Inf2 - Foundations of Data Science 2021-22

October 28, 2024

Inf2-FDS 2024-25: Coursework 1 - Data wrangling and visualisation

Release date, Submission deadline and Late submission rules: See **Coursework Planner** and **Assessment** Section on FDS Learn pages

It is very important that you read and follow the instructions below to the letter: you will be deducted marks for not following the advice below.

Good Scholarly Practice

Please remember the University requirement as regards all assessed work for credit. Details about this can be found at: http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Specifically, this coursework **must be your own work**. We want you to be able to discuss the class material with each other, but the coursework you submit must be your own work. You are free to form study groups and discuss the concepts related to, and the high-level approach to the coursework. **You may never share code or share write-ups**. It is also not permitted to discuss this coursework on Piazza. The only exception is that if you believe there is an error in the coursework, you may ask a private question to the instructors, and if we feel that the issue is justified, we will send out an announcement.

Assessment information and criteria

- This coursework counts for 20% of the marks of Foundations of Data Science.
- The coursework is designed to assess the following course Learning Outcomes:
    1. Describe and apply good practices for storing, manipulating, summarising, and visualising data.
    2. Use standard packages and tools for data analysis and describing this analysis, such as Python and LaTeX.
- It is marked based on the PDF export from a Jupyter notebook (see **General instructions** below), which you are to submit via Gradescope (see **Assessment/Coursework 1 - Data wrangling and visualisation** folder in Learn).
- The assignment is marked out of 100 and the number of points is indicated by each question.
- We will assess your work on the following criteria:
    - functional code that

(a) performs the computations asked for, as measured by verifying some of the numeric outputs from your processing and reading your code if there is doubt about what you have done and
(b) is clear and makes good use of Pandas functions

– the quality of the visualisations, measured against the *Visualisation principles and guidance* handout used in the S1 Week 5 workshop

– the quality of your textual comments - as measured by how accurate, clear, complete and insightful they are.

General instructions

- Read the instructions carefully, answering only what is required.
- Make sure you are running a Jupyter Notebook that saves PDFs in the correct way, via LaTeX – using Noteable with the instructions here works.
- Read through all of a question before starting, as some parts build on each other.
- In order to understand the meaning of the datasets, you may need to follow the links citing the data.
- Fill in your answers in the cells indicated. You may delete text like "Your answer to Q1.2 goes here". **Do not edit or delete any other cells.**
- Once you have finished a question, use a Jupyter notebook server to export your PDF, by selecting **File→Download as→PDF via LaTeX (.pdf)**. Check this PDF document looks as you expect it to. If changes are needed, update the notebook and export again.
- We ask for multiple types of responses:
  1. Numeric responses, which we will use to verify that the processing has been done correctly.
  2. Visualisations, which will be assessed using the parts of the *Visualisation principles and guidance* PDF that apply to the visualisation in question.
  3. Comments on your findings, which may ask you to describe, explain or interpret your results, and which may ask you to reflect on your design choices. These comments will be assessed on how accurate, complete and insightful they are.
- Keep your answers brief and concise - in the PDF output, textual answers should remain inside the box.
- For answers involving numerical values, use correct units where appropriate and format floating point values to a reasonable number of decimal places.
- For answers involving visualisations:
  – Make sure to label the visualisations clearly and provide titles, and legends where necessary, as per the *Visualisation Principles and Guidance*.
  – All visualisations have to fit on one page. For example, 3 graphs on 3 separate pages counts as 3 visualisations; We would only mark one of these 3 visualisations.
  – "One visualisation" could be a figure with multiple subplots, for example created using the Matplotlib subplot and grid system.

  - Creating plots in separate code cells that appear on the same page is not recommended. We may consider both plots as one visualisation, but mark you down for it. At worst we may only mark one visualisation.
  - Please follow the guidance on font size in the *Visualisation Principles and Guidance*, i.e. a real font size of 8 points at a minimum. We instruct the markers to view all submissions at the same magnification and not to zoom in, to give a fair comparison between students.

- We strongly recommend restarting the kernel and running all (**Kernel→Restart & Run all**) from time to time, to test your code works properly.
- Once you have finished all the questions, submit the final PDF using the submission instructions in the **Assessment→Coursework 1 - Data wrangling and visualisation** folder in Learn. **Please allow enough time to upload your PDF before the deadline.**

How long should I spend on this coursework?

We have designed the coursework to take around 10 hours for a student who has completed the lab sessions, attended the lectures and workshops and done the comprehension questions. There will be some spread around this value, but if you're up-to-date with the course, and you're taking much longer than 10 hours (say 14 hours), then you might want to consider how many marks you are gaining for each extra hour of work.

Good luck - we hope you enjoy it!

[59]:
```
# Imports - run this cell first, and add your own imports here.
# You can use any package you want, but we suggest you stick to ones used in␣
 ↪the labs
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import seaborn as sns
import numpy as np

matplotlib.rcParams['figure.dpi'] = 150 # Make figures have reasonable␣
 ↪resolution when exporting

# My imports
import os
```

Overview of the coursework and the Food Balance and Emissions datasets

In Question 1, we will start exploring the Food balance dataset, which we have downloaded from the Food and Agriculture Organization (FAO) of the United Nations[1]. The Food Balance dataset includes detailed information on how much food a country in Europe has supplied during a certain time period (2010-2021). The dataset has the following columns:

- **Area Code**: Code of the country.
- **Area**: Name of the country.
- **Item Code**: A code representing the specific item being reported, for example, 2511 for "Wheat and products".
- **Item**: The name of the specific item being reported, for example, "Wheat and products". This column also contains rows labelled "Population", which reports the population of the corresponding country, and "Grand Total", which shows the total food supply.
- **Element Code**: A code for the aspect of the food measured.
- **Element**: The aspect of the food measured, for example, "Protein supply quantity (g/capita/day)".
- **Unit**: The unit used for measurement in the Element column, for example, "kilograms" or "kilocalories".
- **Y2010**: The food supply value for 2010. Similar columns are available for the remaining years up to 2021, i.e. Y2011, Y2012, …

Thus the data tells us how much food a country had available and what happened to it. It can also tell a story about how the food is used - whether it's exported, used for animals, or processed into other products.

In Question 2, will focus on the Emissions dataset[2], which includes greenhouse gas emissions produced by agricultural and food systems, encompassing gases like methane, nitrous oxide, and carbon dioxide, along with emissions from industrial processes. The data spans from 1961 to 2020, with predictions available for 2030 and 2050 in some cases. Emissions are measured in kilotonnes of CO2 equivalent and originate from various sources within agriculture and food production, including farming, land use changes, and food processing. This dataset comprises the following columns:

- **Area Code**: Code of the country.
- **Area**: Name of the country.
- **Item Code**: A code representing the specific item being reported, for example, 5060 for "Rice Cultivation".
- **Item**: Name of the specific item being reported, for example, "Rice Cultivation".
- **Element Code**: A code representing the specific element or aspect of the item being measured, for example, 7225 for "Emissions (CH4)".
- **Element**: Name of the specific element or aspect of the item being measured, for example, "Emissions (CH4)".

