Analysing European Crop Production Trends with FAOSTAT Dataset

Brief Summary

This notebook explores agricultural trends in Europe by analyzing FAOSTAT data spanning 1961–2023 for 144 major crops. Through an iterative "Pytudes"-style workflow, we will:

1. **Load & Clean** the raw CSV data.
2. **Transform** it into time-series and normalized metrics (e.g., yield per hectare).
3. **Visualize** aggregate and crop-specific trends, highlighting key shifts and anomalies.
4. **Compare** subregions (Western, Eastern, Northern, Southern Europe) to uncover differential growth patterns.
5. **Experiment** with advanced chart types (heatmaps, small multiples, animated race charts) to deepen insights.

Our goal is to demonstrate a reproducible, critique-driven data-visualization process that reveals how European crop production, yields, and harvested areas have evolved over six decades.

Here lie all the imports needed across this exploration

In [88]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import textwrap
import squarify
import matplotlib.ticker as ticker
from matplotlib.colors import LogNorm
```

So as we needed to download the data in 3 separate files (due to site limitations) we want to have all the variable in one place for more easier use, so we are gonna load the 3 datasets, merge them and then clean some of the null values.

In [62]:
```
# 1. Load each CSV separately
df_area = pd.read_csv('FAOSTAT_data_en_5-31-2025-harvest-area.csv')
df_prod = pd.read_csv('FAOSTAT_data_en_5-31-2025-production.csv')
```

```python
df_yld  = pd.read_csv('FAOSTAT_data_en_5-31-2025-yield.csv')

# 2. Inspect columns to identify the common keys (including 'Unit')
print("Area-harvest columns:", df_area.columns.tolist())
print("Production   columns:", df_prod.columns.tolist())
print("Yield        columns:", df_yld.columns.tolist())

# 3. Reduce each DataFrame to the key columns + its own 'Value' and 'Unit', renaming as needed.
#    Keep: ['Area', 'Item', 'Year', 'Value', 'Unit'] → rename Value and Unit
df_area_trim = (
    df_area
    .loc[:, ['Area', 'Item', 'Year', 'Value', 'Unit']]
    .rename(columns={
        'Value': 'Area_harvested',
        'Unit':  'Unit_Area_harvested'
    })
)

df_prod_trim = (
    df_prod
    .loc[:, ['Area', 'Item', 'Year', 'Value', 'Unit']]
    .rename(columns={
        'Value': 'Production',
        'Unit':  'Unit_Production'
    })
)

df_yld_trim = (
    df_yld
    .loc[:, ['Area', 'Item', 'Year', 'Value', 'Unit']]
    .rename(columns={
        'Value': 'Yield',
        'Unit':  'Unit_Yield'
    })
)

# 4. Merge the three DataFrames on the common keys: ['Area','Item','Year']
#    Use outer merge so no data is lost if one file is missing a combination.
df_merge1 = pd.merge(
    df_area_trim,
    df_prod_trim,
    on=['Area', 'Item', 'Year'],
```

```
    how='outer'
)

df_merged = pd.merge(
    df_merge1,
    df_yld_trim,
    on=['Area', 'Item', 'Year'],
    how='outer'
)

# 5. Check the result structure
print("Merged shape:", df_merged.shape)
print(df_merged.sample(10))

# 6. (Optional) Drop rows where all three metrics are NaN
df_merged = df_merged.dropna(subset=['Area_harvested', 'Production', 'Yield'], how='all')

# 7. Inspect how many non-null entries exist, and confirm units
print("\nNon-null counts after merging:")
print(df_merged[['Area_harvested', 'Production', 'Yield']].notna().sum())

print("\nUnique units for each metric:")
print(" Area harvested units:", df_merged['Unit_Area_harvested'].unique())
print(" Production units:    ", df_merged['Unit_Production'].unique())
print(" Yield units:         ", df_merged['Unit_Yield'].unique())
```

```
Area-harvest columns: ['Domain', 'Area', 'Element', 'Item', 'Year', 'Unit', 'Value']
Production   columns: ['Domain', 'Area', 'Element', 'Item', 'Year', 'Unit', 'Value']
Yield        columns: ['Domain', 'Area', 'Element', 'Item', 'Year', 'Unit', 'Value']
Merged shape: (150724, 9)
                          Area  \
32573                  Denmark
37070                  Finland
123003                   Spain
112817                 Romania
98114                   Poland
99432                 Portugal
16183   Bosnia and Herzegovina
112425                 Romania
24397                  Croatia
2636                   Albania

                                                    Item  Year  \
32573                                            Linseed  1998
37070                                 Carrots and turnips  1999
123003  Anise, badian, coriander, cumin, caraway, fenn...  1992
112817                                        Watermelons  2006
98114                                        Strawberries  1998
99432                     Broad beans and horse beans, dry  1979
16183                                     Lemons and limes  2013
112425                           Sunflower-seed oil, crude  1997
24397                               Eggplants (aubergines)  2016
2636                                 Other fruits, n.e.c.  2023

        Area_harvested Unit_Area_harvested  Production Unit_Production  \
32573           3871.0                  ha      1645.0               t
37070           1726.0                  ha     62309.0               t
123003          2242.0                  ha      1055.0               t
112817         30689.0                  ha    587756.0               t
98114          52614.0                  ha    149858.0               t
99432          37700.0                  ha     21300.0               t
16183              1.0                  ha        12.0               t
112425             NaN                 NaN    375000.0               t
24397              NaN                  ha         NaN               t
2636             150.0                  ha      3840.8               t

          Yield Unit_Yield
32573     425.0      kg/ha
```

```
37070   36100.2      kg/ha
123003    470.6      kg/ha
112817  19152.0      kg/ha
98114    2848.3      kg/ha
99432     565.0      kg/ha
16183   12000.0      kg/ha
112425      NaN        NaN
24397       NaN        NaN
2636    25656.6      kg/ha

Non-null counts after merging:
Area_harvested    103634
Production        134279
Yield              97177
dtype: int64

Unique units for each metric:
 Area harvested units: ['ha' nan]
 Production units:    ['t' nan]
 Yield units:         [nan 'kg/ha']
```

In [63]:
```
df_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 134538 entries, 0 to 150723
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   Area                 134538 non-null  object 
 1   Item                 134538 non-null  object 
 2   Year                 134538 non-null  int64  
 3   Area_harvested       103634 non-null  float64
 4   Unit_Area_harvested  108660 non-null  object 
 5   Production           134279 non-null  float64
 6   Unit_Production      134346 non-null  object 
 7   Yield                97177 non-null   float64
 8   Unit_Yield           97177 non-null   object 
dtypes: float64(3), int64(1), object(5)
memory usage: 10.3+ MB
```

As this dataset has a big number of entries, we would like to see exactly how many types of items we have, basically seeing which crops or products we cover in this analysis.

In [64]:
```
unique_items = df_merged['Item'].unique()
print(len(unique_items))
for item in sorted(unique_items):
    print(item)
```

144
Almonds, in shell
Anise, badian, coriander, cumin, caraway, fennel and juniper berries, raw
Apples
Apricots
Artichokes
Asparagus
Avocados
Bananas
Barley
Beans, dry
Beer of barley, malted
Blueberries
Broad beans and horse beans, dry
Broad beans and horse beans, green
Buckwheat
Cabbages
Canary seed
Cantaloupes and other melons
Carrots and turnips
Castor oil seeds
Cauliflowers and broccoli
Cereals n.e.c.
Cherries
Chestnuts, in shell
Chick peas, dry
Chicory roots
Chillies and peppers, dry (Capsicum spp., Pimenta spp.), raw
Chillies and peppers, green (Capsicum spp. and Pimenta spp.)
Coconut oil
Coconuts, in shell
Coffee, green
Cotton lint, ginned
Cotton seed
Cottonseed oil
Cow peas, dry
Cranberries
Cucumbers and gherkins
Currants
Dates
Edible roots and tubers with high starch or inulin content, n.e.c., fresh
Eggplants (aubergines)

Figs
Flax, raw or retted
Gooseberries
Grapes
Green corn (maize)
Green garlic
Green tea (not fermented), black tea (fermented) and partly fermented tea, in immediate packings of a content not exceeding 3 kg
Groundnut oil
Groundnuts, excluding shelled
Hazelnuts, in shell
Hempseed
Hop cones
Kenaf, and other textile bast fibres, raw or retted
Kiwi fruit
Leeks and other alliaceous vegetables
Lemons and limes
Lentils, dry
Lettuce and chicory
Linseed
Locust beans (carobs)
Lupins
Maize (corn)
Margarine and shortening
Melonseed
Millet
Mixed grain
Molasses
Mushrooms and truffles
Mustard seed
Oats
Oil of linseed
Oil of maize
Oil of palm kernel
Oil of sesame seed
Okra
Olive oil
Olives
Onions and shallots, dry (excluding dehydrated)
Onions and shallots, green
Oranges
Other beans, green
Other berries and fruits of the genus vaccinium n.e.c.

Other citrus fruit, n.e.c.
Other fruits, n.e.c.
Other nuts (excluding wild edible nuts and groundnuts), in shell, n.e.c.
Other oil seeds, n.e.c.
Other pome fruits
Other pulses n.e.c.
Other stimulant, spice and aromatic crops, n.e.c.
Other stone fruits
Other tropical fruits, n.e.c.
Other vegetables, fresh n.e.c.
Palm oil
Peaches and nectarines
Pears
Peas, dry
Peas, green
Peppermint, spearmint
Persimmons
Pineapples
Pistachios, in shell
Plums and sloes
Pomelos and grapefruits
Poppy seed
Potatoes
Pumpkins, squash and gourds
Pyrethrum, dried flowers
Quinces
Rape or colza seed
Rapeseed or canola oil, crude
Raspberries
Raw cane or beet sugar (centrifugal only)
Rice
Rye
Safflower seed
Safflower-seed oil, crude
Seed cotton, unginned
Sesame seed
Sorghum
Sour cherries
Soya bean oil
Soya beans
Spinach
Strawberries

```
String beans
Sugar beet
Sugar cane
Sunflower seed
Sunflower-seed oil, crude
Sweet potatoes
Tangerines, mandarins, clementines
Tea leaves
Tomatoes
Triticale
True hemp, raw or retted
Tung nuts
Unmanufactured tobacco
Vetches
Walnuts, in shell
Watermelons
Wheat
Wine
Yams
```

Now having this list we can see that we have 144 individual items, plotting 144 crops on any of the proprieties we have production area harvested or yield will be confusing and hard to read. Now to use some generative AI magic we create a map of all the crops we have to a more general list of categories

In [65]:
```
crop_to_group = {
    'Almonds, in shell':                                   'Nuts',
    'Anise, badian, coriander, cumin, caraway, fennel and juniper berries, raw':
                                                            'Spices & Aromatics',
    'Apples':                                               'Fruits',
    'Apricots':                                             'Fruits',
    'Artichokes':                                           'Vegetables',
    'Asparagus':                                            'Vegetables',
    'Avocados':                                             'Fruits',
    'Bananas':                                              'Fruits',
    'Barley':                                               'Cereals',
    'Beans, dry':                                           'Pulses/Legumes',
    'Beer of barley, malted':                               'Beverages',
    'Blueberries':                                          'Fruits',
    'Broad beans and horse beans, dry':                     'Pulses/Legumes',
    'Broad beans and horse beans, green':                   'Vegetables',
    'Buckwheat':                                            'Cereals',
```

```
    'Cabbages':                                          'Vegetables',
    'Canary seed':                                       'Cereals',
    'Cantaloupes and other melons':                      'Fruits',
    'Carrots and turnips':                               'Roots & Tubers',
    'Castor oil seeds':                                  'Oilseeds',
    'Cauliflowers and broccoli':                         'Vegetables',
    'Cereals n.e.c.':                                    'Cereals',
    'Cherries':                                          'Fruits',
    'Chestnuts, in shell':                               'Nuts',
    'Chick peas, dry':                                   'Pulses/Legumes',
    'Chicory roots':                                     'Vegetables',
    'Chillies and peppers, dry (Capsicum spp., Pimenta spp.), raw':
                                                          'Spices & Aromatics',
    'Chillies and peppers, green (Capsicum spp. and Pimenta spp.)':
                                                          'Vegetables',
    'Coconut oil':                                       'Oils & Fats',
    'Coconuts, in shell':                                'Fruits',
    'Coffee, green':                                     'Beverages',
    'Cotton lint, ginned':                               'Fiber Crops',
    'Cotton seed':                                       'Oilseeds',
    'Cottonseed oil':                                    'Oils & Fats',
    'Cow peas, dry':                                     'Pulses/Legumes',
    'Cranberries':                                       'Fruits',
    'Cucumbers and gherkins':                            'Vegetables',
    'Currants':                                          'Fruits',
    'Dates':                                             'Fruits',
    'Edible roots and tubers with high starch or inulin content, n.e.c., fresh':
                                                          'Roots & Tubers',
    'Eggplants (aubergines)':                            'Vegetables',
    'Figs':                                              'Fruits',
    'Flax, raw or retted':                               'Fiber Crops',
    'Gooseberries':                                      'Fruits',
    'Grapes':                                            'Fruits',
    'Green corn (maize)':                                'Cereals',
    'Green garlic':                                      'Vegetables',
    'Green tea (not fermented), black tea (fermented) and partly fermented tea, in immediate packings of a content not exceeding 3
                                                          'Beverages',
    'Groundnut oil':                                     'Oils & Fats',
    'Groundnuts, excluding shelled':                     'Oilseeds',
    'Hazelnuts, in shell':                               'Nuts',
    'Hempseed':                                          'Oilseeds',
    'Hop cones':                                         'Beverages',
```

```
    'Kenaf, and other textile bast fibres, raw or retted':
                                                     'Fiber Crops',
    'Kiwi fruit':                                   'Fruits',
    'Leeks and other alliaceous vegetables':        'Vegetables',
    'Lemons and limes':                             'Fruits',
    'Lentils, dry':                                 'Pulses/Legumes',
    'Lettuce and chicory':                          'Vegetables',
    'Linseed':                                      'Oilseeds',
    'Locust beans (carobs)':                        'Vegetables',
    'Lupins':                                       'Pulses/Legumes',
    'Maize (corn)':                                 'Cereals',
    'Margarine and shortening':                     'Oils & Fats',
    'Melonseed':                                    'Oilseeds',
    'Millet':                                       'Cereals',
    'Mixed grain':                                  'Cereals',
    'Molasses':                                     'Sugar Crops',
    'Mushrooms and truffles':                       'Vegetables',
    'Mustard seed':                                 'Oilseeds',
    'Oats':                                         'Cereals',
    'Oil of linseed':                               'Oils & Fats',
    'Oil of maize':                                 'Oils & Fats',
    'Oil of palm kernel':                           'Oils & Fats',
    'Oil of sesame seed':                           'Oils & Fats',
    'Okra':                                         'Vegetables',
    'Olive oil':                                    'Oils & Fats',
    'Olives':                                       'Fruits',
    'Onions and shallots, dry (excluding dehydrated)':  'Vegetables',
    'Onions and shallots, green':                   'Vegetables',
    'Oranges':                                      'Fruits',
    'Other beans, green':                           'Vegetables',
    'Other berries and fruits of the genus vaccinium n.e.c.':
                                                     'Fruits',
    'Other citrus fruit, n.e.c.':                   'Fruits',
    'Other fruits, n.e.c.':                         'Fruits',
    'Other nuts (excluding wild edible nuts and groundnuts), in shell, n.e.c.':
                                                     'Nuts',
    'Other oil seeds, n.e.c.':                      'Oilseeds',
    'Other pome fruits':                            'Fruits',
    'Other pulses n.e.c.':                          'Pulses/Legumes',
    'Other stimulant, spice and aromatic crops, n.e.c.':
                                                     'Spices & Aromatics',
    'Other stone fruits':                           'Fruits',
```

