

exercises-pollinators-datasets-exploration

June 5, 2022

1 Exercises - Pollinators datasets exploration

Exercises with some pollinators datasets.

1.1 Packages import

```
[1]: import os # operating system functions
import chardet # Universal Character Encoding Detector
import requests # web requests
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import random
import re # regular expression operations
from sklearn.model_selection import StratifiedShuffleSplit # dataset subsetting
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder # mange categorical data
from sklearn import metrics # results evaluation
from sklearn.impute import SimpleImputer # tool for dealing with missing values
import association_metrics as am # implementation of Cramer's V correlation
import matplotlib as plt # data visualization
from mpl_toolkits.mplot3d import Axes3D # visualization 3D
import seaborn as sb # data visualization
import graphviz # graph visualization
import plotly.express as px # data visualization, also 3D
from matplotlib.animation import FuncAnimation # plot animations
```

We probably will download and save more than 1 dataset so let's make a function for it

```
[20]: def DatasetDownload(dataset_url, dataset_directory_path, dataset_file_name):
    print("Download started")
    request_dataset = requests.get(dataset_url, allow_redirects=True)
    print("Download completed")
    if request_dataset.status_code != 200:
        print(f"Request status: {request_dataset.status_code}")
    else:
        print("Writing started")
        os.makedirs(dataset_directory_path, exist_ok=True)
```

```

        open( dataset_directory_path + dataset_file_name , 'wb').
↪write(request_dataset.content)
        print("Writing completed")
        print("End")
        return

```

1.2 Insect Pollinator Initiative - Natural History Museum Data Portal

Graham N Stone; Alfried Vogler; Adam Vanbergen; Jacqueline Mackenzie-Dodds (2017). Dataset: Insect Pollinators Archive. Resource: Insect Pollinator Initiative. Natural History Museum Data Portal (data.nhm.ac.uk). <https://doi.org/10.5519/0062900>

Retrieved: 16:39 19 Mar 2022 (GMT)

1.2.1 IPI-NHMDP - Data download - (One shoot execution)

Let's use the original website.

Next steps are “one shoot execution”, you should execute it only the first time, once did it you can go directly to *Starting points* that you'll find along the code.

```

[4]: # Dataset url
NHMDP_PI_dataset_url = 'https://data.nhm.ac.uk/dataset/
↪46e122c6-7acd-44ec-a354-81a412da419a/resource/
↪784d74b6-6b0e-4fd4-b0b5-798ac7b1a11b/download/ipifordataportal.xlsx'

# Desired directory
NHMDP_PI_dataset_directory = 'Datasets/Pollinators/NHMDP/PollinatorsInitiative'

# Desired file name
NHMDP_PI_dataset_name = 'PollinatorsInitiative.xlsx'

```

```

[12]: # Download and Save
DatasetDownload(NHMDP_PI_dataset_url, NHMDP_PI_dataset_directory,
↪NHMDP_PI_dataset_name)

```

```

Download started
Download completed
Writing started
Writing completed
End

```

1.2.2 IPI-NHMDP - Data import - Starting point

```

[5]: IPI_NHMDP_dataset = pd.
↪read_excel(NHMDP_PI_dataset_directory+NHMDP_PI_dataset_name,
↪engine='openpyxl')

```

1.2.3 IPI-NHMDP - Exploration

```
[14]: IPI_NHMDP_dataset.describe()
```

```
[14]:      Specimen No/Barcode
count      1.185400e+04
mean       1.006605e+07
std        7.403999e+03
min        1.005246e+07
25%        1.005963e+07
50%        1.006886e+07
75%        1.007182e+07
max        1.007598e+07
```

```
[5]: IPI_NHMDP_dataset.head()
```

```
[5]:      Project Name Specimen No Prefix \
0  Insect Pollinator Initiative - agriland      NHMUK
1  Insect Pollinator Initiative - agriland      NHMUK
2  Insect Pollinator Initiative - agriland      NHMUK
3  Insect Pollinator Initiative - agriland      NHMUK
4  Insect Pollinator Initiative - agriland      NHMUK

      Specimen No/Barcode Specimen Code      Country Province/State/Territory \
0      10052460      AL_11_01750  United Kingdom      England
1      10052461      AL_11_01751  United Kingdom      England
2      10052462      AL_11_01753  United Kingdom      England
3      10052463      AL_11_01754  United Kingdom      England
4      10052464      AL_11_01755  United Kingdom      England

      District/County/Shire Precise Locality Coll Date      Method      Collector \
0      West Yorkshire      Harden Moor 2011-06-27  Pan trap  M. McKerchar
1      West Yorkshire      Harden Moor 2011-06-27  Pan trap  M. McKerchar
2      West Yorkshire      Harden Moor 2011-06-27  Pan trap  M. McKerchar
3      West Yorkshire      Harden Moor 2011-06-27  Pan trap  M. McKerchar
4      West Yorkshire      Harden Moor 2011-06-27  Pan trap  M. McKerchar

      Collector 1 Collector 2      Identifier \
0  M  McKerchar      S P M Roberts
1  M  McKerchar      NaN  S P M Roberts
2  M  McKerchar      NaN  S P M Roberts
3  M  McKerchar      NaN  S P M Roberts
4  M  McKerchar      NaN  S P M Roberts

      Determination      SEX Stage
0  Lasioglossum cupromicans (Pérez, J., 1903)  Female  NaN
1  Lasioglossum cupromicans (Pérez, J., 1903)  Female  NaN
```

```

2 Lasioglossum cupromicans (Pérez, J., 1903) Female NaN
3 Lasioglossum cupromicans (Pérez, J., 1903) Female NaN
4 Lasioglossum fratellum (Perez, 1903) Female NaN

```

```
[6]: IPI_NHMDP_dataset.columns
```

```
[6]: Index(['Project Name', 'Specimen No Prefix', 'Specimen No/Barcode',
        'Specimen Code', 'Country', 'Province/State/Territory',
        'District/County/Shire', 'Precise Locality', 'Coll Date', 'Method',
        'Collector', 'Collector 1', 'Collector 2', 'Identifier',
        'Determination', 'SEX', 'Stage'],
        dtype='object')
```

Mmm I don't see particularly interesting information.

Let's check how many per state different specimens have been collected

```
[14]: IPI_NHMDP_dataset[["Country", "Specimen Code"]].groupby("Country").describe()
```

```
[14]:
```

	Specimen Code			
	count	unique	top freq	
Country				
United Kingdom	11852	11807	Wi-01-3.13-P10003	2

```
[15]: IPI_NHMDP_dataset[["Province/State/Territory", "Specimen Code"]].
      ↪groupby("Province/State/Territory").describe()
```

```
[15]:
```

	Specimen Code			
	count	unique	top freq	
Province/State/Territory				
England	10028	9996	Ca-05-1.12-P30003	2
Scotland	1824	1811	Ay-15-3.12-P50013	2

```
[16]: IPI_NHMDP_dataset[["Province/State/Territory", "District/County/Shire", "Specimen_
      ↪Code"]].groupby("District/County/Shire").describe()
```

```
[16]:
```

	Province/State/Territory		\	
	count	unique	top	freq
District/County/Shire				
Bedfordshire	1053	1	England	1053
Cambridgeshire	2356	1	England	2356
Cumbria	113	1	England	113
Dorset	492	1	England	492
Dumfries and Galloway	137	1	Scotland	137
East Ayrshire	523	1	Scotland	523
East Renfrewshire	29	1	Scotland	29
East Riding of Yorkshire	1471	1	England	1471
Highland	651	1	Scotland	651

Kent	173	1	England	173
Lancashire	219	1	England	219
North Lanarkshire	167	1	Scotland	167
North Yorkshire	254	1	England	254
Renfrewshire	14	1	Scotland	14
South Lanarkshire	303	1	Scotland	303
Staffordshire	1359	1	England	1359
West Yorkshire	895	1	England	895
Wiltshire	1643	1	England	1643

District/County/Shire	Specimen Code			top freq
	count	unique		
Bedfordshire	1053	1052	AL_11_03988	2
Cambridgeshire	2356	2340	Ca-01-1.13-P40002	2
Cumbria	113	113	Yo-08-1.12-P30003	1
Dorset	492	492	AL_12_07052	1
Dumfries and Galloway	137	137	Ay-08-3.12-P10001	1
East Ayrshire	523	523	Ay-01-3.12-P20001	1
East Renfrewshire	29	29	Ay-12-3.12-P10001	1
East Riding of Yorkshire	1471	1467	AL_11_02429	2
Highland	651	643	In-04-1.12-P50001	2
Kent	173	173	AL_12_06790	1
Lancashire	219	219	AL_11_02651	1
North Lanarkshire	167	162	Ay-15-3.12-P50009	2
North Yorkshire	254	253	AL_11_06052	2
Renfrewshire	14	14	Ay-09-3.12-P30001	1
South Lanarkshire	303	303	Ay-04-3.12-P10009	1
Staffordshire	1359	1359	St-02-3.12-P10001	1
West Yorkshire	895	894	AL_11_02507	2
Wiltshire	1643	1634	Wi-01-3.13-P40001	2

Could be nice try to represent these data on a geographical map... but it's a bit out of the exercise scope

1.3 Global pollinator database - Boreux & Klein - Figshare Dataset

Boreux, Virginie; Klein, Alexandra-Maria (2019): Global pollinator database. figshare. Dataset. <https://doi.org/10.6084/m9.figshare.9980471.v1>

1.3.1 GPD-F - Data download - (One shoot execution)

```
[3]: # Dataset url
GPD_F_dataset_url = 'https://figshare.com/ndownloader/files/18003863'

# Desired directory
GPD_F_dataset_directory = 'Datasets/Pollinators/Figshare/'
↳GlobalPollinatorDatabase'
```

```

# Desired file name
GPD_F_dataset_name = 'GlobalPollinatorDatabase.csv'

# Description dataset url
GPD_F_description_dataset_url = 'https://figshare.com/ndownloader/files/
↳18003860'

# Desired file name
GPD_F_description_dataset_name = 'GlobalPollinatorDatabaseDescription.csv'

```

```

[21]: # Download and Save
DatasetDownload(GPD_F_dataset_url, GPD_F_dataset_directory, GPD_F_dataset_name)

```

```

Download started
Download completed
Writing started
Writing completed
End

```

```

[22]: # Download and Save description
DatasetDownload(GPD_F_description_dataset_url, GPD_F_dataset_directory,
↳GPD_F_description_dataset_name)

```

```

Download started
Download completed
Writing started
Writing completed
End

```

1.3.2 GPD - Data import - Starting point

```

[7]: GPD_dataset = pd.read_csv(GPD_F_dataset_directory+GPD_F_dataset_name)

```

read_csv on dataset description rise an error of text decoding: *UnicodeDecodeError: 'utf-8' codec can't decode byte 0x96 in position 292: invalid start byte*

Let's check the encoding

```

[27]: with open(GPD_F_dataset_directory+GPD_F_description_dataset_name, 'rb') as file:
      print(chardet.detect(file.read()))

```

```

{'encoding': 'Windows-1252', 'confidence': 0.73, 'language': ''}

```

```

[28]: with open(GPD_F_dataset_directory+GPD_F_dataset_name, 'rb') as file:
      print(chardet.detect(file.read()))

```

```

{'encoding': 'ascii', 'confidence': 1.0, 'language': ''}

```

```
[29]: GPD_dataset_description = pd.
      ↪read_csv(GPD_F_dataset_directory+GPD_F_description_dataset_name,
      ↪encoding='Windows-1252')
```

1.3.3 GPD-F - Exploration

```
[31]: GPD_dataset.describe()
```

```
[31]:      Unnamed: 0      diameter      tongue      body
count  796.000000  474.000000  293.000000  633.000000
mean    398.500000   27.781814    7.291297   11.592891
std     229.929699   31.164702    4.009739    3.862993
min       1.000000    2.000000    2.000000    2.000000
25%     199.750000   12.200000    5.000000    9.000000
50%     398.500000   25.000000    5.500000   11.500000
75%     597.250000   25.000000    9.000000   13.500000
max     796.000000  150.000000   26.400000   25.000000
```

So... seems we have to deal with a lot of missing values... yeah! XD

```
[33]: GPD_dataset.columns
```

```
[33]: Index(['Unnamed: 0', 'crop', 'type', 'season', 'diameter', 'corolla', 'colour',
        'nectar', 'b.system', 's.pollination', 'inflorescence', 'composite',
        'visitor', 'guild', 'tongue', 'body', 'sociality', 'feeding'],
        dtype='object')
```

```
[34]: GPD_dataset_description.describe()
```

```
[34]:      Unnamed: 0
count    15.000000
mean      8.000000
std       4.472136
min       1.000000
25%       4.500000
50%       8.000000
75%      11.500000
max      15.000000
```

```
[36]: GPD_dataset_description
```

```
[36]:      Unnamed: 0      Name      Group      Type      Unit \
0              1      type      Plant  discrete  levels
1              2      season      Plant  discrete  levels
2              3  diameter      Plant  continuous    mm
3              4    corolla      Plant  discrete  levels
4              5    colour      Plant  discrete  levels
```

5	6	nectar	Plant	discrete	levels
6	7	b.system	Plant	discrete	levels
7	8	s.pollination	Plant	discrete	levels
8	9	inflorescence	Plant	discrete	levels
9	10	composite	Plant	discrete	levels
10	11	guild	Pollinator	discrete	levels
11	12	tongue	Pollinator	continuous	mm
12	13	body	Pollinator	continuous	mm
13	14	sociality	Pollinator	discrete	levels
14	15	feeding	Pollinator	discrete	levels

	Description \
0	arboreous or herbaceous plant
1	Flower season: Describes the seasonal range. F...
2	Flower diameter
3	Flower corolla type
4	Flower colour
5	Whether flower contains nectar
6	Type of bloom system
7	Self pollination
8	Type of inflorescence
9	Whether flower is composite or not
10	Pollinator guild
11	Pollinator tongue length
12	Pollinator body length
13	Whether pollinator is sociality or not
14	Feeding behaviour

	Levels
0	arboreous, herbaceous
1	sprisum, summer, spriaut, spring, autspri, sum...
2	NaN
3	campanulate open, tubular
4	white, yellow, purple, pink, green, blue, red
5	yes, no
6	insects, insects/bats, insects/bats, insects/b...
7	yes, no
8	solitary, solitary/clusters, solitary/pairs, yes
9	yes, no
10	andrenidae, bumblebees, butterflies, coleopter...
11	NaN
12	NaN
13	yes, no
14	oligolectic, parasitic, polylectic

```
[37]: GPD_dataset.head()
```



```
[37]: Unnamed: 0      crop      type      season      diameter \
0      1  Vaccinium_corymbosum  arboreous  sprisum      NaN
1      2  Vaccinium_corymbosum  arboreous  sprisum      NaN
2      3      Brassica_napus  herbaceous  summer      12.5
3      4      Brassica_napus  herbaceous  summer      12.5
4      5      Brassica_napus  herbaceous  summer      12.5

      corolla  colour  nectar      b.system  s.pollination  inflorescence \
0  CAMPANULATE  white   yes      insects      no      yes
1  CAMPANULATE  white   yes      insects      no      yes
2      OPEN  yellow   yes  wind/insects      no      yes
3      OPEN  yellow   yes  wind/insects      no      yes
4      OPEN  yellow   yes  wind/insects      no      yes

      composite      visitor      guild  tongue  body  sociality \
0      no      Andrena_wilkella  ANDRENIDAE  NaN  10.5      no
1      no  Andrena_barbilabris  ANDRENIDAE  NaN  10.5      no
2      no  Andrena_cineraria  ANDRENIDAE  NaN  12.0      no
3      no  Andrena_flavipes  ANDRENIDAE  NaN  11.0      no
4      no  Andrena_gravida  ANDRENIDAE  NaN  13.0      no

      feeding
0  oligolectic
1  polylectic
2  polylectic
3  polylectic
4  polylectic
```

Maybe we can try some clustering technique on this dataset to find out some interesting relationship

Missing values Let's check how many missing values we have and somehow how are distributed

```
[38]: # Number of missing values per column
GPD_dataset.isnull().sum()
```

```
[38]: Unnamed: 0      0
crop      0
type      0
season    30
diameter  322
corolla   3
colour    5
nectar    29
b.system  0
s.pollination  0
inflorescence  0
composite  0
```

```

visitor          0
guild            0
tongue          503
body            163
sociality        32
feeding          51
dtype: int64

```

```

[39]: # Percentage of missing values per column
      GPD_dataset.isnull().sum()/len(GPD_dataset)*100

```

```

[39]: Unnamed: 0      0.000000
      crop          0.000000
      type          0.000000
      season        3.768844
      diameter      40.452261
      corolla        0.376884
      colour        0.628141
      nectar         3.643216
      b.system       0.000000
      s.pollination  0.000000
      inflorescence  0.000000
      composite      0.000000
      visitor        0.000000
      guild          0.000000
      tongue        63.190955
      body          20.477387
      sociality      4.020101
      feeding        6.407035
      dtype: float64

```

```

[64]: # Let's check rows
      # Let's try to select only rows with some missing values
      # Note that GPD_dataset.isnull().sum() is a pandas Series
      len(GPD_dataset.isnull().sum(axis=1)[~GPD_dataset.isnull().sum(axis=1).
      ↪isin([0])])

```

```

[64]: 662

```

```

[9]: # Clearly a lot of rows since only for toungue column we have 60% of missing.
      # Lets' check rows excluding the columns with a consistent number of missing
      ↪(tounge, diametere, body)
      # To make the code more readable let's make two steps
      GPD_dataset_subset = GPD_dataset.loc[:, ~GPD_dataset.columns.
      ↪isin(["tongue", "diameter", "body"])]

```

```
len(GPD_dataset_subset.isnull().sum(axis=1)[~GPD_dataset_subset.isnull().
↪sum(axis=1).isin([0])])
```

[9]: 132

```
[61]: # Let's chek how many have more than 1 missing
len(GPD_dataset_subset.isnull().sum(axis=1)[~GPD_dataset_subset.isnull().
↪sum(axis=1).isin([0,1])])
```

[61]: 17

So maybe we can try to make a first clusterization excluding this 17 rows and the 3 problematic columns.

```
[10]: GPD_dataset_subset = GPD_dataset_subset.drop(GPD_dataset_subset.isnull().
↪sum(axis=1)[~GPD_dataset_subset.isnull().sum(axis=1).isin([0,1])].index)
```

```
[70]: GPD_dataset_subset.describe()
```

```
[70]:      Unnamed: 0
count    779.000000
mean      395.503209
std       230.662477
min         1.000000
25%      195.500000
50%      392.000000
75%      594.500000
max       796.000000
```

```
[71]: GPD_dataset_subset.describe
```

```
[71]: <bound method NDFrame.describe of      Unnamed: 0      crop
type  season      corolla \
0         1  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE
1         2  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE
2         3      Brassica_napus  herbaceous  summer    OPEN
3         4      Brassica_napus  herbaceous  summer    OPEN
4         5      Brassica_napus  herbaceous  summer    OPEN
..      ...      ...      ...      ...      ...
791      792    Allium_oleraceum  herbaceous  summer  CAMPANULATE
792      793    Jatropha_curcas  arboreous  spriaut    OPEN
793      794    Malus_domestica  arboreous  spring    OPEN
794      795  Phaseolus_coccineus  herbaceous  summer    OPEN
795      796    Capparis_spinosa  arboreous  summer    OPEN

      colour nectar      b.system s.pollination inflorescence composite \
0      white   yes      insects          no          yes          no
```

1	white	yes	insects	no	yes	no
2	yellow	yes	wind/insects	no	yes	no
3	yellow	yes	wind/insects	no	yes	no
4	yellow	yes	wind/insects	no	yes	no
..
791	purple	yes	insects	no	yes	no
792	green	yes	insects	no	yes	no
793	white	yes	insects	no	yes	no
794	white	yes	insects	no	yes	no
795	white	yes	insects	no	solitary	no

	visitor	guild	sociality	feeding
0	Andrena_wilkella	ANDRENIDAE	no	oligolectic
1	Andrena_barbilabris	ANDRENIDAE	no	polylectic
2	Andrena_cineraria	ANDRENIDAE	no	polylectic
3	Andrena_flavipes	ANDRENIDAE	no	polylectic
4	Andrena_gravida	ANDRENIDAE	no	polylectic
..
791	Dolichovespula_saxonica	WASPS	yes	polylectic
792	Bembecinus_tridens	WASPS	no	NaN
793	Vespula_vulgaris	WASPS	yes	polylectic
794	Philanthus_triangulum	WASPS	no	polylectic
795	Bembecinus_tridens	WASPS	no	NaN

[779 rows x 15 columns]>

```
[72]: # Percentage of missing values per column
GPD_dataset_subset.isnull().sum()/len(GPD_dataset_subset)*100
```

```
[72]: Unnamed: 0      0.000000
crop             0.000000
type             0.000000
season           2.952503
corolla          0.000000
colour           0.641849
nectar           2.824134
b.system         0.000000
s.pollination    0.000000
inflorescence    0.000000
composite        0.000000
visitor          0.000000
guild            0.000000
sociality        3.209243
feeding          5.134788
dtype: float64
```

We have no way to infer the values of blooming season, flowers colour, nectar presence, sociality or

feeding (I mean no way before the analysis of the dataset and the application of ML algorithms). So for the moment let's add a fixed value "undefined" for the missing.

```
[11]: imput_undefined = SimpleImputer(strategy = 'constant', fill_value =
      ↪ 'undefined')
      GPD_dataset_subset_0missing_array = imput_undefined.
      ↪ fit_transform(GPD_dataset_subset)
      # Note that SimpleImputer returns a numpy array
```

```
[12]: GPD_dataset_subset_ONaN = pd.DataFrame(GPD_dataset_subset_0missing_array,
      ↪ columns = GPD_dataset_subset.columns)
```

```
[13]: GPD_dataset_subset_ONaN.isnull().sum()
```

```
[13]: Unnamed: 0      0
      crop          0
      type          0
      season        0
      corolla        0
      colour         0
      nectar         0
      b.system       0
      s.pollination  0
      inflorescence  0
      composite      0
      visitor        0
      guild          0
      sociality      0
      feeding        0
      dtype: int64
```

Let's save the new dataset

```
[14]: GPD_dataset_subset_ONaN.to_pickle(GPD_F_dataset_directory+"GPD_F_subset_ONaN.
      ↪ pkl")
```

1.4 GPD-F - Post missing cleaning - Starting point

```
[6]: GPD_dataset_subset_ONaN = pd.
      ↪ read_pickle(GPD_F_dataset_directory+"GPD_F_subset_ONaN.pkl")
```

```
[16]: GPD_dataset_subset_ONaN.describe
```

```
[16]: <bound method NDFrame.describe of      Unnamed: 0      crop
      type  season      corolla \
      0      1  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE
      1      2  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE
```

2	3	Brassica_napus	herbaceous	summer	OPEN
3	4	Brassica_napus	herbaceous	summer	OPEN
4	5	Brassica_napus	herbaceous	summer	OPEN
..
774	792	Allium_oleraceum	herbaceous	summer	CAMPANULATE
775	793	Jatropha_curcas	arboreous	spriaut	OPEN
776	794	Malus_domestica	arboreous	spring	OPEN
777	795	Phaseolus_coccineus	herbaceous	summer	OPEN
778	796	Capparis_spinosa	arboreous	summer	OPEN

	colour	nectar	b.system	s.pollination	inflorescence	composite	\
0	white	yes	insects	no	yes	no	
1	white	yes	insects	no	yes	no	
2	yellow	yes	wind/insects	no	yes	no	
3	yellow	yes	wind/insects	no	yes	no	
4	yellow	yes	wind/insects	no	yes	no	
..	
774	purple	yes	insects	no	yes	no	
775	green	yes	insects	no	yes	no	
776	white	yes	insects	no	yes	no	
777	white	yes	insects	no	yes	no	
778	white	yes	insects	no	solitary	no	

	visitor	guild	sociality	feeding
0	Andrena_wilkella	ANDRENIDAE	no	oligolectic
1	Andrena_barbilabris	ANDRENIDAE	no	polylectic
2	Andrena_cineraria	ANDRENIDAE	no	polylectic
3	Andrena_flavipes	ANDRENIDAE	no	polylectic
4	Andrena_gravida	ANDRENIDAE	no	polylectic
..
774	Dolichovespula_saxonica	WASPS	yes	polylectic
775	Bembecinus_tridens	WASPS	no	undefined
776	Vespula_vulgaris	WASPS	yes	polylectic
777	Philanthus_triangulum	WASPS	no	polylectic
778	Bembecinus_tridens	WASPS	no	undefined

[779 rows x 15 columns]>

```
[17]: GPD_dataset_subset_0NaN.isnull().sum()
```

```
[17]: Unnamed: 0      0
      crop          0
      type          0
      season        0
      corolla       0
      colour        0
      nectar        0
```

```

b.system      0
s.pollination 0
inflorescence 0
composite     0
visitor       0
guild         0
sociality     0
feeding       0
dtype: int64

```

Most of the columns are categorical, let's check if we have also some numerical data

```

[44]: for index, column in enumerate(GPD_dataset_subset_0NaN.columns.tolist()[1:]):
        if str(GPD_dataset_subset_0NaN.iloc[1,index+1]).isnumeric():
            print(column)

```

So we have only categorical data.

```

[61]: GPD_dataset_subset_0NaN.dtypes

```

```

[61]: Unnamed: 0      object
crop                object
type               object
season             object
corolla            object
colour             object
nectar             object
b.system           object
s.pollination      object
inflorescence      object
composite          object
visitor            object
guild              object
sociality          object
feeding            object
dtype: object

```

But actually are stored as mixed columns values, so let's remove first column which we are not interested in and convert all the others column in categorical pandas's data type

```

[7]: GPD_dataset_subset2_0NaN = GPD_dataset_subset_0NaN.iloc[:,1:]

```

```

[65]: for column in GPD_dataset_subset2_0NaN.columns.tolist():
        GPD_dataset_subset2_0NaN[column] = GPD_dataset_subset2_0NaN.loc[column].
        ↪astype('category')

```

Input In [65]

```
GPD_dataset_subset2_0NaN.loc[:,column] = GPD_dataset_subset2_0NaN.  
↳loc[:,column].astype('category')
```

```
SyntaxError: invalid syntax
```

```
[67]: GPD_dataset_subset2_0NaN.dtypes
```

```
[67]: crop          category  
type            category  
season          category  
corolla         category  
colour          category  
nectar          category  
b.system        category  
s.pollination   category  
inflorescence   category  
composite       category  
visitor         category  
guild           category  
sociality       category  
feeding         category  
dtype: object
```

```
[66]: GPD_dataset_subset2_0NaN.describe
```

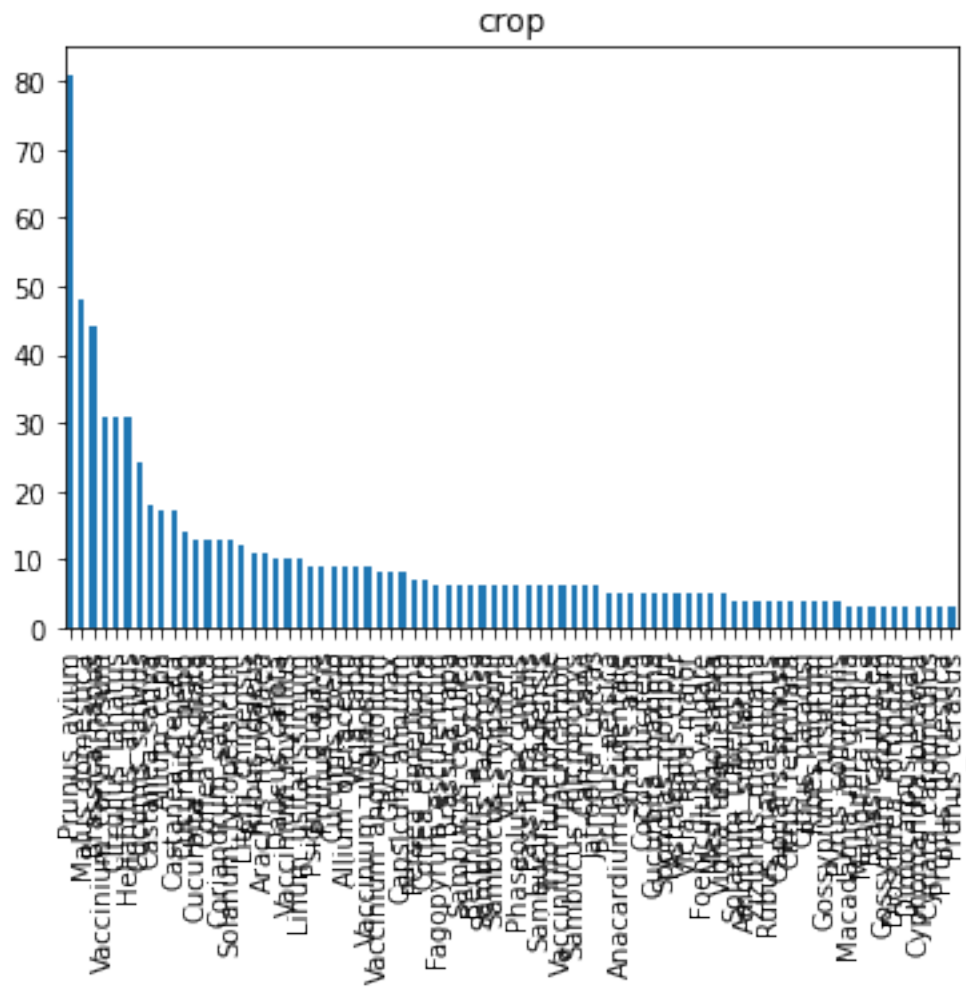
```
[66]: <bound method NDFrame.describe of  
corolla  colour  nectar  \  
0  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE  white  yes  
1  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE  white  yes  
2      Brassica_napus  herbaceous  summer  OPEN  yellow  yes  
3      Brassica_napus  herbaceous  summer  OPEN  yellow  yes  
4      Brassica_napus  herbaceous  summer  OPEN  yellow  yes  
..      ...      ...      ...      ...      ...  
774  Allium_oleraceum  herbaceous  summer  CAMPANULATE  purple  yes  
775  Jatropha_curcas  arboreous  spriaut  OPEN  green  yes  
776  Malus_domestica  arboreous  spring  OPEN  white  yes  
777  Phaseolus_coccineus  herbaceous  summer  OPEN  white  yes  
778  Capparis_spinosa  arboreous  summer  OPEN  white  yes  
  
b.system  s.pollination  inflorescence  composite  \  
0  insects  no  yes  no  
1  insects  no  yes  no  
2  wind/insects  no  yes  no  
3  wind/insects  no  yes  no  
4  wind/insects  no  yes  no  
..      ...      ...      ...      ...
```

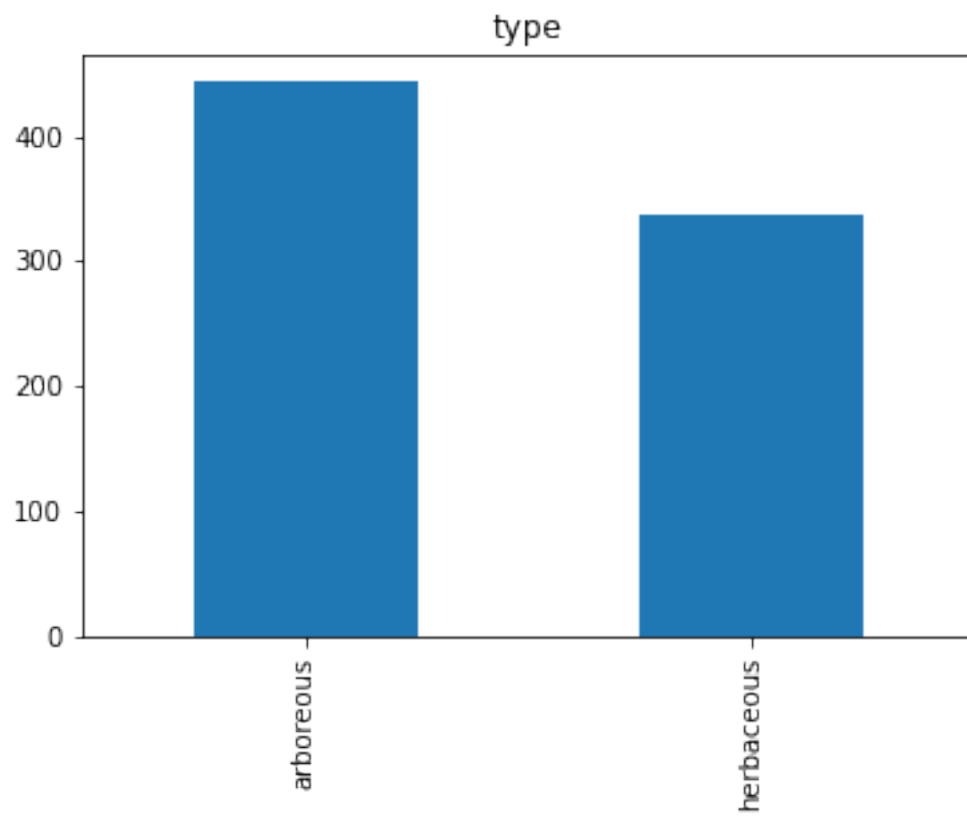

774	insects	no	yes	no
775	insects	no	yes	no
776	insects	no	yes	no
777	insects	no	yes	no
778	insects	no	solitary	no

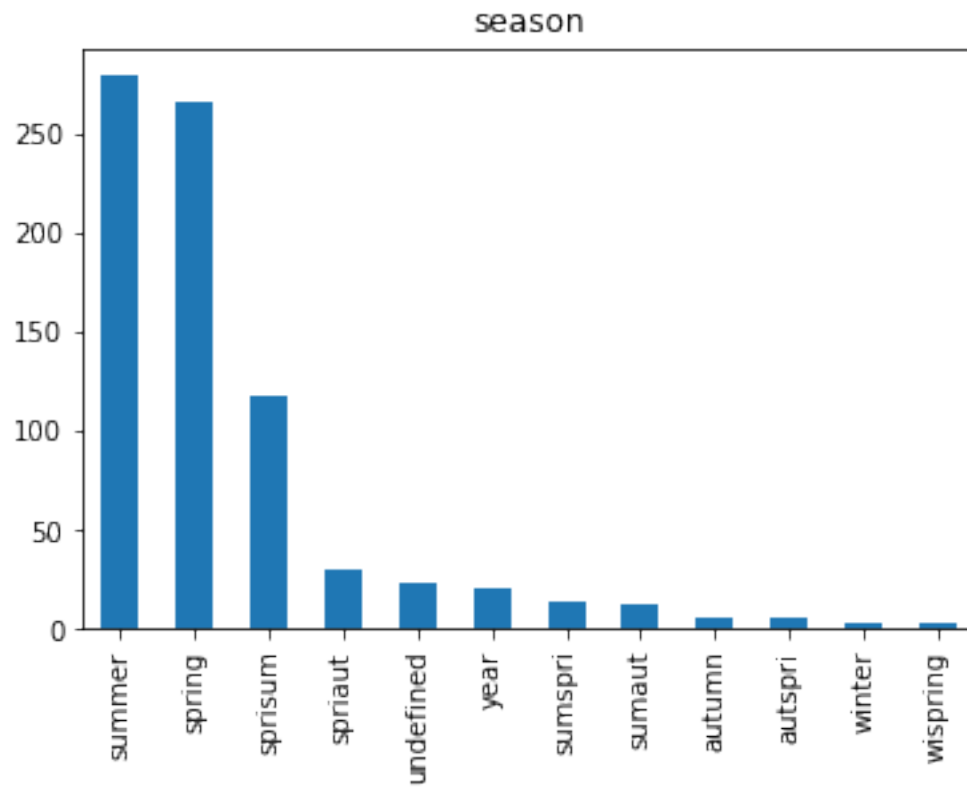
	visitor	guild	sociality	feeding
0	Andrena_wilkella	ANDRENIDAE	no	oligolectic
1	Andrena_barbilabris	ANDRENIDAE	no	polylectic
2	Andrena_cineraria	ANDRENIDAE	no	polylectic
3	Andrena_flavipes	ANDRENIDAE	no	polylectic
4	Andrena_gravida	ANDRENIDAE	no	polylectic
..
774	Dolichovespula_saxonica	WASPS	yes	polylectic
775	Bembecinus_tridens	WASPS	no	undefined
776	Vespula_vulgaris	WASPS	yes	polylectic
777	Philanthus_triangulum	WASPS	no	polylectic
778	Bembecinus_tridens	WASPS	no	undefined