- **Unit**: The unit of measurement used for the data, for example, "kilotonnes (kt)".
- **Y1961**: The emission value for the year 1961. Similar columns are available for the rest of the years.

We'll combine Food Balance data with Emissions data in Question 3 to gain a more comprehensive understanding. Through a series of questions and data manipulation tasks, we will create meaningful visualisations and perform trend analysis to identify patterns regarding food availability, usage, and environmental factors of food production in various European countries. These insights can aid in agricultural planning and facilitate sustainable decision-making to ensure an environmentally friendly food supply. The questions are organised so that the first few questions give more detailed instructions and Question 4 is more open-ended. We also want you to consider whether there are any limitations of the data that should be considered. Finally, the datasets used in this notebook are not large enough to cause issues, but do consider computational complexity and efficiency of your code.

[1] FAO. 2022. Food Balances (2010-). Accessed on 9 October 2024. https://www.fao.org/faostat/en/#data/FBS Licence: CC-BY-NC-4.0.

[2] FAO. 2022. Emissions totals. Accessed on 9 October 2024. https://www.fao.org/faostat/en/#data/GT Licence: CC-BY-NC-4.0.

Question 1 (24 points) - Exploring the Food Balance and Emissions datasets

Let's dive into this journey of exploring the food and emissions datasets, unravelling insights that may contribute to a healthier lifestyle and a more vibrant Earth.

Question 1.1 (2 points)

We have downloaded data from the FAO website, extracted data relating to Europe and done some preproccessing to simplify the data.

- Read in the Food Balance and Emissions datasets. They are located in the folder `dataset` and are called `FoodBalance_Europe.csv` and `Emissions_Europe.csv` respectively. Name the dataframes created by reading in the files `foodbalance_df` and `emissions_df`.
- Then, **print** the number of **rows** and **columns** in the **both** datasets in the format:

`<Dataset>: <n> rows and <m> columns`

Here `<Dataset>` is a string that makes it clear which dataset you're referring to.

[60]:
```
# Your code for Q1.1 goes here
foodbalance_df = pd.read_csv('dataset/FoodBalance_Europe.csv')
emissions_df = pd.read_csv('dataset/Emissions_Europe.csv')

fb_rows, fb_columns = foodbalance_df.shape
em_rows, em_columns = emissions_df.shape

print(f"foodbalance_df: {fb_rows} rows and {fb_columns} columns")
print(f"emissions_df: {em_rows} rows and {em_columns} columns")
```

```
foodbalance_df: 12706 rows and 19 columns
emissions_df: 7969 rows and 70 columns
```

[61]:
```
# Playground: Use this box to run any additional code for exploration purposes␣
↪(e.g., printing data, debugging).
# Feel free to view the data, test code, and so t here.
# Please remember to delete this box before exporting the final notebook as a␣
↪PDF. We do not mark codes and analysis were done here.
```

Question 1.2 (3 points)

Run `foodbalance_df.info()` and `emissions_df.info()` in the cell below. Mention **3** important observations that are true of both datasets. Please make sure that your answer is no longer than 3 lines, i.e. stays in the box in the PDF file you have produced from this notebook. You can delete the text saying "Your answer goes here".

[62]:
```
# Q1.2
# Run this code
foodbalance_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12706 entries, 0 to 12705
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Area Code     12706 non-null  int64
 1   Area          12706 non-null  object
 2   Item Code     12706 non-null  int64
 3   Item          12706 non-null  object
 4   Element Code  12706 non-null  int64
 5   Element       12706 non-null  object
 6   Unit          12706 non-null  object
 7   Y2010         12348 non-null  float64
 8   Y2011         12348 non-null  float64
 9   Y2012         12354 non-null  float64
 10  Y2013         12360 non-null  float64
 11  Y2014         12155 non-null  float64
 12  Y2015         12158 non-null  float64
 13  Y2016         12171 non-null  float64
 14  Y2017         12172 non-null  float64
 15  Y2018         12193 non-null  float64
 16  Y2019         12176 non-null  float64
 17  Y2020         12152 non-null  float64
 18  Y2021         12169 non-null  float64
dtypes: float64(12), int64(3), object(4)
memory usage: 1.8+ MB
```

[63]:
```
# Q1.2
# Run the code
emissions_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7969 entries, 0 to 7968
Data columns (total 70 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Area Code     7969 non-null   int64
```

```
1   Area          7969 non-null   object
2   Item Code     7969 non-null   int64
3   Item          7969 non-null   object
4   Element Code  7969 non-null   int64
5   Element       7969 non-null   object
6   Unit          7969 non-null   object
7   Y1961         963 non-null    float64
8   Y1962         963 non-null    float64
9   Y1963         963 non-null    float64
10  Y1964         963 non-null    float64
11  Y1965         963 non-null    float64
12  Y1966         963 non-null    float64
13  Y1967         963 non-null    float64
14  Y1968         963 non-null    float64
15  Y1969         963 non-null    float64
16  Y1970         963 non-null    float64
17  Y1971         963 non-null    float64
18  Y1972         963 non-null    float64
19  Y1973         963 non-null    float64
20  Y1974         963 non-null    float64
21  Y1975         963 non-null    float64
22  Y1976         963 non-null    float64
23  Y1977         963 non-null    float64
24  Y1978         963 non-null    float64
25  Y1979         963 non-null    float64
26  Y1980         963 non-null    float64
27  Y1981         963 non-null    float64
28  Y1982         963 non-null    float64
29  Y1983         963 non-null    float64
30  Y1984         963 non-null    float64
31  Y1985         963 non-null    float64
32  Y1986         963 non-null    float64
33  Y1987         963 non-null    float64
34  Y1988         963 non-null    float64
35  Y1989         963 non-null    float64
36  Y1990         4925 non-null   float64
37  Y1991         4931 non-null   float64
38  Y1992         6628 non-null   float64
39  Y1993         6827 non-null   float64
40  Y1994         6841 non-null   float64
41  Y1995         6848 non-null   float64
42  Y1996         6853 non-null   float64
43  Y1997         6856 non-null   float64
44  Y1998         6859 non-null   float64
45  Y1999         6861 non-null   float64
46  Y2000         7100 non-null   float64
47  Y2001         7095 non-null   float64
48  Y2002         7111 non-null   float64
```

```
 49  Y2003          7127 non-null   float64
 50  Y2004          7121 non-null   float64
 51  Y2005          7116 non-null   float64
 52  Y2006          7275 non-null   float64
 53  Y2007          7281 non-null   float64
 54  Y2008          7276 non-null   float64
 55  Y2009          7273 non-null   float64
 56  Y2010          7258 non-null   float64
 57  Y2011          7261 non-null   float64
 58  Y2012          7262 non-null   float64
 59  Y2013          7271 non-null   float64
 60  Y2014          7263 non-null   float64
 61  Y2015          7240 non-null   float64
 62  Y2016          7238 non-null   float64
 63  Y2017          7246 non-null   float64
 64  Y2018          7241 non-null   float64
 65  Y2019          7240 non-null   float64
 66  Y2020          7246 non-null   float64
 67  Y2021          6130 non-null   float64
 68  Y2030          1426 non-null   float64
 69  Y2050          1426 non-null   float64
dtypes: float64(63), int64(3), object(4)
memory usage: 4.3+ MB
```

Your answer to Q1.2 goes here: 1. They both contain the same column names, such as Area, Item, etc. 2. They both include columns measured by year, but emissions_df has a broader time span. 3. foodbalance_df has more non-null data, which indicates that it has a larger data volume and better data integrity.