```
    'Other tropical fruits, n.e.c.':                       'Fruits',
    'Other vegetables, fresh n.e.c.':                     'Vegetables',
    'Palm oil':                                           'Oils & Fats',
    'Peaches and nectarines':                             'Fruits',
    'Pears':                                              'Fruits',
    'Peas, dry':                                          'Pulses/Legumes',
    'Peas, green':                                        'Vegetables',
    'Peppermint, spearmint':                              'Spices & Aromatics',
    'Persimmons':                                         'Fruits',
    'Pineapples':                                         'Fruits',
    'Pistachios, in shell':                               'Nuts',
    'Plums and sloes':                                    'Fruits',
    'Pomelos and grapefruits':                            'Fruits',
    'Poppy seed':                                         'Oilseeds',
    'Potatoes':                                           'Roots & Tubers',
    'Pumpkins, squash and gourds':                        'Vegetables',
    'Pyrethrum, dried flowers':                           'Industrial Crops',
    'Quinces':                                            'Fruits',
    'Rape or colza seed':                                 'Oilseeds',
    'Rapeseed or canola oil, crude':                      'Oils & Fats',
    'Raspberries':                                        'Fruits',
    'Raw cane or beet sugar (centrifugal only)':           'Sugar Crops',
    'Rice':                                               'Cereals',
    'Rye':                                                'Cereals',
    'Safflower seed':                                     'Oilseeds',
    'Safflower-seed oil, crude':                          'Oils & Fats',
    'Seed cotton, unginned':                              'Fiber Crops',
    'Sesame seed':                                        'Oilseeds',
    'Sorghum':                                            'Cereals',
    'Sour cherries':                                      'Fruits',
    'Soya bean oil':                                      'Oils & Fats',
    'Soya beans':                                         'Pulses/Legumes',
    'Spinach':                                            'Vegetables',
    'Strawberries':                                       'Fruits',
    'String beans':                                       'Vegetables',
    'Sugar beet':                                         'Sugar Crops',
    'Sugar cane':                                         'Sugar Crops',
    'Sunflower seed':                                     'Oilseeds',
    'Sunflower-seed oil, crude':                          'Oils & Fats',
    'Sweet potatoes':                                     'Roots & Tubers',
    'Tangerines, mandarins, clementines':                 'Fruits',
    'Tea leaves':                                         'Beverages',
```

```
    'Tomatoes':                                        'Vegetables',
    'Triticale':                                       'Cereals',
    'True hemp, raw or retted':                        'Fiber Crops',
    'Tung nuts':                                       'Nuts',
    'Unmanufactured tobacco':                          'Stimulant Crops',
    'Vetches':                                         'Pulses/Legumes',
    'Walnuts, in shell':                               'Nuts',
    'Watermelons':                                     'Fruits',
    'Wheat':                                           'Cereals',
    'Wine':                                            'Beverages',
    'Yams':                                            'Roots & Tubers'
}

#To verify that every one of your 144 items is covered:
assert len(crop_to_group) == 144
```

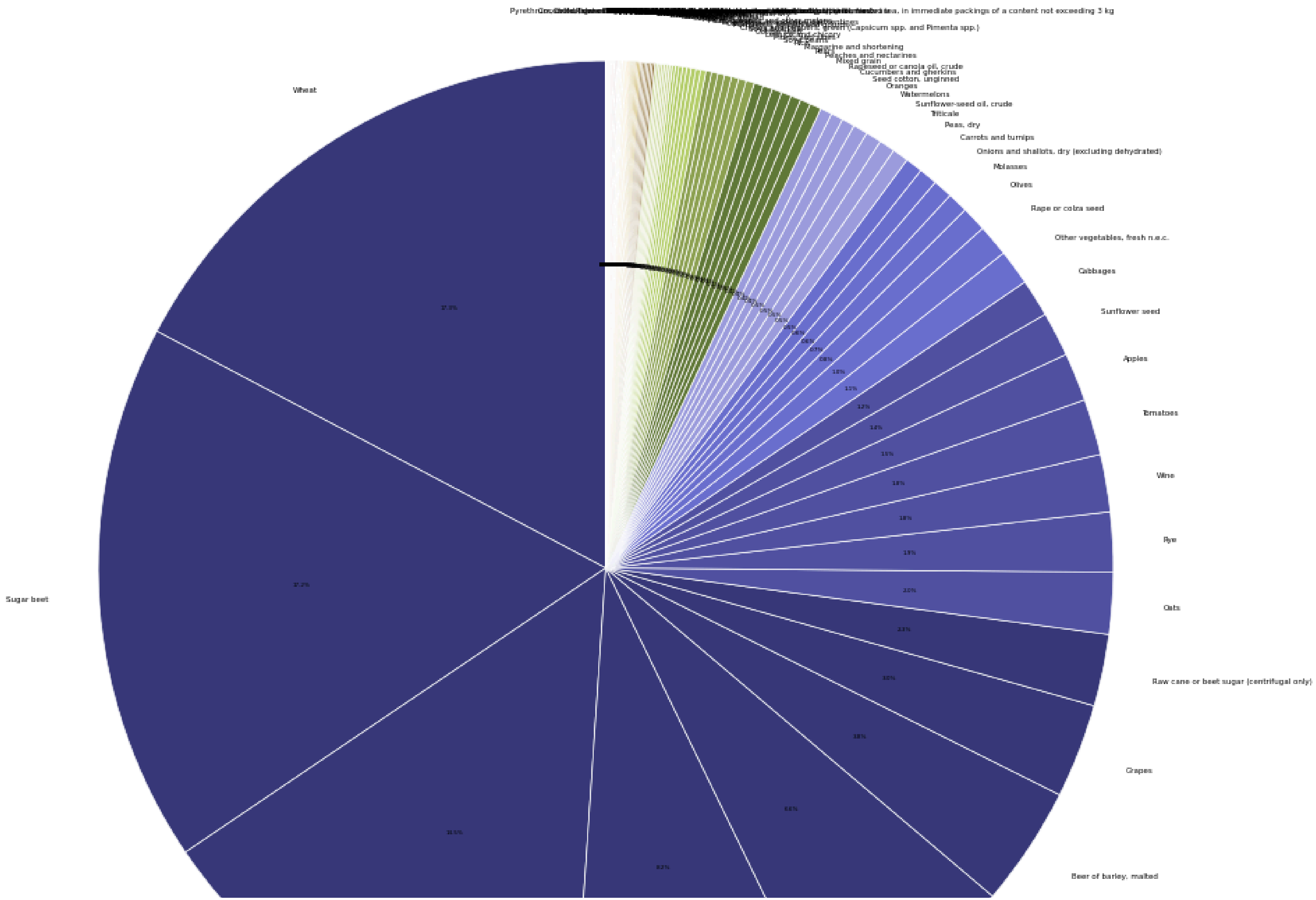
Now lets try a simple visualization by analysing the production of the top 20 crops in our dataset

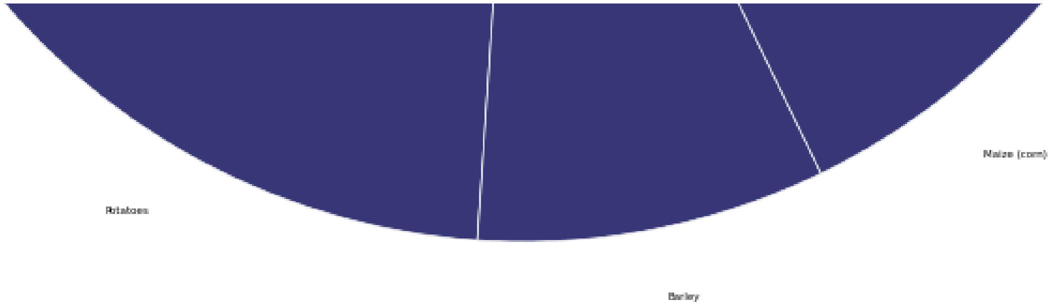
In [71]:
```
# Because we have multiple years and countries and we are currently looking at total production of the crops we need to agreggate t
production_per_item = (
    df_merged
    .groupby('Item')['Production']
    .sum()
    .sort_values(ascending=False)
)

# 2. To compute total production by Group without altering df_merged, create a temporary DataFrame:
df_temp = df_merged.copy()
df_temp['Group'] = df_temp['Item'].map(crop_to_group)

production_by_group = (
    df_temp
    .groupby('Group')['Production']
    .sum()
    .sort_values(ascending=False)
)

# ──────────────────────────────────────────────────────────────────────────────
# (A) Pie chart of all 144 crop items
# ──────────────────────────────────────────────────────────────────────────────
fig1, ax1 = plt.subplots(figsize=(10, 10))
```

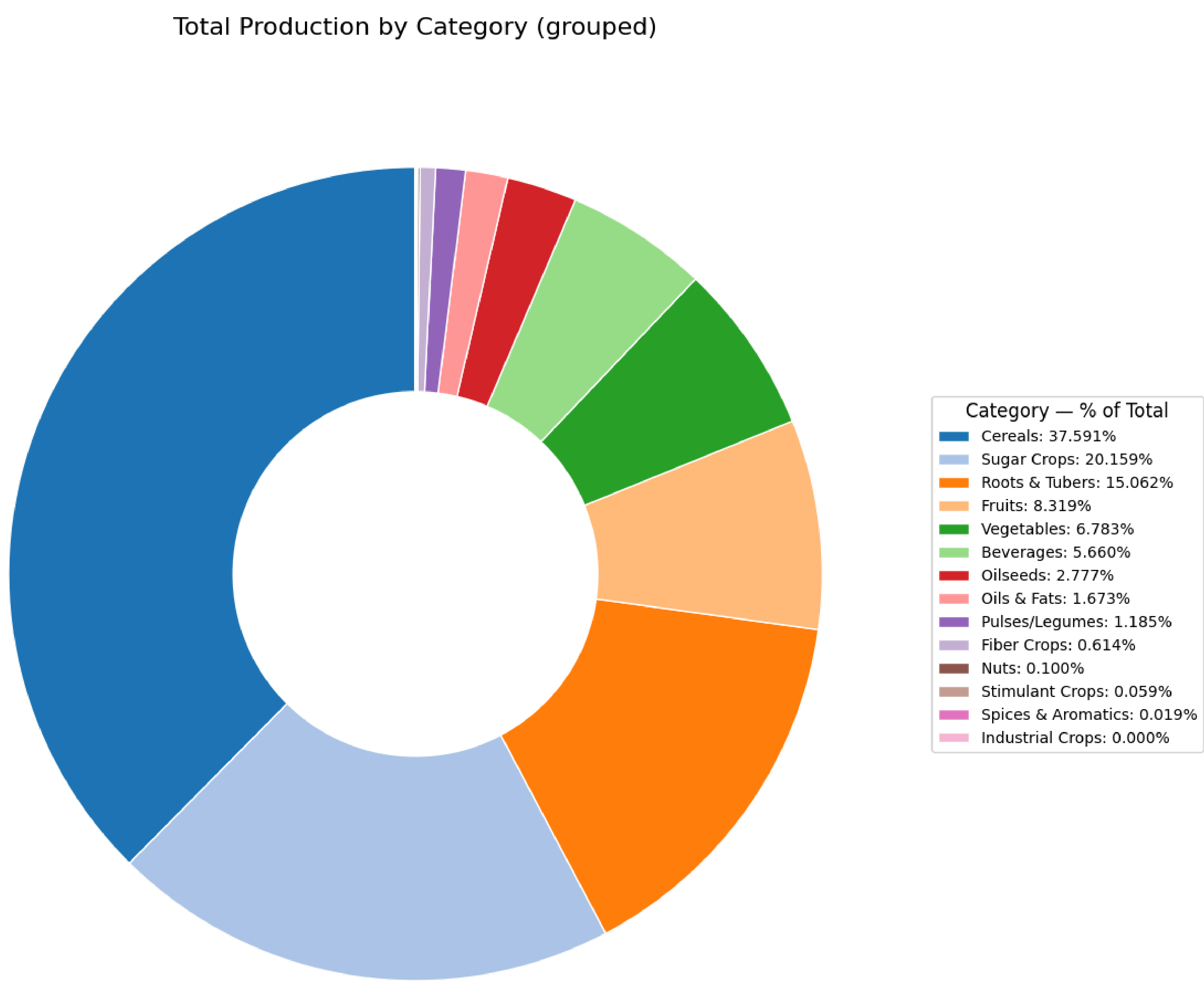
```
# Generate 144 distinct colors from a colormap
colors_items = plt.cm.tab20b(np.linspace(0, 1, len(production_per_item)))

wedges1, texts1, autotexts1 = ax1.pie(
    production_per_item.values,
    labels=production_per_item.index,
    autopct='%1.1f%%',
    colors=colors_items,
    startangle=90,
    wedgeprops=dict(edgecolor='white', linewidth=0.5)
)

# Make labels very small because there are 144 slices
for txt in texts1:
    txt.set_fontsize(4)
for atxt in autotexts1:
    atxt.set_fontsize(3)

ax1.set_title('Total Production by Crop Item (all 144)', fontsize=14, pad=20)
plt.tight_layout()


# ──────────────────────────────────────────────────────────────────────────────
# (B) Pie chart of grouped categories, with legend (no on-slice labels)
# ──────────────────────────────────────────────────────────────────────────────
fig2, ax2 = plt.subplots(figsize=(12, 12))

groups = production_by_group.index
values = production_by_group.values

# Compute percentages for each group
total = values.sum()
percentages = values / total * 100
legend_labels = [f"{grp}: {p:.3f}%" for grp, p in zip(groups, percentages)]

# Use a discrete colormap with at least len(groups) distinct colors
cmap = plt.get_cmap('tab20')
colors_groups = cmap.colors[: len(groups)]

# Create the pie WITHOUT on-slice labels or percentages
wedges2, _ = ax2.pie(
```

```
    values,
    labels=None,         # No labels on the slices
    startangle=90,
    colors=colors_groups,
    wedgeprops=dict(edgecolor='white', linewidth=1)
)

# Add a white circle for donut style (optional)
centre_circle = plt.Circle((0, 0), 0.45, color='white', linewidth=0)
ax2.add_artist(centre_circle)

# Title
ax2.set_title('Total Production by Category (grouped)', fontsize=16, pad=20)

# Add legend on the right with category names + percentages
ax2.legend(
    wedges2,
    legend_labels,
    title="Category – % of Total",
    loc="center left",
    bbox_to_anchor=(1, 0, 0.3, 1),   # Positions legend to the right of the pie
    fontsize=10,
    title_fontsize=12
)

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

Total Production by Crop Item (all 144)




Maize (corn)
Potatoes
Barley


Total Production by Category (grouped)
Category — % of Total
Cereals: 37.591%
Sugar Crops: 20.159%
Roots & Tubers: 15.062%
Fruits: 8.319%
Vegetables: 6.783%
Beverages: 5.660%
Oilseeds: 2.777%
Oils & Fats: 1.673%
Pulses/Legumes: 1.185%
Fiber Crops: 0.614%
Nuts: 0.100%
Stimulant Crops: 0.059%
Spices & Aromatics: 0.019%
Industrial Crops: 0.000%

As we can clearly see in the second visualization our pie chart already starts to look beter, while even the grouped categories seem a bit cluttered at the lower end where the precentages are relatively small to our leader it can offer a better picture in how the dataset is spread across the types of crops that we have.