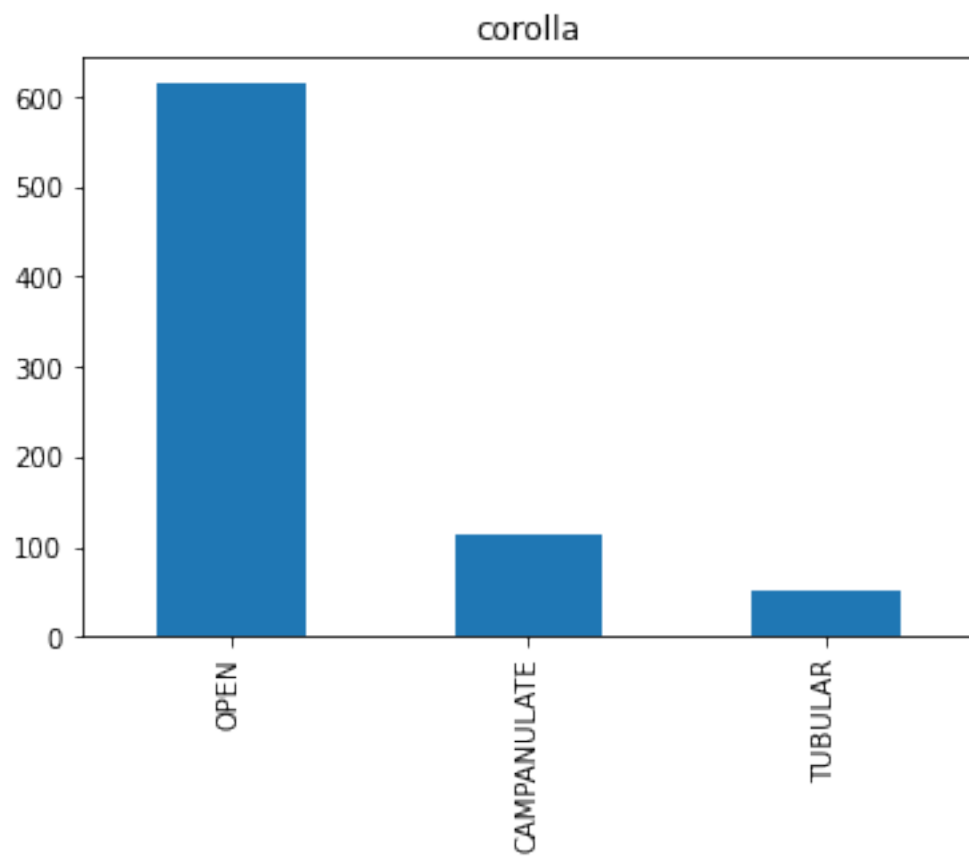
[779 rows x 14 columns]>

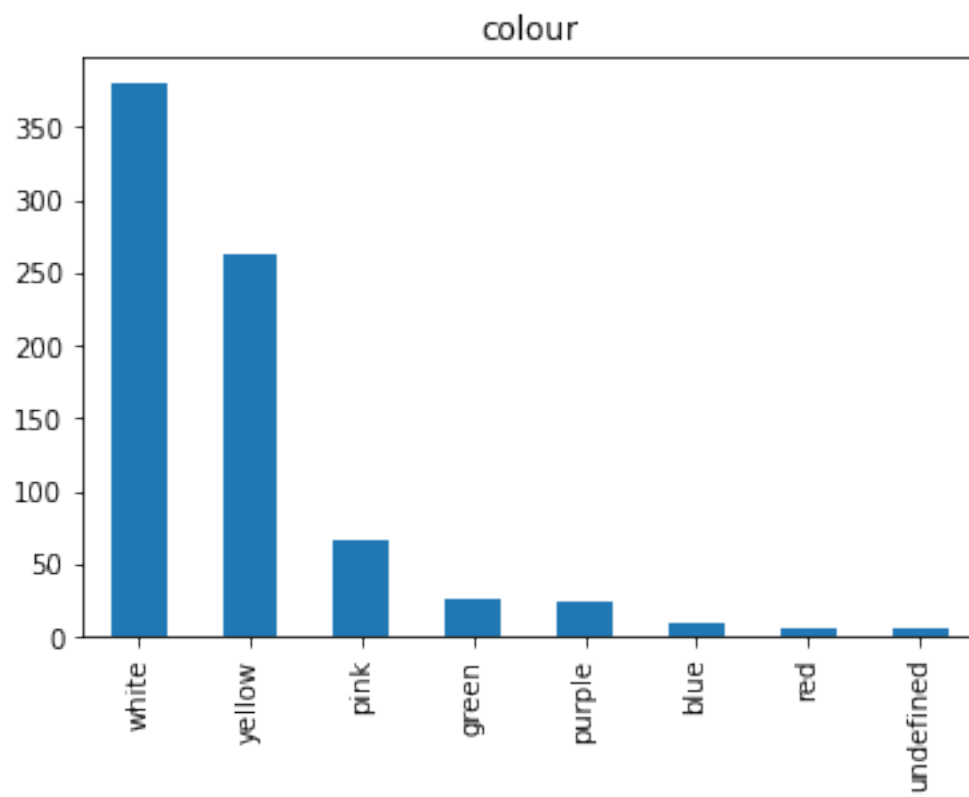
```
[68]: for column in GPD_dataset_subset2_0NaN.columns.tolist():
plt.pyplot.figure()
plt.pyplot.title(column)
GPD_dataset_subset2_0NaN[column].value_counts().plot(kind = 'bar')
```

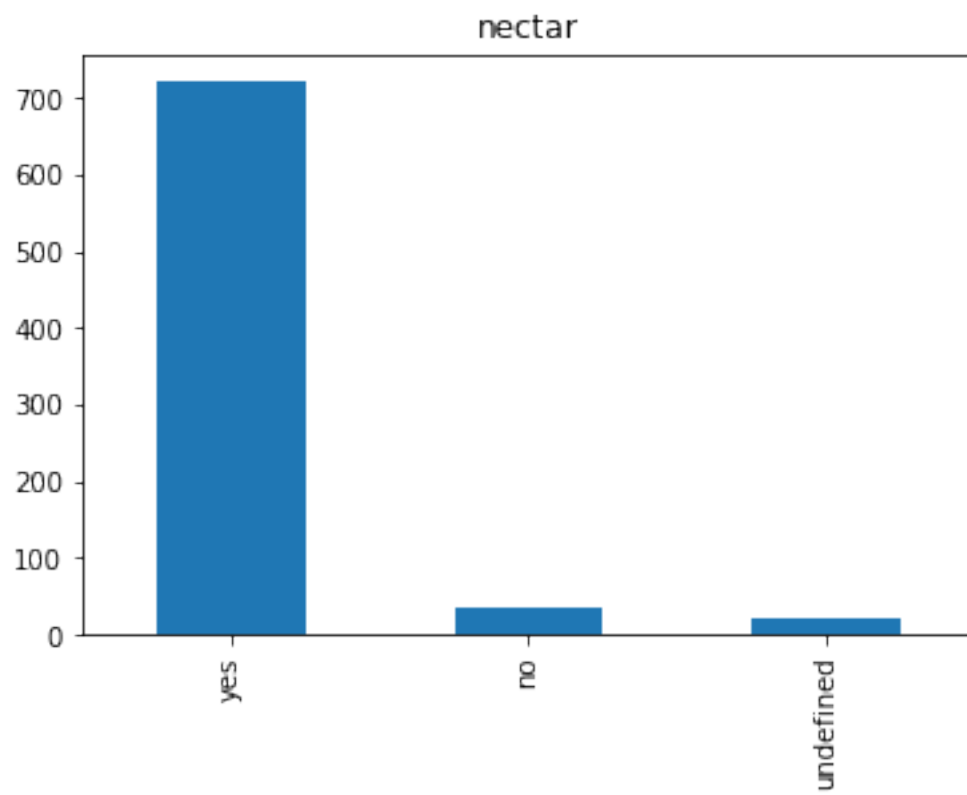


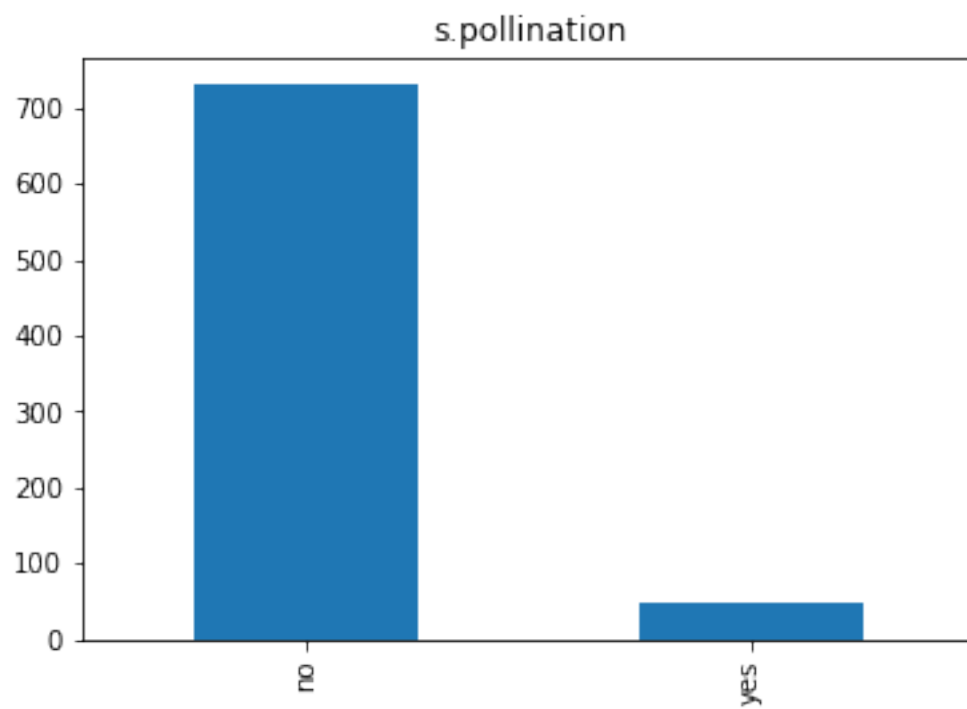
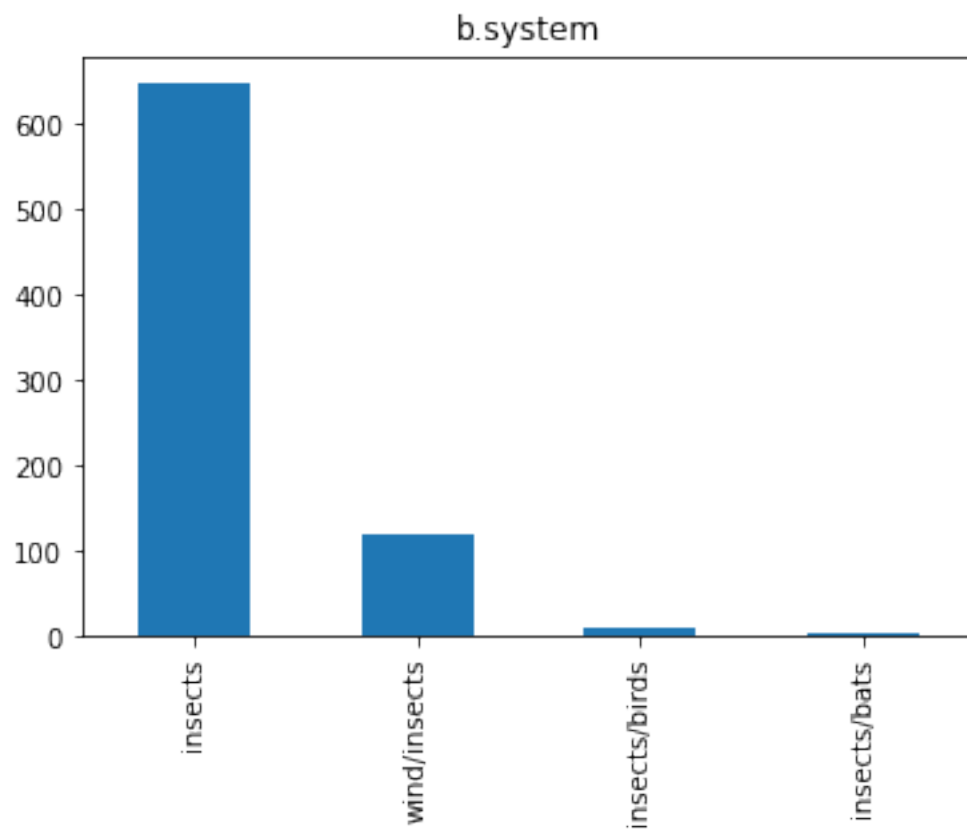


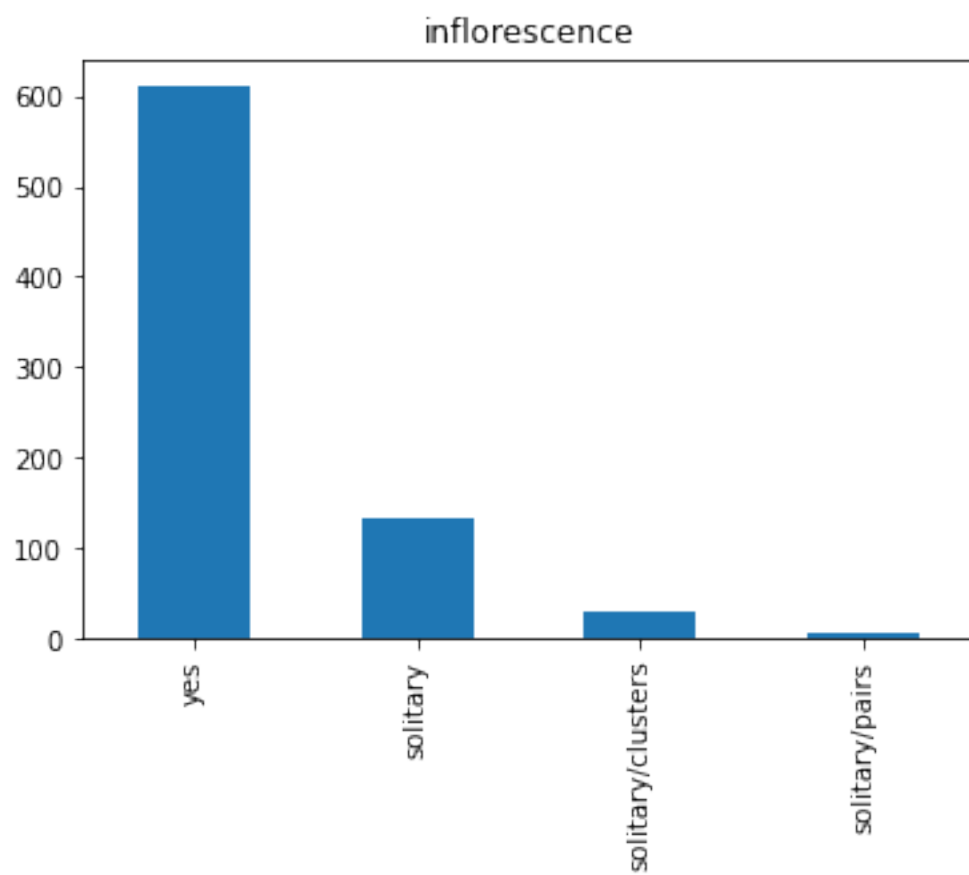


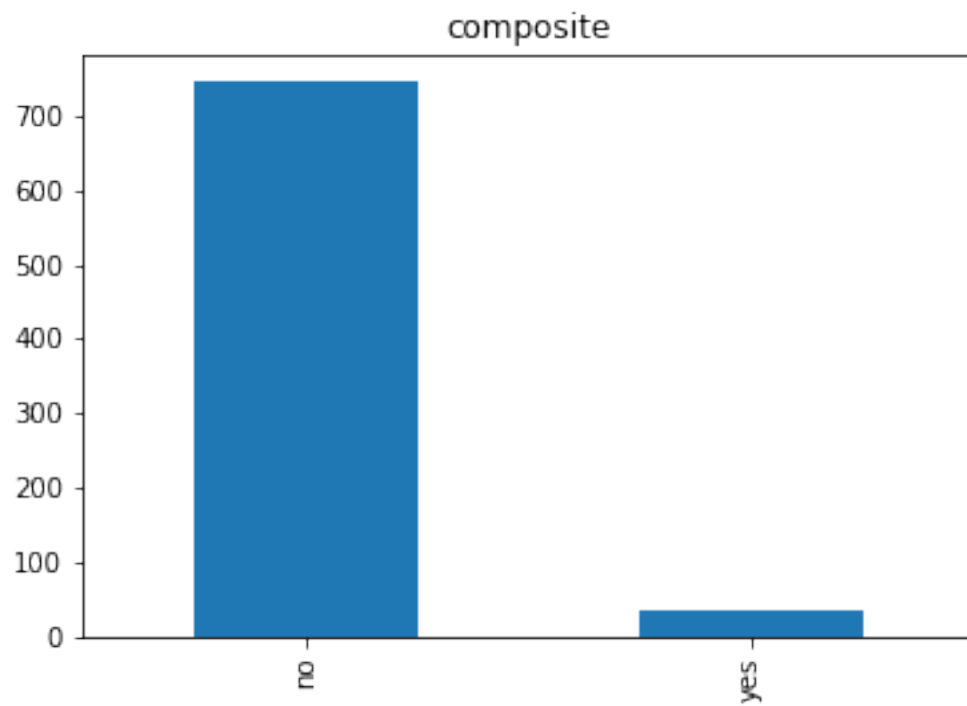


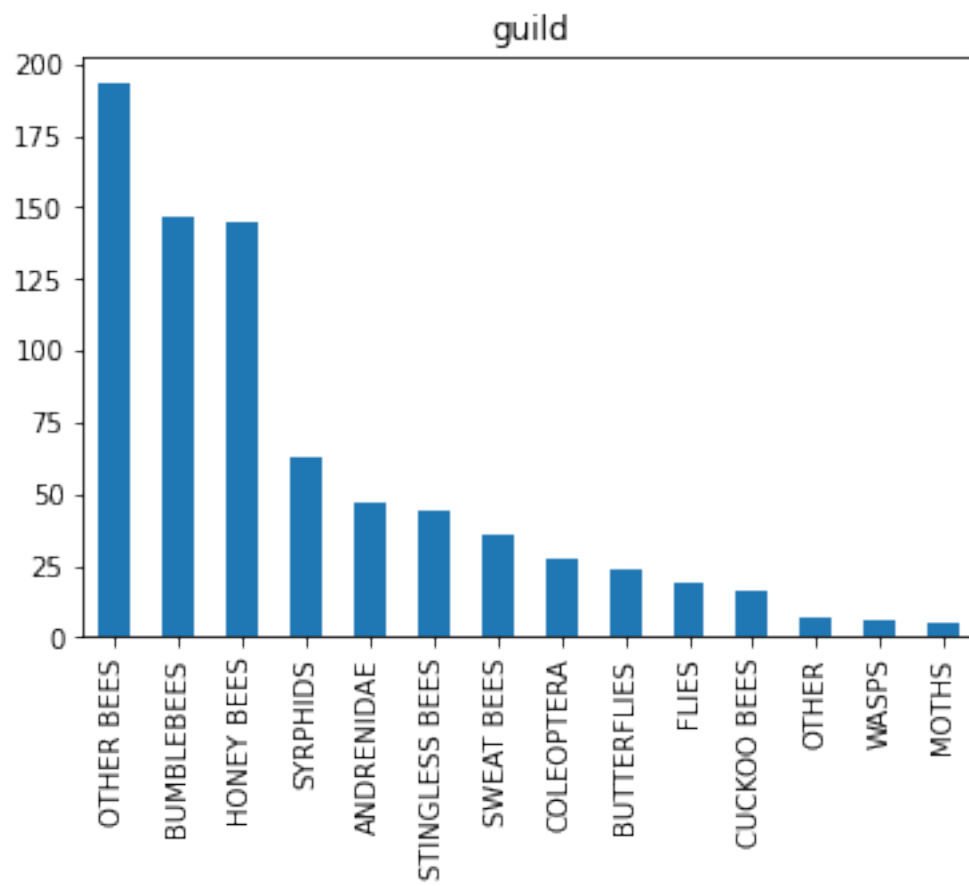


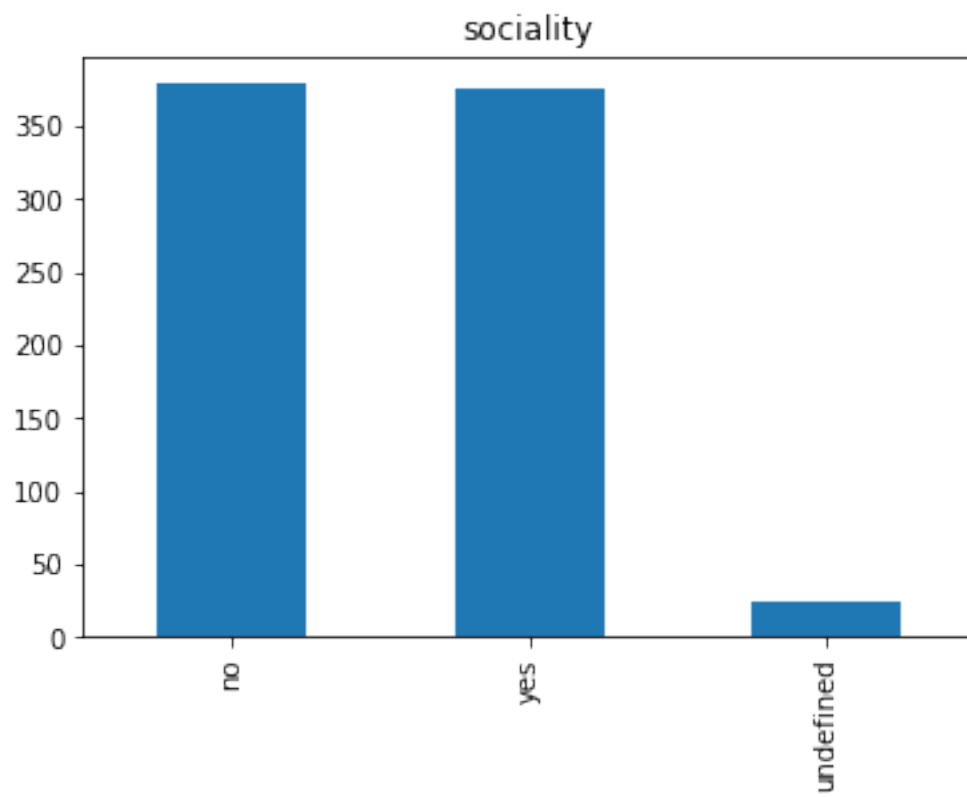


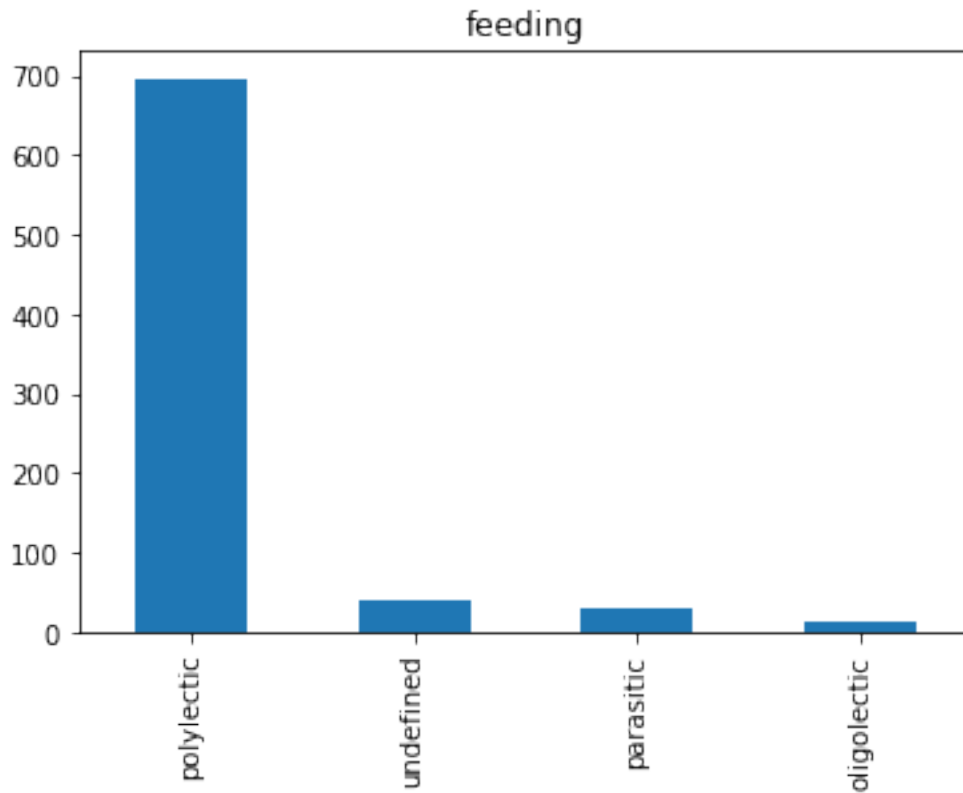












We can use Cramer's V correlation value to present a heatmap of correlation between these categorical variables.

Unfortunately this metric seems a bit biased for “large” number of variables ([Bergsma, Wicher. \(2013\). A bias-correction for Cramér's V and Tschuprow's T. Journal of the Korean Statistical Society. 42. 10.1016/j.jkss.2012.10.002.](#)).

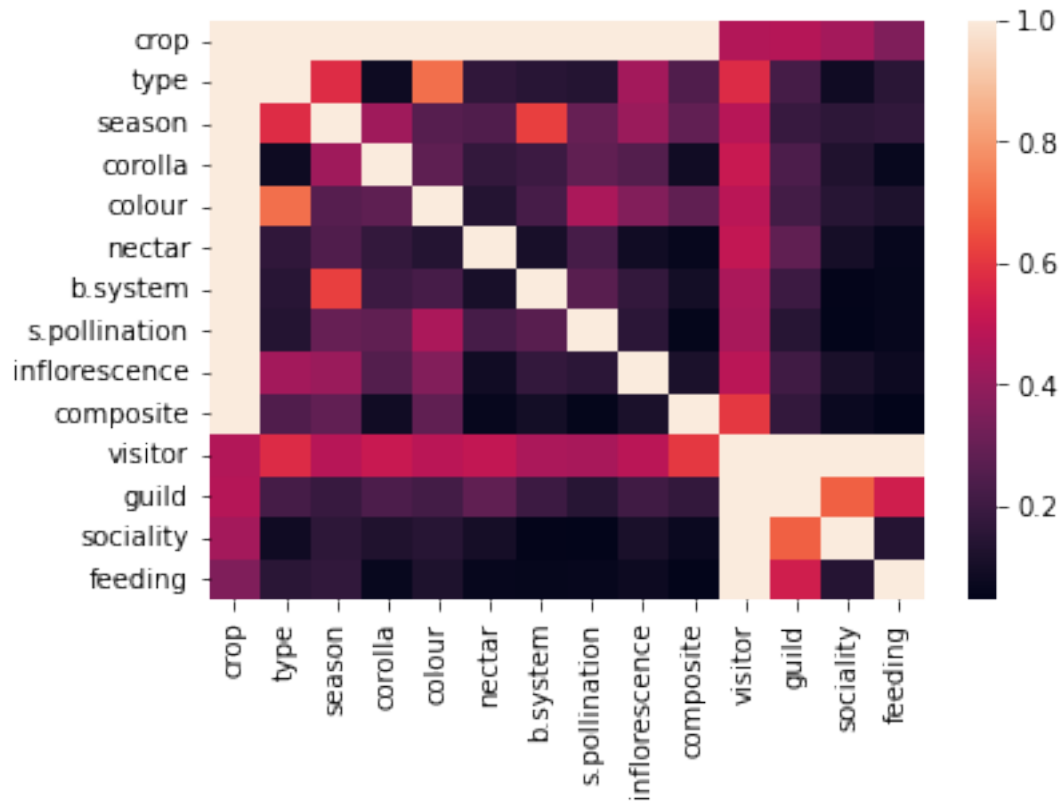
For the moment let's apply Cramer's V in a future we will improve the implementation with the bias correction.

```
[69]: CramersV_GPD_subset_object = am.CramersV(GPD_dataset_subset2_0NaN)
```

```
[70]: CramersV_GPD_subset_matrix = CramersV_GPD_subset_object.fit()
```

```
[71]: sb.heatmap(CramersV_GPD_subset_matrix)
```

```
[71]: <AxesSubplot:>
```

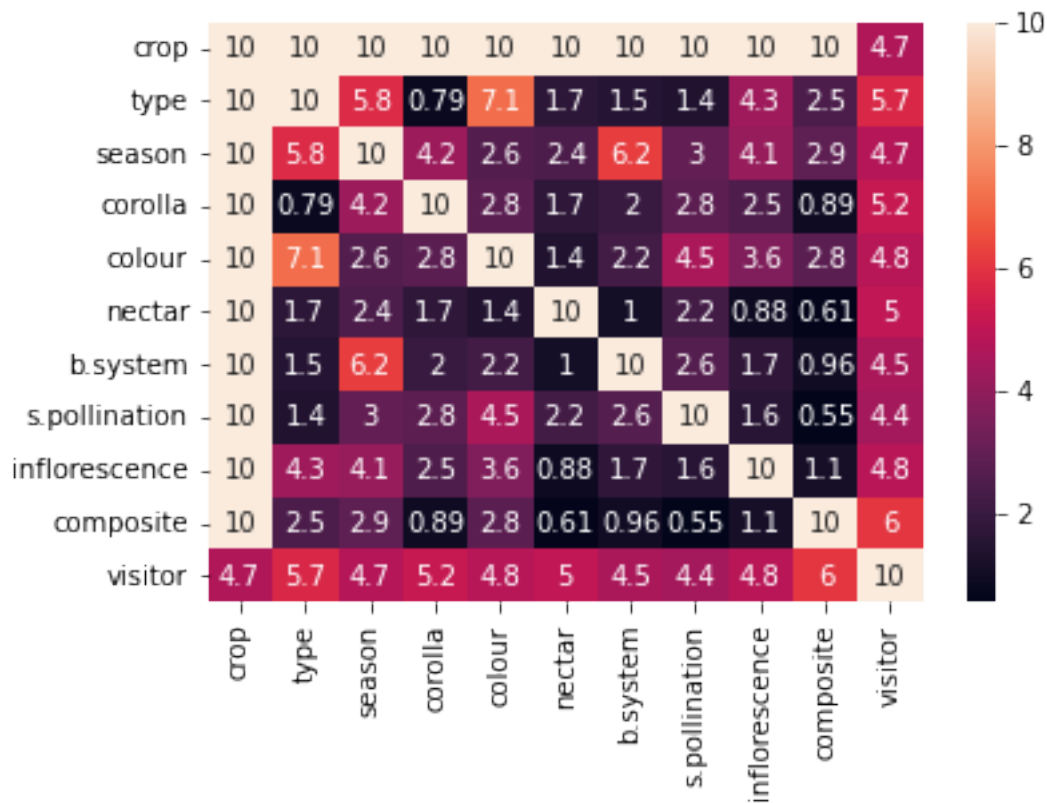


As we could expect we have an evident separation of correlation between plants and bees where the crop is highly correlated with the information about the plants characteristics; the guild is highly related with the pollinators characteristics and the “visitor” variable is the link between the two groups.

Let's focus on the two groups

```
[89]: sb.heatmap(CramersV_GPD_subset_matrix.iloc[:11,:11]*10, annot=True)
      # since we know that values are between 0 and 1 we multiply for 10 to avoid most
      ↳ of unusefull "0."
```

```
[89]: <AxesSubplot:>
```

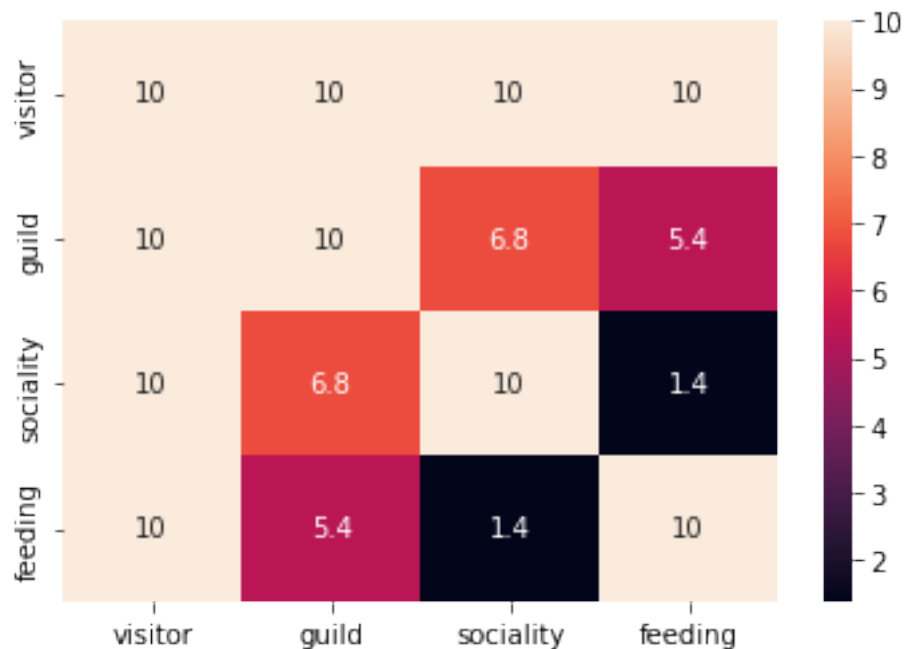


We can see that type (arboreous or heraceous) seems highly related to the flower colour and also quite related with the season.

The bloom system (bytheway from the values seems more a “pollination type”) seems highly related with the flower season. Despite that, the bloom system seems not related with the flower colour and the plant type.

```
[92]: sb.heatmap(CramersV_GPD_subset_matrix.iloc[10:,10:]*10, annot=True)
```

```
[92]: <AxesSubplot:>
```

Quite self-explanatory

Let's have a closer look at the cited plants variables

Multi-categorical plot First of all let's encode the desired variable with numeric values.