Question 1.3 (5 points)

In this subquestion, we'll investigate the structure of the food balance data.

1. Calculate and **print**:
   - the number of unique **Items** in the Food Balance dataset (**Do not remove** the 'Population' and 'Grand Total' values in the 'Item' column in this stage).
   - the number of unique **Areas** (i.e. countries) in the Food Balance dataset.

   Print the results in the following format:

   ```
   Number of unique items in the Food Balance dataset: <value>
   Number of unique areas in the Food Balance dataset: <value>
   ```

2. **Print** the unique values in the 'Item' column of the `foodbalance_df` dataframe.
3. Calculate and **print** the count of each unique value in 'Element' column of the `foodbalance_df` dataframe.
4. Comment on what the output in parts 1-3 tell you about the structure of the food balance dataset.

[64]:
```
## Your code for Q1.3 part 1 goes here
food_uniI_N = foodbalance_df['Item'].nunique()
food_uniA_N = foodbalance_df['Area'].nunique()

print(f"Number of unique items in the Food Balance dataset: {food_uniI_N}")
print(f"Number of unique areas in the Food Balance dataset: {food_uniA_N}")
```

```
Number of unique items in the Food Balance dataset: 21
Number of unique areas in the Food Balance dataset: 39
```

[65]:
```
## Your code for Q1.3 part 2 goes here
food_uniI = foodbalance_df['Item'].unique()

for item in food_uniI:
    print(item)
```

```
Population
Grand Total
Cereals - Excluding Beer
Starchy Roots
Sugar Crops
Sugar & Sweeteners
Pulses
Treenuts
Oilcrops
Vegetable Oils
Vegetables
Fruits - Excluding Wine
Stimulants
```

```
Spices
Alcoholic Beverages
Meat
Offals
Animal fats
Eggs
Milk - Excluding Butter
Miscellaneous
```

[66]:
```
## Your code for Q1.3 part 3 goes here
food_uniE = foodbalance_df['Element'].value_counts()

print(food_uniE)
```

```
Element
Domestic supply quantity                  780
Food supply (kcal)                        753
Food supply (kcal/capita/day)             753
Protein supply quantity (g/capita/day)    753
Protein supply quantity (t)               753
Fat supply quantity (t)                   753
Fat supply quantity (g/capita/day)        753
Residuals                                 741
Import Quantity                           741
Export Quantity                           739
Stock Variation                           722
Food supply quantity (kg/capita/yr)       714
Food                                      714
Production                                648
Losses                                    560
Processing                                444
Other uses (non-food)                     419
Feed                                      389
Tourist consumption                       346
Seed                                      192
Total Population - Both sexes              39
Name: count, dtype: int64
```

Your answer for Q1.3 part 4 goes here

1. Part 1 shows that there are 21 unique items and 39 unique areas in the food balance dataset.
2. Part 2 displays what these 21 unique items are in the food balance dataset, such as Sugar Crops, Sugar & Sweeteners, etc.
3. Part 3 reveals what the elements are in the food balance dataset and lists the occurrences of these elements. These elements indicate how food is used in various aspects.

Question 1.4 (6 points)

In the Food Balance data,

1. Separate out the Population and Grand Total items from the specific items as follows:
   - Select rows with Items equal to "Population" or "Grand Total" and keep those rows in a separate dataframe named `foodbalance_extra_df`.
   - Remove those rows from `foodbalance_df`. The `foodbalance_df` dataframe should not include "Population" and "Grand Total" items any more, covering only food types after this preprocessing. "Population" shows the total population of each country, while "Grand Total" shows the overall food supply. Since neither provides data for individual food items, we want to keep them separate to avoid affecting our future calculations.
   - **Print** the new number of rows and columns in the preprocessed `foodbalance_df` and `foodbalance_extra_df` in the following format:

   ```
   Food Balance dataset: <n> rows and <m> columns
   Food Balance extra data: <n> rows and <m> columns
   ```

2. **Print** the total number of NaN values in the preprocessed `foodbalance_df` in the following format:

```
Total number of NaN values in the preprocessed Food Balance dataset: <Number of NaN values>
```

3. How much data is missing is in which years? Write some code to find out and describe the pattern of the data. Please make sure that your answer is no longer than 3 lines, i.e. stays in the box in the PDF file you have produced from this notebook. You can delete the text saying "Your answer goes here".

[67]:
```
## Your code for Q1.4 part 1 goes here
foodbalance_extra_df = foodbalance_df[(foodbalance_df['Item']=='Population') |␣
 ↪(foodbalance_df['Item']=='Grand Total')]
foodbalance_df = foodbalance_df[~((foodbalance_df['Item'] == 'Population') |␣
 ↪(foodbalance_df['Item'] == 'Grand Total'))]

print(f"Food Balance dataset: {foodbalance_df.shape[0]} rows and␣
 ↪{foodbalance_df.shape[1]} columns")
print(f"Food Balance extra data: {foodbalance_extra_df.shape[0]} rows and␣
 ↪{foodbalance_extra_df.shape[1]} columns")
```

```
Food Balance dataset: 12394 rows and 19 columns
Food Balance extra data: 312 rows and 19 columns
```

[68]:
```
## Your code for Q1.4 part 2 goes here
food_nan_total = foodbalance_df.isna().sum().sum()

print(f"Total number of NaN values in the preprocessed Food Balance dataset:␣
 ↪{food_nan_total}")
```

```
Total number of NaN values in the preprocessed Food Balance dataset: 5716
```

[69]:
```
## Your code for Q1.4 part 3 goes here
food_null_total = foodbalance_df.isnull().sum()

print(f"Null per year:{food_null_total}")
```

```
Null per year:Area Code         0
Area               0
Item Code          0
Item               0
Element Code       0
Element            0
Unit               0
Y2010            358
Y2011            358
Y2012            352
Y2013            346
Y2014            551
Y2015            548
Y2016            535
Y2017            534
Y2018            513
Y2019            530
Y2020            554
Y2021            537
dtype: int64
```

