extraction:

1. Try and catch technique
2. time.sleep(): pause the processing to let the url load fully.

Note: The webscraping was done once and the data was stored locally. Hence, the functions that run scraping are commented out.

Function "selenium_setup()" : It imports all libraries needed for web-scraping and sets up a driver to browse urls.

In [1]:
```
def selenium_setup():
    # !pip install selenium
    # !pip install webdriver-manager
    from selenium import webdriver
```

```
    from selenium.webdriver.common.keys import Keys
    # driver = webdriver.Chrome('/Users/anugy/Downloads/chromedriver')
    from selenium.webdriver.chrome.service import Service
    from webdriver_manager.chrome import ChromeDriverManager
    from selenium.webdriver.common.by import By
    import time
    from selenium.webdriver.support.ui import Select

    s=Service(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=s)
    driver.maximize_window()
# def selenium_setup()
```

Function "rawData_list_initialise()" : It initialises some lists. These are the lists that need to be filled with raw data that we will acquire through web-scraping.

In [2]:

```
def rawData_list_initialise():
    foods_list_1 = []
    proteins_list_1 = []
    hlinks_list_1 = []
    group_list_1 = []
    sub_group_list_1 = []
    carbs_list_2 = []
    foods_list_2 = []
    fat_list_3 = []
    foods_list_3 = []
    fiber_list_4 = []
    foods_list_4 = []
# rawData_list_initialise()
```

Function "fetch_macro_list()" :It scrapes the pages to get a list of food names, each food's specific url list, and a list of macros(protein or fats or...) content.

In [3]:

```
def fetch_macro_list(macros_list, foods_list, hlinks_list):
    time.sleep(5)

    # get the list of food-names.
    foods = driver.find_elements_by_xpath('//td[@class ="sorting_1"]')
    for p in range(len(foods)):
        foods_list.append(foods[p].text)
    time.sleep(5)
    rows = driver.find_elements_by_xpath('//tr[@role ="row"]')

    # get the list of values for macros content.
    i_list = range(1, len(rows))
    for i in i_list:
        p= rows[i].find_element_by_css_selector('td:nth-child(3)')
        macro_text = p.text
        macros_list.append(macro_text)

    # get the list of useful urls.
    links = driver.find_elements_by_xpath('//td[@class ="sorting_1"]/a')
    for l in links:
        hlinks_list.append(l.get_attribute('href'))
```

Function "get_all_macros()" :It interacts with the web pages by changing the drowpdown list of data

displayed and by pressing the next button until the pages are exhausted. First this function enters a url, it then changes the amount of data displayed to 100, then acquires all the data in that page, then presses next and repeats the process until all data is acquired.

In [4]:
```
def get_all_macros(url, macros_list, foods_list, hlinks_list):
    driver.get(url)
    time.sleep(5)

    # choose "display 100 items instead of 10"
    select = Select(driver.find_element(By.NAME, 'DataTables_Table_0_length'))
    select.select_by_value("100")
    while True:
        fetch_macro_list(macros_list, foods_list, hlinks_list)
        try:
            # choose "go to next item-list"
            nextButton = driver.find_element(By.LINK_TEXT, 'Next')
            nextButton.click()
            time.sleep(5)
        except:
            break
    driver.quit()
```

The 2 functions below scrape the pages to get a list of food names, and a list of macros(protein or fats or...) content without each food's url

In [5]:
```
def fetch_only_macro_list(macros_list, foods_list):
    time.sleep(5)

    # get the list of food-names.
    foods = driver.find_elements_by_xpath('//td[@class ="sorting_1"]')
    for p in range(len(foods)):
        foods_list.append(foods[p].text)
    time.sleep(5)

    # get the list of values for macros content.
    rows = driver.find_elements_by_xpath('//tr[@role ="row"]')

    i_list = range(1, len(rows))
    for i in i_list:
        time.sleep(1)
        p= rows[i].find_element_by_css_selector('td:nth-child(3)')
        macro_text = p.text
        macros_list.append(macro_text)
```

In [6]:
```
def get_all_only_macros(url, macros_list, foods_list):

    driver.get(url)
    time.sleep(5)
    select = Select(driver.find_element(By.NAME, 'DataTables_Table_0_length'))
    select.select_by_value("100")
    while True:
        fetch_only_macro_list(macros_list, foods_list)
        try:
            nextButton = driver.find_element(By.LINK_TEXT, 'Next')
            nextButton.click()
```

```
            time.sleep(5)
        except:
            break
    driver.quit()
```

Going through the list of acquired urls.

This Function below goes through a list of urls (specific url for each food), opens those urls, interacts with the opened page and fetches the food's group and sub group name information from the opened page.

In [7]:
```
def inside_each_food(hlinks_list, group_list, sub_group_list,):
    for url in  hlinks_list:
        driver = webdriver.Chrome(service=s)
        driver.maximize_window()
        driver.get(url)

        time.sleep(1)

        # get the list of group categorization for each food-item
        group = driver.find_element_by_css_selector('tbody > tr:nth-child(8) > td')
        group_list.append(group.text)

        # get the list of sub-group categorization for each food-item
        sub_group = driver.find_element_by_css_selector('tbody > tr:nth-child(9) > td')
        sub_group_list.append(sub_group.text)
        driver.quit()
```

Call the functions that scrape the web and fill the empty lists of raw data.

In [8]:
```
def web_to_lists():
    # get_all_macros('https://fooddb.ca/nutrients/FDBN00002', proteins_list_1, foods_li
    inside_each_food(hlinks_list_1, group_list_1, sub_group_list_1)

    # for carbs now : carbs_list_2, foods_list_2
    get_all_only_macros('https://fooddb.ca/nutrients/FDBN00003', carbs_list_2, foods_li

    # for fat now : fat_list_3, foods_list_3
    get_all_only_macros('https://fooddb.ca/nutrients/FDBN00001', fat_list_3, foods_list

    # for fiber now : fiber_list_4, foods_list_4
    get_all_only_macros('https://fooddb.ca/nutrients/FDBN00005', fiber_list_4, foods_li
# web_to_lists()
```

End of Web-scraping: We have collected 10 lists filled with raw data

3. Working through the database.

3.1 Data handling : cleaning the data and downloading them as csv.

In [9]:
```
# importing data handling libraries
```

```
import numpy as np
import pandas as pd
```

Function "list_to_dataframe()" : It has 3 jobs

- Convert raw string to appropriate numerical form.
- Convert processed data to pandas dataframe
- Download pandas data frame to csv file for further use.

In [10]:
```
def list_to_dataframe():
    proteins_per100g = [float(p.split(" ")[0]) for p in proteins_list_1]
    carbs_per100g = [float(p.split(" ")[0]) for p in carbs_list_2]
    fats_per100g = [float(p.split(" ")[0]) for p in fat_list_3]
    fiber_per100g = [float(p.split(" ")[0]) for p in fiber_list_4]
    df3 = pd.DataFrame(list(zip(foods_list_3, fats_per100g)), columns = ["Food_Name", "
    df4 = pd.DataFrame(list(zip(foods_list_4, fiber_per100g)), columns = ["Food_Name",
    df2 = pd.DataFrame(list(zip(foods_list_2, carbs_per100g)), columns = ["Food_Name",
    df1 = pd.DataFrame(list(zip(foods_list_1, group_list_1, sub_group_list_1, proteins_
    df1.to_csv("data1.csv")
    df2.to_csv("data2.csv")
    df3.to_csv("data3.csv")
    df4.to_csv("data4.csv")
# list_to_dataframe()
```

3.2 Using the created csv file from here on:

In [11]:
```
df1 = pd.read_csv("data1.csv", index_col="Unnamed: 0")
df2 = pd.read_csv("data2.csv", index_col="Unnamed: 0")
df3 = pd.read_csv("data3.csv", index_col="Unnamed: 0")
df4 = pd.read_csv("data4.csv", index_col="Unnamed: 0")
```

3.3 Determine Index column

From all the data collected, we have realised that the only thing that these data have in common is the food that they pertain to. Food name is the only comman variable for all the data. So we change food-name-column to the index-column.