For the visualization we can have an advantage encoding with an order even if the variables that we are considering don't have a natural order.

```
[93]: GPD_dataset_subset2_ONaN.columns
```

```
[93]: Index(['crop', 'type', 'season', 'corolla', 'colour', 'nectar', 'b.system',
          's.pollination', 'inflorescence', 'composite', 'visitor', 'guild',
          'sociality', 'feeding'],
          dtype='object')
```

```
[99]: type_encoder = LabelEncoder()
      type_encoder.fit(GPD_dataset_subset2_ONaN.loc[:, 'type'])
      type_encoder.classes_
```

```
[99]: array(['arboreous', 'herbaceous'], dtype=object)
```

```
[100]: colour_encoder = LabelEncoder()
       colour_encoder.fit(GPD_dataset_subset2_ONaN.loc[:, 'colour'])
       colour_encoder.classes_
```

```
[100]: array(['blue', 'green', 'pink', 'purple', 'red', 'undefined', 'white',  
          'yellow'], dtype=object)
```

```
[157]: # let's transform "undefined" in "gray"  
undefined_gray = SimpleImputer(missing_values = 'undefined', strategy =  
    ↪ 'constant', \n  
                                fill_value = 'gray')  
  
gray_column_array = undefined_gray.fit_transform( GPD_dataset_subset2_0NaN.  
    ↪ loc[:, 'colour'].to_numpy().reshape(-1,1) )  
  
GPD_dataset_subset2_0NaN.loc[:, 'colour'] = gray_column_array.reshape(-1,1)
```

/tmp/ipykernel_34953/2547033298.py:7: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
GPD_dataset_subset2_0NaN.loc[:, 'colour'] = gray_column_array.reshape(-1,1)
```

```
[158]: colour_encoder = LabelEncoder()  
colour_encoder.fit(GPD_dataset_subset2_0NaN.loc[:, 'colour'])  
colour_encoder.classes_
```

```
[158]: array(['blue', 'gray', 'green', 'pink', 'purple', 'red', 'white',  
          'yellow'], dtype=object)
```

```
[101]: season_encoder = LabelEncoder()  
season_encoder.fit(GPD_dataset_subset2_0NaN.loc[:, 'season'])  
season_encoder.classes_
```

```
[101]: array(['autspri', 'autumn', 'spriaut', 'spring', 'sprisum', 'sumaut',  
          'summer', 'sumspri', 'undefined', 'winter', 'wispring', 'year'],  
          dtype=object)
```

```
[103]: s_pollination_encoder = LabelEncoder()  
s_pollination_encoder.fit(GPD_dataset_subset2_0NaN.loc[:, 's.pollination'])  
s_pollination_encoder.classes_
```

```
[103]: array(['no', 'yes'], dtype=object)
```

```
[105]: guild_encoder = LabelEncoder()  
guild_encoder.fit(GPD_dataset_subset2_0NaN.loc[:, 'guild'])  
guild_encoder.classes_
```

```
[105]: array(['ANDRENIDAE', 'BUMBLEBEES', 'BUTTERFLIES', 'COLEOPTERA',
             'CUCKOO BEES', 'FLIES', 'HONEY BEES', 'MOTHS', 'OTHER',
             'OTHER BEES', 'STINGLESS BEES', 'SWEAT BEES', 'SYRPHIDS', 'WASPS'],
           dtype=object)
```

We want to use symbols to represent “guild”, so duble encode it

```
[664]: guild_mark_list_
        ↪=['o','v','<','_','3','s','p','*','|','x','d','$\\Omega$', '$\\xi$', '$\\aleph$']
guild_mark_encoder = LabelEncoder()
guild_mark_encoder.fit(guild_mark_list)
guild_mark_encoder.classes_
```

```
[664]: array(['$\\Omega$', '$\\aleph$', '$\\xi$', '*', '3', '<', '_', 'd', 'o',
             'p', 's', 'v', 'x', '|'], dtype='<U8')
```

```
[632]: guild_encoder.transform( guild_encoder.classes_ )
```

```
[632]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13])
```

```
[119]: guild_mark_encoder.transform( guild_mark_encoder.classes_ )
```

```
[119]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13])
```

```
[124]: type(guild_encoder.transform( guild_encoder.classes_ )[0] )
```

```
[124]: numpy.int64
```

```
[665]: guild_mark_legend = dict(zip(guild_encoder.classes_ , \
                                   guild_mark_encoder.inverse_transform( \
                                   guild_encoder.transform( guild_encoder.classes_
                                   ↪) ) ) )

guild_mark_legend
```

```
[665]: {'ANDRENIDAE': '$\\Omega$',
        'BUMBLEBEES': '$\\aleph$',
        'BUTTERFLIES': '$\\xi$',
        'COLEOPTERA': '*',
        'CUCKOO BEES': '3',
        'FLIES': '<',
        'HONEY BEES': '_',
        'MOTHS': 'd',
        'OTHER': 'o',
        'OTHER BEES': 'p',
        'STINGLESS BEES': 's',
        'SWEAT BEES': 'v',
```

```
'SYRPHIDS': 'x',
'WASPS': '|']
```

[]:

we should to add some noise to limitate points overlapping and maybe reshape on higher values, "or use size to plot less points but add the information of the number of points with that value combination. Maybe both

```
[666]: #let's convert colours in matplotlib colour values
colours_list = []
for color_data in GPD_dataset_subset2_ONaN.loc[:, 'colour']:
    colours_list.append(plt.colors.CSS4_COLORS[color_data])

GPD_dataset_subset2_ONaN_T = GPD_dataset_subset2_ONaN[['type', 'season', 's.
    ↪pollination']].copy()
GPD_dataset_subset2_ONaN_T.loc[:, 'type'] = type_encoder.transform (
    ↪GPD_dataset_subset2_ONaN.loc[:, 'type'] )
GPD_dataset_subset2_ONaN_T.loc[:, 'season'] = season_encoder.transform (
    ↪GPD_dataset_subset2_ONaN.loc[:, 'season'] )
GPD_dataset_subset2_ONaN_T.loc[:, 's.pollination'] = s_pollination_encoder.
    ↪transform ( GPD_dataset_subset2_ONaN.loc[:, 's.pollination'] )
GPD_dataset_subset2_ONaN_T.loc[:, 'guild'] = guild_mark_encoder.
    ↪inverse_transform( \
                                guild_encoder.transform(
    ↪GPD_dataset_subset2_ONaN.loc[:, 'guild'] ))

#let's add some noise
random.seed(6)
```

[431]: GPD_dataset_subset2_ONaN_T.describe()

```
[431]:
```

	type	season	s.pollination
count	779.000000	779.000000	779.000000
mean	0.431322	4.668806	0.062901
std	0.495579	1.965170	0.242941
min	0.000000	0.000000	0.000000
25%	0.000000	3.000000	0.000000
50%	0.000000	4.000000	0.000000
75%	1.000000	6.000000	0.000000
max	1.000000	11.000000	1.000000

[669]: GPD_dataset_subset2_ONaN_T.describe

```
[669]: <bound method NDFrame.describe of          type  season  s.pollination    guild
0      0      4      0  $\Omega$
1      0      4      0  $\Omega$
2      1      6      0  $\Omega$
3      1      6      0  $\Omega$
4      1      6      0  $\Omega$
..    ...    ...    ...    ...
774    1      6      0      |
775    0      2      0      |
776    0      3      0      |
777    1      6      0      |
778    0      6      0      |

[779 rows x 4 columns]>
```

```
[668]: fig = plt.pyplot.figure(dpi = 500)
ax = fig.add_subplot( projection = '3d')

ax.set_xlim(0,6)
ax.set_ylim(0,13)
ax.set_zlim(0,6)
ax.set_xticklabels([type_encoder.inverse_transform([0])[0], '', '', '', '', \
                    type_encoder.inverse_transform([1])[0]])
ax.set_yticklabels(season_encoder.
    ↪inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11]))
ax.set_zticklabels([s_pollination_encoder.inverse_transform([0])[0], '', '', ↪
    ↪'', '', \
                    s_pollination_encoder.inverse_transform([1])[0]])
ax.set_facecolor('#6b8e95')

for i, ind in enumerate(GPD_dataset_subset2_0NaN_T.index):
    ax.scatter(xs = GPD_dataset_subset2_0NaN_T.loc[ind, 'type']*5 + random.
    ↪randrange(0, 1000, )/3000 , \
               ys = GPD_dataset_subset2_0NaN_T.loc[ind, 'season'] + random.
    ↪randrange(0, 1000)/3000 , \
               zs = GPD_dataset_subset2_0NaN_T.loc[ind, 's.pollination']*5 + ↪
    ↪random.randrange(0, 1000)/3000 , \
               c = colours_list[i],
               sizes = [1.25], # markers dimension
               marker = GPD_dataset_subset2_0NaN_T.loc[ind, 'guild'])

plt.pyplot.show()
```

/tmp/ipykernel_34953/1936832168.py:7: UserWarning:

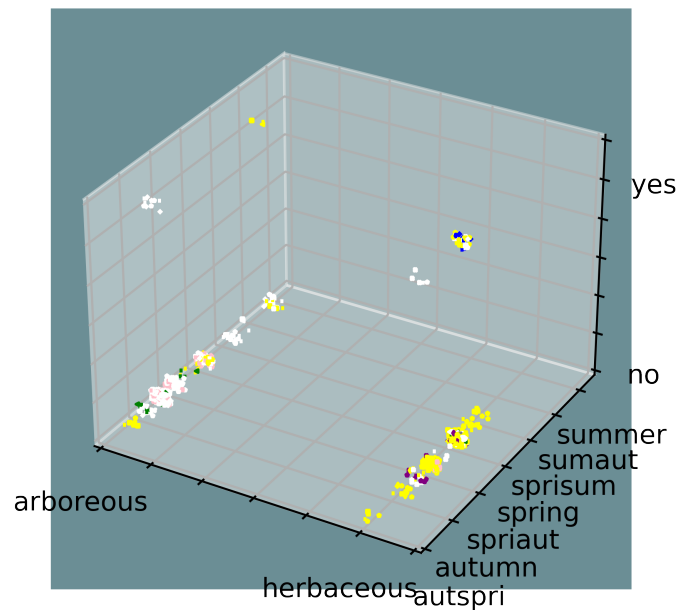
FixedFormatter should only be used together with FixedLocator

/tmp/ipykernel_34953/1936832168.py:9: UserWarning:

FixedFormatter should only be used together with FixedLocator

/tmp/ipykernel_34953/1936832168.py:10: UserWarning:

FixedFormatter should only be used together with FixedLocator



we need more space between season and bigger symbols

```
[730]: %matplotlib inline
fig = plt.pyplot.figure(dpi = 500, figsize=(4,4))
ax = fig.add_subplot( projection = '3d')

ax.set_xlim(0,3)
ax.set_ylim(0,245)
ax.set_zlim(0,3)
ax.set_xticklabels([type_encoder.inverse_transform([0])[0], '', '', '', '', \
                    type_encoder.inverse_transform([1])[0]])
ax.set_yticklabels(season_encoder.
    ↪inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11]))
ax.set_zticklabels([s_pollination_encoder.inverse_transform([0])[0], '', '', \
    ↪'', '', \
                    s_pollination_encoder.inverse_transform([1])[0]])
ax.set_facecolor('#6b8e95')
```

```

for i, ind in enumerate(GPD_dataset_subset2_0NaN_T.index):
    ax.scatter(xs = GPD_dataset_subset2_0NaN_T.loc[ind, 'type']*2 + random.
↳randrange(0, 1000, )/3000 , \
                ys = GPD_dataset_subset2_0NaN_T.loc[ind, 'season']*20 + random.
↳randrange(0, 1000)/3000 , \
                zs = GPD_dataset_subset2_0NaN_T.loc[ind, 's.pollination']*2 +
↳random.randrange(0, 1000)/3000 , \
                c = colours_list[i], \
                sizes = [10], # markers dimension \
                marker = GPD_dataset_subset2_0NaN_T.loc[ind, 'guild'], \
                label = '_nolegend_')

# set legend
for i in guild_mark_legend:
    ax.scatter(xs = -20 , \
                ys = -20 , \
                zs = -20 , \
                c = 'black', \
                sizes = [5], # markers dimension \
                marker = guild_mark_legend[i], \
                label = i)

fig.legend(bbox_to_anchor=(0,1,0.92,0), labelspaceing = 0.1, handletextpad = 0.
↳3, \
            columnspacing = 0.2, fontsize = 'xx-small', ncol = 4 )
plt.pyplot.show()

```

/tmp/ipykernel_34953/579580737.py:8: UserWarning:

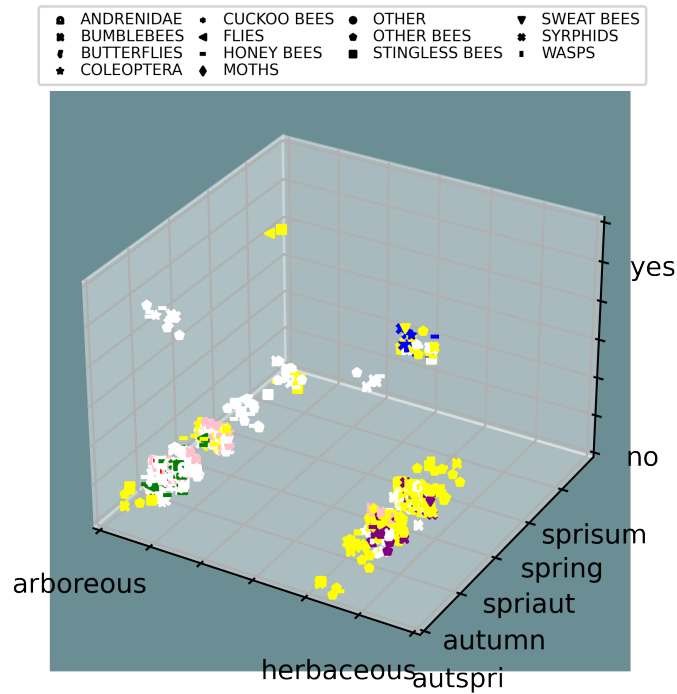
FixedFormatter should only be used together with FixedLocator

/tmp/ipykernel_34953/579580737.py:10: UserWarning:

FixedFormatter should only be used together with FixedLocator

/tmp/ipykernel_34953/579580737.py:11: UserWarning:

FixedFormatter should only be used together with FixedLocator



Let's use the size of the markers according to the number of occurrences and try to add noise to better distribute the binary groups

```
[671]: GPD_dataset_subset2_0NaN_T_colour = GPD_dataset_subset2_0NaN_T.copy()
GPD_dataset_subset2_0NaN_T_colour['colour'] = colours_list
```

```
[734]: GPD_dataset_subset2_0NaN_T_colour.value_counts()
```

```
[734]: type    season  s.pollination  guild    colour
0         3         0              p    #FFFFFF    33
1         6         0             $\aleph$  #FFFF00    32
0         3         0             $\Omega$  #FFFFFF    30
1         4         0              p    #FFFF00    29
0         4         0              p    #FFFFFF    25
..
         1         0              -    #FFFF00     1
1         0         0              -    #FFFF00     1
         $\aleph$  #FFFF00     1
0        10         1              s    #FFFF00     1
1         8         0              s    #FFFF00     1
Length: 177, dtype: int64
```

```
[649]: season_encoder.inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11])
```



```
[649]: array(['autspri', 'autumn', 'spriaut', 'spring', 'sprisum', 'sumaut',
            'summer', 'sumspri', 'undefined', 'winter', 'wispring', 'year'],
            dtype=object)
```

```
[731]: fig = plt.pyplot.figure(dpi = 500, figsize=(4,4))
ax = fig.add_subplot( projection = '3d')

ax.set_xlim(0,450)
ax.set_ylim(0,5)
ax.set_zlim(0,3)

x_label_positions = np.array([0,1,2,3,4,5,6,7,8,9,10,11])*40
ax.set_xticks(x_label_positions.tolist())
ax.set_xticklabels(season_encoder.
    ↪inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11]), \
                    fontsize = 5, rotation = 45, ha="right", va="center")
ax.set_yticks([0,1.5,3,4.5,6])
ax.set_yticklabels(['', type_encoder.inverse_transform([0])[0], '', \
                    type_encoder.inverse_transform([1])[0], '', fontsize = 8])
ax.set_zticks([0,1,2])
ax.set_zticklabels([s_pollination_encoder.inverse_transform([0])[0], '', \
                    s_pollination_encoder.inverse_transform([1])[0]], fontsize=
    ↪8)
ax.set_facecolor('#6b8e95')

for i, ind in enumerate(GPD_dataset_subset2_0NaN_T_colour.value_counts().index):
    ax.scatter(xs = GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪index[i][1]*40 + random.randrange(0, 100)/50 , \
              ys = GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪index[i][0]*3 + random.randrange(0, 100)/50 , \
              zs = GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪index[i][2]*2 + random.randrange(0, 100)/300 , \
              marker = GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪index[i][3],
              c = GPD_dataset_subset2_0NaN_T_colour.value_counts().index[i][4],
              sizes = [GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪iloc[i]] # markers dimension
    )

# set legend
for i in guild_mark_legend:
    ax.scatter(xs = -20 , \
              ys = -20 , \
              zs = -20 , \
              c = 'black', \
              sizes = [5], # markers dimension \
              marker = guild_mark_legend[i], \
```

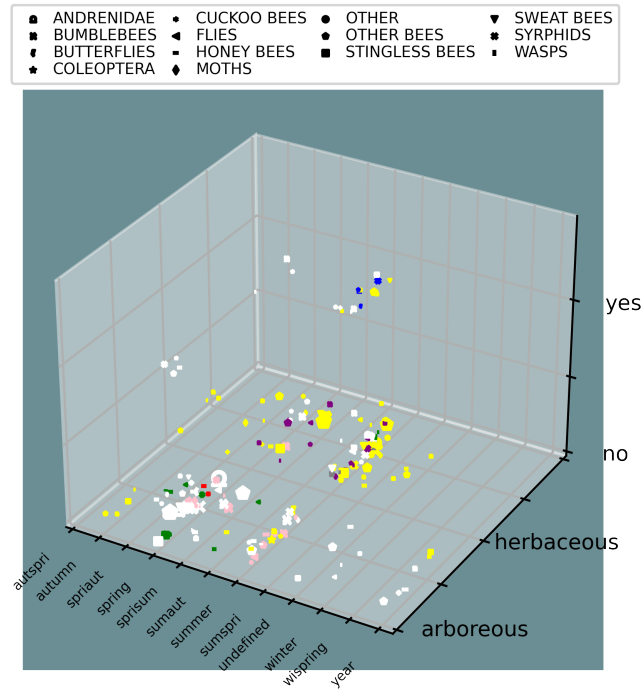
```

        label = i)

fig.legend(bbox_to_anchor=(0,1,0.92,0), labelspace = 0.1, handletextpad = 0.
↪3, \
        columnspacing = 0.2, fontsize = 'xx-small', ncol = 4 )

plt.pyplot.show()

```



```

[764]: for i in type_encoder.inverse_transform([0,1]):
        print(i)
        print(type_encoder.transform([i])+3)
        print(guild_mark_encoder.inverse_transform(type_encoder.transform([i])+3))

```

```

arboreous
[3]
['*']
herbaceous
[4]
['3']

```

Let's make a rotating animation to show the graph from different point o views.


```
# create the animation
anim = FuncAnimation(fig, update, frames = np.arange(0, 360, 2), repeat = True,
↳ \
                    interval = 400, fargs = (fig, ax))
# save the animation in a gif file
#commented to pdf version - latex problems- anim.save('Images/
↳ GPD_subset_exploration.gif', dpi=180, writer='imagemagick', fps=8)
```

[]:

```
[675]: #commented to pdf version - latex problems- anim.save('Images/
↳ GPD_subset_exploration_slow.gif', dpi=150, writer='imagemagick', fps=5)
```

```
[ ]: #Commented due to problems in insertion of gif in pdf via LaTeX
#![SegmentLocal](Images/GPD_subset_exploration.gif "segment")
```

Slower version

```
[742]: #Commented due to problems in insertion of gif in pdf via LaTeX
#![SegmentLocal](Images/GPD_subset_exploration_slow.gif "segment")
```

Maybe we can have a better visualization usign symbols for the type of plant

```
[822]: %matplotlib inline
fig = plt.pyplot.figure(dpi = 800, figsize=(4,4))
ax = fig.add_subplot( projection = '3d')

ax.set_xlim(0,450)
ax.set_ylim(0,600)
ax.set_zlim(0,3)
x_label_positions = np.array([0,1,2,3,4,5,6,7,8,9,10,11])*40
ax.set_xticks(x_label_positions.tolist())
ax.set_xticklabels(season_encoder.
↳ inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11]), \
                    fontsize = 5, rotation = 45, ha="right", va="center")
y_label_positions = np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13])*40
ax.set_yticks(y_label_positions)
ax.set_yticklabels(guild_encoder.
↳ inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11,12,13]), \
                    fontsize = 5, rotation = 90, ha="right", va="center")
ax.set_zticks([0,1,2])
ax.set_zticklabels([s_pollination_encoder.inverse_transform([0])[0], '\
                    s_pollination_encoder.inverse_transform([1])[0]', fontsize
↳ = 8)
ax.set_facecolor('#6b8e95')
```

```

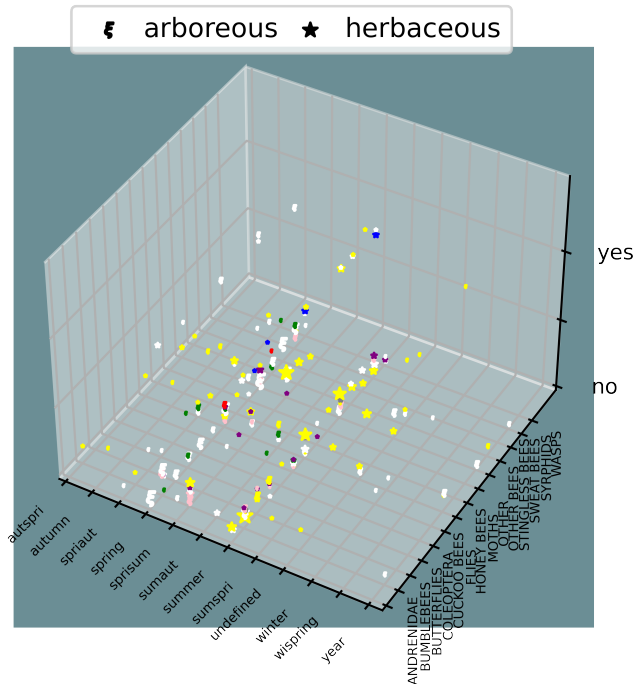
for i, ind in enumerate(GPD_dataset_subset2_0NaN_T_colour.value_counts().index):
    ax.scatter(xs = GPD_dataset_subset2_0NaN_T_colour.value_counts().
        ↪index[i][1]*40 + \
                random.randrange(0, 100)/50 , \
                ys = guild_mark_encoder.transform(\
                    [GPD_dataset_subset2_0NaN_T_colour.value_counts().
        ↪index[i][3]])*40 + \
                random.randrange(0, 100)/50 , \
                zs = GPD_dataset_subset2_0NaN_T_colour.value_counts().
        ↪index[i][2]*2 + \
                random.randrange(0, 100)/300 , \
                marker = guild_mark_encoder.inverse_transform(\
                    [GPD_dataset_subset2_0NaN_T_colour.
        ↪value_counts().index[i][0]+2])[0],
                c = GPD_dataset_subset2_0NaN_T_colour.value_counts().index[i][4],
                sizes = [GPD_dataset_subset2_0NaN_T_colour.value_counts().
        ↪iloc[i]],
                label = '_nolegend_'
    )

# set legend
for i in type_encoder.inverse_transform([0,1]):
    ax.scatter(xs = -20 , \
                ys = -20 , \
                zs = -20 , \
                c = 'black', \
                sizes = [5], # markers dimension \
                marker = guild_mark_encoder.inverse_transform(type_encoder.
        ↪transform([i])+2)[0], \
                label = i)

fig.legend(bbox_to_anchor=(0,0.95,0.8,0), labelspaceing = 0.1, handletextpad = 0.
        ↪3, \
                columnspacing = 0.2, ncol = 2, markerscale = 2.5 )

ax.view_init(elev = 40.)

```



```
[ ]: # create the animation
anim = FuncAnimation(fig, update, frames = np.arange(0, 360, 2), repeat = True,
    ↪ \
        interval = 400, fargs = (fig, ax))
# save the animation in a gif file
#commented to pdf version - latex problems- anim.save('Images/
    ↪ GPD_subset_exploration_2_slow.gif', dpi=150, writer='imagemagick', fps=6)

[ ]: #![SegmentLocal](Images/GPD_subset_exploration_2_slow.gif "segment")
```

Let's reduce 1 dimension with a upper view

```
[833]: %matplotlib inline
fig = plt.pyplot.figure(dpi = 800, figsize=(4,4))
ax = fig.add_subplot( projection = '3d')

ax.set_xlim(0,450)
ax.set_ylim(0,600)
ax.set_zlim(0,3)
x_label_positions = np.array([0,1,2,3,4,5,6,7,8,9,10,11])*40
ax.set_xticks(x_label_positions.tolist())
ax.set_xticklabels(season_encoder.
    ↪ inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11]), \
        fontsize = 5, rotation = 45, ha="right", va="center")
```

```

y_label_positions = np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13])*40
ax.set_yticks(y_label_positions)
ax.set_yticklabels(guild_encoder.
    ↪inverse_transform([0,1,2,3,4,5,6,7,8,9,10,11,12,13]), \
                        fontsize = 5, rotation = 90, ha="right", va="top")
ax.set_zticks([0,1,2])
ax.set_zticklabels([s_pollination_encoder.inverse_transform([0])[0], '\
    ↪s_pollination_encoder.inverse_transform([1])[0]], fontsize=
    ↪8)
ax.set_facecolor('#6b8e95')

for i, ind in enumerate(GPD_dataset_subset2_0NaN_T_colour.value_counts().index):
    ax.scatter(xs = GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪index[i][1]*40 + \
                random.randrange(0, 100)/50 , \
                ys = guild_mark_encoder.transform(\
                    [GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪index[i][3]])*40 + \
                random.randrange(0, 100)/50 , \
                zs = GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪index[i][2]*2 + \
                random.randrange(0, 100)/300 , \
                marker = guild_mark_encoder.inverse_transform(\
                    [GPD_dataset_subset2_0NaN_T_colour.
    ↪value_counts().index[i][0]+2])[0],
                c = GPD_dataset_subset2_0NaN_T_colour.value_counts().index[i][4],
                sizes = [GPD_dataset_subset2_0NaN_T_colour.value_counts().
    ↪iloc[i]],
                label = '_nolegend_'
    )

# set legend
for i in type_encoder.inverse_transform([0,1]):
    ax.scatter(xs = -20 , \
                ys = -20 , \
                zs = -20 , \
                c = 'black', \
                sizes = [5], # markers dimension \
                marker = guild_mark_encoder.inverse_transform(type_encoder.
    ↪transform([i])+2)[0], \
                label = i)

fig.legend(bbox_to_anchor=(0,0.95,0.8,0), labelspaceing = 0.1, handletextpad = 0.
    ↪3, \
            columnspacing = 0.2, ncol = 2, markerscale = 2.5 )

```

```
ax.view_init(90, 0)
```



1.4.1 Year distribution

Let's focus on the distribution of flowers and pollinators during the year

```
[834]: GPD_dataset_subset2_0NaN.describe()
```

```
[834]:
```

	crop	type	season	corolla	colour	nectar	b.system	\
count	779	779	779	779	779	779	779	
unique	78	2	12	3	8	3	4	
top	Prunus_avium	arboreous	summer	OPEN	white	yes	insects	
freq	81	443	279	615	380	721	647	

	s.pollination	inflorescence	composite		visitor		guild	\
count	779	779	779		779		779	
unique	2	4	2		254		14	
top	no	yes	no	Apis_mellifera		OTHER BEES		
freq	730	611	745		67		193	

	sociality	feeding
count	779	779
unique	3	4
top	no	polylectic

freq

379

696

```
[851]: GPD_seasons = GPD_dataset_subset2_0NaN.season.unique()
#let's convert to list to print the complete list without "..."
GPD_seasons.to_list()
```

```
[851]: ['sprisum',
'summer',
'spriaut',
'spring',
'autspri',
'sumspri',
'autumn',
'undefined',
'year',
'sumaut',
'wispring',
'winter']
```

mmm with “spriaut” it means between spring and autumn or in spring and in autumn but not in summer?

Notice that we have both spri-sum and sum-spri so considering that seems the order count we should infer that spri-aut means between spring and autumn including summer.

So let's convert these values in a more manageable format and try to make a nice and useful plot

```
[852]: def season_check(season, season_period):
    if season_period == 'undefined':
        return 0
    elif season_period == 'year':
        return 1
    elif season == 'spring':
        if season_period in ['sprisum', 'spriaut', 'sumspri', 'winspring',
↪ 'spring']:
            return 1
        else:
            return 0
    elif season == 'summer':
        if season_period in ['sprisum', 'spriaut', 'sumspri', 'sumaut',
↪ 'summer']:
            return 1
        else:
            return 0
    elif season == 'autumn':
        if season_period in ['spriaut', 'sumaut', 'sumspri', 'autspri',
↪ 'autumn']:
            return 1
```

```

        else:
            return 0
    elif season == 'winter':
        if season_period in ['winspring', 'sumspri', 'autspri', 'winter']:
            return 1
        else:
            return 0

    return 0

```

```
[853]: GPD_seasons_dataset = GPD_dataset_subset2_0NaN
```

```
[858]: GPD_seasons_dataset.columns
```

```
[858]: Index(['crop', 'type', 'season', 'corolla', 'colour', 'nectar', 'b.system',
          's.pollination', 'inflorescence', 'composite', 'visitor', 'guild',
          'sociality', 'feeding'],
          dtype='object')
```

```
[859]: for s in ['spring', 'summer', 'autumn', 'winter']:
        GPD_seasons_dataset[s] = GPD_seasons_dataset.apply(lambda row:
        ↪season_check(s, row.season), axis =1)
```

```
[865]: GPD_seasons_dataset.describe()
```

```
[865]:
```

	spring	summer	autumn	winter
count	779.000000	779.000000	779.000000	779.000000
mean	0.575096	0.608472	0.112965	0.055199
std	0.494646	0.488406	0.316754	0.228515
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	1.000000	1.000000	0.000000	0.000000
75%	1.000000	1.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000

```
[866]: GPD_seasons_dataset.columns
```

```
[866]: Index(['crop', 'type', 'season', 'corolla', 'colour', 'nectar', 'b.system',
          's.pollination', 'inflorescence', 'composite', 'visitor', 'guild',
          'sociality', 'feeding', 'spring', 'summer', 'autumn', 'winter'],
          dtype='object')
```

```
[867]: GPD_seasons_dataset.to_pickle(GPD_F_dataset_directory+"GPD_seasons_dataset.pkl")
```

1.4.2 Year distribution - Starting point

```
[80]: GPD_seasons_dataset = pd.  
      ↪read_pickle(GPD_F_dataset_directory+"GPD_seasons_dataset.pkl")
```

```
[883]: GPD_crops = GPD_seasons_dataset.crop.unique()  
GPD_crops_seasons_dataset = pd.DataFrame()  
GPD_crops_seasons_dataset = GPD_crops_seasons_dataset.assign(crop = GPD_crops.  
      ↪to_list())  
GPD_crops_seasons_dataset.describe()
```

```
[883]:
```

	crop
count	78
unique	78
top	Vaccinium_corymbosum
freq	1

```
[899]: GPD_crops_seasons_dataset.describe()
```

```
[899]:
```

	crop
count	78
unique	78
top	Vaccinium_corymbosum
freq	1

```
[939]: GPD_crops_seasons_dataset.columns
```

```
[939]: Index(['crop', 'herbaceous', 'arboreous', 'campanulate_corolla',  
          'open_corolla', 'tubular_corolla', 'composite_flower',  
          'self_pollination', 'nectar', 'spring', 'summer', 'autumn', 'winter'],  
          dtype='object')
```

Let's assume that "type", "corolla", "nectar", "self pollination", "inflorescence", and "composite" are unique for crop.

```
[905]: GPD_seasons_dataset.corolla.unique().to_list()
```

```
[905]: ['CAMPANULATE', 'OPEN', 'TUBULAR']
```

```
[906]: GPD_seasons_dataset.inflorescence.unique().to_list()
```

```
[906]: ['yes', 'solitary', 'solitary/clusters', 'solitary/pairs']
```

mmm inflorescence values seems to indicate that it is not so much a valuable feature.

```
[ ]: """To do: insert the cited columns considering the firsth value of each crop,␣  
      ↪maybe doing it we can also make a one hot encoding  
      GPD_crops_seasons_dataset = GPD_crops_seasons_dataset.assign(herbaceous = \
```

```

    ↪GPD_seasons_dataset.loc[GPD_crops_seasons_dataset['crop'] == , 'type'])
"""

```

```

[907]: GPD_crops_seasons_dataset = GPD_crops_seasons_dataset.assign(herbaceous =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                arboreous =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
    ↪campanulate_corolla = [0]*len(GPD_crops_seasons_dataset.crop),
                                open_corolla =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                tubular_corolla =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                composite_flower =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                self_pollination =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                nectar =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                spring =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                summer =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                autumn =
    ↪[0]*len(GPD_crops_seasons_dataset.crop),
                                winter =
    ↪[0]*len(GPD_crops_seasons_dataset.crop))
GPD_crops_seasons_dataset.describe()

```

```

[907]:      herbaceous  arboreous  campanulate_corolla  open_corolla  \
count          78.0        78.0                78.0          78.0
mean           0.0         0.0                  0.0           0.0
std            0.0         0.0                  0.0           0.0
min            0.0         0.0                  0.0           0.0
25%            0.0         0.0                  0.0           0.0
50%            0.0         0.0                  0.0           0.0
75%            0.0         0.0                  0.0           0.0
max            0.0         0.0                  0.0           0.0

      tubular_corolla  composite_flower  self_pollination  nectar  spring  \
count          78.0          78.0          78.0        78.0    78.0
mean           0.0           0.0           0.0         0.0     0.0
std            0.0           0.0           0.0         0.0     0.0
min            0.0           0.0           0.0         0.0     0.0
25%            0.0           0.0           0.0         0.0     0.0

```

50%	0.0	0.0	0.0	0.0	0.0
75%	0.0	0.0	0.0	0.0	0.0
max	0.0	0.0	0.0	0.0	0.0

	summer	autumn	winter
count	78.0	78.0	78.0
mean	0.0	0.0	0.0
std	0.0	0.0	0.0
min	0.0	0.0	0.0
25%	0.0	0.0	0.0
50%	0.0	0.0	0.0
75%	0.0	0.0	0.0
max	0.0	0.0	0.0

```
[913]: GPD_crops_seasons_dataset.columns
```

```
[913]: Index(['crop', 'herbaceous', 'arboreous', 'campanulate_corolla',
          'open_corolla', 'tubular_corolla', 'composite_flower',
          'self_pollination', 'nectar', 'spring', 'summer', 'autumn', 'winter'],
          dtype='object')
```

```
[940]: #there is for sure a better way, but that's the easiest to think about for
        ↪today XD
for i in range(0, len(GPD_crops_seasons_dataset.crop)):
    GPD_crops_seasons_dataset.loc[i, 'herbaceous'] = 1 if \
        GPD_seasons_dataset.loc[GPD_seasons_dataset.crop ==
        ↪GPD_crops_seasons_dataset.iloc[i,0] , \
                                'type'].to_list()[0] == 'herbaceous' else 0
    GPD_crops_seasons_dataset.loc[i, 'arboreous'] = 1 if \
        GPD_seasons_dataset.loc[GPD_seasons_dataset.crop ==
        ↪GPD_crops_seasons_dataset.iloc[i,0] , \
                                'type'].to_list()[0] == 'arboreous' else 0
    GPD_crops_seasons_dataset.loc[i, 'campanulate_corolla'] = 1 if \
        GPD_seasons_dataset.loc[GPD_seasons_dataset.crop ==
        ↪GPD_crops_seasons_dataset.iloc[i,0] , \
                                'corolla'].to_list()[0] == 'CAMPANULATE' else 0
    GPD_crops_seasons_dataset.loc[i, 'open_corolla'] = 1 if \
        GPD_seasons_dataset.loc[GPD_seasons_dataset.crop ==
        ↪GPD_crops_seasons_dataset.iloc[i,0] , \
                                'corolla'].to_list()[0] == 'OPEN' else 0
    GPD_crops_seasons_dataset.loc[i, 'tubular_corolla'] = 1 if \
        GPD_seasons_dataset.loc[GPD_seasons_dataset.crop ==
        ↪GPD_crops_seasons_dataset.iloc[i,0] , \
                                'corolla'].to_list()[0] == 'TUBULAR' else 0
    GPD_crops_seasons_dataset.loc[i, 'composite_flower'] = 1 if \
        GPD_seasons_dataset.loc[GPD_seasons_dataset.crop ==
        ↪GPD_crops_seasons_dataset.iloc[i,0] , \
```

```

        'composite'].to_list()[0] == 'yes' else 0
    GPD_crops_seasons_dataset.loc[i, 'self_pollination'] = 1 if \
    GPD_seasons_dataset.loc[GPD_seasons_dataset.crop == \
    ↪GPD_crops_seasons_dataset.iloc[i,0] , \
        's.pollination'].to_list()[0] == 'yes' else 0
    GPD_crops_seasons_dataset.loc[i, 'self_pollination'] = 1 if \
    GPD_seasons_dataset.loc[GPD_seasons_dataset.crop == \
    ↪GPD_crops_seasons_dataset.iloc[i,0] , \
        's.pollination'].to_list()[0] == 'yes' else 0
    GPD_crops_seasons_dataset.loc[i, 'nectar'] = 1 if \
    GPD_seasons_dataset.loc[GPD_seasons_dataset.crop == \
    ↪GPD_crops_seasons_dataset.iloc[i,0] , \
        'nectar'].to_list()[0] == 'yes' else 0
    GPD_crops_seasons_dataset.loc[i, 'spring'] = \
    GPD_seasons_dataset.loc[GPD_seasons_dataset.crop == \
    ↪GPD_crops_seasons_dataset.iloc[i,0] , \
        'spring'].sum()
    GPD_crops_seasons_dataset.loc[i, 'summer'] = \
    GPD_seasons_dataset.loc[GPD_seasons_dataset.crop == \
    ↪GPD_crops_seasons_dataset.iloc[i,0] , \
        'summer'].sum()
    GPD_crops_seasons_dataset.loc[i, 'autumn'] = \
    GPD_seasons_dataset.loc[GPD_seasons_dataset.crop == \
    ↪GPD_crops_seasons_dataset.iloc[i,0] , \
        'autumn'].sum()
    GPD_crops_seasons_dataset.loc[i, 'winter'] = \
    GPD_seasons_dataset.loc[GPD_seasons_dataset.crop == \
    ↪GPD_crops_seasons_dataset.iloc[i,0] , \
        'winter'].sum()

```

```
[941]: GPD_crops_seasons_dataset.describe()
```

```
[941]:
```

	herbaceous	arboreous	campanulate_corolla	open_corolla	\
count	78.000000	78.000000	78.000000	78.000000	
mean	0.435897	0.564103	0.141026	0.756410	
std	0.499083	0.499083	0.350301	0.432026	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	1.000000	
50%	0.000000	1.000000	0.000000	1.000000	
75%	1.000000	1.000000	0.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	

	tubular_corolla	composite_flower	self_pollination	nectar	\
count	78.000000	78.000000	78.000000	78.000000	

mean	0.102564	0.025641	0.076923	0.910256
std	0.305352	0.159085	0.268194	0.287664
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	1.000000
75%	0.000000	0.000000	0.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000

	spring	summer	autumn	winter
count	78.000000	78.000000	78.000000	78.000000
mean	5.743590	6.076923	1.128205	0.551282
std	11.843608	8.338254	2.769776	2.055385
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	1.500000	4.000000	0.000000	0.000000
75%	6.000000	6.750000	0.000000	0.000000
max	81.000000	44.000000	14.000000	14.000000

```
[942]: GPD_crops_seasons_dataset.
        ↪to_pickle(GPD_F_dataset_directory+"GPD_crops_seasons_dataset.pkl")
```

1.4.3 Year distribution - Starting point

```
[4]: GPD_crops_seasons_dataset = pd.
        ↪read_pickle(GPD_F_dataset_directory+"GPD_crops_seasons_dataset.pkl")
```

```
[5]: GPD_crops_seasons_dataset.describe()
```

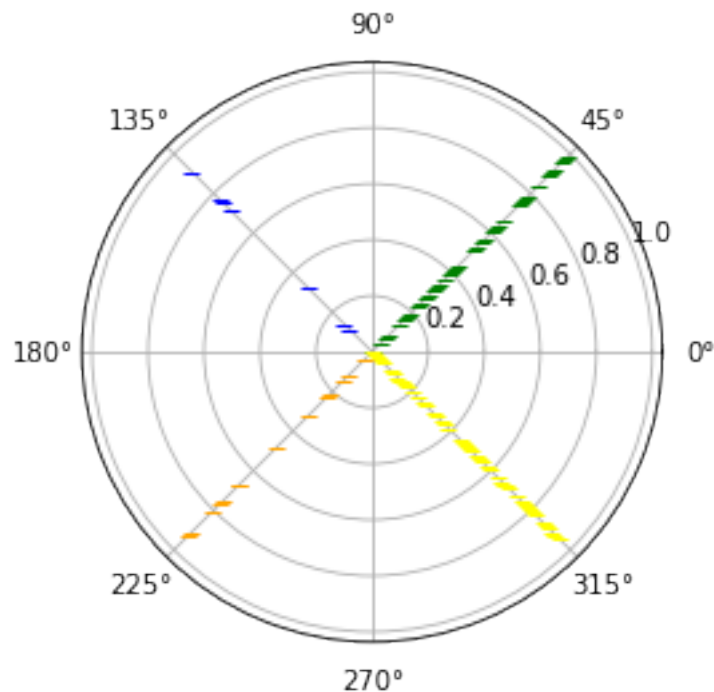
```
[5]:
```

	herbaceous	arboreous	campanulate_corolla	open_corolla	\
count	78.000000	78.000000	78.000000	78.000000	
mean	0.435897	0.564103	0.141026	0.756410	
std	0.499083	0.499083	0.350301	0.432026	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	1.000000	
50%	0.000000	1.000000	0.000000	1.000000	
75%	1.000000	1.000000	0.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	

	tubular_corolla	composite_flower	self_pollination	nectar	\
count	78.000000	78.000000	78.000000	78.000000	
mean	0.102564	0.025641	0.076923	0.910256	
std	0.305352	0.159085	0.268194	0.287664	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	1.000000	
50%	0.000000	0.000000	0.000000	1.000000	
75%	0.000000	0.000000	0.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	

	spring	summer	autumn	winter
count	78.000000	78.000000	78.000000	78.000000
mean	5.743590	6.076923	1.128205	0.551282
std	11.843608	8.338254	2.769776	2.055385
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	1.500000	4.000000	0.000000	0.000000
75%	6.000000	6.750000	0.000000	0.000000
max	81.000000	44.000000	14.000000	14.000000