Your answer to Q1.4 part 3 goes here:

In earlier years (before 2014), there is significantly more missing data, with at least about 200 fewer data entries compared to after 2014.

print(foodbalance_df.head())

Question 1.5 (8 points)

1. Add a column to `foodbalance_df` called "Total", that contains the total of the columns relating to the years 2012 to 2021 for each row.
2. Print the top 3 countries (in descending order) with the highest total **"Production"** of "Alcoholic Beverages" from 2012 to 2021. For each of the 3 countries show the country name ("Area"), the total Production value, and the Units. You can use a data frame to display the output.
3. For the top 3 countries reported above, for the Alcoholic Beverages, calculate the total "Production", "Import Quantity", "Export Quantity", "Domestic supply quantity", and "Losses" (**Hint:** find them in Element column) from 2012 to 2021. Make sure to show the unit for each. You can use a data frame to display the output.
4. According to the observed values, compare these three countries regarding alcoholic beverages (e.g. which country had the highest alcoholic beverages production?). Please make sure that your answer is no longer than 5 lines, i.e. stays in the box in the PDF file you have produced from this notebook. You can delete the text saying "Your answer goes here".

[70]:
```
# Your code for Q1.5 part 1 goes here
perYear = [f'Y{year}'for year in range (2012,2022)]

food_copy = foodbalance_df.copy()
food_copy['Total'] = food_copy[perYear].sum(axis = 1)
```

[71]:
```
# Your code for Q1.5 part 2 goes here
food_AB = food_copy[(food_copy['Item'] == 'Alcoholic Beverages') &␣
 ↪(food_copy['Element'] == 'Production')].copy()

food_AB['Total Production'] = food_AB[perYear].sum(axis=1)

topC =  food_AB[['Area','Total Production','Unit']].sort_values(by = 'Total␣
 ↪Production', ascending=False).head(3)

print(topC)
```

```
                  Area  Total Production    Unit
4077             Germany          100311.0  1000 t
9979  Russian Federation           97665.0  1000 t
3736              France           84026.0  1000 t
```

[72]:
```
# Your code for Q1.5 part 3 goes here
arranged_country = []

for index, row in topC.iterrows():
    area = row['Area']
    country_data = foodbalance_df[ (foodbalance_df['Area'] ==␣
 ↪area)&(foodbalance_df['Item'] == 'Alcoholic Beverages')]
```

```
    total_Production = country_data[country_data['Element'] ==␣
↪'Production'][perYear].sum(axis=1).sum()
    total_Import_Quantity = country_data[country_data['Element'] == 'Import␣
↪Quantity'][perYear].sum(axis=1).sum()
    total_Export_Quantity = country_data[country_data['Element'] == 'Export␣
↪Quantity'][perYear].sum(axis=1).sum()
    total_Domestic_Supply_Quantity = country_data[country_data['Element'] ==␣
↪'Domestic supply quantity'][perYear].sum(axis=1).sum()
    total_Losses = country_data[country_data['Element'] == 'Losses'][perYear].
↪sum(axis=1).sum()

    unit = country_data['Unit'].iloc[0]

    arranged_country.append({
        'Area': area,
        'Total Production': total_Production,
        'Total Import Quantity': total_Import_Quantity,
        'Total Export Quantity': total_Export_Quantity,
        'Total Domestic Supply Quantity':  total_Domestic_Supply_Quantity,
        'Total Losses': total_Losses,
        'Unit': unit
    })

arranged_country_df = pd.DataFrame(arranged_country)
print(arranged_country_df[['Area', 'Total Production', 'Unit', 'Total Import␣
↪Quantity', 'Unit','Total Export Quantity','Unit',
                           'Total Domestic Supply Quantity', 'Unit','Total␣
↪Losses','Unit']])
```

```
                Area  Total Production    Unit  Total Import Quantity  \
0            Germany          100311.0  1000 t                36404.0
1  Russian Federation           97665.0  1000 t                13710.0
2             France           84026.0  1000 t                20615.0

     Unit  Total Export Quantity    Unit  Total Domestic Supply Quantity  \
0  1000 t                26495.0  1000 t                         112133.0
1  1000 t                 5527.0  1000 t                         108031.0
2  1000 t                32551.0  1000 t                          71070.0

     Unit  Total Losses    Unit
0  1000 t           0.0  1000 t
1  1000 t           0.0  1000 t
2  1000 t         573.0  1000 t
```

Your answer to Q1.5 part 4 goes here:

Germany has the highest production and supply, with 100,311 kt and 112,133 kt, respectively; Russia follows with 97,665 kt and 108,031 kt; France is the lowest with 84,026 kt and 71,070 kt. Germany has the most imports, while France leads in exports and losses.

Question 2 (20 points) - Merging Food Balance and Emissions Datasets

Food production is the biggest source of emissions of nitrous oxide (N2O), which is a major greenhouse gas[1]. N2O comes from fertilizer and manure used on crops. N2O emissions could be reduced by efficiency improvements in the use of fertilizer, and by reducing the consumption of meat and dairy products[1].

We would like to compare the N2O emissions data with food supply information for each country. In this question we will extract N2O emissions relating to food in each country and year from `emissions_df`, and then combine this data with the food supply data from `foodbalance_df` created in **Q1.4**.

[1] Reay et al. (2012) 'Global agriculture and nitrous oxide emissions'. *Nature Climate Change* 2:410-416. https://doi.org/10.1038/nclimate1458

Question 2.1 (4 points)

In this subquestion we will produce a transformed version of the emissions data and filter it, so that the years it covers correspond to the food balance data.

1. Run the code cell below to create `emissions_long_df`. Explain how the code has transformed the data frame, and why this transformation may be helpful.
2. Remove data from `emissions_long_df` for the years before 2010 and after 2021. Keep the filtered data in a dataframe named `filtered_emissions_long_df`. You may wish to transform the type of the "Year" column to make the filtering easier. **Print** the number of rows in the `filtered_emissions_long_df`.
3. Based on the created dataframe (`filtered_emissions_long_df`), **print** both the **mean** and **standard deviation** of **emissions (N2O)** due to **food household consumption** in the **"United Kingdom of Great Britain and Northern Ireland"** for the years **2010 to 2021** in the following format, with outputs given to **two decimal places**.

```
Mean and std of N2O related to Food Household Consumption
in United Kingdom for 2010 to 2021: mean: <value/unit>, std: <value/unit>
```

[73]:
```
# Run this cell - do not edit!
id_vars = ['Area Code', 'Area', 'Item Code', 'Item', 'Element Code', 'Element',␣
 ↪'Unit']
emissions_long_df = emissions_df.melt(id_vars, var_name='Year')
emissions_long_df.head()
```

[73]:

	Area Code	Area	Item Code	Item	Element Code	\
0	3	Albania	5064	Crop Residues	7234	
1	3	Albania	5064	Crop Residues	7236	
2	3	Albania	5064	Crop Residues	7230	
3	3	Albania	5064	Crop Residues	724313	
4	3	Albania	5064	Crop Residues	723113	

	Element	Unit	Year	value
0	Direct emissions (N2O)	kt	Y1961	0.0721
1	Indirect emissions (N2O)	kt	Y1961	0.0162
2	Emissions (N2O)	kt	Y1961	0.0883
3	Emissions (CO2eq) from N2O (AR5)	kt	Y1961	23.3995
4	Emissions (CO2eq) (AR5)	kt	Y1961	23.3995

Your answer to 2.1 Part 1 goes here

This code associates each index in the list with its actual value, making information about countries, items, etc., clearer. It also transforms the data into a sorted list, making it easier to find corresponding values and reflect regions, quantities, and so on.