In [12]:
```
df1_copy = df1.copy()
df1_copy.set_index("Food_Name", inplace = True)
df2_copy = df2.copy()
df2_copy.set_index("Food_Name", inplace = True)
df3_copy = df3.copy()
df3_copy.set_index("Food_Name", inplace = True)
df4_copy = df4.copy()
df4_copy.set_index("Food_Name", inplace = True)
```

3.4 Match and combine dataframes

Match and Combine all the dataframes into one,while making sure that all data pertaining to a single food remains in the same row.

In [13]:
```
combined_df = df1_copy.merge(df2_copy,how = "outer", left_index = True, right_index = T
```

In [14]:
```
combined_df = combined_df.merge(df3_copy,how = "outer", left_index = True, right_index
```

In [15]:
```
combined_df = combined_df.merge(df4_copy,how = "outer", left_index = True, right_index
```

In [16]:
```
# Display the dataframe in full
# pd.set_option('display.max_rows', None)
# combined_df
```

3.5 Version Control:

Introduce some version control here. Make a copy of the combined and matched dataframe.

In [17]:
```
final_df = combined_df.copy()
```

3.6 NaN and "Not Available" data Handling:

Our final combined data frame has NAn in some cells, that arose because some foods dont contains certain macronutrients. It should infact be filled with 0 rather than NaN. Also, some foods are not categorised into sub groups and groups, and so have not available data.

In [18]:
```
# finding out the columns that need nan handling
#only the last 3 columns in the dataframe that need nan to be changed to 0
col_names = final_df.columns
col_names = col_names[2:]
```

In [19]:
```
# actual nan handling
final_df[col_names] = final_df[col_names].fillna(0)
# final_df
```

3.7 Data visualisation

Importing libraries for calculations, data handling and data visualisation. We will use these later for graphs and charts.

In [20]:
```
import seaborn as sns
from matplotlib import rcParams
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
import matplotlib.transforms as transforms
```

3.8 Data Analysis:

3.8.1 Step 1 - Food-group analysis

Sometimes, rather than just focusing on individual food items , we will also have to look at food-groups as a whole, determine their characteristics and establish trend. So lets work to create a basis for food-group analysis. Delete all foods without a proper food group categorization.

In [21]:
```
#Delete rows where group column cells are filled with either na or "Not Available"
food_group_df = final_df.copy()
food_group_df.dropna(subset=['Group'], how='all', inplace=True)
food_group_df = food_group_df[food_group_df["Group"].str.contains("Not Available")==Fal
```

In [22]:
```
food_group_df
```

Out[22]:

Food_Name	Group	Sub-Group	Protein_Content	Carbs_Content	Fats_Content	Fiber_Content
Adzuki bean	Pulses	Beans	10000.000	21400.000	1000.0	2600.000
Agar	Aquatic foods	Seaweed	2300.000	72500.000	300.0	2500.000
Alfalfa	Herbs and Spices	Herbs	19550.000	19500.000	6650.0	2450.000
Allium	Vegetables	Onion-family vegetables	2550.000	14562.500	12150.0	1212.500
Almond	Nuts	Nuts	20500.000	24900.000	39100.0	4600.000
...	...	...	...	...	...	...
Wild celery	Herbs and Spices	Spices	700.000	4400.000	200.0	800.000
Wonton wrapper	Baking goods	Wrappers	9800.000	57900.000	0.0	1800.000
Yam	Vegetables	Tubers	1500.000	25850.000	200.0	2050.000
Yogurt	Milk and milk products	Fermented milk products	2622.381	22882.581	2920.0	309.677
Zwieback	Cereals and cereal products	Leavened breads	10100.000	74200.000	0.0	2500.000

259 rows × 6 columns

3.8.2 Step 2 - Create functions that can be utilised on the data

Creating some reusable functions that either filter data based on certain conditions or calculate statistical parameters such as mean and standard deviation of the given data.

In [23]:

```
# This function calculates the mean of given data.
def find_mean(df_name, column_name):
    mean_value = df_name[column_name].values
    return np.mean(mean_value)

# This function creates a filtered data column.
def create_subset_df(df_name, group_name):
    return df_name.loc[df_name['Group'] == group_name]

# This function creates a list of unique food-group options.
group_list = np.unique(food_group_df['Group'])

# This function calculates the standard deviation of given data.
def find_rms(x_list, y_list):
    return np.sqrt(np.sum(((x_list - np.mean(x_list))**2 + (y_list - np.mean(y_list))**

# This function gets radius and center of a circle where center is the mean and raidius
def find_circle_params(subset_df, x_axis, y_axis):
    center_point = (find_mean(subset_df, x_axis), find_mean(subset_df, y_axis))
    circle_radius = find_rms(subset_df[x_axis], subset_df[y_axis])

    return center_point, circle_radius
```