no lets do the same for area harvested and yield

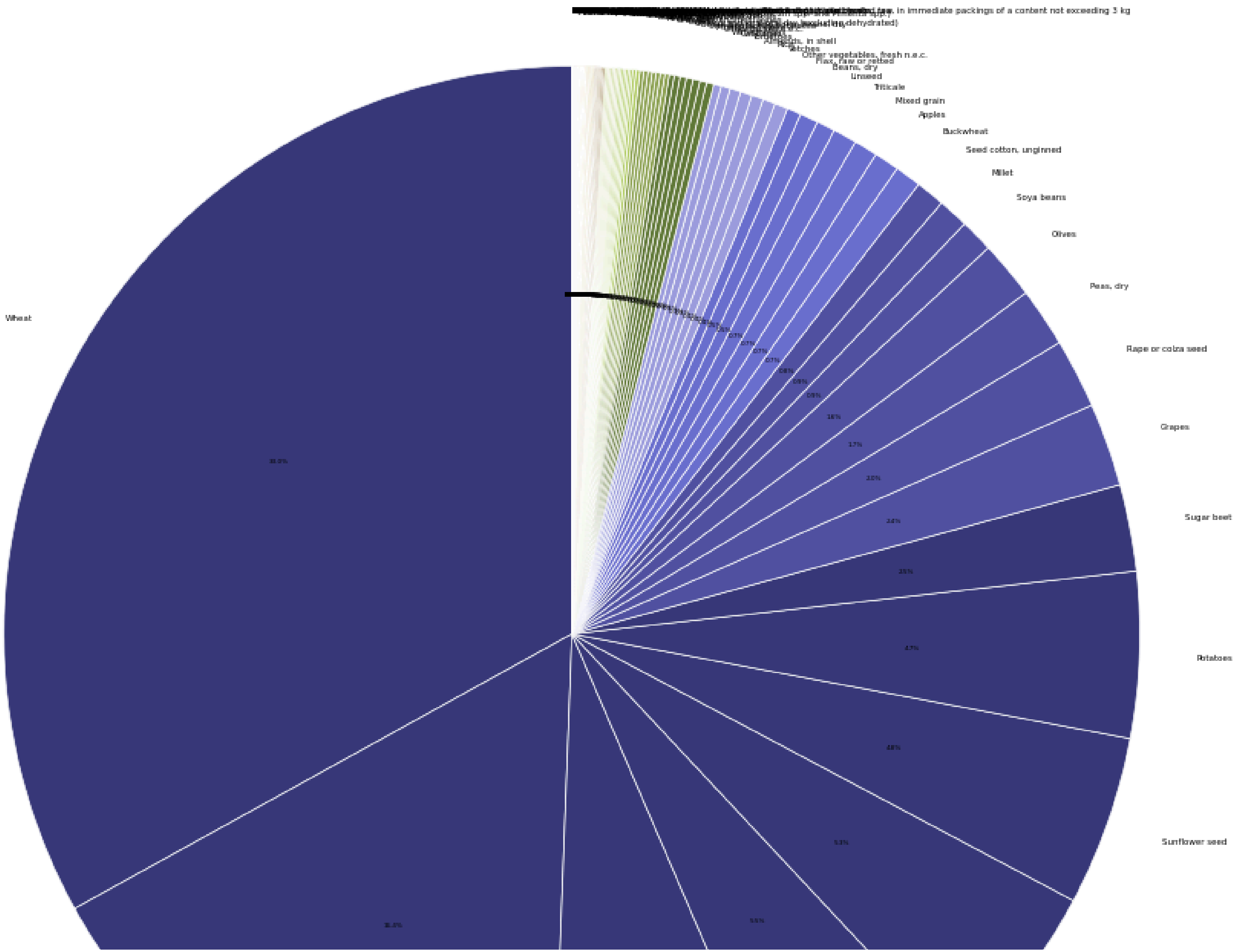
In [72]:
```
# ------------------------------------------------------------------------------
# (B) AREA_HARVESTED PIES (exactly as before)
# ------------------------------------------------------------------------------

# 1B. Aggregate total Area_Harvested by Item
area_per_item = (
    df_merged
    .groupby('Item')['Area_harvested']
    .sum()
    .sort_values(ascending=False)
)

# 2B. Aggregate total Area_Harvested by Group (using a temporary copy)
df_temp_area = df_merged.copy()
df_temp_area['Group'] = df_temp_area['Item'].map(crop_to_group)
area_by_group = (
    df_temp_area
    .groupby('Group')['Area_harvested']
    .sum()
    .sort_values(ascending=False)
)

# 3B. Plot: (B1) Pie of all 144 Items by Area_Harvested
figB1, axB1 = plt.subplots(figsize=(10, 10))
colors_items_B = plt.cm.tab20b(np.linspace(0, 1, len(area_per_item)))
wedgesB1, textsB1, autotextsB1 = axB1.pie(
    area_per_item.values,
    labels=area_per_item.index,
    autopct='%1.1f%%',
    colors=colors_items_B,
```

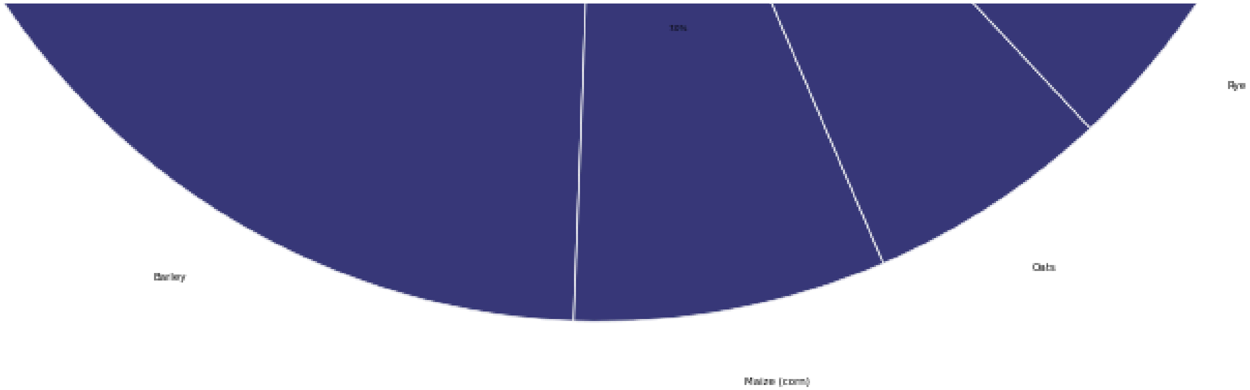
```
    startangle=90,
    wedgeprops=dict(edgecolor='white', linewidth=0.5)
)
for txt in textsB1:
    txt.set_fontsize(4)
for atxt in autotextsB1:
    atxt.set_fontsize(3)
axB1.set_title('Total Area Harvested by Crop Item (all 144)', fontsize=14, pad=20)
plt.tight_layout()

# 4B. Plot: (B2) Pie of Groups by Area_Harvested (legend on right)
figB2, axB2 = plt.subplots(figsize=(12, 8))
groups_B = area_by_group.index
values_B = area_by_group.values
total_B = values_B.sum()
pct_B = values_B / total_B * 100
legend_labels_B = [f"{grp}: {p:.3f}%" for grp, p in zip(groups_B, pct_B)]
cmap_B = plt.get_cmap('tab20')
colors_groups_B = cmap_B.colors[: len(groups_B)]
wedgesB2, _ = axB2.pie(
    values_B,
    labels=None,
    startangle=90,
    colors=colors_groups_B,
    wedgeprops=dict(edgecolor='white', linewidth=1)
)
centreB2 = plt.Circle((0, 0), 0.45, color='white', linewidth=0)
axB2.add_artist(centreB2)
axB2.set_title('Total Area Harvested by Category (grouped)', fontsize=16, pad=20)
axB2.legend(
    wedgesB2,
    legend_labels_B,
    title="Category – % of Total Area",
    loc="center left",
    bbox_to_anchor=(1, 0, 0.3, 1),
    fontsize=10,
    title_fontsize=12
)
plt.tight_layout()


# ------------------------------------------------------------------------------
```

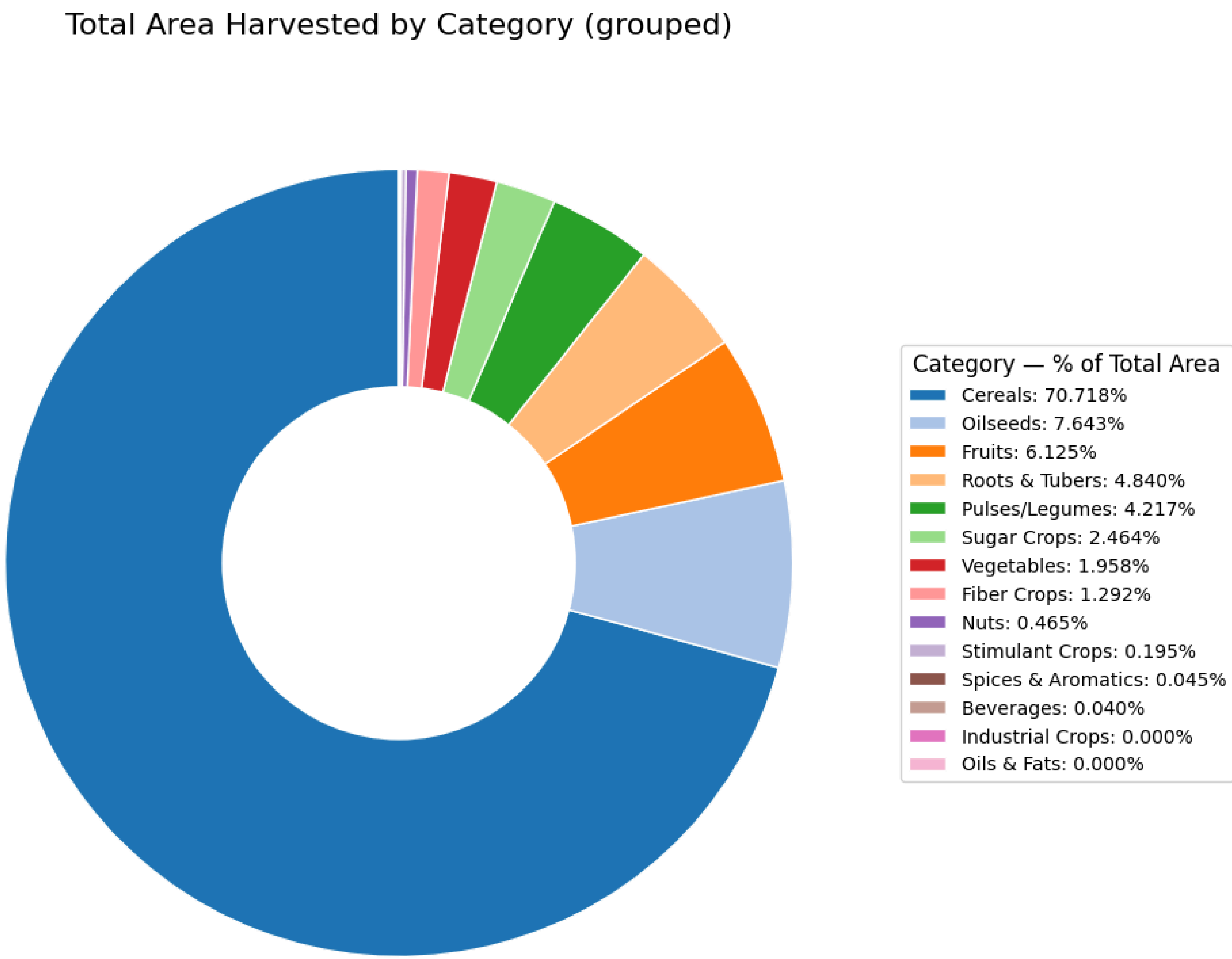
```python
# (C) YIELD PIES (using the same naming as AREA_HARVESTED)
# -------------------------------------------------------------------------------

# 1C. Aggregate total Yield by Item
yield_per_item = (
    df_merged
    .groupby('Item')['Yield']
    .sum()
    .sort_values(ascending=False)
)

# 2C. Aggregate total Yield by Group (using a temporary copy)
df_temp_yield = df_merged.copy()
df_temp_yield['Group'] = df_temp_yield['Item'].map(crop_to_group)
yield_by_group = (
    df_temp_yield
    .groupby('Group')['Yield']
    .sum()
    .sort_values(ascending=False)
)

# 3C. Plot: (C1) Pie of all 144 Items by Yield
figC1, axC1 = plt.subplots(figsize=(10, 10))
colors_items_C = plt.cm.tab20b(np.linspace(0, 1, len(yield_per_item)))
wedgesC1, textsC1, autotextsC1 = axC1.pie(
    yield_per_item.values,
    labels=yield_per_item.index,
    autopct='%1.1f%%',
    colors=colors_items_C,
    startangle=90,
    wedgeprops=dict(edgecolor='white', linewidth=0.5)
)
for txt in textsC1:
    txt.set_fontsize(4)
for atxt in autotextsC1:
    atxt.set_fontsize(3)
axC1.set_title('Total Yield by Crop Item (all 144)', fontsize=14, pad=20)
plt.tight_layout()

# 4C. Plot: (C2) Pie of Groups by Yield (legend on right)
figC2, axC2 = plt.subplots(figsize=(12, 8))
groups_C = yield_by_group.index
```

```
values_C = yield_by_group.values
total_C = values_C.sum()
pct_C = values_C / total_C * 100
legend_labels_C = [f"{grp}: {p:.3f}%" for grp, p in zip(groups_C, pct_C)]
cmap_C = plt.get_cmap('tab20')
colors_groups_C = cmap_C.colors[: len(groups_C)]
wedgesC2, _ = axC2.pie(
    values_C,
    labels=None,
    startangle=90,
    colors=colors_groups_C,
    wedgeprops=dict(edgecolor='white', linewidth=1)
)
centreC2 = plt.Circle((0, 0), 0.45, color='white', linewidth=0)
axC2.add_artist(centreC2)
axC2.set_title('Total Yield by Category (grouped)', fontsize=16, pad=20)
axC2.legend(
    wedgesC2,
    legend_labels_C,
    title="Category – % of Total Yield",
    loc="center left",
    bbox_to_anchor=(1, 0, 0.3, 1),
    fontsize=10,
    title_fontsize=12
)
plt.tight_layout()

# ──────────────────────────────────────────────────────────────────────────────
# DISPLAY ALL FIGURES
# ──────────────────────────────────────────────────────────────────────────────
plt.show()
```

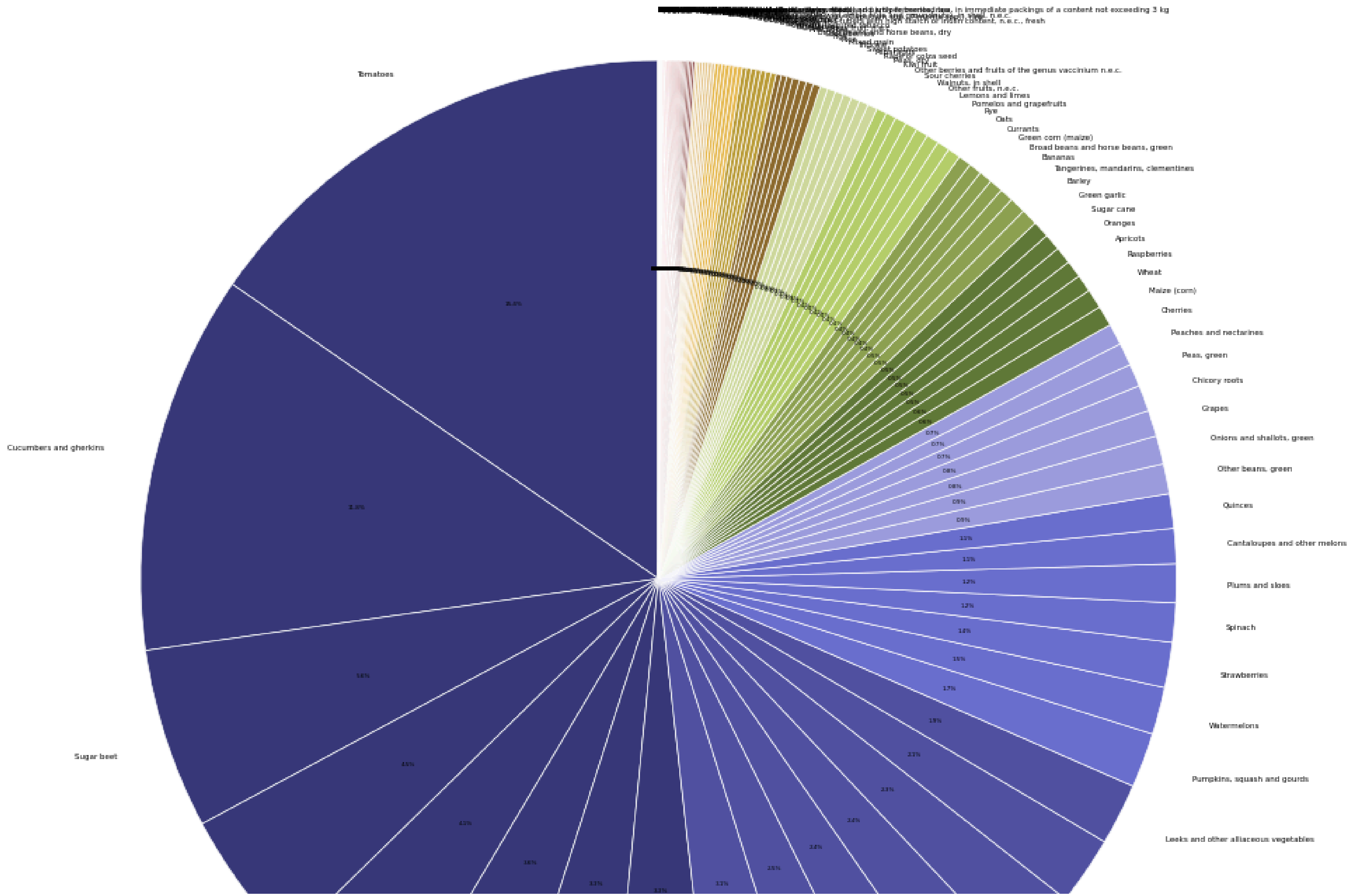
Total Area Harvested by Crop Item (all 144)