```
[962]: plt.pyplot.axes(projection = 'polar')
for i in range(0, len(GPD_crops_seasons_dataset.crop)):
    if GPD_crops_seasons_dataset.loc[i,'spring'] > 0:
        plt.pyplot.polar(np.pi/4, i/78, color = 'green', marker = '_')
    if GPD_crops_seasons_dataset.loc[i,'summer'] > 0:
        plt.pyplot.polar(7*np.pi/4, i/78, color = 'yellow', marker = '_')
    if GPD_crops_seasons_dataset.loc[i,'autumn'] > 0:
        plt.pyplot.polar(5*np.pi/4, i/78, color = 'orange', marker = '_')
    if GPD_crops_seasons_dataset.loc[i,'winter'] > 0:
        plt.pyplot.polar(3*np.pi/4, i/78, color = 'blue', marker = '_')
plt.pyplot.show()
```




```
[ ]: #To do: make understandable the complete period for each flower,
#maybe we should come back to the categorical variable to correctly identify
↳ the starting and ending period
```

```
[ ]: """
plt.pyplot.axes(projection = 'polar')
for unique_crop_index in range(0, len(GPD_crops_seasons_dataset.crop)):
    crop = GPD_crops_seasons_dataset.loc[unique_crop_index, 'crop']
    for full_crop_index in range(0, len(GPD_dataset_subset2_0NaN.
↳ loc[GPD_dataset_subset2_0NaN['crop'] == crop, 'season'])):
        angle_range = [0]
        season_period = GPD_dataset_subset2_0NaN.loc[
            GPD_dataset_subset2_0NaN['crop'] == crop 'season'].
↳ to_list()[full_crop_index]
        if season_period == 'year':
            angle_range = np.arange(0, (2 * np.pi), 0.01)
        elif season_period == 'spring':
            angle_range = np.arange(np.pi/4, -np.pi/4, -0.01)
        elif season_period == 'sprisum':
            angle_range = np.arange(np.pi/4, -3*np.pi/4, -0.01)
        elif season_period == 'spriaut':
            angle_range = np.arange(np.pi/4, -5*np.pi/4, -0.01)
        elif season_period == 'winspring':
            angle_range = np.arange(np.pi/4, 3*np.pi/4, 0.01)
        elif season_period == 'sumspri': #is not year?
            angle_range = np.arange(0, 6*np.pi/4, 0.01)
        elif season_period == 'summer':
            angle_range = np.arange(5*np.pi/4, 7*np.pi/4, 0.01)
        elif season_period == 'sumaut':
            angle_range = np.arange(3*np.pi/4, 7*np.pi/4, 0.01)
        elif season_period == 'autumn':
            angle_range = np.arange(3*np.pi/4, 5*np.pi/4, 0.01)
        elif season_period == 'autspri':
            angle_range = np.arange(-1*np.pi/4, 5*np.pi/4, 0.01)
        elif season_period == 'winter':
            angle_range = np.arange(1*np.pi/4, 3*np.pi/4, 0.01)
        for angle in angle_range:
            plt.pyplot.polar(angle, (0.5 + unique_crop_index/(78*2)), \
                color = 'orange', marker = '.')

plt.pyplot.show()
"""
```

```
[27]: plt.pyplot.axes(projection = 'polar')
for unique_crop_index in range(0, len(GPD_crops_seasons_dataset.crop)):
    crop = GPD_crops_seasons_dataset.loc[unique_crop_index, 'crop']
    season_period = GPD_dataset_subset2_0NaN.loc[
```

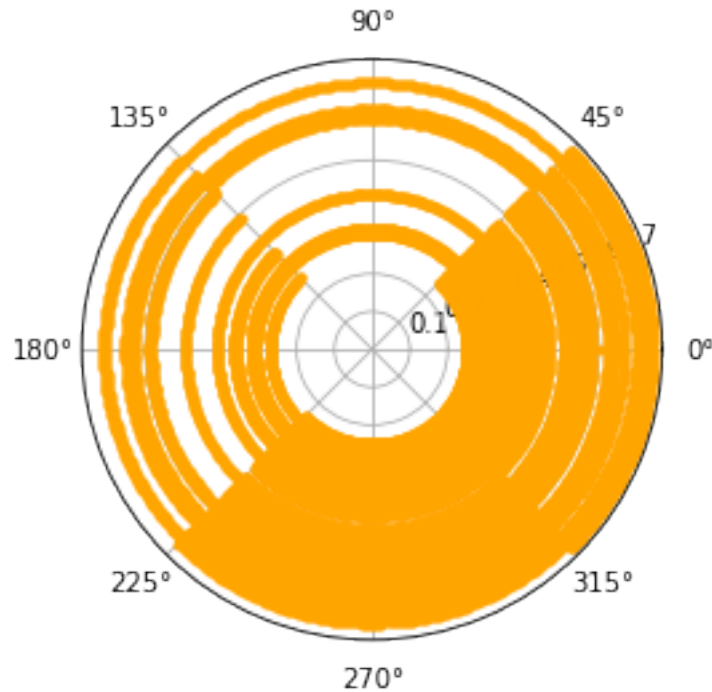
```

GPD_dataset_subset2_0NaN['crop'] == crop, 'season'].
↳to_list()[0]
    angle_range = [0]
    if season_period == 'year':
        angle_range = np.arange(0, (2 * np.pi), 0.01)
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -np.pi/4, -0.01)
    elif season_period == 'sprisum':
        angle_range = np.arange(np.pi/4, -3*np.pi/4, -0.01)
    elif season_period == 'spriaut':
        angle_range = np.arange(np.pi/4, -5*np.pi/4, -0.01)
    elif season_period == 'winspring':
        angle_range = np.arange(np.pi/4, 3*np.pi/4, 0.01)
    elif season_period == 'sumspri': #is not year?
        angle_range = np.arange(0, 6*np.pi/4, 0.01)
    elif season_period == 'summer':
        angle_range = np.arange(5*np.pi/4, 7*np.pi/4, 0.01)
    elif season_period == 'sumaut':
        angle_range = np.arange(3*np.pi/4, 7*np.pi/4, 0.01)
    elif season_period == 'autumn':
        angle_range = np.arange(3*np.pi/4, 5*np.pi/4, 0.01)
    elif season_period == 'autspri':
        angle_range = np.arange(-1*np.pi/4, 5*np.pi/4, 0.01)
    elif season_period == 'winter':
        angle_range = np.arange(1*np.pi/4, 3*np.pi/4, 0.01)

    if season_period != 'undefined':
        for angle in angle_range:
            plt.pyplot.polar(angle, (0.25 + unique_crop_index/(78*2)), \
                            color = 'orange', marker = '.')

plt.pyplot.show()

```



```
[35]: plt.pyplot.axes(projection = 'polar')
for unique_crop_index in range(0, len(GPD_crops_seasons_dataset.crop)):
    crop = GPD_crops_seasons_dataset.loc[unique_crop_index, 'crop']
    season_period = GPD_dataset_subset2_ONaN.loc[
        GPD_dataset_subset2_ONaN['crop'] == crop, 'season'].
        to_list()[0]
    angle_range = np.zeros(1)
    season_color = 'gray'
    if season_period == 'year':
        angle_range = np.arange(0, (2 * np.pi), 0.1)
        season_color = 'green'
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'sprisum':
        angle_range = np.arange(np.pi/4, -3*np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'spriaut':
        angle_range = np.arange(np.pi/4, -5*np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'winspring':
        angle_range = np.arange(np.pi/4, 3*np.pi/4, 0.1)
        season_color = 'blue'
    elif season_period == 'sumspri': #is not year?
```

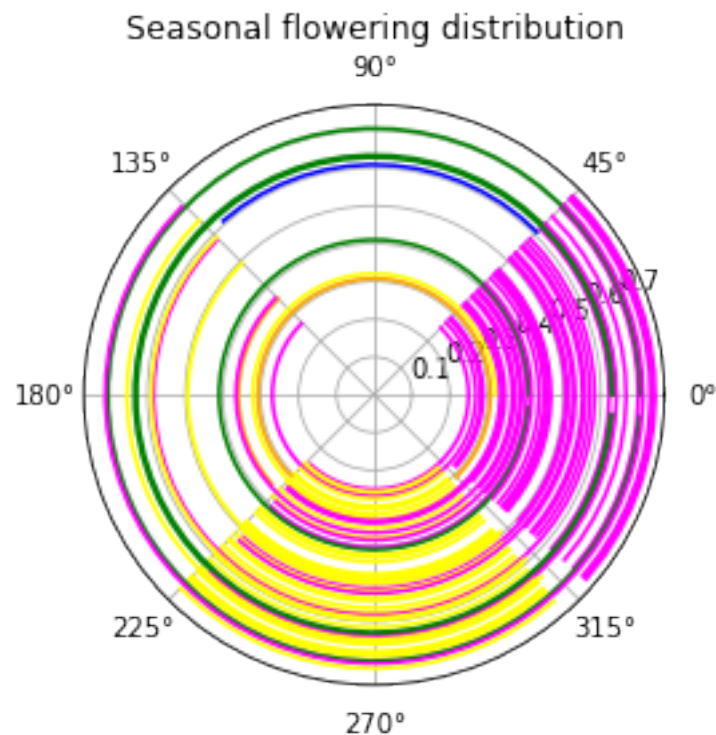
```

    angle_range = np.arange(0, 6*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'summer':
    angle_range = np.arange(5*np.pi/4, 7*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'sumaut':
    angle_range = np.arange(3*np.pi/4, 7*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'autumn':
    angle_range = np.arange(3*np.pi/4, 5*np.pi/4, 0.1)
    season_color = 'orange'
elif season_period == 'autspri':
    angle_range = np.arange(-1*np.pi/4, 5*np.pi/4, 0.1)
    season_color = 'orange'
elif season_period == 'winter':
    angle_range = np.arange(1*np.pi/4, 3*np.pi/4, 0.1)
    season_color = 'blue'

if season_period != 'undefined':
    positions = np.full(shape = angle_range.shape, \
                        fill_value = (0.25 + unique_crop_index/(78*2)))
    plt.pyplot.polar(angle_range, positions, color = season_color)

plt.pyplot.title('Seasonal flowering distribution')
plt.pyplot.show()

```



```

[21]: #let's convert colours in matplotlib colour values
colours_list = []
for color_data in GPD_dataset_subset2_0NaN.loc[:, 'colour']:
    if color_data == 'undefined':
        color_data = 'gray'
    colours_list.append(plt.colors.CSS4_COLORS[color_data])

[62]: fig, (ax1, ax2) = plt.pyplot.subplots(1, 2, subplot_kw = dict(polar = True),
↳ figsize=(10,10))

#flowering cycles differentiated by the type of plant (herbaceous or arboreous)
for unique_crop_index in range(0, len(GPD_crops_seasons_dataset.crop)):
    crop = GPD_crops_seasons_dataset.loc[unique_crop_index, 'crop']
    season_period = GPD_dataset_subset2_0NaN.loc[
        GPD_dataset_subset2_0NaN['crop'] == crop, 'season'].
↳to_list()[0]
    first_element_index = GPD_dataset_subset2_0NaN.loc[
        GPD_dataset_subset2_0NaN['crop'] == crop, :].index.
↳to_list()[0]

    angle_range = np.zeros(1)
    season_color = 'gray'
    if season_period == 'year':
        angle_range = np.arange(0, (2 * np.pi), 0.1)
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -np.pi/4, -0.1)
    elif season_period == 'sprisum':
        angle_range = np.arange(np.pi/4, -3*np.pi/4, -0.1)
    elif season_period == 'spriaut':
        angle_range = np.arange(np.pi/4, -5*np.pi/4, -0.1)
    elif season_period == 'winspring':
        angle_range = np.arange(np.pi/4, 3*np.pi/4, 0.1)
    elif season_period == 'sumspri': #is not year?
        angle_range = np.arange(0, 6*np.pi/4, 0.1)
    elif season_period == 'summer':
        angle_range = np.arange(5*np.pi/4, 7*np.pi/4, 0.1)
    elif season_period == 'sumaut':
        angle_range = np.arange(3*np.pi/4, 7*np.pi/4, 0.1)
    elif season_period == 'autumn':
        angle_range = np.arange(3*np.pi/4, 5*np.pi/4, 0.1)
    elif season_period == 'autspri':
        angle_range = np.arange(-1*np.pi/4, 5*np.pi/4, 0.1)
    elif season_period == 'winter':
        angle_range = np.arange(1*np.pi/4, 3*np.pi/4, 0.1)

```

```

    if season_period != 'undefined':
        positions = np.full(shape = angle_range.shape, \
                             fill_value = (0.25 + unique_crop_index/(78*2)))
        if GPD_crops_seasons_dataset.loc[unique_crop_index, 'herbaceous'] == 1:
            ax1.plot(angle_range, positions, color = colours_list[first_element_index] )
        else:
            ax2.plot(angle_range, positions, color = colours_list[first_element_index] )

ax1.set_title('Herbaceous')
ax1.set_facecolor('#D3D3D3')
ax2.set_title('Arboreous')
ax2.set_facecolor('#D3D3D3')

fig.suptitle('Seasonal flowering distribution', fontsize=25, y=0.8)
fig.text(0.35, 0.2, 'Global pollinator database', fontsize=16)
fig.text(0.36, 0.15, 'Boreux & Klein - Figshare Dataset', fontsize=12)
fig.text(0.37, 0.1, 'https://doi.org/10.6084/m9.figshare.9980471.v1',
        fontsize=8)

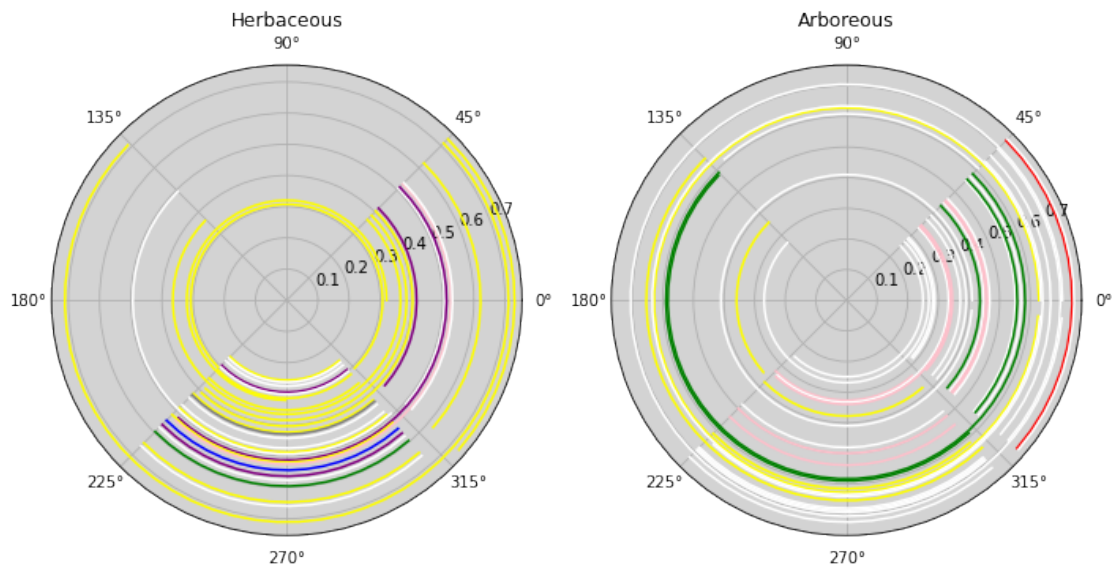
#set spacing between plots
fig.tight_layout()

plt.pyplot.savefig('Images/Seasonal flowering distribution.png', dpi=150)
plt.pyplot.savefig('Images/Seasonal flowering distribution.jpg', dpi=150)

plt.pyplot.show()

```

Seasonal flowering distribution



Global pollinator database

Boreux & Klein - Figshare Dataset

<https://doi.org/10.6084/m9.figshare.9980471.v1>

So now we should check the seasonal distribution of pollinators, maybe it's better looking for pollinator's guild

```
[87]: GPD_pollinators_guilds = GPD_seasons_dataset.guild.unique()
      GPD_pollinators_guilds_seasons_dataset = pd.DataFrame()
      GPD_pollinators_guilds_seasons_dataset = GPD_pollinators_guilds_seasons_dataset.
      ↪assign(guild = GPD_pollinators_guilds.to_list())
      GPD_pollinators_guilds_seasons_dataset.describe()
```

```
[87]:
```

	guild
count	14
unique	14
top	ANDRENIDAE
freq	1

```
[95]: GPD_pollinators_guilds_seasons_dataset.describe
```

```
[95]: <bound method NDFrame.describe of                                guild
0      ANDRENIDAE
1      BUMBLEBEES
2      BUTTERFLIES
```

```

3      COLEOPTERA
4      CUCKOO BEES
5      FLIES
6      HONEY BEES
7      MOTHS
8      OTHER
9      OTHER BEES
10     STINGLESS BEES
11     SWEAT BEES
12     SYRPHIDS
13     WASPS>

```

```
[97]: GPD_seasons_dataset.guild.value_counts()
```

```

[97]: OTHER BEES      193
      BUMBLEBEES     147
      HONEY BEES     145
      SYRPHIDS        63
      ANDRENIDAE      47
      STINGLESS BEES   44
      SWEAT BEES      36
      COLEOPTERA      27
      BUTTERFLIES     24
      FLIES           19
      CUCKOO BEES     16
      OTHER           7
      WASPS           6
      MOTHS           5
      Name: guild, dtype: int64

```

```

[91]: GPD_pollinators = GPD_seasons_dataset.visitor.unique()
      GPD_pollinators_seasons_dataset = pd.DataFrame()
      GPD_pollinators_seasons_dataset = GPD_pollinators_seasons_dataset.
      ↪assign(pollinator = GPD_pollinators.to_list())
      GPD_pollinators_seasons_dataset.describe()

```

```

[91]:          pollinator
count                254
unique                254
top      Andrena_wilkella
freq                  1

```

```
[92]: GPD_pollinators.describe
```

```

[92]: <bound method Categorical.describe of ['Andrena_wilkella',
      'Andrena_barbilabris', 'Andrena_cineraria', 'Andrena_flavipes',
      'Andrena_gravida', ..., 'Dolichovespula_norwegica', 'Dolichovespula_saxonica',

```



```
'Bembecinus_tridens', 'Vespula_vulgaris', 'Philanthus_triangulum']
Length: 254
Categories (254, object): ['Adalia_decempunctata',
'Agapanthia_villosoviridescens', 'Agapostemon_virescens', 'Aglais_urticae', ...,
'Xylocopa_hottentotta', 'Xylocopa_valga', 'Xylocopa_violacea',
'Xylocopa_virginica']>
```

```
[90]: GPD_seasons_dataset.describe
```

```
[90]: <bound method NDFrame.describe of
corolla colour nectar \
0 Vaccinium_corymbosum arboreous sprisum CAMPANULATE white yes
1 Vaccinium_corymbosum arboreous sprisum CAMPANULATE white yes
2 Brassica_napus herbaceous summer OPEN yellow yes
3 Brassica_napus herbaceous summer OPEN yellow yes
4 Brassica_napus herbaceous summer OPEN yellow yes
..
774 Allium_oleraceum herbaceous summer CAMPANULATE purple yes
775 Jatropha_curcas arboreous spriaut OPEN green yes
776 Malus_domestica arboreous spring OPEN white yes
777 Phaseolus_coccineus herbaceous summer OPEN white yes
778 Capparis_spinosa arboreous summer OPEN white yes

b.system s.pollination inflorescence composite \
0 insects no yes no
1 insects no yes no
2 wind/insects no yes no
3 wind/insects no yes no
4 wind/insects no yes no
..
774 insects no yes no
775 insects no yes no
776 insects no yes no
777 insects no yes no
778 insects no solitary no

visitor guild sociality feeding spring \
0 Andrena_wilkella ANDRENIDAE no oligolectic 1
1 Andrena_barbilabris ANDRENIDAE no polylectic 1
2 Andrena_cineraria ANDRENIDAE no polylectic 0
3 Andrena_flavipes ANDRENIDAE no polylectic 0
4 Andrena_gravida ANDRENIDAE no polylectic 0
..
774 Dolichovespula_saxonica WASPS yes polylectic 0
775 Bembecinus_tridens WASPS no undefined 1
776 Vespula_vulgaris WASPS yes polylectic 1
777 Philanthus_triangulum WASPS no polylectic 0
```

```

778      Bembecinus_tridens      WASPS      no      undefined      0

      summer  autumn  winter
0         1         0         0
1         1         0         0
2         1         0         0
3         1         0         0
4         1         0         0
..      ...      ...      ...
774        1         0         0
775        1         1         0
776        0         0         0
777        1         0         0
778        1         0         0

```

```
[779 rows x 18 columns]>
```

```
[98]: GPD_seasons_dataset.visitor.value_counts()
```

```

[98]: Apis_mellifera      67
      Apis_dorsata      22
      Apis_florea      20
      Apis_cerana      20
      Bombus_terrestris  20
      ..
      Leucozona_lucorum   1
      Calliphora_vicina   1
      Calliphora_vomitoria 1
      Lasioglossum_subhirtum 1
      Adalia_decempunctata 1
      Name: visitor, Length: 254, dtype: int64

```

Let's check the number of pollinators for each season and the guild distribution

```

[105]: GPD_seasons_dataset.loc[GPD_seasons_dataset['spring'] == 1, 'visitor'].
      ↪value_counts().sum()

```

```
[105]: 448
```

```

[106]: GPD_seasons_dataset.loc[GPD_seasons_dataset['summer'] == 1, 'visitor'].
      ↪value_counts().sum()

```

```
[106]: 474
```

```

[107]: GPD_seasons_dataset.loc[GPD_seasons_dataset['autumn'] == 1, 'visitor'].
      ↪value_counts().sum()

```

[107]: 88

```
[108]: GPD_seasons_dataset.loc[GPD_seasons_dataset['winter'] == 1, 'visitor'].  
       ↪value_counts().sum()
```

[108]: 43

```
[111]: GPD_seasons_dataset.loc[GPD_seasons_dataset['spring'] == 1, 'guild'].  
       ↪value_counts()
```

```
[111]: OTHER BEES          127  
       BUMBLEBEES         79  
       HONEY BEES         74  
       ANDRENIDAE         33  
       SYRPHIDS           31  
       SWEAT BEES         27  
       STINGLESS BEES      23  
       CUCKOO BEES        15  
       COLEOPTERA         14  
       BUTTERFLIES        10  
       FLIES              5  
       OTHER              5  
       MOTHS              3  
       WASPS              2  
       Name: guild, dtype: int64
```

```
[112]: GPD_seasons_dataset.loc[GPD_seasons_dataset['summer'] == 1, 'guild'].  
       ↪value_counts()
```

```
[112]: OTHER BEES          130  
       BUMBLEBEES        105  
       HONEY BEES         94  
       SYRPHIDS           37  
       STINGLESS BEES      22  
       ANDRENIDAE         17  
       BUTTERFLIES        15  
       COLEOPTERA         14  
       FLIES              14  
       SWEAT BEES         13  
       WASPS              5  
       MOTHS              4  
       CUCKOO BEES        2  
       OTHER              2  
       Name: guild, dtype: int64
```

```
[113]: GPD_seasons_dataset.loc[GPD_seasons_dataset['autumn'] == 1, 'guild'].  
       ↪value_counts()
```

```
[113]: HONEY BEES      32
      OTHER BEES      28
      BUMBLEBEES      9
      STINGLESS BEES   5
      SYRPHIDS        5
      FLIES           2
      MOTHS           2
      ANDRENIDAE      1
      BUTTERFLIES     1
      COLEOPTERA      1
      SWEAT BEES      1
      WASPS           1
      CUCKOO BEES     0
      OTHER           0
      Name: guild, dtype: int64
```

```
[114]: GPD_seasons_dataset.loc[GPD_seasons_dataset['winter'] == 1, 'guild'].
      ↪value_counts()
```

```
[114]: OTHER BEES      17
      HONEY BEES      15
      BUMBLEBEES      4
      STINGLESS BEES   4
      SYRPHIDS        2
      FLIES           1
      ANDRENIDAE      0
      BUTTERFLIES     0
      COLEOPTERA      0
      CUCKOO BEES     0
      MOTHS           0
      OTHER           0
      SWEAT BEES      0
      WASPS           0
      Name: guild, dtype: int64
```

Let's check how many different kinds of pollinators we can find in each season. The precedent count of single pollinators could be not so usefull because could be strongly biased by the sampling. Note that also this new count could be biased but should be a bit less biased.

```
[140]: GPD_pollinators_count_serie = GPD_seasons_dataset.
      ↪loc[GPD_seasons_dataset['spring'] == 1, 'visitor'].value_counts()
      len(GPD_pollinators_count_serie[GPD_pollinators_count_serie != 0].to_list())
```

```
[140]: 212
```

```
[141]: GPD_pollinators_count_serie = GPD_seasons_dataset.
      ↪loc[GPD_seasons_dataset['summer'] == 1, 'visitor'].value_counts()
```

```
len(GPD_pollinators_count_serie[GPD_pollinators_count_serie != 0].to_list())
```

[141]: 168

```
[142]: GPD_pollinators_count_serie = GPD_seasons_dataset.  
↳loc[GPD_seasons_dataset['autumn'] == 1, 'visitor'].value_counts()  
len(GPD_pollinators_count_serie[GPD_pollinators_count_serie != 0].to_list())
```

[142]: 50

```
[143]: GPD_pollinators_count_serie = GPD_seasons_dataset.  
↳loc[GPD_seasons_dataset['winter'] == 1, 'visitor'].value_counts()  
len(GPD_pollinators_count_serie[GPD_pollinators_count_serie != 0].to_list())
```

[143]: 31

Let's check if in each season there is only one record for each couple of crop and visitor

```
[146]: GPD_pollinators_crop_count_serie = GPD_seasons_dataset.  
↳loc[GPD_seasons_dataset['spring'] == 1, ['visitor', 'crop']].value_counts()  
len(GPD_pollinators_crop_count_serie[GPD_pollinators_crop_count_serie > 1 ].  
↳to_list())
```

[146]: 0

```
[147]: GPD_pollinators_crop_count_serie = GPD_seasons_dataset.  
↳loc[GPD_seasons_dataset['summer'] == 1, ['visitor', 'crop']].value_counts()  
len(GPD_pollinators_crop_count_serie[GPD_pollinators_crop_count_serie > 1 ].  
↳to_list())
```

[147]: 0

```
[148]: GPD_pollinators_crop_count_serie = GPD_seasons_dataset.  
↳loc[GPD_seasons_dataset['autumn'] == 1, ['visitor', 'crop']].value_counts()  
len(GPD_pollinators_crop_count_serie[GPD_pollinators_crop_count_serie > 1 ].  
↳to_list())
```

[148]: 0

```
[149]: GPD_pollinators_crop_count_serie = GPD_seasons_dataset.  
↳loc[GPD_seasons_dataset['winter'] == 1, ['visitor', 'crop']].value_counts()  
len(GPD_pollinators_crop_count_serie[GPD_pollinators_crop_count_serie > 1 ].  
↳to_list())
```

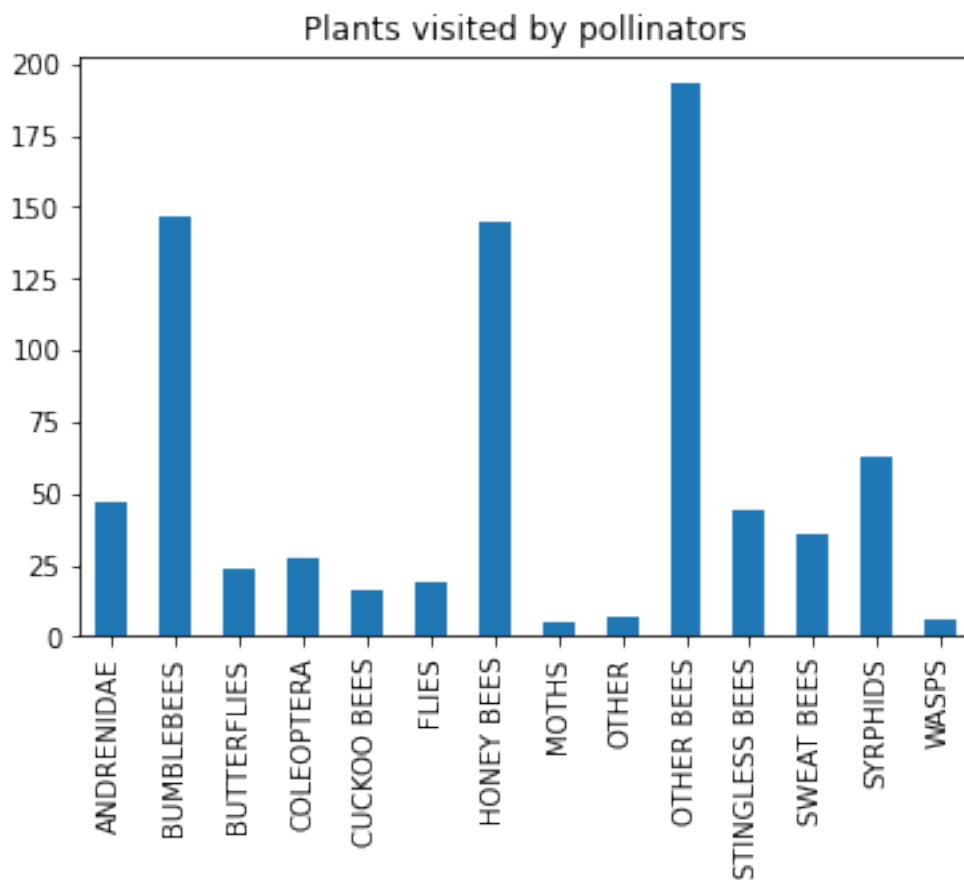
[149]: 0

OK, so we have confirmed that the observers declared one record for each different kind of visitor

for each crop, so we do not have information of how much frequently a kind of visitor visit a crop but we have information on how many kind of crops visit each visitor

```
[170]: ''' wrong plot title - see below '''
'''
plt.pyplot.figure()
plt.pyplot.title('Plants visited by pollinators')
GPD_seasons_dataset['guild'].value_counts(sort=False).plot(kind = 'bar')

plt.pyplot.savefig('Images/Plants for pollinators.png', dpi=150)
plt.pyplot.savefig('Images/Plants for pollinators.jpg', dpi=150)
'''
```



```
[168]: ''' wrong plot title - see below '''
'''
fig, ((ax1, ax2), (ax3, ax4)) = plt.pyplot.subplots(2, 2, figsize=(10,10))

fig.suptitle('Plants visited by pollinators during each season', fontsize=25,
↳y=1)
```

```

ax1.set_title('Spring')
ax2.set_title('Summer')
ax3.set_title('Autumn')
ax4.set_title('Winter')

GPD_seasons_dataset.loc[GPD_seasons_dataset['spring'] == 1, 'guild'].
    ↪value_counts(\
                    sort = False).plot(kind = 'bar', ax=ax1)
GPD_seasons_dataset.loc[GPD_seasons_dataset['summer'] == 1, 'guild'].
    ↪value_counts(\
                    sort = False).plot(kind = 'bar', ax=ax2)
GPD_seasons_dataset.loc[GPD_seasons_dataset['autumn'] == 1, 'guild'].
    ↪value_counts(\
                    sort = False).plot(kind = 'bar', ax=ax3)
GPD_seasons_dataset.loc[GPD_seasons_dataset['winter'] == 1, 'guild'].
    ↪value_counts(\
                    sort = False).plot(kind = 'bar', ax=ax4)

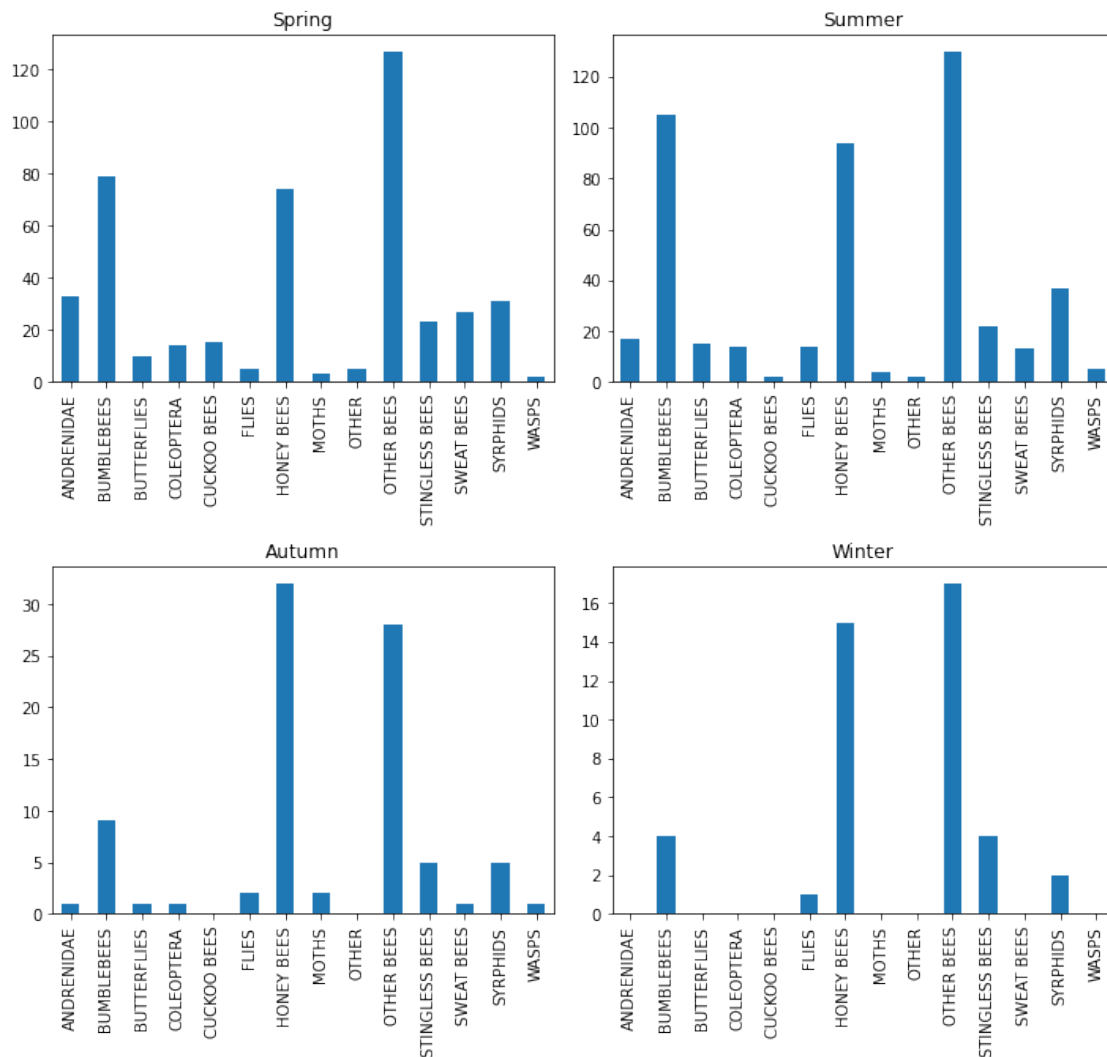
fig.tight_layout()

plt.pyplot.savefig('Images/Seasonal plants for pollinators.png', dpi=150)
plt.pyplot.savefig('Images/Seasonal plants for pollinators.jpg', dpi=150)

plt.pyplot.show()
'''

```

Plants visited by pollinators during each season



Pay attention at the y scale! Maybe we should make a unique graph with the seasonal differentiation by colours

```
[173]: GPD_seasons_dataset['guild'].unique().to_list()
```

```
[173]: ['ANDRENIDAE',
        'BUMBLEBEES',
        'BUTTERFLIES',
        'COLEOPTERA',
        'CUCKOO BEES',
        'FLIES',
        'HONEY BEES',
        'MOTHS',
```



```

'OTHER',
'OTHER BEES',
'STINGLESS BEES',
'SWEAT BEES',
'SYRPHIDS',
'WASPS']

```

```

[196]: ''' wrong plot title - see below '''
'''
plt.pyplot.figure(figsize=(15,15))

bar_number = len(GPD_seasons_dataset['guild'].unique().to_list())
x_range = np.arange(bar_number)
width = 0.18

spring = GPD_seasons_dataset.loc[GPD_seasons_dataset['spring'] == 1, 'guild'].
↳value_counts(\
                sort = False)
summer = GPD_seasons_dataset.loc[GPD_seasons_dataset['summer'] == 1, 'guild'].
↳value_counts(\
                sort = False)
autumn = GPD_seasons_dataset.loc[GPD_seasons_dataset['autumn'] == 1, 'guild'].
↳value_counts(\
                sort = False)
winter = GPD_seasons_dataset.loc[GPD_seasons_dataset['winter'] == 1, 'guild'].
↳value_counts(\
                sort = False)
total = GPD_seasons_dataset['guild'].value_counts(sort = False)

plt.pyplot.bar(x_range , spring, color = 'green',
               width = width, edgecolor = 'black',
               label='Spring')
plt.pyplot.bar(x_range + width, summer, color = 'yellow',
               width = width, edgecolor = 'black',
               label='Summer')
plt.pyplot.bar(x_range + 2*width, autumn, color = 'orange',
               width = width, edgecolor = 'black',
               label='Autumn')
plt.pyplot.bar(x_range + 3*width, winter, color = 'blue',
               width = width, edgecolor = 'black',
               label='Winter')
plt.pyplot.bar(x_range + 4*width, total, color = 'red',
               width = width, edgecolor = 'black',
               label='Year')

plt.pyplot.xlabel("Pollinators guilds")

```

```

plt.pyplot.ylabel("Number of crops visited")
plt.pyplot.title("Plants visited by pollinators")