3.8.3 Step 3 - Analyse the distribution of protein and fiber content for all individual foods as well as food-groups

Our Numerical Focus: Fiber content in protein-rich foods

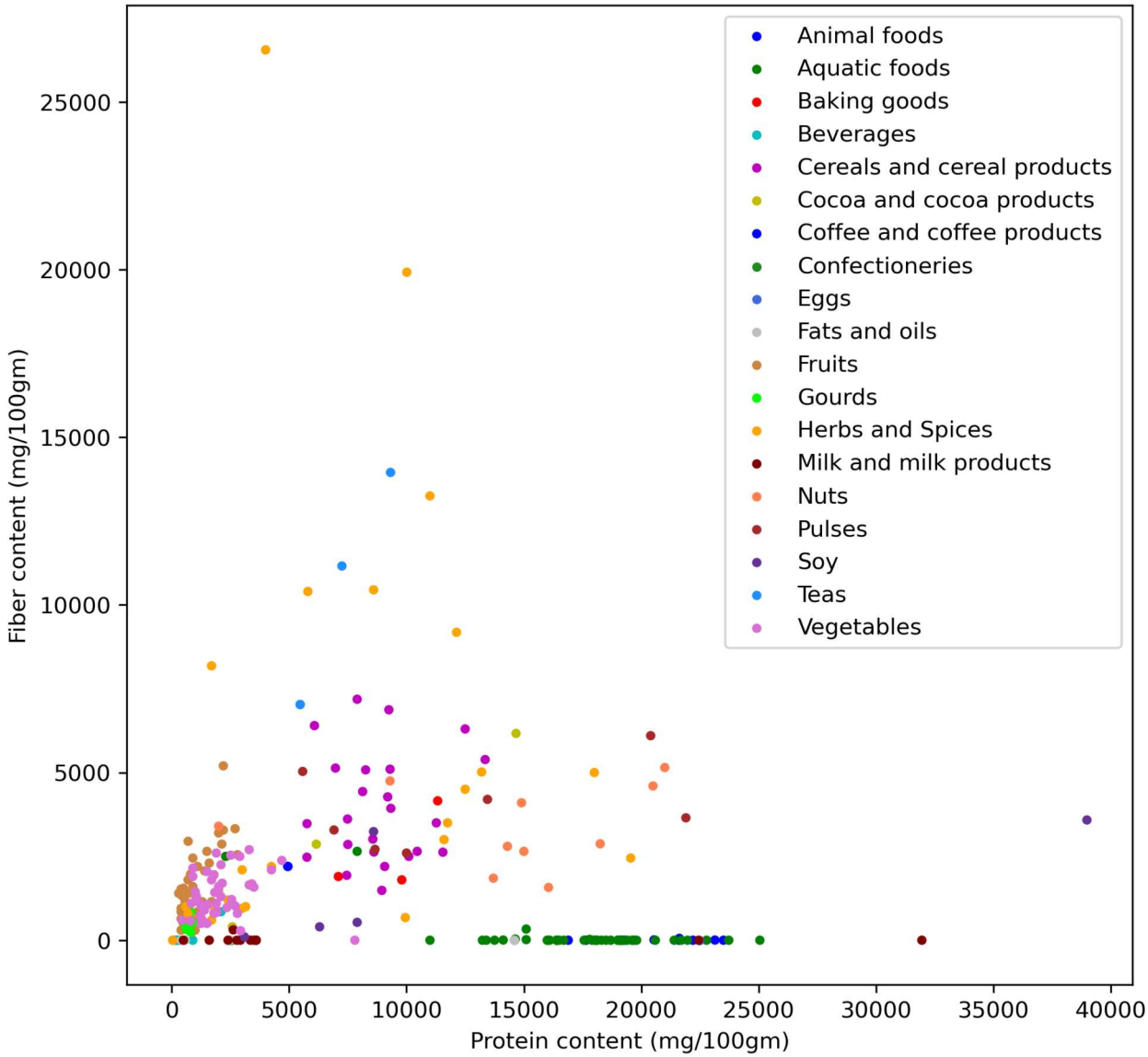
First, we will study the distribution of protein and fiber content for all of the foods.

We need to first plot the scatter point for each food on a protein-content-axis Vs fiber-content-axis. Once the points are plotted , we determine an ellipse that covers the general area of foods that belong to the same food group. This ellipse will have the center point at the mean of x and y values of the food points of the same food group and radius as the standard deviation of these points. This function below, takes in a list of x-cordinates list and a y-coordinates list and draws these informative ellipses to the matplotlib axis that we feed it.

In [24]:

```
def confidence_ellipse(x, y, ax, n_std=3.0, facecolor='none', **kwargs):
    #Source: original code from matplotlib.org
    """
    Create a plot of the covariance confidence ellipse of *x* and *y*.
    Parameters
    ----------
    x, y : array-like, shape (n, )
        Input data.
    ax : matplotlib.axes.Axes
        The axes object to draw the ellipse into.
    n_std : float
        The number of standard deviations to determine the ellipse's radiuses.
    **kwargs
        Forwarded to `~matplotlib.patches.Ellipse`
    Returns
    -------
    matplotlib.patches.Ellipse
    """
```

```
    if x.size != y.size:
        raise ValueError("x and y must be the same size")

    cov = np.cov(x, y)
    pearson = cov[0, 1]/np.sqrt(cov[0, 0] * cov[1, 1])
    # Using a special case to obtain the eigenvalues of this
    # two-dimensionl dataset.
    ell_radius_x = np.sqrt(1 + pearson)
    ell_radius_y = np.sqrt(1 - pearson)
    ellipse = Ellipse((0, 0), width=ell_radius_x * 2, height=ell_radius_y * 2,
                      facecolor=facecolor, **kwargs)

    # Calculating the standard deviation of x from
    # the squareroot of the variance and multiplying
    # with the given number of standard deviations.
    scale_x = np.sqrt(cov[0, 0]) * n_std
    mean_x = np.mean(x)

    # calculating the standard deviation of y ...
    scale_y = np.sqrt(cov[1, 1]) * n_std
    mean_y = np.mean(y)

    transf = transforms.Affine2D() \
        .rotate_deg(45) \
        .scale(scale_x, scale_y) \
        .translate(mean_x, mean_y)

    ellipse.set_transform(transf + ax.transData)
    return ax.add_patch(ellipse)


fig = plt.figure( dpi=300, figsize=(8,8))
ax1 = fig.add_subplot()
nutrient1 = "Protein_Content"
nutrient2 = "Fiber_Content"

# colornames = list(mcolors.CSS4_COLORS)
# color_list = [colornames[5*i] for i in range(len(group_list))]

color_list = ['b', 'g', 'r', 'c', 'm', 'y', 'b', 'forestgreen', 'royalblue', 'silver',

for i, g in enumerate(group_list):

    #if g not in ['Fruits', 'Vegetables']:
        #continue

    subset_df = create_subset_df(food_group_df, g)

    ax1.scatter(subset_df[nutrient1], subset_df[nutrient2], s=10, c=len(subset_df)*[col
    ax1.set_xlabel("Protein content (mg/100gm)")
    ax1.set_ylabel("Fiber content (mg/100gm)")

    #confidence_ellipse(subset_df[nutrient1], subset_df[nutrient2], ax1, edgecolor=colo


plt.legend()

plt.show()
```



Preliminary observation from this graph:

From the above graph, we can see that there is no obvious group which shows high protein and fiber contents. We can however see that pulses and nuts could be possible candidates for both high-protein-content as well as high-fiber-content . So, we carry on our analysis with "Nuts" and "Pulses" subgroup. Now, fruits and vegetables are the most recommended sources of fiber[4] so we also continue our study on these 2 food-groups to see if they can simultaneously provide protein as well as fiber.

Moving on:

- Carry on further analysis on "Nuts" and "Pulses" as potential high-protein-candidates.
- Carry on further analysis on "Fruits" and "Vegetables" as potential high-fiber-candidates.