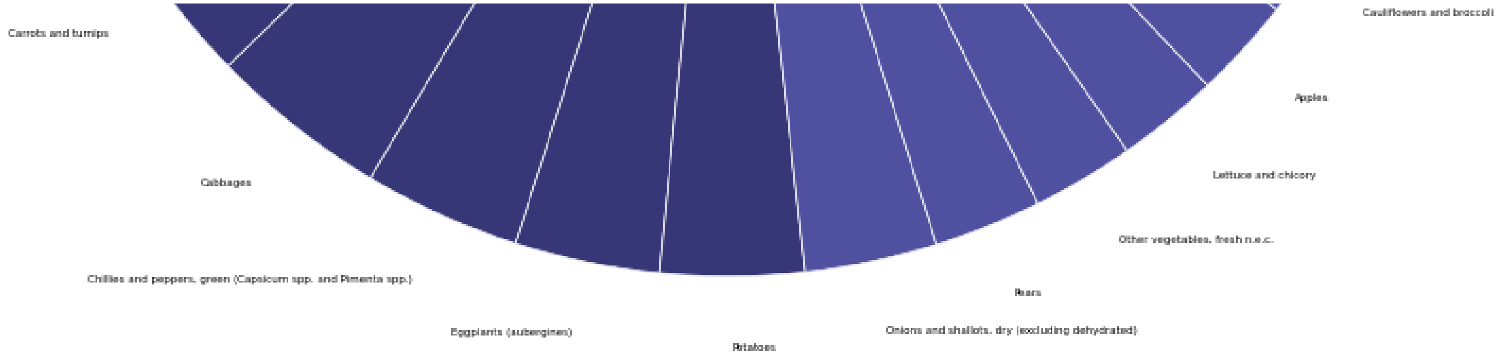


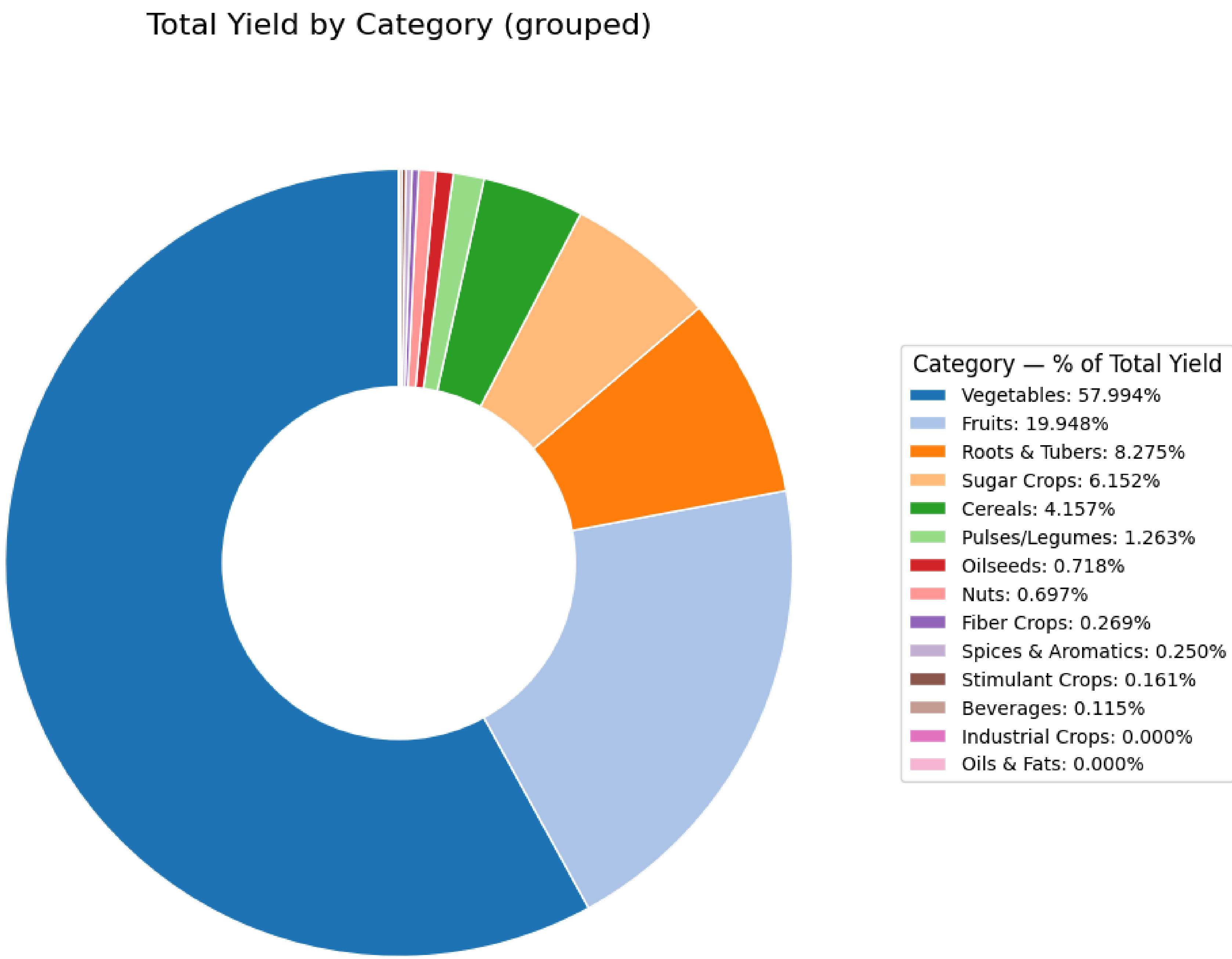

Rye
Oats
Barley
Maize (corn)


Total Area Harvested by Category (grouped)
Category — % of Total Area
Cereals: 70.718%
Oilseeds: 7.643%
Fruits: 6.125%
Roots & Tubers: 4.840%
Pulses/Legumes: 4.217%
Sugar Crops: 2.464%
Vegetables: 1.958%
Fiber Crops: 1.292%
Nuts: 0.465%
Stimulant Crops: 0.195%
Spices & Aromatics: 0.045%
Beverages: 0.040%
Industrial Crops: 0.000%
Oils & Fats: 0.000%

Total Yield by Crop Item (all 144)




Carrots and turnips
Cabbages
Chillies and peppers, green (Capsicum spp. and Pimenta spp.)
Eggplants (aubergines)
Potatoes
Onions and shallots, dry (excluding dehydrated)
Pears
Other vegetables, fresh n.e.c.
Lettuce and chicory
Apples
Cauliflowers and broccoli


Total Yield by Category (grouped)
Category — % of Total Yield
Vegetables: 57.994%
Fruits: 19.948%
Roots & Tubers: 8.275%
Sugar Crops: 6.152%
Cereals: 4.157%
Pulses/Legumes: 1.263%
Oilseeds: 0.718%
Nuts: 0.697%
Fiber Crops: 0.269%
Spices & Aromatics: 0.250%
Stimulant Crops: 0.161%
Beverages: 0.115%
Industrial Crops: 0.000%
Oils & Fats: 0.000%

Now that we have established that given the number of crops in our dataset we kind of have to group them i have chose to alter the dataset such that these

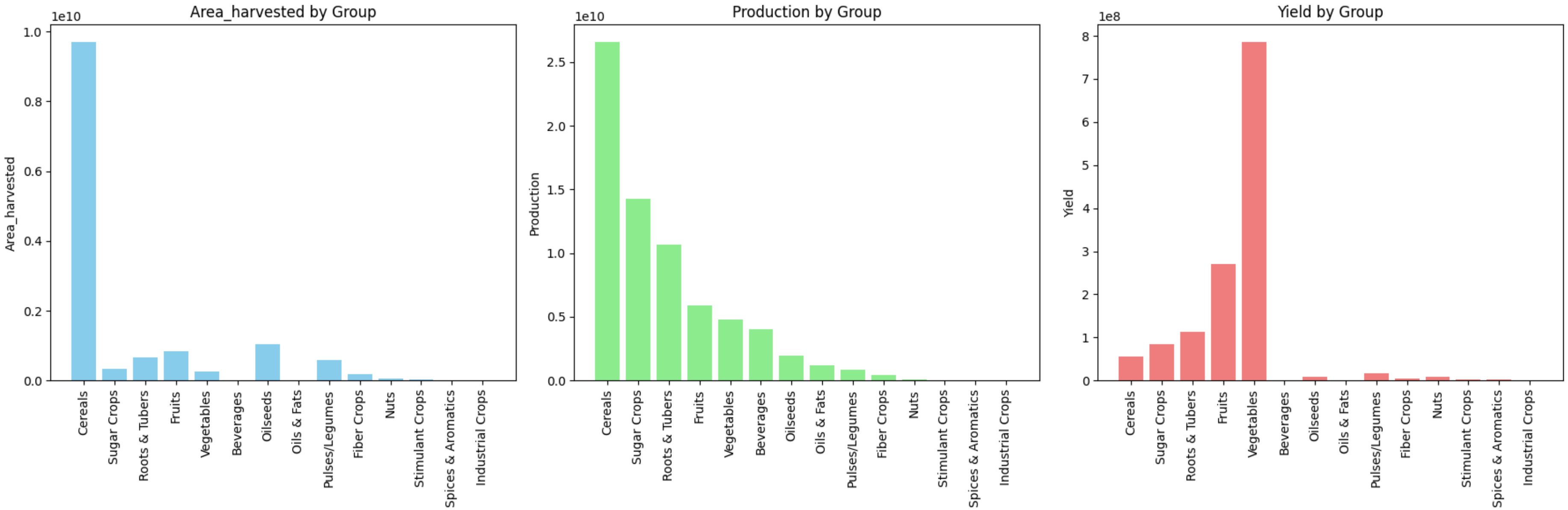
In [76]:
```python
# 1) Copy df_merged so the original remains unchanged
df_temp = df_merged.copy()

# 2) Add the 'Group' column by mapping each 'Item' to its category
df_temp['Group'] = df_temp['Item'].map(crop_to_group)

# 3) Define a helper to pick the first non-null value in a series
def first_non_null(series):
    non_null = series.dropna()
    return non_null.iloc[0] if not non_null.empty else np.nan

# 4) Group by both 'Group' and 'Year' to retain year information
df_grouped = (
    df_temp
    .groupby(['Group', 'Year'])
    .agg({
        'Area_harvested':       'sum',
        'Unit_Area_harvested':  first_non_null,
        'Production':           'sum',
        'Unit_Production':      first_non_null,
        'Yield':                'sum',
        'Unit_Yield':           first_non_null
    })
    .reset_index()
)

# 5) (Optional) To display large numbers without scientific notation:
pd.set_option('display.float_format', '{:,.0f}'.format)

# 6) View the result
print(df_grouped)
```

```
         Group  Year  Area_harvested Unit_Area_harvested  Production  \
0    Beverages  1961         105,975                  ha  39,687,081   
1    Beverages  1962         110,761                  ha  46,654,232   
2    Beverages  1963         117,763                  ha  44,456,323   
3    Beverages  1964         121,524                  ha  49,621,382   
4    Beverages  1965         123,504                  ha  50,226,655   
..         ...   ...             ...                 ...         ...   
870 Vegetables  2019       2,878,939                  ha  76,594,172   
871 Vegetables  2020       2,907,378                  ha  77,633,080   
872 Vegetables  2021       2,907,543                  ha  79,882,333   
873 Vegetables  2022       2,675,089                  ha  71,661,490   
874 Vegetables  2023       2,686,924                  ha  72,559,786   

    Unit_Production      Yield Unit_Yield  
0                 t     15,032      kg/ha  
1                 t     15,989      kg/ha  
2                 t     17,234      kg/ha  
3                 t     18,440      kg/ha  
4                 t     18,082      kg/ha  
..              ...        ...        ...  
870               t 19,465,401      kg/ha  
871               t 19,975,897      kg/ha  
872               t 20,327,850      kg/ha  
873               t 19,863,159      kg/ha  
874               t 19,937,441      kg/ha  

[875 rows x 8 columns]
```

In [80]:

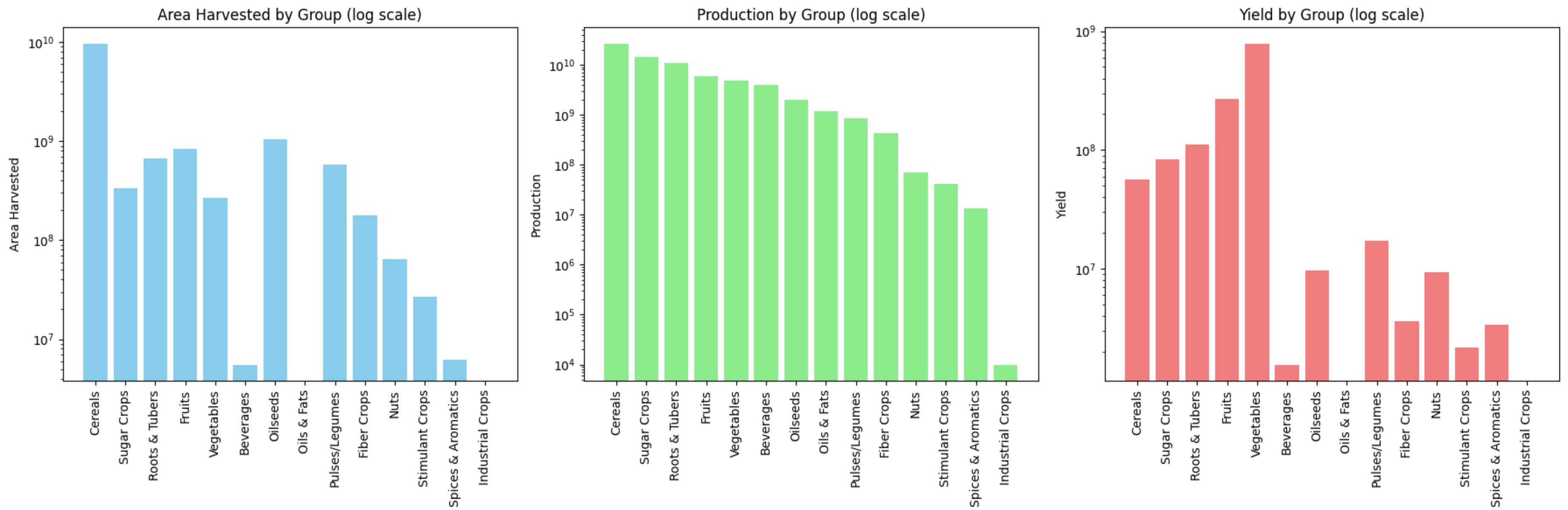
```python
# 2) Aggregate across years so that each Group has a single total for each metric
df_summary = df_grouped.groupby('Group').agg({
    'Area_harvested': 'sum',
    'Production':     'sum',
    'Yield':          'sum'
}).reset_index()

# (Optional) Sort by Production so bars appear in descending order of production
df_summary = df_summary.sort_values('Production', ascending=False)

# 3) Create a figure with three subplots side by side
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True)

# 3A) Bar chart: Total Area_harvested by Group (no units in ylabel)
```

```
axes[0].bar(df_summary['Group'], df_summary['Area_harvested'], color='skyblue')
axes[0].set_title('Area_harvested by Group')
axes[0].set_ylabel('Area_harvested')
axes[0].tick_params(axis='x', rotation=90)

# 3B) Bar chart: Total Production by Group (no units in ylabel)
axes[1].bar(df_summary['Group'], df_summary['Production'], color='lightgreen')
axes[1].set_title('Production by Group')
axes[1].set_ylabel('Production')
axes[1].tick_params(axis='x', rotation=90)

# 3C) Bar chart: Total Yield by Group (no units in ylabel)
axes[2].bar(df_summary['Group'], df_summary['Yield'], color='lightcoral')
axes[2].set_title('Yield by Group')
axes[2].set_ylabel('Yield')
axes[2].tick_params(axis='x', rotation=90)

# 4) Tidy up layout so labels aren’t cut off
plt.tight_layout()
plt.show()
```