[74]:
```
# Your code for Q2.1 part 2 goes here

emissions_long_df['Year'] = emissions_long_df['Year'].str.replace('Y', '').
 ↪astype(int)
```

```
filtered_emissions_long_df = emissions_long_df[(emissions_long_df['Year'] >=␣
 ↪2010) & (emissions_long_df['Year'] <= 2021)]

print("number of rows ")
print(f"emissions_long_df:{len(emissions_long_df)}")
print(f"filtered_emissions_long_df:{len(filtered_emissions_long_df)}")
```

```
number of rows
emissions_long_df:502047
filtered_emissions_long_df:95628
```

[75]:
```
# Your code for Q2.1 part 3 goes here

UK_emission = filtered_emissions_long_df[
    (filtered_emissions_long_df['Area'] == 'United Kingdom of Great Britain and␣
 ↪Northern Ireland') &
    (filtered_emissions_long_df['Element'] == 'Emissions (N2O)') &
    (filtered_emissions_long_df['Year'].between(2010, 2021))
]

meanN2O_UK = UK_emission['value'].mean()
stdN2O_UK = UK_emission['value'].std()

print(f"Mean and std of N2O related to Food Household Consumption\n"
      f"in United Kingdom for 2010 to 2021: mean: {meanN2O_UK:.2f}/{unit}, std:␣
 ↪{stdN2O_UK:.2f}/{unit}")
```

```
Mean and std of N2O related to Food Household Consumption
in United Kingdom for 2010 to 2021: mean: 5.23/1000 t, std: 6.01/1000 t
```

Question 2.2 (3 points)

Here we will create a dataframe called `food_emissions_df` that contains information about N2O emissions from food. To do this:

- **Remove** rows in `filtered_emissions_long_df` not related to food production: 'Energy', 'International bunkers', 'IPPU', 'Other', 'Waste'
- **Select** rows containing the element of "Emissions (N2O)"
- **Store** the resulting data frame as `food_emissions_df`.
- **Rename** the 'value' column to 'Emissions (N2O)'
- **Print** the column names of `food_emissions_df`
- **Print** the number of rows in `food_emissions_df`.

[76]:
```
# Your code for Q2.2 goes here

filtered_emissions_long_df = filtered_emissions_long_df[␣
 ↪~filtered_emissions_long_df['Item'].isin(['Energy', 'International bunkers',␣
 ↪'IPPU', 'Other', 'Waste'])]

food_emissions_df =␣
 ↪filtered_emissions_long_df[filtered_emissions_long_df['Element']=='Emissions␣
 ↪(N2O)'].copy()

food_emissions_df.rename(columns={'value': 'Emissions (N2O)'}, inplace=True)

print("columm name")
print(food_emissions_df.columns)

print(f"Number of rows in food_emissions_df: {food_emissions_df.shape[0]}")
```

```
columm name
Index(['Area Code', 'Area', 'Item Code', 'Item', 'Element Code', 'Element',
       'Unit', 'Year', 'Emissions (N2O)'],
      dtype='object')
Number of rows in food_emissions_df: 12756
```

Question 2.3 (4 points)

Use `food_emissions_df` (created in **Q2.2**) to calculate the total N2O emissions due to food in each year for each country.

1. Name the resulting dataframe `total_food_emissions_df`. **Print** the number of rows in `total_food_emissions_df`.
2. Print the three countries with the highest N2O emissions in 2021, including the "Area" and "Unit" columns in your output.
3. Are these the European countries you might expect to have the highest N2O emissions? Explain the reasoning behind your answer briefly.

[77]:
```
# Your code for Q2.3 Part 1 goes here

total_food_emissions_df = food_emissions_df.groupby(['Area','Year','Unit'],␣
 ↪as_index=False)['Emissions (N2O)'].sum()
total_food_emissions_df = total_food_emissions_df[['Area', 'Year','Emissions␣
 ↪(N2O)', 'Unit']]
print(f"number of rows:{total_food_emissions_df.shape[0]}")
```

```
number of rows:648
```

[78]:
```
# Your code for 2.3 Part 2 goes here

topC_2021 = total_food_emissions_df[total_food_emissions_df["Year"] == 2021].
 ↪nlargest(3, 'Emissions (N2O)')
print("top 3 Country with the heightest Emissions (N2O) in 2021")
print(topC_2021)
```

```
top 3 Country with the heightest Emissions (N2O) in 2021
                  Area  Year  Emissions (N2O) Unit
491  Russian Federation  2021         245.4877   kt
203              France  2021         109.4676   kt
215             Germany  2021          93.8710   kt
```

Your answer for Q2.2 Part 3 goes here

Russia has a very large area and a highly developed food manufacturing industry. The country widely uses fertilizers in agriculture, especially nitrogen fertilizers. During the transformation process of nitrogen fertilizers in the soil, N2O gas is released. Therefore, the high usage rate of fertilizers may directly lead to an increase in N2O emissions.

France is smaller in area compared to Russia, and its food demand is relatively lower than that of Russia.

Germany has strict regulations and standards for greenhouse gas emissions and actively promotes environmental protection measures, striving to achieve carbon neutrality. This may result in Germany's overall emissions being lower than those of other countries.

Question 2.4 (5 points)

We will now create a dataframe that, for each combination of Area (Country) and Year, contains the total N2O emissions due to food and the total food supply.

1. **Convert** `foodbalance_extra_df` into a dataframe `total_food_supply_df`, with a similar format `emissions_long_df` that you created in **Q2.1**.
2. **Filter** `total_food_supply_df` on the 'Food supply (kcal)' Element and the 'Grand Total' Item.
3. **Rename** the 'value' column to 'Grand Total'
4. **Merge** the `total_food_supply_df` and `total_food_emissions_df` (created in **Q2.3**). The new dataframe should contain emissions and food balance information for each country for each year from 2010 to 2021. Call the dataframe `merged_food_emissions_df`.
   - **Print** the number of rows in `total_food_supply_df`, `total_food_emissions_df` and `merged_food_emissions_df`.
   - **Print** the head of the `merged_food_emissions_df`.