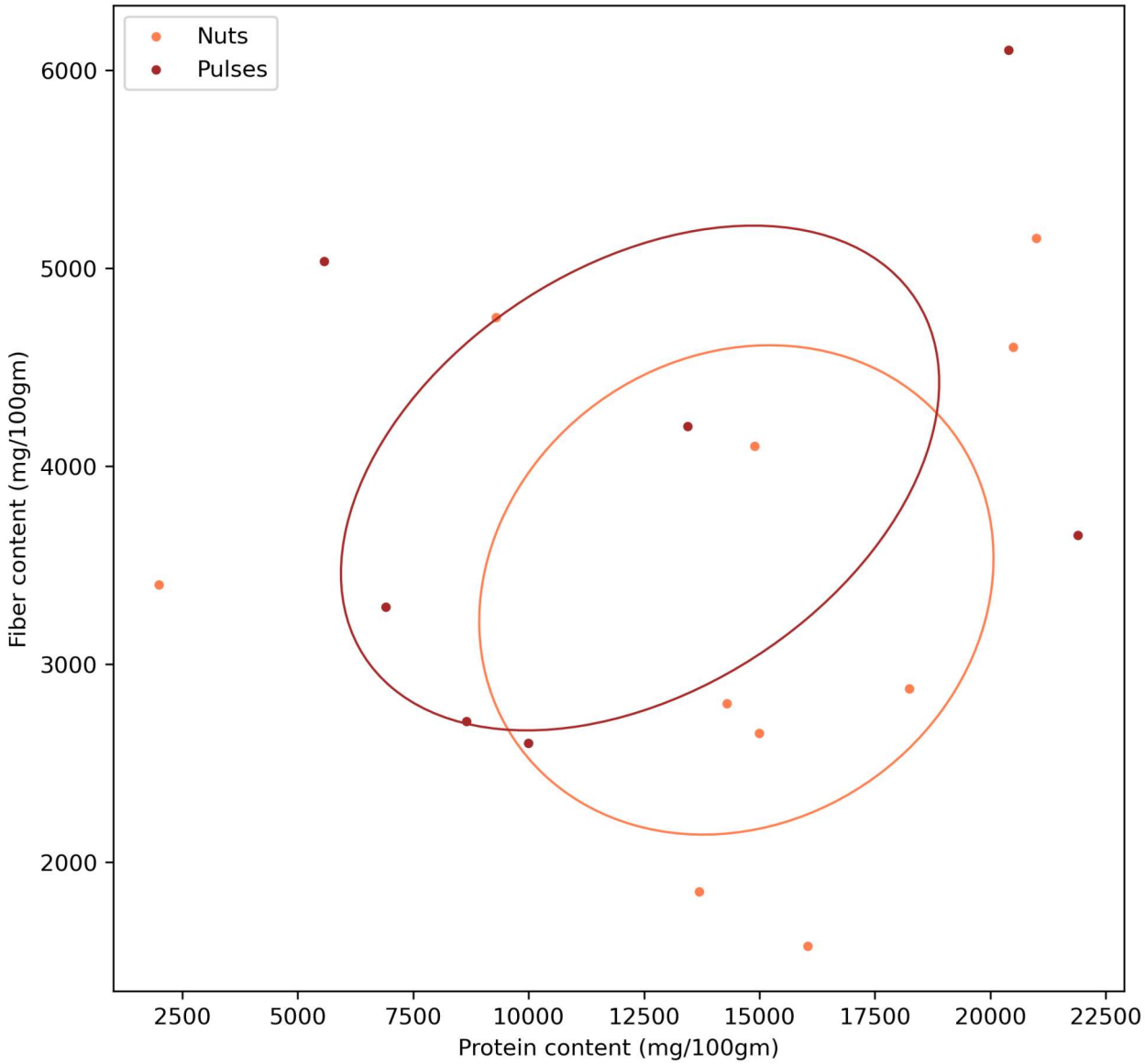
3.8.4 Step 4 - Study the high-protein-content candidates established from step 3.

Scatter plot with confidence ellipses: (high-protein-candidates)

We will now draw scatter plot just for pulses and nuts and also draw confidence ellipses for these 2 groups. Confidence ellipses show the spread of the data for that particular food group.

In [25]:

```
fig = plt.figure( dpi=300, figsize=(8,8))
ax1 = fig.add_subplot()
nutrient1 = "Protein_Content"
nutrient2 = "Fiber_Content"

color_list = ['b', 'g', 'r', 'c', 'm', 'y', 'b', 'forestgreen', 'royalblue', 'silver',

for i, g in enumerate(group_list):
    if g not in ['Nuts', 'Pulses']:
        continue

    subset_df = create_subset_df(food_group_df, g)
    ax1.scatter(subset_df[nutrient1], subset_df[nutrient2], s=10, c=len(subset_df)*[col
    confidence_ellipse(subset_df[nutrient1], subset_df[nutrient2], ax1, edgecolor=color
    ax1.set_xlabel("Protein content (mg/100gm)")
    ax1.set_ylabel("Fiber content (mg/100gm)")
plt.legend()
plt.show()
```



What about fruits and vegetables? We plot the protein content against fiber content for these groups together with nuts and pulses for a comparison.

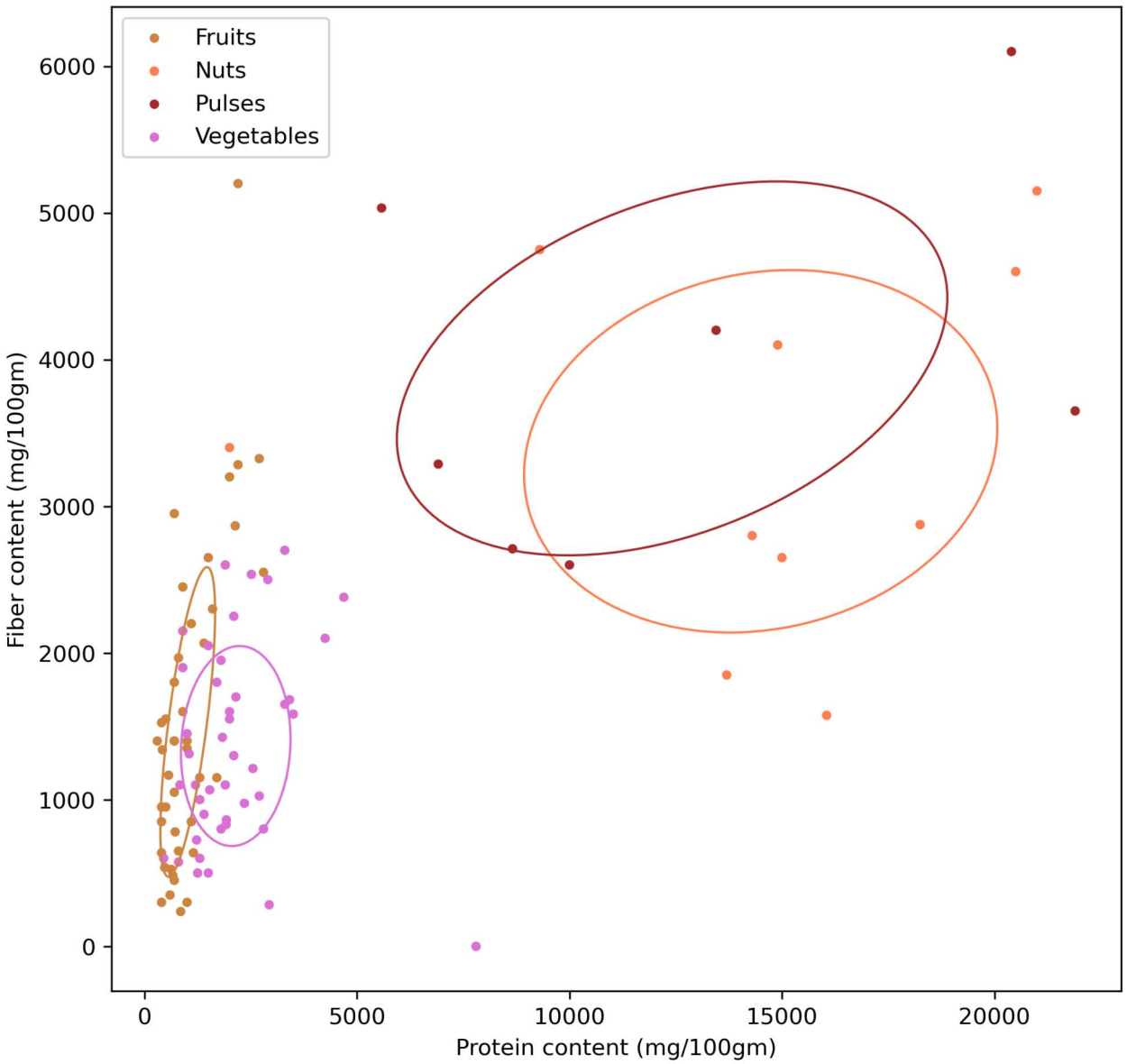
3.8.5 Step 5 - Study the high-fiber-content candidates established from step 3.