In [97]:
```
# 2) Aggregate across years so that each Group has a single total for each metric
df_summary = df_grouped.groupby('Group').agg({
```

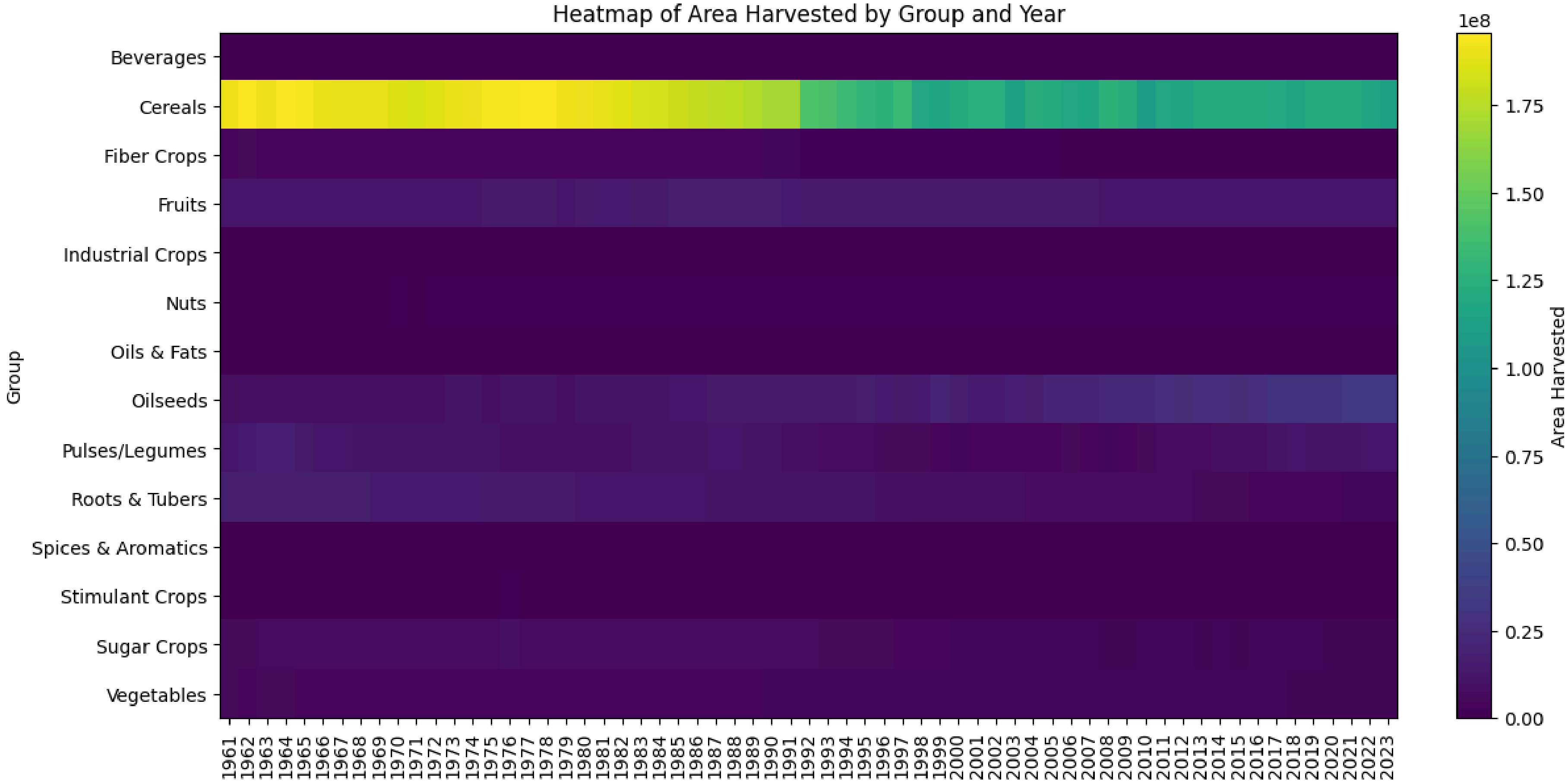
```python
    'Area_harvested': 'sum',
    'Production':     'sum',
    'Yield':          'sum'
}).reset_index()

# (Optional) Sort by Production so bars appear in descending order of production
df_summary = df_summary.sort_values('Production', ascending=False)

# 3) Create a figure with three subplots side by side
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=False)

# 3A) Bar chart: Total Area_harvested by Group (log scale)
axes[0].bar(df_summary['Group'], df_summary['Area_harvested'], color='skyblue')
axes[0].set_title('Area Harvested by Group (log scale)')
axes[0].set_ylabel('Area Harvested')
axes[0].tick_params(axis='x', rotation=90)
axes[0].set_yscale('log')

# 3B) Bar chart: Total Production by Group (log scale)
axes[1].bar(df_summary['Group'], df_summary['Production'], color='lightgreen')
axes[1].set_title('Production by Group (log scale)')
axes[1].set_ylabel('Production')
axes[1].tick_params(axis='x', rotation=90)
axes[1].set_yscale('log')

# 3C) Bar chart: Total Yield by Group (log scale)
axes[2].bar(df_summary['Group'], df_summary['Yield'], color='lightcoral')
axes[2].set_title('Yield by Group (log scale)')
axes[2].set_ylabel('Yield')
axes[2].tick_params(axis='x', rotation=90)
axes[2].set_yscale('log')

# 4) Tidy up layout so labels aren’t cut off
plt.tight_layout()
plt.show()
```



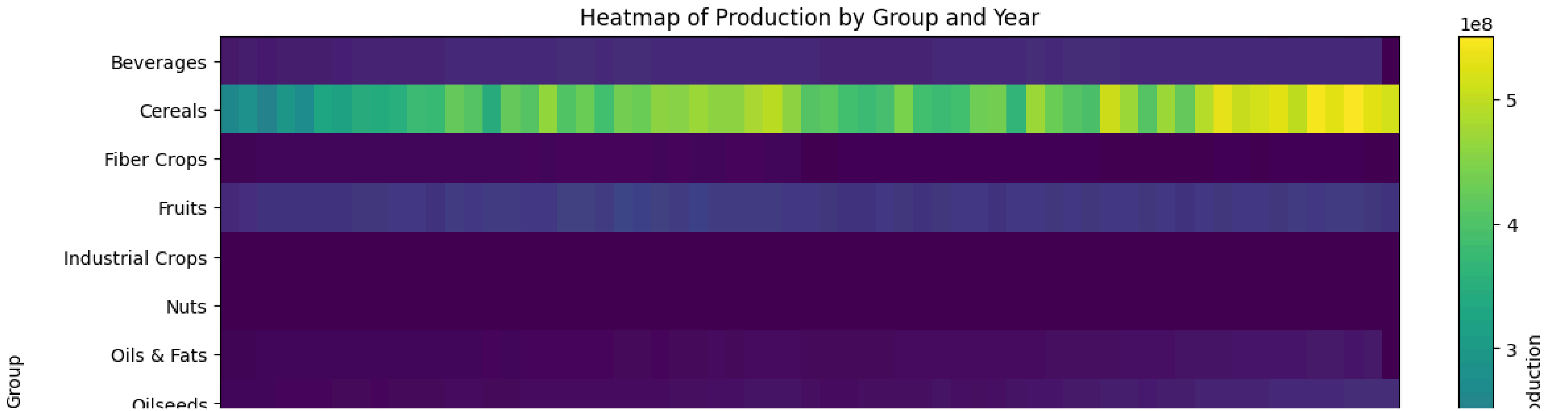
In [90]:

```python
# Create pivot tables for each metric
pivot_area = df_grouped.pivot(index='Group', columns='Year', values='Area_harvested').fillna(0)
pivot_production = df_grouped.pivot(index='Group', columns='Year', values='Production').fillna(0)
pivot_yield = df_grouped.pivot(index='Group', columns='Year', values='Yield').fillna(0)

# Plotting heatmaps using matplotlib
fig, axes = plt.subplots(3, 1, figsize=(12, 18), constrained_layout=True)

# 1) Heatmap for Area_harvested
im1 = axes[0].imshow(pivot_area, aspect='auto', cmap='viridis')
axes[0].set_title('Heatmap of Area Harvested by Group and Year')
axes[0].set_ylabel('Group')
axes[0].set_xticks(np.arange(len(pivot_area.columns)))
axes[0].set_yticks(np.arange(len(pivot_area.index)))
axes[0].set_xticklabels(pivot_area.columns, rotation=90)
axes[0].set_yticklabels(pivot_area.index)
fig.colorbar(im1, ax=axes[0], orientation='vertical', label='Area Harvested')

# 2) Heatmap for Production
im2 = axes[1].imshow(pivot_production, aspect='auto', cmap='viridis')
axes[1].set_title('Heatmap of Production by Group and Year')
axes[1].set_ylabel('Group')
```

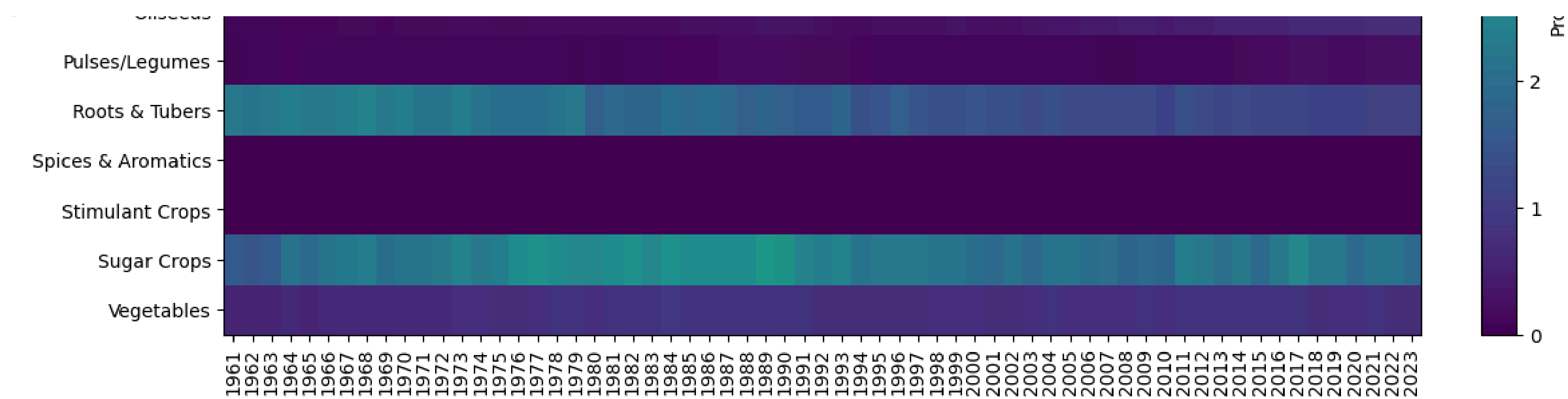
```python
axes[1].set_xticks(np.arange(len(pivot_production.columns)))
axes[1].set_yticks(np.arange(len(pivot_production.index)))
axes[1].set_xticklabels(pivot_production.columns, rotation=90)
axes[1].set_yticklabels(pivot_production.index)
fig.colorbar(im2, ax=axes[1], orientation='vertical', label='Production')

# 3) Heatmap for Yield
im3 = axes[2].imshow(pivot_yield, aspect='auto', cmap='viridis')
axes[2].set_title('Heatmap of Yield by Group and Year')
axes[2].set_ylabel('Group')
axes[2].set_xticks(np.arange(len(pivot_yield.columns)))
axes[2].set_yticks(np.arange(len(pivot_yield.index)))
axes[2].set_xticklabels(pivot_yield.columns, rotation=90)
axes[2].set_yticklabels(pivot_yield.index)
fig.colorbar(im3, ax=axes[2], orientation='vertical', label='Yield')

plt.show()
```

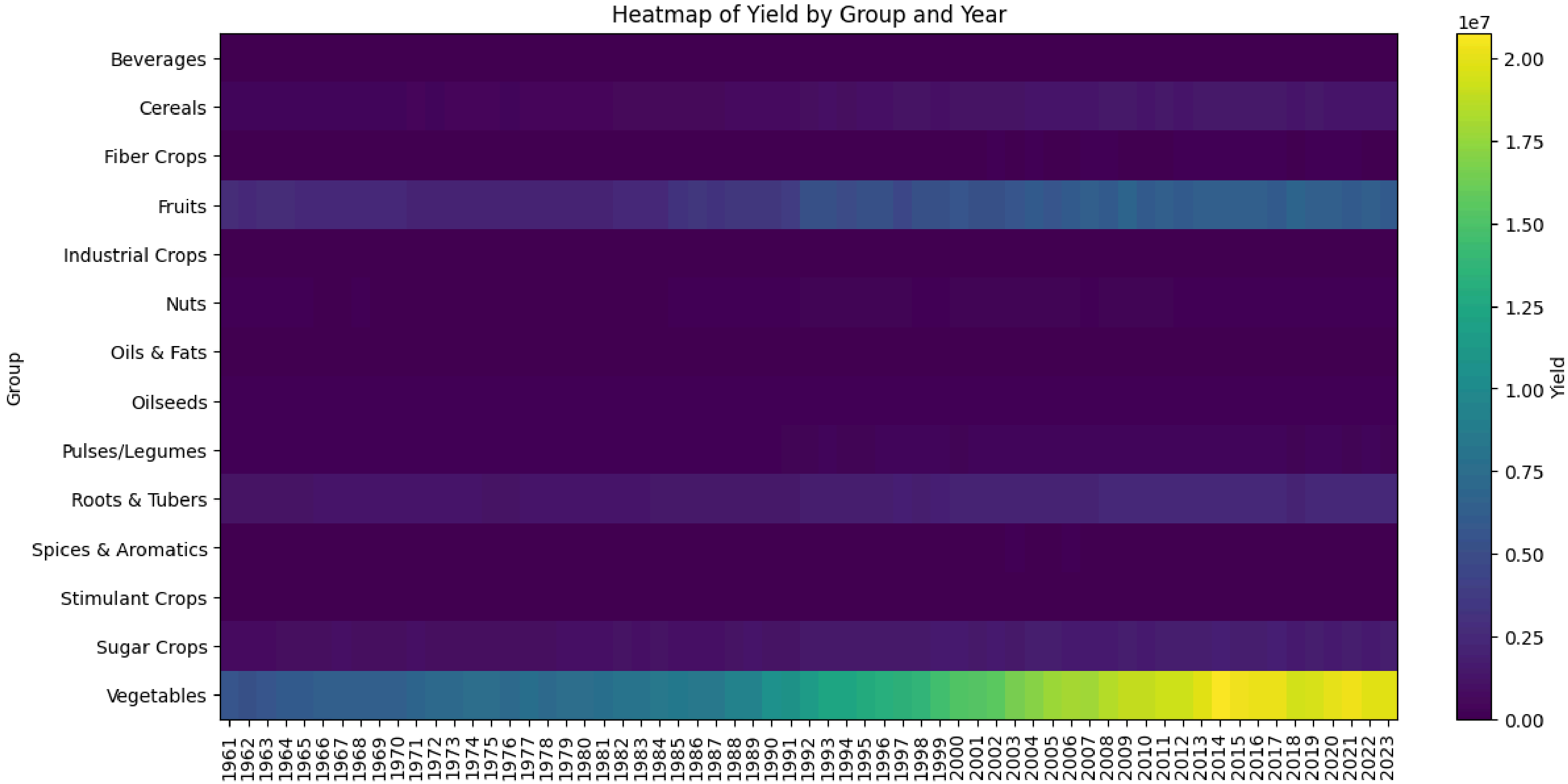


Heatmap of Area Harvested by Group and Year
Beverages
Cereals
Fiber Crops
Fruits
Industrial Crops
Nuts
Oils & Fats
Oilseeds
Pulses/Legumes
Roots & Tubers
Spices & Aromatics
Stimulant Crops
Sugar Crops
Vegetables
Group
1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
1e8
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Area Harvested



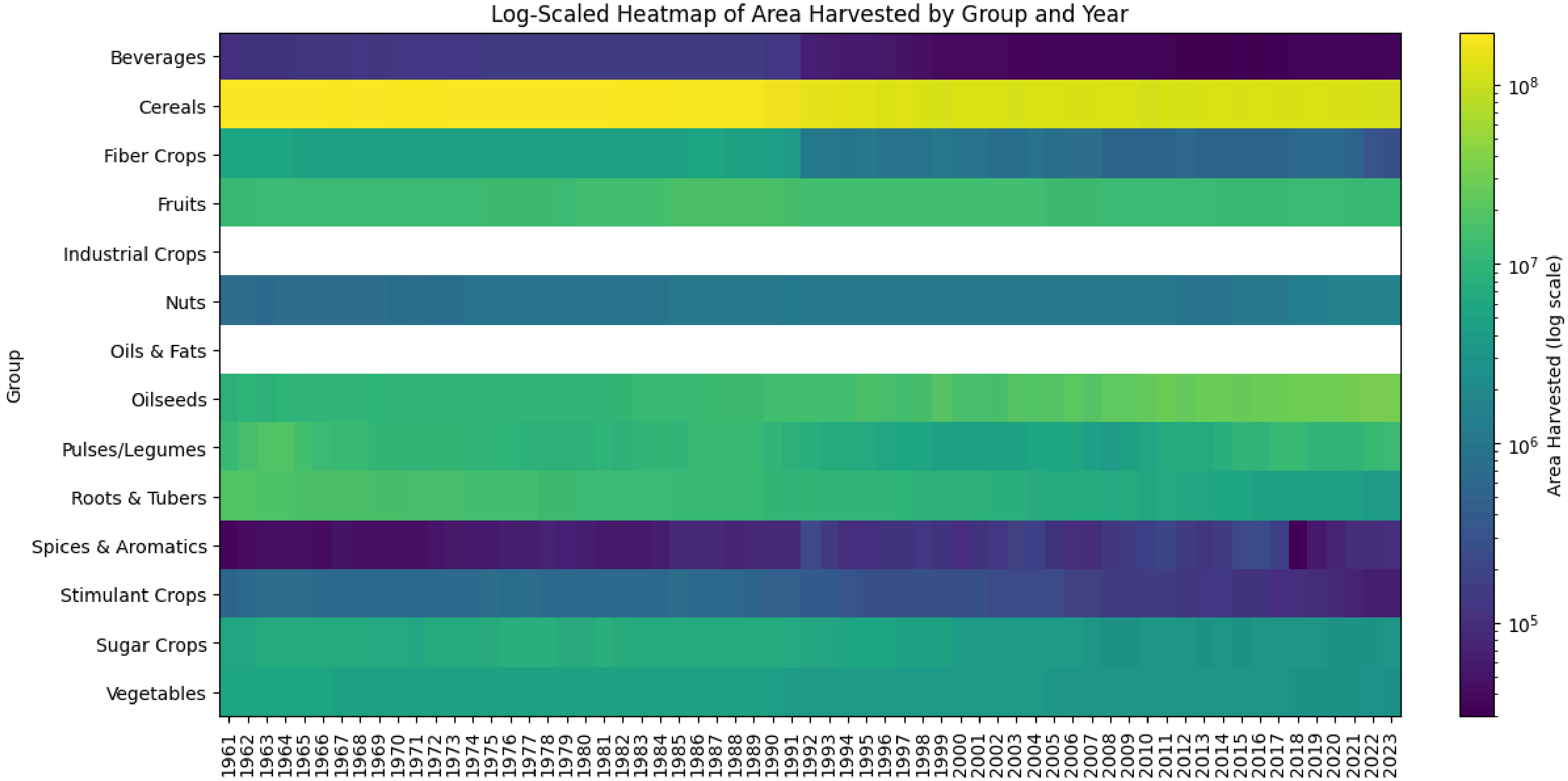
Heatmap of Production by Group and Year
Beverages
Cereals
Fiber Crops
Fruits
Industrial Crops
Nuts
Oils & Fats
Group
1e8
5
4
3


Oilseeds
Pulses/Legumes
Roots & Tubers
Spices & Aromatics
Stimulant Crops
Sugar Crops
Vegetables
1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
0
1
2

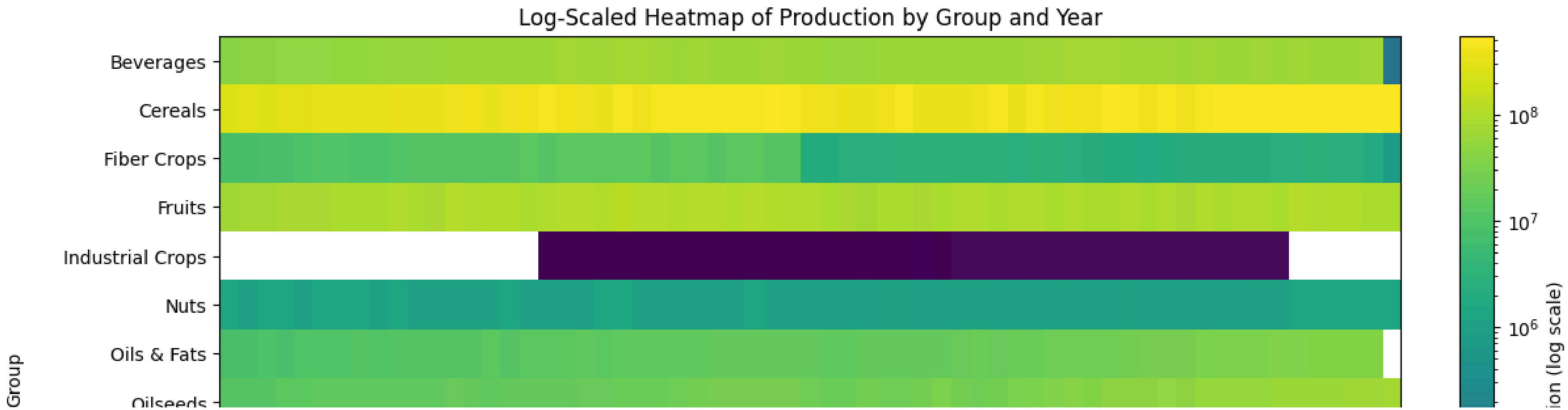

Heatmap of Yield by Group and Year
Group
Beverages
Cereals
Fiber Crops
Fruits
Industrial Crops
Nuts
Oils & Fats
Oilseeds
Pulses/Legumes
Roots & Tubers
Spices & Aromatics
Stimulant Crops
Sugar Crops
Vegetables
1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
1e7
Yield
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

In [89]:
```
# Create pivot tables for each metric
pivot_area = df_grouped.pivot(index='Group', columns='Year', values='Area_harvested').fillna(0)
pivot_production = df_grouped.pivot(index='Group', columns='Year', values='Production').fillna(0)
pivot_yield = df_grouped.pivot(index='Group', columns='Year', values='Yield').fillna(0)

# Determine nonzero minima for log scaling
area_min = pivot_area[pivot_area > 0].min().min()
prod_min = pivot_production[pivot_production > 0].min().min()
yield_min = pivot_yield[pivot_yield > 0].min().min()

# Plotting heatmaps using matplotlib with log normalization
fig, axes = plt.subplots(3, 1, figsize=(12, 18), constrained_layout=True)

# 1) Heatmap for Area_harvested (log scale)
im1 = axes[0].imshow(
    pivot_area,
    aspect='auto',
    cmap='viridis',
    norm=LogNorm(vmin=area_min, vmax=pivot_area.max().max())
)
axes[0].set_title('Log-Scaled Heatmap of Area Harvested by Group and Year')
axes[0].set_ylabel('Group')
axes[0].set_xticks(np.arange(len(pivot_area.columns)))
axes[0].set_yticks(np.arange(len(pivot_area.index)))
axes[0].set_xticklabels(pivot_area.columns, rotation=90)
axes[0].set_yticklabels(pivot_area.index)
fig.colorbar(im1, ax=axes[0], orientation='vertical', label='Area Harvested (log scale)')

# 2) Heatmap for Production (log scale)
im2 = axes[1].imshow(
    pivot_production,
    aspect='auto',
    cmap='viridis',
    norm=LogNorm(vmin=prod_min, vmax=pivot_production.max().max())
)
axes[1].set_title('Log-Scaled Heatmap of Production by Group and Year')
axes[1].set_ylabel('Group')
axes[1].set_xticks(np.arange(len(pivot_production.columns)))
axes[1].set_yticks(np.arange(len(pivot_production.index)))
axes[1].set_xticklabels(pivot_production.columns, rotation=90)
axes[1].set_yticklabels(pivot_production.index)
```

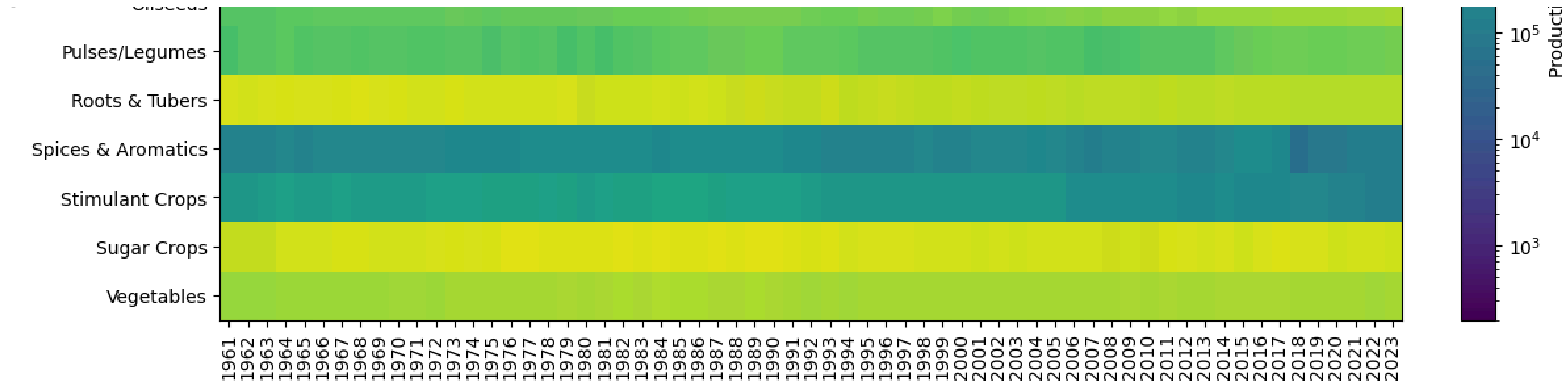
```
fig.colorbar(im2, ax=axes[1], orientation='vertical', label='Production (log scale)')

# 3) Heatmap for Yield (log scale)
im3 = axes[2].imshow(
    pivot_yield,
    aspect='auto',
    cmap='viridis',
    norm=LogNorm(vmin=yield_min, vmax=pivot_yield.max().max())
)
axes[2].set_title('Log-Scaled Heatmap of Yield by Group and Year')
axes[2].set_ylabel('Group')
axes[2].set_xticks(np.arange(len(pivot_yield.columns)))
axes[2].set_yticks(np.arange(len(pivot_yield.index)))
axes[2].set_xticklabels(pivot_yield.columns, rotation=90)
axes[2].set_yticklabels(pivot_yield.index)
fig.colorbar(im3, ax=axes[2], orientation='vertical', label='Yield (log scale)')

plt.show()
```

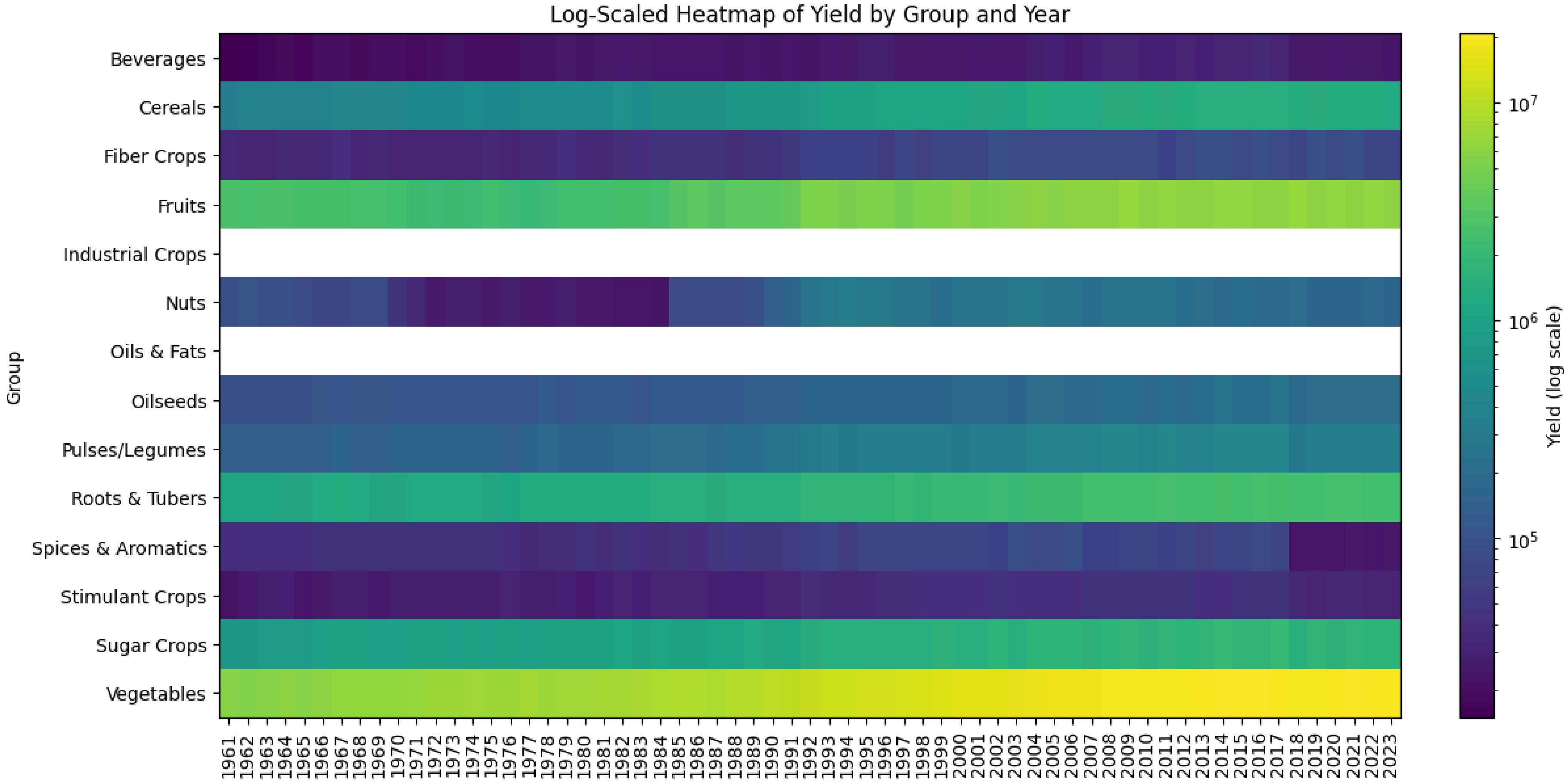


Log-Scaled Heatmap of Area Harvested by Group and Year
Beverages
Cereals
Fiber Crops
Fruits
Industrial Crops
Nuts
Oils & Fats
Oilseeds
Pulses/Legumes
Roots & Tubers
Spices & Aromatics
Stimulant Crops
Sugar Crops
Vegetables
Group
1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
10^8
10^7
10^6
10^5
Area Harvested (log scale)



Log-Scaled Heatmap of Production by Group and Year
Beverages
Cereals
Fiber Crops
Fruits
Industrial Crops
Nuts
Oils & Fats
Oilseeds
Group
10^8
10^7
10^6
ion (log scale)

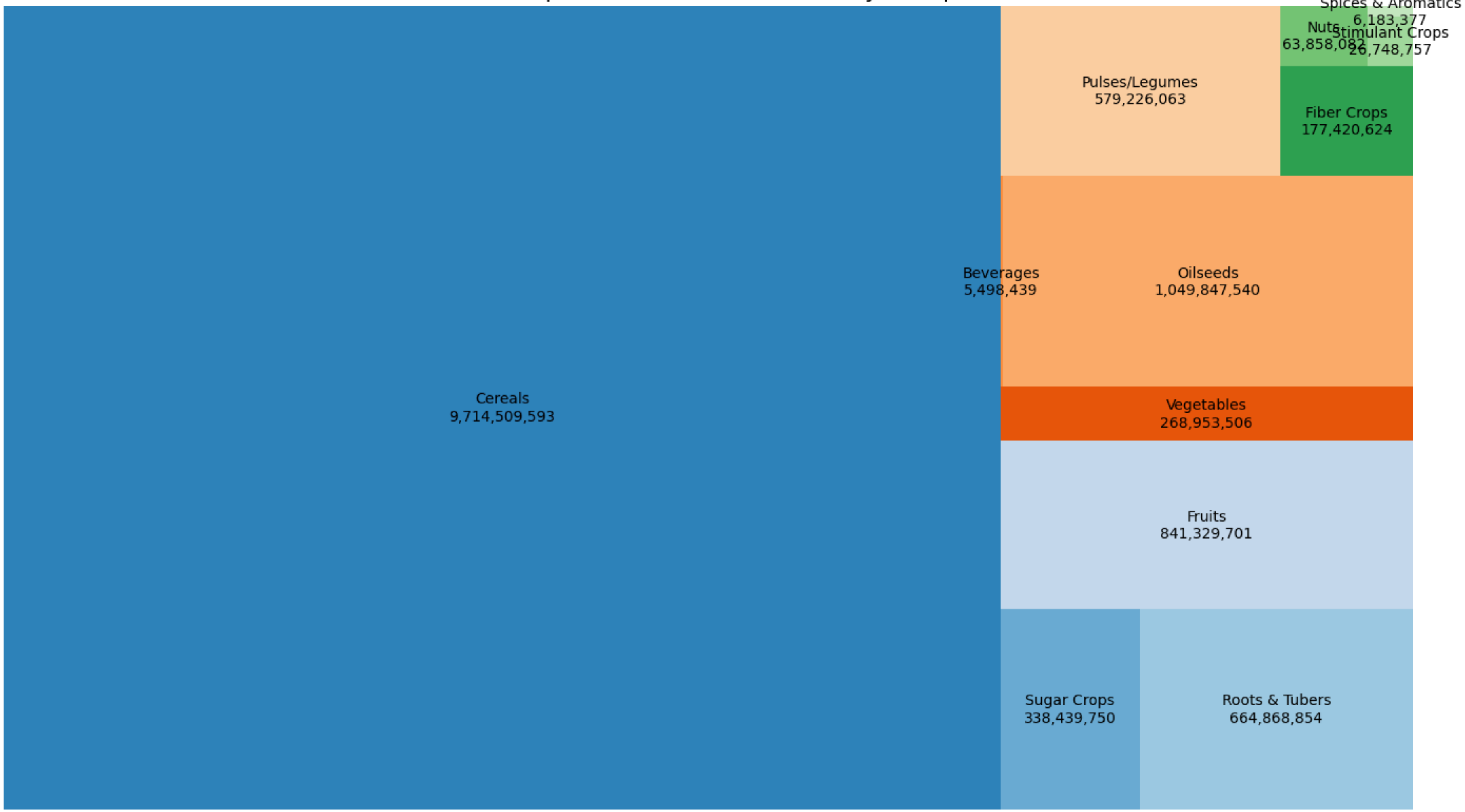

Pulses/Legumes
Roots & Tubers
Spices & Aromatics
Stimulant Crops
Sugar Crops
Vegetables
1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
10^5
10^4
10^3


Log-Scaled Heatmap of Yield by Group and Year
Group
Beverages
Cereals
Fiber Crops
Fruits
Industrial Crops
Nuts
Oils & Fats
Oilseeds
Pulses/Legumes
Roots & Tubers
Spices & Aromatics
Stimulant Crops
Sugar Crops
Vegetables
1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023
Yield (log scale)
10^7
10^6
10^5

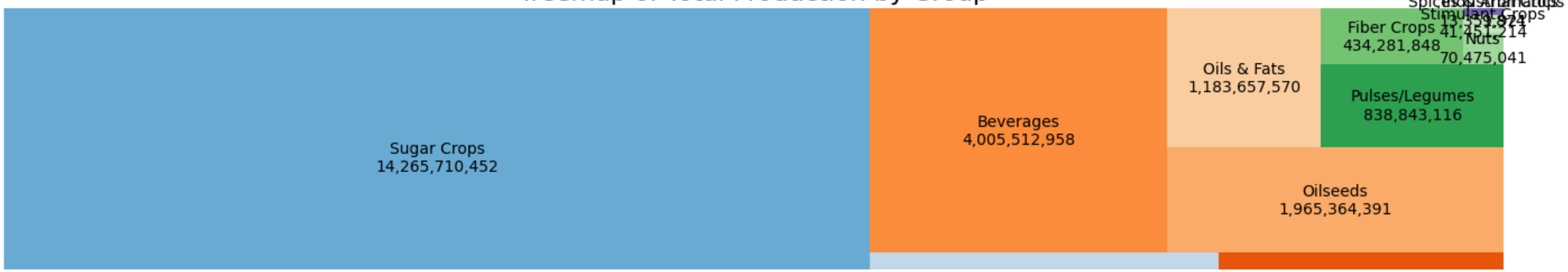
In [95]:
```python
df_summary = (
    df_grouped
    .groupby('Group')
    .agg({
        'Area_harvested': 'sum',
        'Production':     'sum',
        'Yield':          'sum'
    })
    .reset_index()
)

# Sort (e.g. by Production) so the order is consistent in all treemaps:
df_summary = df_summary.sort_values('Production', ascending=False)

# ──────────────────────────────────────────────────────────────────────────────
# Now draw three treemaps stacked **vertically**, but with a very large figure
# so that small rectangles get more absolute pixel-space.
# ──────────────────────────────────────────────────────────────────────────────

fig, axes = plt.subplots(3, 1, figsize=(14, 24), constrained_layout=True)

# 1) TREEMAP FOR AREA_HARVESTED
df_area = df_summary[df_summary['Area_harvested'] > 0].copy()
sizes_area  = df_area['Area_harvested'].values
labels_area = [f"{grp}\n{val:,.0f}" for grp, val in zip(df_area['Group'], df_area['Area_harvested'])]

axes[0].set_title("Treemap of Total Area Harvested by Group", fontsize=16)
squarify.plot(
    sizes=sizes_area,
    label=labels_area,
    ax=axes[0],
    pad=0,                                  # no padding → each rectangle uses max space
    color=plt.cm.tab20c.colors[:len(sizes_area)],
    text_kwargs={'fontsize': 10}      # moderately sized text
)
axes[0].axis('off')


# 2) TREEMAP FOR PRODUCTION
df_prod = df_summary[df_summary['Production'] > 0].copy()
sizes_prod  = df_prod['Production'].values
```

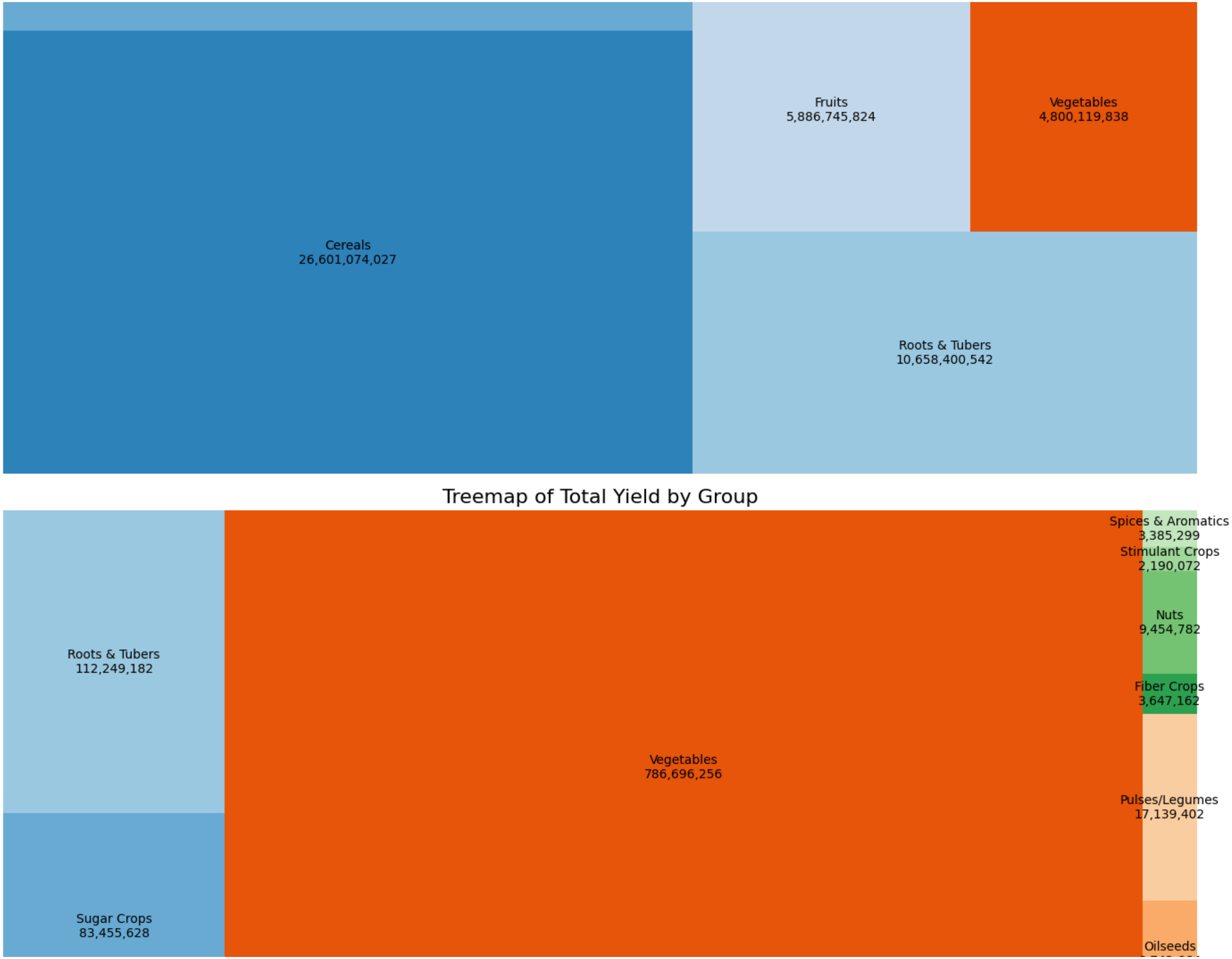
```python
labels_prod = [f"{grp}\n{val:,.0f}" for grp, val in zip(df_prod['Group'], df_prod['Production'])]

axes[1].set_title("Treemap of Total Production by Group", fontsize=16)
squarify.plot(
    sizes=sizes_prod,
    label=labels_prod,
    ax=axes[1],
    pad=0,
    color=plt.cm.tab20c.colors[:len(sizes_prod)],
    text_kwargs={'fontsize': 10}
)
axes[1].axis('off')


# 3) TREEMAP FOR YIELD
df_yield = df_summary[df_summary['Yield'] > 0].copy()
sizes_yield  = df_yield['Yield'].values
labels_yield = [f"{grp}\n{val:,.0f}" for grp, val in zip(df_yield['Group'], df_yield['Yield'])]

axes[2].set_title("Treemap of Total Yield by Group", fontsize=16)
squarify.plot(
    sizes=sizes_yield,
    label=labels_yield,
    ax=axes[2],
    pad=0,
    color=plt.cm.tab20c.colors[:len(sizes_yield)],
    text_kwargs={'fontsize': 10}
)
axes[2].axis('off')

plt.show()
```

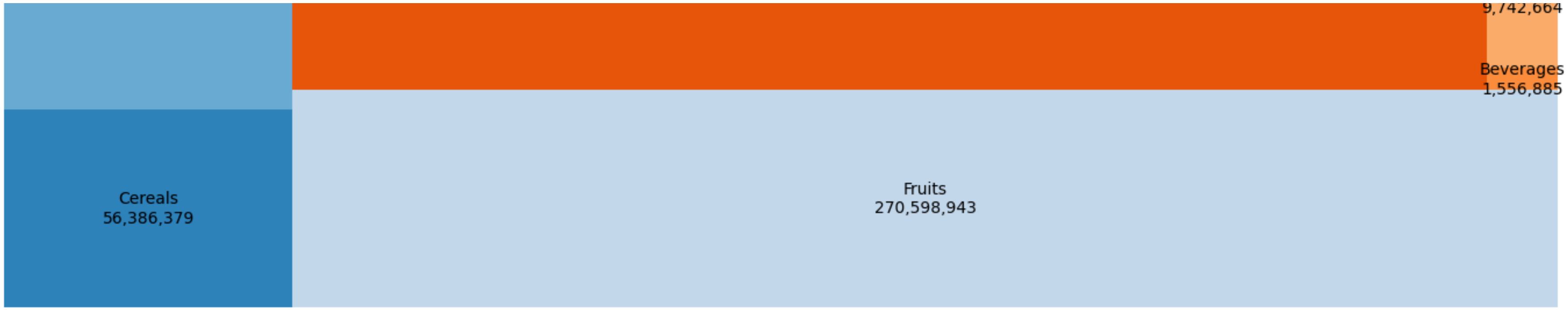
Treemap of Total Area Harvested by Group



Treemap of Total Production by Group




Cereals
26,601,074,027
Fruits
5,886,745,824
Vegetables
4,800,119,838
Roots & Tubers
10,658,400,542
Treemap of Total Yield by Group
Roots & Tubers
112,249,182
Sugar Crops
83,455,628
Vegetables
786,696,256
Spices & Aromatics
3,385,299
Stimulant Crops
2,190,072
Nuts
9,454,782
Fiber Crops
3,647,162
Pulses/Legumes
17,139,402
Oilseeds



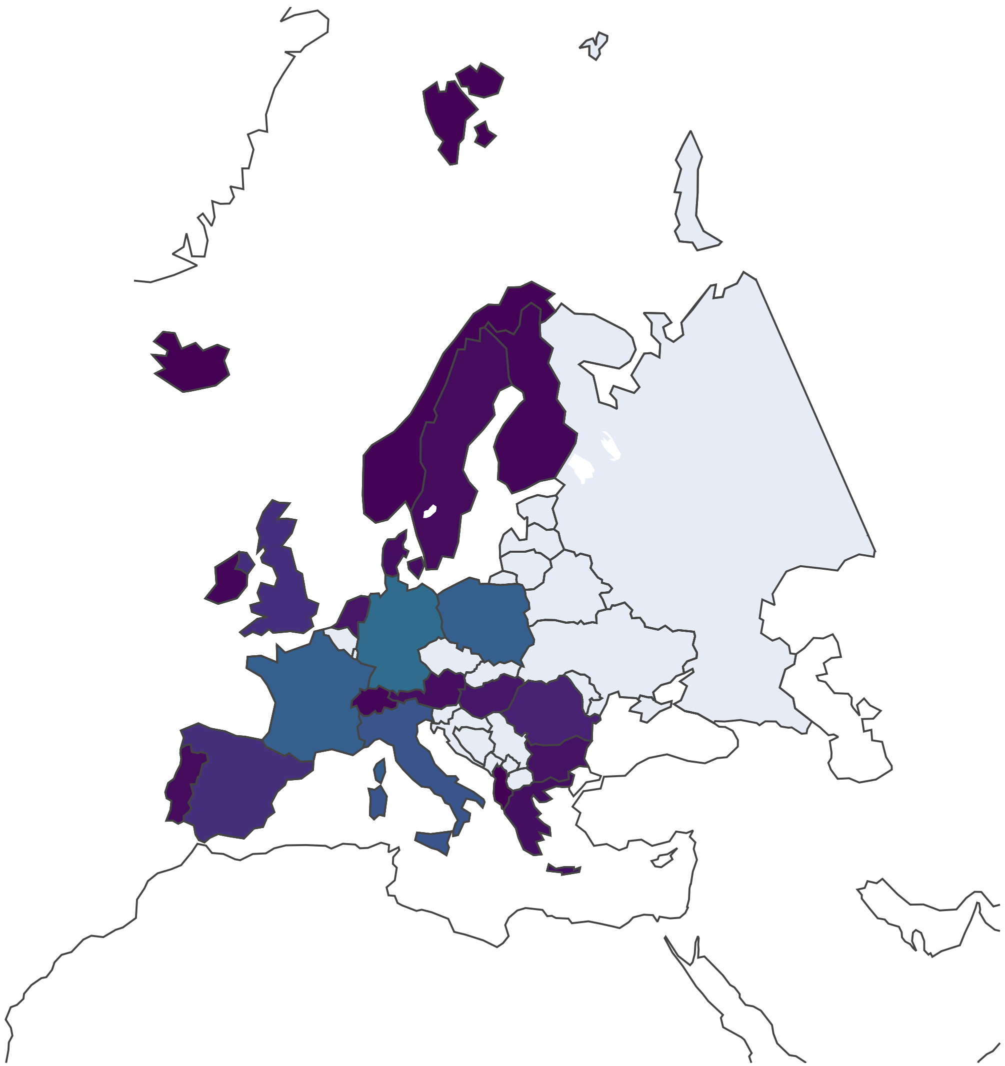
In [96]:
```python
# 2) Define a helper function to map FAOSTAT names → ISO-3 codes
def faostat_to_iso3(name):
    try:
        return pycountry.countries.lookup(name).alpha_3
    except (LookupError, AttributeError):
        return None

# 3) Collect all three metrics into one “long” DataFrame for convenience
long_df = (
    df_merged
    .melt(
        id_vars=['Area', 'Year'],
        value_vars=['Production', 'Area_harvested', 'Yield'],
        var_name='Metric',
        value_name='Value'
    )
    # Drop any NaN values in Value (just in case)
    .dropna(subset=['Value'])
)

# 4) Lookup ISO3 for each row’s “Area”
long_df['iso_alpha'] = long_df['Area'].apply(faostat_to_iso3)

# 5) Identify any unmapped areas (once) and override manually if needed
missing_iso = long_df.loc[long_df['iso_alpha'].isna(), 'Area'].unique()
if len(missing_iso) > 0:
    print("Unmapped FAOSTAT areas (will be dropped or overridden):", missing_iso)

# 6) Manual overrides for known FAOSTAT names that pycountry can’t resolve:
manual_overrides = {
```

```python
    "Belgium-Luxembourg":             None,  # drop
    "Czechoslovakia":                 None,  # drop
    "Netherlands (Kingdom of the)":   "NLD",  # Netherlands
    "Serbia and Montenegro":          None,  # drop
    "USSR":                           None,  # drop
    "Yugoslav SFR":                   None   # drop
}

# 7) Apply overrides
long_df.loc[long_df['iso_alpha'].isna(), 'iso_alpha'] = (
    long_df.loc[long_df['iso_alpha'].isna(), 'Area']
    .map(manual_overrides)
)

# 8) Drop any rows still missing iso_alpha
long_df = long_df.dropna(subset=['iso_alpha'])

# 9) For each metric, build and display an animated choropleth
for metric_name, title, color_label in [
        ("Production",     "Crop Production by Country over Time (Europe, 1961–2023)",     "Production (t)"),
        ("Area_harvested", "Crop Area Harvested by Country over Time (Europe, 1961–2023)", "Area Harvested (ha)"),
        ("Yield",          "Crop Yield by Country over Time (Europe, 1961–2023)",          "Yield (kg/ha)")
    ]:

    # 9a) Filter the long DF to just this metric
    df_metric = long_df[long_df['Metric'] == metric_name].copy()

    # 9b) Group & sum by (Area, Year, iso_alpha)—just in case there were multiple rows per country/year
    country_year = (
        df_metric
        .groupby(['Area', 'Year', 'iso_alpha'], as_index=False)['Value']
        .sum()
    )

    # 9c) Build the animated choropleth
    fig = px.choropleth(
        country_year,
        locations="iso_alpha",
        color="Value",
        hover_name="Area",
        animation_frame="Year",
        range_color=[country_year["Value"].min(), country_year["Value"].max()],
```

```python
        color_continuous_scale="Viridis",
        title=title,
        scope="europe"
    )

    # 9d) Tweak layout (size, remove frame)
    fig.update_layout(
        width=1200,
        height=800,
        geo=dict(showframe=False, showcoastlines=True),
        title_x=0.5
    )

    # 9e) Change the colorbar label
    fig.update_coloraxes(colorbar_title=color_label)

    # 9f) Render as HTML in the notebook
    html_str = fig.to_html(include_plotlyjs="cdn")
    display(HTML(html_str))
```

```
Unmapped FAOSTAT areas (will be dropped or overridden): ['Belgium-Luxembourg' 'Czechoslovakia' 'Netherlands (Kingdom of the)'
 'Serbia and Montenegro' 'USSR' 'Yugoslav SFR']
```

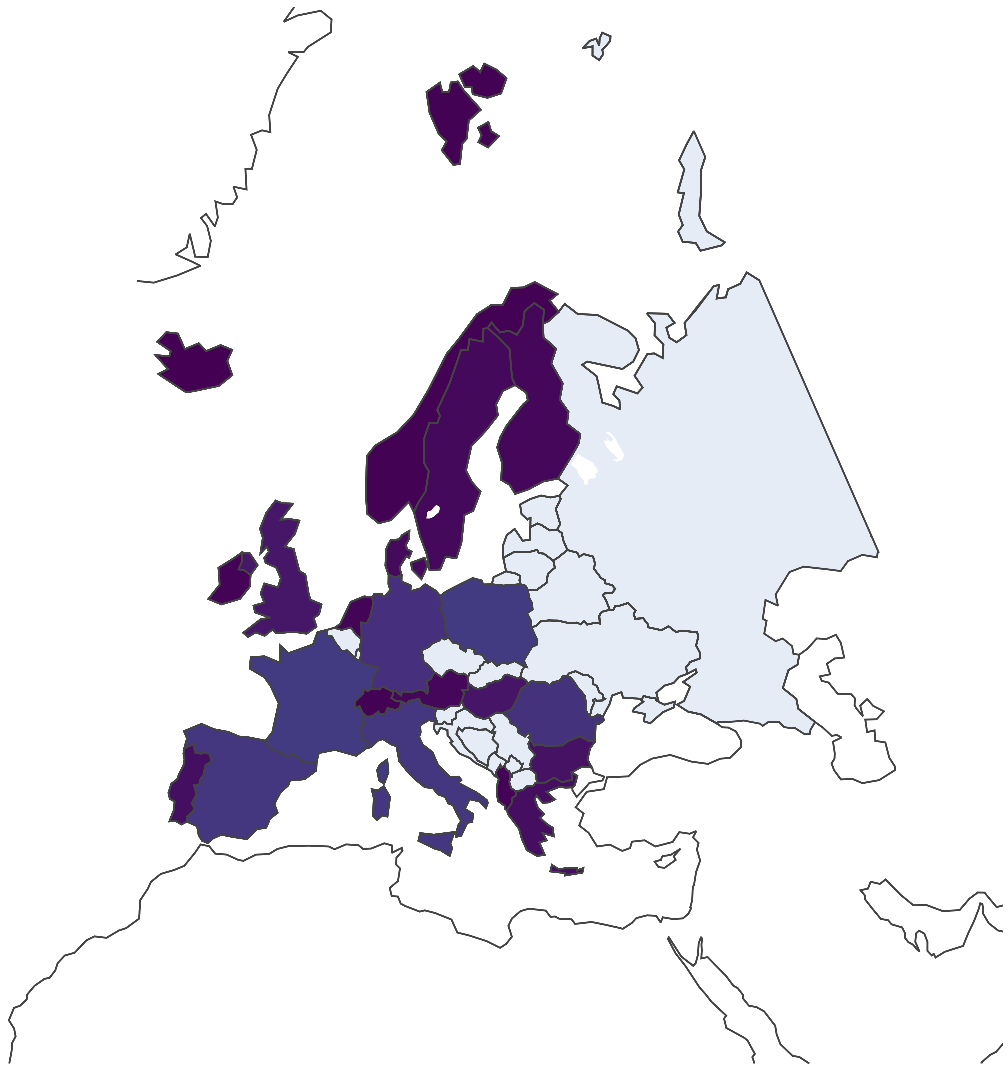
Crop Production by Country over Time (Europe, 1961–2023)



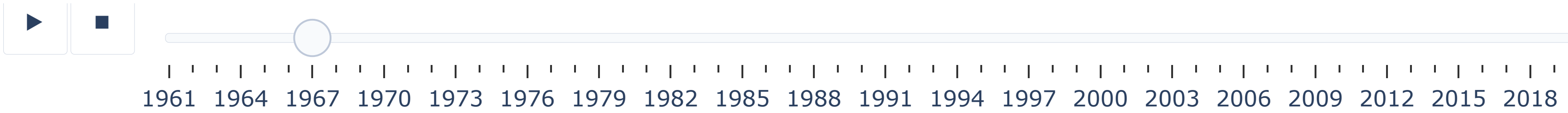
Year=1961

1961 1964 1967 1970 1973 1976 1979 1982 1985 1988 1991 1994 1997 2000 2003 2006 2009 2012 2015

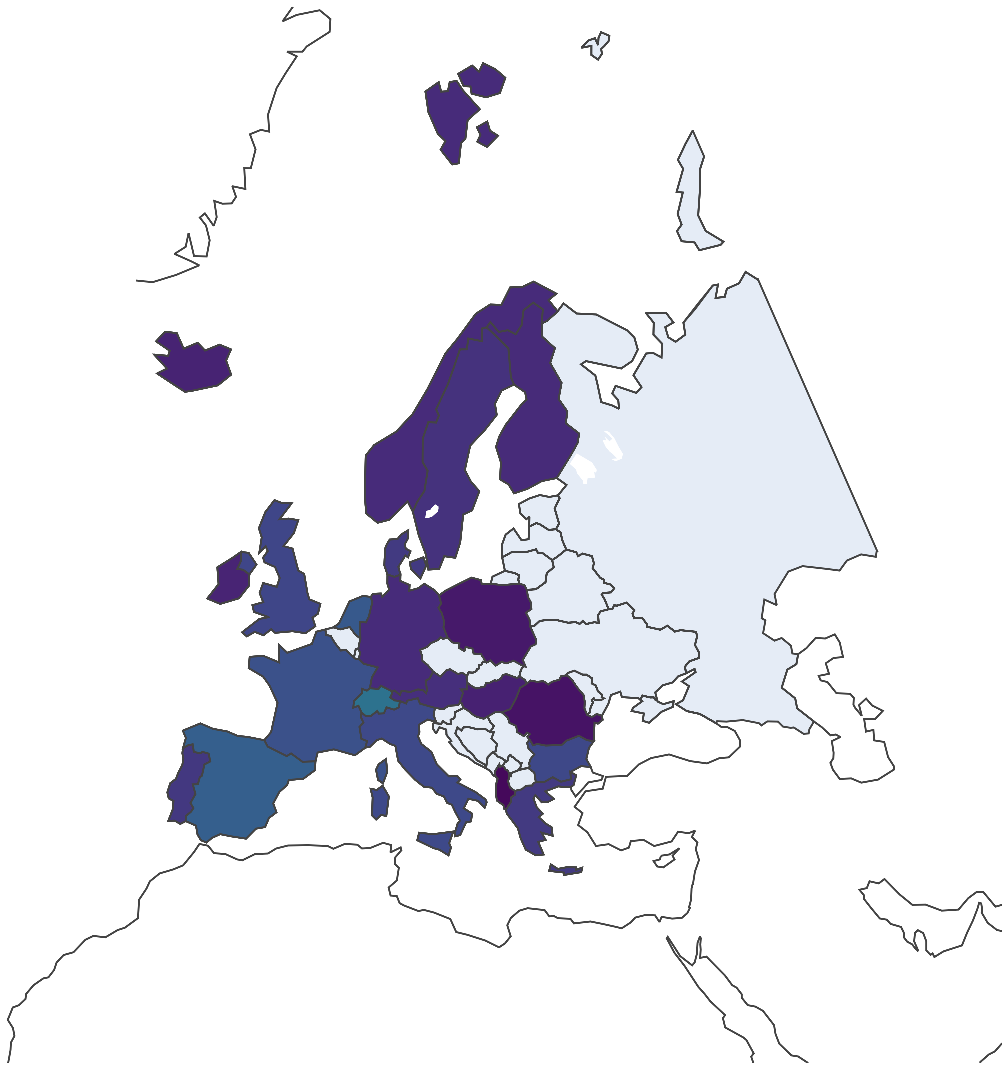
Crop Area Harvested by Country over Time (Europe, 1961–2023)



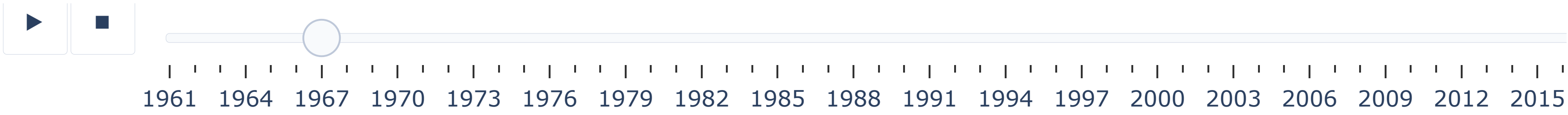
Year=1967


1961
1964
1967
1970
1973
1976
1979
1982
1985
1988
1991
1994
1997
2000
2003
2006
2009
2012
2015
2018

Crop Yield by Country over Time (Europe, 1961–2023)



Year=1967


1961 1964 1967 1970 1973 1976 1979 1982 1985 1988 1991 1994 1997 2000 2003 2006 2009 2012 2015