[79]:
```
# Your code for Q2.4 goes here

total_food_supply_df = foodbalance_extra_df.
 ↪loc[(foodbalance_extra_df['Element']== 'Food supply (kcal)')&
(foodbalance_extra_df['Item'] == 'Grand Total')].copy()

total_food_supply_df.rename(columns={'value': 'Grand Total'}, inplace=True)
total_food_supply_df = total_food_supply_df.melt(id_vars, var_name='Year')
total_food_supply_df['Year'] = total_food_supply_df['Year'].str.replace('Y',␣
 ↪'').astype(int)

merged_food_emissions_df = pd.merge(total_food_emissions_df,␣
 ↪total_food_supply_df, on=['Area', 'Year'], how='inner')


print("Number of rows in total_food_supply_df:", total_food_supply_df.shape[0])
print("Number of rows in total_food_emissions_df:", total_food_emissions_df.
 ↪shape[0])
print("Number of rows in merged_food_emissions_df:", merged_food_emissions_df.
 ↪shape[0])

print(merged_food_emissions_df.head())
```

```
Number of rows in total_food_supply_df: 468
Number of rows in total_food_emissions_df: 648
Number of rows in merged_food_emissions_df: 468
      Area  Year  Emissions (N2O) Unit_x  Area Code  Item Code         Item  \
0  Albania  2010           3.6723     kt          3       2901  Grand Total
1  Albania  2011           3.8170     kt          3       2901  Grand Total
```

2	Albania	2012	3.8186	kt	3	2901	Grand Total
3	Albania	2013	3.7191	kt	3	2901	Grand Total
4	Albania	2014	3.7948	kt	3	2901	Grand Total

	Element Code	Element	Unit_y	value
0	661	Food supply (kcal)	million Kcal	3462488.58
1	661	Food supply (kcal)	million Kcal	3490045.25
2	661	Food supply (kcal)	million Kcal	3532407.27
3	661	Food supply (kcal)	million Kcal	3522042.97
4	661	Food supply (kcal)	million Kcal	3492255.06

Question 2.5 (4 points)

1. Use your discretion to remove unnecessary columns from `merged_food_emissions_df` and rename any columns so that they are as meaningful as possible. Remember, the aim is to be able to compare for each year and for each country the Grand Total of food supply and the Emissions (N2O).
2. Select the `merged_food_emissions_df` corresponding to the year 2010, sorted in descending order of Emissions (N2O). Print the first 5 of these rows.
3. Comment on what the output shows.

[80]:
```
# Your code for Q2.5 part 1 goes here

merged_food_emissions_df = merged_food_emissions_df.drop(columns=['Element',␣
 ↪'Area Code', 'Item Code', 'Element Code','Item'])
merged_food_emissions_df.rename(columns={'value': 'Food supply (kcal)'},␣
 ↪inplace=True)
merged_food_emissions_df.rename(columns={'Unit_x': 'Unit_Emission'},␣
 ↪inplace=True)
merged_food_emissions_df.rename(columns={'Unit_y': 'Unit_Food'}, inplace=True)
cols = [col for col in merged_food_emissions_df.columns if col != 'Unit_Food']␣
 ↪+ ['Unit_Food']
merged_food_emissions_df = merged_food_emissions_df[cols]
print(merged_food_emissions_df.head())
```

```
      Area  Year  Emissions (N2O) Unit_Emission  Food supply (kcal)  \
0  Albania  2010          3.6723            kt          3462488.58
1  Albania  2011          3.8170            kt          3490045.25
2  Albania  2012          3.8186            kt          3532407.27
3  Albania  2013          3.7191            kt          3522042.97
4  Albania  2014          3.7948            kt          3492255.06

      Unit_Food
0  million Kcal
1  million Kcal
2  million Kcal
3  million Kcal
4  million Kcal
```

[81]:
```
# Your code for Q2.5 part 2 goes here

merged_food_emissions_2010 =␣
 ↪merged_food_emissions_df[merged_food_emissions_df['Year'] == 2010].
 ↪sort_values(by='Emissions (N2O)', ascending=False)
print(merged_food_emissions_2010.head())
```

```
                                    Area  Year  Emissions (N2O)  \
360                  Russian Federation  2010         355.2769
```

```
132                                                 France  2010         185.8387
144                                                Germany  2010         165.8738
312                                                 Poland  2010         130.3286
456  United Kingdom of Great Britain and Northern I…  2010         122.1775

    Unit_Emission  Food supply (kcal)     Unit_Food
360            kt        1.723729e+08  million Kcal
132            kt        8.106688e+07  million Kcal
144            kt        1.067518e+08  million Kcal
312            kt        4.747855e+07  million Kcal
456            kt        7.604840e+07  million Kcal
```

Your answer to Q2.5 part 3 goes here

The emissions (N2O) of Russia are very high, far exceeding those of other countries, while the emissions from other countries do not show significant differences.

Question 3 (26) - Trends and anomalies across areas

In this question we will see

- if there are any anomalies in the data.
- if there are any trends across areas.

Question 3.1 - Anomaly detection (8 points)

Based on the `total_food_emissions_df`, **created in Q2.1**, check for any anomalies (countries as anomaly) in the data. - Use any appropriate approach in your choice to identify possible anomalies. - Explain your observation in a maximum of 5 lines. Are there any anomalies in the data? Do you think they are outliers? Should we remove them?

If you have not done Q2, feel free to use original dataframe to answer this question. We only mark your final answer.

[82]:
```
# Your code for Q3.1 part 1 goes here

Q1 = total_food_emissions_df['Emissions (N2O)'].quantile(0.25)
Q3 = total_food_emissions_df['Emissions (N2O)'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

anomalies = total_food_emissions_df[(total_food_emissions_df['Emissions (N2O)']␣
 ↪< lower_bound) |
                                    (total_food_emissions_df['Emissions␣
 ↪(N2O)'] > upper_bound)]

print("Detected anomalies:")
print(anomalies)
```

```
Detected anomalies:
                                                  Area  Year  Emissions (N2O)  \
36                                             Belarus  2010          85.0623
37                                             Belarus  2011          88.3914
38                                             Belarus  2012          87.1696
39                                             Belarus  2013          86.0896
40                                             Belarus  2014          83.5646
..                                                 …   …              …
631  United Kingdom of Great Britain and Northern I…  2017         124.7567
632  United Kingdom of Great Britain and Northern I…  2018         123.2473
633  United Kingdom of Great Britain and Northern I…  2019         123.9373
634  United Kingdom of Great Britain and Northern I…  2020         118.3631
635  United Kingdom of Great Britain and Northern I…  2021          85.4072

    Unit
36    kt
37    kt
38    kt
39    kt
40    kt
..   …
631   kt
```

```
632   kt
633   kt
634   kt
635   kt

[105 rows x 4 columns]
```