Scatter plot with confidence ellipses: (high-fiber-candidates)

We will now add scatter plot of fruits and vegetables to that of pulses and nuts and also draw confidence ellipses for these 2 groups.

In [26]:
```
fig = plt.figure( dpi=300, figsize=(8,8))
ax1 = fig.add_subplot()
nutrient1 = "Protein_Content"
nutrient2 = "Fiber_Content"


color_list = ['b', 'g', 'r', 'c', 'm', 'y', 'b', 'forestgreen', 'royalblue', 'silver',
```

```
for i, g in enumerate(group_list):

    if g not in ['Nuts', 'Pulses', 'Vegetables', 'Fruits']:
        continue

    subset_df = create_subset_df(food_group_df, g)

    ax1.scatter(subset_df[nutrient1], subset_df[nutrient2], s=10, c=len(subset_df)*[col

    confidence_ellipse(subset_df[nutrient1], subset_df[nutrient2], ax1, edgecolor=color
    ax1.set_xlabel("Protein content (mg/100gm)")
    ax1.set_ylabel("Fiber content (mg/100gm)")


plt.legend()

plt.show()
```



Preliminary observations from this graph:

We can see that while some fruits and vegetables can have a large amount of fiber, for most of the foods in these groups, the protein content is not comparable to nuts and pulses. Moreover, looking at the efficiency with respect to obtaining both protein and fiber at the same time, fruits and vegetables may not be the best option. Hence, "Pulses" and "Nuts" food-group remain the most suitable candidate for our research question.

Moving on:

Rank food-groups based on protein-content.

3.8.6 Step 6 - Analysis on the top 25% of foods with highest protein-content

In order to rank the food items, we will create a list of foods that are at the top 25% in terms of their protein content.

In [27]:
```
high_protein_df = food_group_df.nlargest(int(0.25*len(food_group_df)), 'Protein_Content
```

We will conduct an analysis of the food groups that are included in this list of high-protein foods.

In [28]:
```
high_protein_df
```

Out[28]:

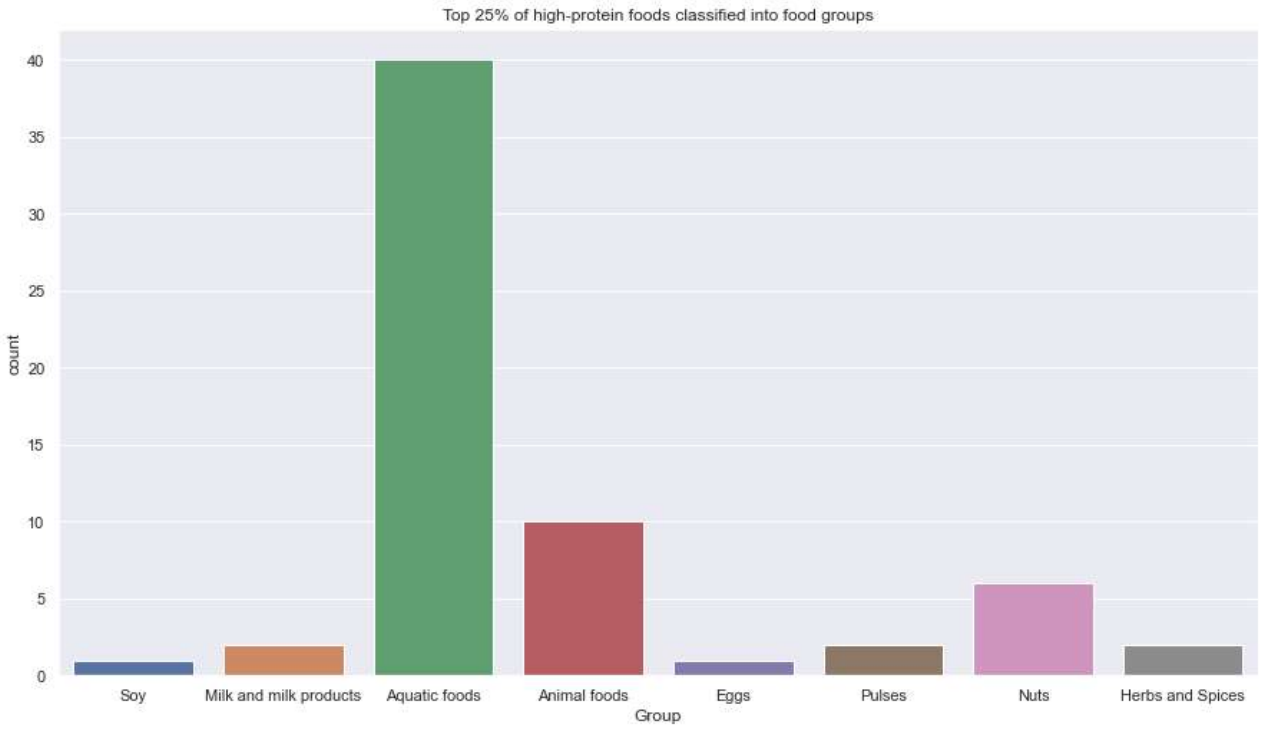
Food_Name	Group	Sub-Group	Protein_Content	Carbs_Content	Fats_Content	Fiber_Content
Soy bean	Soy	Soy	38979.474	11380.800	23733.333	3586.364
Dried milk	Milk and milk products	Other milk products	31950.000	44600.000	13800.000	0.000
Charr	Aquatic foods	Fishes	25050.000	0.000	6500.000	0.000
Thunnus	Aquatic foods	Fishes	23725.000	125.000	4875.000	0.000
Rock ptarmigan	Animal foods	Poultry	23500.000	0.000	2000.000	0.000
...	...	...	...	...	...	...
Snail	Aquatic foods	Mollusks	16000.000	2000.000	1000.000	0.000
Gadus (Common cod)	Aquatic foods	Fishes	15100.000	1325.000	21921.429	17.857
Pleuronectidae (Dab, Halibut, Plaice)	Aquatic foods	Fishes	15100.000	12633.333	5266.667	333.333
Brazil nut	Nuts	Nuts	15000.000	11150.000	65000.000	2650.000

Food_Name	Group	Sub-Group	Protein_Content	Carbs_Content	Fats_Content	Fiber_Content
Hazelnut	Nuts	Nuts	14900.000	25200.000	49300.000	4100.000

64 rows × 6 columns

In [29]:
```
sns.set(rc={'figure.figsize':(15,8.27)})

p = sns.countplot(x="Group", data=high_protein_df)
p.set(title='Top 25% of high-protein foods classified into food groups')
```

Out[29]: [Text(0.5, 1.0, 'Top 25% of high-protein foods classified into food groups')]



Preliminary observation from the graph:

We can see that a majority of the high-protein foods are aquatic foods and animal foods. We can look at the fiber to protein ratio for each group to look at the pattern of the data.

Next, we will look at the fiber-to-protein ratio for each of these food items.