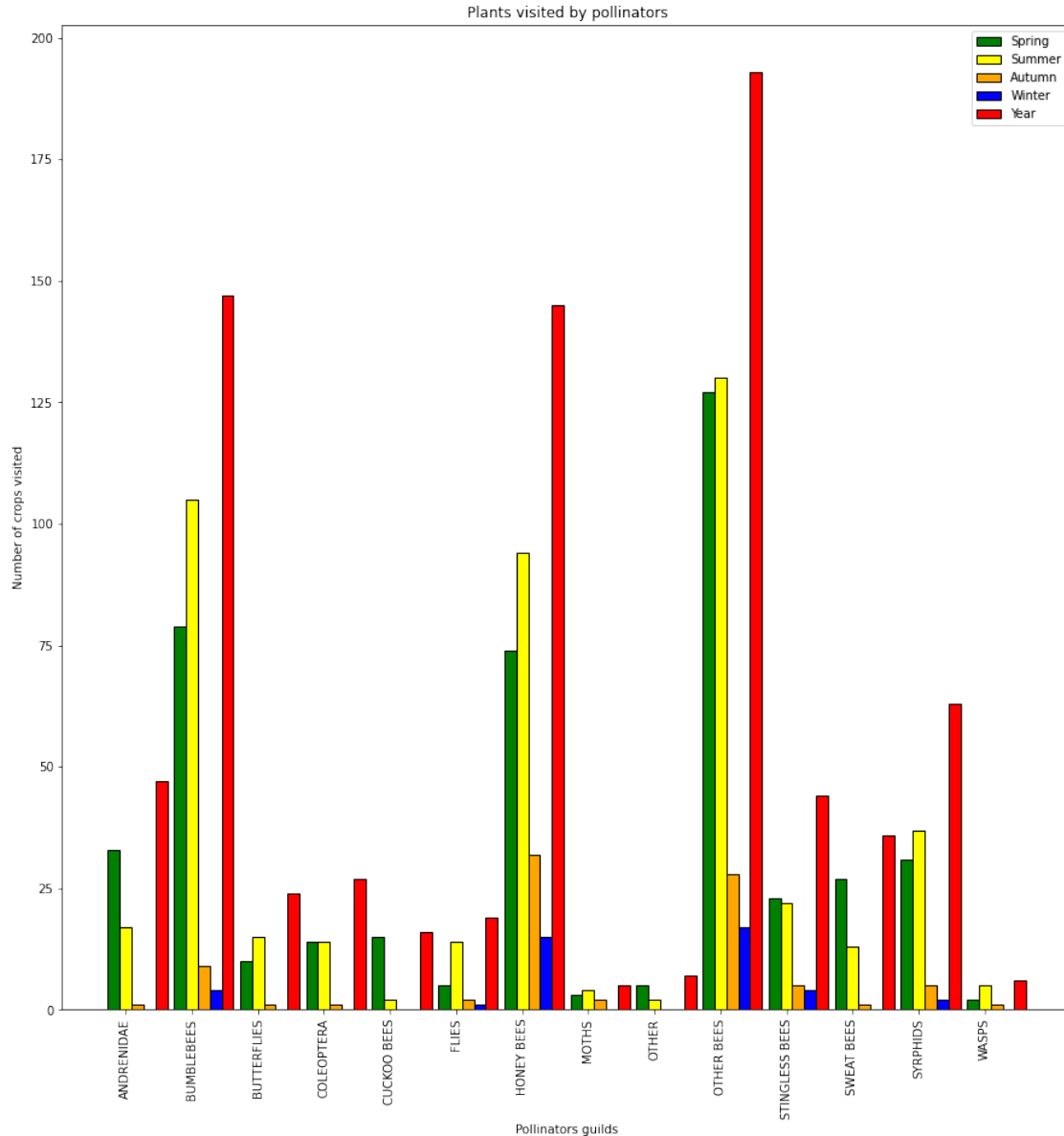
plt.pyplot.xticks(x_range + width, GPD_seasons_dataset['guild'].unique().
    ↪to_list(), \
                    rotation = 'vertical')
plt.pyplot.legend()

#fig.tight_layout()

plt.pyplot.savefig('Images/Plants visited by pollinators.png', dpi=150)
plt.pyplot.savefig('Images/Plants visited by pollinators.jpg', dpi=150)

plt.pyplot.show()
'''

```



1.5 Previous plots on pollinators are wrong

That wasn't the number of plants for pollinators. We have used the guilds so, as example, we can have 3 values in a guild because we have 3 species in that guild and not 3 different crops

```
[273]: spring_guild_crop_serie = GPD_seasons_dataset.loc[GPD_seasons_dataset['spring']_
      ↳ == 1, \
      ↳ ['guild', 'crop']].
      ↳ value_counts()
```

```
[270]: spring_guild_crop_serie
```

```
[270]: guild      crop
      ANDRENIDAE Prunus_avium      25
      SWEAT BEES Prunus_avium      23
      OTHER BEES Vaccinium_corymbosum 22
              Citrullus_lanatus    19
              Malus_domestica      15
              ..
      HONEY BEES Sambucus_racemosa   1
              Prunus_avium         1
              Persea_americana     1
              Cyphomandra_betacea  1
      WASPS      Malus_domestica     1
      Length: 128, dtype: int64
```

```
[279]: spring_guild_crop_serie.unstack(level = 0)
```

```
[279]: guild      ANDRENIDAE  BUMBLEBEES  BUTTERFLIES  COLEOPTERA  \
      crop
      Prunus_avium      25.0      7.0      NaN      NaN
      Vaccinium_corymbosum 2.0      5.0      NaN      NaN
      Citrullus_lanatus   NaN      5.0      NaN      NaN
      Malus_domestica     5.0      5.0      2.0      3.0
      Cucumis_sativus    NaN      5.0      NaN      NaN
      Vaccinium_myrtillus NaN      9.0      NaN      NaN
      Vaccinium_uliginosum NaN      9.0      NaN      NaN
      Cucurbita_pepo     NaN      1.0      NaN      NaN
      Coffea_arabica     NaN      NaN      NaN      NaN
      Castanea_crenata   NaN      2.0      7.0      6.0
      Trifolium_pratense  NaN      6.0      NaN      NaN
      Persea_americana   NaN      1.0      NaN      NaN
      Glycine_max        NaN      1.0      NaN      NaN
      Vicia_faba         NaN      5.0      NaN      1.0
      Anacardium_occidentale NaN      NaN      NaN      NaN
      Vaccinium_angustifolium NaN      4.0      NaN      NaN
      Coriandrum_sativum  1.0      NaN      1.0      1.0
      Litchi_chinensis   NaN      NaN      NaN      1.0
      Coffea_canephora   NaN      NaN      NaN      NaN
      Jatropha_curcas    NaN      NaN      NaN      NaN
      Brassica_rapa      NaN      1.0      NaN      NaN
      Dimocarpus_longan  NaN      NaN      NaN      NaN
      Sambucus_simpsonii NaN      NaN      NaN      NaN
      Cocos_nucifera     NaN      NaN      NaN      NaN
      Sambucus_racemosa  NaN      NaN      NaN      NaN
      Prunus_dulcis      NaN      2.0      NaN      NaN
      Vicia_faba_major   NaN      3.0      NaN      NaN
      Brassica_juncea    NaN      NaN      NaN      NaN
      Vitis_vinifera     NaN      NaN      NaN      2.0
```

Citrus_paradisi	NaN	1.0	NaN	NaN
Cyphomandra_betacea	NaN	2.0	NaN	NaN
Vigna_unguiculata	NaN	2.0	NaN	NaN
Citrus_reticulata	NaN	NaN	NaN	NaN
Sinapis_alba	NaN	NaN	NaN	NaN
Prunus_domestica	NaN	NaN	NaN	NaN
Pyrus_communis	NaN	1.0	NaN	NaN
Prunus_persica	NaN	1.0	NaN	NaN
Prunus_cerasus	NaN	1.0	NaN	NaN
Amomum_subulatum	NaN	NaN	NaN	NaN

guild	CUCKOO	BEES	FLIES	HONEY	BEES	MOTHS	OTHER	\
crop								
Prunus_avium	14.0	NaN		1.0	NaN	NaN		
Vaccinium_corymbosum	1.0	NaN		1.0	NaN	NaN		
Citrullus_lanatus	NaN	NaN		3.0	NaN	NaN		
Malus_domestica	NaN	2.0		2.0	NaN	4.0		
Cucumis_sativus	NaN	NaN		3.0	NaN	NaN		
Vaccinium_myrtillus	NaN	NaN		1.0	NaN	NaN		
Vaccinium_uliginosum	NaN	NaN		NaN	NaN	NaN		
Cucurbita_pepo	NaN	NaN		4.0	NaN	NaN		
Coffea_arabica	NaN	NaN		4.0	NaN	NaN		
Castanea_crenata	NaN	NaN		1.0	NaN	NaN		
Trifolium_pratense	NaN	NaN		NaN	NaN	NaN		
Persea_americana	NaN	NaN		1.0	NaN	NaN		
Glycine_max	NaN	NaN		2.0	NaN	NaN		
Vicia_faba	NaN	NaN		NaN	NaN	1.0		
Anacardium_occidentale	NaN	NaN		4.0	NaN	NaN		
Vaccinium_angustifolium	NaN	NaN		1.0	NaN	NaN		
Coriandrum_sativum	NaN	NaN		4.0	NaN	NaN		
Litchi_chinensis	NaN	NaN		4.0	NaN	NaN		
Coffea_canephora	NaN	NaN		4.0	NaN	NaN		
Jatropha_curcas	NaN	1.0		4.0	NaN	NaN		
Brassica_rapa	NaN	NaN		NaN	NaN	NaN		
Dimocarpus_longan	NaN	NaN		3.0	NaN	NaN		
Sambucus_simpsonii	NaN	NaN		1.0	NaN	NaN		
Cocos_nucifera	NaN	NaN		3.0	NaN	NaN		
Sambucus_racemosa	NaN	NaN		1.0	NaN	NaN		
Prunus_dulcis	NaN	NaN		2.0	NaN	NaN		
Vicia_faba_major	NaN	NaN		1.0	NaN	NaN		
Brassica_junceae	NaN	NaN		3.0	NaN	NaN		
Vitis_vinifera	NaN	NaN		1.0	NaN	NaN		
Citrus_paradisi	NaN	NaN		2.0	NaN	NaN		
Cyphomandra_betacea	NaN	NaN		1.0	NaN	NaN		
Vigna_unguiculata	NaN	1.0		1.0	NaN	NaN		
Citrus_reticulata	NaN	1.0		2.0	NaN	NaN		
Sinapis_alba	NaN	NaN		1.0	2.0	NaN		

Prunus_domestica	NaN	NaN	1.0	NaN	NaN
Pyrus_communis	NaN	NaN	2.0	NaN	NaN
Prunus_persica	NaN	NaN	2.0	NaN	NaN
Prunus_cerasus	NaN	NaN	2.0	NaN	NaN
Amomum_subulatum	NaN	NaN	1.0	1.0	NaN

guild	OTHER BEES	STINGLESS BEES	SWEAT BEES	SYRPHIDS	\
crop					
Prunus_avium	11.0	NaN	23.0	NaN	
Vaccinium_corymbosum	22.0	NaN	NaN	NaN	
Citrullus_lanatus	19.0	4.0	NaN	NaN	
Malus_domestica	15.0	NaN	NaN	9.0	
Cucumis_sativus	10.0	3.0	3.0	NaN	
Vaccinium_myrtillus	NaN	NaN	NaN	NaN	
Vaccinium_uliginosum	NaN	NaN	NaN	NaN	
Cucurbita_pepo	9.0	NaN	NaN	NaN	
Coffea_arabica	2.0	7.0	NaN	NaN	
Castanea_crenata	1.0	NaN	NaN	NaN	
Trifolium_pratense	NaN	NaN	NaN	NaN	
Persea_americana	NaN	5.0	NaN	NaN	
Glycine_max	5.0	NaN	NaN	NaN	
Vicia_faba	2.0	NaN	NaN	NaN	
Anacardium_occidentale	NaN	1.0	NaN	NaN	
Vaccinium_angustifolium	3.0	NaN	NaN	NaN	
Coriandrum_sativum	3.0	NaN	1.0	2.0	
Litchi_chinensis	3.0	NaN	NaN	4.0	
Coffea_canephora	3.0	NaN	NaN	NaN	
Jatropha_curcas	NaN	NaN	NaN	NaN	
Brassica_rapa	4.0	NaN	NaN	1.0	
Dimocarpus_longan	NaN	NaN	NaN	NaN	
Sambucus_simpsonii	3.0	NaN	NaN	2.0	
Cocos_nucifera	NaN	2.0	NaN	NaN	
Sambucus_racemosa	3.0	NaN	NaN	2.0	
Prunus_dulcis	3.0	NaN	NaN	2.0	
Vicia_faba_major	1.0	NaN	NaN	NaN	
Brassica_juncea	NaN	NaN	NaN	NaN	
Vitis_vinifera	NaN	NaN	NaN	3.0	
Citrus_paradisi	NaN	1.0	NaN	NaN	
Cyphomandra_betacea	NaN	NaN	NaN	NaN	
Vigna_unguiculata	1.0	NaN	NaN	NaN	
Citrus_reticulata	NaN	NaN	NaN	1.0	
Sinapis_alba	2.0	NaN	NaN	NaN	
Prunus_domestica	NaN	NaN	NaN	2.0	
Pyrus_communis	1.0	NaN	NaN	NaN	
Prunus_persica	NaN	NaN	NaN	2.0	
Prunus_cerasus	NaN	NaN	NaN	NaN	
Amomum_subulatum	1.0	NaN	NaN	1.0	

guild	WASPS
crop	
Prunus_avium	NaN
Vaccinium_corymbosum	NaN
Citrullus_lanatus	NaN
Malus_domestica	1.0
Cucumis_sativus	NaN
Vaccinium_myrtillus	NaN
Vaccinium_uliginosum	NaN
Cucurbita_pepo	NaN
Coffea_arabica	NaN
Castanea_crenata	NaN
Trifolium_pratense	NaN
Persea_americana	NaN
Glycine_max	NaN
Vicia_faba	NaN
Anacardium_occidentale	NaN
Vaccinium_angustifolium	NaN
Coriandrum_sativum	NaN
Litchi_chinensis	NaN
Coffea_canephora	NaN
Jatropha_curcas	1.0
Brassica_rapa	NaN
Dimocarpus_longan	NaN
Sambucus_simpsonii	NaN
Cocos_nucifera	NaN
Sambucus_racemosa	NaN
Prunus_dulcis	NaN
Vicia_faba_major	NaN
Brassica_juncea	NaN
Vitis_vinifera	NaN
Citrus_paradisi	NaN
Cyphomandra_betacea	NaN
Vigna_unguiculata	NaN
Citrus_reticulata	NaN
Sinapis_alba	NaN
Prunus_domestica	NaN
Pyrus_communis	NaN
Prunus_persica	NaN
Prunus_cerasus	NaN
Amomum_subulatum	NaN

```
[309]: spring_guild_crop_dataset = GPD_seasons_dataset.
        ↳loc[GPD_seasons_dataset['spring'] == 1, \
            ['guild', 'crop']].value_counts(sort =_
        ↳False\
```

```

        ).unstack(level = 0)
summer_guild_crop_dataset = GPD_seasons_dataset.
    ↳loc[GPD_seasons_dataset['summer'] == 1, \
        ['guild', 'crop']].value_counts(sort = ␣
    ↳False\
        ).unstack(level = 0)
autumn_guild_crop_dataset = GPD_seasons_dataset.
    ↳loc[GPD_seasons_dataset['autumn'] == 1, \
        ['guild', 'crop']].value_counts(sort = ␣
    ↳False\
        ).unstack(level = 0)
winter_guild_crop_dataset = GPD_seasons_dataset.
    ↳loc[GPD_seasons_dataset['winter'] == 1, \
        ['guild', 'crop']].value_counts(sort = ␣
    ↳False\
        ).unstack(level = 0)
year_guild_crop_dataset = GPD_seasons_dataset[['guild', 'crop']].value_counts(\
    sort = False).unstack(level = 0)

```

```

[310]: print(len(spring_guild_crop_dataset.count()))
       print(len(summer_guild_crop_dataset.count()))
       print(len(autumn_guild_crop_dataset.count()))
       print(len(winter_guild_crop_dataset.count()))
       print(len(year_guild_crop_dataset.count()))

```

```

14
14
12
6
14

```

Mmm whe have a problem of dimensions

```

[ ]: winter_guild_crop_serie = year_guild_crop_dataset.count().apply(\
    lambda x: df1.loc[x['Year'] == df1['Year'], ␣
    ↳x['State']].reset_index(drop=True), axis=1)

```

```

[311]: spring_guild_crop_dataset.count().index

```

```

[311]: CategoricalIndex(['ANDRENIDAE', 'BUMBLEBEES', 'BUTTERFLIES', 'COLEOPTERA',
                        'CUCKOO BEES', 'FLIES', 'HONEY BEES', 'MOTHS', 'OTHER',
                        'OTHER BEES', 'STINGLESS BEES', 'SWEAT BEES', 'SYRPHIDS',
                        'WASPS'],
                        categories=['ANDRENIDAE', 'BUMBLEBEES', 'BUTTERFLIES',
                        'COLEOPTERA', 'CUCKOO BEES', 'FLIES', 'HONEY BEES', 'MOTHS', ...],
                        ordered=False, dtype='category', name='guild')

```



```
[312]: winter_guild_crop_dataset.count().index
```

```
[312]: CategoricalIndex(['BUMBLEBEES', 'FLIES', 'HONEY BEES', 'OTHER BEES',  
                        'STINGLESS BEES', 'SYRPHIDS'],  
                        categories=['ANDRENIDAE', 'BUMBLEBEES', 'BUTTERFLIES',  
                        'COLEOPTERA', 'CUCKOO BEES', 'FLIES', 'HONEY BEES', 'MOTHS', ...],  
                        ordered=False, dtype='category', name='guild')
```

```
[324]: winter_guild_crop_dataset_full = year_guild_crop_dataset*0 +  
      ↪winter_guild_crop_dataset
```

```
[328]: winter_guild_crop_dataset_full.count()
```

```
[328]: guild  
ANDRENIDAE      0  
BUMBLEBEES      3  
BUTTERFLIES     0  
COLEOPTERA      0  
CUCKOO BEES     0  
FLIES           1  
HONEY BEES      7  
MOTHS           0  
OTHER           0  
OTHER BEES      4  
STINGLESS BEES   3  
SWEAT BEES      0  
SYRPHIDS        1  
WASPS           0  
dtype: int64
```

```
[329]: autumn_guild_crop_dataset_full = year_guild_crop_dataset*0 +  
      ↪autumn_guild_crop_dataset
```

```
[342]: plt.pyplot.figure(figsize=(15,15))  
  
bar_number = len(GPD_seasons_dataset['guild'].unique().to_list())  
x_range = np.arange(bar_number)  
width = 0.20  
  
plt.pyplot.bar(x_range + 1.5*width, year_guild_crop_dataset.count(), color =  
      ↪'red',  
               width = 4*width, edgecolor = 'black', label='Year')  
plt.pyplot.bar(x_range , spring_guild_crop_dataset.count(), color = 'green',  
               width = width, edgecolor = 'black', hatch='/', label='Spring')  
plt.pyplot.bar(x_range + width, summer_guild_crop_dataset.count(), color =  
      ↪'yellow',  
               width = width, edgecolor = 'black', hatch= '*', label='Summer')
```

```

plt.pyplot.bar(x_range + 2*width, autumn_guild_crop_dataset_full.count(), color=
    ↪ 'orange',
               width = width, edgecolor = 'black', hatch='-', label='Autumn')
plt.pyplot.bar(x_range + 3*width, winter_guild_crop_dataset_full.count(),
    ↪ color = 'blue',
               width = width, edgecolor = 'black', hatch='x', label='Winter')

plt.pyplot.xlabel("Pollinators guilds")
plt.pyplot.ylabel("Number of crops visited")
plt.pyplot.title("Plants visited by pollinators", fontsize=25)

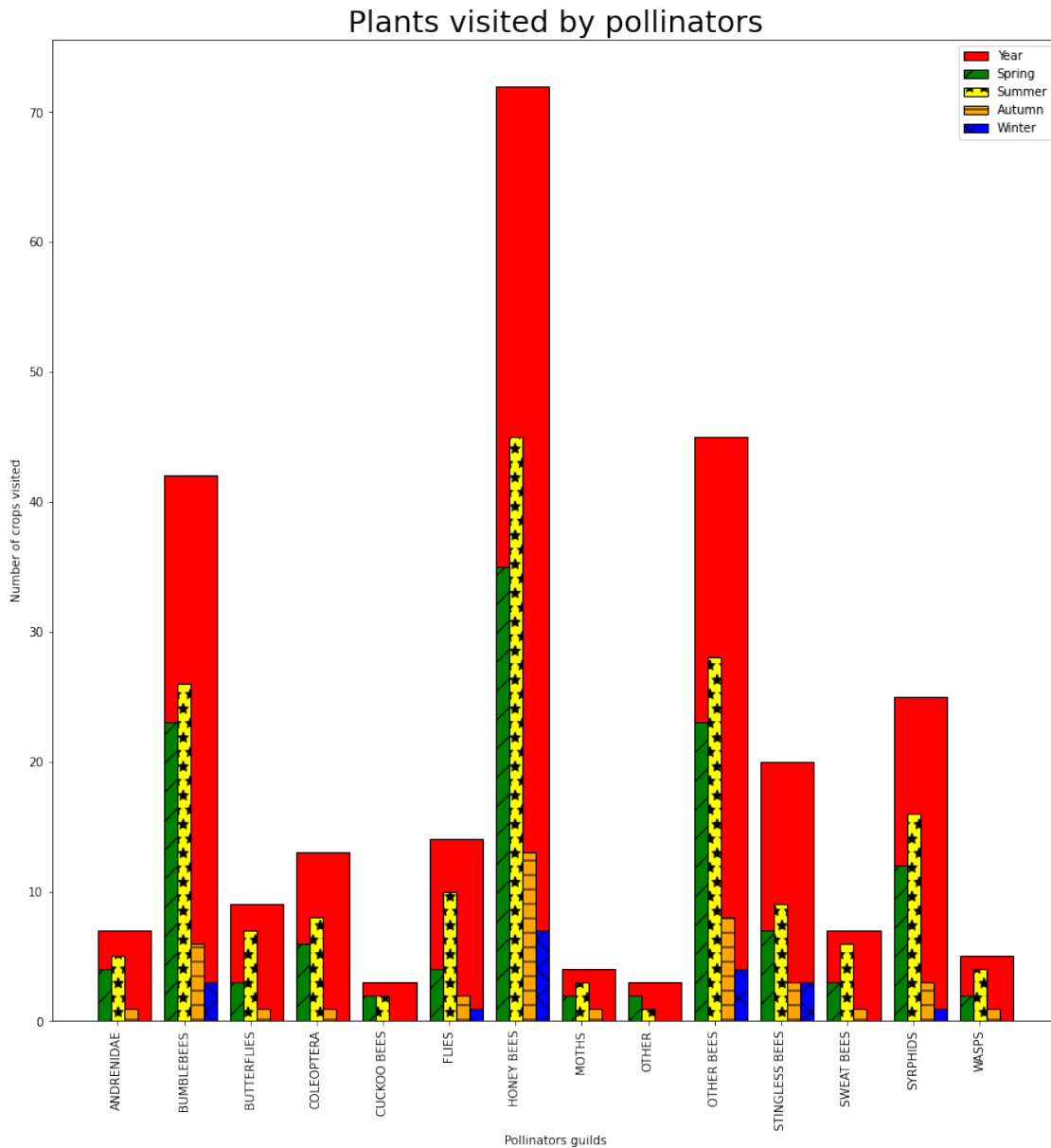
plt.pyplot.xticks(x_range + width, GPD_seasons_dataset['guild'].unique().
    ↪ to_list(), \
               rotation = 'vertical')
plt.pyplot.legend()

#fig.tight_layout()

plt.pyplot.savefig('Images/Plants visited by pollinators.png', dpi=150)
plt.pyplot.savefig('Images/Plants visited by pollinators.jpg', dpi=150)

plt.pyplot.show()

```



Could be interesting watch the same graph differentiated by herbaceous and arboreous plants

```
[348]: GPD_arboreous_seasons_dataset = GPD_seasons_dataset.loc[\
                                             GPD_seasons_dataset['type'] == 'arboreous',]
GPD_herbaceous_seasons_dataset = GPD_seasons_dataset.loc[\
                                   GPD_seasons_dataset['type'] == 'herbaceous',]
```

```
[390]: arboreous_spring_guild_crop_dataset = GPD_arboreous_seasons_dataset.\
        ↪loc[GPD_arboreous_seasons_dataset['spring'] == 1, \
```

```

                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
arboreous_summer_guild_crop_dataset = GPD_arboreous_seasons_dataset.
↪loc[GPD_arboreous_seasons_dataset['summer'] == 1, \
                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
arboreous_autumn_guild_crop_dataset = GPD_arboreous_seasons_dataset.
↪loc[GPD_arboreous_seasons_dataset['autumn'] == 1, \
                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
arboreous_winter_guild_crop_dataset = GPD_arboreous_seasons_dataset.
↪loc[GPD_arboreous_seasons_dataset['winter'] == 1, \
                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
arboreous_year_guild_crop_dataset = GPD_arboreous_seasons_dataset[['guild',
↪'crop']].value_counts(\
                                sort = False).unstack(level = 0)

herbaceous_spring_guild_crop_dataset = GPD_herbaceous_seasons_dataset.
↪loc[GPD_herbaceous_seasons_dataset['spring'] == 1, \
                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
herbaceous_summer_guild_crop_dataset = GPD_herbaceous_seasons_dataset.
↪loc[GPD_herbaceous_seasons_dataset['summer'] == 1, \
                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
herbaceous_autumn_guild_crop_dataset = GPD_herbaceous_seasons_dataset.
↪loc[GPD_herbaceous_seasons_dataset['autumn'] == 1, \
                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
herbaceous_winter_guild_crop_dataset = GPD_herbaceous_seasons_dataset.
↪loc[GPD_herbaceous_seasons_dataset['winter'] == 1, \
                                ['guild', 'crop']].value_counts(sort =
↪False\
                                ).unstack(level = 0)
herbaceous_year_guild_crop_dataset = GPD_herbaceous_seasons_dataset[['guild',
↪'crop']].value_counts(\
                                sort = False).unstack(level = 0)

```

```
[391]: arboreous_spring_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_spring_guild_crop_dataset
arboreous_summer_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_summer_guild_crop_dataset
arboreous_autumn_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_autumn_guild_crop_dataset
arboreous_winter_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_winter_guild_crop_dataset
arboreous_year_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_year_guild_crop_dataset

herbaceous_spring_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_spring_guild_crop_dataset
herbaceous_summer_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_summer_guild_crop_dataset
herbaceous_autumn_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_autumn_guild_crop_dataset
herbaceous_winter_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_winter_guild_crop_dataset
herbaceous_year_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_year_guild_crop_dataset
```

```
[354]: plt.pyplot.figure(figsize=(15,15))

bar_number = len(GPD_seasons_dataset['guild'].unique().to_list())
x_range = np.arange(bar_number)
width = 0.20

plt.pyplot.bar(x_range + 1.5*width, arboreous_year_guild_crop_dataset_full.
    ↪count(), \
        color = 'red', width = 4*width, edgecolor = 'black', ↪
    ↪label='Year')
plt.pyplot.bar(x_range , arboreous_spring_guild_crop_dataset_full.count(), \
        color = 'green', width = width, edgecolor = 'black', hatch='/', ↪
    ↪label='Spring')
plt.pyplot.bar(x_range + width, arboreous_summer_guild_crop_dataset_full.
    ↪count(), \
        color = 'yellow', width = width, edgecolor = 'black', hatch=↪
    ↪' ', label='Summer')
plt.pyplot.bar(x_range + 2*width, arboreous_autumn_guild_crop_dataset_full.
    ↪count(), \
        color = 'orange', width = width, edgecolor = 'black', hatch='- ', ↪
    ↪label='Autumn')
plt.pyplot.bar(x_range + 3*width, arboreous_winter_guild_crop_dataset_full.
    ↪count(), \
```

```

        color = 'blue', width = width, edgecolor = 'black', hatch='x',
        ↪label='Winter')

plt.pyplot.xlabel("Pollinators guilds")
plt.pyplot.ylabel("Number of different arboreous crops visited")
plt.pyplot.title("Arboreous plants visited by pollinators", fontsize=25)

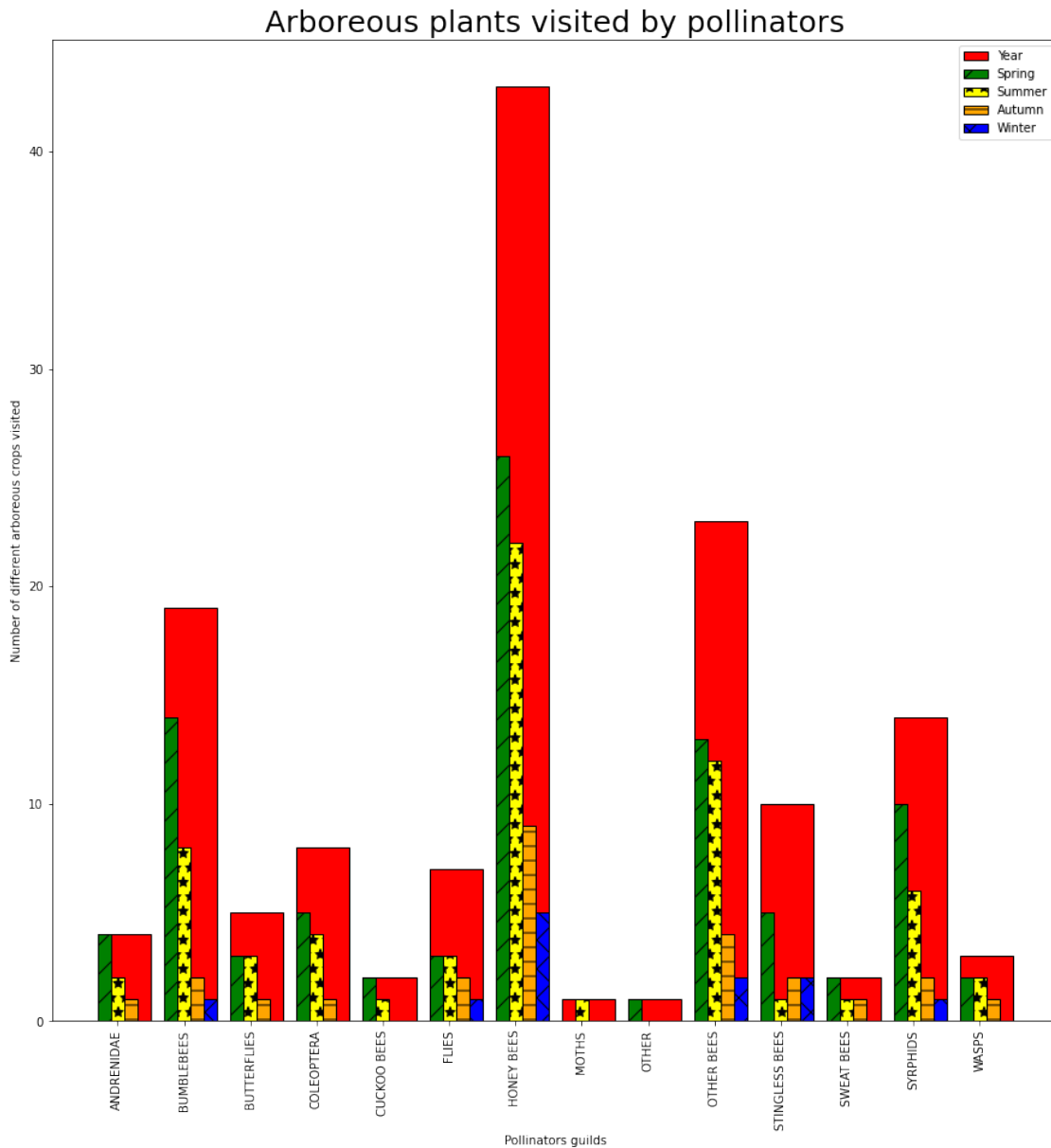
plt.pyplot.xticks(x_range + width, GPD_seasons_dataset['guild'].unique().
    ↪to_list(), \
                    rotation = 'vertical')
plt.pyplot.legend()

#fig.tight_layout()

plt.pyplot.savefig('Images/Arboreous plants visited by pollinators.png',
    ↪dpi=150)
plt.pyplot.savefig('Images/Arboreous plants visited by pollinators.jpg',
    ↪dpi=150)

plt.pyplot.show()

```



```
[355]: plt.pyplot.figure(figsize=(15,15))

bar_number = len(GPD_seasons_dataset['guild'].unique().to_list())
x_range = np.arange(bar_number)
width = 0.20

plt.pyplot.bar(x_range + 1.5*width, herbaceous_year_guild_crop_dataset_full.
    ↪count(), \
                color = 'red', width = 4*width, edgecolor = 'black',
    ↪label='Year')
```

```

plt.pyplot.bar(x_range , herbaceous_spring_guild_crop_dataset_full.count(), \
               color = 'green', width = width, edgecolor = 'black', hatch='/', \
               ↪label='Spring')
plt.pyplot.bar(x_range + width, herbaceous_summer_guild_crop_dataset_full.
               ↪count(), \
               color = 'yellow', width = width, edgecolor = 'black', hatch= \
               ↪' *', label='Summer')
plt.pyplot.bar(x_range + 2*width, herbaceous_autumn_guild_crop_dataset_full.
               ↪count(), color = 'orange',
               width = width, edgecolor = 'black', hatch='-', label='Autumn')
plt.pyplot.bar(x_range + 3*width, herbaceous_winter_guild_crop_dataset_full.
               ↪count(), color = 'blue',
               width = width, edgecolor = 'black', hatch='x', label='Winter')

plt.pyplot.xlabel("Pollinators guilds")
plt.pyplot.ylabel("Number of different herbaceous crops visited")
plt.pyplot.title("Herbaceous plants visited by pollinators", fontsize=25)

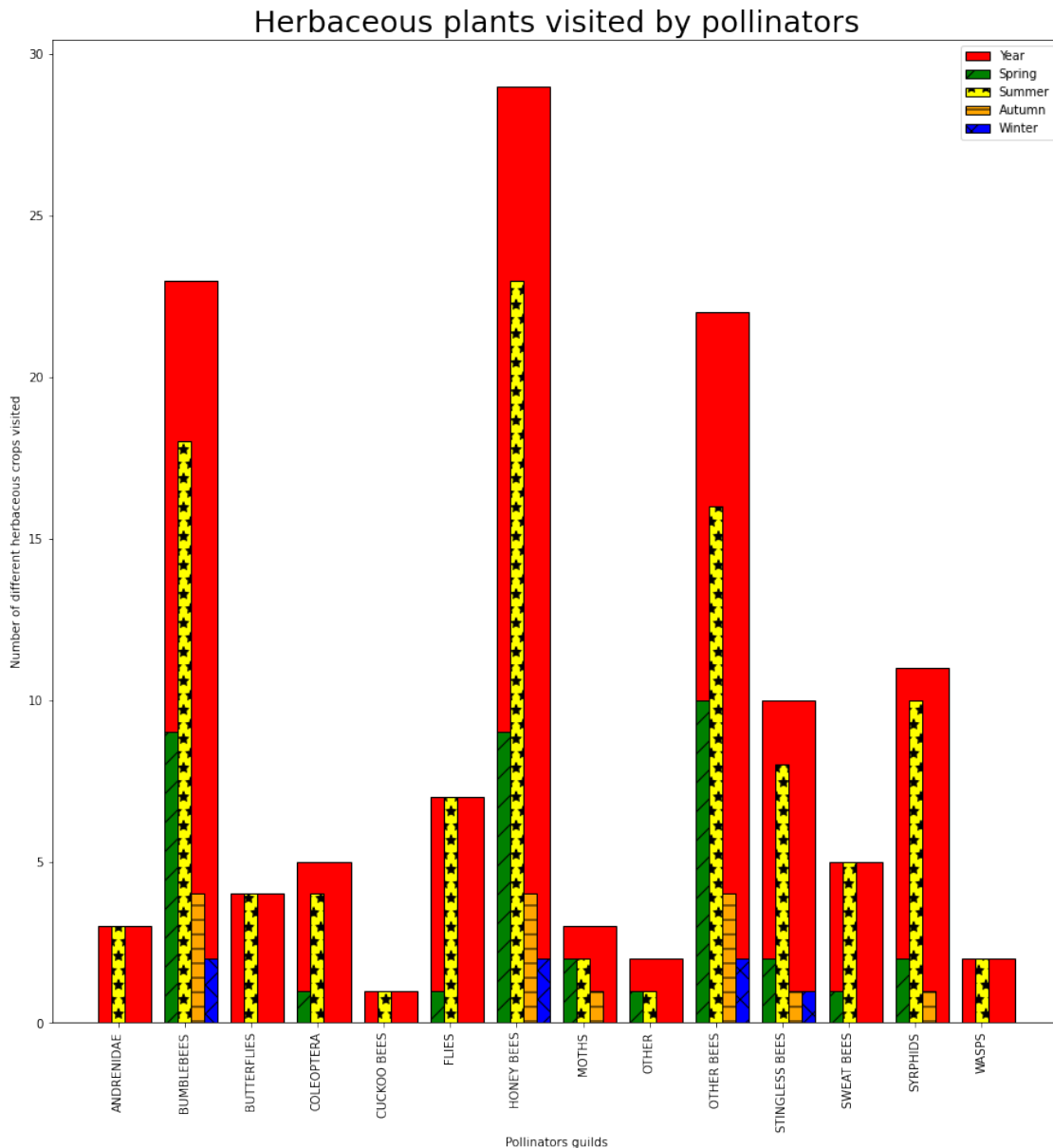
plt.pyplot.xticks(x_range + width, GPD_seasons_dataset['guild'].unique().
               ↪to_list(), \
               rotation = 'vertical')
plt.pyplot.legend()

#fig.tight_layout()

plt.pyplot.savefig('Images/Herbaceous plants visited by pollinators.png', \
               ↪dpi=150)
plt.pyplot.savefig('Images/Herbaceous plants visited by pollinators.jpg', \
               ↪dpi=150)

plt.pyplot.show()

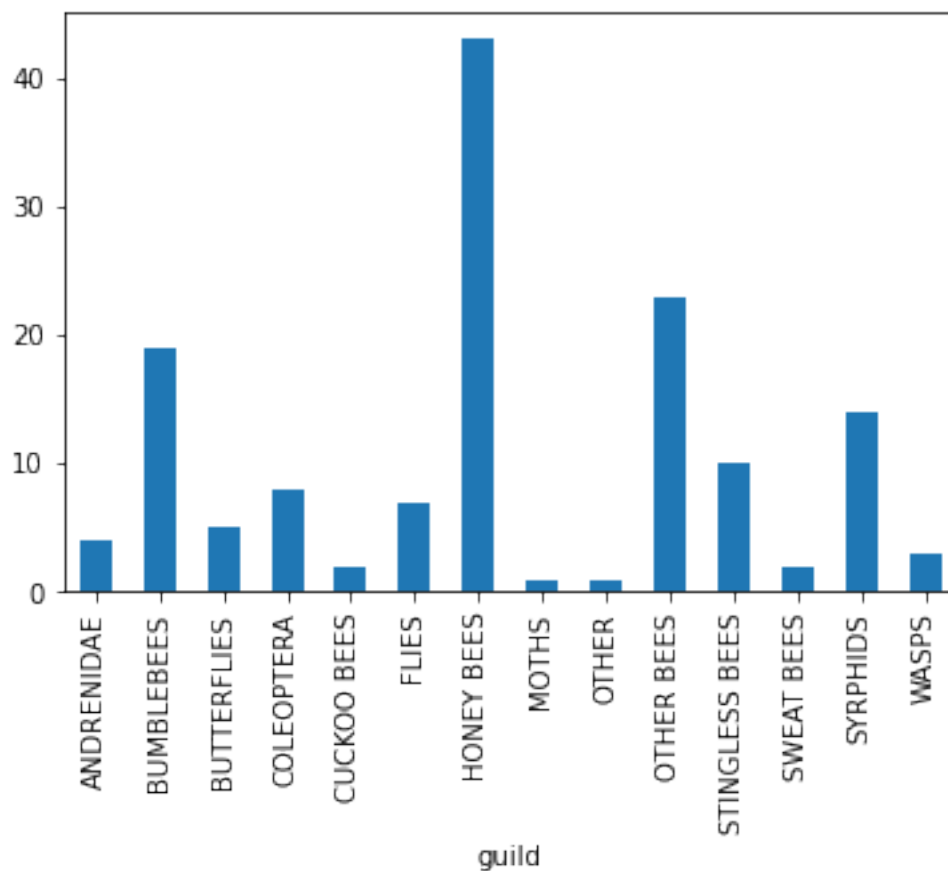
```

Maybe we should add the total amount of different crops visited by pollinators in each season, it will evidence if honeybees are able to visit all the crops in the dataset or there are some crops which are visited only by certain pollinators. Let's make a different graph for each season

```
[359]: arboreous_year_guild_crop_dataset_full.count().plot(kind='bar')
```

```
[359]: <AxesSubplot:xlabel='guild'>
```



```
[368]: len(herbaceous_year_guild_crop_dataset_full.index.to_list())
```

```
[368]: 78
```

```
[387]: GPD_arboreous_seasons_dataset.loc[GPD_arboreous_seasons_dataset['spring'] == 1, \
      \
      ['guild', 'crop']].value_counts(sort = \
      \
      False \
      ).unstack(level = 1)
```

```
[387]: guild      ANDRENIDAE  BUMBLEBEES  BUTTERFLIES  COLEOPTERA  \
crop
Coriandrum_sativum      1.0         NaN         1.0         1.0
Malus_domestica         5.0         5.0         2.0         3.0
Prunus_avium           25.0         7.0         NaN         NaN
Vaccinium_corymbosum    2.0         5.0         NaN         NaN
Castanea_crenata        NaN         2.0         7.0         6.0
Citrus_paradisi         NaN         1.0         NaN         NaN
Cyphomandra_betacea     NaN         2.0         NaN         NaN
```

Persea_americana	NaN	1.0	NaN	NaN	
Prunus_cerasus	NaN	1.0	NaN	NaN	
Prunus_dulcis	NaN	2.0	NaN	NaN	
Prunus_persica	NaN	1.0	NaN	NaN	
Pyrus_communis	NaN	1.0	NaN	NaN	
Vaccinium_angustifolium	NaN	4.0	NaN	NaN	
Vaccinium_myrtillus	NaN	9.0	NaN	NaN	
Vaccinium_uliginosum	NaN	9.0	NaN	NaN	
Litchi_chinensis	NaN	NaN	NaN	1.0	
Vitis_vinifera	NaN	NaN	NaN	2.0	
Citrus_reticulata	NaN	NaN	NaN	NaN	
Jatropha_curcas	NaN	NaN	NaN	NaN	
Anacardium_occidentale	NaN	NaN	NaN	NaN	
Cocos_nucifera	NaN	NaN	NaN	NaN	
Coffea_arabica	NaN	NaN	NaN	NaN	
Coffea_canephora	NaN	NaN	NaN	NaN	
Dimocarpus_longan	NaN	NaN	NaN	NaN	
Prunus_domestica	NaN	NaN	NaN	NaN	
Sambucus_racemosa	NaN	NaN	NaN	NaN	
Sambucus_simpsonii	NaN	NaN	NaN	NaN	
guild	CUCKOO BEES	FLIES	HONEY BEES	OTHER	OTHER BEES \
crop					
Coriandrum_sativum	NaN	NaN	4.0	NaN	3.0
Malus_domestica	NaN	2.0	2.0	4.0	15.0
Prunus_avium	14.0	NaN	1.0	NaN	11.0
Vaccinium_corymbosum	1.0	NaN	1.0	NaN	22.0
Castanea_crenata	NaN	NaN	1.0	NaN	1.0
Citrus_paradisi	NaN	NaN	2.0	NaN	NaN
Cyphomandra_betacea	NaN	NaN	1.0	NaN	NaN
Persea_americana	NaN	NaN	1.0	NaN	NaN
Prunus_cerasus	NaN	NaN	2.0	NaN	NaN
Prunus_dulcis	NaN	NaN	2.0	NaN	3.0
Prunus_persica	NaN	NaN	2.0	NaN	NaN
Pyrus_communis	NaN	NaN	2.0	NaN	1.0
Vaccinium_angustifolium	NaN	NaN	1.0	NaN	3.0
Vaccinium_myrtillus	NaN	NaN	1.0	NaN	NaN
Vaccinium_uliginosum	NaN	NaN	NaN	NaN	NaN
Litchi_chinensis	NaN	NaN	4.0	NaN	3.0
Vitis_vinifera	NaN	NaN	1.0	NaN	NaN
Citrus_reticulata	NaN	1.0	2.0	NaN	NaN
Jatropha_curcas	NaN	1.0	4.0	NaN	NaN
Anacardium_occidentale	NaN	NaN	4.0	NaN	NaN
Cocos_nucifera	NaN	NaN	3.0	NaN	NaN
Coffea_arabica	NaN	NaN	4.0	NaN	2.0
Coffea_canephora	NaN	NaN	4.0	NaN	3.0
Dimocarpus_longan	NaN	NaN	3.0	NaN	NaN

Prunus_domestica	NaN	NaN	1.0	NaN	NaN
Sambucus_racemosa	NaN	NaN	1.0	NaN	3.0
Sambucus_simpsonii	NaN	NaN	1.0	NaN	3.0

guild	STINGLESS BEES	SWEAT BEES	SYRPHIDS	WASPS
crop				
Coriandrum_sativum	NaN	1.0	2.0	NaN
Malus_domestica	NaN	NaN	9.0	1.0
Prunus_avium	NaN	23.0	NaN	NaN
Vaccinium_corymbosum	NaN	NaN	NaN	NaN
Castanea_crenata	NaN	NaN	NaN	NaN
Citrus_paradisi	1.0	NaN	NaN	NaN
Cyphomandra_betacea	NaN	NaN	NaN	NaN
Persea_americana	5.0	NaN	NaN	NaN
Prunus_cerasus	NaN	NaN	NaN	NaN
Prunus_dulcis	NaN	NaN	2.0	NaN
Prunus_persica	NaN	NaN	2.0	NaN
Pyrus_communis	NaN	NaN	NaN	NaN
Vaccinium_angustifolium	NaN	NaN	NaN	NaN
Vaccinium_myrtillus	NaN	NaN	NaN	NaN
Vaccinium_uliginosum	NaN	NaN	NaN	NaN
Litchi_chinensis	NaN	NaN	4.0	NaN
Vitis_vinifera	NaN	NaN	3.0	NaN
Citrus_reticulata	NaN	NaN	1.0	NaN
Jatropha_curcas	NaN	NaN	NaN	1.0
Anacardium_occidentale	1.0	NaN	NaN	NaN
Cocos_nucifera	2.0	NaN	NaN	NaN
Coffea_arabica	7.0	NaN	NaN	NaN
Coffea_canephora	NaN	NaN	NaN	NaN
Dimocarpus_longan	NaN	NaN	NaN	NaN
Prunus_domestica	NaN	NaN	2.0	NaN
Sambucus_racemosa	NaN	NaN	2.0	NaN
Sambucus_simpsonii	NaN	NaN	2.0	NaN

```
[393]: len(GPD_arboreous_seasons_dataset.loc[GPD_arboreous_seasons_dataset['spring']_
      == 1, \
      ['guild', 'crop']].value_counts(sort =_
      False\
      ).unstack(level = 1).count())
```

[393]: 27

```
[394]: len(GPD_arboreous_seasons_dataset.loc[GPD_arboreous_seasons_dataset['summer']_
      == 1, \
      ['guild', 'crop']].value_counts(sort =_
      False\
      ).unstack(level = 1).count())
```

[394]: 22

```
[395]: len(GPD_arboreous_seasons_dataset.loc[GPD_arboreous_seasons_dataset['autumn']_
↳== 1, \
                                             ['guild', 'crop']].value_counts(sort =_
↳False\
                                             ).unstack(level = 1).count())
```

[395]: 9

```
[ ]: len(GPD_arboreous_seasons_dataset.loc[GPD_arboreous_seasons_dataset['winter']_
↳== 1, \
                                             ['guild', 'crop']].value_counts(sort =_
↳False\
                                             ).unstack(level = 1).count())
```

```
[371]: total_pollinated_plants_array = np.ones([len(GPD_seasons_dataset['guild'].
↳unique().to_list())]*len(herbaceous_year_guild_crop_dataset_full.index.
↳to_list()))
```

```
[357]: plt.pyplot.bar(x_range + 1.5*width, herbaceous_year_guild_crop_dataset_full.
↳count(), \
                    color = 'red', width = 4*width, edgecolor = 'black',_
↳label='Year')
```

```
[357]: guild
ANDRENIDAE          0
BUMBLEBEES          9
BUTTERFLIES         0
COLEOPTERA          1
CUCKOO BEES         0
FLIES               1
HONEY BEES          9
MOTHS               2
OTHER               1
OTHER BEES         10
STINGLESS BEES       2
SWEAT BEES          1
SYRPHIDS            2
WASPS               0
dtype: int64
```

```
[437]: fig, ((ax1, ax2), (ax3, ax4)) = plt.pyplot.subplots(2, 2, figsize=(15,15))

fig.suptitle('Arboreous plants visited by pollinators during each season',_
↳fontsize=25, y=0.95)
ax1.set_title('Spring')
```

```

ax2.set_title('Summer')
ax3.set_title('Autumn')
ax4.set_title('Winter')

total_spring_pollinated_crops = len(GPD_arboreous_seasons_dataset.loc[
    GPD_arboreous_seasons_dataset['spring']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())
total_summer_pollinated_crops = len(GPD_arboreous_seasons_dataset.loc[
    GPD_arboreous_seasons_dataset['summer']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())
total_autumn_pollinated_crops = len(GPD_arboreous_seasons_dataset.loc[
    GPD_arboreous_seasons_dataset['autumn']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())
total_winter_pollinated_crops = len(GPD_arboreous_seasons_dataset.loc[
    GPD_arboreous_seasons_dataset['winter']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())

ax1.set_ylim(ymax= total_spring_pollinated_crops +
    ↪total_spring_pollinated_crops/10 )
ax2.set_ylim(ymax= total_summer_pollinated_crops +
    ↪total_summer_pollinated_crops/10 )
ax3.set_ylim(ymax= total_autumn_pollinated_crops +
    ↪total_autumn_pollinated_crops/10 )
ax4.set_ylim(ymax= total_winter_pollinated_crops +
    ↪total_winter_pollinated_crops/10 )

bar_number = len(GPD_seasons_dataset['guild'].unique().to_list())

width = 0.10

plt.pyplot.axes(ax1)
plt.pyplot.bar((bar_number-1)/2, total_spring_pollinated_crops, \
    color = 'gray', width = bar_number, edgecolor = 'black',
    ↪label='Total',\
    hatch='..')
arboreous_spring_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax1,
    ↪color = 'green')

```

```

plt.pyplot.axes(ax2)
plt.pyplot.bar((bar_number-1)/2, total_summer_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',\
               ↪label='Total',\
               hatch='..')
arboreous_summer_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax2,\
               ↪color = 'yellow')

plt.pyplot.axes(ax3)
plt.pyplot.bar((bar_number-1)/2, total_autumn_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',\
               ↪label='Total',\
               hatch='..')
arboreous_autumn_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax3,\
               ↪color = 'orange')

plt.pyplot.axes(ax4)
plt.pyplot.bar((bar_number-1)/2, total_winter_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',\
               ↪label='Total',\
               hatch='..')
arboreous_winter_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax4,\
               ↪color = 'blue')

#plt.pyplot.legend()

fig.text(0.20, 0.90, 'Gray dotted background indicate the total amount of\
               ↪plants pollinated in the season',\
               fontsize=15)

fig.text(0.30, 0.025, 'Global pollinator database - Boreux & Klein - Figshare\
               ↪Dataset',\
               fontsize=15)
fig.text(0.40, 0.01, 'https://doi.org/10.6084/m9.figshare.9980471.v1',\
               ↪fontsize=10)

fig.tight_layout(pad=5)
#plt.pyplot.margins(2000)

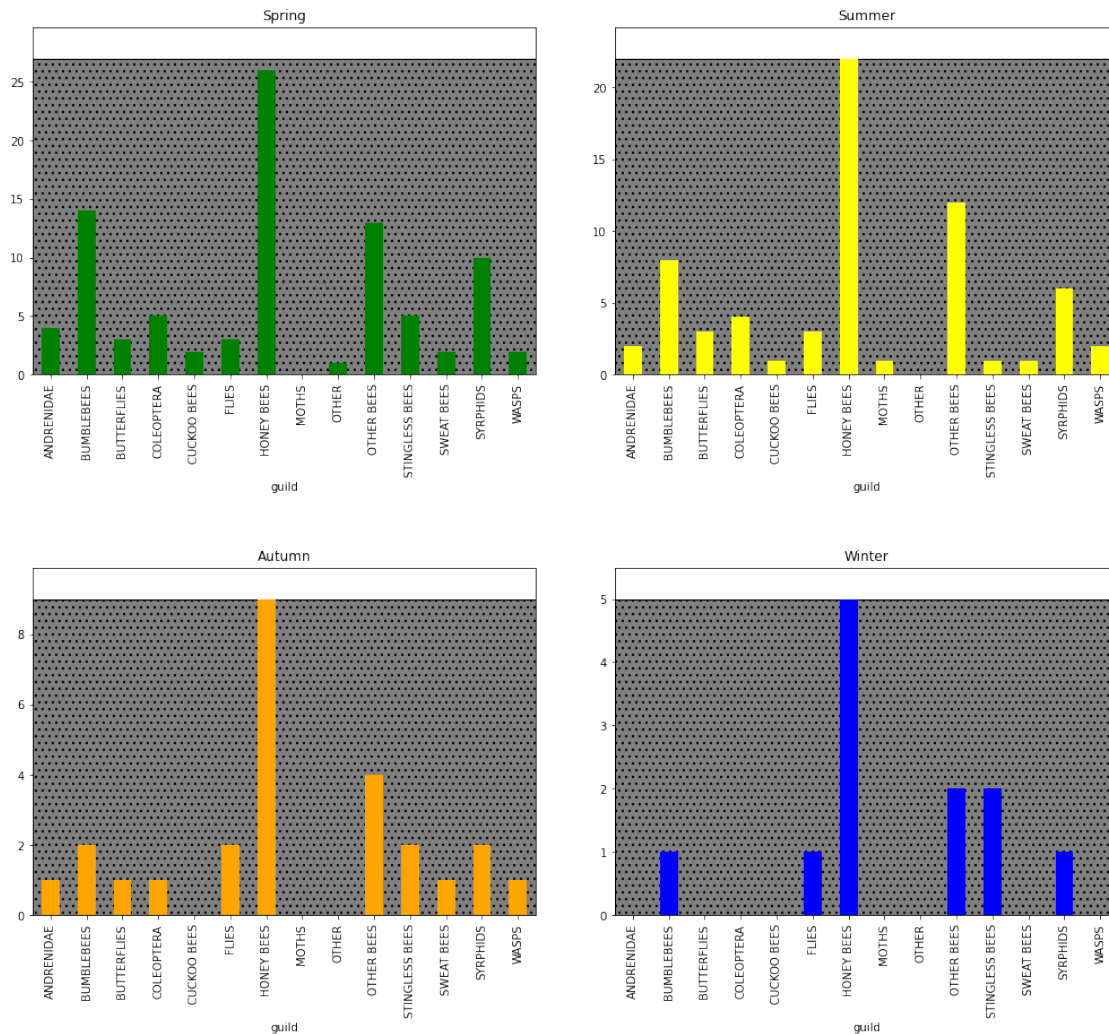
plt.pyplot.savefig('Images/Seasonal arboreous plants for pollinators.png',\
               ↪dpi=150)
plt.pyplot.savefig('Images/Seasonal arboreous plants for pollinators.jpg',\
               ↪dpi=150)

```

```
plt.pyplot.show()
```

Arboreous plants visited by pollinators during each season

Gray dotted background indicate the total amount of plants pollinated in the season



Global pollinator database - Boreux & Klein - Figshare Dataset
<https://doi.org/10.6084/m9.figshare.9980471.v1>

```
[491]: fig, ((ax1, ax2), (ax3, ax4)) = plt.pyplot.subplots(2, 2, figsize=(15,15))

fig.suptitle('Herbaceous plants visited by pollinators during each season',
             ↪fontsize=25, y=0.95)
ax1.set_title('Spring')
ax2.set_title('Summer')
ax3.set_title('Autumn')
```



```

ax4.set_title('Winter')

total_spring_pollinated_crops = len(GPD_herbaceous_seasons_dataset.loc[
    GPD_herbaceous_seasons_dataset['spring']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())
total_summer_pollinated_crops = len(GPD_herbaceous_seasons_dataset.loc[
    GPD_herbaceous_seasons_dataset['summer']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())
total_autumn_pollinated_crops = len(GPD_herbaceous_seasons_dataset.loc[
    GPD_herbaceous_seasons_dataset['autumn']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())
total_winter_pollinated_crops = len(GPD_herbaceous_seasons_dataset.loc[
    GPD_herbaceous_seasons_dataset['winter']\
    == 1, ['guild', 'crop']].value_counts(sort_
    ↪= False \
                                ).unstack(level = 1).count())

ax1.set_ylim(ymax= total_spring_pollinated_crops +
    ↪total_spring_pollinated_crops/10 )
ax2.set_ylim(ymax= total_summer_pollinated_crops +
    ↪total_summer_pollinated_crops/10 )
ax3.set_ylim(ymax= total_autumn_pollinated_crops +
    ↪total_autumn_pollinated_crops/10 )
ax4.set_ylim(ymax= total_winter_pollinated_crops +
    ↪total_winter_pollinated_crops/10 )

bar_number = len(GPD_seasons_dataset['guild'].unique().to_list())

width = 0.10

plt.pyplot.axes(ax1)
plt.pyplot.bar((bar_number-1)/2, total_spring_pollinated_crops, \
    color = 'gray', width = bar_number, edgecolor = 'black',
    ↪label='Total',\
    hatch='..')
herbaceous_spring_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax1,
    ↪color = 'green')

plt.pyplot.axes(ax2)

```

```

plt.pyplot.bar((bar_number-1)/2, total_summer_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black', \
               ↪label='Total', \
               hatch='..')
herbaceous_summer_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax2, \
               ↪color = 'yellow')

plt.pyplot.axes(ax3)
plt.pyplot.bar((bar_number-1)/2, total_autumn_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black', \
               ↪label='Total', \
               hatch='..')
herbaceous_autumn_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax3, \
               ↪color = 'orange')

plt.pyplot.axes(ax4)
plt.pyplot.bar((bar_number-1)/2, total_winter_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black', \
               ↪label='Total', \
               hatch='..')
herbaceous_winter_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax4, \
               ↪color = 'blue')

#plt.pyplot.legend()

fig.text(0.20, 0.90, 'Gray dotted background indicate the total amount of \
               ↪plants pollinated in the season', \
               fontsize=15)

fig.text(0.30, 0.025, 'Global pollinator database - Boreux & Klein - Figshare \
               ↪Dataset', \
               fontsize=15)
fig.text(0.40, 0.01, 'https://doi.org/10.6084/m9.figshare.9980471.v1', \
               ↪fontsize=10)

fig.tight_layout(pad=5)
#plt.pyplot.margins(2000)

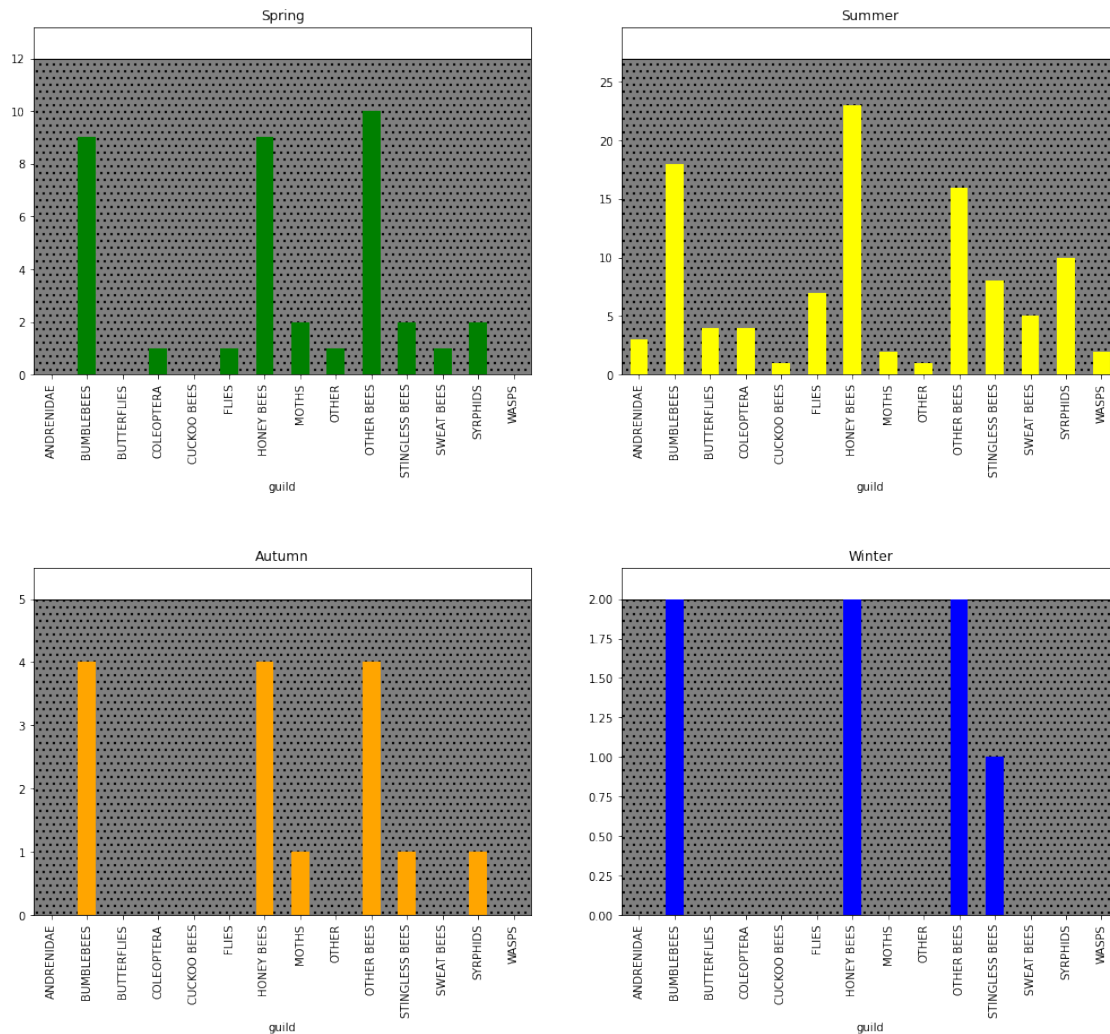
plt.pyplot.savefig('Images/Seasonal herbaceous plants for pollinators.png', \
               ↪dpi=150)
plt.pyplot.savefig('Images/Seasonal herbaceous plants for pollinators.jpg', \
               ↪dpi=150)

```

```
plt.pyplot.show()
```

Herbaceous plants visited by pollinators during each season

Gray dotted background indicate the total amount of plants pollinated in the season



Global pollinator database - Boreux & Klein - Figshare Dataset
<https://doi.org/10.6084/m9.figshare.9980471.v1>

```
[482]: GPD_seasons_dataset.loc[GPD_seasons_dataset['spring'] == 1, \
      'guild'].unique().to_list()
```

```
[482]: ['ANDRENIDAE',
      'BUMBLEBEES',
      'BUTTERFLIES',
      'COLEOPTERA',
      'CUCKOO BEES',
      'FLIES',
      'HONEY BEES',
```

```
'MOTHS',
'OTHER',
'OTHER BEES',
'STINGLESS BEES',
'SWEAT BEES',
'SYRPHIDS',
'WASPS']
```

```
[489]: GPD_honeybees_count_serie = GPD_seasons_dataset.
↳loc[(GPD_seasons_dataset['spring'] == 1) & \
      (GPD_seasons_dataset['guild'] == 'HONEY BEES') , \
      'visitor'].value_counts()
```

```
[484]: len(GPD_honeybees_count_serie[GPD_honeybees_count_serie != 0].to_list())
```

```
[484]: 8
```

```
[488]: GPD_honeybees_count_serie[GPD_honeybees_count_serie != 0]
```

```
[488]: Apis_mellifera          30
Apis_cerana                 12
Apis_florea                 11
Apis_dorsata                11
Apis_cerana_indica          4
Apis_mellifera_scutellata   4
Apis_mellifera_ligustica    1
Apis_mellifera_carnica      1
Name: visitor, dtype: int64
```

OK... we have a problem: seems that the authors put together different datasets without check for uniformity of data. We have the same pollinators named in different way and that alterate the meaning of the previous plots

```
[493]: GPD_bumblebees_count_serie = GPD_seasons_dataset.
↳loc[(GPD_seasons_dataset['spring'] == 1) & \
      (GPD_seasons_dataset['guild'] == 'BUMBLEBEES') , \
      'visitor'].value_counts()
GPD_bumblebees_count_serie[GPD_bumblebees_count_serie != 0]
```

```
[493]: Bombus_terrestris        12
Bombus_pascuorum          8
Bombus_impatiens          6
Bombus_pratorum           5
Bombus_hortorum           4
Bombus_vagans             4
Bombus_lapidarius         4
Bombus_griseocollis       3
```

Bombus_fervidus	3
Bombus_lucorum	3
Bombus_lapponicus	2
Bombus_balteatus	2
Bombus_bimaculatus	2
Bombus_vestalis	2
Bombus_hyperboreus	2
Bombus_hypnorum	2
Bombus_jonellus	2
Bombus_alpinus	2
Bombus_terricola	2
Bombus_ternarius	2
Bombus_hypocrita	1
Bombus_pensylvanicus	1
Bombus_sylvarum	1
Bombus_bohemicus	1
Bombus_sylvestris	1
Bombus_atratus	1
Bombus_vosnesenskii	1

Name: visitor, dtype: int64

```
[496]: GPD_otherbees_count_serie = GPD_seasons_dataset.
      ↪loc[(GPD_seasons_dataset['spring'] == 1) & \
          (GPD_seasons_dataset['guild'] == 'OTHER BEES') , \
          'visitor'].value_counts()
      GPD_otherbees_count_serie[GPD_otherbees_count_serie != 0]
```

Osmia_lignaria	7
Halictus_rubicundus	6
Osmia_cornuta	5
Melissodes_bimaculata	5
Halictus_confusus	5
Ceratina_dupla	4
Osmia_cornifrons	4
Lasioglossum_coriaceum	4
Ceratina_smaragdula	4
Lasioglossum_versatum	4
Xylocopa_virginica	3
Augochlorella_aurata	3
Augochlora_pura	3
Megachile_lanata	3
Peponapis_pruinosa	3
Halictus_ligatus	3
Megachile_rotundata	3
Agapostemon_virescens	3
Xylocopa_fenestrata	3
Lasioglossum_pilosum	3

Lasioglossum_cressonii	3
Xylocopa_aestuans	3
Anthophora_plumipes	2
Nomioides	2
Xylocopa_violacea	2
Augochloropsis_metallica	2
Osmia_lignaria_propinqua	2
Osmia_rufa	2
Megachile_frontalis	2
Megachile_centuncularis	2
Megachile_brevis	2
Anthidium_manicatum	2
Osmia_bicornis	2
Macropis_fulvipes	2
Megachile_mendica	2
Ceratina_cucurbitina	2
Osmia_bicolor	1
Ceratina_chalcites	1
Osmia_gallarum	1
Xylocopa_valga	1
Halictus_tripartitus	1
Melissodes_agilis	1
Megachile_ligniseca	1
Colletes_cunicularius	1
Megachile_pilidens	1
Megachile_apicalis	1
Osmia_aurulenta	1
Megachile_alpicola	1
Megachile_versicolor	1
Megachile_addenda	1
Heriades_truncorum	1

Name: visitor, dtype: int64

```
[495]: GPD_stinglessbees_count_serie = GPD_seasons_dataset.
        ↪loc[(GPD_seasons_dataset['spring'] == 1) & \
              (GPD_seasons_dataset['guild'] == 'STINGLESS BEES') , \
              'visitor'].value_counts()
        GPD_stinglessbees_count_serie[GPD_stinglessbees_count_serie != 0]
```

Trigona_fulviventris	5
Partamona_bilineata	4
Nannotrigona_perilampoides	4
Plebeia_frontalis	3
Trigona_nigerrima	2
Tetragonisca_angustula	2
Trigona_spinipes	1
Trigona_amalthea	1

```
Melipona_quadrifasciata      1
Name: visitor, dtype: int64
```

```
[498]: GPD_sweatbees_count_serie = GPD_seasons_dataset.
        ↪loc[(GPD_seasons_dataset['spring'] == 1) & \
              (GPD_seasons_dataset['guild'] == 'SWEAT BEES') , \
              'visitor'].value_counts()
        GPD_sweatbees_count_serie[GPD_sweatbees_count_serie != 0]
```

```
[498]: Lasioglossum_puncticolle      2
        Halictus_scabiosae           2
        Lasioglossum_reticulatum     1
        Lasioglossum_parvulum        1
        Sphecodes_niger              1
        Lasioglossum_subhirtum       1
        Sphecodes_monilicornis       1
        Lasioglossum_pygmaeum        1
        Lasioglossum_punctatissimum  1
        Lasioglossum_politum         1
        Sphecodes_majalis            1
        Lasioglossum_pauxillum       1
        Lasioglossum_pauperatum      1
        Lasioglossum_morio           1
        Lasioglossum_minutulum       1
        Lasioglossum_minutissimum    1
        Lasioglossum_malachurum      1
        Lasioglossum_lineare         1
        Halictus_tumulorum           1
        Lasioglossum_laticeps        1
        Lasioglossum_glabriusculum   1
        Sphecodes_longulus           1
        Sphecodes_albilabris         1
        Lasioglossum_calceatum       1
        Lasioglossum_xanthopus       1
        Name: visitor, dtype: int64
```

So let's now check for the pollinators with at least 2 " " and then we will check if exist the less specific definition with only one ""

```
[591]: visitor_subsp_check_list = GPD_seasons_dataset.visitor.str.contains(r".*_.*_.
        ↪*").to_list()
        visitor_subsp_indexes = []
        for index, check in enumerate(visitor_subsp_check_list):
            if check:
                visitor_subsp_indexes.append(index)
        len(visitor_subsp_indexes)
```

[591]: 19

Apparently we have 19 visitor that could be considered as duplicated values. To be honest seems not such a big alteration of the results of the previous plots... Let's check over if that is the situation: for each of these 19 we also have its generic definition in the dataset?

```
[657]: visitor_subsp_list = GPD_seasons_dataset.iloc[visitor_subsp_indexes,].visitor
visitor_subsp_list
```

```
[657]: 296      Apis_cerana_indica
315      Apis_mellifera_scutellata
320      Apis_mellifera_scutellata
322      Apis_cerana_indica
347      Apis_mellifera_scutellata
355      Apis_mellifera_ligustica
360      Apis_cerana_indica
376      Apis_mellifera_carnica
377      Apis_cerana_indica
378      Apis_cerana_indica
380      Apis_cerana_indica
381      Apis_mellifera_ligustica
399      Apis_mellifera_scutellata
406      Apis_cerana_indica
416      Apis_cerana_indica
421      Apis_mellifera_scutellata
567      Osmia_lignaria_propinqua
587      Osmia_lignaria_propinqua
638      Trigona_fulviventris_guianae
Name: visitor, dtype: category
Categories (254, object): ['Adalia_decompunctata',
'Agapanthia_villosoviridescens', 'Agapostemon_virescens', 'Aglais_urticae', ...,
'Xylocopa_hottentotta', 'Xylocopa_valga', 'Xylocopa_violacea',
'Xylocopa_virginica']
```

```
[663]: # now let's remove duplicated values
visitor_subsp_list = pd.DataFrame(visitor_subsp_list).visitor.unique().tolist()
visitor_subsp_list
```

```
[663]: ['Apis_cerana_indica',
'Apis_mellifera_scutellata',
'Apis_mellifera_ligustica',
'Apis_mellifera_carnica',
'Osmia_lignaria_propinqua',
'Trigona_fulviventris_guianae']
```

```
[622]: # Let's create a list removing characters from the last '_' to the end for each
↪ element in
```



```

# the visitor subspecies's list
visitor_rel_sp_list=[]
for visitor in visitor_subsp_list:
    visitor_rel_sp_list.append(re.search('.*_.*_', visitor).group()[:-1])
# now let's remove duplicated values
visitor_rel_sp_list = pd.DataFrame(visitor_rel_sp_list)[0].unique().tolist()
visitor_rel_sp_list

```

```
[622]: ['Apis_cerana', 'Apis_mellifera', 'Osmia_lignaria', 'Trigona_fulviventris']
```

```

[643]: #Now let's check that the species related to the selected subspecies are in the
        ↪dataset

for specie in visitor_rel_sp_list:
    print(specie)
    check_list = GPD_seasons_dataset.visitor.str.contains(fr"{specie}")
    print(check_list[check_list != False].count())

```

```

Apis_cerana
28
Apis_mellifera
75
Osmia_lignaria
10
Trigona_fulviventris
7

```

We should check also the crops

```

[648]: crop_subsp_check_list = GPD_seasons_dataset.crop.str.contains(r".*_.*_.*").
        ↪to_list()
crop_subsp_indexes = []
for index, check in enumerate(crop_subsp_check_list):
    if check:
        crop_subsp_indexes.append(index)
len(crop_subsp_indexes)

```

```
[648]: 5
```

```

[649]: crop_subsp_list = GPD_seasons_dataset.iloc[crop_subsp_indexes,].crop
crop_subsp_list

```

```

[649]: 191    Vicia_faba_major
      192    Vicia_faba_major
      193    Vicia_faba_major
      423    Vicia_faba_major
      629    Vicia_faba_major
      Name: crop, dtype: category

```

```
Categories (78, object): ['Allium_cepa', 'Allium_oleraceum', 'Amomum_subulatum',  
'Anacardium_occidentale', ..., 'Vicia_faba_major', 'Vigna_unguiculata',  
'Vitis_vinifera', 'Ziziphus_mauritiana']
```

```
[651]: crop_rel_sp = re.search('.*_*_', crop_subsp_list[191]).group()[:-1]  
crop_rel_sp
```

```
[651]: 'Vicia_faba'
```

```
[652]: check_list = GPD_seasons_dataset.crop.str.contains(fr"{crop_rel_sp}")  
check_list[check_list != False].count()
```

```
[652]: 14
```

Let's uniform the data overriding the subspecies values with related species values

```
[655]: # replace crop  
GPD_seasons_dataset_sp = GPD_seasons_dataset.replace(crop_subsp_list[191],  
↳ crop_rel_sp)  
check_list = GPD_seasons_dataset_sp.crop.str.  
↳ contains(fr"{crop_subsp_list[191]}")  
check_list[check_list != False].count()
```

```
[655]: 0
```

```
[665]: # replace visitor  
for subsp in visitor_subsp_list:  
    GPD_seasons_dataset_sp = GPD_seasons_dataset_sp.replace(subsp ,\  
        re.search('.*_*_', subsp).group()[:-1])  
  
    print(subsp)  
    print(re.search('.*_*_', subsp).group()[:-1])  
    check_list = GPD_seasons_dataset_sp.visitor.str.contains(fr"{subsp}")  
    print(check_list[check_list != False].count())
```

```
Apis_cerana_indica  
Apis_cerana  
0  
Apis_mellifera_scutellata  
Apis_mellifera  
0  
Apis_mellifera_ligustica  
Apis_mellifera  
0  
Apis_mellifera_carnica  
Apis_mellifera  
0  
Osmia_lignaria_propinqua
```

```

Osmia_lignaria
0
Trigona_fulviventris_guianae
Trigona_fulviventris
0

```

```

[675]: GPD_seasons_dataset_sp.
      ↪to_pickle(GPD_F_dataset_directory+"GPD_seasons_dataset_sp.pkl")

```

1.5.1 Year distribution - post subspecies uniformation - Starting point

```

[676]: GPD_seasons_dataset_sp = pd.
      ↪read_pickle(GPD_F_dataset_directory+"GPD_seasons_dataset_sp.pkl")

```

```

[679]: GPD_seasons_dataset_sp.describe

```

```

[679]: <bound method NDFrame.describe of
corolla  colour nectar  \
0  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE  white  yes
1  Vaccinium_corymbosum  arboreous  sprisum  CAMPANULATE  white  yes
2      Brassica_napus  herbaceous  summer  OPEN  yellow  yes
3      Brassica_napus  herbaceous  summer  OPEN  yellow  yes
4      Brassica_napus  herbaceous  summer  OPEN  yellow  yes
..      ...
774  Allium_oleraceum  herbaceous  summer  CAMPANULATE  purple  yes
775  Jatropha_curcas  arboreous  spriaut  OPEN  green  yes
776  Malus_domestica  arboreous  spring  OPEN  white  yes
777  Phaseolus_coccineus  herbaceous  summer  OPEN  white  yes
778  Capparis_spinosa  arboreous  summer  OPEN  white  yes

      b.system s.pollination inflorescence composite  \
0      insects      no      yes      no
1      insects      no      yes      no
2  wind/insects      no      yes      no
3  wind/insects      no      yes      no
4  wind/insects      no      yes      no
..      ...
774  insects      no      yes      no
775  insects      no      yes      no
776  insects      no      yes      no
777  insects      no      yes      no
778  insects      no  solitary      no

      visitor      guild sociality      feeding  spring  \
0  Andrena_wilkella  ANDRENIDAE      no  oligolectic      1
1  Andrena_barbilabris  ANDRENIDAE      no  polylectic      1
2  Andrena_cineraria  ANDRENIDAE      no  polylectic      0

```

3	Andrena_flavipes	ANDRENIDAE	no	polylectic	0
4	Andrena_gravida	ANDRENIDAE	no	polylectic	0
..
774	Dolichovespula_saxonica	WASPS	yes	polylectic	0
775	Bembecinus_tridens	WASPS	no	undefined	1
776	Vespula_vulgaris	WASPS	yes	polylectic	1
777	Philanthus_triangulum	WASPS	no	polylectic	0
778	Bembecinus_tridens	WASPS	no	undefined	0

	summer	autumn	winter
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
..
774	1	0	0
775	1	1	0
776	0	0	0
777	1	0	0
778	1	0	0

[779 rows x 18 columns]>

OK, now let's reproduce the previous plots

```
[669]: plt.pyplot.axes(projection = 'polar')
for unique_crop_index in range(0, len(GPD_seasons_dataset_sp.crop.unique())):
    crop = GPD_seasons_dataset_sp.crop.unique()[unique_crop_index]
    season_period = GPD_seasons_dataset_sp.loc[
        GPD_seasons_dataset_sp['crop'] == crop, 'season'].
        to_list()[0]
    angle_range = np.zeros(1)
    season_color = 'gray'
    if season_period == 'year':
        angle_range = np.arange(0, (2 * np.pi), 0.1)
        season_color = 'green'
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'sprisum':
        angle_range = np.arange(np.pi/4, -3*np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'spriaut':
        angle_range = np.arange(np.pi/4, -5*np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'winspring':
```

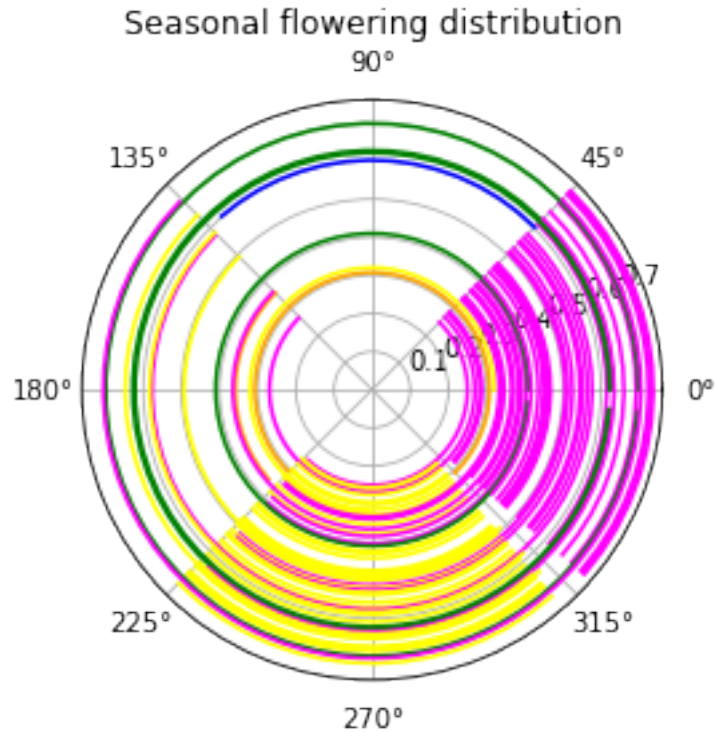
```

    angle_range = np.arange(np.pi/4, 3*np.pi/4, 0.1)
    season_color = 'blue'
elif season_period == 'sumspri': #is not year?
    angle_range = np.arange(0, 6*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'summer':
    angle_range = np.arange(5*np.pi/4, 7*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'sumaut':
    angle_range = np.arange(3*np.pi/4, 7*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'autumn':
    angle_range = np.arange(3*np.pi/4, 5*np.pi/4, 0.1)
    season_color = 'orange'
elif season_period == 'autspri':
    angle_range = np.arange(-1*np.pi/4, 5*np.pi/4, 0.1)
    season_color = 'orange'
elif season_period == 'winter':
    angle_range = np.arange(1*np.pi/4, 3*np.pi/4, 0.1)
    season_color = 'blue'

if season_period != 'undefined':
    positions = np.full(shape = angle_range.shape, \
                        fill_value = (0.25 + unique_crop_index/(78*2)))
    plt.pyplot.polar(angle_range, positions, color = season_color)

plt.pyplot.title('Seasonal flowering distribution')
plt.pyplot.show()

```



```
[672]: fig, (ax1, ax2) = plt.pyplot.subplots(1, 2, subplot_kw = dict(polar = True),
↳ figsize=(10,10))

#flowering cycles differentiated by the type of plant (herbaceous or arboreous)
for unique_crop_index in range(0, len(GPD_seasons_dataset_sp.crop.unique())):
    crop = GPD_seasons_dataset_sp.crop.unique()[unique_crop_index]
    season_period = GPD_seasons_dataset_sp.loc[
        GPD_seasons_dataset_sp['crop'] == crop, 'season'].
↳to_list()[0]
    first_element_index = GPD_seasons_dataset_sp.loc[
        GPD_seasons_dataset_sp['crop'] == crop, :].index.
↳to_list()[0]

    angle_range = np.zeros(1)
    season_color = 'gray'
    if season_period == 'year':
        angle_range = np.arange(0, (2 * np.pi), 0.1)
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -np.pi/4, -0.1)
    elif season_period == 'sprisum':
        angle_range = np.arange(np.pi/4, -3*np.pi/4, -0.1)
    elif season_period == 'spriaut':
        angle_range = np.arange(np.pi/4, -5*np.pi/4, -0.1)
```

```

elif season_period == 'winspring':
    angle_range = np.arange(np.pi/4, 3*np.pi/4, 0.1)
elif season_period == 'sumspri': #is not year?
    angle_range = np.arange(0, 6*np.pi/4, 0.1)
elif season_period == 'summer':
    angle_range = np.arange(5*np.pi/4, 7*np.pi/4, 0.1)
elif season_period == 'sumaut':
    angle_range = np.arange(3*np.pi/4, 7*np.pi/4, 0.1)
elif season_period == 'autumn':
    angle_range = np.arange(3*np.pi/4, 5*np.pi/4, 0.1)
elif season_period == 'autspri':
    angle_range = np.arange(-1*np.pi/4, 5*np.pi/4, 0.1)
elif season_period == 'winter':
    angle_range = np.arange(1*np.pi/4, 3*np.pi/4, 0.1)

if season_period != 'undefined':
    positions = np.full(shape = angle_range.shape, \
                        fill_value = (0.25 + unique_crop_index/(78*2)))
    if GPD_seasons_dataset_sp.loc[first_element_index, 'type'] == '
↳'herbaceous':
        ax1.plot(angle_range, positions, color = '
↳colours_list[first_element_index] )
    else:
        ax2.plot(angle_range, positions, color = '
↳colours_list[first_element_index] )

ax1.set_title('Herbaceous')
ax1.set_facecolor('#D3D3D3')
ax2.set_title('Arboreous')
ax2.set_facecolor('#D3D3D3')

fig.suptitle('Seasonal flowering distribution', fontsize=25, y=0.8)
fig.text(0.35, 0.2, 'Global pollinator database', fontsize=16)
fig.text(0.36, 0.15, 'Boreux & Klein - Figshare Dataset', fontsize=12)
fig.text(0.37, 0.1, 'https://doi.org/10.6084/m9.figshare.9980471.v1',
↳fontsize=8)

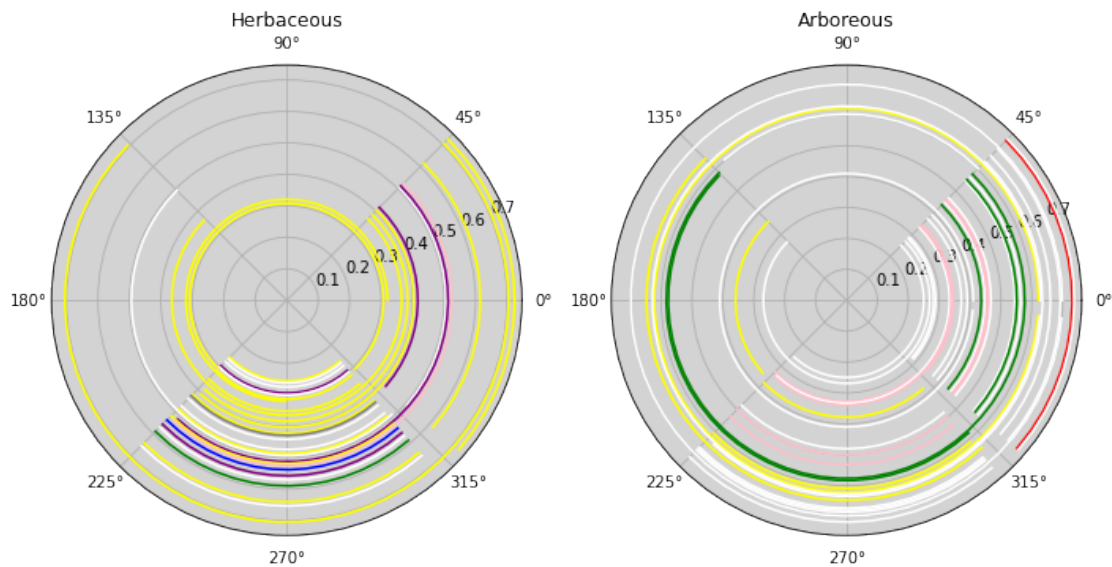
#set spacing between plots
fig.tight_layout()

plt.pyplot.savefig('Images/Seasonal flowering distribution.png', dpi=150)
plt.pyplot.savefig('Images/Seasonal flowering distribution.jpg', dpi=150)

plt.pyplot.show()

```

Seasonal flowering distribution



Global pollinator database

Boreux & Klein - Figshare Dataset

<https://doi.org/10.6084/m9.figshare.9980471.v1>

```
[674]: plt.pyplot.axes(projection = 'polar')
for unique_visitor_index in range(0, len(GPD_seasons_dataset_sp.visitor.
↳unique())):
    visitor = GPD_seasons_dataset_sp.visitor.unique()[unique_visitor_index]
    season_period = GPD_seasons_dataset_sp.loc[
        GPD_seasons_dataset_sp['visitor'] == visitor, 'season'].
↳to_list()[0]
    angle_range = np.zeros(1)
    season_color = 'gray'
    if season_period == 'year':
        angle_range = np.arange(0, (2 * np.pi), 0.1)
        season_color = 'green'
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -3*np.pi/4, -0.1)
        season_color = 'magenta'
    elif season_period == 'spring':
        angle_range = np.arange(np.pi/4, -5*np.pi/4, -0.1)
        season_color = 'magenta'
```



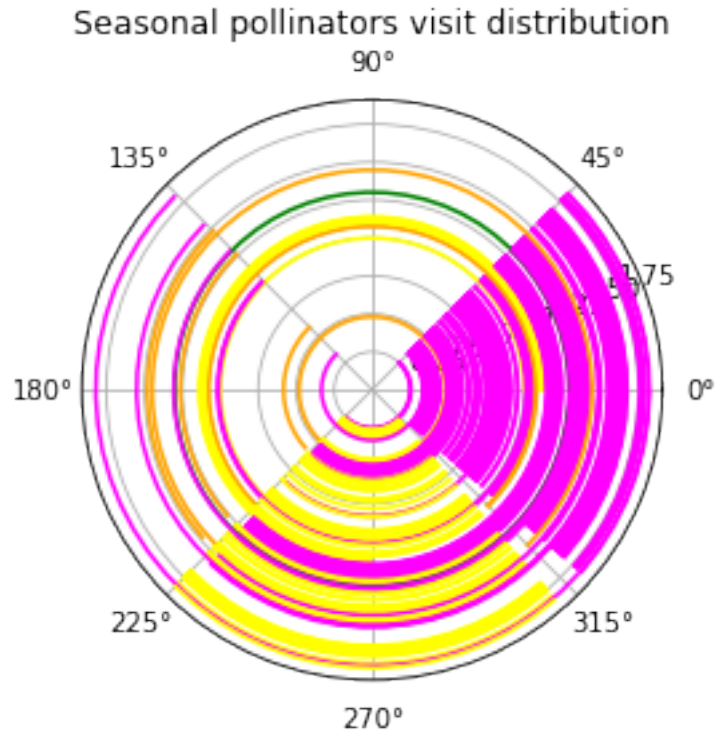
```

elif season_period == 'winspring':
    angle_range = np.arange(np.pi/4, 3*np.pi/4, 0.1)
    season_color = 'blue'
elif season_period == 'sumspri': #is not year?
    angle_range = np.arange(0, 6*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'summer':
    angle_range = np.arange(5*np.pi/4, 7*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'sumaut':
    angle_range = np.arange(3*np.pi/4, 7*np.pi/4, 0.1)
    season_color = 'yellow'
elif season_period == 'autumn':
    angle_range = np.arange(3*np.pi/4, 5*np.pi/4, 0.1)
    season_color = 'orange'
elif season_period == 'autspri':
    angle_range = np.arange(-1*np.pi/4, 5*np.pi/4, 0.1)
    season_color = 'orange'
elif season_period == 'winter':
    angle_range = np.arange(1*np.pi/4, 3*np.pi/4, 0.1)
    season_color = 'blue'

if season_period != 'undefined':
    positions = np.full(shape = angle_range.shape, \
                        fill_value = (0.25 + unique_visitor_index/(78*2)))
    plt.pyplot.polar(angle_range, positions, color = season_color)

plt.pyplot.title('Seasonal pollinators visit distribution')
plt.pyplot.show()

```



```
[693]: GPD_herbaceous_seasons_dataset_sp = GPD_seasons_dataset_sp.loc[ \
                                             GPD_seasons_dataset_sp['type'] ==
↳ 'herbaceous', : ]
GPD_arboreous_seasons_dataset_sp = GPD_seasons_dataset_sp.loc[ \
                                   GPD_seasons_dataset_sp['type'] ==
↳ 'arboreous', : ]
```

```
[695]: spring_guild_crop_dataset = GPD_seasons_dataset_sp.
↳ loc[GPD_seasons_dataset_sp['spring'] == 1, \
      ['guild', 'crop']].value_counts(sort =
↳ False\
                                   ).unstack(level = 0)
summer_guild_crop_dataset = GPD_seasons_dataset_sp.
↳ loc[GPD_seasons_dataset_sp['summer'] == 1, \
      ['guild', 'crop']].value_counts(sort =
↳ False\
                                   ).unstack(level = 0)
autumn_guild_crop_dataset = GPD_seasons_dataset_sp.
↳ loc[GPD_seasons_dataset_sp['autumn'] == 1, \
      ['guild', 'crop']].value_counts(sort =
↳ False\
                                   ).unstack(level = 0)
```

```

winter_guild_crop_dataset = GPD_seasons_dataset_sp.
    ↳loc[GPD_seasons_dataset_sp['winter'] == 1, \
        ['guild', 'crop']].value_counts(sort =_
    ↳False\
        ).unstack(level = 0)
year_guild_crop_dataset = GPD_seasons_dataset_sp[['guild', 'crop']].
    ↳value_counts(\
        sort = False).unstack(level = 0)

print(len(spring_guild_crop_dataset.count()))
print(len(summer_guild_crop_dataset.count()))
print(len(autumn_guild_crop_dataset.count()))
print(len(winter_guild_crop_dataset.count()))
print(len(year_guild_crop_dataset.count()))

```

14
14
12
6
14

```

[696]: arboreous_spring_guild_crop_dataset = GPD_arboreous_seasons_dataset_sp.loc[\
    ↳
    ↳GPD_arboreous_seasons_dataset_sp['spring'] == 1, \
        ['guild', 'crop']].value_counts(sort =_
    ↳False\
        ).unstack(level = 0)
arboreous_summer_guild_crop_dataset = GPD_arboreous_seasons_dataset_sp.loc[\
    ↳
    ↳GPD_arboreous_seasons_dataset_sp['summer'] == 1, \
        ['guild', 'crop']].value_counts(sort =_
    ↳False\
        ).unstack(level = 0)
arboreous_autumn_guild_crop_dataset = GPD_arboreous_seasons_dataset_sp.loc[\
    ↳
    ↳GPD_arboreous_seasons_dataset_sp['autumn'] == 1, \
        ['guild', 'crop']].value_counts(sort =_
    ↳False\
        ).unstack(level = 0)
arboreous_winter_guild_crop_dataset = GPD_arboreous_seasons_dataset_sp.loc[\
    ↳
    ↳GPD_arboreous_seasons_dataset_sp['winter'] == 1, \
        ['guild', 'crop']].value_counts(sort =_
    ↳False\

```

```

        ).unstack(level = 0)
arboreous_year_guild_crop_dataset = GPD_arboreous_seasons_dataset_sp[['guild', \
↪ 'crop']].value_counts(\
        sort = False).unstack(level = 0)

herbaceous_spring_guild_crop_dataset = GPD_herbaceous_seasons_dataset_sp.loc[\
        ↪
↪ GPD_herbaceous_seasons_dataset_sp['spring'] == 1, \
        ['guild', 'crop']].value_counts(sort = \
↪ False\
        ).unstack(level = 0)
herbaceous_summer_guild_crop_dataset = GPD_herbaceous_seasons_dataset_sp.loc[\
        ↪
↪ GPD_herbaceous_seasons_dataset_sp['summer'] == 1, \
        ['guild', 'crop']].value_counts(sort = \
↪ False\
        ).unstack(level = 0)
herbaceous_autumn_guild_crop_dataset = GPD_herbaceous_seasons_dataset_sp.loc[\
        ↪
↪ GPD_herbaceous_seasons_dataset_sp['autumn'] == 1, \
        ['guild', 'crop']].value_counts(sort = \
↪ False\
        ).unstack(level = 0)
herbaceous_winter_guild_crop_dataset = GPD_herbaceous_seasons_dataset_sp.loc[\
        ↪
↪ GPD_herbaceous_seasons_dataset['winter'] == 1, \
        ['guild', 'crop']].value_counts(sort = \
↪ False\
        ).unstack(level = 0)
herbaceous_year_guild_crop_dataset = GPD_herbaceous_seasons_dataset_sp[\
        ['guild', 'crop']].value_counts(\
        sort = False).unstack(level = 0)

```

[694]: year_guild_crop_dataset

```

[694]: guild          ANDRENIDAE  BUMBLEBEES  BUTTERFLIES  COLEOPTERA  \
crop
Allium_cepa          NaN          NaN          1.0          1.0
Allium_oleraceum     NaN          1.0          1.0          NaN
Amomum_subulatum     NaN          NaN          NaN          NaN
Anacardium_occidentale NaN          NaN          NaN          NaN
Arachis_hypogaea     NaN          1.0          NaN          NaN
...                  ...          ...          ...          ...
Vicia_faba           NaN          5.0          NaN          1.0
Vicia_faba_major     NaN          3.0          NaN          NaN
Vigna_unguiculata    NaN          2.0          NaN          NaN

```

Vitis_vinifera	NaN	NaN	NaN	2.0
Ziziphus_mauritiana	NaN	NaN	NaN	NaN

guild	CUCKOO	BEES	FLIES	HONEY	BEES	MOTHS	OTHER	\
crop								
Allium_cepa		NaN	4.0		4.0	NaN	NaN	
Allium_oleraceum		NaN	NaN		NaN	1.0	NaN	
Amomum_subulatum		NaN	NaN		1.0	1.0	NaN	
Anacardium_occidentale		NaN	NaN		4.0	NaN	NaN	
Arachis_hypogaea		NaN	NaN		2.0	NaN	NaN	
...		
Vicia_faba		NaN	NaN		NaN	NaN	1.0	
Vicia_faba_major		NaN	NaN		1.0	NaN	NaN	
Vigna_unguiculata		NaN	1.0		1.0	NaN	NaN	
Vitis_vinifera		NaN	NaN		1.0	NaN	NaN	
Ziziphus_mauritiana		NaN	1.0		3.0	NaN	NaN	

guild	OTHER	BEES	STINGLESS	BEES	SWEAT	BEES	SYRPHIDS	\
crop								
Allium_cepa		3.0		NaN		NaN	4.0	
Allium_oleraceum		NaN		NaN		NaN	4.0	
Amomum_subulatum		1.0		NaN		NaN	1.0	
Anacardium_occidentale		NaN		1.0		NaN	NaN	
Arachis_hypogaea		6.0		1.0		1.0	NaN	
...		
Vicia_faba		2.0		NaN		NaN	NaN	
Vicia_faba_major		1.0		NaN		NaN	NaN	
Vigna_unguiculata		1.0		NaN		NaN	NaN	
Vitis_vinifera		NaN		NaN		NaN	3.0	
Ziziphus_mauritiana		NaN		NaN		NaN	NaN	

guild	WASPS
crop	
Allium_cepa	NaN
Allium_oleraceum	2.0
Amomum_subulatum	NaN
Anacardium_occidentale	NaN
Arachis_hypogaea	NaN
...	...
Vicia_faba	NaN
Vicia_faba_major	NaN
Vigna_unguiculata	NaN
Vitis_vinifera	NaN
Ziziphus_mauritiana	NaN

[78 rows x 14 columns]

```
[697]: arboreous_spring_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_spring_guild_crop_dataset
arboreous_summer_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_summer_guild_crop_dataset
arboreous_autumn_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_autumn_guild_crop_dataset
arboreous_winter_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_winter_guild_crop_dataset
arboreous_year_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        arboreous_year_guild_crop_dataset

herbaceous_spring_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_spring_guild_crop_dataset
herbaceous_summer_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_summer_guild_crop_dataset
herbaceous_autumn_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_autumn_guild_crop_dataset
herbaceous_winter_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_winter_guild_crop_dataset
herbaceous_year_guild_crop_dataset_full = year_guild_crop_dataset*0 + \
        herbaceous_year_guild_crop_dataset
```

```
[737]: fig, ((ax1, ax2), (ax3, ax4)) = plt.pyplot.subplots(2, 2, figsize=(15,15))

fig.suptitle('Number of herbaceous crops visited by pollinators during each_
↳season', fontsize=25, y=0.95)
ax1.set_title('Spring')
ax2.set_title('Summer')
ax3.set_title('Autumn')
ax4.set_title('Winter')

total_spring_pollinated_crops = len(GPD_herbaceous_seasons_dataset_sp.loc[\
        GPD_herbaceous_seasons_dataset_sp['spring']\
        == 1, ['guild', 'crop']].value_counts(sort_
↳= False \
        ).unstack(level = 1).count())
total_summer_pollinated_crops = len(GPD_herbaceous_seasons_dataset_sp.loc[\
        GPD_herbaceous_seasons_dataset_sp['summer']\
        == 1, ['guild', 'crop']].value_counts(sort_
↳= False \
        ).unstack(level = 1).count())
total_autumn_pollinated_crops = len(GPD_herbaceous_seasons_dataset_sp.loc[\
        GPD_herbaceous_seasons_dataset_sp['autumn']\
        == 1, ['guild', 'crop']].value_counts(sort_
↳= False \
        ).unstack(level = 1).count())
total_winter_pollinated_crops = len(GPD_herbaceous_seasons_dataset_sp.loc[\
```

```

GPD_herbaceous_seasons_dataset_sp['winter']\
== 1, ['guild', 'crop']].value_counts(sort_
↳= False \

                                ).unstack(level = 1).count())

ax1.set_ylim(ymax= total_spring_pollinated_crops +
↳total_spring_pollinated_crops/10 )
ax2.set_ylim(ymax= total_summer_pollinated_crops +
↳total_summer_pollinated_crops/10 )
ax3.set_ylim(ymax= total_autumn_pollinated_crops +
↳total_autumn_pollinated_crops/10 )
ax4.set_ylim(ymax= total_winter_pollinated_crops +
↳total_winter_pollinated_crops/10 )

bar_number = len(GPD_seasons_dataset_sp['guild'].unique().to_list())

width = 0.10

plt.pyplot.axes(ax1)
plt.pyplot.bar((bar_number-1)/2, total_spring_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',
↳label='Total',\
               hatch='..')
herbaceous_spring_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax1,
↳color = 'green')

plt.pyplot.axes(ax2)
plt.pyplot.bar((bar_number-1)/2, total_summer_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',
↳label='Total',\
               hatch='..')
herbaceous_summer_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax2,
↳color = 'yellow')

plt.pyplot.axes(ax3)
plt.pyplot.bar((bar_number-1)/2, total_autumn_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',
↳label='Total',\
               hatch='..')
herbaceous_autumn_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax3,
↳color = 'orange')

plt.pyplot.axes(ax4)
plt.pyplot.bar((bar_number-1)/2, total_winter_pollinated_crops, \

```

```

        color = 'gray', width = bar_number, edgecolor = 'black',\
        ↪label='Total',\
        hatch='..')
herbaceous_winter_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax4,\
        ↪color = 'blue')

plt.pyplot.legend()

fig.text(0.20, 0.90, 'Gray dotted background indicate the total amount of crops\
        ↪visited in the season',\
        fontsize=15)

fig.text(0.30, 0.025, 'Global pollinator database - Boreux & Klein - Figshare\
        ↪Dataset',\
        fontsize=15)
fig.text(0.40, 0.01, 'https://doi.org/10.6084/m9.figshare.9980471.v1',\
        ↪fontsize=10)

fig.tight_layout(pad=5)
plt.pyplot.margins(2000)

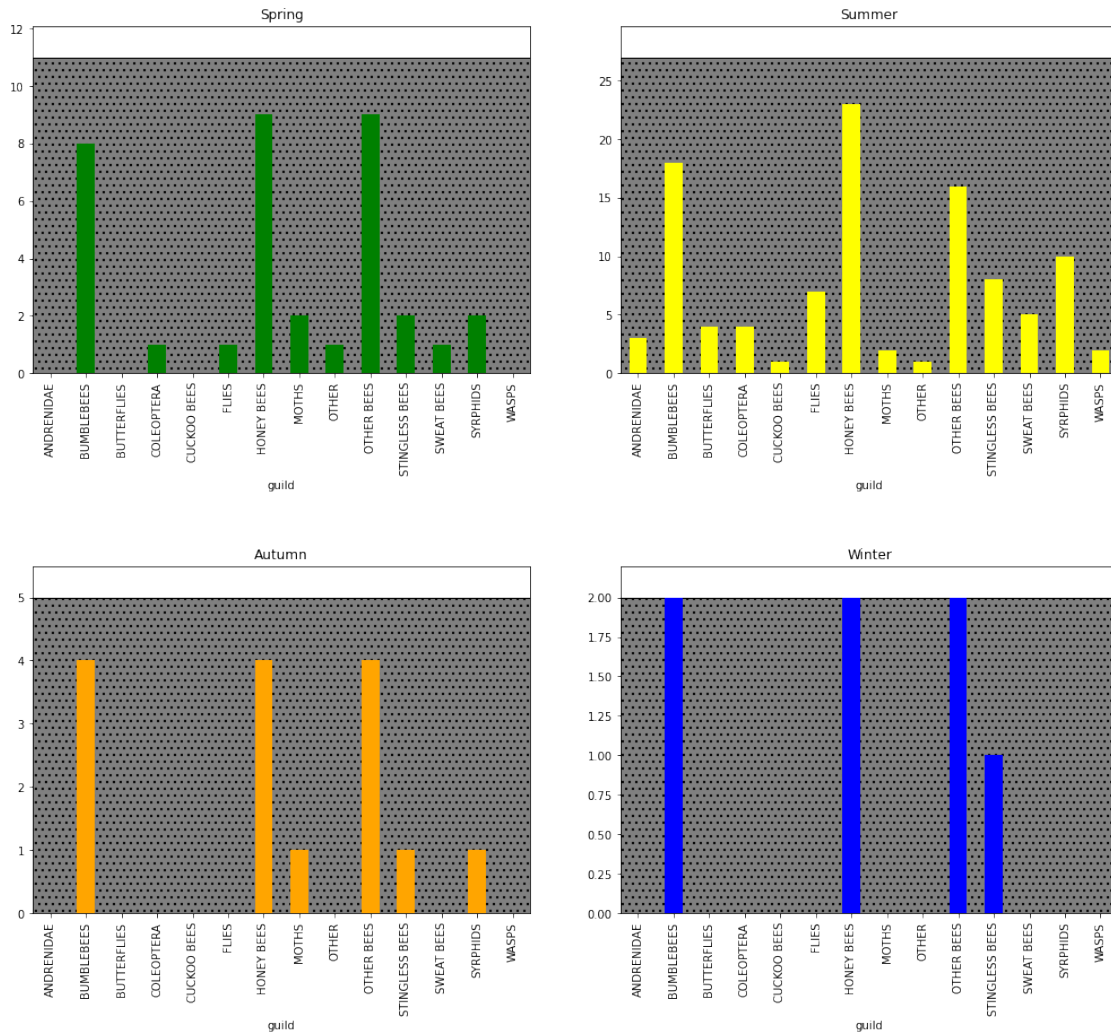
plt.pyplot.savefig('Images/Seasonal herbaceous plants for pollinators.png',\
        ↪dpi=150)
plt.pyplot.savefig('Images/Seasonal herbaceous plants for pollinators.jpg',\
        ↪dpi=150)

plt.pyplot.show()

```


Number of herbaceous crops visited by pollinators during each season

Gray dotted background indicate the total amount of crops visited in the season



Global pollinator database - Boreux & Klein - Figshare Dataset

<https://doi.org/10.6084/m9.figshare.9980471.v1>

```
[738]: fig, ((ax1, ax2), (ax3, ax4)) = plt.pyplot.subplots(2, 2, figsize=(15,15))

fig.suptitle('Number of arboreous crops visited by pollinators during each_
↪season', fontsize=25, y=0.95)
ax1.set_title('Spring')
ax2.set_title('Summer')
ax3.set_title('Autumn')
ax4.set_title('Winter')

total_spring_pollinated_crops = len(GPD_arboreous_seasons_dataset_sp.loc[\
                                     GPD_arboreous_seasons_dataset_sp['spring']\
```

```

                                == 1, ['guild', 'crop']].value_counts(sort_
↳= False \
                                ).unstack(level = 1).count())
total_summer_pollinated_crops = len(GPD_arboreous_seasons_dataset_sp.loc[
                                GPD_arboreous_seasons_dataset_sp['summer']\
                                == 1, ['guild', 'crop']].value_counts(sort_
↳= False \
                                ).unstack(level = 1).count())
total_autumn_pollinated_crops = len(GPD_arboreous_seasons_dataset_sp.loc[
                                GPD_arboreous_seasons_dataset_sp['autumn']\
                                == 1, ['guild', 'crop']].value_counts(sort_
↳= False \
                                ).unstack(level = 1).count())
total_winter_pollinated_crops = len(GPD_arboreous_seasons_dataset_sp.loc[
                                GPD_arboreous_seasons_dataset_sp['winter']\
                                == 1, ['guild', 'crop']].value_counts(sort_
↳= False \
                                ).unstack(level = 1).count())

ax1.set_ylim(ymax= total_spring_pollinated_crops +
↳total_spring_pollinated_crops/10 )
ax2.set_ylim(ymax= total_summer_pollinated_crops +
↳total_summer_pollinated_crops/10 )
ax3.set_ylim(ymax= total_autumn_pollinated_crops +
↳total_autumn_pollinated_crops/10 )
ax4.set_ylim(ymax= total_winter_pollinated_crops +
↳total_winter_pollinated_crops/10 )

bar_number = len(GPD_seasons_dataset['guild'].unique().to_list())

width = 0.10

plt.pyplot.axes(ax1)
plt.pyplot.bar((bar_number-1)/2, total_spring_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',
↳label='Total',\
               hatch='..')
arboreous_spring_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax1,
↳color = 'green')

plt.pyplot.axes(ax2)
plt.pyplot.bar((bar_number-1)/2, total_summer_pollinated_crops, \
               color = 'gray', width = bar_number, edgecolor = 'black',
↳label='Total',\
               hatch='..')

```

```

arboreous_summer_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax2,
    color = 'yellow')

plt.pyplot.axes(ax3)
plt.pyplot.bar((bar_number-1)/2, total_autumn_pollinated_crops, \
    color = 'gray', width = bar_number, edgecolor = 'black',
    label='Total',\
    hatch='..')
arboreous_autumn_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax3,
    color = 'orange')

plt.pyplot.axes(ax4)
plt.pyplot.bar((bar_number-1)/2, total_winter_pollinated_crops, \
    color = 'gray', width = bar_number, edgecolor = 'black',
    label='Total',\
    hatch='..')
arboreous_winter_guild_crop_dataset_full.count().plot(kind = 'bar', ax=ax4,
    color = 'blue')

#plt.pyplot.legend()

fig.text(0.20, 0.90, 'Gray dotted background indicate the total amount of
    plants visited in the season',\
    fontsize=15)

fig.text(0.30, 0.025, 'Global pollinator database - Boreux & Klein - Figshare
    Dataset',\
    fontsize=15)
fig.text(0.40, 0.01, 'https://doi.org/10.6084/m9.figshare.9980471.v1',
    fontsize=10)

fig.tight_layout(pad=5)
#plt.pyplot.margins(2000)

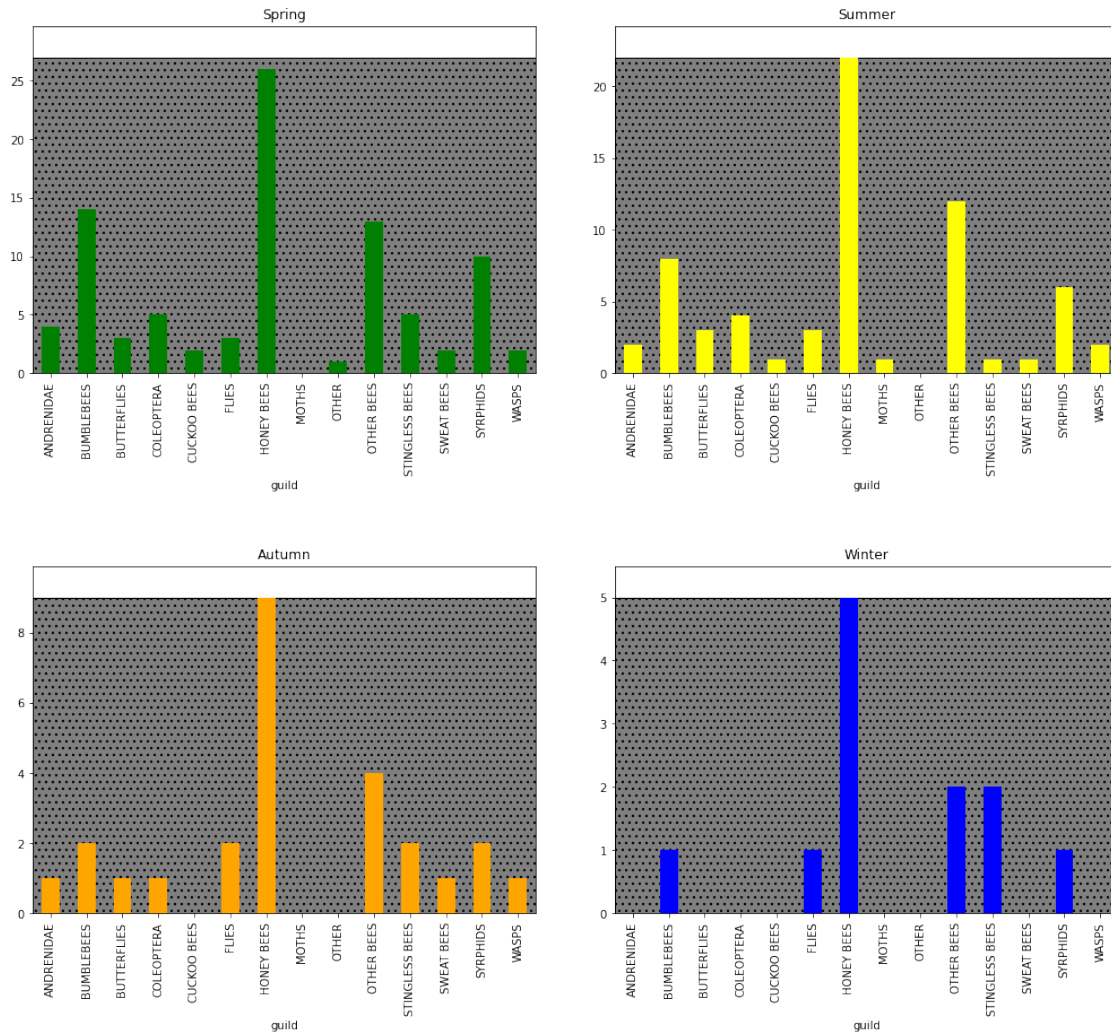
plt.pyplot.savefig('Images/Seasonal arboreous plants for pollinators.png',
    dpi=150)
plt.pyplot.savefig('Images/Seasonal arboreous plants for pollinators.jpg',
    dpi=150)

plt.pyplot.show()

```

Number of arboreous crops visited by pollinators during each season

Gray dotted background indicate the total amount of plants visited in the season



Global pollinator database - Boreux & Klein - Figshare Dataset

<https://doi.org/10.6084/m9.figshare.9980471.v1>

```
[759]: plt.pyplot.figure(figsize=(15,15))

bar_number = len(GPD_seasons_dataset_sp['guild'].unique().to_list())
x_range = np.arange(bar_number)
width = 0.20

plt.pyplot.bar(x_range + 1.5*width, herbaceous_year_guild_crop_dataset_full.
    ↪count(), \
                color = 'red', width = 4*width, edgecolor = 'black', \
    ↪label='Year')
plt.pyplot.bar(x_range , herbaceous_spring_guild_crop_dataset_full.count(), \
```

```

        color = 'green', width = width, edgecolor = 'black', hatch='/',\
        ↪label='Spring')
plt.pyplot.bar(x_range + width, herbaceous_summer_guild_crop_dataset_full.
        ↪count(),\
        color = 'yellow', width = width, edgecolor = 'black', hatch=□
        ↪' *', label='Summer')
plt.pyplot.bar(x_range + 2*width, herbaceous_autumn_guild_crop_dataset_full.
        ↪count(), color = 'orange',
        width = width, edgecolor = 'black', hatch='-', label='Autumn')
plt.pyplot.bar(x_range + 3*width, herbaceous_winter_guild_crop_dataset_full.
        ↪count(), color = 'blue',
        width = width, edgecolor = 'black', hatch='x', label='Winter')

plt.pyplot.xlabel("Pollinators guilds")
plt.pyplot.ylabel("Number of different herbaceous crops visited")
plt.pyplot.title("Number of different herbaceous crops visited by pollinators",\
        ↪fontsize=25)

plt.pyplot.xticks(x_range + width, GPD_seasons_dataset_sp['guild'].unique().
        ↪to_list(), \
        rotation = 'vertical')
plt.pyplot.legend()

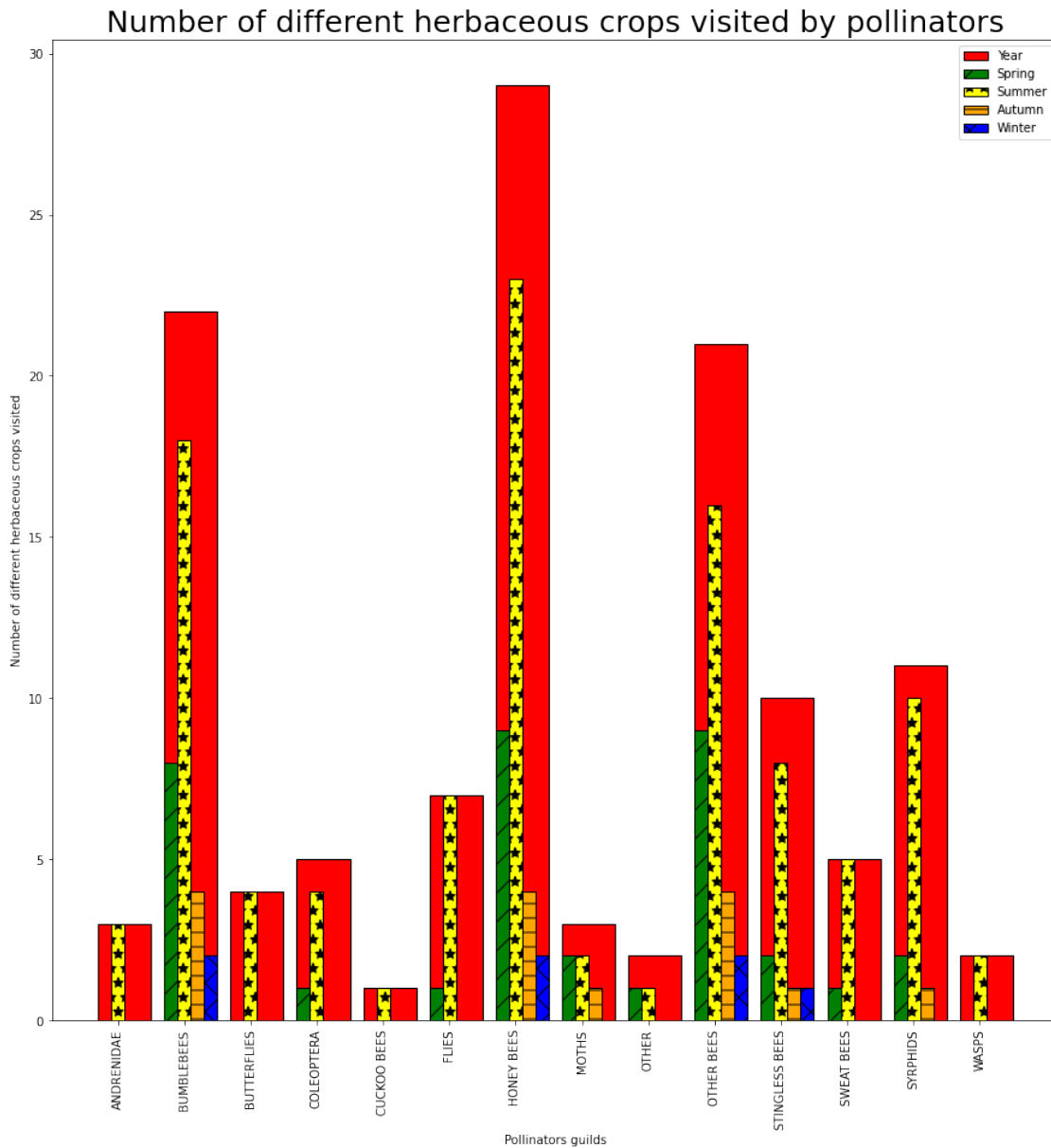
#fig.tight_layout()

plt.pyplot.text(3, -6, 'Global pollinator database - Boreux & Klein - Figshare\
        ↪Dataset',\
        fontsize=15)
plt.pyplot.text(5, -7.5, 'https://doi.org/10.6084/m9.figshare.9980471.v1',\
        ↪fontsize=10)

plt.pyplot.savefig('Images/Herbaceous plants visited by pollinators.png',\
        ↪dpi=150)
plt.pyplot.savefig('Images/Herbaceous plants visited by pollinators.jpg',\
        ↪dpi=150)

plt.pyplot.show()

```



Global pollinator database - Boreux & Klein - Figshare Dataset

<https://doi.org/10.6084/m9.figshare.9980471.v1>

```
[761]: plt.pyplot.figure(figsize=(15,15))

bar_number = len(GPD_seasons_dataset_sp['guild'].unique().to_list())
x_range = np.arange(bar_number)
width = 0.20
```

```

plt.pyplot.bar(x_range + 1.5*width, arboreous_year_guild_crop_dataset_full.
    ↪count(), \
                color = 'red', width = 4*width, edgecolor = 'black',
    ↪label='Year')
plt.pyplot.bar(x_range , arboreous_spring_guild_crop_dataset_full.count(), \
                color = 'green', width = width, edgecolor = 'black', hatch='/',
    ↪label='Spring')
plt.pyplot.bar(x_range + width, arboreous_summer_guild_crop_dataset_full.
    ↪count(), \
                color = 'yellow', width = width, edgecolor = 'black', hatch=
    ↪'*', label='Summer')
plt.pyplot.bar(x_range + 2*width, arboreous_autumn_guild_crop_dataset_full.
    ↪count(), \
                color = 'orange', width = width, edgecolor = 'black', hatch='-',
    ↪label='Autumn')
plt.pyplot.bar(x_range + 3*width, arboreous_winter_guild_crop_dataset_full.
    ↪count(), \
                color = 'blue', width = width, edgecolor = 'black', hatch='x',
    ↪label='Winter')

plt.pyplot.xlabel("Pollinators guilds")
plt.pyplot.ylabel("Number of different arboreous crops visited")
plt.pyplot.title("Number of different arboreous crops visited by pollinators",
    ↪fontsize=25)

plt.pyplot.xticks(x_range + width, GPD_seasons_dataset_sp['guild'].unique().
    ↪to_list(), \
                rotation = 'vertical')
plt.pyplot.legend()

#fig.tight_layout()

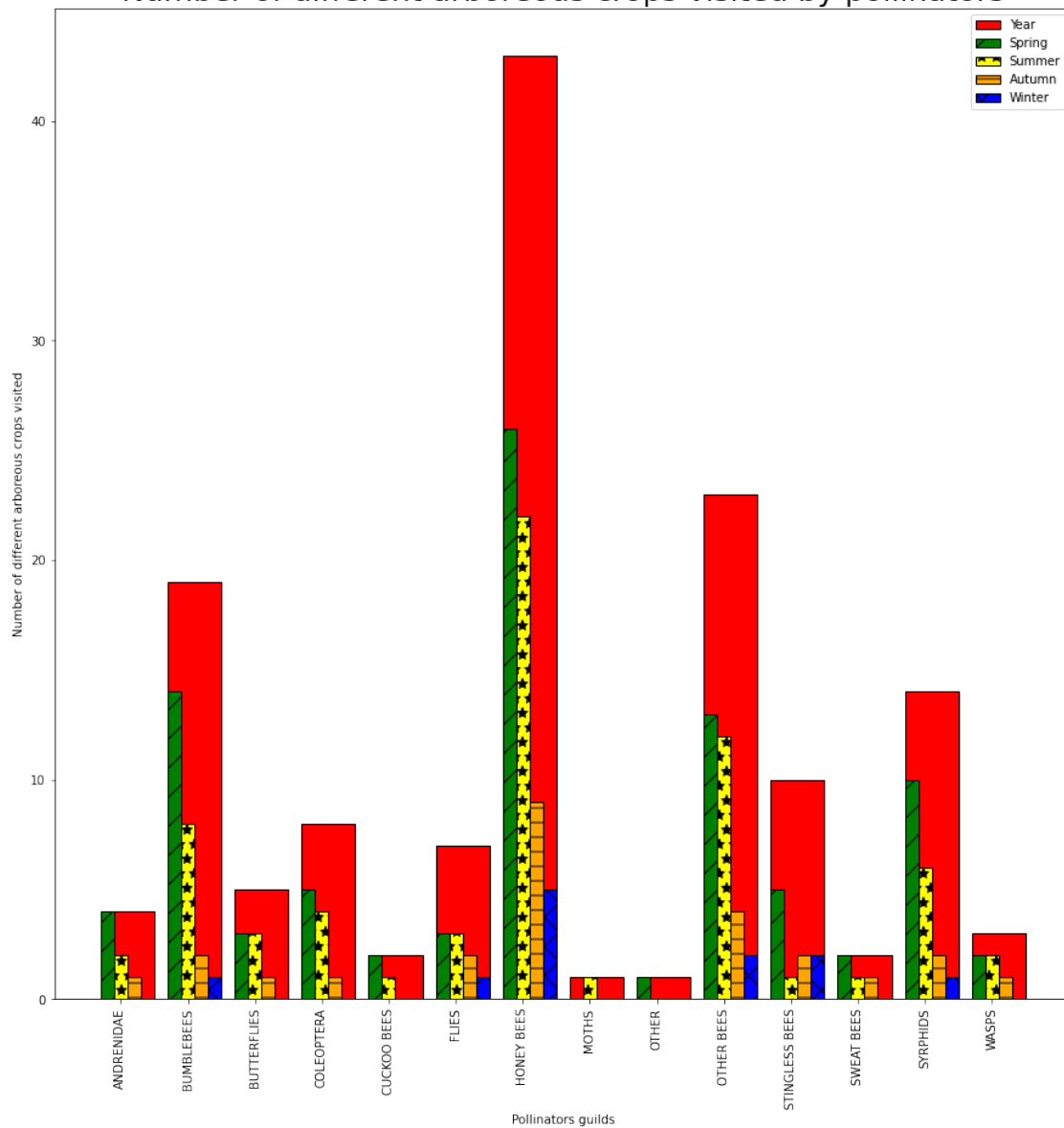
plt.pyplot.text(3, -7, 'Global pollinator database - Boreux & Klein - Figshare
    ↪Dataset', \
                fontsize=15)
plt.pyplot.text(5, -8.5, 'https://doi.org/10.6084/m9.figshare.9980471.v1',
    ↪fontsize=10)

plt.pyplot.savefig('Images/Arboreous plants visited by pollinators.png',
    ↪dpi=150)
plt.pyplot.savefig('Images/Arboreous plants visited by pollinators.jpg',
    ↪dpi=150)

plt.pyplot.show()

```

Number of different arboreous crops visited by pollinators



Global pollinator database - Boreux & Klein - Figshare Dataset

<https://doi.org/10.6084/m9.figshare.9980471.v1>

```
[756]: plt.pyplot.figure(figsize=(15,15))

bar_number = len(GPD_seasons_dataset_sp['guild'].unique().to_list())
x_range = np.arange(bar_number)
width = 0.20

plt.pyplot.bar(x_range + 1.5*width, year_guild_crop_dataset.count(), color = 'red',
```



```

        width = 4*width, edgecolor = 'black', label='Year')
plt.pyplot.bar(x_range , spring_guild_crop_dataset.count(), color = 'green',
               width = width, edgecolor = 'black', hatch='/', label='Spring')
plt.pyplot.bar(x_range + width, summer_guild_crop_dataset.count(), color =
    ↪'yellow',
               width = width, edgecolor = 'black', hatch= '*', label='Summer')
plt.pyplot.bar(x_range + 2*width, autumn_guild_crop_dataset_full.count(), color
    ↪= 'orange',
               width = width, edgecolor = 'black', hatch='-', label='Autumn')
plt.pyplot.bar(x_range + 3*width, winter_guild_crop_dataset_full.count(),
    ↪color = 'blue',
               width = width, edgecolor = 'black', hatch='x', label='Winter')

plt.pyplot.xlabel("Pollinators guilds")
plt.pyplot.ylabel("Number of different crops visited")
plt.pyplot.title("Number of different crops visited by pollinators",
    ↪fontsize=25)

plt.pyplot.xticks(x_range + width, GPD_seasons_dataset_sp['guild'].unique().
    ↪to_list(), \
                  rotation = 'vertical')
plt.pyplot.legend()

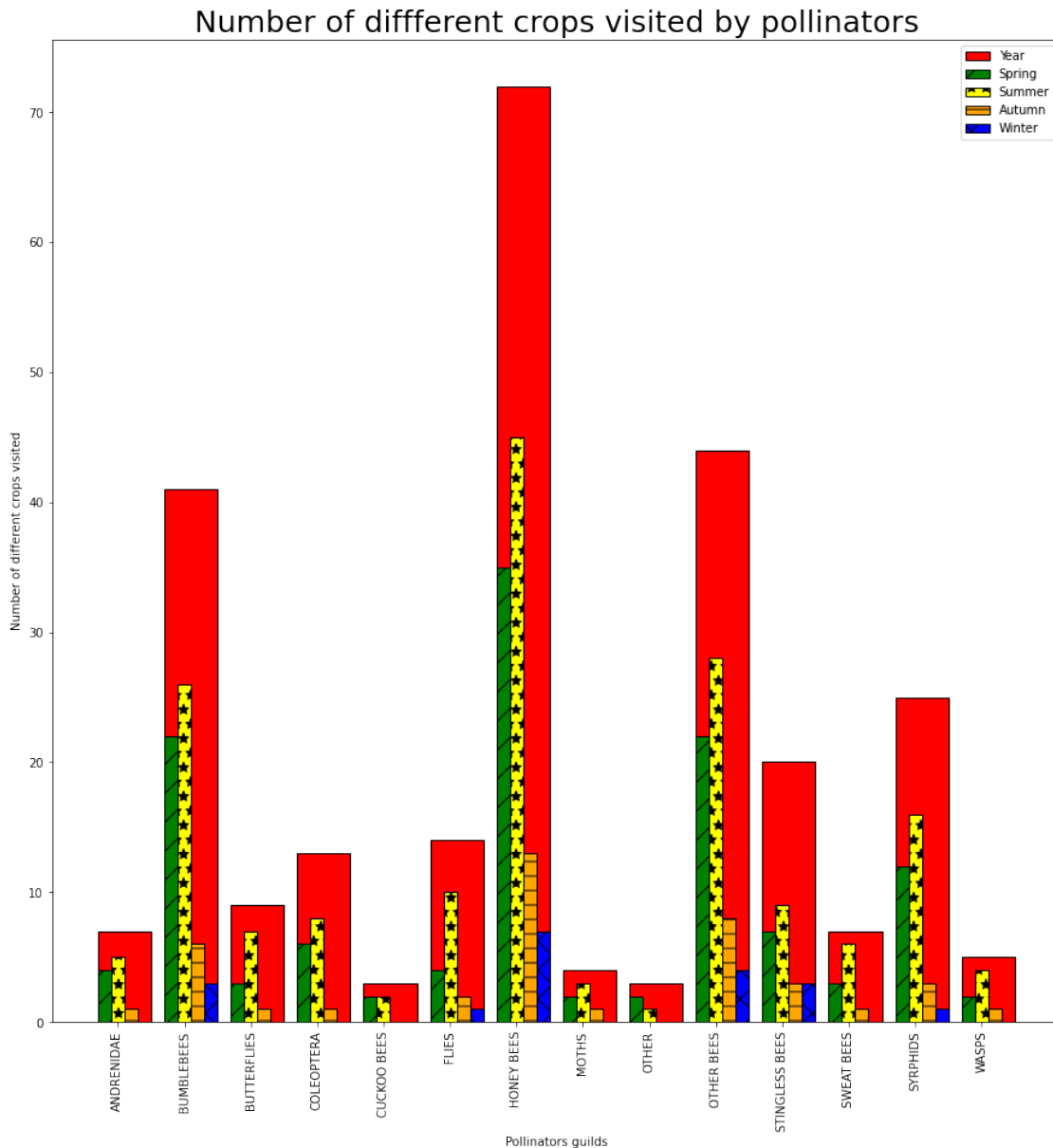
#fig.tight_layout()

plt.pyplot.text(3, -12, 'Global pollinator database - Boreux & Klein - Figshare
    ↪Dataset',\
               fontsize=15)
plt.pyplot.text(5, -13.5, 'https://doi.org/10.6084/m9.figshare.9980471.v1',
    ↪fontsize=10)

plt.pyplot.savefig('Images/Plants visited by pollinators.png', dpi=150)
plt.pyplot.savefig('Images/Plants visited by pollinators.jpg', dpi=150)

plt.pyplot.show()

```



Global pollinator database - Boreux & Klein - Figshare Dataset
<https://doi.org/10.6084/m9.figshare.9980471.v1>

Seems that, with some few exceptions, honeybees visit almost all visitable crop in the dataset. Could be interesting check if without honeybees the number of globally visited crops decrease, or in other words if there are crops visited only by honeybees.

```
[771]: #total number of different visited crops
len(GPD_seasons_dataset_sp.crop.unique().to_list())
```

[771]: 77

```
[789]: #total number of different visited crops except honeybee visit
len(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['guild'] != 'HONEY_
↳BEES',\

                                     'crop'].unique().to_list())
```

[789]: 73

```
[790]: #let's check the difference
set(GPD_seasons_dataset_sp.crop.unique().to_list()).difference(\
    set(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['guild'] !=_
↳'HONEY BEES',\

                                     'crop'].unique().to_list()))
```

```
[790]: {'Brassica_juncea',
        'Dimocarpus_longan',
        'Gossypium_arboreum',
        'Gossypium_hirsutum'}
```

```
[781]: #total number of spring different visited crops
len(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['spring'] == 1,\
                                     'crop'].unique().to_list())
```

[781]: 38

```
[797]: #total number of spring different visited crops except honeybee visit
len(GPD_seasons_dataset_sp.loc[ (GPD_seasons_dataset_sp['spring'] == 1) \
                                & (GPD_seasons_dataset_sp['guild'] != 'HONEY_
↳BEES'),\

                                     'crop'].unique().to_list())
```

[797]: 36

```
[798]: #let's check the difference
set(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['spring'] == 1,\
                                'crop'].unique().to_list()).difference(\
    set(GPD_seasons_dataset_sp.loc[ (GPD_seasons_dataset_sp['spring'] == 1)_
↳\

                                     & (GPD_seasons_dataset_sp['guild'] != 'HONEY_
↳BEES'),\

                                     'crop'].unique().to_list()))
```

```
[798]: {'Brassica_juncea', 'Dimocarpus_longan'}
```

```
[782]: #total number of summer different visited crops
len(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['summer'] == 1,\
                                     'crop'].unique().to_list())
```

[782]: 49

```
[802]: #total number of summer different visited crops except honeybee visit
len(GPD_seasons_dataset_sp.loc[ (GPD_seasons_dataset_sp['summer'] == 1) \
                                & (GPD_seasons_dataset_sp['guild'] != 'HONEY_
↪BEES')],\
                                'crop'].unique().to_list())
```

[802]: 46

```
[803]: #let's check the difference
set(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['summer'] == 1,\
                                'crop'].unique().to_list()).difference(\
    set(GPD_seasons_dataset_sp.loc[ (GPD_seasons_dataset_sp['summer'] == 1)\
↪\
                                & (GPD_seasons_dataset_sp['guild'] != 'HONEY_
↪BEES')],\
                                'crop'].unique().to_list()))
```

[803]: {'Dimocarpus_longan', 'Gossypium_arboreum', 'Gossypium_hirsutum'}

```
[783]: #total number of autumn different visited crops
len(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['autumn'] == 1,\
                                'crop'].unique().to_list())
```

[783]: 14

```
[804]: #total number of autumn different visited crops except honeybee visit
len(GPD_seasons_dataset_sp.loc[ (GPD_seasons_dataset_sp['autumn'] == 1)\
                                & (GPD_seasons_dataset_sp['guild'] != 'HONEY_
↪BEES')],\
                                'crop'].unique().to_list())
```

[804]: 13

```
[805]: #let's check the difference
set(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['autumn'] == 1,\
                                'crop'].unique().to_list()).difference(\
    set(GPD_seasons_dataset_sp.loc[ (GPD_seasons_dataset_sp['autumn'] == 1\
↪)\
                                & (GPD_seasons_dataset_sp['guild'] != 'HONEY_
↪BEES')],\
                                'crop'].unique().to_list()))
```

[805]: {'Gossypium_hirsutum'}

```
[784]: #total number of winter different visited crops
len(GPD_seasons_dataset_sp.loc[ GPD_seasons_dataset_sp['winter'] == 1,\
                                'crop'].unique().to_list())
```

[784]: 7

```
[806]: #total number of winter different visited crops except honeybee visit
len(GPD_seasons_dataset_sp.loc[ (GPD_seasons_dataset_sp['winter'] == 1) \
                                & (GPD_seasons_dataset_sp['guild'] != 'HONEY_
↪BEES'),\
                                'crop'].unique().to_list())
```

[806]: 7

So overall there are some plants visited only by honeybees but most of the crops are not visited only by honeybees.

In particular, we have the highest number of crops visited only by honeybee in summer (Dimocarpus_longan, Gossypium_arboreum, Gossypium_hirsutum), then in spring (Brassica_juncea, Dimocarpus_longan), in autumn is only Gossypium_hirsutum and in winter no visited plant is visited only by honeybees.

Let's focus a bit on these plants

```
[815]: GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['crop'] ==_
↪'Dimocarpus_longan' , ]
```

```
[815]:
```

	crop	type	season	corolla	colour	nectar	b.system	\
337	Dimocarpus_longan	arboreous	sprism	OPEN	white	yes	insects	
338	Dimocarpus_longan	arboreous	sprism	OPEN	white	yes	insects	
339	Dimocarpus_longan	arboreous	sprism	OPEN	white	yes	insects	

	s.pollination	inflorescence	composite	visitor	guild	\
337	no	yes	no	Apis_cerana	HONEY BEES	
338	no	yes	no	Apis_florea	HONEY BEES	
339	no	yes	no	Apis_mellifera	HONEY BEES	

	sociality	feeding	spring	summer	autumn	winter
337	yes	polylectic	1	1	0	0
338	yes	polylectic	1	1	0	0
339	yes	polylectic	1	1	0	0

```
[816]: GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['crop'] ==_
↪'Gossypium_arboreum' , ]
```

```
[816]:
```

	crop	type	season	corolla	colour	nectar	\
289	Gossypium_arboreum	arboreous	summer	CAMPANULATE	yellow	yes	
290	Gossypium_arboreum	arboreous	summer	CAMPANULATE	yellow	yes	

291	Gossypium_arboreum	arboreous	summer	CAMPANULATE	yellow	yes
-----	--------------------	-----------	--------	-------------	--------	-----

	b.system	s.pollination	inflorescence	composite	visitor	\
289	insects	no	solitary	no	Apis_cerana	
290	insects	no	solitary	no	Apis_dorsata	
291	insects	no	solitary	no	Apis_mellifera	

	guild	sociality	feeding	spring	summer	autumn	winter
289	HONEY BEES	yes	polylectic	0	1	0	0
290	HONEY BEES	yes	polylectic	0	1	0	0
291	HONEY BEES	yes	polylectic	0	1	0	0

```
[817]: GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['crop'] ==
↳ 'Gossypium_hirsutum' , ]
```

```
[817]:
```

	crop	type	season	corolla	colour	nectar	b.system	\
348	Gossypium_hirsutum	arboreous	sumaut	OPEN	yellow	yes	insects	
349	Gossypium_hirsutum	arboreous	sumaut	OPEN	yellow	yes	insects	
350	Gossypium_hirsutum	arboreous	sumaut	OPEN	yellow	yes	insects	
351	Gossypium_hirsutum	arboreous	sumaut	OPEN	yellow	yes	insects	

	s.pollination	inflorescence	composite	visitor	guild	\
348	no	yes	no	Apis_cerana	HONEY BEES	
349	no	yes	no	Apis_dorsata	HONEY BEES	
350	no	yes	no	Apis_florea	HONEY BEES	
351	no	yes	no	Apis_mellifera	HONEY BEES	

	sociality	feeding	spring	summer	autumn	winter
348	yes	polylectic	0	1	1	0
349	yes	polylectic	0	1	1	0
350	yes	polylectic	0	1	1	0
351	yes	polylectic	0	1	1	0

```
[818]: GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['crop'] == 'Brassica_juncea'
↳ , ]
```

```
[818]:
```

	crop	type	season	corolla	colour	nectar	b.system	\
302	Brassica_juncea	herbaceous	spring	OPEN	yellow	yes	insects	
303	Brassica_juncea	herbaceous	spring	OPEN	yellow	yes	insects	
304	Brassica_juncea	herbaceous	spring	OPEN	yellow	yes	insects	

	s.pollination	inflorescence	composite	visitor	guild	\
302	no	yes	no	Apis_dorsata	HONEY BEES	
303	no	yes	no	Apis_florea	HONEY BEES	
304	no	yes	no	Apis_mellifera	HONEY BEES	

	sociality	feeding	spring	summer	autumn	winter
--	-----------	---------	--------	--------	--------	--------

302	yes	polylectic	1	0	0	0
303	yes	polylectic	1	0	0	0
304	yes	polylectic	1	0	0	0

We could check wich crops are visited by the highest number of pollinators and wich crops has the logest flowering (or better visiting) period

```
[ ]: GPD_seasons_dataset_sp['seasons_sum'] = \
    GPD_seasons_dataset_sp[['spring', 'summer', \
    'autumn', 'winter']].sum(axis = 1)
```

```
[ ]: GPD_seasons_dataset_sp.sort_values(by = 'seasons_sum').tail(20)
```

```
[834]: #number of different crops with 4 season visit
len(GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['seasons_sum'] == 4,\
                                'crop'].unique().to_list())
```

[834] : 5

[illegible]

```
[835]: ['Cucurbita_pepo',
        'Cyphomandra_betacea',
        'Cocos_nucifera',
        'Coffea_canephora',
        'Sambucus_simpsonii']
```

```
[842]: for crop in GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['seasons_sum'] == 4,\
        'crop'].unique().to_list():
        visitors_number = len(GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['crop'] == crop,\
        'visitor'].unique().to_list())
        guilds_list = GPD_seasons_dataset_sp.loc[GPD_seasons_dataset_sp['crop'] == 'Cucurbita pepo',\
        'guild'].unique().to_list()
        print(f'Crop: {crop}')
        print(f'Number of visitors: {visitors_number}')
        print(f'Guilds: {guilds_list}')
```

```
Crop: Cucurbita_pepo
Number of visitors: 14
Guilds: ['BUMBLEBEES', 'HONEY BEES', 'OTHER BEES']
Crop: Cyphomandra betacea
```

```

Number of visitors: 3
Guilds: ['BUMBLEBEES', 'HONEY BEES', 'OTHER BEES']
Crop: Cocos_nucifera
Number of visitors: 4
Guilds: ['BUMBLEBEES', 'HONEY BEES', 'OTHER BEES']
Crop: Coffea_canephora
Number of visitors: 6
Guilds: ['BUMBLEBEES', 'HONEY BEES', 'OTHER BEES']
Crop: Sambucus_simpsonii
Number of visitors: 6
Guilds: ['BUMBLEBEES', 'HONEY BEES', 'OTHER BEES']

```

```

[844]: year_guild_crop_dataset['guilds_number'] = year_guild_crop_dataset.count(axis=
      ↪ 1)
year_guild_crop_dataset['visitors_number'] = year_guild_crop_dataset.sum(axis =
      ↪ 1)

```

```

[845]: year_guild_crop_dataset.sort_values(by = 'guilds_number').tail()

```

```

[845]: guild          ANDRENIDAE  BUMBLEBEES  BUTTERFLIES  COLEOPTERA  \
crop
Prunus_avium          25.0           7.0           NaN           NaN
Daucus_carota          2.0           NaN           NaN           1.0
Coriandrum_sativum     1.0           NaN           1.0           1.0
Malus_domestica         5.0           5.0           2.0           3.0
Brassica_napus         10.0           7.0           2.0           2.0

guild          CUCKOO BEES  FLIES  HONEY BEES  MOTHS  OTHER  OTHER BEES  \
crop
Prunus_avium          14.0     NaN           1.0     NaN     NaN          11.0
Daucus_carota          NaN     1.0           1.0     NaN     NaN           2.0
Coriandrum_sativum     NaN     NaN           4.0     NaN     NaN           3.0
Malus_domestica         NaN     2.0           2.0     NaN     4.0          15.0
Brassica_napus          1.0     1.0           3.0     NaN     2.0           6.0

guild          STINGLESS BEES  SWEAT BEES  SYRPHIDS  WASPS  \
crop
Prunus_avium          NaN          23.0         NaN     NaN
Daucus_carota          NaN           2.0         1.0     NaN
Coriandrum_sativum     NaN           1.0         2.0     NaN
Malus_domestica         NaN          NaN          9.0     1.0
Brassica_napus          1.0           4.0         5.0     NaN

guild          guilds_number  visitors_number
crop
Prunus_avium           6          87.0
Daucus_carota           7          17.0

```


Coriandrum_sativum	7	20.0
Malus_domestica	10	58.0
Brassica_napus	12	56.0

```
[846]: year_guild_crop_dataset.sort_values(by = 'visitors_number').tail()
```

```
[846]: guild      ANDRENIDAE  BUMBLEBEES  BUTTERFLIES  COLEOPTERA  \
crop
Vaccinium_corymbosum      2.0         5.0         NaN         NaN
Helianthus_annuus         NaN        16.0         1.0         NaN
Brassica_napus            10.0         7.0         2.0         2.0
Malus_domestica           5.0         5.0         2.0         3.0
Prunus_avium              25.0         7.0         NaN         NaN

guild      CUCKOO BEES  FLIES  HONEY BEES  MOTHS  OTHER  \
crop
Vaccinium_corymbosum      1.0    NaN         1.0    NaN    NaN
Helianthus_annuus         NaN    NaN         4.0    NaN    NaN
Brassica_napus            1.0    1.0         3.0    NaN    2.0
Malus_domestica           NaN    2.0         2.0    NaN    4.0
Prunus_avium              14.0    NaN         1.0    NaN    NaN

guild      OTHER BEES  STINGLESS BEES  SWEAT BEES  SYRPHIDS  WASPS  \
crop
Vaccinium_corymbosum      22.0         NaN         NaN         NaN    NaN
Helianthus_annuus         9.0         NaN         NaN         1.0    NaN
Brassica_napus            6.0         1.0         4.0         5.0    NaN
Malus_domestica           15.0         NaN         NaN         9.0    1.0
Prunus_avium              11.0         NaN        23.0         NaN    NaN

guild      guilds_number  visitors_number
crop
Vaccinium_corymbosum      5         36.0
Helianthus_annuus         5         36.0
Brassica_napus            12        56.0
Malus_domestica           10        58.0
Prunus_avium              6         87.0
```

```
[862]: for crop in year_guild_crop_dataset.sort_values(by = 'guilds_number').tail(10).
        ↪index.to_list():
        visitors_number = year_guild_crop_dataset.loc[crop, \
                                                         'visitors_number']

        guilds_list = year_guild_crop_dataset.columns[\
            year_guild_crop_dataset.loc[crop].isnull()].to_list()

        print(f'Crop: {crop}')
        print(f'Number of visitors: {visitors_number}')
```

```
print(f'Guilds: {guilds_list}')
```

```
Crop: Allium_oleraceum
Number of visitors: 14.0
Guilds: ['ANDRENIDAE', 'COLEOPTERA', 'CUCKOO BEES', 'FLIES', 'HONEY BEES',
'OTHER', 'OTHER BEES', 'STINGLESS BEES', 'SWEAT BEES']
Crop: Helianthus_annuus
Number of visitors: 36.0
Guilds: ['ANDRENIDAE', 'COLEOPTERA', 'CUCKOO BEES', 'FLIES', 'MOTHS', 'OTHER',
'STINGLESS BEES', 'SWEAT BEES', 'WASPS']
Crop: Vaccinium_corymbosum
Number of visitors: 36.0
Guilds: ['BUTTERFLIES', 'COLEOPTERA', 'FLIES', 'MOTHS', 'OTHER', 'STINGLESS
BEES', 'SWEAT BEES', 'SYRPHIDS', 'WASPS']
Crop: Arachis_hypogaea
Number of visitors: 16.0
Guilds: ['ANDRENIDAE', 'BUTTERFLIES', 'COLEOPTERA', 'CUCKOO BEES', 'FLIES',
'MOTHS', 'OTHER', 'SYRPHIDS', 'WASPS']
Crop: Allium_cepa
Number of visitors: 23.0
Guilds: ['ANDRENIDAE', 'BUMBLEBEES', 'CUCKOO BEES', 'MOTHS', 'OTHER', 'STINGLESS
BEES', 'SWEAT BEES', 'WASPS']
Crop: Prunus_avium
Number of visitors: 87.0
Guilds: ['BUTTERFLIES', 'COLEOPTERA', 'FLIES', 'MOTHS', 'OTHER', 'STINGLESS
BEES', 'SYRPHIDS', 'WASPS']
Crop: Daucus_carota
Number of visitors: 17.0
Guilds: ['BUMBLEBEES', 'BUTTERFLIES', 'CUCKOO BEES', 'MOTHS', 'OTHER',
'STINGLESS BEES', 'WASPS']
Crop: Coriandrum_sativum
Number of visitors: 20.0
Guilds: ['BUMBLEBEES', 'CUCKOO BEES', 'FLIES', 'MOTHS', 'OTHER', 'STINGLESS
BEES', 'WASPS']
Crop: Malus_domestica
Number of visitors: 58.0
Guilds: ['CUCKOO BEES', 'MOTHS', 'STINGLESS BEES', 'SWEAT BEES']
Crop: Brassica_napus
Number of visitors: 56.0
Guilds: ['MOTHS', 'WASPS']
```