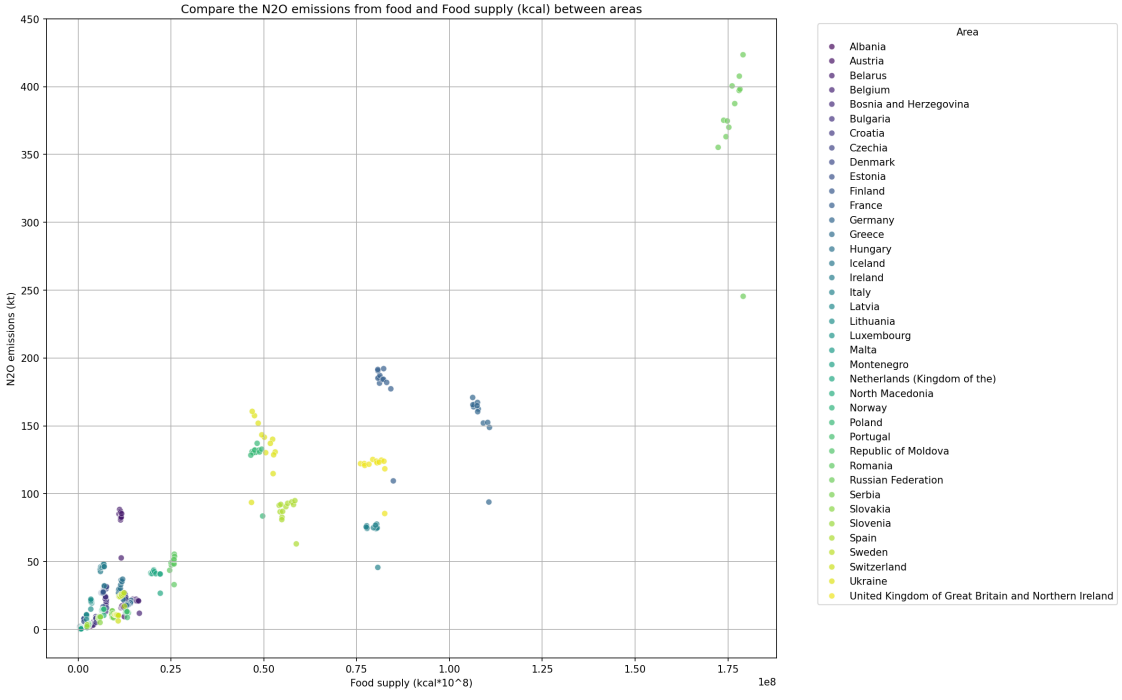
Your answer for Q3.1 part 2 goes here

Through the IQR method, anomalies were identified, showing that the emissions (N2O) from these countries are significantly higher than those from others. However, these anomalies are based on real data collected from various countries and cannot be considered outliers. Furthermore, the countries with high emissions (N2O) generally have large populations and are highly developed in industries such as agriculture and manufacturing,such as Russian and UK, so high emissions (N2O) are to be expected.

Question 3.2, Food and Emission Visualisation (10 points)

Compare the N2O emissions from food and 'Food supply (kcal)' between areas using a visualisation of your choice. You can use either separate DataFrames or the `merged_food_emissions_df` (which you may have created in Q2.3). The points here are for the visualisation - you won't lose points for not having done Q2.

[83]:
```
# Your code for Q3.2 goes here

plt.figure(figsize=(16, 10))
sns.scatterplot(data=merged_food_emissions_df, x='Food supply (kcal)',␣
↪y='Emissions (N2O)', hue='Area', palette='viridis', alpha=0.7, sizes=(600,␣
↪600))

plt.title('Compare the N2O emissions from food and Food supply (kcal) between␣
↪areas')
plt.xlabel('Food supply (kcal*10^8)')
plt.ylabel('N2O emissions (kt)')
plt.legend(title='Area', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.yticks(np.arange(0, 500, 50))
plt.grid(True)

plt.tight_layout()
plt.show()
```



Question 3.3 (8 points)

1. Explain your reasons behind your visualisation design choice and summarise its pros and cons, including any features you would add given more time.
2. Interpret the `Food and Emission` visualisation that you created; i.e. summarise the patterns in the visualisation and any particularly noteworthy features.

Please make sure that your answer is no longer than 30 lines, i.e. stays in the box in the PDF file you have produced from this notebook. You can delete the text saying "Your answer goes here".

Your answer for Q3.3 goes here

PART 1 reason for design: 1.The direct data from each country varies significantly, especially between large and small countries. The N2O emissions also differ greatly. Therefore, a scatter plot is used, which can visually represent the disparities among the data of various countries. 2.The scatter plot clearly displays the vast amounts of data.

Advantage: 1.The scatter plot allows for a clearer view of the distribution of each value. 2.It provides a clear observation of the relationship between N2O emissions and food supply; from the graph, it can be seen that N2O emissions are positively proportional to the amount of food. 3.The scatter plot allows for the observation of data concentration and makes it easier to identify outliers.

disadvantage 1.When there are too many data points, the scatter plot may overlap, making it difficult to distinguish the distribution and trends of the individual data points

PART 2 1.It is clear from the graph that these two variables are positively correlated; as food supply increases, N2O emissions also increase. 2.The values for each country can be observed through different colors. 3.The graph clearly indicates that for developed countries and larger nations, N2O emissions and food supply far exceed those of other countries.

Question 4: Create your own visualisation (30 points)

It's now your opportunity to create your own visualisation of the Food and Emissions data. You can use either the Food balance or Emissions data, or you can use the combined DataFrame created in Question 2.
We also provided you with World Emission and Food datasets (`FoodBalance_Global.csv.gz` and `Emissions_Global.csv.gz`) that include data for all countries worldwide. It is up to you whether to use the European countries' datasets or the global datasets. Please don't bring in external datasets - the idea is that this is a small exercise, not a whole project.

Here are some ideas:

- How have production and consumption patterns of major food commodities (e.g., grains, meat, dairy) changed over the past few decades?
- Are there noticeable shifts in dietary preferences at a global or regional level?
- What are the trends in agricultural emissions, and how are they related to changes in production of foods?