3.8.7 Step 7 - Analyse fiber-to-protein ratio on the lists of protein-rich food-groups obtained in step 6.

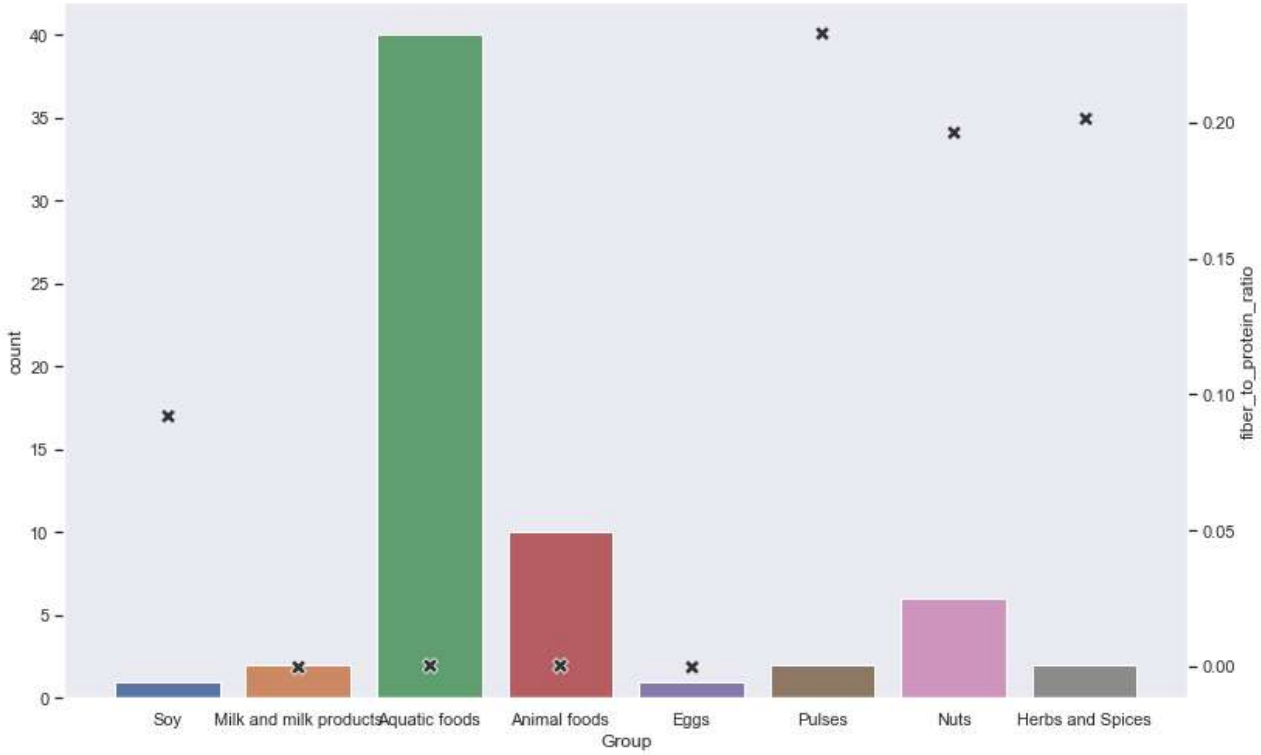
1. Calculate mean protein-content and mean fiber-content of every food group by taking averages of all the foods in that group.
2. Delete columns with carbs and fats data.
3. Draw bar charts

In [30]:
```
#Calculate mean values for food-groups and fill the table
high_protein_df['fiber_to_protein_ratio'] = high_protein_df.apply(lambda row: row.Fiber
mean_df = high_protein_df.groupby('Group', sort=False).mean()
mean_df['Group'] = mean_df.index
#delete not usuable columns
mean_df= mean_df.drop(['Carbs_Content', 'Fats_Content'], axis=1 )
mean_df
```

Out[30]:

Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio	Group
Soy	38979.474000	3586.364000	0.092006	Soy
Milk and milk products	27196.666500	0.000000	0.000000	Milk and milk products
Aquatic foods	18680.541675	9.404750	0.000617	Aquatic foods
Animal foods	20562.476000	7.542700	0.000353	Animal foods
Eggs	22469.231000	0.000000	0.000000	Eggs
Pulses	21150.000000	4875.000000	0.232843	Pulses
Nuts	17616.666667	3491.666667	0.196188	Nuts
Herbs and Spices	18775.000000	3725.000000	0.201549	Herbs and Spices

In [31]:
```
fig, ax1 = plt.subplots(figsize=(13,8.27))

sns.countplot(x="Group",data=high_protein_df)
ax2 = ax1.twinx()

sns.scatterplot(data = mean_df, x='Group', y='fiber_to_protein_ratio', s=100, color=".2

# Turns off grid on the left Axis.
ax1.grid(False)

# Turns off grid on the secondary (right) Axis.
ax2.grid(False)
```



Description and preliminary observation from the graph:

In this graph, the cross-marks represent the fiber to protein ratio of each food and the bars represent the no. of foods in a partcular food group. Here, we can see that top 2 highest bars have very low fiber to protein ratio. So, we will have to move on to 3rd and 4th highest bars . These are the bars for "Nuts" and "Pulses" and also happen to have a high fiber to protein ratio.

Moving on:

We are done with our food-group analysis and now we will dig deeper and look at each individual foods rather than food-groups.

Individual food analysis:

- Next, we will look at the fiber-to-protein ratio for each of these food items that fall in the "top 10 high protein food".
- Since we are already looking at the top 10 protein rich foods , we now sort out list by fiber-to - protein ratio.
- And so we get a preliminary list of **"Ten food items that we can include in our diets for maximizing both the protein and the fiber intake."**

3.8.8 Step 8 - Analyse fiber-content of top 10 protein rich foods.

In [32]:
```
#get the top 10 protein-rich foods and then sort it by fiber-to-protein ratio
top_ten_df = high_protein_df.nlargest((12), 'fiber_to_protein_ratio')
top_ten_df= top_ten_df.drop(['Carbs_Content', 'Fats_Content'], axis=1 )
top_ten_df
```

Out[32]:

Food_Name	Group	Sub-Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio
Chickpea	Pulses	Peas	20400.000	6100.000	0.299020
Poppy	Herbs and Spices	Oilseed crops	18000.000	5000.000	0.277778
Hazelnut	Nuts	Nuts	14900.000	4100.000	0.275168
Pistachio	Nuts	Nuts	21000.000	5150.000	0.245238
Almond	Nuts	Nuts	20500.000	4600.000	0.224390
Brazil nut	Nuts	Nuts	15000.000	2650.000	0.176667
Lentils	Pulses	Lentils	21900.000	3650.000	0.166667
Peanut	Nuts	Nuts	18250.000	2875.000	0.157534
Alfalfa	Herbs and Spices	Herbs	19550.000	2450.000	0.125320
Cashew nut	Nuts	Nuts	16050.000	1575.000	0.098131
Soy bean	Soy	Soy	38979.474	3586.364	0.092006
Pleuronectidae (Dab, Halibut, Plaice)	Aquatic foods	Fishes	15100.000	333.333	0.022075

Removal of impractical Food-groups Non-whole food groups such as "herbs and spices" and "Cocoa products" will be removed from this list, as they cannot be consumed in large enough quantites.

In [33]:
```
top_ten_df.drop(top_ten_df.index[top_ten_df['Group'] == "Herbs and Spices"] | top_ten_d
#Final list of our top 10 foods is ready
top_ten_df
```

```
<ipython-input-33-9615999ee9c2>:1: FutureWarning: Index.__or__ operating as a set operat
ion is deprecated, in the future this will be a logical operation matching Series.__or_
_.  Use index.union(other) instead
  top_ten_df.drop(top_ten_df.index[top_ten_df['Group'] == "Herbs and Spices"] | top_ten_
df.index[top_ten_df['Group'] == "Cocoa and cocoa products"]  , inplace=True)
```

Out[33]:

Food_Name	Group	Sub-Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio
Chickpea	Pulses	Peas	20400.000	6100.000	0.299020
Hazelnut	Nuts	Nuts	14900.000	4100.000	0.275168
Pistachio	Nuts	Nuts	21000.000	5150.000	0.245238
Almond	Nuts	Nuts	20500.000	4600.000	0.224390
Brazil nut	Nuts	Nuts	15000.000	2650.000	0.176667
Lentils	Pulses	Lentils	21900.000	3650.000	0.166667

Food_Name	Group	Sub-Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio
Peanut	Nuts	Nuts	18250.000	2875.000	0.157534
Cashew nut	Nuts	Nuts	16050.000	1575.000	0.098131
Soy bean	Soy	Soy	38979.474	3586.364	0.092006
Pleuronectidae (Dab, Halibut, Plaice)	Aquatic foods	Fishes	15100.000	333.333	0.022075

Final list of our top 10 foods is ready

We can thus conclude that including lots of nuts and pulses in our diets is the optimum choice. In particular, remember to include chickpeas in your daily diet!

4 Price Analysis

- find out price for our final top 10 food items.

In [34]:
```
top_ten_df
```

Out[34]:

Food_Name	Group	Sub-Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio
Chickpea	Pulses	Peas	20400.000	6100.000	0.299020
Hazelnut	Nuts	Nuts	14900.000	4100.000	0.275168
Pistachio	Nuts	Nuts	21000.000	5150.000	0.245238
Almond	Nuts	Nuts	20500.000	4600.000	0.224390
Brazil nut	Nuts	Nuts	15000.000	2650.000	0.176667
Lentils	Pulses	Lentils	21900.000	3650.000	0.166667
Peanut	Nuts	Nuts	18250.000	2875.000	0.157534
Cashew nut	Nuts	Nuts	16050.000	1575.000	0.098131
Soy bean	Soy	Soy	38979.474	3586.364	0.092006
Pleuronectidae (Dab, Halibut, Plaice)	Aquatic foods	Fishes	15100.000	333.333	0.022075

In [35]:
```
# per 100 oz chickpea, hazelnut, pistachio, almonds, brazil nut, lentils, peanut, cashe
# usd_per_pound  = [1.28, 6, 6.37, 4.96, 10, 1.34, 1.98, 7.82, 2, 26]
```

Sources of food prices: https://data.ers.usda.gov/reports.aspx?programArea=fruit&groupName=Tree%20nuts&ms_key=F&reportPath=/FruitAndVeg/MarketSegment_

https://www.agmrc.org/commodities-products/vegetables/chickpeas
https://www.ers.usda.gov/data-products/vegetables-and-pulses-data/vegetables-and-pulses-yearbook-tables/

4.1 Steps in Price analysis:

- Find out price(USD$) per mass(pound) for our final top 10 food items.
- Get the daily requirement of protein and fiber for humans.
- Calculate mass(grams) of each 10 foods required to meet the protein and fiber daily requirements of humans.
- Calculate the price of total amount of each food required to meet the protein and fiber requirement.
- Determine the most affordable food-item, which is also the most suitabe to meet our protein and fiber daily prequisites.

In [36]:
```
finalDf = top_ten_df
price_df = pd.read_csv("food_prices.csv", index_col = 0 )
finalDf = finalDf.merge(price_df, how = "outer", left_index = True, right_index=True)
```

We will also be interested in finding the most economical method of obtaining both protein and fiber. For this, the price data for each of the top ten food was obtained from the retail market information from the United States Department of Agriculture at usda.gov. Potential improvement in this step can include the use of web scraping for updated retail price of the food items.

In [37]:
```
finalDf
```

Out[37]:

	Group	Sub-Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio	Price (USD/lb)
Chickpea	Pulses	Peas	20400.000	6100.000	0.299020	1.28
Hazelnut	Nuts	Nuts	14900.000	4100.000	0.275168	6.00
Pistachio	Nuts	Nuts	21000.000	5150.000	0.245238	6.37
Almond	Nuts	Nuts	20500.000	4600.000	0.224390	4.96
Brazil nut	Nuts	Nuts	15000.000	2650.000	0.176667	10.00
Lentils	Pulses	Lentils	21900.000	3650.000	0.166667	1.34
Peanut	Nuts	Nuts	18250.000	2875.000	0.157534	1.98
Cashew nut	Nuts	Nuts	16050.000	1575.000	0.098131	7.82
Soy bean	Soy	Soy	38979.474	3586.364	0.092006	2.00
Pleuronectidae (Dab, Halibut, Plaice)	Aquatic foods	Fishes	15100.000	333.333	0.022075	26.00

In [48]:
```
# 2 constant that tell you daily prerequisites : source : average from mayoclinic.org
```

```
daily_fiber_prerequisite_recommendation_gram = 28.5
daily_protein_prerequisite_recommendation_gram = 51

# daily_grams_required
finalDf["daily_grams_required_for_protein"] = (28.5 * 100000)/finalDf["Protein_Content"
finalDf["daily_grams_required_for_fiber"] = (51 * 100000)/finalDf["Fiber_Content"]

# Daily minimum grams required for both fiber and protein needs
finalDf["daily_grams_required"] = np.maximum( finalDf["daily_grams_required_for_protein

# Cost to meet the daily fiber and protein needs
finalDf["priceUSD_per_100grams"] =  finalDf["Price (USD/lb)"] * 0.220462
finalDf["price_required_grams"] =  finalDf["priceUSD_per_100grams"] * finalDf["daily_gr
```

In [49]:
```
finalDf
```

Out[49]:

	Group	Sub-Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio	Price (USD/lb)	daily_g
Chickpea	Pulses	Peas	20400.000	6100.000	0.299020	1.28	
Hazelnut	Nuts	Nuts	14900.000	4100.000	0.275168	6.00	
Pistachio	Nuts	Nuts	21000.000	5150.000	0.245238	6.37	
Almond	Nuts	Nuts	20500.000	4600.000	0.224390	4.96	
Brazil nut	Nuts	Nuts	15000.000	2650.000	0.176667	10.00	
Lentils	Pulses	Lentils	21900.000	3650.000	0.166667	1.34	
Peanut	Nuts	Nuts	18250.000	2875.000	0.157534	1.98	
Cashew nut	Nuts	Nuts	16050.000	1575.000	0.098131	7.82	
Soy bean	Soy	Soy	38979.474	3586.364	0.092006	2.00	
Pleuronectidae (Dab, Halibut, Plaice)	Aquatic foods	Fishes	15100.000	333.333	0.022075	26.00	

5. Summary of the study:

In [52]:
```
# sort based on affordability:
#Top 3 most cost effective protein-rich foods with highest fiber-to-protein ratio
results_Df = finalDf.nsmallest((5), 'price-required-grams')
```

In [53]:
```
# display only necessary info
results_Df = results_Df.drop(['daily_grams_required_for_protein', 'daily_grams_required
results_Df
```

Out[53]:

	Group	Sub-Group	Protein_Content	Fiber_Content	fiber_to_protein_ratio	Price (USD/lb)	priceUSD_per_
Chickpea	Pulses	Peas	20400.000	6100.000	0.299020	1.28	
Lentils	Pulses	Lentils	21900.000	3650.000	0.166667	1.34	
Soy bean	Soy	Soy	38979.474	3586.364	0.092006	2.00	
Peanut	Nuts	Nuts	18250.000	2875.000	0.157534	1.98	
Almond	Nuts	Nuts	20500.000	4600.000	0.224390	4.96	

After looking through 3 lenses:

1. lense 1: high protein-content food (By calculating top 25% protein rich foods, and through food group analysis)
2. lense 2: High fiber-to-protein ratio
   - By calculating fiber-to-protein ratio of the foods obtained through lense 1
   - Getting the Top 10 food list by sorting the above list based on fiber-to-protein ratio
3. lense 3: Most cost-effective
   - Calculating the cost of foods to get daily requirement of protein and fiber, for the top 10 foods obtained through lense 2.
   - Get top 5 foods after sorting based on cost-effectiveness.

Final recommended products are : 'Chickpea', 'Lentils', 'Soy bean', 'Peanut', 'Almond'

In [42]:
```
Final_recommended_foods = list(results_Df.index)
```

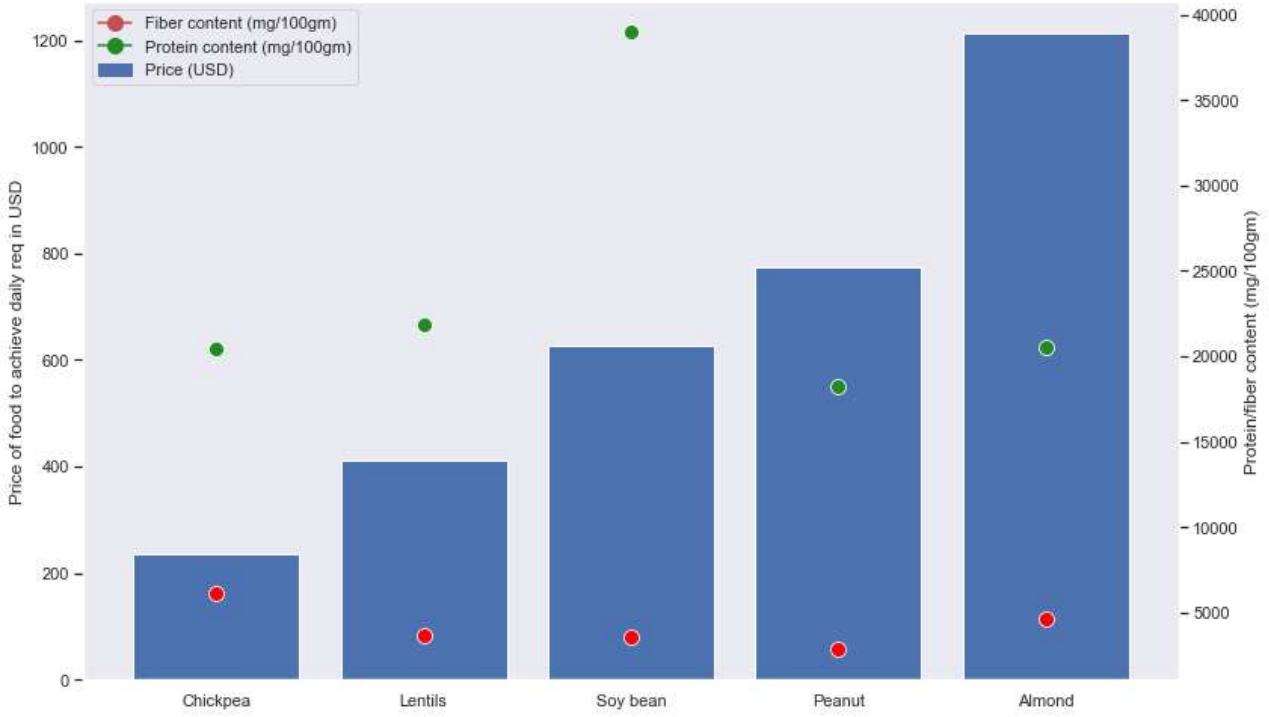
In [43]:
```
Final_recommended_foods
```

Out[43]:
```
['Chickpea', 'Lentils', 'Soy bean', 'Peanut', 'Almond']
```

5.1 Plot of our final-recommended foods:

In [44]:
```
fig, ax1 = plt.subplots(figsize=(13,8.27))

plt.bar(x = results_Df.index, height ="price-required-grams",data=results_Df, color = '
ax2 = ax1.twinx()

sns.scatterplot(data = results_Df, x=results_Df.index, y='Protein_Content', s=100, colo
sns.scatterplot(data = results_Df, x=results_Df.index, y='Fiber_Content', s=100, color=

from matplotlib.patches import Patch
from matplotlib.lines import Line2D

legend_elements = [Line2D([0], [0], marker='o', color='r', label='Fiber content (mg/100
                          markerfacecolor='r', markersize=10),
                   Line2D([0], [0], marker='o', color='forestgreen', label='Protein con
```

```
                        markerfacecolor='forestgreen', markersize=10),
                  Patch(facecolor='b', edgecolor='b',
                        label='Price (USD)')]

# Create the figure
ax2.legend(handles=legend_elements, loc='upper left')
ax1.set_ylabel("Price of food to achieve daily req in USD")
ax2.set_ylabel("Protein/fiber content (mg/100gm)")

# Turns off grid on the left Axis.
ax1.grid(False)

# Turns off grid on the secondary (right) Axis.
ax2.grid(False)
```



5.2 Description of the bar chart:

Our Top 5 most recommended food for protein/fiber/price-point

- Left axis is price of the food such that with that price you can buy enough of the food to meet the average daily requirement of both protein and fiber. This price is the height of each bars.
- Right axis has a scale for protein or fiber content in mg per 100 grams of the food.
- Height of red dots denote the fiber content.
- Height of green dots denote the protein content.

6. Further Developments:

1. Include a larger set of data, comprising of all non-processed, basic foods found in all geographical reasons of the earth.
2. Further develop the study to not just recommend food-items but to recommend a complete meal plan , not just for protein rich foods but for any macronutrients requirement or diet-

specific meal plans.

3. Develop price analysis further to produce localised price analysis for different market reasons.
4. Include environmental impact assessment of each food-list.
5. Season factor analysis as price and availability differs from season to season.
6. Create a data product GUI where users can design their own meal.

References:

[1] Clemens, R., Kranz, S., Mobley, A. R., Nicklas, T. A., Raimondi, M. P., Rodriguez, J. C., ... & Warshaw, H. (2012). Filling America's fiber intake gap: summary of a roundtable to probe realistic solutions with a focus on grain-based foods. The Journal of nutrition, 142(7), 1390S-1401S.

[2] Xu, H., Rossi, M., Campbell, K. L., Sencion, G. L., Ärnlöv, J., Cederholm, T., ... & Carrero, J. J. (2016). Excess protein intake relative to fiber and cardiovascular events in elderly men with chronic kidney disease. Nutrition, Metabolism and Cardiovascular Diseases, 26(7), 597-602.

[3] Deborah Fleischer, Green Impact (2018), Almond Milk is Taking a Toll on the Environment, https://sustainability.ucsf.edu/1.713, Retrieved: Jan 9, 2022.

[4] Julia Belluz (2019), Nearly all Americans fail to eat enough of this actual superfood, https://www.vox.com/2019/3/20/18214505/fiber-diet-weight-loss, Retrieved January 9, 2021.