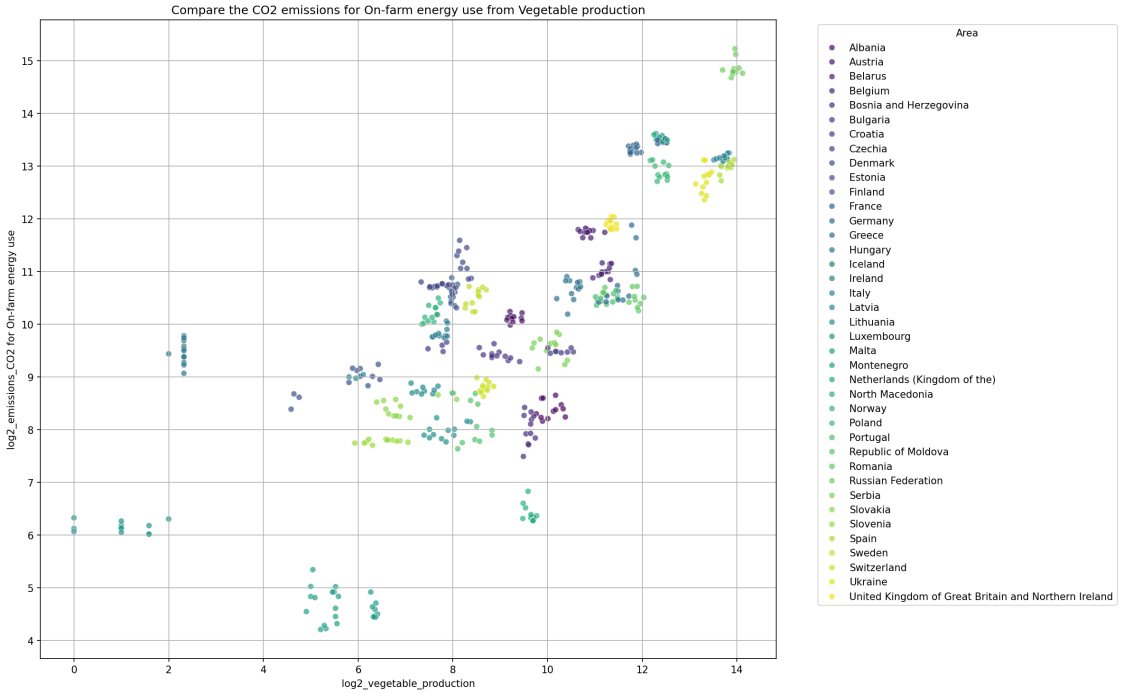
We will give credit for particularly interesting questions, good visualisations (as measured by the criteria in the visualisation principles handout) and insightful interpretation.

Question 4.1 (18 points)

Create your visualisation here.

[84]:
```
# Your code for Q4.1 goes here
food_vege = foodbalance_df[(foodbalance_df['Item'] =='Vegetables') &␣
 ↪(foodbalance_df['Element'] =='Production')]
emision_energy_CO2 = emissions_df[(emissions_df['Item'] == 'On-farm energy␣
 ↪use') & (emissions_df['Element'] == 'Emissions (CO2)')]
id_vars = ['Area Code', 'Area', 'Item Code', 'Item', 'Element Code', 'Element',␣
 ↪'Unit']
emision_energy_CO2 = emision_energy_CO2.melt(id_vars, var_name='Year')
food_vege = food_vege.melt(id_vars, var_name='Year')
emision_energy_CO2['Year'] = emision_energy_CO2['Year'].str.replace('Y', '').
 ↪astype(int)
emision_energy_CO2 = emision_energy_CO2[(emision_energy_CO2['Year'] >= 2010) &␣
 ↪(emision_energy_CO2['Year'] <= 2021)]
food_vege['Year'] =food_vege['Year'].str.replace('Y', '').astype(int)

food_vege_modified = food_vege.drop(columns=['Element', 'Area Code', 'Item␣
 ↪Code', 'Element Code','Item'])
food_vege_modified.rename(columns={'value': 'vegetable_production'},␣
 ↪inplace=True)
unit_column = food_vege_modified.pop('Unit')
food_vege_modified['log2_vegetable_production'] = np.
 ↪log2(food_vege_modified['vegetable_production'])
food_vege_modified['Unit'] = unit_column

emision_energy_CO2_modified = emision_energy_CO2.drop(columns=['Element', 'Area␣
 ↪Code', 'Item Code', 'Element Code','Item'])
emision_energy_CO2_modified.rename(columns={'value': 'Emissions (CO2) for␣
 ↪On-farm energy use'}, inplace=True)
unit_column_E = emision_energy_CO2_modified.pop('Unit')
emision_energy_CO2_modified['Unit'] = unit_column_E
emision_energy_CO2_modified['log2_emissions_CO2'] = np.
 ↪log2(emision_energy_CO2_modified['Emissions (CO2) for On-farm energy use'])

merged_FE = pd.merge(emision_energy_CO2_modified, food_vege_modified,␣
 ↪on=['Area', 'Year'], how='inner')

plt.figure(figsize=(16, 10))
sns.scatterplot(data=merged_FE, x='log2_vegetable_production',␣
 ↪y='log2_emissions_CO2', hue='Area', palette='viridis', alpha=0.7,␣
 ↪sizes=(600, 600))

plt.title('Compare the CO2 emissions for On-farm energy use from Vegetable␣
 ↪production')
```

```
plt.xlabel('log2_vegetable_production')
plt.ylabel('log2_emissions_CO2 for On-farm energy use')
plt.legend(title='Area', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.yticks(np.arange(4, 16, 1))
plt.grid(True)

plt.tight_layout()
plt.show()
```



Question 4.2 (12 points)

1. Note the question your visualisation addresses.
2. Explain your reasons behind your visualisation design choice and summarise its pros and cons. Please make sure that your answer is no longer than 5 lines.
3. Write your interpretation of the visualisation. Please make sure that your answer is no longer than 20 lines (plus to 5 lines for your references in the case that you referenced to external resources), i.e. stays in the box in the PDF file you have produced from this notebook. You can delete the text saying "Your answer goes here". reference visualisation principle exactly.

Your answer to Q4.2 goes here

The advantage of a scatter plot is that it can include as much data as possible while effectively displaying the distribution and trends of the data. The disadvantage is that when the data volume is very large, the points can easily overlap. In the original data, the differences between the data from different countries are very significant, causing countries with smaller values to be compressed into one corner. Therefore, I decided to take the logarithm base 2 of both the x-axis and y-axis. This way, the image can expand and is less crowded than the original image. The log-transformed image allows for better trend analysis and observation of the data points.

Observation: In the image, CO2 emissions from on-farm energy in the UK are higher than in some countries with greater vegetable yields. This may be due to the UK's advanced agricultural mechanization as a developed country, where large machinery contributes significantly to carbon emissions.

Russia, the country with the highest emissions, not only has the largest vegetable yield but also the highest rural carbon emissions. This is likely because of its rapid industrialization and mechanization, combined with its vast land area, which places it in the upper right corner of the chart.

Most countries' data are relatively concentrated, and some countries have high vegetable yields but low carbon emissions. This may be because they do not use large machinery that generates high carbon emissions.

For developed countries like the UK and Russia, carbon emissions tend to be proportional to vegetable yields. However, in countries with less mechanization, high vegetable yields may not correlate with high carbon emissions.

End of assignment playground - we will mark nothing beyond this point!

[ ]:

[ ]: