DOPPThreatenedSpecies

February 11, 2021

[1]:
```python
from pathlib import Path
import yaml
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import (
    GridSearchCV,
    LeaveOneGroupOut,
    LeaveOneOut,
    LeavePGroupsOut,
    cross_val_score,
    train_test_split,
)
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import (
    LabelEncoder,
    MinMaxScaler,
    OneHotEncoder,
    StandardScaler,
)
from sklearn.svm import SVR

import warnings
warnings.filterwarnings('ignore')
```

1 Task

1.1 Aim

We were tasked to evaluate the endangerment of species as well as their developement and the characteristics of the countries they inhabit.

1.2 Questions

- How many species are endangered in total?
- How many species are endangered by group/country?
- What characteristics influence the overall trend of endangerment in a country?
- Can the trend of endangerement be predicted inside a country?
- Can the number of endangered species be predicted inside a country?

2 Approach

To reach our goal we gathered and processed Data from 3 Sources. Firstly we used the IUCN Redlist to identify species per country as well as their endangerment status. We were unable to gather historic data, as only the recent data was freely available. Secondly as support datasets we gathered country characteristics from OECD from their open database. After considering the available dataset we decided for their greenhouse gases, land cover and land usage datasets were the most useful for this task. Thirdly we decided to supplement our data with climate information, because we suspect climate change to have a large role in the endangerment of species. For this purpose we used the data available from worldbank.

The data from the IUCN Redlist had to be web scraped, as no usable format was openly available, here we decided on only handling land based animals to increase the connection to the countries. For the other data sources a csv download was available. Afterwards the downloaded data was normalized, grouping the species into their respective classes (e.g. mammal) and standardizing the endangerment threat levels. Missing data was handled and a relative value according to the groups were defined to help in further steps.

The support datasets were handled in the schema

1. Loading Data
2. Cleaning Data
3. Feature Preperation (including the interpolation of current data were neccesary)
4. Data Exploration

In the next steps the datasets were combined and an analysis on the combined dataset was done.

Finally we generated models that were capable to predict the trend of species endangerement by their respective groups inside countries, as well as predict the number of endangered species by country relative to the total number of species in that country.

3 Data

We use three data sources for our analysis:

Our main data source is the International Union For Conservation of Nature (IUCN). This is an international organization working on the field of nature conservation. They provide the most relevant and detaild data on threatened and extinct species.

We selected the OECD repository as our second source of data because it provides different good quality data sets on enviornment and biodiversity. Data is provided on a per country level. As not all countries are members of the OECD or have a close relationship with it this limits the number of countries we can use for our analysis.

As the third data source we selected climate dataset provided by the World Bank Group. The datasets constists of temperature and rainfall data for the years 1990 to 2016 on a country level.

[2]:
```
# set data path
DATA_PATH = Path('./data/')
```

3.1 Selected countries

We select all countries we have information on across all used data sets. Countries selected are listed in a seperate yaml file "countries.yml" and structured by region which is needed for scraping the data from the IUCN webpage. This further provides a single method to filter our data and make sure all data sets have information on the same countries. We initally chose 65 countries that are present in all datasets available from the OECD repository and on the IUCN webpage.

[3]:
```
COUNTRIES_YAML = Path('./countries.yml')

# get names of selected countries from YAML file
def get_country_list():
    countries = []
    with open(COUNTRIES_YAML, 'r') as cfg_file:
        cfg = yaml.safe_load(cfg_file)
    for region in cfg['countries']:
        countries += cfg['countries'][region]
    return countries

# get names of selected countries from YAML file
# names of countries slightly differ for the IUCN webpage
def get_countries_for_IUCN():
    region_country_list = []
    with open(COUNTRIES_YAML, 'r') as cfg_file:
        cfg = yaml.safe_load(cfg_file)
    for region_name in cfg['countries']:
        for country_name in cfg['countries'][region_name]:
            country_dict = {}
            country_dict['region_name'] = region_name
            country_dict['country_name'] = country_name
            if country_name in list(cfg['IUCN_name_transform'].keys()):
                country_dict['country_iucn'] = ␣
↪cfg['IUCN_name_transform'][country_name]
            else:
                country_dict['country_iucn'] = country_name
            region_country_list.append(country_dict)

    return region_country_list

SELECTED_COUNTRIES = get_country_list()
COUNTY_IUCN_DICT = get_countries_for_IUCN()
print(f'Inital number selected countries: {len(SELECTED_COUNTRIES)}')
```

```
Inital number selected countries: 65
```

3.2 IUCN Redlist Data

Difficulties We encountered several difficulties for utilizing the data provided by the IUCN.

First, summery statistics. The provided data is mostly in a format that is not machine readable (PDFs) or when machine readable files (CSVs) are provided the data is not in sufficient detail per country.

Second, spatial data. This type of data provides detailed information per group of species. The data is provided as polygons but as our goal is to compare different characteristics of countries we would have to map the polygons to countries which is not a trivial task as the IUCNs process to define which species is resident in which country is very sophisticated and not easy to reproduce.

Third, there is no "historical" data on threatened species. Only some PDFs document the changes in status per species and year but the IUCN specifically states that "This table (Table 7) should not be used to calculate a Red List Index (RLI); for this it is necessary to analyse the underlying Red List data to identify genuine status changes between specific years for specific taxonomic groups."

Approach The approach we therefore took was to scrape the needed data from the IUCN web page using the advanced search at: https://www.iucnredlist.org/search/list. Web scraping was performed prior to all other tasks to ensure we have the data in sufficient detail. The process of data collection can be found in the "IUCN_web_scraping.py" module and is not included in this notebook. This is because scraping the data from the web is a time intensive process and also error prone as several runs had to be performd to ensure all data is loaded. We use **selenium** and **beautifulsoup4** as they let us navigate the IUCN Web page and extract the species on a per country level.

We filter for only animals as tracking other species like plants or fungi is more problematic. There are still many of these species that have not yet been assessed for the IUCN Red List and therefore their status is not known (i.e., these groups have not yet been completely assessed). Further, we filter on the "Country Legend" as descibed at https://www.iucnredlist.org/resources/summary-statistics under Tables 5 & 6: Summaries by country. This is done to ensure that the data is consistent with the IUCN Tables 5 and 6 which are organized by country. Tags filtered for are:'Extant', 'Extant & Reintroduced', 'Extinct', 'Extinct & Reintroduced', 'Possibly Extinct', and 'Possibly Extinct & Reintroduced'.

3.2.1 Load IUCN Data

First, we load the scraped data. Data was stored as one CSV by country. For each DataFrame we add the country as a separate column and afterwards concatenate all DataFrames.

[4]:
```
def load_IUCN_data():
    all_countries = []
    DATA_PATH = Path('./data/IUCN/scraped/')
    file_paths = DATA_PATH.glob('*.csv')

    for file_path in file_paths:
        df = pd.read_csv(file_path)
```

```
        df['Country'] = file_path.stem
        all_countries.append(df)
    return pd.concat(all_countries, ignore_index=True)
IUCN_raw_data = load_IUCN_data()
IUCN_raw_data.shape
```

[4]: (136624, 7)

3.2.2 Clean IUCN Data

In this step we inspect the raw data and handle major difficulties in the scraped data. The data is then preprocessed to transform it to the desired form.

The major difficulties we ecnountered are the following: - The kingdom in the kingdom_class column is the same for all values as we filtered for animals only during scraping. - The common name for species is missing alot. - The trend is missing for a lot of species. - The region is not usable because most of the time it includes "Global" and we are interested on a per country level. - The threat_level includes data for 41 species that was missing on the IUCN webpage. Only some JS message is stored.

[5]:
```
# kingdom the same for all values
IUCN_raw_data['kingdom_class'].unique()
```

[5]:
```
array(['animalia - actinopterygii', 'animalia - reptilia',
       'animalia - mammalia', 'animalia - amphibia',
       'animalia - chondrichthyes', 'animalia - insecta',
       'animalia - cephalopoda', 'animalia - gastropoda',
       'animalia - holothuroidea', 'animalia - cephalaspidomorphi',
       'animalia - aves', 'animalia - anthozoa',
       'animalia - malacostraca', 'animalia - merostomata',
       'animalia - clitellata', 'animalia - bivalvia',
       'animalia - hydrozoa', 'animalia - arachnida',
       'animalia - maxillopoda', 'animalia - myxini',
       'animalia - sarcopterygii', 'animalia - polychaeta',
       'animalia - echinoidea', 'animalia - branchiopoda',
       'animalia - asteroidea', 'animalia - ostracoda',
       'animalia - onychophora', 'animalia - enopla',
       'animalia - turbellaria', 'animalia - monoplacophora',
       'animalia - diplopoda', 'animalia - entognatha'], dtype=object)
```

[6]:
```
# check for missing numbers
IUCN_raw_data.isna().sum()
```

[6]:
```
kingdom_class          0
common_name        43403
scientific_name        0
trend               2986
region                 0
```

```
threat_level          0
Country               0
dtype: int64
```

[7]:
```
# region values not usable
IUCN_raw_data['region'].unique()
```

[7]:
```
array(['Global', 'Global, Arabian Sea', 'Global, Europe',
       'Global, Mediterranean', 'Global, Europe, Mediterranean',
       'Global, Caribbean', 'Global, Northern Africa, Pan-Africa',
       'Global, Caribbean, Gulf of Mexico', 'Global, Gulf of Mexico',
       'Global, Pan-Africa', 'Global, Pan-Africa, S. Africa FW',
       'Global, Eastern Africa, Pan-Africa', 'Global, Persian Gulf'],
      dtype=object)
```

[8]:
```
# missing data on webpage "[missing "en.shared.categories.cd" translation]"
IUCN_raw_data[IUCN_raw_data.threat_level == '[missing "en.shared.categories.cd"␣
 ↪translation]'].shape
```

[8]:
```
(41, 7)
```

Preparing the data Several steps are taken to clean the raw IUCN data: - The common name for each species is dropped as we can use the scientific name which is never missing. - The observations where the threat_level is "missing" is renamed to the existing group "Data Deficient". - We checked the species directly on the web page and saw that they were not categorized for any threat level. - The missing trend values are filled with the existing group "Unknown". - The class is extracted from each kingdom_class column. - We chose "group" for the new feature name as python would encounter problems with the name "class". - The groups include species which are of no interest for our analysis. So all sea species are excluded. - Mammals, Insects, Amphibians, Birds and Reptiles are kept - we renamed these as the scientific name is harder to recognize - The threat_level is renamed to its abbreviation.

We have to note that for reptiles there are still many species that have not yet been assessed.

Translation of scientific class names

- mammalia: mammals
- actinopterygii: ray-finned fishes
- insecta: insects
- amphibia: amphibians
- aves: birds
- bivalvia: clams, oysters, cockles, mussels, scallops
- gastropoda: snails and slugs
- cephalaspidomorphi: jaw-less fishes
- clitellata: worms
- reptilia: reptiles
- chondrichthyes: cartilaginous fishes
- malacostraca: crustaceans

- hydrozoa: individually very small, predatory animals, most living in salt water
- turbellaria: flatworms

[9]:
```
def IUCN_clean_data(data, filter_terrestrial=True):
    # remove column common name and region
    data = data.drop(columns=['common_name', 'region'])

    # categorize missing scraped data for trend to existing Data Deficient␣
↪category
    data.threat_level.replace({
        '[missing "en.shared.categories.cd" translation]': 'Data Deficient'},
        inplace=True)

    # fill nan vlaues in trend with existing Unknown category
    data.trend.fillna('Unknown', inplace=True)

    # extract only class as kingdom is always animalia
    data['kingdom_class'] = data.apply(lambda row: row['kingdom_class'].
↪split()[-1], axis=1)
    data = data.rename(columns={'kingdom_class': 'group'})

    # only select none sea animals
    if filter_terrestrial:
        none_sea_animals = ['mammalia', 'insecta', 'amphibia', 'aves',␣
↪'reptilia']
        data = data[data.group.isin(none_sea_animals)]

    # rename classes
    data.group.replace({
        'mammalia': 'mammals',
        'insecta': 'insects',
        'amphibia': 'amphibians',
        'aves': 'birds',
        'reptilia': 'reptiles',
        },
        inplace=True)

    # rename threat levels
    data.threat_level.replace({
        'Extinct': 'EX',
        'Extinct in the Wild': 'EW',
        'Critically Endangered': 'CR',
        'Endangered': 'EN',
        'Vulnerable': 'VU',
        'Near Threatened': 'NT',
        'Least Concern': 'LC',
        'Data Deficient': 'DD',
```

```
        },
        inplace=True)

    return data
IUCN_cleaned_data = IUCN_clean_data(IUCN_raw_data)
IUCN_cleaned_data.shape
```

[9]: (66669, 5)

[10]:
```
# have a look at the cleand data
IUCN_cleaned_data.head()
```

[10]:
```
         group          scientific_name       trend threat_level Country
9     reptiles  Goniurosaurus splendens  Decreasing           EN   Japan
10     mammals            Phoca vitulina     Unknown           LC   Japan
15    reptiles    Hemidactylus frenatus      Stable           LC   Japan
18  amphibians          Odorrana narina  Decreasing           EN   Japan
19  amphibians       Hynobius nebulosus  Decreasing           LC   Japan
```

3.2.3 Check if scraped data is complete

As the web scraping process is error prone we need to check if the number of species by country we extracted from the IUCN web page make sense. For this we use Table 6a of the IUCN Summary Statistics: https://www.iucnredlist.org/resources/summary-statistics

First we load our scraped data for all species and bring it in the same format as Table 6a. Then we compare the difference in number of species by our selected countries.

For 19 countries we have different numbers of total species but the differences are not large (between 1 and 7). We can attribute these differences due to the fact that the Table 6a of the summary statistics is not up to date. Further, the process how species are attributed to a country could be different in the summery statistic compared to the data on the web. As there is no major difference in species for any country, we can assume that the web scraping process did not encounter any major problems or missed collecting some data.

Here is a short description of the threat levels contained in Table 6a. IUCN Red List Categories: EX - Extinct, EW - Extinct in the Wild, CR - Critically Endangered (includes CR(PE) and CR(PEW)), EN - Endangered, VU - Vulnerable, LR/cd - Lower Risk/conservation dependent, NT - Near Threatened (includes LR/nt - Lower Risk/near threatened), DD - Data Deficient, LC - Least Concern (includes LR/lc - Lower Risk/least concern).

[11]:
```
# load all species by country
IUCN_cleaned_all = IUCN_clean_data(IUCN_raw_data, filter_terrestrial=False)
# bring data In same format as Table 6a
grouped = IUCN_cleaned_all.groupby(['Country',␣
↪'threat_level'])['scientific_name'].count().reset_index(name='count')
species_tl = grouped.pivot_table(index='Country', columns='threat_level',␣
↪values='count')
species_tl = species_tl.fillna(0.0)
```

```
species_tl['Total'] = species_tl.sum(axis=1)
species_tl = species_tl.convert_dtypes(convert_integer=True)
# show number of species by threat level
species_tl.head()
```

```
[11]:             CR   DD   EN  EW  EX    LC   NT   VU  Total
      Country
      Argentina    40  178   69   3   3  2210  127  118   2748
      Armenia       7   18    8   0   0   471   38   26    568
      Australia   137  664  255   0  42  5867  442  613   8020
      Austria      24   66   29   0   3   851   81   52   1106
      Azerbaijan   13   45   10   1   0   583   41   31    724
```

```
[12]: def IUCN_load_table6a(threat_levels, country_rename_mapper, country_list):
          # load data
          DATA_PATH = Path('./data/IUCN')
          data = pd.read_csv(DATA_PATH / 'Table 6a Animal species (kingdom Animalia)␣
       ↪by country - show all.csv', thousands=',')
          # rename columns
          data = data.rename(columns={
              'Name': 'Country',
              'NT or LR/nt': 'NT',
              'LC or LR/lc': 'LC',
          })
          # add LR/cd (Lower Risk/conservation dependent) to Least Concern
          data['LC'] = data['LC'] + data['LR/cd']
          # only select needed threat_levels
          data = data[['Country'] + threat_levels]
          # rename countries
          data['Country'].replace(country_rename_mapper, inplace=True)
          # only select needed countries
          data = data[data['Country'].isin(country_list)]
          data = data.sort_values('Country')
          data = data.set_index('Country')

          return data

      # get only the threat levels we are interested in (others are sub or super␣
       ↪groups)
      threat_levels = list(species_tl.columns)
      # countries need to be renamed
      country_rename_mapper = {d['country_iucn']: d['country_name'] for d in␣
       ↪COUNTY_IUCN_DICT}
      table6a = IUCN_load_table6a(threat_levels, country_rename_mapper,␣
       ↪SELECTED_COUNTRIES)
      # check if all countries the same
      assert len(SELECTED_COUNTRIES) == table6a.shape[0]
```

[13]:
```python
# compare difference in total species per country
species_tl_total = species_tl[['Total']].rename(columns={'Total':␣
↪'Total_scraped'})
species_tl_total = species_tl_total.reset_index()
evaluate_difference = table6a.merge(species_tl_total, how='left', on='Country').
↪set_index('Country')
evaluate_difference['diff'] = evaluate_difference['Total'] -␣
↪evaluate_difference['Total_scraped']
evaluate_difference[evaluate_difference['diff'] != 0]
```

[13]:
```
                          CR    DD   EN  EW   EX    LC   NT   VU  Total  \
Country
Argentina                 41   178   69   3    3  2211  127  118   2750
Australia                138   661  255   0   42  5871  442  613   8022
Brazil                   105   700  144   2   11  4777  230  287   6256
Canada                    18    87   32   0    9  1902   64   84   2196
Chile                     25   201   62   0    1  1302   84   81   1756
Colombia                 126   586  227   0    1  5285  259  361   6845
Costa Rica                35   235   82   0    4  3392  107  154   4009
Greenland                  2    20    5   0    1   217    8   20    273
India                     94   868  230   0    0  4334  331  398   6255
Indonesia                185  1392  323   0    3  6311  640  654   9508
Japan                     46   508  149   1   14  3375  267  256   4616
Mexico                   202   585  343   9   21  4931  217  362   6670
New Caledonia             41   182   56   0    5  2190  163  159   2796
New Zealand               45   210   77   0   23  1066   66  109   1596
Northern Mariana Islands   9    70   24   0    2  1296   91   72   1564
Peru                      58   474  155   0    1  3893  204  209   4994
Russia                    28   233   49   1    3  1735  118  111   2278
South Africa              85   340  178   0    6  3722  189  213   4733
United States            224   609  298   4  237  5724  336  566   7998

                          Total_scraped  diff
Country
Argentina                          2748     2
Australia                          8020     2
Brazil                             6249     7
Canada                             2195     1
Chile                              1754     2
Colombia                           6840     5
Costa Rica                         4008     1
Greenland                           272     1
India                              6253     2
Indonesia                          9502     6
Japan                              4615     1
Mexico                             6669     1
New Caledonia                      2795     1
```

```
New Zealand                        1594     2
Northern Mariana Islands           1562     2
Peru                               4991     3
Russia                             2277     1
South Africa                       4732     1
United States                      7994     4
```

3.3 Feature preparation

In the next section we compute relative numbers for interesting target variables. We decided to use relative trends and threat levels four our selected groups of animals on a per country level.

3.3.1 Create relative threatened species per group

Threatened species are listed in any of the three categories Critically Endangered (CR), Endangered (EN) or Vulnerable (VU).

We compute and extract the relative threatened species per group and in total. Because some countries have no species in each group we also create features to define if a group of species is resident in a given country. This is done because otherwise zero threatened species and zero species could not be distinguished.

[14]:
```
def IUCN_threatened_by_group(species_by_country):
    relative_threatened = []
    # iterate all groups of animals and the total relative value
    groups = ['total'] + list(species_by_country.group.unique())
    for group in groups:
        filtered = species_by_country
        if group != 'total':
            filtered = species_by_country[species_by_country.group == group]
        # count the number of species for each threat level
        grouped = filtered.groupby(['Country',␣
↪'threat_level'])['scientific_name']
        grouped = grouped.count().reset_index(name='count')
        # transform data so we have the value counts per threat level in the␣
↪DataFrame
        current_group = grouped.pivot_table(index='Country',␣
↪columns='threat_level', values='count')
        # fill nan values because if there are no species by one threat level␣
↪we have NaNs
        current_group = current_group.fillna(0.0)
        # calculate relative numbers
        relative = current_group[['CR', 'EN', 'VU']].sum(axis=1) /␣
↪current_group.sum(axis=1)
        # rename the column
        relative = relative.to_frame(f'{group}_threatened').round(4)
        relative_threatened.append(relative)
```

```
    combined_data = pd.concat(relative_threatened, axis=1)
    # as some countries don't have species in each group we create features
    # to define if a group of species is resident in a given country
    species_resident = combined_data.notna()
    column_names = [f'{group}_resident' for group in groups]
    species_resident.columns = column_names
    species_resident = species_resident.drop(columns='total_resident')

    # fill NaNs for relative threatened if there is no species in this country␣
↪and group
    combined_data = combined_data.fillna(0.0)

    return combined_data.join(species_resident)

threatened_by_group = IUCN_threatened_by_group(IUCN_cleaned_data)

ds_threatened_by_group = threatened_by_group.reset_index(drop=False).
↪rename(columns={'index':'Country'}).copy()
```

3.3.2 Create relative numbers per trend

There are three trends we are interested in “Decreasing”, “Increasing” and “Stable”. The trend includes also the value “Unknown” which states that the trend is not assessed by the IUCN. All NaN values for trend where imputed with this “Unknown” category.

We compute and extract the relative trend for species per group.

[15]:
```
def IUCN_trend_by_group(species_by_country):

    grouped = IUCN_cleaned_data.groupby(['Country',␣
↪'trend'])['scientific_name'].count().reset_index(name='count')
    species_trend_country = grouped.pivot_table(index='Country',␣
↪columns='trend', values='count')
    species_trend_country = species_trend_country.fillna(0.0)
    species_trend_country['Total'] = species_trend_country.sum(axis=1)

    trends = list(IUCN_cleaned_data.trend.unique())


    relative_trends = []
    # iterate all groups of animals and the total relative value
    trends = list(species_by_country.trend.unique())
    for trend in trends:
        species_trend_country[trend] = species_trend_country[trend] /␣
↪species_trend_country['Total']
    species_trend_country = species_trend_country.drop(columns='Total')
```

```
    # drop unknown trends
    species_trend_country = species_trend_country.drop(columns='Unknown')

    # rename column headings
    species_trend_country = species_trend_country.rename(columns=lambda x: f'{x.
↪lower()}_trend')

    return species_trend_country

ds_trend_by_group = IUCN_trend_by_group(IUCN_cleaned_data)
```

3.4 Data Exploration

Next we explore the data in general. We focus on the whole data set as the created features are explored in the modeling section of the notebook.

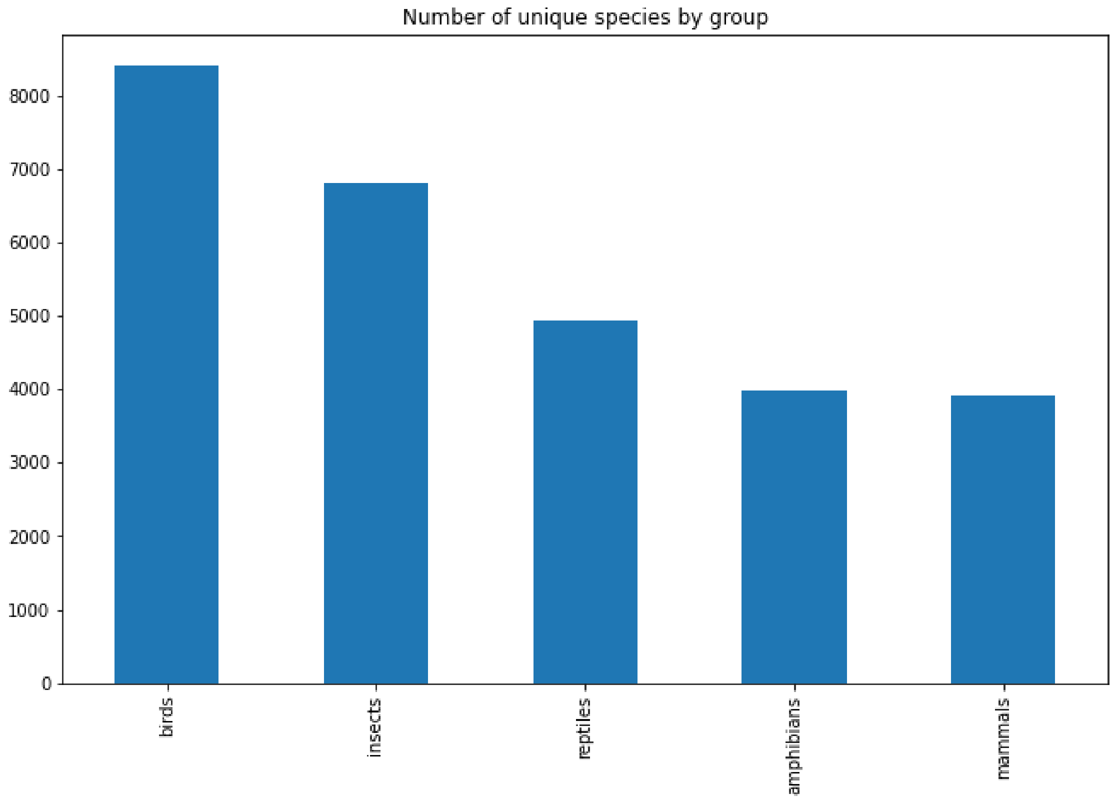
[16]:
```
# number of unique animal species in all countrys
species = IUCN_cleaned_data.drop(columns=['Country']).drop_duplicates()
species.shape
```

[16]:
```
(28080, 4)
```

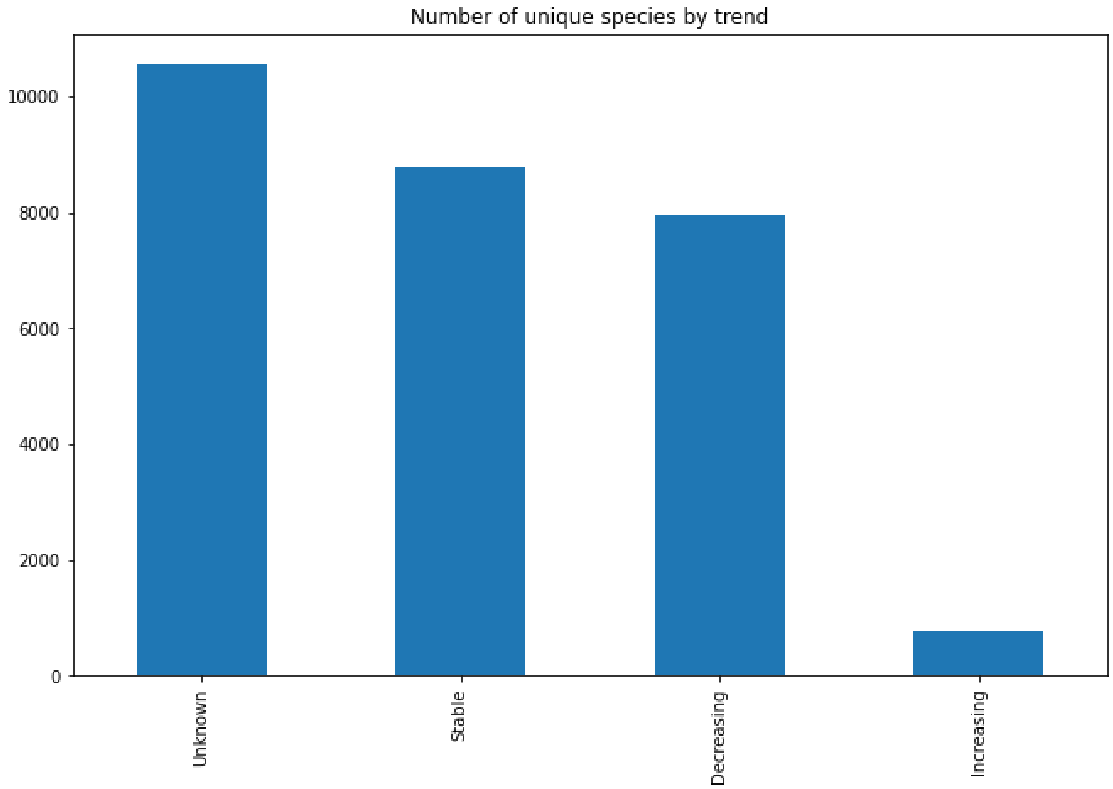
[17]:
```
species.group.value_counts()
```

[17]:
```
birds         8410
insects       6826
reptiles      4934
amphibians    3986
mammals       3924
Name: group, dtype: int64
```

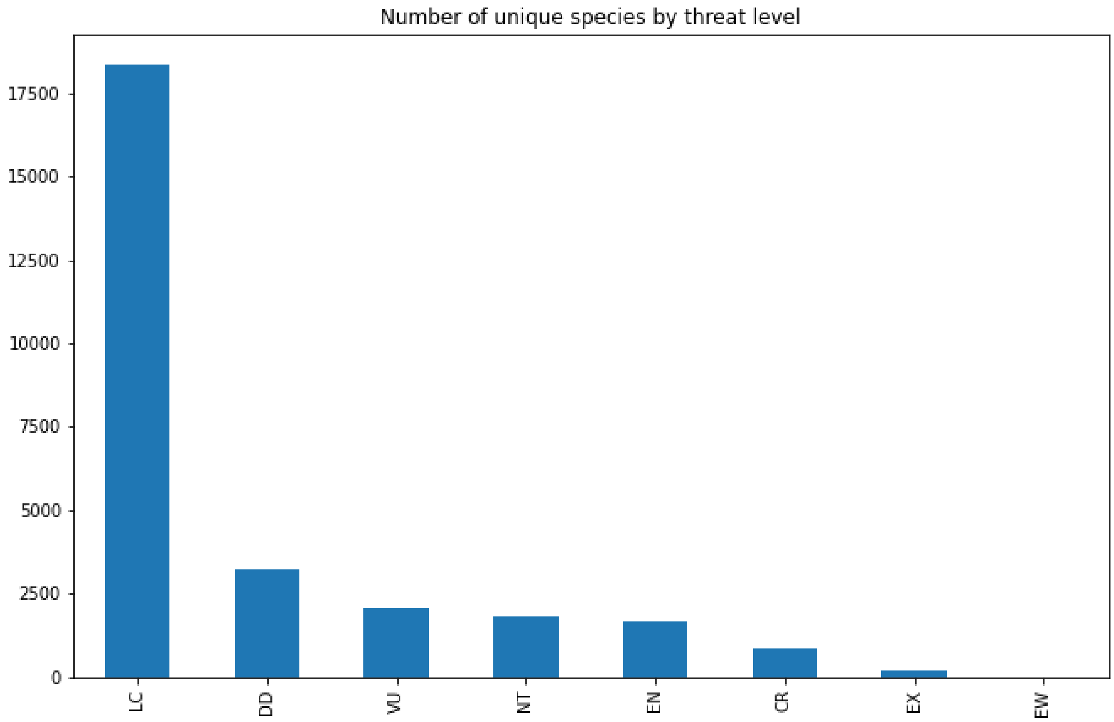
[18]:
```
fig, ax = plt.subplots(figsize=(11, 7))
ax.set_title('Number of unique species by group')
species.group.value_counts().plot(kind='bar')
plt.show()
```



[19]:
```
fig, ax = plt.subplots(figsize=(11, 7))
ax.set_title('Number of unique species by trend')
species.trend.value_counts().plot(kind='bar')
plt.show()
```



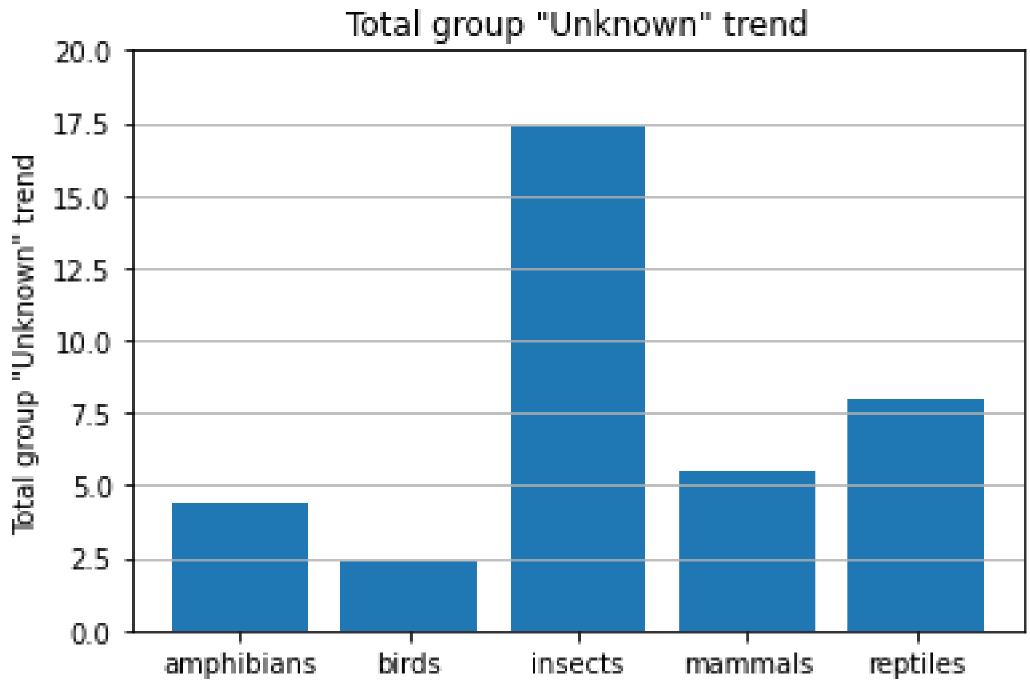
[20]:
```
fig, ax = plt.subplots(figsize=(11, 7))
ax.set_title('Number of unique species by threat level')
species.threat_level.value_counts().plot(kind='bar')
plt.show()
```



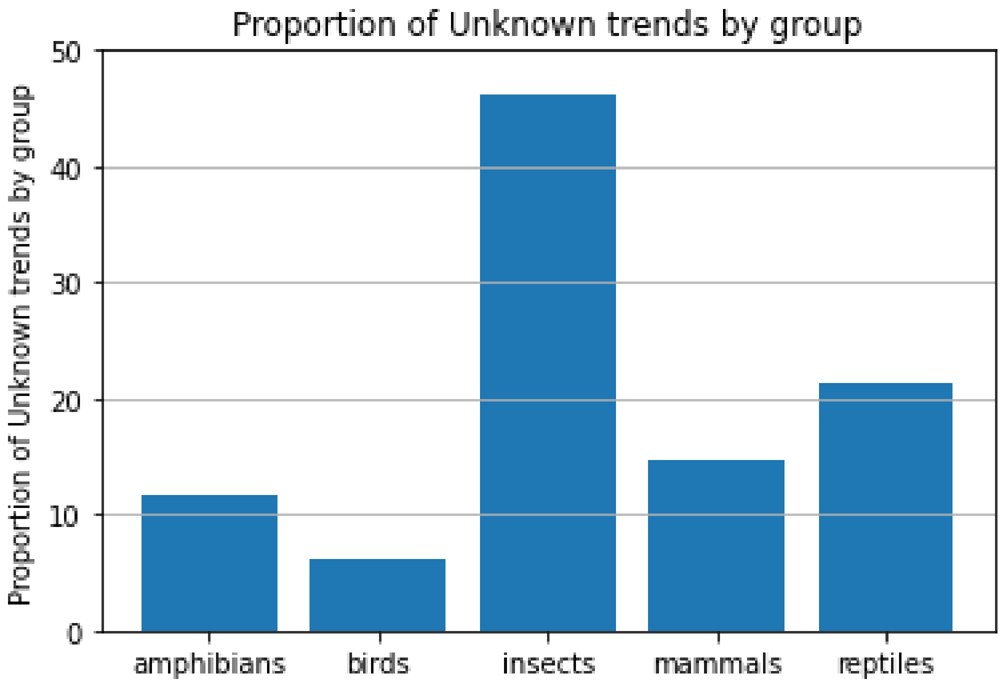
3.5 Evaluate reason for missing trends

It can be seen above that a significant portion of the trends are classified as 'Unknown'. Here we will analyse how these are distributed to be able to conclude if and how the predictions for the trends may be impeded.

[21]:
```
miss_trend = species[species.trend == 'Unknown'].sort_values('group')

miss_trend_group = miss_trend.drop(['threat_level'], axis=1)

miss_trend_group = miss_trend_group.groupby(['trend', 'group']).count().
 ↪scientific_name
total_group = species.sort_values('group').groupby(['group']).count().
 ↪scientific_name

plt.bar(species.sort_values('group').group.unique(), miss_trend_group /␣
 ↪sum(total_group) * 100)
plt.title('Total group "Unknown" trend')
plt.ylabel('Total group "Unknown" trend')
plt.ylim(0,20)
plt.grid(axis='y')
plt.show()
```



[22]:
```
plt.bar(species.sort_values('group').group.unique(), miss_trend_group /
 ↪sum(miss_trend_group) * 100)
plt.title('Proportion of Unknown trends by group')
plt.ylabel('Proportion of Unknown trends by group')
plt.ylim(0,50)
plt.grid(axis='y')

plt.show()
```

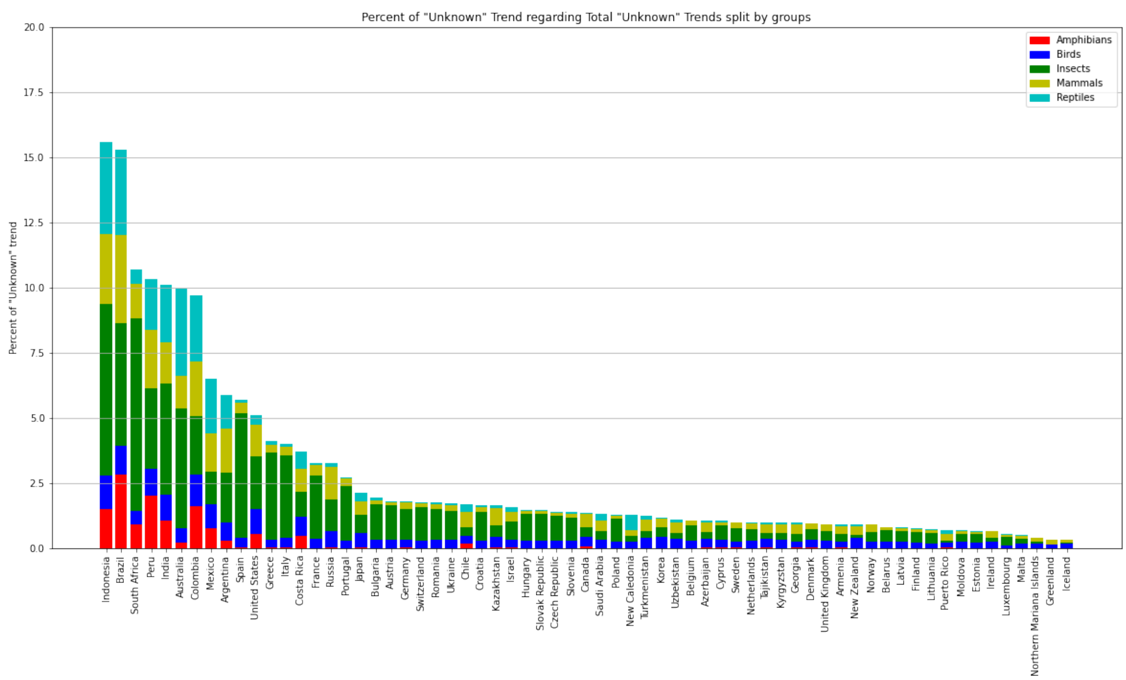


[23]:
```
# transform data
grouped = IUCN_cleaned_data.groupby(['Country', 'trend'])['scientific_name'].
    ↪count().reset_index(name='count')
species_trend_country = grouped.pivot_table(index='Country', columns='trend',␣
    ↪values='count')
species_trend_country = species_trend_country.fillna(0.0)
species_trend_country['Total'] = species_trend_country.sum(axis=1)
species_trend_country.shape

miss_trend_country = species_trend_country.copy()
miss_trend_country['Percent_Unknown'] = miss_trend_country.Unknown /␣
    ↪sum(miss_trend_group) * 100
miss_trend_country['Proportion_Unknown'] = miss_trend_country.Unknown /␣
    ↪miss_trend_country.Total * 100
```

[24]:
```
miss_trend_country_group = IUCN_cleaned_data[IUCN_cleaned_data.
    ↪trend=='Unknown'].groupby(['Country', 'group'])['scientific_name'].count().
    ↪reset_index(name='count')

miss_trend_country_group = miss_trend_country_group.set_index('Country')

ind = miss_trend_country.drop(['Decreasing', 'Increasing', 'Stable'], axis=1).
    ↪sort_values('Percent_Unknown', ascending=False)
```

```
miss_tcg_amphibians = miss_trend_country_group[miss_trend_country_group.group␣
↪== 'amphibians']
miss_tcg_birds = miss_trend_country_group[miss_trend_country_group.group ==␣
↪'birds']
miss_tcg_insects = miss_trend_country_group[miss_trend_country_group.group ==␣
↪'insects']
miss_tcg_mammals = miss_trend_country_group[miss_trend_country_group.group ==␣
↪'mammals']
miss_tcg_reptiles = miss_trend_country_group[miss_trend_country_group.group ==␣
↪'reptiles']

miss_tcg_amphibians = pd.concat([ind, miss_tcg_amphibians], axis=1).fillna(0).
↪drop(['Unknown', 'group'], axis=1)
miss_tcg_birds = pd.concat([ind, miss_tcg_birds], axis=1).fillna(0).
↪drop(['Unknown', 'group'], axis=1)
miss_tcg_insects = pd.concat([ind, miss_tcg_insects], axis=1).fillna(0).
↪drop(['Unknown', 'group'], axis=1)
miss_tcg_mammals = pd.concat([ind, miss_tcg_mammals], axis=1).fillna(0).
↪drop(['Unknown', 'group'], axis=1)
miss_tcg_reptiles = pd.concat([ind, miss_tcg_reptiles], axis=1).fillna(0).
↪drop(['Unknown', 'group'], axis=1)

miss_tcg_amphibians['Percent_Unknown'] = miss_tcg_amphibians['count'] /␣
↪sum(miss_trend_group) * 100
miss_tcg_birds['Percent_Unknown'] = miss_tcg_birds['count'] /␣
↪sum(miss_trend_group) * 100
miss_tcg_insects['Percent_Unknown'] = miss_tcg_insects['count'] /␣
↪sum(miss_trend_group) * 100
miss_tcg_mammals['Percent_Unknown'] = miss_tcg_mammals['count'] /␣
↪sum(miss_trend_group) * 100
miss_tcg_reptiles['Percent_Unknown'] = miss_tcg_reptiles['count'] /␣
↪sum(miss_trend_group) * 100

countries = ind.reset_index()['Country']

amphibians = miss_tcg_amphibians['Percent_Unknown'].values
birds = miss_tcg_birds['Percent_Unknown'].values
insects = miss_tcg_insects['Percent_Unknown'].values
mammals = miss_tcg_mammals['Percent_Unknown'].values
reptiles = miss_tcg_reptiles['Percent_Unknown'].values

plt.figure(figsize=(20,10))
plt.bar(countries, amphibians, color='r')
plt.bar(countries, birds, bottom=amphibians, color='b')
plt.bar(countries, insects, bottom=birds + amphibians, color='g')
plt.bar(countries, mammals, bottom=insects + birds + amphibians, color='y')
```

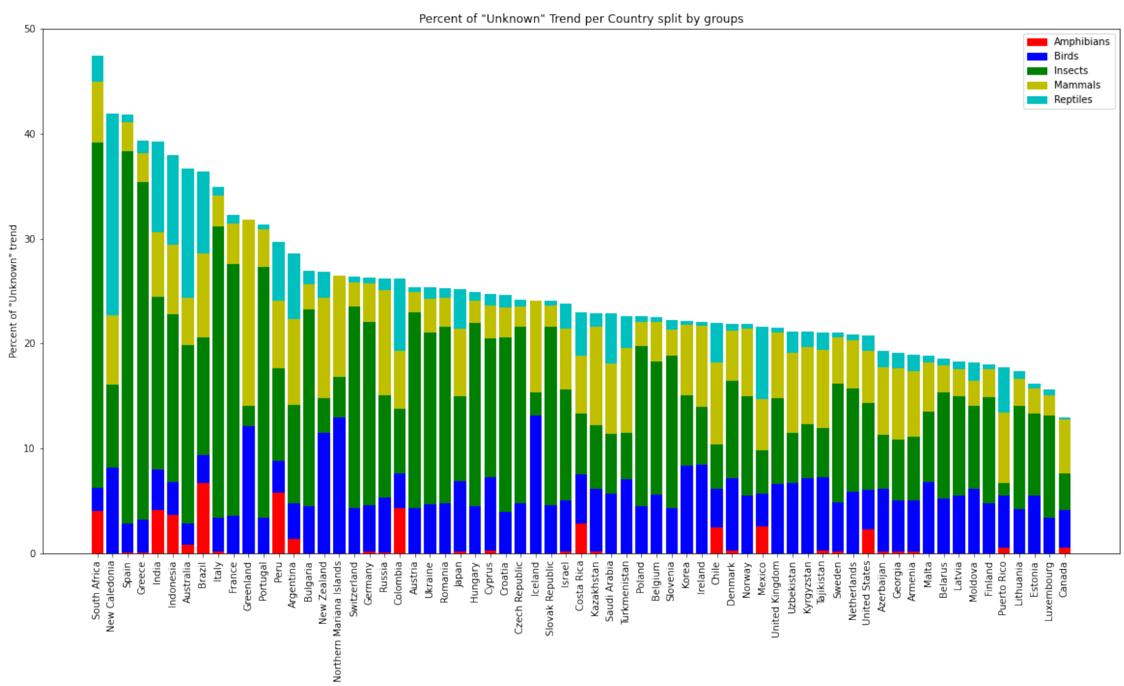
```
plt.bar(countries, reptiles, bottom=mammals + insects + birds + amphibians,␣
 ↪color='c')

plt.title('Percent of "Unknown" Trend regarding Total "Unknown" Trends split by␣
 ↪groups')
plt.xticks(rotation='vertical')
plt.ylim(0,20)
plt.ylabel('Percent of "Unknown" trend')

colors = {'Amphibians':'r', 'Birds':'b', 'Insects':'g', 'Mammals':'y',␣
 ↪'Reptiles':'c'}
labels = list(colors.keys())
handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]
plt.legend(handles, labels)
plt.grid(axis='y')

plt.show()
```



[25]:
```
miss_trend_country_group = IUCN_cleaned_data[IUCN_cleaned_data.
 ↪trend=='Unknown'].groupby(['Country', 'group'])['scientific_name'].count().
 ↪reset_index(name='count')

miss_trend_country_group = miss_trend_country_group.set_index('Country')
```

```
ind = miss_trend_country.drop(['Decreasing', 'Increasing', 'Stable'], axis=1).
 ↪sort_values('Proportion_Unknown', ascending=False)

miss_tcg_amphibians = miss_trend_country_group[miss_trend_country_group.group␣
 ↪== 'amphibians']
miss_tcg_birds = miss_trend_country_group[miss_trend_country_group.group ==␣
 ↪'birds']
miss_tcg_insects = miss_trend_country_group[miss_trend_country_group.group ==␣
 ↪'insects']
miss_tcg_mammals = miss_trend_country_group[miss_trend_country_group.group ==␣
 ↪'mammals']
miss_tcg_reptiles = miss_trend_country_group[miss_trend_country_group.group ==␣
 ↪'reptiles']

miss_tcg_amphibians = pd.concat([ind, miss_tcg_amphibians], axis=1).fillna(0).
 ↪drop(['Unknown', 'group'], axis=1)
miss_tcg_birds = pd.concat([ind, miss_tcg_birds], axis=1).fillna(0).
 ↪drop(['Unknown', 'group'], axis=1)
miss_tcg_insects = pd.concat([ind, miss_tcg_insects], axis=1).fillna(0).
 ↪drop(['Unknown', 'group'], axis=1)
miss_tcg_mammals = pd.concat([ind, miss_tcg_mammals], axis=1).fillna(0).
 ↪drop(['Unknown', 'group'], axis=1)
miss_tcg_reptiles = pd.concat([ind, miss_tcg_reptiles], axis=1).fillna(0).
 ↪drop(['Unknown', 'group'], axis=1)

miss_tcg_amphibians['Proportion_Unknown'] = miss_tcg_amphibians['count'] /␣
 ↪miss_tcg_amphibians.Total * 100
miss_tcg_birds['Proportion_Unknown'] = miss_tcg_birds['count'] / miss_tcg_birds.
 ↪Total * 100
miss_tcg_insects['Proportion_Unknown'] = miss_tcg_insects['count'] /␣
 ↪miss_tcg_insects.Total * 100
miss_tcg_mammals['Proportion_Unknown'] = miss_tcg_mammals['count'] /␣
 ↪miss_tcg_mammals.Total * 100
miss_tcg_reptiles['Proportion_Unknown'] = miss_tcg_reptiles['count'] /␣
 ↪miss_tcg_reptiles.Total * 100

countries = ind.reset_index()['Country']

amphibians = miss_tcg_amphibians['Proportion_Unknown'].values
birds = miss_tcg_birds['Proportion_Unknown'].values
insects = miss_tcg_insects['Proportion_Unknown'].values
mammals = miss_tcg_mammals['Proportion_Unknown'].values
reptiles = miss_tcg_reptiles['Proportion_Unknown'].values

plt.figure(figsize=(20,10))
plt.bar(countries, amphibians, color='r')
```

```
plt.bar(countries, birds, bottom=amphibians, color='b')
plt.bar(countries, insects, bottom=birds + amphibians, color='g')
plt.bar(countries, mammals, bottom=insects + birds + amphibians, color='y')
plt.bar(countries, reptiles, bottom=mammals + insects + birds + amphibians, color='c')

plt.title('Percent of "Unknown" Trend per Country split by groups')
plt.xticks(rotation='vertical')
plt.ylim(0,50)
plt.ylabel('Percent of "Unknown" trend')

colors = {'Amphibians':'r', 'Birds':'b', 'Insects':'g', 'Mammals':'y', 'Reptiles':'c'}
labels = list(colors.keys())
handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]
plt.legend(handles, labels)

plt.show()
```



It can be seen, that most of the missing values can be atributed to insects and reptiles, this can influence the predictions for this groups. Regarding distribution amoung countries, some countries with a higher biodiversity have a larger share of missing values. When comparing the unknown trends amoung the other trends inside the country most countries have around 20-40 Percent of entries categorized as “Unknown”, therefore we don’t expect a difference in predictability coming from the countries.

4 Support Data Preparation

4.1 World Bank Climate

The climate dataset was obtained from The World Bank Group. Sadly there was no data accessible for the climate of 2020. Thus we had to work with data from 1990 to 2016 which was available. Because the data wasn't completely representative for the year 2020, we tried to extract features of the temperature growth. ### Constants

[26]:
```
TEMP_DATA = 'data/climate/temperature_data_1991_2016.csv'
RAIN_DATA = 'data/climate/rain_data_1991_2016.csv'
OUTPUT_PATH = 'data/climate/climate_features.csv'
```

4.1.1 Temperature Data

Load and Transform Data

[27]:
```
data_temp = pd.read_csv(
    TEMP_DATA,
    sep=',',
    names=['Temperature', 'Year', 'Statistics', 'Country', 'ISO_Country',
           '_']).drop(0)
data_temp['Month'] = data_temp['Statistics'].apply(lambda x: x.split()[0])
data_temp['Country'] = data_temp['Country'].apply(lambda x: x.lstrip())
data_temp['Temperature'] = data_temp['Temperature'].astype(float)
data_temp['Year'] = data_temp['Year'].astype(int)
data_temp = data_temp[['Temperature', 'Year', 'Month', 'Country']]
```

4.1.2 Overview

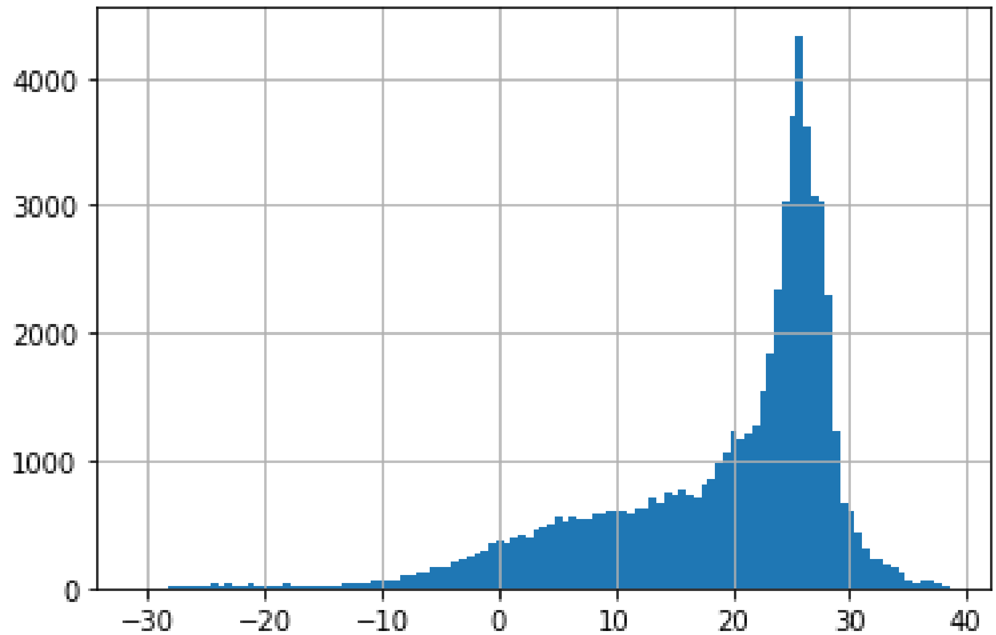
[28]:
```
data_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 61152 entries, 1 to 61152
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Temperature  61152 non-null  float64
 1   Year         61152 non-null  int64
 2   Month        61152 non-null  object
 3   Country      61152 non-null  object
dtypes: float64(1), int64(1), object(2)
memory usage: 2.3+ MB
```

Temperature

[29]:
```
data_temp['Temperature'].hist(bins='auto')
data_temp['Temperature'].describe()
```

```
[29]: count    61152.000000
      mean        19.224302
      std         10.136161
      min        -30.859000
      25%         13.824000
      50%         23.322650
      75%         26.244025
      max         38.566900
      Name: Temperature, dtype: float64
```

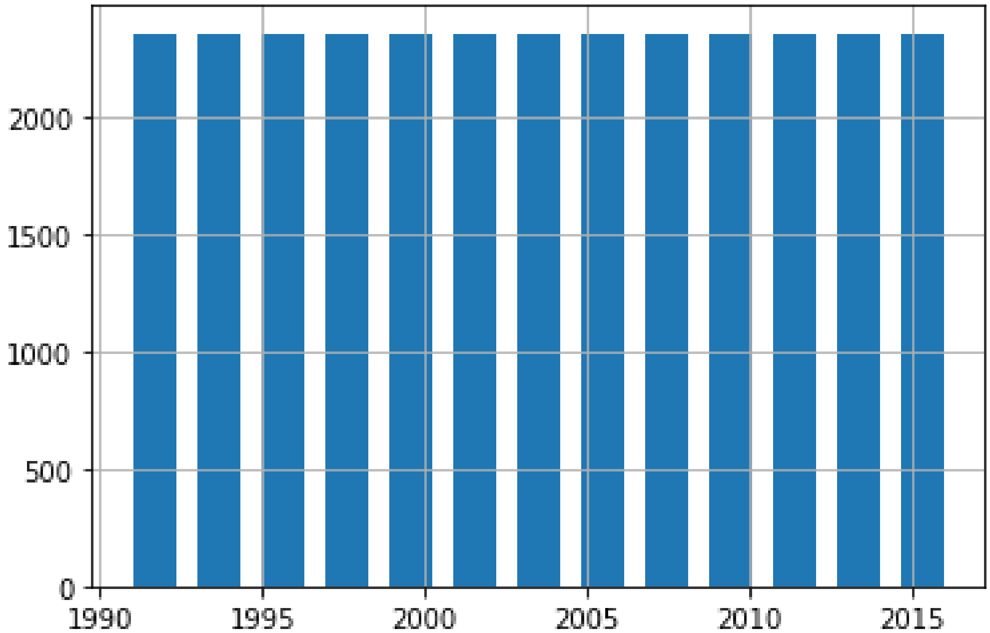


Year

```
[30]: display(data_temp['Year'].describe())
      data_temp['Year'].hist(bins='auto')
```

```
count    61152.000000
mean      2003.500000
std          7.500061
min       1991.000000
25%       1997.000000
50%       2003.500000
75%       2010.000000
max       2016.000000
Name: Year, dtype: float64
```

```
[30]: <AxesSubplot:>
```

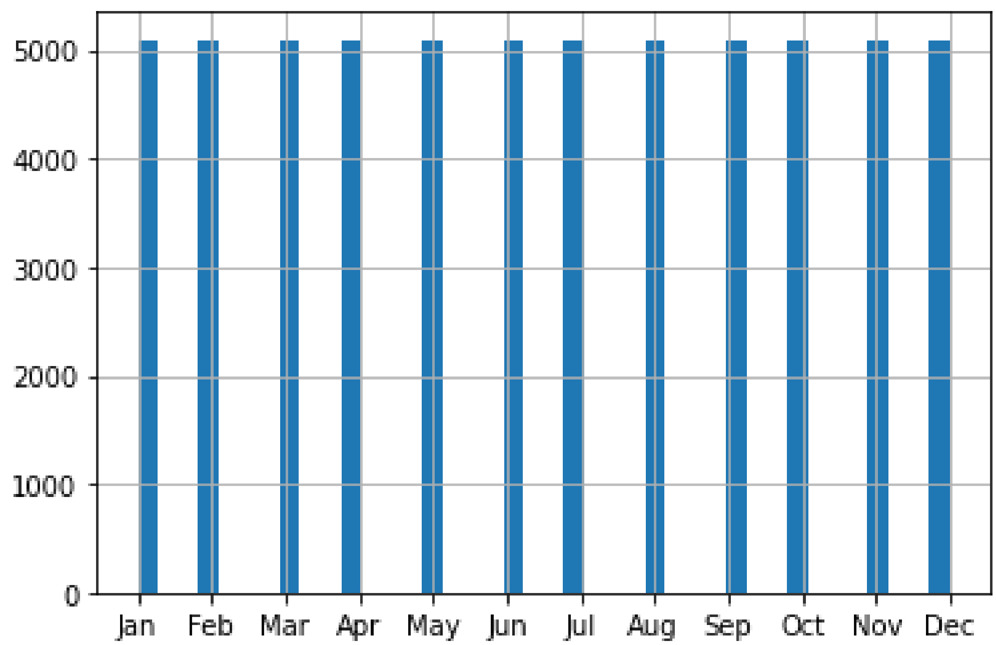


Month

[31]:
```
display(data_temp['Month'].describe())
data_temp['Month'].hist(bins='auto')
```

```
count     61152
unique       12
top         Jul
freq       5096
Name: Month, dtype: object
```

[31]: <AxesSubplot:>



Country

[32]:
```
display(data_temp['Country'].describe())
```

```
count      61152
unique       195
top        Korea
freq         624
Name: Country, dtype: object
```

[33]:
```
# Check if data contains all Countries we have in our country list
set(SELECTED_COUNTRIES).difference(set(data_temp['Country'].unique()))
```

[33]:
```
{'Slovak Republic'}
```

[34]:
```
# --> Slovakia has to be renamed to Slovak Republic
data_temp.loc[data_temp['Country'] == 'Slovakia', 'Country'] = 'Slovak Republic'
set(SELECTED_COUNTRIES).difference(set(data_temp['Country'].unique()))
```

[34]:
```
set()
```

4.1.3 Rainfall Data

Load and Transform Data

[35]:
```
data_rain = pd.read_csv(RAIN_DATA, sep=',',
    names=['Rainfall', 'Year', 'Statistics', 'Country', 'ISO_Country',
```

```
          '_']).drop(0)
data_rain['Rainfall'] = data_rain['Rainfall'].astype(float)
data_rain['Month'] = data_rain['Statistics'].apply(lambda x: x.split()[0])
data_rain['Country'] = data_rain['Country'].apply(lambda x: x.lstrip())
data_rain['Year'] = data_rain['Year'].astype(int)
```

4.1.4 Overview

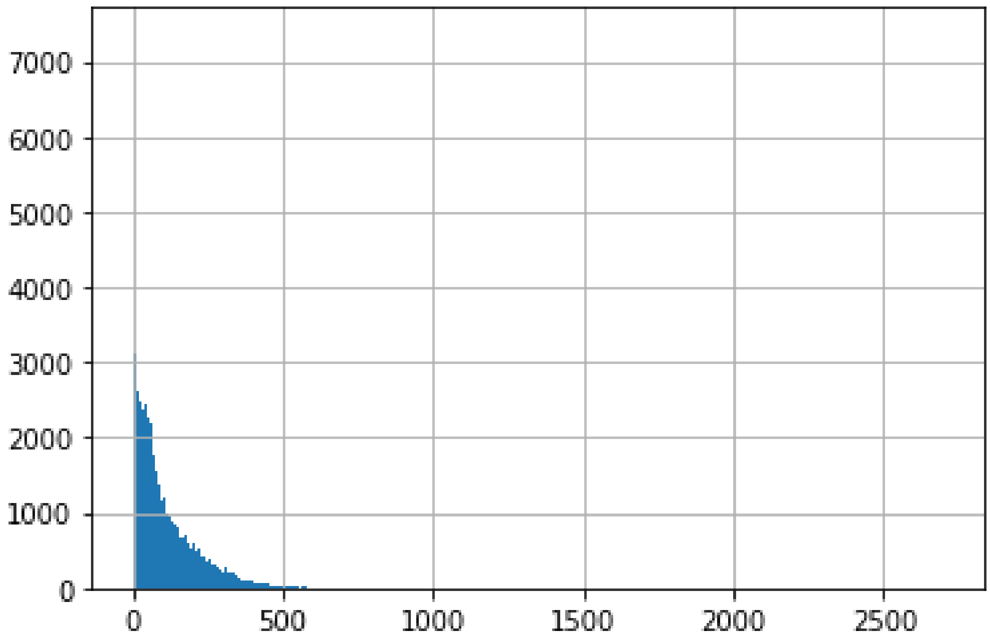
[36]:
```
data_rain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 61152 entries, 1 to 61152
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Rainfall     61152 non-null  float64
 1   Year         61152 non-null  int64
 2   Statistics   61152 non-null  object
 3   Country      61152 non-null  object
 4   ISO_Country  61152 non-null  object
 5   _            1560 non-null   object
 6   Month        61152 non-null  object
dtypes: float64(1), int64(1), object(5)
memory usage: 3.7+ MB
```

4.1.5 Rainfall

[37]:
```
data_rain['Rainfall'].hist(bins='auto')
data_rain['Rainfall'].describe()
```

[37]:
```
count    61152.000000
mean       103.581125
std        114.130057
min          0.000000
25%         24.123175
50%         66.192300
75%        149.172000
max       2699.190000
Name: Rainfall, dtype: float64
```

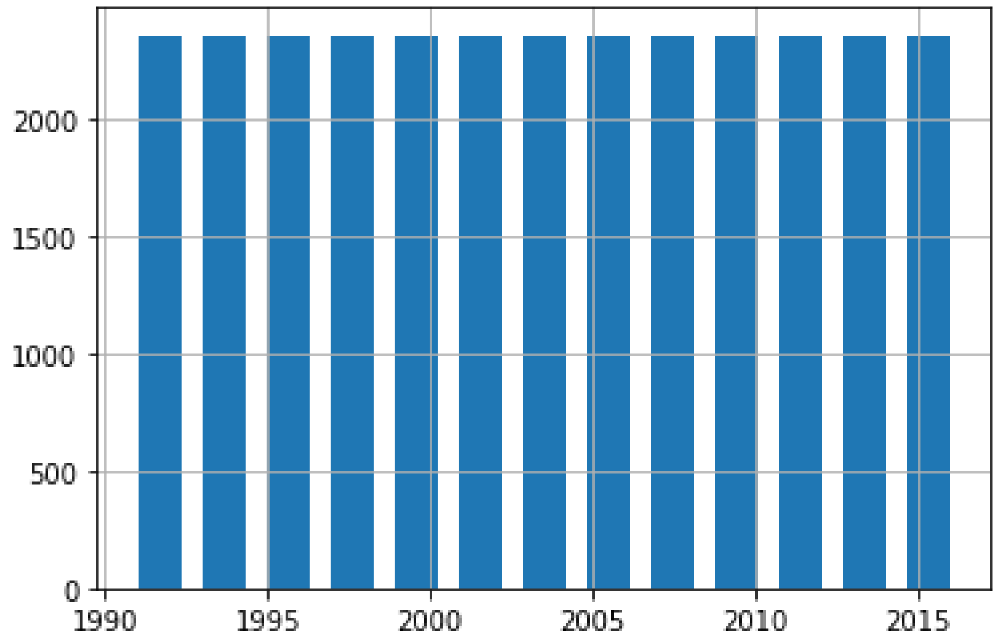


Year

[38]:
```
display(data_rain['Year'].describe())
data_rain['Year'].hist(bins='auto')
```

```
count    61152.000000
mean      2003.500000
std          7.500061
min       1991.000000
25%       1997.000000
50%       2003.500000
75%       2010.000000
max       2016.000000
Name: Year, dtype: float64
```

[38]: <AxesSubplot:>

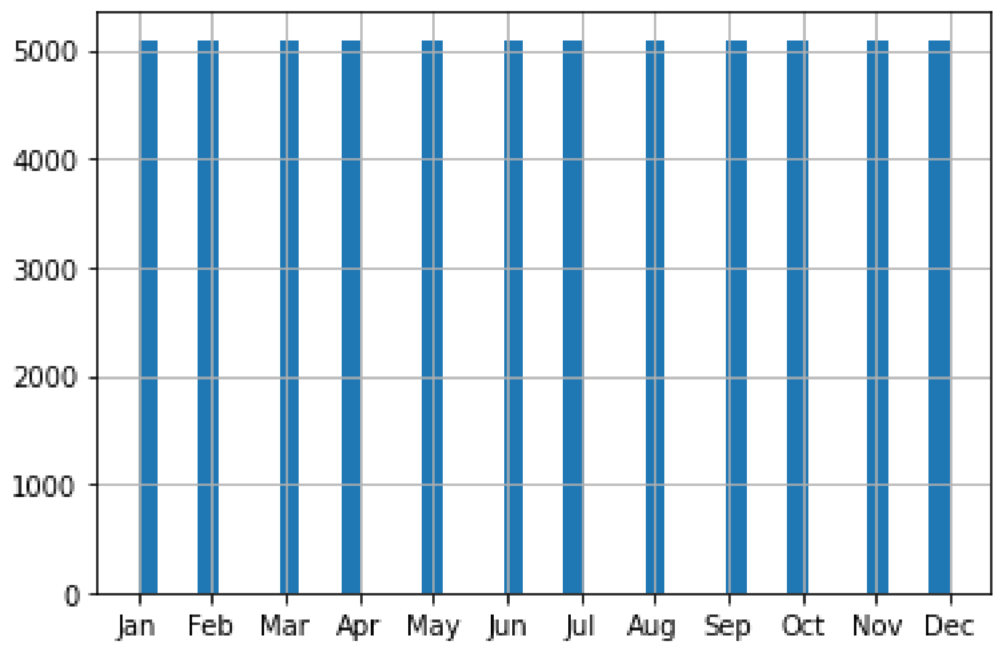


Month

[39]:
```
display(data_rain['Month'].describe())
data_rain['Month'].hist(bins='auto')
```

```
count      61152
unique        12
top          Jul
freq        5096
Name: Month, dtype: object
```

[39]: <AxesSubplot:>



Country

[40]:
```
# Check if data contains all Countries we have in our country list
set(SELECTED_COUNTRIES).difference(set(data_rain['Country'].unique()))
```

[40]:
```
{'Slovak Republic'}
```

[41]:
```
# --> Slovakia has to be renamed to Slovak Republic
data_rain.loc[data_rain['Country'] == 'Slovakia', 'Country'] = 'Slovak Republic'
set(SELECTED_COUNTRIES).difference(set(data_rain['Country'].unique()))
```

[41]:
```
set()
```

4.1.6 Merge Datasets

Check if Countries, Years and Months are identical

[42]:
```
country_temp = set(data_temp['Country'].unique())
country_rain = set(data_rain['Country'].unique())
print('Matching country keys: {}'.format(country_temp == country_rain))
```

```
Matching country keys: True
```

[43]:
```
year_temp = set(data_temp['Year'].unique())
year_rain = set(data_rain['Year'].unique())
print('Matching country keys: {}'.format(year_temp == year_rain))
```

```
Matching country keys: True
```

[44]:
```
month_temp = set(data_temp['Month'].unique())
month_rain = set(data_rain['Month'].unique())
print('Matching country keys: {}'.format(month_temp == month_rain))
```

```
Matching country keys: True
```

Merge

[45]:
```
data_full = data_temp.merge(data_rain)
```

4.1.7 Show change over time for all countries averaged

GroupBy Year and Country

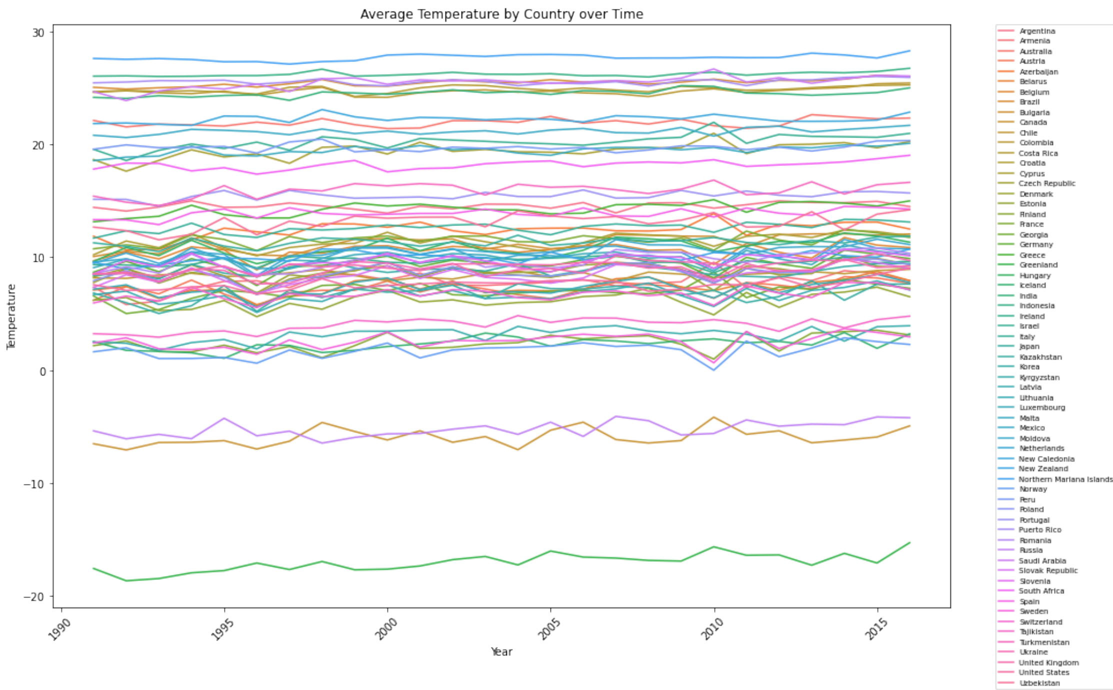
[46]:
```
by_year = data_full.groupby(['Year', 'Country']).agg(np.mean).reset_index()
```

Only show OECD Countries

[47]:
```
by_year = by_year[by_year['Country'].isin(SELECTED_COUNTRIES)]
```

Plot Average Temperature by Country over Time

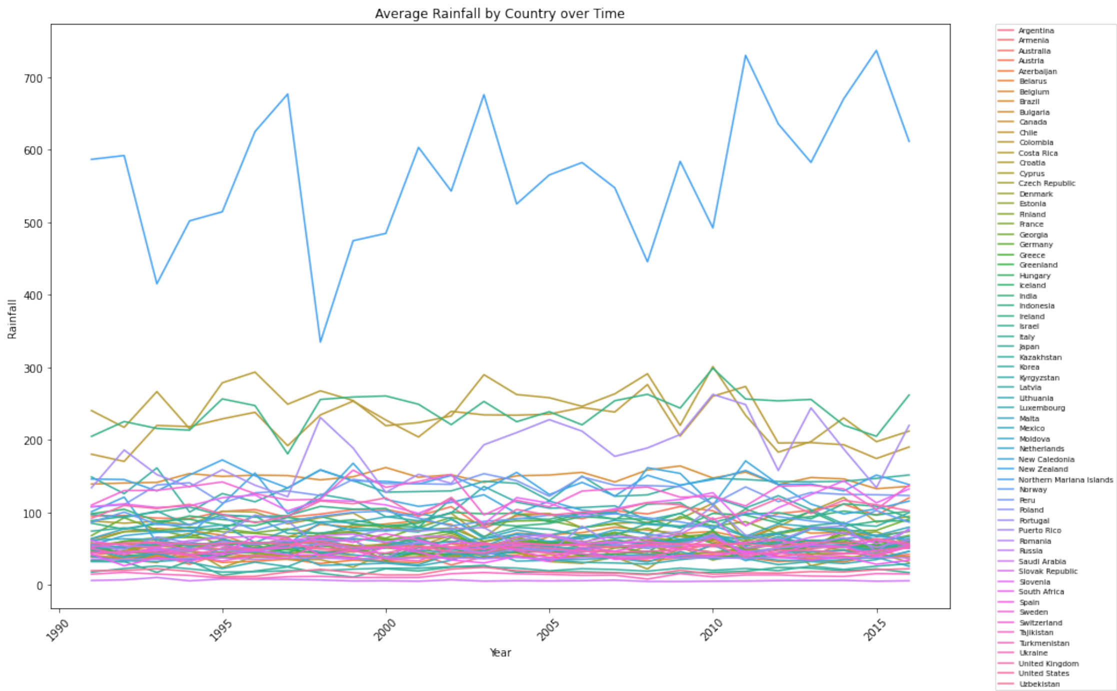
[48]:
```
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Average Temperature by Country over Time')

sns.lineplot(data=by_year.reset_index(),
             x='Year',
             y='Temperature',
             hue='Country')
plt.xticks(rotation=45)

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)

plt.show()
```



There is low variance in the temperature of the last 30 years. As we don't get any data in better quality (and from 2020), we have to extract features out of this dataset and use it as support data.

Plot Average Rainfall by Country over Time

[49]:
```
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Average Rainfall by Country over Time')

sns.lineplot(data=by_year.reset_index(),
             x='Year',
             y='Rainfall',
             hue='Country')
plt.xticks(rotation=45)

plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)

plt.show()
```



The rainfall for a particular country doesn't seem to be as stable as the temperature. As we have a lot of other different support data sets, we are going to discard this feature.

4.1.8 Extract Temperature Features

[50]:
```python
features = pd.DataFrame(data_full['Country'].unique(), columns=['Country'])

## fit regression line to data
def extract_slope(x, y):
    m, b = np.polyfit(x, y, 1)
    return m

## extract percentual temperature gain from first to last year
def extract_gain_percentage(country_df, past_years=1):

    min_year = country_df['Year'].min()
    start_mean_temp = country_df[country_df['Year'] ==
                                 min_year]['Temperature'].mean()

    max_year = country_df['Year'].max()

    past_years_mean_temp = country_df[country_df['Year'] > (
        max_year - past_years)]['Temperature'].mean()

    return (past_years_mean_temp / start_mean_temp - 1) * 100
```

```
## extract absolute temperature difference from first to last year
def extract_difference(country_df, past_years=1):
    min_year = country_df['Year'].min()
    start_mean_temp = country_df[country_df['Year'] ==
                                  min_year]['Temperature'].mean()

    max_year = country_df['Year'].max()

    past_years_mean_temp = country_df[country_df['Year'] > (
        max_year - past_years)]['Temperature'].mean()

    return past_years_mean_temp - start_mean_temp

for country in features['Country'].unique():
    sel_c = data_full.loc[data_full['Country'] == country, :]

    ## extract temperature slope
    features.loc[features['Country'] == country,
                 'temp_slope'] = extract_slope(sel_c['Year'],
                                               sel_c['Temperature'])
    ## extract temperature gain percentage
    features.loc[features['Country'] == country,
                 'gain_percentage'] = extract_gain_percentage(sel_c)

    ## extract temperature difference
    features.loc[features['Country'] == country,
                 'temp_difference'] = extract_difference(sel_c)
```

[51]:
```
ds_climate = features.copy()
```

4.2 Greenhouse gasses

The greenhouse gases were taken as a dataset, because of our believe, that the output of greenhouse gasses can decrease the survivability in the region. We used the openly available dataset from the OECD that can be downloaded from https://stats.oecd.org/Index.aspx?DataSetCode=AIR_GHG To normalize these values and make them comparable to each other we calculated the output per inhabitant. This makes large countries comparable to smaller countries. For this step we used the historic Population dataset from oecd (https://stats.oecd.org/Index.aspx?DataSetCode=HISTPOP)

4.2.1 Load Data

[52]:
```
AIR_GHG = DATA_PATH / 'OECD' / 'AIR_GHG.csv'
df = pd.read_csv(AIR_GHG)
```

4.2.2 Cleaning Data

Resolve Power

[53]:
```
df.Value = df.Value * 10 **  df['PowerCode Code']
```

Filter only for totals

[54]:
```
df = df[df['VAR'] == 'TOTAL']
```

Drop Estimates

[55]:
```
df = df[df['Flag Codes'].isnull()]
```

Delete unneaded columns

[56]:
```
df = df.drop(labels=['COU','Pollutant', 'VAR', 'Variable', 'Year', 'Unit Code',␣
↪'Unit', 'PowerCode Code', 'PowerCode', 'Reference Period Code', 'Reference␣
↪Period', 'Flag Codes', 'Flags'], axis=1)
```

Delete old Data (< 2005)

[57]:
```
df = df[df.YEA > 2005]
```

4.2.3 Feature Preperation

Normalize Data

[58]:
```
HISTPOP = DATA_PATH / 'OECD' / 'HISTPOP.csv'

pop = pd.read_csv(HISTPOP)
pop = pop[pop.SEX == 'T']
pop = pop[pop.AGE == 'TOTAL']

for i in df.index:
    ctr = df['Country'][i]
    yea = df['YEA'][i]
    norm = pop[(pop.Country == ctr) & (pop.Time == yea)].Value
    if norm.empty:
        norm = 1.0

    df['Value'][i]=  df['Value'][i] / norm
```

Transform data into years

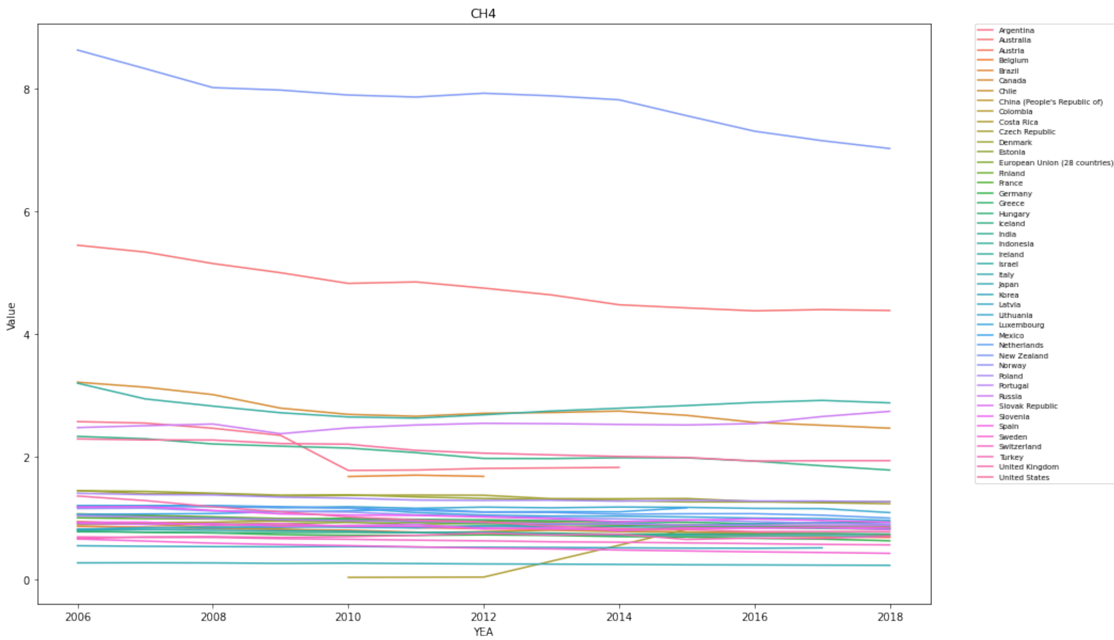
[59]:
```
df = df.pivot(index=['Country', 'POL'], columns='YEA', values=['Value']).
↪reset_index()

df['2019'] = np.NaN
df['2020'] = np.NaN
```

Extract Polution Type

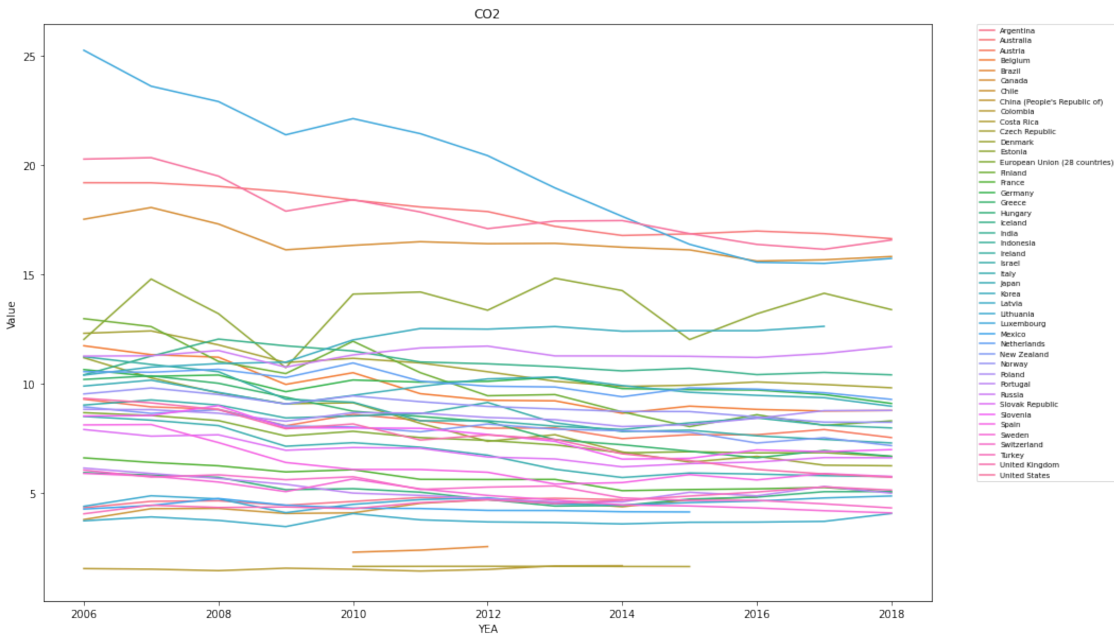
[60]:
```
CH4 = df[df['POL'] == 'CH4']
CO2 = df[df['POL'] == 'CO2']
HFC = df[df['POL'] == 'HFC']
HFC_PFC = df[df['POL'] == 'HFC_PFC']
N2O = df[df['POL'] == 'N2O']
NF3 = df[df['POL'] == 'NF3']
PFC = df[df['POL'] == 'PFC']
SF6 = df[df['POL'] == 'SF6']

CH4 = CH4.drop(labels=['POL'], axis=1)
CO2 = CO2.drop(labels=['POL'], axis=1)
HFC = HFC.drop(labels=['POL'], axis=1)
HFC_PFC = HFC_PFC.drop(labels=['POL'], axis=1)
N2O = N2O.drop(labels=['POL'], axis=1)
NF3 = NF3.drop(labels=['POL'], axis=1)
PFC = PFC.drop(labels=['POL'], axis=1)
SF6 = SF6.drop(labels=['POL'], axis=1)
```

4.2.4 Analysis of previous Years

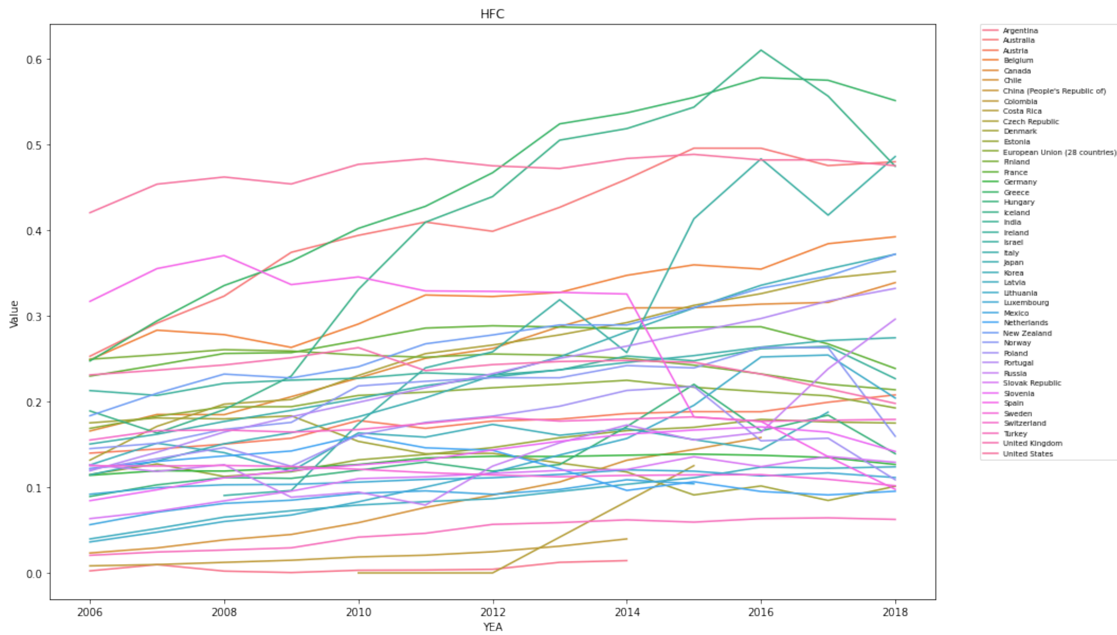
[61]:
```
CH4_plot = CH4.set_index('Country').stack().reset_index().drop(['2019',␣
↪'2020'], axis=1)
CH4_plot = CH4_plot[CH4_plot.YEA != '']
CH4_plot = CH4_plot[~pd.to_numeric(CH4_plot['Value'], errors='coerce').isnull()]
CH4_plot.Value = CH4_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('CH4')

sns.lineplot(
    data=CH4_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



[62]:
```
CO2_plot = CO2.set_index('Country').stack().reset_index().drop(['2019',␣
↪'2020'], axis=1)
CO2_plot = CO2_plot[CO2_plot.YEA != '']
CO2_plot = CO2_plot[~pd.to_numeric(CO2_plot['Value'], errors='coerce').isnull()]
CO2_plot.Value = CO2_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('CO2')

sns.lineplot(
    data=CO2_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```

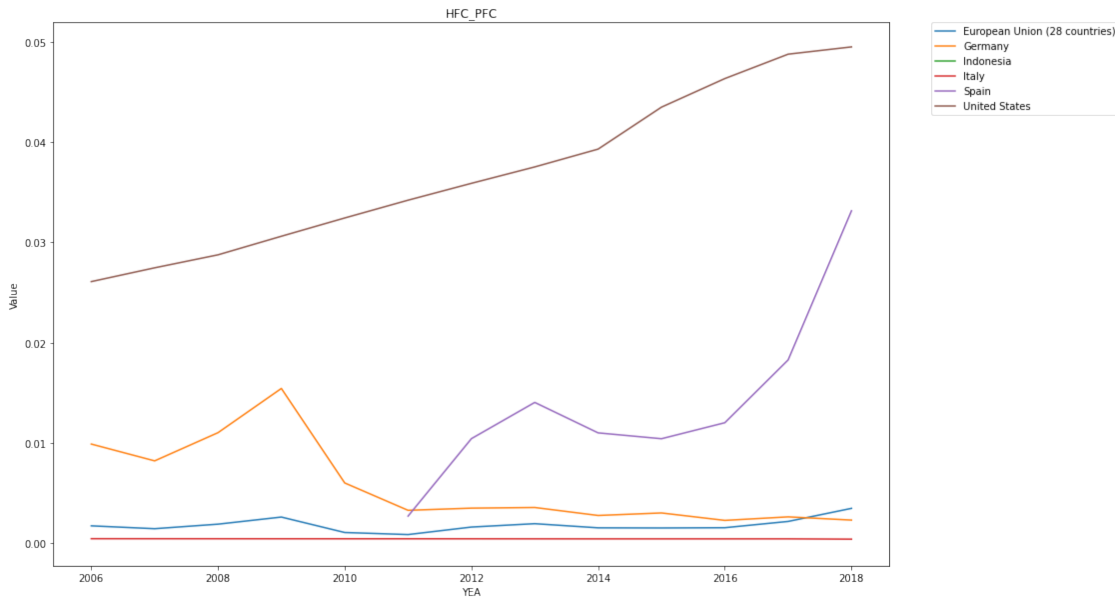


[63]:
```
HFC_plot = HFC.set_index('Country').stack().reset_index().drop(['2019',␣
 ↪'2020'], axis=1)
HFC_plot = HFC_plot[HFC_plot.YEA != '']
HFC_plot = HFC_plot[~pd.to_numeric(HFC_plot['Value'], errors='coerce').isnull()]
HFC_plot.Value = HFC_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('HFC')

sns.lineplot(
    data=HFC_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```

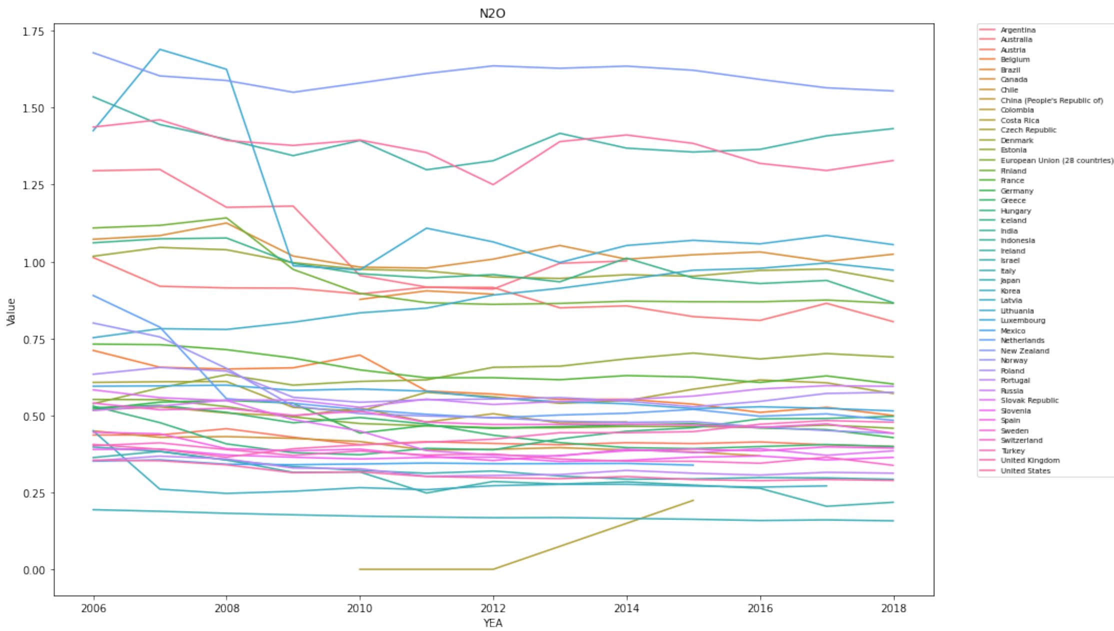


[64]:

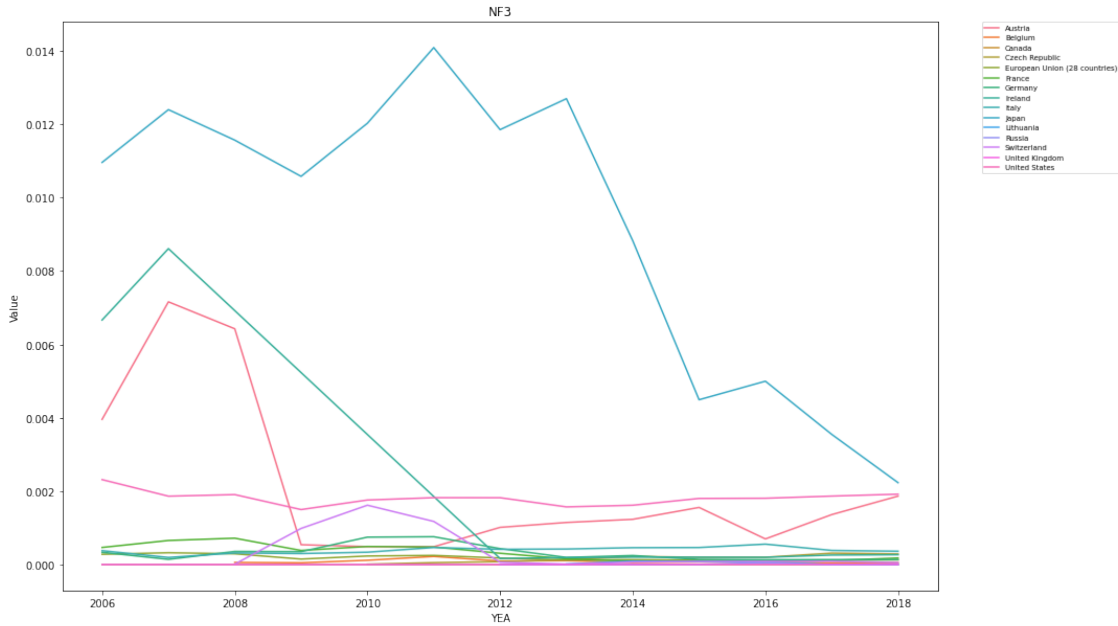
```python
HFC_PFC_plot = HFC_PFC.set_index('Country').stack().reset_index().drop(['2019',␣
↪'2020'], axis=1)
HFC_PFC_plot = HFC_PFC_plot[HFC_PFC_plot.YEA != '']
HFC_PFC_plot = HFC_PFC_plot[~pd.to_numeric(HFC_PFC_plot['Value'],␣
↪errors='coerce').isnull()]
HFC_PFC_plot.Value = HFC_PFC_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('HFC_PFC')

sns.lineplot(
    data=HFC_PFC_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.show()
```



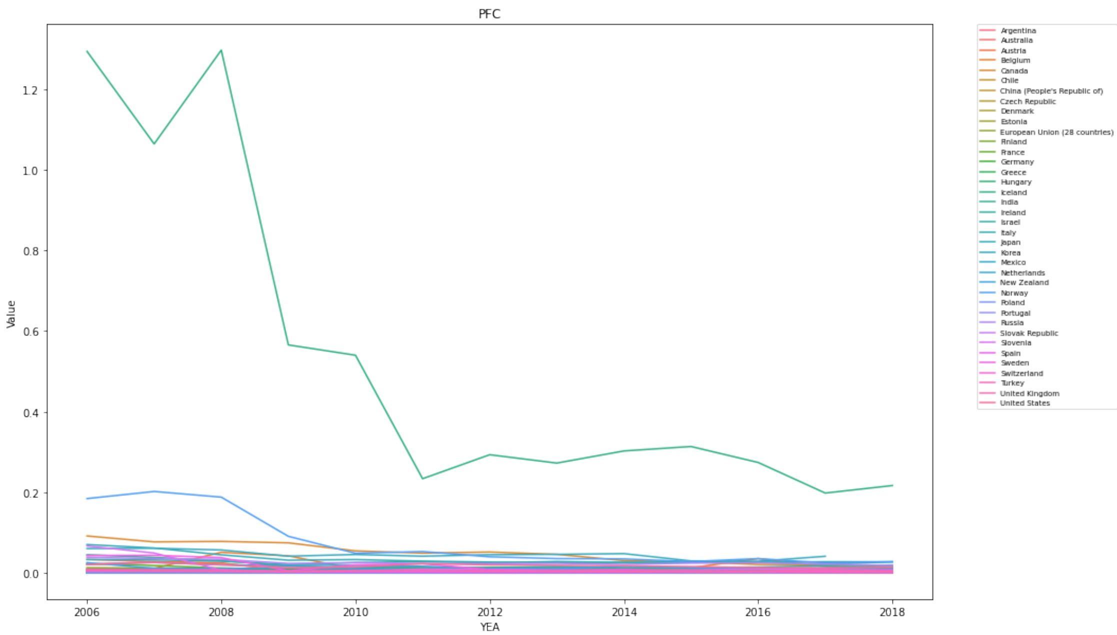
[65]:
```
N2O_plot = N2O.set_index('Country').stack().reset_index().drop(['2019',␣
 ↪'2020'], axis=1)
N2O_plot = N2O_plot[N2O_plot.YEA != '']
N2O_plot = N2O_plot[~pd.to_numeric(N2O_plot['Value'], errors='coerce').isnull()]
N2O_plot.Value = N2O_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('N2O')

sns.lineplot(
    data=N2O_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



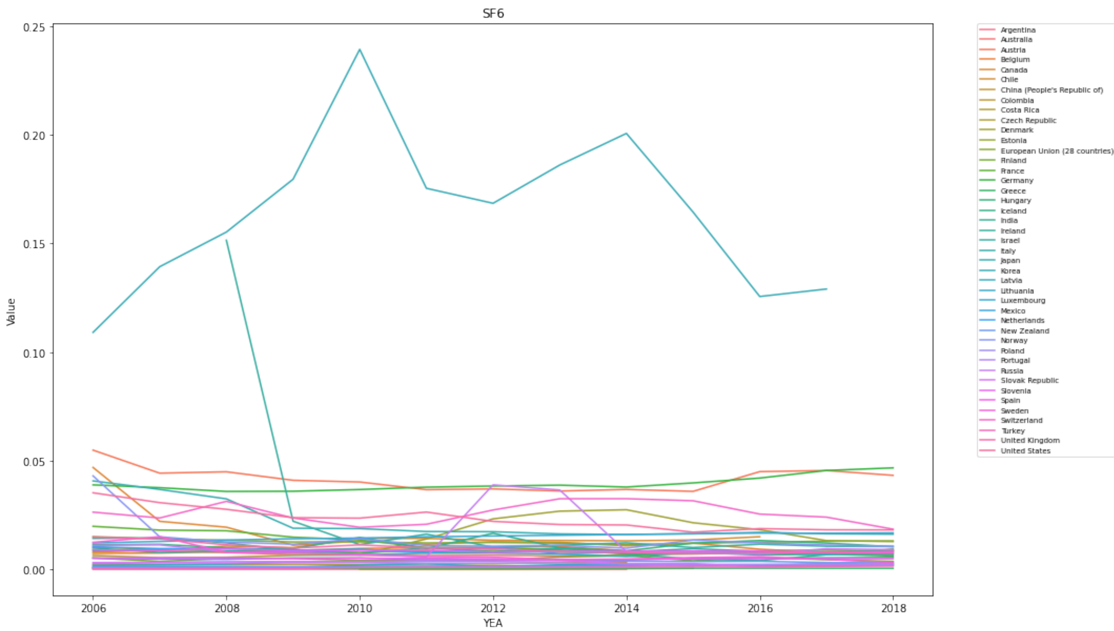
[66]:
```
NF3_plot = NF3.set_index('Country').stack().reset_index().drop(['2019',␣
↪'2020'], axis=1)
NF3_plot = NF3_plot[NF3_plot.YEA != '']
NF3_plot = NF3_plot[~pd.to_numeric(NF3_plot['Value'], errors='coerce').isnull()]
NF3_plot.Value = NF3_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('NF3')

sns.lineplot(
    data=NF3_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



[67]:
```
PFC_plot = PFC.set_index('Country').stack().reset_index().drop(['2019',␣
↪'2020'], axis=1)
PFC_plot = PFC_plot[PFC_plot.YEA != '']
PFC_plot = PFC_plot[~pd.to_numeric(PFC_plot['Value'], errors='coerce').isnull()]
PFC_plot.Value = PFC_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('PFC')

sns.lineplot(
    data=PFC_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



[68]:
```
SF6_plot = SF6.set_index('Country').stack().reset_index().drop(['2019',␣
 ↪'2020'], axis=1)
SF6_plot = SF6_plot[SF6_plot.YEA != '']
SF6_plot = SF6_plot[~pd.to_numeric(SF6_plot['Value'], errors='coerce').isnull()]
SF6_plot.Value = SF6_plot.Value.astype(float)

f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('SF6')

sns.lineplot(
    data=SF6_plot,
    x='YEA',
    y='Value',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



Most of the values where stable for a country the resent years. Because of this we decided, that the most recent value per country per greenhouse gas can be used as the current Value. Additionally we decided can use all features except the HFC_PFC in the next steps. HFC_PFC was discarded, as only 6 Countries had mesasurements for this type. For all other measurements it was decided to replace missing values with -1 to be usable and differentiable in the next steps.

4.2.5 Autofill 2020

[69]:
```
CH4 = CH4.transpose().fillna(method='ffill').transpose()
CO2 = CO2.transpose().fillna(method='ffill').transpose()
HFC = HFC.transpose().fillna(method='ffill').transpose()
HFC_PFC = HFC_PFC.transpose().fillna(method='ffill').transpose()
N2O = N2O.transpose().fillna(method='ffill').transpose()
NF3 = NF3.transpose().fillna(method='ffill').transpose()
PFC = PFC.transpose().fillna(method='ffill').transpose()
SF6 = SF6.transpose().fillna(method='ffill').transpose()

conc = [
    CH4[['Country', '2020']].set_index('Country'),
    CO2[['Country', '2020']].set_index('Country'),
    HFC[['Country', '2020']].set_index('Country'),
    HFC_PFC[['Country', '2020']].set_index('Country'),
    N2O[['Country', '2020']].set_index('Country'),
    NF3[['Country', '2020']].set_index('Country'),
    PFC[['Country', '2020']].set_index('Country'),
    SF6[['Country', '2020']].set_index('Country'),
]
```

```
res = pd.concat(conc, axis=1, join='outer')
res.columns = ['CH4', 'CO2', 'HFC', 'HFC_PFC', 'N2O', 'NF3', 'PFC', 'SF6']
```

4.2.6 Remove unclear columns (Columns with too many missing values)

[70]:
```
res = res.drop('HFC_PFC', axis=1)
res = res.fillna(-1)
```

4.2.7 Save Dataset

[71]:
```
res = res.reset_index(drop=False)
res = res.rename(columns={'index':'Country'})

ds_ghg = res.copy()
```

4.3 Land Cover

Data about land cover of several OECD and non OECD countries (https://stats.oecd.org/Index.aspx?DataSetCode=LAND_COVER#) The data set obtained from the OECD website contains various properties of the land surfaces of several OCED and non OECD countries. As we chose to limit our analysis on mainly land living animals, the characteristics of the surface and it's state could propose a valueable feature for our prediction, because it tells us a lot about the habitats of those animals. Given features are: -Artificial surfaces -Bare area -Inland water -Cropland -Grassland -Shrubland -Sparse vegetation -Tree Cover -Wetland The database contained data from 4 different years, which we later use to approximate our data for 2020.

4.3.1 Loading Data

The shape of the data obtained from the OECD website was not suitable for our analysis. We want to use the relative numbers of the land cover values, so that a difference in the total size of the country is not influentital for our analysis. The table contains also the absolute values, which we do not need for now. The shape of the raw data set can be seen below and we want to only keep the country and the year as an index and the relative values of our attributes for our first analysis.

[72]:
```
LAND_COVER = DATA_PATH / 'OECD' / 'LAND_COVER_DATA.csv'
TOTAL_AREA =  DATA_PATH / 'Worldbank' / 'API_AG.SRF.TOTL.
↪K2_DS2_en_csv_v2_1927208.csv'

land_cover = pd.read_csv(LAND_COVER)
land_cover.shape
```

[72]: (17694, 21)

[73]:
```
land_cover.head()
```

[73]:
```
   COU    Country SMALL_SUBNATIONAL_REGION Small subnational region  \
0  AUS  Australia                    TOTAL                    Total
1  AUS  Australia                    TOTAL                    Total
2  AUS  Australia                    TOTAL                    Total
3  AUS  Australia                    TOTAL                    Total
4  AUS  Australia                    TOTAL                    Total

  LARGE_SUBNATIONAL_REGION Large subnational region           MEAS  \
0                    TOTAL                    Total  THOUSAND_SQKM
1                    TOTAL                    Total  THOUSAND_SQKM
2                    TOTAL                    Total  THOUSAND_SQKM
3                    TOTAL                    Total  THOUSAND_SQKM
4                    TOTAL                    Total  THOUSAND_SQKM

                   Measure VARIABLE Land cover class  …  Year  Unit Code  \
0  Square kilometers (000's)   FOREST       Tree cover  …  1992        NaN
1  Square kilometers (000's)   FOREST       Tree cover  …  2004        NaN
2  Square kilometers (000's)   FOREST       Tree cover  …  2015        NaN
3  Square kilometers (000's)   FOREST       Tree cover  …  2018        NaN
4  Square kilometers (000's)     GRSL        Grassland  …  1992        NaN

   Unit  PowerCode Code  PowerCode Reference Period Code  Reference Period  \
0   NaN               0      Units                   NaN               NaN
1   NaN               0      Units                   NaN               NaN
2   NaN               0      Units                   NaN               NaN
3   NaN               0      Units                   NaN               NaN
4   NaN               0      Units                   NaN               NaN

         Value  Flag Codes  Flags
0   911.890687         NaN    NaN
1   890.559607         NaN    NaN
2   896.524077         NaN    NaN
3   904.706598         NaN    NaN
4  1205.405426         NaN    NaN

[5 rows x 21 columns]
```

[74]:
```
#number of unique values per column
land_cover.nunique()
```

[74]:
```
COU                           246
Country                       246
SMALL_SUBNATIONAL_REGION        1
Small subnational region        1
LARGE_SUBNATIONAL_REGION        1
Large subnational region        1
MEAS                            2
```

```
Measure                        2
VARIABLE                       9
Land cover class               9
YEA                            4
Year                           4
Unit Code                      0
Unit                           0
PowerCode Code                 1
PowerCode                      1
Reference Period Code          0
Reference Period               0
Value                      12556
Flag Codes                     0
Flags                          0
dtype: int64
```

[75]:
```
#for our analysis we use the relative data, to make it comparable across␣
↪countries of different sizes
land_cover_rel = land_cover.copy()
land_cover_rel = land_cover_rel[land_cover_rel['MEAS'] == 'PCNT']
land_cover_rel.shape
```

[75]:
```
(8838, 21)
```

4.4 Cleaning Data

For this step we first remove entries with countries we do not need. Then we drop all columns that aren't relevant and bring the data in the final shape for our feature preparation.

[76]:
```
#we look at the unique countries (246 as shown before) in our new data frame
land_cover_rel.Country.unique()
```

[76]:
```
array(['Australia', 'Belgium', 'Canada', 'Czech Republic', 'Denmark',
       'Finland', 'France', 'Germany', 'Greece', 'Hungary', 'Iceland',
       'Ireland', 'Italy', 'Japan', 'Luxembourg', 'Mexico', 'New Zealand',
       'Norway', 'Poland', 'Portugal', 'Slovak Republic', 'Spain',
       'Sweden', 'Switzerland', 'Turkey', 'United Kingdom',
       'United States', 'Albania', 'Algeria', 'American Samoa', 'Andorra',
       'Angola', 'Argentina', 'Aruba', 'Bahamas', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Benin', 'Bhutan', 'Bolivia',
       'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'British Virgin Islands', 'Bulgaria', 'Burkina Faso', 'Burundi',
       'Cambodia', 'Cabo Verde', 'Cayman Islands', 'Chad',
       "China (People's Republic of)", 'Comoros', 'Congo', 'Cook Islands',
       'Costa Rica', "Côte d'Ivoire", 'Croatia', 'Cuba', 'Cyprus',
       "Democratic People's Republic of Korea",
       'Democratic Republic of the Congo', 'Djibouti',
```

```
        'Dominican Republic', 'El Salvador', 'Equatorial Guinea',
        'Eritrea', 'Estonia', 'Ethiopia', 'Faeroe Islands',
        'Falkland Islands (Malvinas)', 'Fiji', 'French Polynesia', 'Gabon',
        'Gambia', 'Georgia', 'Ghana', 'Gibraltar', 'Greenland', 'Grenada',
        'Guam', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti',
        'Holy See', 'Honduras', 'Hong Kong, China', 'India', 'Indonesia',
        'Iran', 'Iraq', 'Israel', 'Jamaica', 'Jordan', 'Kazakhstan',
        'Kenya', 'Kiribati', 'Kyrgyzstan',
        "Lao People's Democratic Republic", 'Latvia', 'Lebanon', 'Liberia',
        'Lithuania', 'North Macedonia', 'Madagascar', 'Malawi', 'Malaysia',
        'Maldives', 'Mali', 'Malta', 'Mauritania', 'Mauritius',
        'Micronesia', 'Moldova', 'Montserrat', 'Morocco', 'Namibia',
        'Nepal', 'Netherlands Antilles', 'Niger',
        'Northern Mariana Islands', 'Oman', 'Papua New Guinea',
        'Saint Helena', 'Saint Kitts and Nevis', 'Saint Lucia',
        'Saint Pierre and Miquelon', 'Saint Vincent and the Grenadines',
        'Samoa', 'San Marino', 'Sao Tome and Principe', 'Saudi Arabia',
        'Senegal', 'Seychelles', 'Singapore', 'Slovenia',
        'Solomon Islands', 'Somalia', 'South Africa', 'Sri Lanka', 'Sudan',
        'Svalbard and Jan Mayen', 'Eswatini', 'Chinese Taipei', 'Tanzania',
        'Timor-Leste', 'Tokelau', 'Tonga', 'Tunisia', 'Turkmenistan',
        'Tuvalu', 'Uganda', 'Ukraine', 'United Arab Emirates', 'Uruguay',
        'Uzbekistan', 'Vanuatu', 'Venezuela', 'Viet Nam',
        'Wallis and Futuna', 'Yemen', 'Western Sahara', 'Macau, China',
        'Antarctica', 'Heard Island and McDonald Islands',
        'British Indian Ocean Territory', 'Montenegro', 'Guernsey',
        'Jersey',
        'BRIICS economies - Brazil, Russia, India, Indonesia, China and South
Africa',
        'European Union (28 countries)', 'OECD - Europe',
        'OECD Asia Oceania', 'OECD America', 'Latin America and Caribbean',
        'Middle East and North Africa', 'Palau', 'Bouvet Island',
        'Suriname', 'Colombia', 'Azerbaijan', 'Tajikistan', 'Sierra Leone',
        'Mozambique', 'Thailand', 'Chile', 'Kuwait', 'Peru',
        'Antigua and Barbuda', 'United States Virgin Islands',
        'French Southern and Antarctic Lands', 'Paraguay', 'Togo', 'G20',
        'Panama', 'Pakistan', 'Niue', 'Ecuador', 'Mongolia',
        'Trinidad and Tobago', 'Armenia', 'Marshall Islands', 'Qatar',
        'Anguilla', 'Russia', 'Syrian Arab Republic', 'Myanmar', 'Korea',
        'Christmas Islands', 'Puerto Rico', 'Afghanistan',
        'Turks and Caicos Islands', 'Rwanda', 'Libya', 'Lesotho',
        'South Sudan ', 'Pitcairn', 'Romania', 'Cocos (Keeling) Islands',
        'Bermuda', 'Netherlands',
        'Palestinian Authority or West Bank and Gaza Strip', 'Isle of Man',
        'Nauru', 'Liechtenstein', 'Philippines', 'Nigeria',
        'New Caledonia', 'Zimbabwe', 'Brunei Darussalam', 'Nicaragua',
        'Dominica', 'Norfolk Island', 'Zambia', 'OECD - Total', 'Belize',
```

```
       'Austria', 'Cameroon', 'Central African Republic', 'Egypt',
       'South Georgia and the South Sandwich Islands', 'Serbia'],
      dtype=object)
```

[77]:
```
#There are several entries that summarize a number of countries, which we remove
remove = ['OECD - Total','European Union (28 countries)','OECD -␣
 ↪Europe','BRIICS economies - Brazil, Russia, India, Indonesia, China and␣
 ↪South Africa','OECD Asia Oceania','OECD America','Latin America and␣
 ↪Caribbean','Middle East and North Africa','G20']
land_cover_rel = land_cover_rel[~land_cover_rel['Country'].isin(remove)]
land_cover_rel.shape
```

[77]: (8514, 21)

[78]:
```
#select subset of relevant columns Country, Year, Land cover class and Value
land_cover_rel = land_cover_rel[['Country', 'Year', 'Land cover class',␣
 ↪'Value']]
land_cover_rel.head()
```

[78]:

	Country	Year	Land cover class	Value
16	Australia	1992	Artificial surfaces	0.100015
17	Australia	2004	Artificial surfaces	0.136482
18	Australia	2015	Artificial surfaces	0.154930
19	Australia	2018	Artificial surfaces	0.159319
20	Australia	1992	Inland water	0.171305

[79]:
```
#list of all land cover attributes
land_cover_rel['Land cover class'].unique()
```

[79]:
```
array(['Artificial surfaces', 'Inland water', 'Bare area', 'Tree cover',
       'Shrubland', 'Sparse vegetation', 'Cropland', 'Wetland',
       'Grassland'], dtype=object)
```

[80]:
```
#no missing values
land_cover_rel.isna().sum()
land_cover_rel
```

[80]:

	Country	Year	Land cover class	Value
16	Australia	1992	Artificial surfaces	0.100015
17	Australia	2004	Artificial surfaces	0.136482
18	Australia	2015	Artificial surfaces	0.154930
19	Australia	2018	Artificial surfaces	0.159319
20	Australia	1992	Inland water	0.171305
…	…	…	…	…
17653	Sudan	2004	Cropland	20.276832
17658	Kazakhstan	1992	Inland water	6.825496
17659	Kazakhstan	2004	Inland water	6.486520

```
17660  Kazakhstan  2015          Inland water   6.214264
17661  Kazakhstan  2018          Inland water   6.247544

[8514 rows x 4 columns]
```

[81]:
```
#setting year and country as index and our targets as columns
land_cover_rel = land_cover_rel.
 ↪pivot_table(index=['Country','Year'],columns='Land cover class',␣
 ↪values='Value')
land_cover_rel.columns.name = None
land_cover_rel.head()
```

[81]:
```
                  Artificial surfaces  Bare area   Cropland  Grassland  \
Country     Year
Afghanistan 1992             0.052857  39.283534  12.584400  36.040998
            2004             0.081033  39.527342  12.131194  37.041703
            2015             0.125987  39.126500  12.227564  37.305171
            2018             0.144171  39.153980  12.120421  37.161205
Albania     1992             0.607273   1.840586  54.521300   5.444811

                  Inland water  Shrubland  Sparse vegetation  Tree cover  \
Country     Year
Afghanistan 1992      0.405237   3.741778           6.647211    1.234941
            2004      0.105870   3.547692           6.325024    1.229056
            2015      0.099157   3.545375           6.330105    1.229056
            2018      0.099181   3.545002           6.528826    1.236130
Albania     1992      2.193421   3.337453           1.321790   30.452653

                   Wetland
Country     Year
Afghanistan 1992  0.009044
            2004  0.011085
            2015  0.011085
            2018  0.011085
Albania     1992  0.280713
```

[82]:
```
land_cover_rel.shape
```

[82]:
```
(946, 9)
```

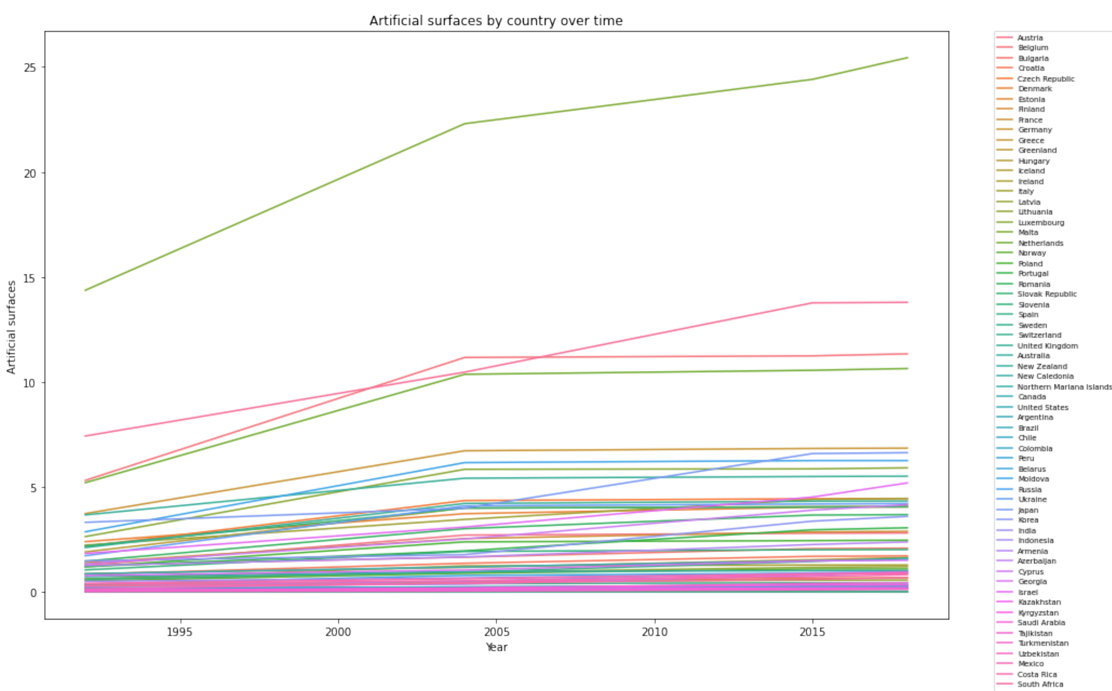
4.5 Feature Preparation

Because we only have records for a few years (1992, 2004, 2015, 2018) and the data we want to predict is from 2020, we investigate the trends of the land cover values over the years. The goal of that is to answer the question, if it is sufficient to take the most recent year and project it's data onto our 2020 prediction goal. That means that we want to see if the surface properties of a country change over time. We do not expect that to be the case, because reshaping the structures of the whole country area is usually (especially if it happens naturally) a process that takes a big amount of time. For a comprehensible visualization we only take the list of countries containing mostly OECD countries created before.

[83]:
```
#for better visualization, we only plot the values for the countries in our␣
↪country list
land_cover_OECD = land_cover_rel.loc[SELECTED_COUNTRIES]
land_cover_OECD.shape
```

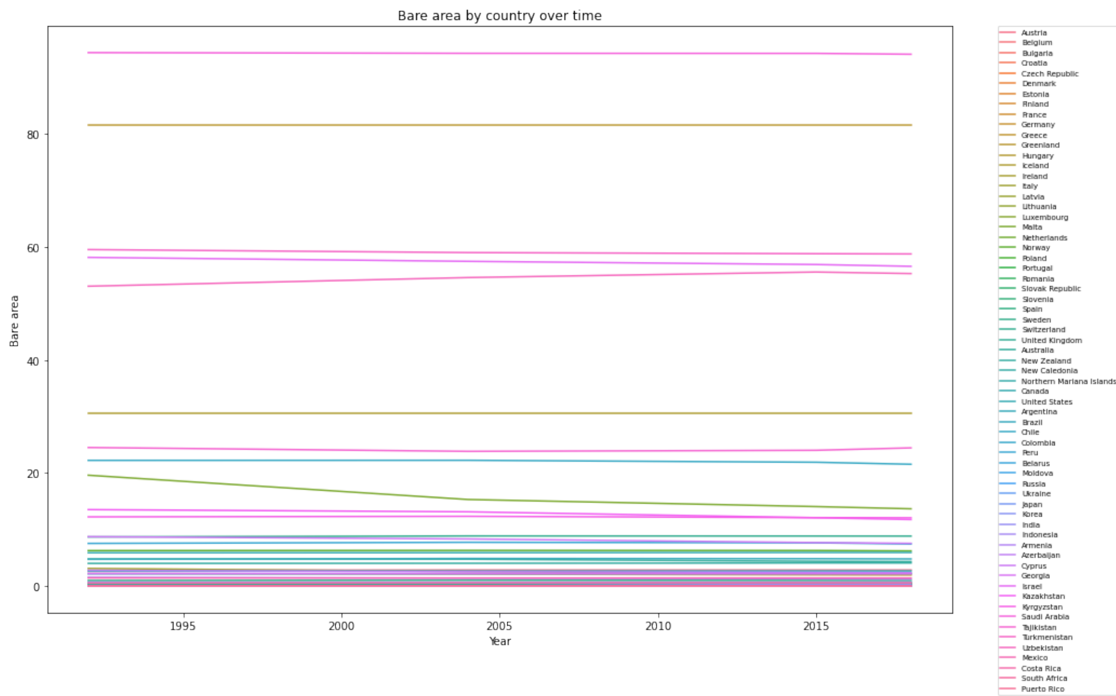
[83]:
```
(260, 9)
```

[84]:
```
#plot the artifical surfaces over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Artificial surfaces by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Artificial surfaces',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```

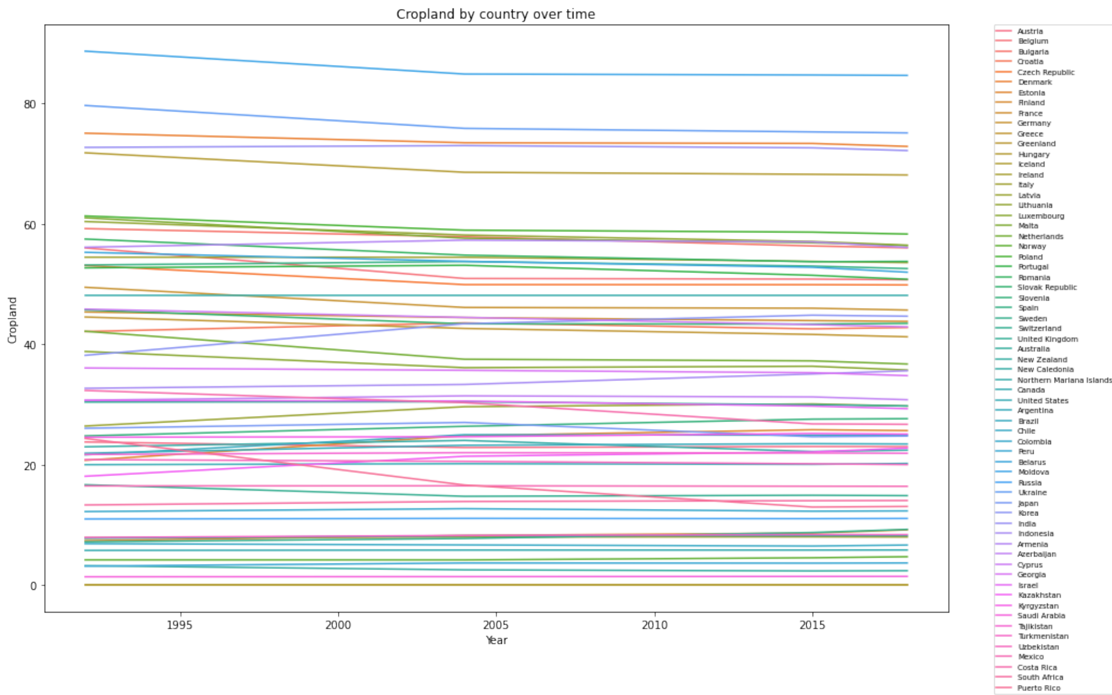


[85]:
```
#plot the bare area over years by country
f, ax = plt.subplots(figsize=(15, 10))
```

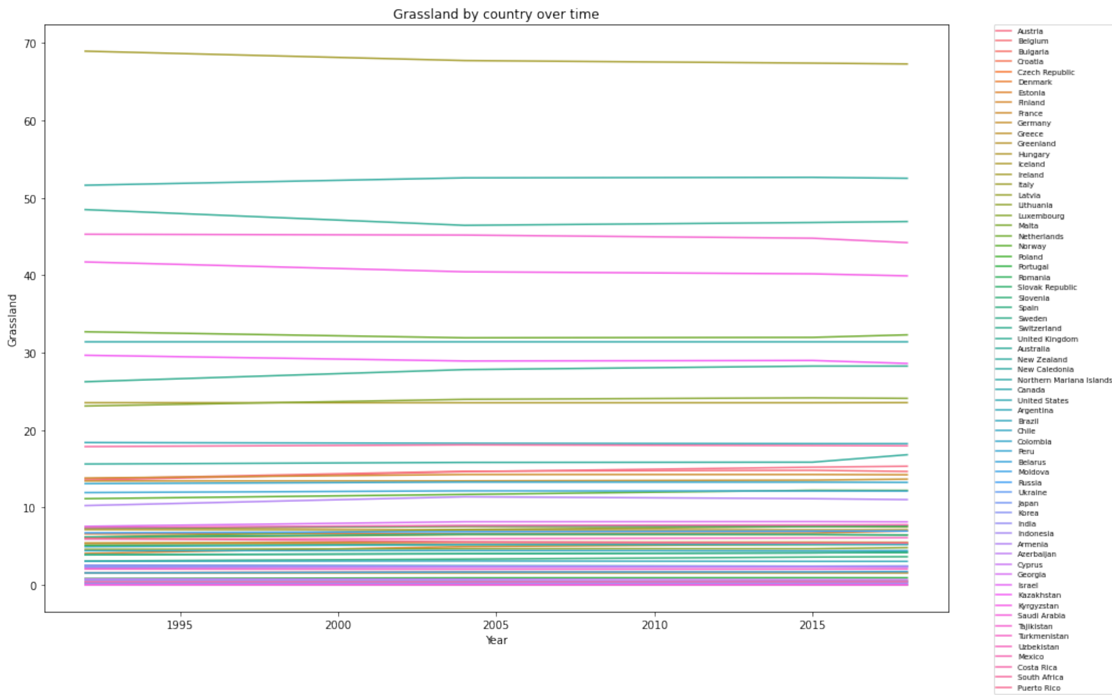
```python
ax.set_title('Bare area by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Bare area',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



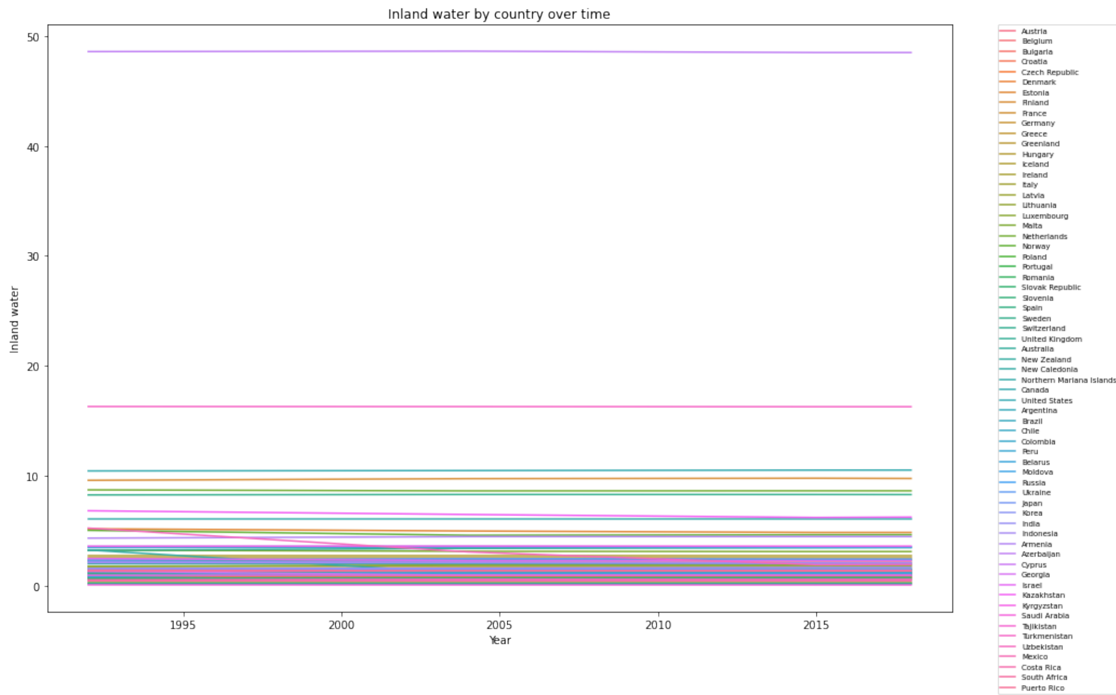
[86]:
```python
#plot the cropland over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Cropland by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Cropland',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



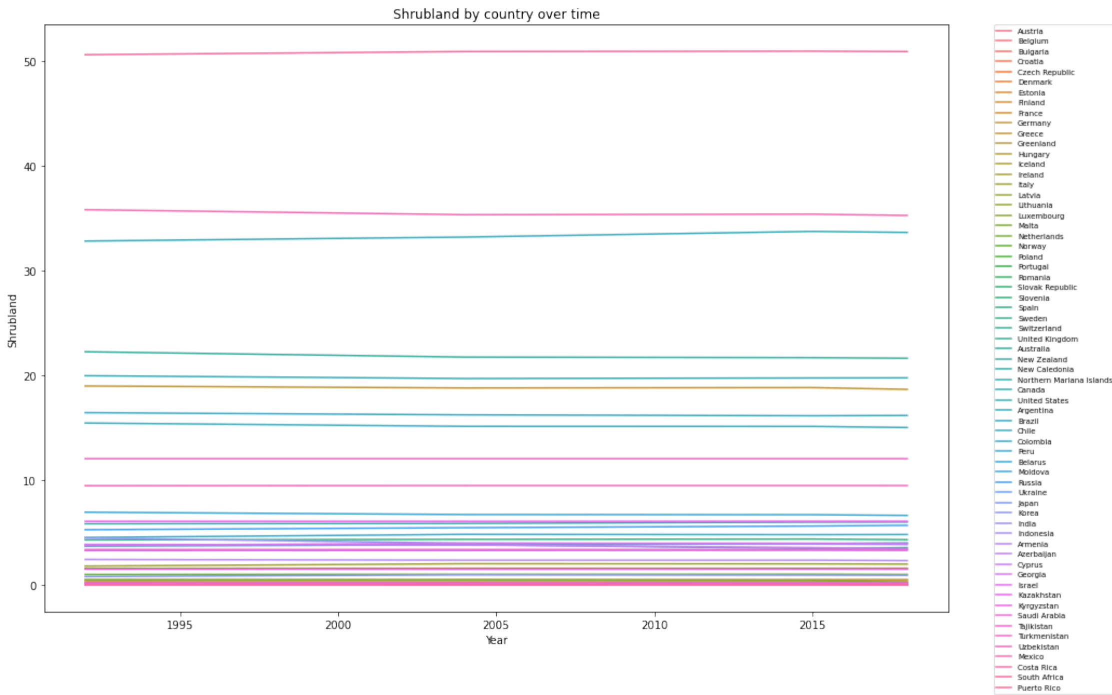
[87]:
```python
#plot the grassland over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Grassland by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Grassland',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



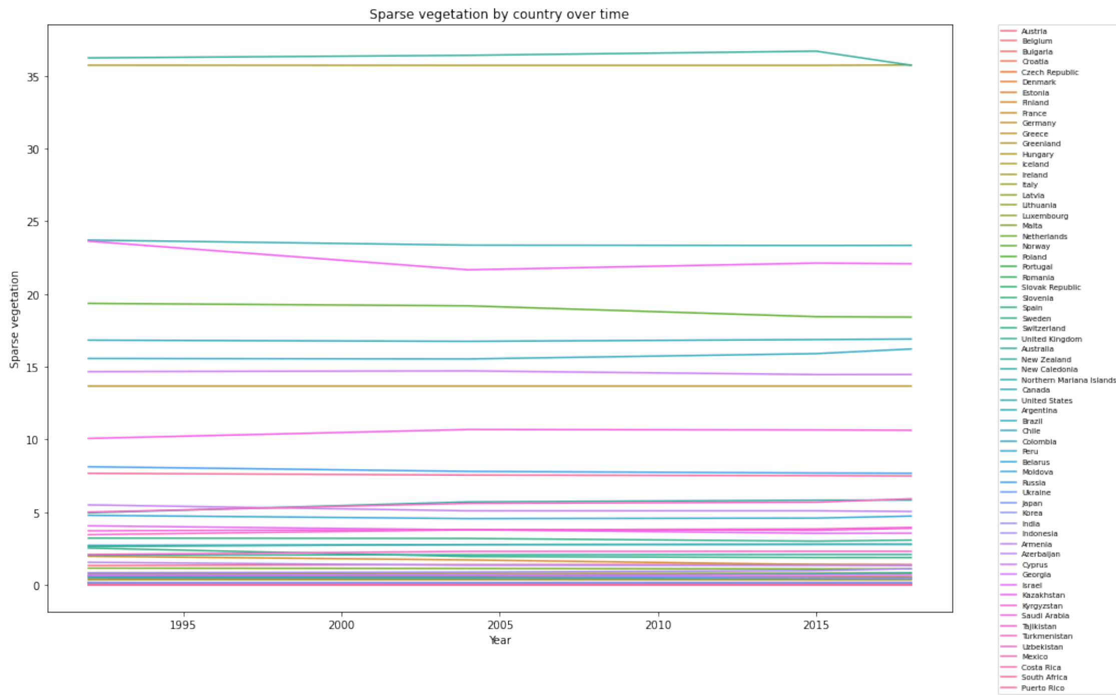
[88]:
```
#plot the inland water over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Inland water by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Inland water',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



[89]:
```python
#plot the shrubland over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Shrubland by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Shrubland',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```

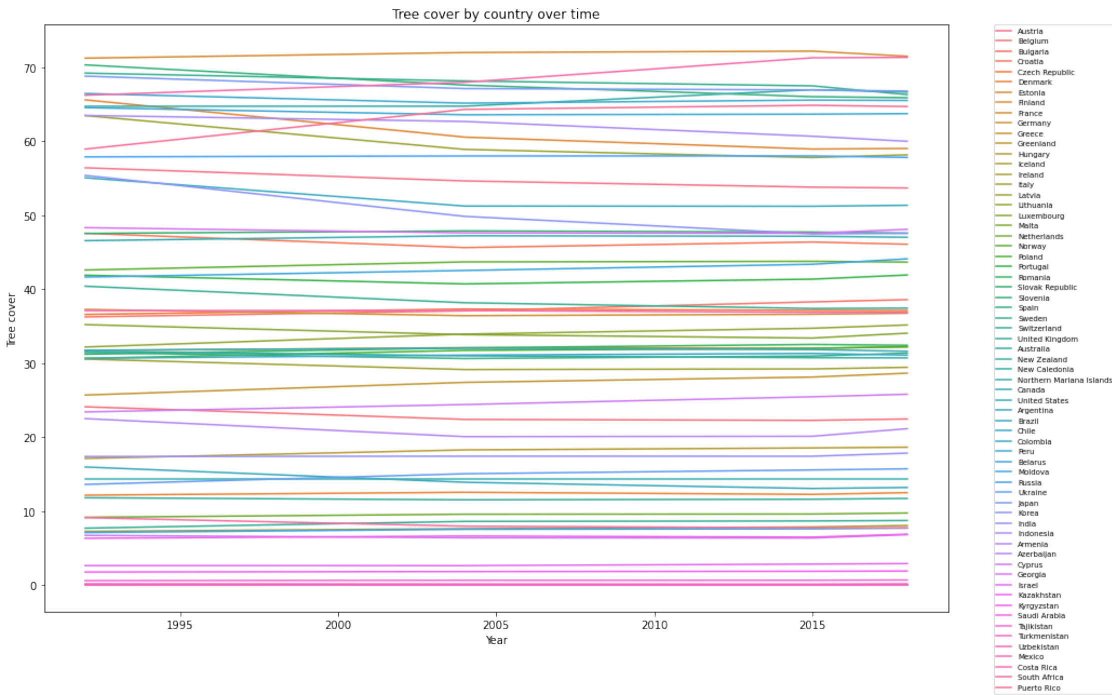


[90]:
```python
#plot the sparse vegetation over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Sparse vegetation by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Sparse vegetation',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```

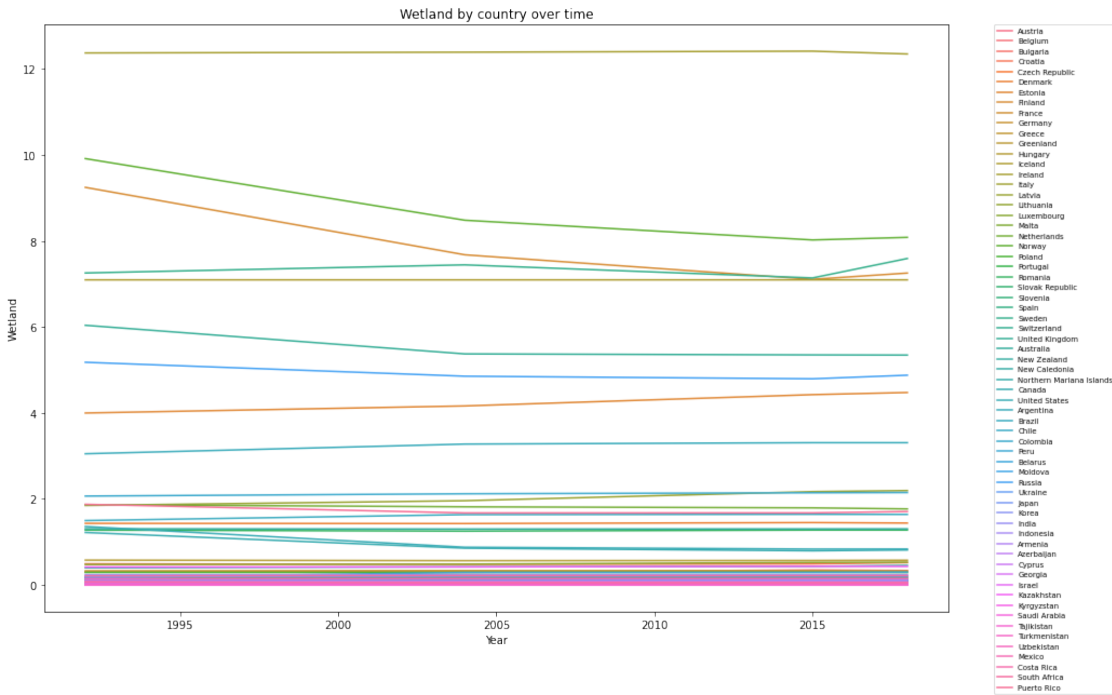


[91]:

```
#plot the tree cover over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Tree cover by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Tree cover',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



[92]:
```
#plot the wetland over years by country
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Wetland by country over time')

sns.lineplot(
    data=land_cover_OECD.reset_index(),
    x='Year',
    y='Wetland',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



4.5.1 Interpreting Results

For the artificial surfaces we can see a major change for our biggest timespan from 1992-2004 in some countries and for a few countries we can see a jump from 2004-2015. However for The shortest timespan from 2015-2018 we can hardly see any markable change across all variables. Excluding artificial surfaces, the data ist quite constant over the whole timespan, as we expected. Especially the ratio 2015/2018 suggests that one can use recent land cover data to approximate the data of the following years, as the properties don't significantly change over a short period of time.

[93]:
```
#extract our 2018 data
land_cover2018 =   land_cover_rel.iloc[land_cover_rel.index.
 ↪get_level_values('Year') == 2018]
land_cover2018.shape
```

[93]:
```
(236, 9)
```

[94]:
```
#descriptive statistics
land_cover2018.describe()
```

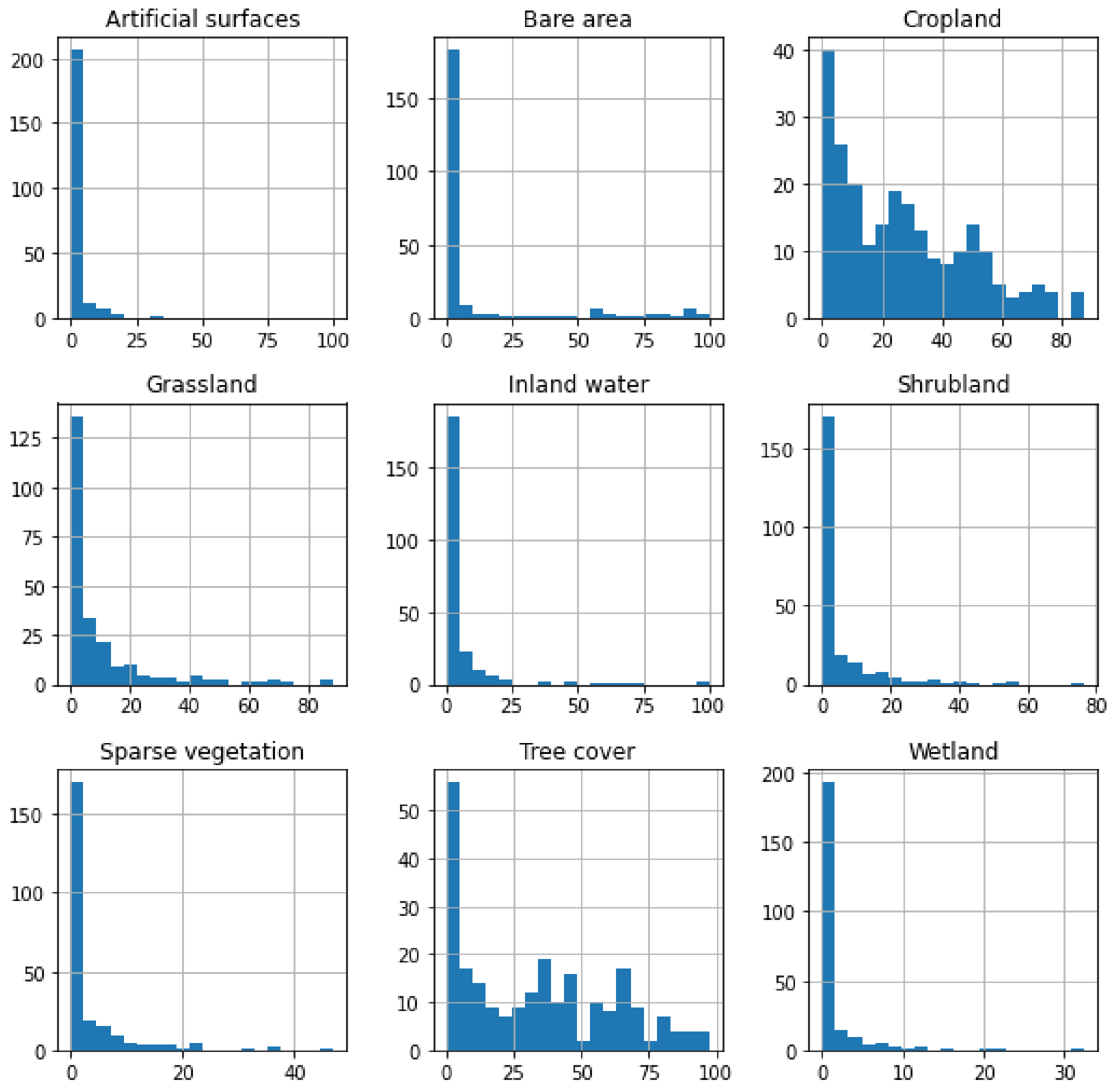
[94]:
```
       Artificial surfaces   Bare area    Cropland   Grassland  Inland water  \
count           236.000000  236.000000  236.000000  236.000000    236.000000
mean              3.274819   12.117159   27.178701    8.833833      5.794420
std               9.807513   26.336930   22.388281   15.280070     13.562647
min               0.000000    0.000000    0.000000    0.000000      0.000000
25%               0.123466    0.000000    7.806774    0.096397      0.799938
```

```
50%               0.569402    0.072513   23.364426    2.765517      1.674404
75%               1.965186    4.094473   43.805507   10.689944      4.474193
max             100.000000   99.871986   87.579618   88.624788    100.000000

        Shrubland  Sparse vegetation  Tree cover     Wetland
count  236.000000         236.000000  236.000000  236.000000
mean     5.278640           3.060210   33.086850    1.375368
std     10.848790           6.588589   28.104437    3.655866
min      0.000000           0.000000    0.000000    0.000000
25%      0.000000           0.008034    5.868930    0.001665
50%      0.376116           0.139986   31.230774    0.118505
75%      5.030730           3.081576   57.213742    0.700044
max     76.679242          46.950629   97.392105   32.498908
```

4.6 Data Exploration

We now only have the entries for 2018 left, which is the data we will use for the rest of the analysis, as we confirmed our expectations that we can assume that there wouldn't be major changes until 2020. However a problem already seen and confirmed in the histograms below (for all countries) is that for a few countries all entries of some values are 0 over all years. This can suggest that the data just isn't available for that country or that the % of given land cover is actually 0. Our first assumption was that most of those zero values are missing/non reported values, because there are so many of them. We then took some steps described below to confirm or discard this assumption.

[95]:
```
fig, axes = plt.subplots(len(land_cover2018.columns)//3, 3, figsize=(10, 10))

i = 0
for triaxis in axes:
    for axis in triaxis:
        land_cover2018.hist(column = land_cover2018.columns[i], bins = 20,␣
 ↪ax=axis)
        i = i+1
```



[96]:
```
land_cover2018.index = land_cover2018.index.droplevel('Year')
land_cover2018.head()
```

[96]:
```
                Artificial surfaces  Bare area   Cropland  Grassland  \
Country
Afghanistan                0.144171  39.153980  12.120421  37.161205
Albania                    1.076736   1.682400  48.785857   5.348827
Algeria                    0.123073  89.988986   4.599071   0.003802
American Samoa             0.038506   0.000000  59.992299   0.000000
Andorra                    0.877637   4.135021   1.012658  23.881857

                Inland water  Shrubland  Sparse vegetation  Tree cover  \
Country
```

```
Afghanistan         0.099181   3.545002            6.528826    1.236130
Albania             2.259645   3.053523            1.328761   36.185951
Algeria             0.064360   0.715815            3.637610    0.866315
American Samoa      4.736234   0.000000            0.077012   35.155949
Andorra             0.000000   0.033755           10.464135   59.594937

                 Wetland
Country
Afghanistan     0.011085
Albania         0.278300
Algeria         0.000967
American Samoa  0.000000
Andorra         0.000000
```

The first test to see if the many 0 values are reasonable, we add a column that sums up all the percentages for each country and check if they rouhgly sum up to 1 (100%).

[97]:
```
land_cover_sum = land_cover2018.copy()
land_cover_sum['Total'] = land_cover_sum.sum(axis=1)
land_cover_sum.head()
```

[97]:
```
                Artificial surfaces  Bare area   Cropland  Grassland  \
Country
Afghanistan                0.144171  39.153980  12.120421  37.161205
Albania                    1.076736   1.682400  48.785857   5.348827
Algeria                    0.123073  89.988986   4.599071   0.003802
American Samoa             0.038506   0.000000  59.992299   0.000000
Andorra                    0.877637   4.135021   1.012658  23.881857

                Inland water  Shrubland  Sparse vegetation  Tree cover  \
Country
Afghanistan         0.099181   3.545002           6.528826    1.236130
Albania             2.259645   3.053523           1.328761   36.185951
Algeria             0.064360   0.715815           3.637610    0.866315
American Samoa      4.736234   0.000000           0.077012   35.155949
Andorra             0.000000   0.033755          10.464135   59.594937

                 Wetland  Total
Country
Afghanistan     0.011085  100.0
Albania         0.278300  100.0
Algeria         0.000967  100.0
American Samoa  0.000000  100.0
Andorra         0.000000  100.0
```

[98]:
```
land_cover_sum.describe()
```

[98]:

	Artificial surfaces	Bare area	Cropland	Grassland	Inland water
count	236.000000	236.000000	236.000000	236.000000	236.000000
mean	3.274819	12.117159	27.178701	8.833833	5.794420
std	9.807513	26.336930	22.388281	15.280070	13.562647
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.123466	0.000000	7.806774	0.096397	0.799938
50%	0.569402	0.072513	23.364426	2.765517	1.674404
75%	1.965186	4.094473	43.805507	10.689944	4.474193
max	100.000000	99.871986	87.579618	88.624788	100.000000

	Shrubland	Sparse vegetation	Tree cover	Wetland	Total
count	236.000000	236.000000	236.000000	236.000000	2.360000e+02
mean	5.278640	3.060210	33.086850	1.375368	1.000000e+02
std	10.848790	6.588589	28.104437	3.655866	4.760214e-13
min	0.000000	0.000000	0.000000	0.000000	1.000000e+02
25%	0.000000	0.008034	5.868930	0.001665	1.000000e+02
50%	0.376116	0.139986	31.230774	0.118505	1.000000e+02
75%	5.030730	3.081576	57.213742	0.700044	1.000000e+02
max	76.679242	46.950629	97.392105	32.498908	1.000000e+02

Our minimum and maximum value for the total column is 100, which means that the land coverage indeed sums up to 100%. But this is still no evidence that those types of land coverage are just not present in those cases. It could still be the case that this table is just derived from the absolute values and those have missing values.

This is why we inspect the absolute values in the next step, sum them up and compare them to the total area of the countries and see if there are major differences. The total area of the countries is from a different source (world bank data), which could lead to slightly different numbers, so we assume that those 0 values are true 0 values even when the total area alues differ a bit in the end.

[99]:
```
#for our analysis we use the relative data, to make it comparable across␣
↪countries of different sizes
land_cover_abs = land_cover.copy()
land_cover_abs = land_cover_abs[land_cover_abs['MEAS'] == 'THOUSAND_SQKM']
land_cover_abs.shape
```

[99]: (8856, 21)

[100]:
```
#getting the data in the same shape as our relative data
land_cover_abs = land_cover_abs[['Country', 'Year', 'Land cover class',␣
↪'Value']]
land_cover_abs = land_cover_abs.
↪pivot_table(index=['Country','Year'],columns='Land cover class',␣
↪values='Value')
land_cover_abs.columns.name = None
land_cover2018_abs =  land_cover_abs.iloc[land_cover_abs.index.
↪get_level_values('Year') == 2018]
land_cover2018_abs.shape
```

[100]: (246, 9)

[101]:
```
#sum up the land coverage
land_cover2018_abs_sum =  land_cover2018_abs.copy()
land_cover2018_abs_sum['Sum'] = land_cover2018_abs_sum.sum(axis=1)
land_cover2018_abs_sum.head()
```

[101]:

Country	Year	Artificial surfaces	Bare area	Cropland	Grassland
Afghanistan	2018	0.927717	251.949704	77.993003	239.126511
Albania	2018	0.310373	0.484957	14.062674	1.541816
Algeria	2018	2.852011	2085.339874	106.575563	0.088104
American Samoa	2018	0.000077	0.000000	0.120408	0.000000
Andorra	2018	0.004019	0.018935	0.004637	0.109357

Country	Year	Inland water	Shrubland	Sparse vegetation	Tree cover
Afghanistan	2018	0.638211	22.811532	42.011969	7.954301
Albania	2018	0.651350	0.880187	0.383020	10.430712
Algeria	2018	1.491427	16.587774	84.295359	20.075369
American Samoa	2018	0.009506	0.000000	0.000155	0.070560
Andorra	2018	0.000000	0.000155	0.047916	0.272890

Country	Year	Wetland	Sum
Afghanistan	2018	0.071333	643.484282
Albania	2018	0.080221	28.825309
Algeria	2018	0.022412	2317.327894
American Samoa	2018	0.000000	0.200707
Andorra	2018	0.000000	0.457908

[102]:
```
#prepare for merging
land_cover_abs_mer = land_cover2018_abs_sum.copy()
land_cover_abs_mer = land_cover_abs_mer.reset_index()
land_cover_abs_mer = land_cover_abs_mer[['Country','Sum']]
```

[103]:
```
#getting data for total country area and only keep the country name and the␣
↪value for 2018 (and dividing them by 1000,
#as this is the unit used for our data)
TOTAL_AREA = pd.read_csv(TOTAL_AREA)
total_area = TOTAL_AREA.copy()
total_area = total_area[['Country Name','2018']]
total_area = total_area.rename(columns={'Country Name': 'Country', '2018':␣
↪'Total'})
total_area['Total']=total_area['Total']/1000
total_area.head()
```

[103]:

	Country	Total
0	Aruba	0.18
1	Afghanistan	652.86
2	Angola	1246.70
3	Albania	28.75
4	Andorra	0.47

[104]:

```
#merge country area data with land cover data, compute differences and show␣
↪descriptive statistics (we reduce the inspected
#countries to the OECD countries again, as those are (almost all) relevant for␣
↪our analysis and it gives a better overview)
area_merged = land_cover_abs_mer.merge(total_area, on='Country', how='inner')
area_merged['Difference'] = area_merged['Total']-area_merged['Sum']
#difference relative to the total area
area_merged['Relative_Difference'] = abs(area_merged['Difference'])/
↪area_merged['Total']
area_merged_OECD = area_merged[area_merged['Country'].isin(SELECTED_COUNTRIES)]
area_merged_OECD.describe()
```

[104]:

	Sum	Total	Difference	Relative_Difference
count	62.000000	62.000000	62.000000	62.000000
mean	1038.169504	1027.662742	-10.506762	0.107145
std	2212.262495	2252.495884	240.247610	0.550100
min	0.314778	0.320000	-1742.914324	0.000160
25%	49.575097	49.547500	-0.455296	0.001853
50%	185.968682	174.490000	0.036203	0.004185
75%	628.771630	589.934248	0.610940	0.012500
max	9807.449189	9984.670000	435.199635	4.246350

We can see a huge outlier with our max value, where the total area is 4 times bigger than our OECD sum. This is the case for greendland, which is not present in our final data. We remove this entry and compute our stats again.

[105]:

```
index = area_merged_OECD[area_merged_OECD['Country'] == 'Greenland'].index
area_merged_OECD.drop(index , inplace=True)
area_merged_OECD.describe()
```

[105]:

	Sum	Total	Difference	Relative_Difference
count	61.000000	61.000000	61.000000	61.000000
mean	1019.887622	1037.780984	17.893362	0.039289
std	2225.894497	2269.767989	88.545393	0.131991
min	0.314778	0.320000	-114.266445	0.000160
25%	48.978658	49.030000	-0.415003	0.001798
50%	164.954119	141.380000	0.051342	0.004162
75%	599.871684	603.550000	0.616755	0.012137
max	9807.449189	9984.670000	435.199635	0.904782

4.6.1 Decision about 0 values

The difference between the sum of land cover data and the total land area from the worldbank data has a mean of 3% (after removing our outlier) for our sample of countries. This leads to solid evidence for the assumption that those 0 values are natural and can be seen as valid features for our future model. The fact that there is a difference will most likely be based on different sources and different measurement criteria for the total area, as well as inaccuracies.

Returning to our relevant data: As we have percentages for every value, we calculate the corresponding decimal value for the future work, which is also a benefit, because we then naturally have our values in a range from 0-1. This is the final shape of our data after exploration.

[106]:
```
land_cover2018 = land_cover2018/100
land_cover2018
```

[106]:
```
                   Artificial surfaces  Bare area  Cropland  Grassland  \
Country
Afghanistan                   0.001442   0.391540  0.121204   0.371612
Albania                       0.010767   0.016824  0.487859   0.053488
Algeria                       0.001231   0.899890  0.045991   0.000038
American Samoa                0.000385   0.000000  0.599923   0.000000
Andorra                       0.008776   0.041350  0.010127   0.238819
…                                  …          …         …         …
Wallis and Futuna             0.003272   0.000000  0.524537   0.000000
Western Sahara                0.000089   0.998720  0.000267   0.000000
Yemen                         0.000927   0.788603  0.047031   0.000966
Zambia                        0.001160   0.000057  0.168463   0.015544
Zimbabwe                      0.001771   0.002299  0.369198   0.037601

                   Inland water  Shrubland  Sparse vegetation  Tree cover  \
Country
Afghanistan            0.000992   0.035450           0.065288    0.012361
Albania                0.022596   0.030535           0.013288    0.361860
Algeria                0.000644   0.007158           0.036376    0.008663
American Samoa         0.047362   0.000000           0.000770    0.351559
Andorra                0.000000   0.000338           0.104641    0.595949
…                           …         …                 …          …
Wallis and Futuna      0.227917   0.000000           0.000000    0.244275
Western Sahara         0.000183   0.000000           0.000741    0.000000
Yemen                  0.002263   0.034993           0.114995    0.010203
Zambia                 0.018611   0.136371           0.000000    0.622014
Zimbabwe               0.011432   0.270810           0.000460    0.305763

                    Wetland
Country
Afghanistan        0.000111
Albania            0.002783
Algeria            0.000010
```

```
American Samoa     0.000000
Andorra            0.000000
…                       …
Wallis and Futuna  0.000000
Western Sahara     0.000000
Yemen              0.000019
Zambia             0.037779
Zimbabwe           0.000668

[236 rows x 9 columns]
```

[107]:
```
ds_land_cover = land_cover2018.reset_index(drop=False).copy()
```

4.7 Protected Areas by Management Objective

This data set was obtained from the OECD repository and answers the questions: how extensive are protected areas and what management objectives are pursued via protected area designation? The numbers are provided in square km but also as relative numbers. We only use relative numbers of terrestrial areas for our analysis.

Because overlaps among protected areas are relatively common, the total protected area for a country is typically less than the sum of the disaggregated areas.

We have to note that not all protected areas have a designation date recorded. When there is no designation date the protected area is deemed to have always existed, therefore historical data maybe be overestimated.

The data was last updated in June 2020 and is therefore reasonable up-to-date.

4.7.1 Load Protected Areas Data

First, we load the raw CSV file.

[108]:
```
# load data
def load_protected_area():
    PROTECTED_AREAS = DATA_PATH / 'OECD' / 'PROTECTED_AREAS_OBJECTIVE.csv'
    data = pd.read_csv(PROTECTED_AREAS)
    return data
protected_area_raw = load_protected_area()
protected_area_raw.shape
```

[108]:
```
(71910, 23)
```

4.7.2 Clean Protected Area Data

The data contains a lot of redundant and useless information. Thus, we only select a subset of needed columns. Further, only units with “Percentage” are selected. These percentages need to be transformed to relative numbers to be consistent over all data sets. Some countries have relative protected areas above 100%. The metadata tells us that this is due to the fact that some countries have some protected areas recorded as points with a reported area. This point data is more uncertain than protected areas reported as polygons because overlaps cannot be identified or resolved. Because only 4 countries are effected by this and all of them are small islands we assume a total protected area of 100% for them.

[109]:
```
# unique values per column
protected_area_raw.nunique()
```

[109]:
```
COU                        127
Country                    127
DESIG                        9
Designation                  9
DOMAIN                       2
Domain                       2
MEASURE                      2
Measure                      2
CALCULATION                  1
Calculation method           1
SCOPE                        1
Scope                        1
YEA                         17
Year                        17
Unit Code                    2
Unit                         2
PowerCode Code               1
PowerCode                    1
Reference Period Code        0
Reference Period             0
Value                    10090
Flag Codes                   0
Flags                        0
dtype: int64
```

[110]:
```
# countries with more than 100 % protected area
protected_area_lt_100 = protected_area_raw[protected_area_raw['Unit'] ==␣
↪'Percentage']
protected_area_lt_100 = protected_area_lt_100[protected_area_lt_100['Value'] >␣
↪100]
protected_area_lt_100['Country'].unique()
```

[110]:
```
array(['New Caledonia', 'Saint Helena', 'Bouvet Island',
       'British Indian Ocean Territory'], dtype=object)
```

[111]:
```
def clean_protected_area(raw_data):
    # filter only Terrestrial protected area
    data = raw_data[raw_data['Domain'] == 'Terrestrial']
```

```
    # filter only percentages
    data = data[data['Unit'] == 'Percentage']

    # select subset of columns needed
    data = data[['Country', 'Year', 'Designation', 'Value']]

    # calculate percentages
    data['Value'] = data['Value'] / 100
    # assume 100% protected area for small islands
    data['Value'] = data['Value'].apply(lambda x: 1 if x > 1 else x)

    return data
protected_area_cleaned = clean_protected_area(protected_area_raw)
protected_area_cleaned.shape
```

[111]: (18972, 4)

[112]:
```
protected_area_cleaned.head()
```

[112]:

	Country	Year	Designation	Value
17	Australia	1970	Ia: Strict Nature Reserve	0.0045
18	Australia	1980	Ia: Strict Nature Reserve	0.0135
19	Australia	1990	Ia: Strict Nature Reserve	0.0166
20	Australia	1995	Ia: Strict Nature Reserve	0.0174
21	Australia	2000	Ia: Strict Nature Reserve	0.0177

[113]:
```
protected_area_cleaned.describe()
```

[113]:

	Year	Value
count	18972.000000	18972.000000
mean	2006.176471	0.039366
std	14.114342	0.118906
min	1970.000000	0.000000
25%	2000.000000	0.000000
50%	2012.000000	0.000500
75%	2016.000000	0.019300
max	2020.000000	1.000000

The data includes information from 1970 up to 2020. For most of the protected areas by management objective we observe a rather small percentage with a mean of 4%. But we have to take into account that there are 8 categories.

4.8 Feature preparation

As we want to know the relative numbers by management objective we transform the data to have one column by management objective. We also rename the objectives to make them more consistent with other column naming. Further, the data is filtered for selected countries and for the year 2020.

[114]:
```
protected_area_cleaned['Designation'].unique()
```

[114]:
```
array(['Ia: Strict Nature Reserve', 'Ib: Wilderness Area',
       'II: National Park', 'III: Natural Monument or Feature',
       'IV: Habitat or Species Management Area',
       'V: Protected Landscape or Seascape',
       'VI: Protected area with sustainable use of natural resources',
       'No IUCN category provided',
       'All, including data recorded as points'], dtype=object)
```

[115]:
```
# check for missing values
protected_area_cleaned.isna().sum()
```

[115]:
```
Country        0
Year           0
Designation    0
Value          0
dtype: int64
```

[116]:
```
def transform_protected_area(cleaned_data, filter_year=None,␣
↪filter_countries=None):

    # create DataFrame with all vealues per country and year
    data = cleaned_data.pivot_table(index=['Country', 'Year'],␣
↪columns='Designation', values='Value')
    data.columns.name = None
    data = data.reset_index()

    # rename columns by management objective
    data = data.rename(columns={
        'Ia: Strict Nature Reserve': 'perc_area_protected_obj_1a',
        'Ib: Wilderness Area': 'perc_area_protected_obj_1b',
        'II: National Park': 'perc_area_protected_obj_2',
        'III: Natural Monument or Feature': 'perc_area_protected_obj_3',
        'IV: Habitat or Species Management Area': 'perc_area_protected_obj_4',
        'V: Protected Landscape or Seascape': 'perc_area_protected_obj_5',
        'VI: Protected area with sustainable use of natural resources':␣
↪'perc_area_protected_obj_6',
        'No IUCN category provided': 'perc_area_protected_no_obj',
        'All, including data recorded as points': 'perc_area_protected_all_obj'
    })

    if filter_year is not None:
        data = data[data['Year'] == filter_year]

    if filter_countries is not None:
       data = data[data['Country'].isin(filter_countries)]
```

```
    return data

# load all data for selected countries and all years
protected_area_all_years = transform_protected_area(
    protected_area_cleaned,
    filter_countries=SELECTED_COUNTRIES)
assert len(SELECTED_COUNTRIES) == protected_area_all_years['Country'].unique().
 ↪shape[0]
protected_area_all_years.shape
```

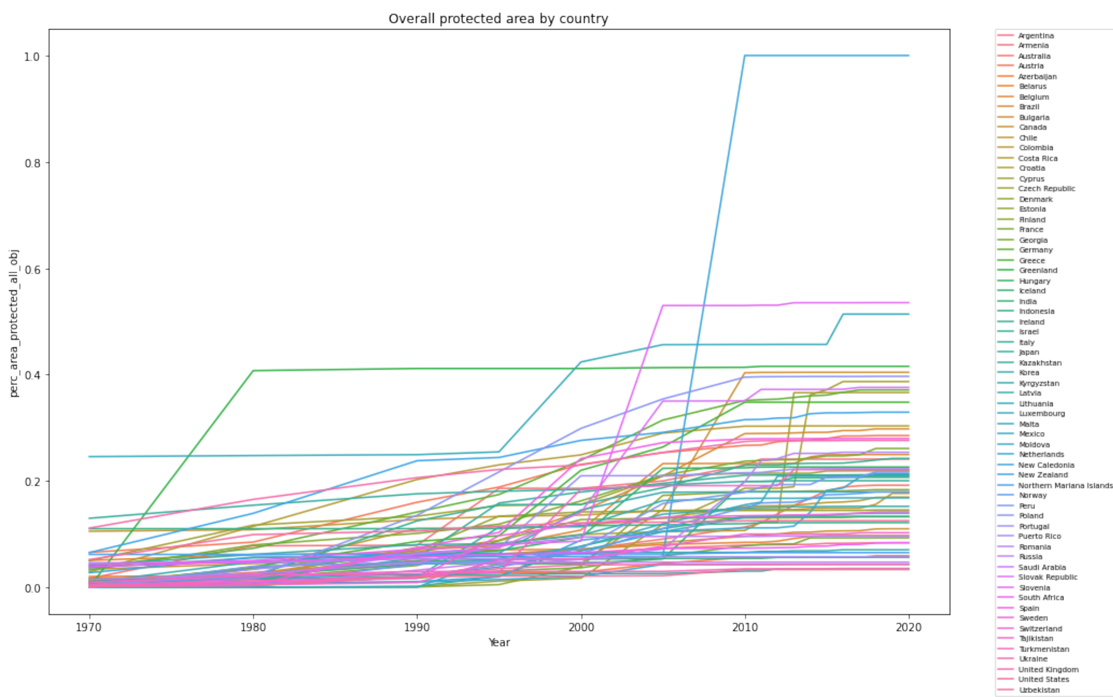
[116]: (1105, 11)

[117]:
```
protected_area_all_years.head()
```

[117]:
```
      Country  Year  perc_area_protected_all_obj  perc_area_protected_obj_2  \
68  Argentina  1970                       0.0104                     0.0066
69  Argentina  1980                       0.0249                     0.0110
70  Argentina  1990                       0.0430                     0.0122
71  Argentina  1995                       0.0521                     0.0136
72  Argentina  2000                       0.0636                     0.0152

    perc_area_protected_obj_3  perc_area_protected_obj_4  \
68                     0.0001                     0.0000
69                     0.0001                     0.0015
70                     0.0001                     0.0015
71                     0.0003                     0.0017
72                     0.0004                     0.0018

    perc_area_protected_obj_1a  perc_area_protected_obj_1b  \
68                      0.0001                         0.0
69                      0.0001                         0.0
70                      0.0025                         0.0
71                      0.0026                         0.0
72                      0.0027                         0.0

    perc_area_protected_no_obj  perc_area_protected_obj_5  \
68                      0.0004                     0.0000
69                      0.0004                     0.0000
70                      0.0007                     0.0000
71                      0.0008                     0.0009
72                      0.0046                     0.0030

    perc_area_protected_obj_6
68                     0.0029
69                     0.0115
70                     0.0251
71                     0.0312
```

```
72                          0.0349
```

[118]:
```python
# plot changes over years in overall protected area
f, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Overall protected area by country')

sns.lineplot(
    data=protected_area_all_years,
    x='Year',
    y='perc_area_protected_all_obj',
    hue="Country")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0., fontsize=7)
plt.show()
```



As we in the above figure most of the countries have a total protected area under 20% and almost all them under 40%. The major increases in protected area happened in the 1990 and early 2000. In the last years almost no country increased their protected areas. Thus, we decided against computing features that help predicting the trend but only use the most recent year for our analysis.

[119]:
```python
# load all data for selected countries and all year 2020
ds_protected_areas = transform_protected_area(
    protected_area_cleaned,
    filter_year=2020,
    filter_countries=SELECTED_COUNTRIES)
```

5 Dataset Merging

Merge IUCN Data with different Support Datasets. Use Country as keys. ## Constants

[120]:
```
RANDOM_STATE = 42
```

5.1 IUCN Data

5.1.1 Load IUCN

[121]:
```
iucn_data = IUCN_cleaned_data.copy()
```

5.2 Country Characteristics Data

5.2.1 Load and Merge

[122]:
```
coun_list = []
country_sets = []

for ds in [ds_protected_areas, ds_land_cover, ds_climate, ds_ghg]:
    coun_list.append(ds)
    country_sets.append(set(ds['Country']))
```

[123]:
```
intersect_countries = country_sets[0].intersection(
    country_sets[1], country_sets[2], country_sets[3])  ## add country_sets[3]
print(
    'The Country Characterisitcs Datasets contain {} intersecting countries,␣
↪which are: {}'
    .format(len(intersect_countries), intersect_countries))
```

```
The Country Characterisitcs Datasets contain 42 intersecting countries, which
are: {'Iceland', 'Greece', 'Spain', 'India', 'Latvia', 'Germany', 'United
States', 'Slovak Republic', 'Ireland', 'Chile', 'Belgium', 'Norway', 'Brazil',
'Estonia', 'Canada', 'Colombia', 'Sweden', 'Japan', 'Portugal', 'Russia',
'Italy', 'Lithuania', 'New Zealand', 'France', 'Argentina', 'Israel',
'Luxembourg', 'Mexico', 'Australia', 'Korea', 'Hungary', 'Denmark', 'Czech
Republic', 'Finland', 'Poland', 'Indonesia', 'Slovenia', 'Costa Rica',
'Switzerland', 'Netherlands', 'United Kingdom', 'Austria'}
```

–> Join Datasets on those countries.

[124]:
```
coun_data = coun_list[0]
for coun in coun_list[1:]:
    coun_data = pd.merge(coun_data, coun, on='Country', how='inner')
coun_data.head()
```

[124]:
```
     Country  Year  perc_area_protected_all_obj  perc_area_protected_obj_2  \
0  Argentina  2020                       0.0839                     0.0184
1  Australia  2020                       0.1920                     0.0417
```

```
2    Austria  2020                         0.2854                     0.0238
3    Belgium  2020                         0.2497                     0.0007
4     Brazil  2020                         0.2980                     0.0413

   perc_area_protected_obj_3  perc_area_protected_obj_4  \
0                     0.0004                     0.0019
1                     0.0024                     0.0027
2                     0.0001                     0.0580
3                     0.0000                     0.0154
4                     0.0007                     0.0003

   perc_area_protected_obj_1a  perc_area_protected_obj_1b  \
0                      0.0027                      0.0003
1                      0.0201                      0.0057
2                      0.0001                      0.0012
3                      0.0000                      0.0000
4                      0.0206                      0.0000

   perc_area_protected_no_obj  perc_area_protected_obj_5  …  temp_slope  \
0                      0.0146                     0.0033  …    0.014949
1                      0.0034                     0.0098  …    0.017804
2                      0.0462                     0.1535  …    0.038525
3                      0.0919                     0.1259  …    0.028670
4                      0.1183                     0.0458  …    0.033281

   gain_percentage  temp_difference       CH4        CO2       HFC       N2O  \
0         0.500525         0.072246  1.828861   4.710241  0.014370  1.002114
1         0.969423         0.214683  4.382540  16.642911  0.479420  0.804808
2        24.241043         1.498389  0.728541   7.549433  0.207606  0.398982
3        12.044550         1.167659  0.688260   8.787278  0.391962  0.500037
4         3.703937         0.929317  1.683181   2.566549 -1.000000  0.894033

        NF3       PFC       SF6
0 -1.000000  0.003744  0.000042
1 -1.000000  0.009443  0.009144
2  0.001868  0.003680  0.043241
3  0.000057  0.011516  0.008337
4 -1.000000 -1.000000 -1.000000

[5 rows x 30 columns]
```

5.2.2 Analysis

General Information

[125]:
```
coun_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 42 entries, 0 to 41
Data columns (total 30 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Country                      42 non-null     object
 1   Year                         42 non-null     int64
 2   perc_area_protected_all_obj  42 non-null     float64
 3   perc_area_protected_obj_2    42 non-null     float64
 4   perc_area_protected_obj_3    42 non-null     float64
 5   perc_area_protected_obj_4    42 non-null     float64
 6   perc_area_protected_obj_1a   42 non-null     float64
 7   perc_area_protected_obj_1b   42 non-null     float64
 8   perc_area_protected_no_obj   42 non-null     float64
 9   perc_area_protected_obj_5    42 non-null     float64
 10  perc_area_protected_obj_6    42 non-null     float64
 11  Artificial surfaces          42 non-null     float64
 12  Bare area                    42 non-null     float64
 13  Cropland                     42 non-null     float64
 14  Grassland                    42 non-null     float64
 15  Inland water                 42 non-null     float64
 16  Shrubland                    42 non-null     float64
 17  Sparse vegetation            42 non-null     float64
 18  Tree cover                   42 non-null     float64
 19  Wetland                      42 non-null     float64
 20  temp_slope                   42 non-null     float64
 21  gain_percentage              42 non-null     float64
 22  temp_difference              42 non-null     float64
 23  CH4                          42 non-null     float64
 24  CO2                          42 non-null     float64
 25  HFC                          42 non-null     float64
 26  N2O                          42 non-null     float64
 27  NF3                          42 non-null     float64
 28  PFC                          42 non-null     float64
 29  SF6                          42 non-null     float64
dtypes: float64(28), int64(1), object(1)
memory usage: 10.2+ KB
```

There are no missing values in the dataset.

[126]:
```
coun_data.describe()
```

[126]:
```
         Year  perc_area_protected_all_obj  perc_area_protected_obj_2  \
count    42.0                    42.000000                  42.000000
mean   2020.0                     0.222829                   0.041517
std       0.0                     0.107664                   0.059849
min    2020.0                     0.056400                   0.000000
25%    2020.0                     0.142875                   0.006525
50%    2020.0                     0.203700                   0.020250
```

```
75%    2020.0                    0.278925                  0.048125
max    2020.0                    0.535300                  0.326100

       perc_area_protected_obj_3  perc_area_protected_obj_4  \
count                  42.000000                  42.000000
mean                    0.003624                   0.029648
std                     0.019175                   0.034700
min                     0.000000                   0.000100
25%                     0.000000                   0.004400
50%                     0.000200                   0.015550
75%                     0.000700                   0.038775
max                     0.124700                   0.147400

       perc_area_protected_obj_1a  perc_area_protected_obj_1b  \
count                   42.000000                   42.000000
mean                     0.004457                    0.008726
std                      0.006766                    0.018277
min                      0.000000                    0.000000
25%                      0.000000                    0.000000
50%                      0.000700                    0.000050
75%                      0.005450                    0.005250
max                      0.021800                    0.075300

       perc_area_protected_no_obj  perc_area_protected_obj_5  \
count                   42.000000                  42.000000
mean                     0.069005                   0.047824
std                      0.087672                   0.059991
min                      0.000000                   0.000000
25%                      0.004725                   0.001100
50%                      0.033750                   0.014300
75%                      0.112900                   0.077800
max                      0.403000                   0.252400

       perc_area_protected_obj_6  …  temp_slope  gain_percentage  \
count                  42.000000  …   42.000000        42.000000
mean                    0.015333  …    0.032865        10.864795
std                     0.027411  …    0.013226        12.305045
min                     0.000000  …    0.006156       -24.178355
25%                     0.000000  …    0.021700         3.832235
50%                     0.000850  …    0.034182         9.547614
75%                     0.015525  …    0.038667        17.897105
max                     0.106300  …    0.058883        43.855657

       temp_difference        CH4        CO2        HFC        N2O        NF3  \
count        42.000000  42.000000  42.000000  42.000000  42.000000  42.000000
mean          0.987359   1.284412   7.478007   0.164214   0.586226  -0.666490
std           0.448411   1.194647   4.017188   0.297642   0.349903   0.477372
```

```
min           0.072246   0.236113   1.278950  -1.000000   0.092906  -1.000000
25%           0.659961   0.750174   4.848182   0.109285   0.345235  -1.000000
50%           0.943211   0.893499   6.846338   0.176891   0.481730  -1.000000
75%           1.382112   1.261372   9.078078   0.322812   0.849686   0.000009
max           1.852533   7.017958  16.642911   0.550778   1.554079   0.002234

             PFC        SF6
count  42.000000  42.000000
mean   -0.154093  -0.036179
std     0.384327   0.219119
min    -1.000000  -1.000000
25%     0.000021   0.002429
50%     0.003712   0.007283
75%     0.012341   0.010464
max     0.216726   0.128989

[8 rows x 29 columns]
```

[127]:

```
coun_data['Country'].nunique()
```

[127]: 42

5.3 Full Data

5.3.1 Merge both datasets

Check Keys (countries)

[128]:

```
iucn_countries = set(iucn_data['Country'].unique())
char_countries = set(coun_data['Country'].unique())
print('IUCN \ CHAR: {}'.format(iucn_countries.difference(char_countries)))
print('CHAR \ IUCN: {}'.format(char_countries.difference(iucn_countries)))
```

```
IUCN \ CHAR: {'Georgia', 'Malta', 'Saudi Arabia', 'Turkmenistan', 'Kyrgyzstan',
'Moldova', 'Armenia', 'Peru', 'Belarus', 'Cyprus', 'Uzbekistan', 'Ukraine',
'Kazakhstan', 'New Caledonia', 'South Africa', 'Azerbaijan', 'Tajikistan',
'Greenland', 'Puerto Rico', 'Bulgaria', 'Northern Mariana Islands', 'Croatia',
'Romania'}
CHAR \ IUCN: set()
```

[129]:

```
oecd_list = [
    'AUSTRALIA', 'AUSTRIA', 'BELGIUM', 'CANADA', 'CHILE', 'COLOMBIA',
    'CZECH REPUBLIC', 'DENMARK', 'ESTONIA', 'FINLAND', 'FRANCE', 'GERMANY',
    'GREECE', 'HUNGARY', 'ICELAND', 'IRELAND', 'ISRAEL', 'ITALY', 'JAPAN',
    'KOREA', 'LATVIA', 'LITHUANIA', 'LUXEMBOURG', 'MEXICO', 'NETHERLANDS',
    'NEW ZEALAND', 'NORWAY', 'POLAND', 'PORTUGAL', 'SLOVAK REPUBLIC',
    'SLOVENIA', 'SPAIN', 'SWEDEN', 'SWITZERLAND', 'TURKEY', 'UNITED KINGDOM',
    'UNITED STATES'
```

```
]
oecd_list = [c.title() for c in oecd_list]
len(oecd_list)
```

[129]: 37

[130]:
```
set(oecd_list).difference(iucn_countries)
```

[130]: {'Turkey'}

[131]:
```
set(oecd_list).difference(char_countries)
```

[131]: {'Turkey'}

[132]:
```
iucn_diff_list = [c for c in iucn_countries.difference(char_countries)]
set(iucn_diff_list).intersection(set(oecd_list))
```

[132]: set()

Rename Country Data to Match IUCN Data

[133]:
```
coun_data.loc[coun_data['Country'] == 'New Zealand', 'Country'] = 'New_Zealand'
coun_data.loc[coun_data['Country'] == 'Slovak Republic', 'Country'] = 'Slovakia'
coun_data.loc[coun_data['Country'] == 'United Kingdom', 'Country'] =␣
 ↪'United_Kingdom'
coun_data.loc[coun_data['Country'] == 'United States', 'Country'] =␣
 ↪'United_States'
coun_data.loc[coun_data['Country'] == 'Czech Republic', 'Country'] = 'Czechia'
coun_data.loc[coun_data['Country'] == 'Korea', 'Country'] = 'Korea,_Republic_of'
```

Merge

[134]:
```
iucn_data = iucn_data.rename(columns={'country':'Country'})
full_data = iucn_data.merge(coun_data, on='Country', how='inner')
full_data['Country'].nunique()
```

[134]: 36

[135]:
```
full_data.columns.values
```

[135]:
```
array(['group', 'scientific_name', 'trend', 'threat_level', 'Country',
       'Year', 'perc_area_protected_all_obj', 'perc_area_protected_obj_2',
       'perc_area_protected_obj_3', 'perc_area_protected_obj_4',
       'perc_area_protected_obj_1a', 'perc_area_protected_obj_1b',
       'perc_area_protected_no_obj', 'perc_area_protected_obj_5',
       'perc_area_protected_obj_6', 'Artificial surfaces', 'Bare area',
       'Cropland', 'Grassland', 'Inland water', 'Shrubland',
```

```
       'Sparse vegetation', 'Tree cover', 'Wetland', 'temp_slope',
       'gain_percentage', 'temp_difference', 'CH4', 'CO2', 'HFC', 'N2O',
       'NF3', 'PFC', 'SF6'], dtype=object)
```

Merge with the relative number of threatened species For the prediction of the number of species per country relative to the number of total described species, we merge our coun_data with the relative number of threatened species per country. After those steps we have got 36 countries left, with 29 different characteristics for each one. We also have the relative number of threatened species by the taxonomic group for the major land living groups. Those are mammals, insects, amphibians, birds and reptiles.

[136]:
```
threatened_relative = ds_threatened_by_group.copy()
threatened_relative = threatened_relative.rename(columns={'country': 'Country'})
```

[137]:
```
threatened_relative
```

[137]:
```
           Country  total_threatened  reptiles_threatened  mammals_threatened  \
0        Argentina            0.0791               0.0771              0.1003
1          Armenia            0.0682               0.1628              0.0761
2        Australia            0.1120               0.0766              0.1864
3          Austria            0.0628               0.0769              0.0568
4       Azerbaijan            0.0648               0.1800              0.0577
..             …                 …                    …                   …
60    Turkmenistan            0.0686               0.1111              0.0947
61         Ukraine            0.0846               0.0625              0.1150
62  United Kingdom            0.0509               0.1429              0.0533
63   United States            0.1292               0.1231              0.0948
64      Uzbekistan            0.0704               0.1842              0.1111

    amphibians_threatened  insects_threatened  birds_threatened  \
0                  0.2061              0.0744            0.0519
1                  0.0000              0.0959            0.0471
2                  0.2108              0.1403            0.0716
3                  0.0000              0.0867            0.0426
4                  0.0909              0.0685            0.0489
..                    …                   …                 …
60                 0.0000              0.0923            0.0519
61                 0.0000              0.1189            0.0535
62                 0.0000              0.0737            0.0412
63                 0.2044              0.1540            0.1044
64                 0.0000              0.0351            0.0538

    reptiles_resident  mammals_resident  amphibians_resident  \
0                True              True                 True
1                True              True                 True
2                True              True                 True
3                True              True                 True
```

```
4                 True              True                    True
..                 …               …                    …
60                True              True                    True
61                True              True                    True
62                True              True                    True
63                True              True                    True
64                True              True                    True

    insects_resident  birds_resident
0               True            True
1               True            True
2               True            True
3               True            True
4               True            True
..               …             …
60              True            True
61              True            True
62              True            True
63              True            True
64              True            True

[65 rows x 12 columns]
```

[138]:
```
threatened_countries = set(threatened_relative['Country'].unique())
print('THREAT \ CHAR: {}'.format(threatened_countries.
 ↪difference(char_countries)))
print('CHAR \ THREAT: {}'.format(char_countries.
 ↪difference(threatened_countries)))
```

```
THREAT \ CHAR: {'Georgia', 'Malta', 'Saudi Arabia', 'Turkmenistan',
'Kyrgyzstan', 'Moldova', 'Armenia', 'Peru', 'Belarus', 'Cyprus', 'Uzbekistan',
'Ukraine', 'Kazakhstan', 'New Caledonia', 'South Africa', 'Azerbaijan',
'Tajikistan', 'Greenland', 'Puerto Rico', 'Bulgaria', 'Northern Mariana
Islands', 'Croatia', 'Romania'}
CHAR \ THREAT: set()
```

[139]:
```
full_threatened = threatened_relative.merge(coun_data, on='Country',␣
 ↪how='inner')
full_threatened.head()
```

[139]:
```
     Country  total_threatened  reptiles_threatened  mammals_threatened  \
0  Argentina            0.0791               0.0771              0.1003
1  Australia            0.1120               0.0766              0.1864
2    Austria            0.0628               0.0769              0.0568
3    Belgium            0.0378               0.0000              0.0417
4     Brazil            0.0861               0.0766              0.1360
```

```
   amphibians_threatened  insects_threatened  birds_threatened  \
0                 0.2061              0.0744            0.0519
1                 0.2108              0.1403            0.0716
2                 0.0000              0.0867            0.0426
3                 0.0000              0.0544            0.0303
4                 0.0430              0.0807            0.0914

   reptiles_resident  mammals_resident  amphibians_resident  …  temp_slope  \
0               True              True                 True  …    0.014949
1               True              True                 True  …    0.017804
2               True              True                 True  …    0.038525
3               True              True                 True  …    0.028670
4               True              True                 True  …    0.033281

   gain_percentage  temp_difference       CH4        CO2       HFC       N2O  \
0         0.500525         0.072246  1.828861   4.710241  0.014370  1.002114
1         0.969423         0.214683  4.382540  16.642911  0.479420  0.804808
2        24.241043         1.498389  0.728541   7.549433  0.207606  0.398982
3        12.044550         1.167659  0.688260   8.787278  0.391962  0.500037
4         3.703937         0.929317  1.683181   2.566549 -1.000000  0.894033

        NF3       PFC       SF6
0 -1.000000  0.003744  0.000042
1 -1.000000  0.009443  0.009144
2  0.001868  0.003680  0.043241
3  0.000057  0.011516  0.008337
4 -1.000000 -1.000000 -1.000000

[5 rows x 41 columns]
```

[140]:
```
#descriptive satistics for our final data frame
full_threatened.describe()
```

[140]:
```
       total_threatened  reptiles_threatened  mammals_threatened  \
count         36.000000            36.000000           36.000000
mean           0.084056             0.095097            0.101114
std            0.043261             0.111864            0.070868
min            0.016700             0.000000            0.000000
25%            0.047375             0.000000            0.050100
50%            0.071400             0.076750            0.089700
75%            0.121650             0.153800            0.138025
max            0.166500             0.500000            0.294000

       amphibians_threatened  insects_threatened  birds_threatened    Year  \
count              36.000000           36.000000         36.000000    36.0
mean                0.114533            0.087264          0.051697  2020.0
std                 0.156417            0.051773          0.021114     0.0
```

```
min                   0.000000            0.000000           0.016000  2020.0
25%                   0.000000            0.060475           0.037575  2020.0
50%                   0.032150            0.083700           0.044300  2020.0
75%                   0.207275            0.100175           0.062050  2020.0
max                   0.563000            0.234400           0.111900  2020.0

       perc_area_protected_all_obj  perc_area_protected_obj_2  \
count                    36.000000                  36.000000
mean                      0.218903                   0.042853
std                       0.109881                   0.063160
min                       0.056400                   0.000000
25%                       0.141350                   0.006425
50%                       0.196250                   0.020250
75%                       0.265775                   0.049350
max                       0.535300                   0.326100

       perc_area_protected_obj_3  …  temp_slope  gain_percentage  \
count                  36.000000  …   36.000000        36.000000
mean                    0.000589  …    0.032756        10.010823
std                     0.001175  …    0.012915        12.942997
min                     0.000000  …    0.006156       -24.178355
25%                     0.000000  …    0.022128         3.639039
50%                     0.000150  …    0.034182         8.858960
75%                     0.000500  …    0.038572        15.000800
max                     0.005900  …    0.058883        43.855657

       temp_difference        CH4        CO2        HFC        N2O        NF3  \
count        36.000000  36.000000  36.000000  36.000000  36.000000  36.000000
mean          0.938828   1.156474   7.097104   0.144013   0.561717  -0.694300
std           0.446922   0.820514   3.919144   0.313502   0.306269   0.467397
min           0.072246   0.236113   1.278950  -1.000000   0.092906  -1.000000
25%           0.603596   0.740512   4.616156   0.103406   0.357740  -1.000000
50%           0.903967   0.893499   6.477036   0.158652   0.481730  -1.000000
75%           1.220734   1.195826   8.839010   0.279662   0.819767   0.000005
max           1.852533   4.382540  16.642911   0.550778   1.431682   0.002234

                PFC        SF6
count  3.600000e+01  36.000000
mean  -1.818770e-01  -0.046840
std    4.091898e-01   0.234670
min   -1.000000e+00  -1.000000
25%    9.067425e-07   0.002302
50%    3.587118e-03   0.007283
75%    1.057792e-02   0.010420
max    2.167259e-01   0.046683

[8 rows x 35 columns]
```

5.4 Trends

5.4.1 Preprocessing

Transform Trends to [-1, 1]

[141]:
```
full_data = full_data.dropna()
full_data = full_data[full_data['trend'] != 'Unknown']
full_data.loc[full_data['trend'] == 'Decreasing', 'trend_num'] = -1
full_data.loc[full_data['trend'] == 'Stable', 'trend_num'] = 0
full_data.loc[full_data['trend'] == 'Increasing', 'trend_num'] = 1
full_data.head()
```

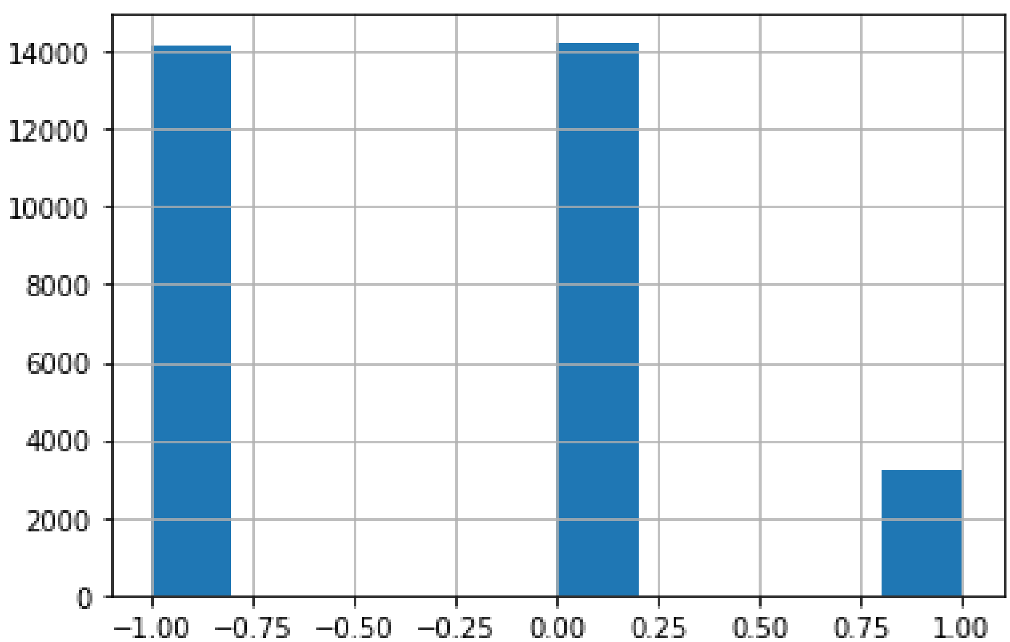
[141]:
```
        group           scientific_name       trend threat_level Country  Year  \
0    reptiles  Goniurosaurus splendens  Decreasing           EN   Japan  2020
2    reptiles    Hemidactylus frenatus      Stable           LC   Japan  2020
3  amphibians           Odorrana narina  Decreasing           EN   Japan  2020
4  amphibians       Hynobius nebulosus  Decreasing           LC   Japan  2020
5  amphibians         Cynops ensicauda  Decreasing           EN   Japan  2020

   perc_area_protected_all_obj  perc_area_protected_obj_2  \
0                       0.2005                     0.0327
2                       0.2005                     0.0327
3                       0.2005                     0.0327
4                       0.2005                     0.0327
5                       0.2005                     0.0327

   perc_area_protected_obj_3  perc_area_protected_obj_4  …  gain_percentage  \
0                     0.0001                     0.0663  …         5.224242
2                     0.0001                     0.0663  …         5.224242
3                     0.0001                     0.0663  …         5.224242
4                     0.0001                     0.0663  …         5.224242
5                     0.0001                     0.0663  …         5.224242

   temp_difference       CH4       CO2       HFC       N2O       NF3  \
0         0.589587  0.236113  8.981805  0.371611  0.158174  0.002234
2         0.589587  0.236113  8.981805  0.371611  0.158174  0.002234
3         0.589587  0.236113  8.981805  0.371611  0.158174  0.002234
4         0.589587  0.236113  8.981805  0.371611  0.158174  0.002234
5         0.589587  0.236113  8.981805  0.371611  0.158174  0.002234

        PFC       SF6  trend_num
0  0.027576  0.016157       -1.0
2  0.027576  0.016157        0.0
3  0.027576  0.016157       -1.0
4  0.027576  0.016157       -1.0
5  0.027576  0.016157       -1.0

[5 rows x 35 columns]
```

[142]:
```
def z_score_normalize(full_data):
    num_cols = []
    for col in full_data.columns.values:
        if full_data[col].dtype == float:
            num_cols.append(col)
    df_zscore = pd.DataFrame(zscore(full_data[num_cols], axis=1) ,␣
↪columns=num_cols)
    return df_zscore
```

5.4.2 Analysis

[143]:
```
full_data['trend_num'].hist()
```

[143]: <AxesSubplot:>



5.4.3 Correlations

Logistic Regression on Trend per Feature See: https://medium.com/@outside2SDs/an-overview-of-correlation-measures-between-categorical-and-continuous-variables-4c7f85610365#:~:text=A%20simple%20approach%20could%20be,variance%20of%20the%20continuous%20variabl

[144]:
```
full_data = full_data.dropna()
full_data = full_data[full_data['trend'] != 'Unknown']
```

[145]:

```
clf = LogisticRegression(random_state=RANDOM_STATE)
runs = []
for col in full_data.columns.values:
    if (full_data[col].dtype == 'int64') | (full_data[col].dtype == 'float64'):
        y = full_data['trend'].copy()
        #y[y.isna()] = 'NULL'
        X = full_data[col]
        X_train, X_test, y_train, y_test = train_test_split(X,
                                                            y,
                                                            test_size=0.33, stratify=y,
                                                            random_state=RANDOM_STATE,␣
 ↪shuffle=True)
        X_train = np.array(X_train).reshape(-1,1)
        X_test =  np.array(X_test).reshape(-1,1)
        y_train = np.array(y_train)
        y_test = np.array(y_test)
        clf.fit(X_train, y_train)

        y_hat = clf.predict(X_test)
        acc = clf.score(X_test, y_test)

        entry = {'column': col, 'acc': acc, 'y': y_test, 'y_hat':y_hat}
        runs.append(entry)
        print('{} predictor accuracy: {}'.format(col, acc))
    # acc_per_col[col] =
```

```
Year predictor accuracy: 0.4506137322593019
perc_area_protected_all_obj predictor accuracy: 0.47103950901419256
perc_area_protected_obj_2 predictor accuracy: 0.4512850019179133
perc_area_protected_obj_3 predictor accuracy: 0.4506137322593019
perc_area_protected_obj_4 predictor accuracy: 0.45349060222477944
perc_area_protected_obj_1a predictor accuracy: 0.4506137322593019
perc_area_protected_obj_1b predictor accuracy: 0.47803989259685464
perc_area_protected_no_obj predictor accuracy: 0.4739163789796701
perc_area_protected_obj_5 predictor accuracy: 0.47056003068661295
perc_area_protected_obj_6 predictor accuracy: 0.4750671269658611
Artificial surfaces predictor accuracy: 0.4672036823935558
Bare area predictor accuracy: 0.4842731108553893
Cropland predictor accuracy: 0.4785193709244342
Grassland predictor accuracy: 0.45454545454545453
Inland water predictor accuracy: 0.43862677406981204
Shrubland predictor accuracy: 0.4766014576141158
Sparse vegetation predictor accuracy: 0.48647871116225544
Tree cover predictor accuracy: 0.4645186037591101
Wetland predictor accuracy: 0.4420790180283851
temp_slope predictor accuracy: 0.4595320291522823
gain_percentage predictor accuracy: 0.4639432297660146
```

```
temp_difference predictor accuracy: 0.46183352512466436
CH4 predictor accuracy: 0.47583429228998847
CO2 predictor accuracy: 0.4619294207901803
HFC predictor accuracy: 0.468162639048715
N2O predictor accuracy: 0.47449175297276563
NF3 predictor accuracy: 0.446394322976601 5
PFC predictor accuracy: 0.47257383966244726
SF6 predictor accuracy: 0.468162639048715
trend_num predictor accuracy: 1.0
```

[146]:
```python
run_df = pd.DataFrame(runs)
run_df.describe()
```

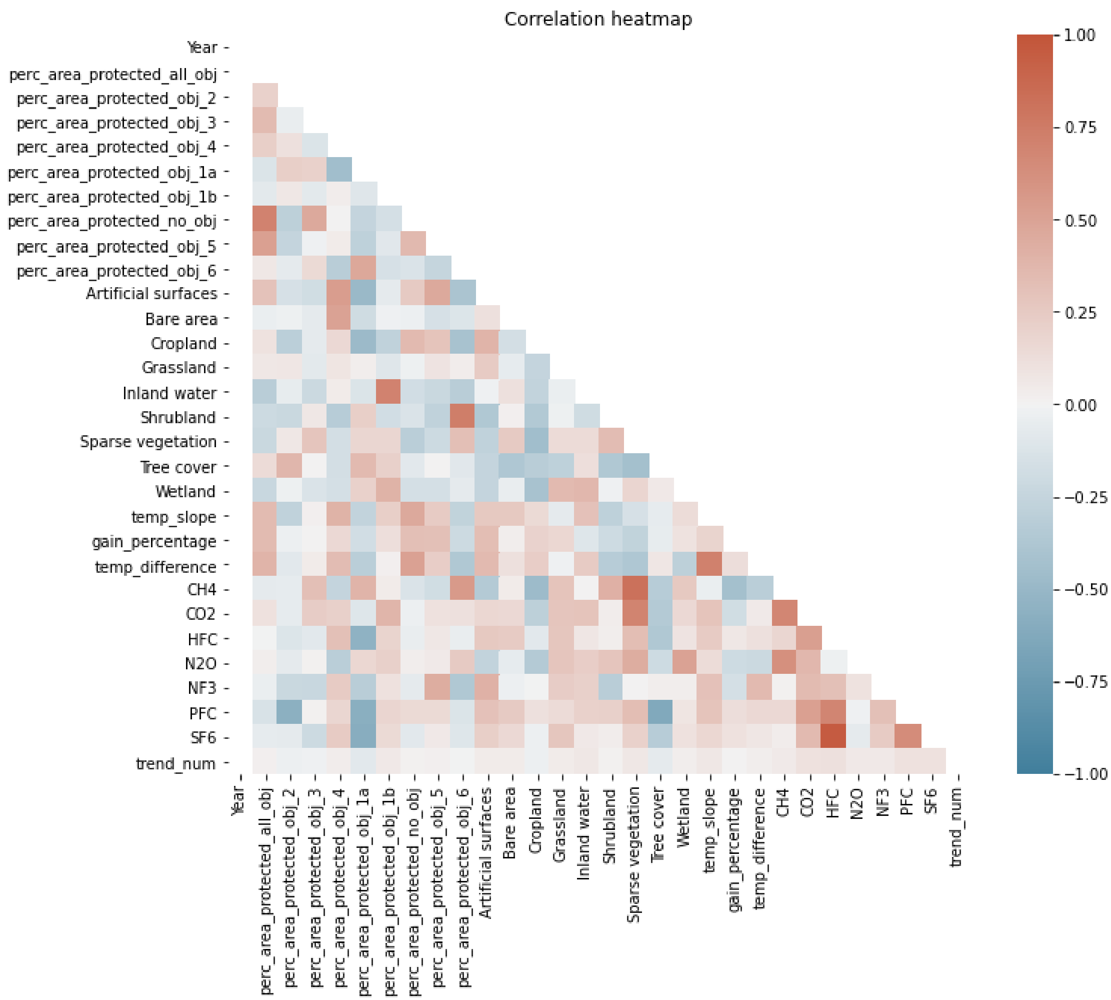
[146]:

	acc
count	30.000000
mean	0.482365
std	0.098560
min	0.438627
25%	0.453754
50%	0.467683
75%	0.474923
max	1.000000

Pearson Correlation

Plot Correlation Matrix

[147]:
```python
correlation_matrix =  full_data.corr(method= 'pearson')
#visualization of the correlation matrix as heatmap

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))
ax.set_title('Correlation heatmap')

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

# create heatmap
sns.heatmap(correlation_matrix, mask=mask,  cmap=cmap, vmin=-1, vmax=1,␣
 ↪center=0)
plt.show()
```



Conclusion There appears to be only a slight correlation between the different support features and the trend. Pearson correlation seems to show more interpretable results. The attributes with the highest correlation for the logistic regression were The transformation of the trend variable before analysing the correlation should be further analysed. Although there only appears to be a slight correlation, we are going to train models for this task.

6 Models

[148]:
```
def extract_from_cv_results(cv_results, row_name=None):
    cv_results = pd.DataFrame(cv_results)
    best_scores = {}
    best_params = {}

    scores = [col.replace('mean_test_', '') for col in cv_results.columns if␣
↪col.startswith('mean_test_')]
```

```
    params = [col for col in cv_results.columns if col.startswith('param_')]

    for score in scores:
        best_params[score] = {}
        # rank is 1 for multiple models if score is equal
        best_model_by_score = cv_results[cv_results[f'rank_test_{score}'] == 1].
↪iloc[0]
        best_scores[f'mean_{score}'] = best_model_by_score[f'mean_test_{score}']
        best_scores[f'std_{score}'] = best_model_by_score[f'std_test_{score}']
        for param in params:
            p = param.split('__')[-1]
            best_params[score][p] = best_model_by_score[param]

    best_params = pd.DataFrame(best_params)
    if row_name is None:
        best_scores = pd.DataFrame(best_scores, index=['value'])
    else:
        best_scores = pd.DataFrame(best_scores, index=[row_name])
        best_params['target'] = row_name
        best_params = best_params.reset_index().set_index(['target', 'index'])
    return best_scores, best_params

def results_by_target(data, cv_results_by_target):
    target_columns = [col for col in data.columns if col.endswith('threatened')␣
↪or col.endswith('trend')]
    best_scores = []
    best_params = []
    for target in target_columns:
        scores, params = extract_from_cv_results(cv_results_by_target[target],␣
↪target)
        best_scores.append(scores)
        best_params.append(params)

    best_scores = pd.concat(best_scores)
    best_scores = best_scores.rename(columns={
        'mean_neg_root_mean_squared_error': 'RMSE',
        'std_neg_root_mean_squared_error': 'RMSE_var',
        'mean_neg_mean_absolute_error': 'MAE',
        'std_neg_mean_absolute_error': 'MAE_var'
    })
    best_scores[['RMSE', 'MAE']] = best_scores[['RMSE', 'MAE']].apply(lambda x:␣
↪-x)
    best_scores[['RMSE_var', 'MAE_var']] = best_scores[['RMSE_var', 'MAE_var']].
↪apply(lambda x: x**2)

    best_params = pd.concat(best_params)
    best_params = best_params.rename(columns={
```

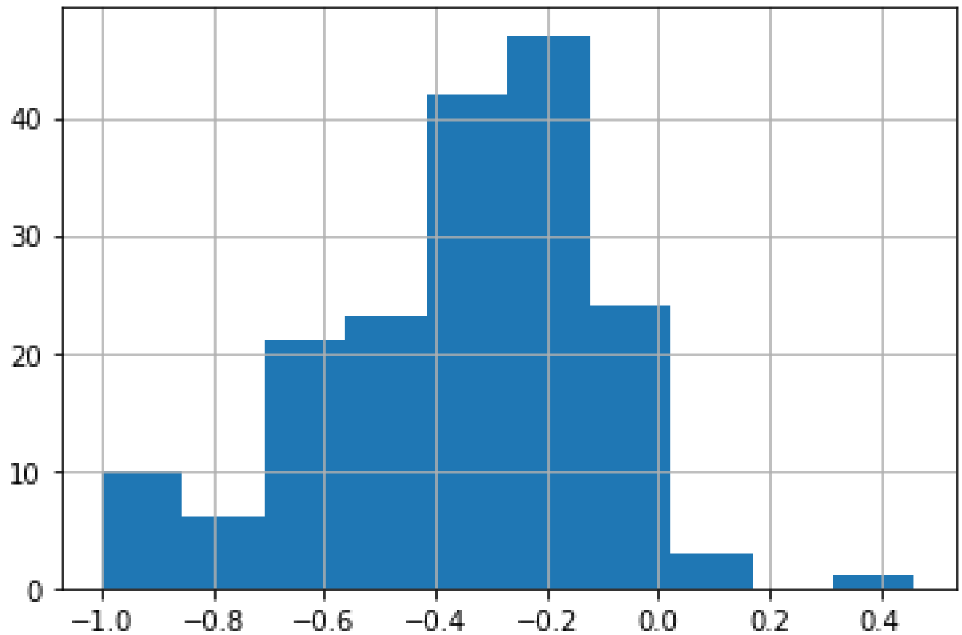
```
        'mean_neg_root_mean_squared_error': 'RMSE',
        'mean_neg_mean_absolute_error': 'MAE',
    })

    return best_scores, best_params

def extract_cv_scores(data, cv_results, score):
    target_columns = [col for col in data.columns if col.endswith('threatened')␣
 ↪or col.endswith('trend')]
    results_by_target = {}
    for target in target_columns:
        results = pd.DataFrame(cv_results[target])
        best_results = results[results[f'rank_test_{score}'] == 1].iloc[0]
        cv_scores = [best_results[f'split{i}_test_{score}'] for i in range(data.
 ↪shape[0])]
        cv_scores = [-val for val in cv_scores]
        results_by_target[target] = cv_scores
    return pd.DataFrame(results_by_target)
```

6.1 Trend

With the data that was merged above, we are now going to train 3 different trend prediction models. The goal is to predict a trend for a given class of species (kingdom_class) with a given country. To do this, we first of all grouped the dataset by kingdom_class and Country. The beforehand transformed trend values were averaged. The task was formulated as regression task. As models we used a Support Vector Machine, a K-NN and a Random Forest. Each of the models was tested on different parameters, by using a grid search with Leave2GroupsOut Cross Validation. The groups that are left out are the respective countries. So for each iteration 2 different countries are held out. ### Group Data by Country and Kingdom. Mean aggregate

[149]:
```
by_country_kingdom = full_data.groupby(['group', 'Country']).mean()
by_country_kingdom = by_country_kingdom.reset_index(drop=False)
by_country_kingdom['trend_num'].hist()
```

[149]: <AxesSubplot:>



Generate Country Labels (Integers) for LeaveGroupOutCV

[150]:
```
country_encoder = LabelEncoder()
country_encoder.fit(by_country_kingdom['Country'])
country_labels = country_encoder.transform(by_country_kingdom['Country'])
```

One Hot Encode kingdom_class and Country

[151]:
```
by_country_kingdom['group'].unique()
```

[151]:
```
array(['amphibians', 'birds', 'insects', 'mammals', 'reptiles'],
      dtype=object)
```

[152]:
```
enc = OneHotEncoder(handle_unknown='ignore')
enc.fit(by_country_kingdom[['group', 'Country']])
oht_features = enc.transform(by_country_kingdom[['group', 'Country']])
col_names = [
    *by_country_kingdom['group'].unique(),
    *by_country_kingdom['Country'].unique()
]
oht_features = pd.DataFrame(oht_features.todense(), columns=col_names)
```

[153]:
```
y = by_country_kingdom['trend_num']
X = by_country_kingdom.drop(['group', 'Country', 'trend_num'], axis=1)
X = pd.concat([X, oht_features], axis=1)
X.describe()
```

[153]:
```
        Year  perc_area_protected_all_obj  perc_area_protected_obj_2  \
count  177.0                   177.000000                 177.000000
mean  2020.0                     0.219106                   0.041050
std      0.0                     0.109557                   0.061404
min   2020.0                     0.056400                   0.000000
25%   2020.0                     0.139400                   0.006200
50%   2020.0                     0.192000                   0.020200
75%   2020.0                     0.279800                   0.048700
max   2020.0                     0.535300                   0.326100

       perc_area_protected_obj_3  perc_area_protected_obj_4  \
count                 177.000000                 177.000000
mean                    0.000584                   0.029312
std                     0.001171                   0.035269
min                     0.000000                   0.000100
25%                     0.000000                   0.003600
50%                     0.000100                   0.015700
75%                     0.000500                   0.039600
max                     0.005900                   0.147400

       perc_area_protected_obj_1a  perc_area_protected_obj_1b  \
count                  177.000000                  177.000000
mean                     0.004686                    0.008090
std                      0.007077                    0.018325
min                      0.000000                    0.000000
25%                      0.000000                    0.000000
50%                      0.000700                    0.000000
75%                      0.005700                    0.003400
max                      0.021800                    0.075300

       perc_area_protected_no_obj  perc_area_protected_obj_5  \
count                  177.000000                 177.000000
mean                     0.073736                   0.041680
std                      0.089332                   0.058039
min                      0.000000                   0.000000
25%                      0.005500                   0.000400
50%                      0.034700                   0.007800
75%                      0.114600                   0.072100
max                      0.403000                   0.252400

       perc_area_protected_obj_6  …  Netherlands      Norway      Poland  \
count                 177.000000  …   177.000000  177.000000  177.000000
mean                    0.016945  …     0.028249    0.028249    0.028249
std                     0.028931  …     0.166152    0.166152    0.166152
min                     0.000000  …     0.000000    0.000000    0.000000
25%                     0.000000  …     0.000000    0.000000    0.000000
50%                     0.000300  …     0.000000    0.000000    0.000000
```

```
75%                        0.024800  …     0.000000     0.000000     0.000000
max                        0.106300  …     1.000000     1.000000     1.000000

         Portugal      Russia    Slovenia       Spain      Sweden  \
count  177.000000  177.000000  177.000000  177.000000  177.000000
mean     0.028249    0.028249    0.028249    0.028249    0.028249
std      0.166152    0.166152    0.166152    0.166152    0.166152
min      0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.000000    0.000000    0.000000    0.000000    0.000000
75%      0.000000    0.000000    0.000000    0.000000    0.000000
max      1.000000    1.000000    1.000000    1.000000    1.000000

       Switzerland     Iceland
count   177.000000  177.000000
mean      0.028249    0.028249
std       0.166152    0.166152
min       0.000000    0.000000
25%       0.000000    0.000000
50%       0.000000    0.000000
75%       0.000000    0.000000
max       1.000000    1.000000

[8 rows x 70 columns]
```

Train Models

[154]:
```
# create pipeline for Model
def train_model(X, y, model='svr'):
    if model == 'svr':
        instance = SVR()
        params = {
        'svr__C': [0.1, 0.4, 1, 5, 10], # todo: inform on parameter ranges
        'svr__kernel': ['linear', 'poly', 'rbf', 'sigmoid']
    }
    elif model == 'knn':
        instance = KNeighborsRegressor()
        params = {
            'knn__n_neighbors': [1, 2, 3 , 4 , 5],
            'knn__weights': ['uniform', 'distance']
    }
    elif model == 'rf':
        instance = RandomForestRegressor(max_features = 'sqrt',␣
 ↪random_state=RANDOM_STATE)
        params = {
        'rf__n_estimators': [int(x) for x in np.linspace(start = 50, stop =␣
 ↪500, num = 10)]
```

```
    }



    pipeline = Pipeline([
        ('scaling', StandardScaler()), (model, instance)
    ])

    logo = LeavePGroupsOut(n_groups=2)

    grid_search = GridSearchCV(pipeline,
                               cv=logo,

                               param_grid=params,
                               scoring=['neg_root_mean_squared_error',␣
↪'neg_mean_absolute_error'],
                               refit='neg_root_mean_squared_error',␣
↪verbose=2, n_jobs=-1)

    grid_search.fit(X, y, groups=country_labels)

    return pd.DataFrame(grid_search.cv_results_)
```

Support Vector Machine

[155]:
```
svr_results = train_model(X, y, model='svr')
```

```
Fitting 630 folds for each of 20 candidates, totalling 12600 fits
```

K-NN

[156]:
```
knn_results = train_model(X, y, model='knn')
```

```
Fitting 630 folds for each of 10 candidates, totalling 6300 fits
```

Random Forest

[157]:
```
rf_results = train_model(X, y, model='rf')
```

```
Fitting 630 folds for each of 10 candidates, totalling 6300 fits
```

Support Vector Machine

[158]:
```
svr_best_scores, svr_best_params = extract_from_cv_results(svr_results)
display(svr_best_scores)
display(svr_best_params)
```

```
       mean_neg_root_mean_squared_error  std_neg_root_mean_squared_error  \
value                         -0.189325                         0.049983
```

```
       mean_neg_mean_absolute_error  std_neg_mean_absolute_error
value                        -0.1491                     0.040948
```

```
       neg_root_mean_squared_error neg_mean_absolute_error
C                                5                       1
kernel                         rbf                     rbf
```

K-NN

[159]:
```
knn_best_scores, knn_best_params = extract_from_cv_results(knn_results)
display(knn_best_scores)
display(knn_best_params)
```

```
       mean_neg_root_mean_squared_error  std_neg_root_mean_squared_error  \
value                          -0.17491                         0.065511

       mean_neg_mean_absolute_error  std_neg_mean_absolute_error
value                     -0.121368                     0.050412
```

```
            neg_root_mean_squared_error neg_mean_absolute_error
n_neighbors                           1                       1
weights                         uniform                 uniform
```

Random Forest

[160]:
```
rf_best_scores, rf_best_params = extract_from_cv_results(rf_results)
display(rf_best_scores)
display(rf_best_params)
```

```
       mean_neg_root_mean_squared_error  std_neg_root_mean_squared_error  \
value                         -0.175428                         0.056499

       mean_neg_mean_absolute_error  std_neg_mean_absolute_error
value                      -0.13161                     0.042879
```

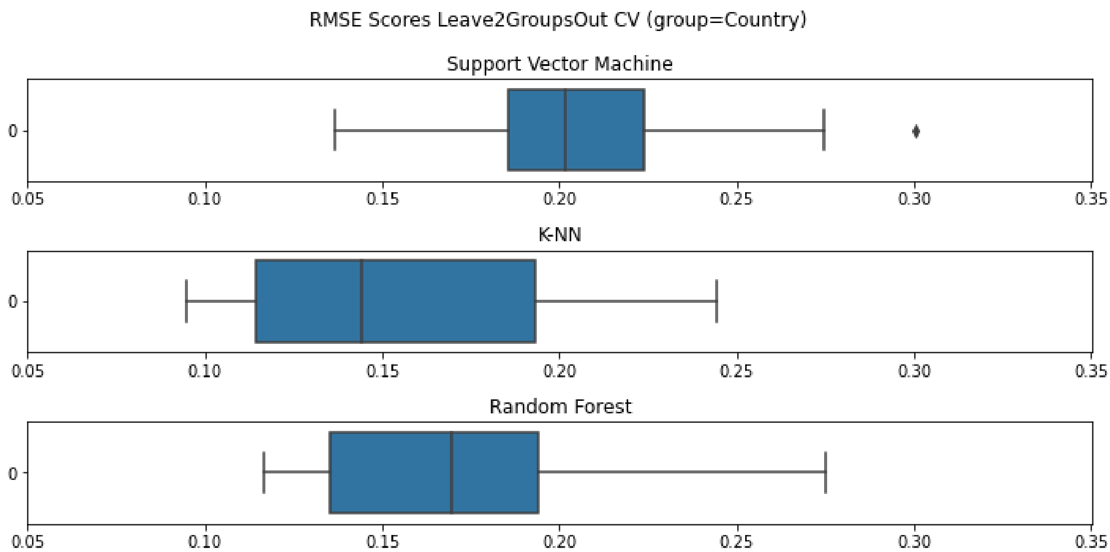
```
              neg_root_mean_squared_error  neg_mean_absolute_error
n_estimators                          400                      400
```

Comparision of classifiers

[161]:
```
def boxplot_results(cv_result_list):
    score = 'neg_root_mean_squared_error'
    fig, axs = plt.subplots(3, figsize=(10, 5))
    fig.suptitle('RMSE Scores Leave2GroupsOut CV (group=Country)')
    max_x = 0
    for i, cv_res in enumerate(cv_result_list):
        results = cv_res
        best_results = results[results[f'rank_test_{score}'] == 1].iloc[0]
        cv_scores = [best_results[f'split{i}_test_{score}'] for i in␣
 ↪range(by_country_kingdom['Country'].nunique())]
        cv_scores = [-val for val in cv_scores]
```

```
        temp_max = np.max(cv_scores)
        if temp_max > max_x:
            max_x = temp_max
        sns.boxplot(data=cv_scores, orient='h', ax=axs[i])
    axs[0].set_title('Support Vector Machine')
    axs[1].set_title('K-NN')
    axs[2].set_title('Random Forest')

    axs[0].set_xlim([0.05,max_x + 0.05])
    axs[1].set_xlim([0.05,max_x + 0.05])
    axs[2].set_xlim([0.05,max_x + 0.05])

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

[162]:
```
boxplot_results([svr_results, knn_results, rf_results])
```



Conclusion The k-NN model seems to work best for the trend prediction. In the best parameter configuration the k-NN reaches a mean RMSE of 0.24 and a MAE of 0.17. The best configuration ranked by RMSE has 3 neighbors and distance weights.

Altough the RMSE of the classifiers seems quite low, the predictions are not really accurate, because the target ranges only between -1.0 and 0.4 (after the grouping).

The reason for this is possibly the data quality (missing time information, different datasets from different years, no historical data, …) and the low correlation between features and target.

6.2 Relative Threatened Species

6.2.1 Check correlation of our features and target values for the relative threatened species by country and group

For our correlation analysis, we exclude our non continuous variables (the binary value whether a taxonomic group has records for that specific country)

[163]:
```
full_threatened_corr = full_threatened.copy()
full_threatened_corr = full_threatened.drop(columns=['mammals_resident',␣
↪'insects_resident', "amphibians_resident", "birds_resident",␣
↪"reptiles_resident", "Year"])
```

[164]:
```
#full correlation matrix to also get an overview of correlations between␣
↪different features
full_threatened_corr.corr(method= 'pearson')
```

[164]:

	total_threatened	reptiles_threatened \
total_threatened	1.000000	0.487928
reptiles_threatened	0.487928	1.000000
mammals_threatened	0.743809	0.396207
amphibians_threatened	0.780868	0.420404
insects_threatened	0.650051	0.182543
birds_threatened	0.688018	0.420726
perc_area_protected_all_obj	-0.190702	-0.138408
perc_area_protected_obj_2	-0.083430	-0.172750
perc_area_protected_obj_3	0.158631	0.093315
perc_area_protected_obj_4	-0.133931	0.013010
perc_area_protected_obj_1a	0.153258	-0.170222
perc_area_protected_obj_1b	-0.303935	-0.226246
perc_area_protected_no_obj	-0.049815	0.113133
perc_area_protected_obj_5	-0.148558	-0.125528
perc_area_protected_obj_6	0.222062	-0.063772
Artificial surfaces	-0.204129	-0.122641
Bare area	0.088596	0.123024
Cropland	-0.009619	-0.094644
Grassland	-0.292366	0.261549
Inland water	-0.303951	-0.168664
Shrubland	0.457536	0.151074
Sparse vegetation	0.093454	-0.056311
Tree cover	0.008004	-0.148923
Wetland	-0.321303	0.119531
temp_slope	-0.304042	-0.348568
gain_percentage	-0.242096	-0.383717
temp_difference	-0.018998	-0.062601
CH4	-0.049575	0.199637
CO2	-0.318721	-0.071803
HFC	0.006884	0.138146

N2O	-0.477619	0.008032
NF3	-0.083185	0.216389
PFC	0.138525	0.228273
SF6	-0.115493	0.040442

	mammals_threatened	amphibians_threatened	\
total_threatened	0.743809	0.780868	
reptiles_threatened	0.396207	0.420404	
mammals_threatened	1.000000	0.483242	
amphibians_threatened	0.483242	1.000000	
insects_threatened	0.183281	0.314565	
birds_threatened	0.783615	0.456921	
perc_area_protected_all_obj	-0.242621	-0.138516	
perc_area_protected_obj_2	-0.008793	0.085890	
perc_area_protected_obj_3	0.234621	0.069787	
perc_area_protected_obj_4	-0.189815	-0.055111	
perc_area_protected_obj_1a	0.275885	0.056411	
perc_area_protected_obj_1b	-0.278684	-0.249485	
perc_area_protected_no_obj	-0.112947	-0.133317	
perc_area_protected_obj_5	-0.181586	-0.255866	
perc_area_protected_obj_6	0.145151	0.352722	
Artificial surfaces	-0.239905	-0.189753	
Bare area	0.266896	0.138933	
Cropland	-0.108520	-0.110511	
Grassland	-0.147231	-0.268581	
Inland water	-0.242812	-0.280541	
Shrubland	0.302860	0.582266	
Sparse vegetation	0.340685	0.047150	
Tree cover	-0.151825	0.032722	
Wetland	-0.125941	-0.354566	
temp_slope	-0.314469	-0.471431	
gain_percentage	-0.207480	-0.234124	
temp_difference	-0.137928	-0.151020	
CH4	0.131589	-0.104334	
CO2	-0.181242	-0.345070	
HFC	-0.229674	0.050534	
N2O	-0.330833	-0.436389	
NF3	-0.181694	-0.209035	
PFC	0.091623	0.014653	
SF6	-0.399398	0.084288	

	insects_threatened	birds_threatened	\
total_threatened	0.650051	0.688018	
reptiles_threatened	0.182543	0.420726	
mammals_threatened	0.183281	0.783615	
amphibians_threatened	0.314565	0.456921	
insects_threatened	1.000000	0.192471	

```
birds_threatened                  0.192471          1.000000
perc_area_protected_all_obj       0.063769         -0.353233
perc_area_protected_obj_2        -0.274201         -0.067479
perc_area_protected_obj_3         0.182625          0.102690
perc_area_protected_obj_4        -0.033785         -0.187105
perc_area_protected_obj_1a        0.048049          0.421984
perc_area_protected_obj_1b       -0.204087         -0.162613
perc_area_protected_no_obj        0.220617         -0.251556
perc_area_protected_obj_5         0.116599         -0.173002
perc_area_protected_obj_6         0.118447          0.176887
Artificial surfaces               0.038503         -0.288874
Bare area                        -0.151139          0.061288
Cropland                          0.254121         -0.174032
Grassland                        -0.351841         -0.258580
Inland water                     -0.236959         -0.127803
Shrubland                         0.226350          0.308445
Sparse vegetation                -0.106568          0.193678
Tree cover                        0.076214          0.135339
Wetland                          -0.421772         -0.063319
temp_slope                        0.033120         -0.270901
gain_percentage                  -0.090725         -0.331767
temp_difference                   0.214659         -0.220272
CH4                              -0.085329          0.142518
CO2                              -0.142365         -0.195325
HFC                               0.166304         -0.360639
N2O                              -0.392401         -0.199088
NF3                               0.053978         -0.020152
PFC                               0.206650         -0.063348
SF6                               0.040041         -0.507985

                            perc_area_protected_all_obj  \
total_threatened                              -0.190702
reptiles_threatened                           -0.138408
mammals_threatened                            -0.242621
amphibians_threatened                         -0.138516
insects_threatened                             0.063769
birds_threatened                              -0.353233
perc_area_protected_all_obj                    1.000000
perc_area_protected_obj_2                      0.342751
perc_area_protected_obj_3                      0.437965
perc_area_protected_obj_4                      0.304553
perc_area_protected_obj_1a                    -0.270099
perc_area_protected_obj_1b                    -0.109045
perc_area_protected_no_obj                     0.623451
perc_area_protected_obj_5                      0.431824
perc_area_protected_obj_6                     -0.037072
Artificial surfaces                            0.304389
```

Bare area	-0.061586
Cropland	0.220840
Grassland	-0.011000
Inland water	-0.414317
Shrubland	-0.190822
Sparse vegetation	-0.242571
Tree cover	0.078702
Wetland	-0.349141
temp_slope	0.241882
gain_percentage	0.239970
temp_difference	0.451564
CH4	-0.175326
CO2	0.199779
HFC	0.041876
N2O	-0.217727
NF3	-0.120748
PFC	-0.144137
SF6	0.031387

	perc_area_protected_obj_2 \
total_threatened	-0.083430
reptiles_threatened	-0.172750
mammals_threatened	-0.008793
amphibians_threatened	0.085890
insects_threatened	-0.274201
birds_threatened	-0.067479
perc_area_protected_all_obj	0.342751
perc_area_protected_obj_2	1.000000
perc_area_protected_obj_3	-0.035364
perc_area_protected_obj_4	0.341617
perc_area_protected_obj_1a	0.063397
perc_area_protected_obj_1b	0.144976
perc_area_protected_no_obj	-0.297550
perc_area_protected_obj_5	-0.238586
perc_area_protected_obj_6	-0.070508
Artificial surfaces	-0.058796
Bare area	0.104776
Cropland	-0.257100
Grassland	0.104067
Inland water	-0.086440
Shrubland	-0.066081
Sparse vegetation	0.266820
Tree cover	0.063818
Wetland	-0.020581
temp_slope	-0.160454
gain_percentage	0.079518
temp_difference	-0.055133

```
CH4                                                 -0.003942
CO2                                                  0.237847
HFC                                                 -0.068479
N2O                                                 -0.049670
NF3                                                 -0.252865
PFC                                                 -0.445276
SF6                                                 -0.028860

                             perc_area_protected_obj_3  \
total_threatened                              0.158631
reptiles_threatened                           0.093315
mammals_threatened                            0.234621
amphibians_threatened                         0.069787
insects_threatened                            0.182625
birds_threatened                              0.102690
perc_area_protected_all_obj                   0.437965
perc_area_protected_obj_2                    -0.035364
perc_area_protected_obj_3                     1.000000
perc_area_protected_obj_4                    -0.147122
perc_area_protected_obj_1a                    0.076860
perc_area_protected_obj_1b                   -0.097715
perc_area_protected_no_obj                    0.575890
perc_area_protected_obj_5                     0.000976
perc_area_protected_obj_6                     0.167304
Artificial surfaces                          -0.207200
Bare area                                    -0.060732
Cropland                                     -0.059717
Grassland                                    -0.145111
Inland water                                 -0.226226
Shrubland                                     0.168077
Sparse vegetation                             0.126819
Tree cover                                    0.123182
Wetland                                      -0.141988
temp_slope                                    0.093957
gain_percentage                               0.005061
temp_difference                               0.211492
CH4                                           0.178681
CO2                                           0.061374
HFC                                          -0.008055
N2O                                          -0.099817
NF3                                          -0.248710
PFC                                           0.065248
SF6                                          -0.122293

                             perc_area_protected_obj_4  …  temp_slope  \
total_threatened                             -0.133931  …   -0.304042
reptiles_threatened                           0.013010  …   -0.348568
```

mammals_threatened	-0.189815	…	-0.314469
amphibians_threatened	-0.055111	…	-0.471431
insects_threatened	-0.033785	…	0.033120
birds_threatened	-0.187105	…	-0.270901
perc_area_protected_all_obj	0.304553	…	0.241882
perc_area_protected_obj_2	0.341617	…	-0.160454
perc_area_protected_obj_3	-0.147122	…	0.093957
perc_area_protected_obj_4	1.000000	…	0.261813
perc_area_protected_obj_1a	-0.315319	…	-0.155026
perc_area_protected_obj_1b	-0.016646	…	0.194946
perc_area_protected_no_obj	-0.119119	…	0.314547
perc_area_protected_obj_5	-0.083504	…	0.100167
perc_area_protected_obj_6	-0.176005	…	-0.247446
Artificial surfaces	0.452115	…	0.102427
Bare area	0.401496	…	0.283061
Cropland	0.051990	…	0.158579
Grassland	0.096529	…	-0.276609
Inland water	-0.066407	…	0.234408
Shrubland	-0.200052	…	-0.304867
Sparse vegetation	-0.204527	…	-0.066613
Tree cover	-0.153041	…	-0.011481
Wetland	-0.254120	…	-0.021914
temp_slope	0.261813	…	1.000000
gain_percentage	0.023241	…	0.286844
temp_difference	0.287294	…	0.646629
CH4	-0.246204	…	-0.136897
CO2	0.278410	…	0.228880
HFC	0.249179	…	0.275832
N2O	-0.311729	…	-0.038762
NF3	0.066048	…	-0.012243
PFC	-0.088547	…	0.239156
SF6	0.178582	…	0.178723

	gain_percentage	temp_difference	CH4	\
total_threatened	-0.242096	-0.018998	-0.049575	
reptiles_threatened	-0.383717	-0.062601	0.199637	
mammals_threatened	-0.207480	-0.137928	0.131589	
amphibians_threatened	-0.234124	-0.151020	-0.104334	
insects_threatened	-0.090725	0.214659	-0.085329	
birds_threatened	-0.331767	-0.220272	0.142518	
perc_area_protected_all_obj	0.239970	0.451564	-0.175326	
perc_area_protected_obj_2	0.079518	-0.055133	-0.003942	
perc_area_protected_obj_3	0.005061	0.211492	0.178681	
perc_area_protected_obj_4	0.023241	0.287294	-0.246204	
perc_area_protected_obj_1a	-0.029378	-0.308639	0.311201	
perc_area_protected_obj_1b	0.270173	-0.120579	-0.056308	
perc_area_protected_no_obj	0.166415	0.512871	-0.086042	

perc_area_protected_obj_5	0.179241	0.193640	-0.269094
perc_area_protected_obj_6	-0.281555	-0.301630	0.432951
Artificial surfaces	0.138753	0.362684	-0.343022
Bare area	0.066499	0.099780	0.033368
Cropland	-0.020582	0.283444	-0.389897
Grassland	0.092766	-0.110044	0.349298
Inland water	0.097376	0.040022	0.030144
Shrubland	-0.263196	-0.205757	0.366741
Sparse vegetation	-0.056834	-0.254263	0.625749
Tree cover	0.003027	-0.049373	-0.298557
Wetland	0.260720	-0.429397	0.323698
temp_slope	0.286844	0.646629	-0.136897
gain_percentage	1.000000	0.102618	-0.432879
temp_difference	0.102618	1.000000	-0.247403
CH4	-0.432879	-0.247403	1.000000
CO2	-0.123687	0.088991	0.526680
HFC	0.081595	0.144746	0.133682
N2O	-0.128619	-0.340967	0.609650
NF3	-0.224060	0.192270	0.064788
PFC	0.186874	0.212732	0.096548
SF6	0.137347	0.080700	-0.016862

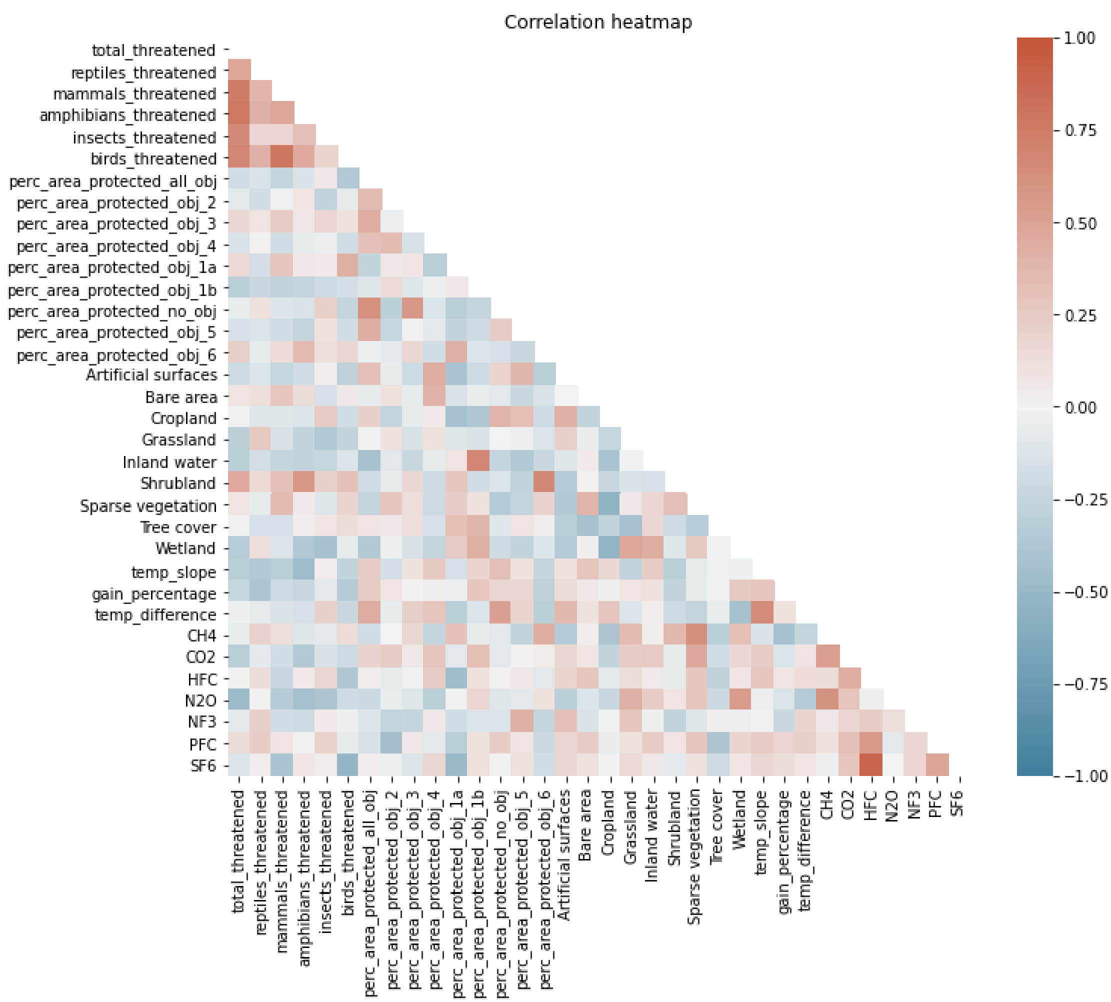
	CO2	HFC	N2O	NF3	PFC \
total_threatened	-0.318721	0.006884	-0.477619	-0.083185	0.138525
reptiles_threatened	-0.071803	0.138146	0.008032	0.216389	0.228273
mammals_threatened	-0.181242	-0.229674	-0.330833	-0.181694	0.091623
amphibians_threatened	-0.345070	0.050534	-0.436389	-0.209035	0.014653
insects_threatened	-0.142365	0.166304	-0.392401	0.053978	0.206650
birds_threatened	-0.195325	-0.360639	-0.199088	-0.020152	-0.063348
perc_area_protected_all_obj	0.199779	0.041876	-0.217727	-0.120748	-0.144137
perc_area_protected_obj_2	0.237847	-0.068479	-0.049670	-0.252865	-0.445276
perc_area_protected_obj_3	0.061374	-0.008055	-0.099817	-0.248710	0.065248
perc_area_protected_obj_4	0.278410	0.249179	-0.311729	0.066048	-0.088547
perc_area_protected_obj_1a	-0.088792	-0.450892	0.010026	-0.183743	-0.315735
perc_area_protected_obj_1b	0.327811	0.110123	0.180279	-0.117689	0.098447
perc_area_protected_no_obj	-0.082928	0.013579	-0.096691	-0.132804	0.230877
perc_area_protected_obj_5	0.015081	0.055060	-0.083105	0.424400	0.087713
perc_area_protected_obj_6	0.044327	-0.088838	0.102866	-0.243762	-0.218115
Artificial surfaces	0.166480	0.191015	-0.298447	0.307646	0.178194
Bare area	0.082796	0.280550	-0.111638	-0.135591	0.231254
Cropland	-0.272775	-0.084155	-0.223289	-0.009721	-0.041661
Grassland	0.229430	0.136050	0.414629	0.289647	0.110109
Inland water	0.250312	0.052937	0.230707	-0.022912	0.232662
Shrubland	-0.069625	-0.067377	0.092544	-0.260078	0.088137
Sparse vegetation	0.465685	0.265656	0.288458	-0.100597	0.283018
Tree cover	-0.174769	-0.282723	-0.213456	0.030960	-0.396906
Wetland	0.159443	0.081298	0.543903	-0.028320	0.192014

temp_slope	0.228880	0.275832	-0.038762	-0.012243	0.239156
gain_percentage	-0.123687	0.081595	-0.128619	-0.224060	0.186874
temp_difference	0.088991	0.144746	-0.340967	0.192270	0.212732
CH4	0.526680	0.133682	0.609650	0.064788	0.096548
CO2	1.000000	0.442377	0.290425	0.185835	0.314702
HFC	0.442377	1.000000	0.017071	0.247790	0.559334
N2O	0.290425	0.017071	1.000000	0.128588	-0.085272
NF3	0.185835	0.247790	0.128588	1.000000	0.169645
PFC	0.314702	0.559334	-0.085272	0.169645	1.000000
SF6	0.295890	0.897198	-0.000536	0.179777	0.496733

	SF6
total_threatened	-0.115493
reptiles_threatened	0.040442
mammals_threatened	-0.399398
amphibians_threatened	0.084288
insects_threatened	0.040041
birds_threatened	-0.507985
perc_area_protected_all_obj	0.031387
perc_area_protected_obj_2	-0.028860
perc_area_protected_obj_3	-0.122293
perc_area_protected_obj_4	0.178582
perc_area_protected_obj_1a	-0.498784
perc_area_protected_obj_1b	0.104810
perc_area_protected_no_obj	0.003607
perc_area_protected_obj_5	0.098653
perc_area_protected_obj_6	-0.185290
Artificial surfaces	0.178994
Bare area	0.098992
Cropland	0.006913
Grassland	0.156174
Inland water	0.067721
Shrubland	-0.085620
Sparse vegetation	0.107343
Tree cover	-0.209873
Wetland	0.085516
temp_slope	0.178723
gain_percentage	0.137347
temp_difference	0.080700
CH4	-0.016862
CO2	0.295890
HFC	0.897198
N2O	-0.000536
NF3	0.179777
PFC	0.496733
SF6	1.000000

[34 rows x 34 columns]

[165]:
```python
#visualization of the correlation matrix as heatmap
correlation_matrix = full_threatened_corr.corr(method='pearson')
# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))
ax.set_title('Correlation heatmap')

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

# create heatmap
sns.heatmap(correlation_matrix, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0)
plt.show()
```



In our correlation matrix and our heatmap we can see the contribution of our features to the different target values, as well as the intercorrelation between the features. In general we can of course see that the number of total threatened species correlate with the numbers per group. Interestingly we also have a strong correlation between the mammals and the birds, which could suggest that there are common factors the lead to the threat of those specific taxonomic groups in a country. What we also can see and what is surprising is that there are hardly any strong intercorrelations between features. Our highest correlation can be seen for the greenhouse gases SF6 and HFC.

To get a better overview of our targets and the correlation of the different features with them we calculate the correlation of the features with each target separately.

[166]:
```
#pearsons rho for the correlation of the values for all groups with our features
for col in full_threatened_corr:
    target = 'total_threatened'
    exclude =  ['Country','total_threatened', 'mammals_threatened',␣
↪'insects_threatened', 'amphibians_threatened','birds_threatened',␣
↪'reptiles_threatened']
    correlations = {}
    if col not in exclude:
        correlations[col + ' pearsons correlation with ' + target] =␣
↪round(full_threatened_corr[col].corr(full_threatened_corr[target]),5)
        print(correlations)
```

```
{'perc_area_protected_all_obj pearsons correlation with total_threatened':
-0.1907}
{'perc_area_protected_obj_2 pearsons correlation with total_threatened':
-0.08343}
{'perc_area_protected_obj_3 pearsons correlation with total_threatened':
0.15863}
{'perc_area_protected_obj_4 pearsons correlation with total_threatened':
-0.13393}
{'perc_area_protected_obj_1a pearsons correlation with total_threatened':
0.15326}
{'perc_area_protected_obj_1b pearsons correlation with total_threatened':
-0.30394}
{'perc_area_protected_no_obj pearsons correlation with total_threatened':
-0.04981}
{'perc_area_protected_obj_5 pearsons correlation with total_threatened':
-0.14856}
{'perc_area_protected_obj_6 pearsons correlation with total_threatened':
0.22206}
{'Artificial surfaces pearsons correlation with total_threatened': -0.20413}
{'Bare area pearsons correlation with total_threatened': 0.0886}
{'Cropland pearsons correlation with total_threatened': -0.00962}
{'Grassland pearsons correlation with total_threatened': -0.29237}
{'Inland water pearsons correlation with total_threatened': -0.30395}
```

```
{'Shrubland pearsons correlation with total_threatened': 0.45754}
{'Sparse vegetation pearsons correlation with total_threatened': 0.09345}
{'Tree cover pearsons correlation with total_threatened': 0.008}
{'Wetland pearsons correlation with total_threatened': -0.3213}
{'temp_slope pearsons correlation with total_threatened': -0.30404}
{'gain_percentage pearsons correlation with total_threatened': -0.2421}
{'temp_difference pearsons correlation with total_threatened': -0.019}
{'CH4 pearsons correlation with total_threatened': -0.04958}
{'CO2 pearsons correlation with total_threatened': -0.31872}
{'HFC pearsons correlation with total_threatened': 0.00688}
{'N2O pearsons correlation with total_threatened': -0.47762}
{'NF3 pearsons correlation with total_threatened': -0.08318}
{'PFC pearsons correlation with total_threatened': 0.13853}
{'SF6 pearsons correlation with total_threatened': -0.11549}
```

[167]:
```
#pearsons rho for the correlation of the threatened mammals with our features
for col in full_threatened_corr:
    target = 'mammals_threatened'
    exclude =  ['Country','total_threatened', 'mammals_threatened',␣
 ↪'insects_threatened', 'amphibians_threatened','birds_threatened',␣
 ↪'reptiles_threatened']
    correlations = {}
    if col not in exclude:
        correlations[col + ' pearsons correlation with ' + target] =␣
 ↪round(full_threatened_corr[col].corr(full_threatened_corr[target]),5)
        print(correlations)
```

```
{'perc_area_protected_all_obj pearsons correlation with mammals_threatened':
-0.24262}
{'perc_area_protected_obj_2 pearsons correlation with mammals_threatened':
-0.00879}
{'perc_area_protected_obj_3 pearsons correlation with mammals_threatened':
0.23462}
{'perc_area_protected_obj_4 pearsons correlation with mammals_threatened':
-0.18982}
{'perc_area_protected_obj_1a pearsons correlation with mammals_threatened':
0.27588}
{'perc_area_protected_obj_1b pearsons correlation with mammals_threatened':
-0.27868}
{'perc_area_protected_no_obj pearsons correlation with mammals_threatened':
-0.11295}
{'perc_area_protected_obj_5 pearsons correlation with mammals_threatened':
-0.18159}
{'perc_area_protected_obj_6 pearsons correlation with mammals_threatened':
0.14515}
{'Artificial surfaces pearsons correlation with mammals_threatened': -0.2399}
{'Bare area pearsons correlation with mammals_threatened': 0.2669}
{'Cropland pearsons correlation with mammals_threatened': -0.10852}
```

```
{'Grassland pearsons correlation with mammals_threatened': -0.14723}
{'Inland water pearsons correlation with mammals_threatened': -0.24281}
{'Shrubland pearsons correlation with mammals_threatened': 0.30286}
{'Sparse vegetation pearsons correlation with mammals_threatened': 0.34068}
{'Tree cover pearsons correlation with mammals_threatened': -0.15183}
{'Wetland pearsons correlation with mammals_threatened': -0.12594}
{'temp_slope pearsons correlation with mammals_threatened': -0.31447}
{'gain_percentage pearsons correlation with mammals_threatened': -0.20748}
{'temp_difference pearsons correlation with mammals_threatened': -0.13793}
{'CH4 pearsons correlation with mammals_threatened': 0.13159}
{'CO2 pearsons correlation with mammals_threatened': -0.18124}
{'HFC pearsons correlation with mammals_threatened': -0.22967}
{'N2O pearsons correlation with mammals_threatened': -0.33083}
{'NF3 pearsons correlation with mammals_threatened': -0.18169}
{'PFC pearsons correlation with mammals_threatened': 0.09162}
{'SF6 pearsons correlation with mammals_threatened': -0.3994}
```

[168]:

```
#pearsons rho for the correlation of the threatened insects with our features
for col in full_threatened_corr:
    target = 'insects_threatened'
    exclude =  ['Country','total_threatened', 'mammals_threatened',␣
 ↪'insects_threatened', 'amphibians_threatened','birds_threatened',␣
 ↪'reptiles_threatened']
    correlations = {}
    if col not in exclude:
        correlations[col + ' pearsons correlation with ' + target] =␣
 ↪round(full_threatened_corr[col].corr(full_threatened_corr[target]),5)
        print(correlations)
```

```
{'perc_area_protected_all_obj pearsons correlation with insects_threatened':
0.06377}
{'perc_area_protected_obj_2 pearsons correlation with insects_threatened':
-0.2742}
{'perc_area_protected_obj_3 pearsons correlation with insects_threatened':
0.18263}
{'perc_area_protected_obj_4 pearsons correlation with insects_threatened':
-0.03378}
{'perc_area_protected_obj_1a pearsons correlation with insects_threatened':
0.04805}
{'perc_area_protected_obj_1b pearsons correlation with insects_threatened':
-0.20409}
{'perc_area_protected_no_obj pearsons correlation with insects_threatened':
0.22062}
{'perc_area_protected_obj_5 pearsons correlation with insects_threatened':
0.1166}
{'perc_area_protected_obj_6 pearsons correlation with insects_threatened':
0.11845}
{'Artificial surfaces pearsons correlation with insects_threatened': 0.0385}
```

```
{'Bare area pearsons correlation with insects_threatened': -0.15114}
{'Cropland pearsons correlation with insects_threatened': 0.25412}
{'Grassland pearsons correlation with insects_threatened': -0.35184}
{'Inland water pearsons correlation with insects_threatened': -0.23696}
{'Shrubland pearsons correlation with insects_threatened': 0.22635}
{'Sparse vegetation pearsons correlation with insects_threatened': -0.10657}
{'Tree cover pearsons correlation with insects_threatened': 0.07621}
{'Wetland pearsons correlation with insects_threatened': -0.42177}
{'temp_slope pearsons correlation with insects_threatened': 0.03312}
{'gain_percentage pearsons correlation with insects_threatened': -0.09072}
{'temp_difference pearsons correlation with insects_threatened': 0.21466}
{'CH4 pearsons correlation with insects_threatened': -0.08533}
{'CO2 pearsons correlation with insects_threatened': -0.14237}
{'HFC pearsons correlation with insects_threatened': 0.1663}
{'N2O pearsons correlation with insects_threatened': -0.3924}
{'NF3 pearsons correlation with insects_threatened': 0.05398}
{'PFC pearsons correlation with insects_threatened': 0.20665}
{'SF6 pearsons correlation with insects_threatened': 0.04004}
```

[169]:

```
#pearsons rho for the correlation of the threatened amphibians with our features
for col in full_threatened_corr:
    target = 'amphibians_threatened'
    exclude =  ['Country','total_threatened', 'mammals_threatened',␣
↪'insects_threatened', 'amphibians_threatened','birds_threatened',␣
↪'reptiles_threatened']
    correlations = {}
    if col not in exclude:
        correlations[col + ' pearsons correlation with ' + target] =␣
↪round(full_threatened_corr[col].corr(full_threatened_corr[target]),5)
        print(correlations)
```

```
{'perc_area_protected_all_obj pearsons correlation with amphibians_threatened':
-0.13852}
{'perc_area_protected_obj_2 pearsons correlation with amphibians_threatened':
0.08589}
{'perc_area_protected_obj_3 pearsons correlation with amphibians_threatened':
0.06979}
{'perc_area_protected_obj_4 pearsons correlation with amphibians_threatened':
-0.05511}
{'perc_area_protected_obj_1a pearsons correlation with amphibians_threatened':
0.05641}
{'perc_area_protected_obj_1b pearsons correlation with amphibians_threatened':
-0.24949}
{'perc_area_protected_no_obj pearsons correlation with amphibians_threatened':
-0.13332}
{'perc_area_protected_obj_5 pearsons correlation with amphibians_threatened':
-0.25587}
{'perc_area_protected_obj_6 pearsons correlation with amphibians_threatened':
```

```
0.35272}
{'Artificial surfaces pearsons correlation with amphibians_threatened':
-0.18975}
{'Bare area pearsons correlation with amphibians_threatened': 0.13893}
{'Cropland pearsons correlation with amphibians_threatened': -0.11051}
{'Grassland pearsons correlation with amphibians_threatened': -0.26858}
{'Inland water pearsons correlation with amphibians_threatened': -0.28054}
{'Shrubland pearsons correlation with amphibians_threatened': 0.58227}
{'Sparse vegetation pearsons correlation with amphibians_threatened': 0.04715}
{'Tree cover pearsons correlation with amphibians_threatened': 0.03272}
{'Wetland pearsons correlation with amphibians_threatened': -0.35457}
{'temp_slope pearsons correlation with amphibians_threatened': -0.47143}
{'gain_percentage pearsons correlation with amphibians_threatened': -0.23412}
{'temp_difference pearsons correlation with amphibians_threatened': -0.15102}
{'CH4 pearsons correlation with amphibians_threatened': -0.10433}
{'CO2 pearsons correlation with amphibians_threatened': -0.34507}
{'HFC pearsons correlation with amphibians_threatened': 0.05053}
{'N2O pearsons correlation with amphibians_threatened': -0.43639}
{'NF3 pearsons correlation with amphibians_threatened': -0.20903}
{'PFC pearsons correlation with amphibians_threatened': 0.01465}
{'SF6 pearsons correlation with amphibians_threatened': 0.08429}
```

[170]:
```
#pearsons rho for the correlation of the threatened birds with our features
for col in full_threatened_corr:
    target = 'birds_threatened'
    exclude =  ['Country','total_threatened', 'mammals_threatened',␣
 ↪'insects_threatened', 'amphibians_threatened','birds_threatened',␣
 ↪'reptiles_threatened']
    correlations = {}
    if col not in exclude:
        correlations[col + ' pearsons correlation with ' + target] =␣
 ↪round(full_threatened_corr[col].corr(full_threatened_corr[target]),5)
        print(correlations)
```

```
{'perc_area_protected_all_obj pearsons correlation with birds_threatened':
-0.35323}
{'perc_area_protected_obj_2 pearsons correlation with birds_threatened':
-0.06748}
{'perc_area_protected_obj_3 pearsons correlation with birds_threatened':
0.10269}
{'perc_area_protected_obj_4 pearsons correlation with birds_threatened':
-0.18711}
{'perc_area_protected_obj_1a pearsons correlation with birds_threatened':
0.42198}
{'perc_area_protected_obj_1b pearsons correlation with birds_threatened':
-0.16261}
{'perc_area_protected_no_obj pearsons correlation with birds_threatened':
-0.25156}
```

```
{'perc_area_protected_obj_5 pearsons correlation with birds_threatened': -0.173}
{'perc_area_protected_obj_6 pearsons correlation with birds_threatened':
0.17689}
{'Artificial surfaces pearsons correlation with birds_threatened': -0.28887}
{'Bare area pearsons correlation with birds_threatened': 0.06129}
{'Cropland pearsons correlation with birds_threatened': -0.17403}
{'Grassland pearsons correlation with birds_threatened': -0.25858}
{'Inland water pearsons correlation with birds_threatened': -0.1278}
{'Shrubland pearsons correlation with birds_threatened': 0.30845}
{'Sparse vegetation pearsons correlation with birds_threatened': 0.19368}
{'Tree cover pearsons correlation with birds_threatened': 0.13534}
{'Wetland pearsons correlation with birds_threatened': -0.06332}
{'temp_slope pearsons correlation with birds_threatened': -0.2709}
{'gain_percentage pearsons correlation with birds_threatened': -0.33177}
{'temp_difference pearsons correlation with birds_threatened': -0.22027}
{'CH4 pearsons correlation with birds_threatened': 0.14252}
{'CO2 pearsons correlation with birds_threatened': -0.19532}
{'HFC pearsons correlation with birds_threatened': -0.36064}
{'N2O pearsons correlation with birds_threatened': -0.19909}
{'NF3 pearsons correlation with birds_threatened': -0.02015}
{'PFC pearsons correlation with birds_threatened': -0.06335}
{'SF6 pearsons correlation with birds_threatened': -0.50798}
```

[171]:
```python
#pearsons rho for the correlation of the threatened reptiles with our features
for col in full_threatened_corr:
    target = 'reptiles_threatened'
    exclude =  ['Country','total_threatened', 'mammals_threatened',␣
↪'insects_threatened', 'amphibians_threatened','birds_threatened',␣
↪'reptiles_threatened']
    correlations = {}
    if col not in exclude:
        correlations[col + ' pearsons correlation with ' + target] =␣
↪round(full_threatened_corr[col].corr(full_threatened_corr[target]),5)
        print(correlations)
```

```
{'perc_area_protected_all_obj pearsons correlation with reptiles_threatened':
-0.13841}
{'perc_area_protected_obj_2 pearsons correlation with reptiles_threatened':
-0.17275}
{'perc_area_protected_obj_3 pearsons correlation with reptiles_threatened':
0.09332}
{'perc_area_protected_obj_4 pearsons correlation with reptiles_threatened':
0.01301}
{'perc_area_protected_obj_1a pearsons correlation with reptiles_threatened':
-0.17022}
{'perc_area_protected_obj_1b pearsons correlation with reptiles_threatened':
-0.22625}
{'perc_area_protected_no_obj pearsons correlation with reptiles_threatened':
```

```
0.11313}
{'perc_area_protected_obj_5 pearsons correlation with reptiles_threatened':
-0.12553}
{'perc_area_protected_obj_6 pearsons correlation with reptiles_threatened':
-0.06377}
{'Artificial surfaces pearsons correlation with reptiles_threatened': -0.12264}
{'Bare area pearsons correlation with reptiles_threatened': 0.12302}
{'Cropland pearsons correlation with reptiles_threatened': -0.09464}
{'Grassland pearsons correlation with reptiles_threatened': 0.26155}
{'Inland water pearsons correlation with reptiles_threatened': -0.16866}
{'Shrubland pearsons correlation with reptiles_threatened': 0.15107}
{'Sparse vegetation pearsons correlation with reptiles_threatened': -0.05631}
{'Tree cover pearsons correlation with reptiles_threatened': -0.14892}
{'Wetland pearsons correlation with reptiles_threatened': 0.11953}
{'temp_slope pearsons correlation with reptiles_threatened': -0.34857}
{'gain_percentage pearsons correlation with reptiles_threatened': -0.38372}
{'temp_difference pearsons correlation with reptiles_threatened': -0.0626}
{'CH4 pearsons correlation with reptiles_threatened': 0.19964}
{'CO2 pearsons correlation with reptiles_threatened': -0.0718}
{'HFC pearsons correlation with reptiles_threatened': 0.13815}
{'N2O pearsons correlation with reptiles_threatened': 0.00803}
{'NF3 pearsons correlation with reptiles_threatened': 0.21639}
{'PFC pearsons correlation with reptiles_threatened': 0.22827}
{'SF6 pearsons correlation with reptiles_threatened': 0.04044}
```

6.2.2 Correlation Findings

For the all taxonomic groups combined we have the highest positive correlation with the shrubland (0.46) and the highest neagtive correlation with nitrous oxide(-0.48). For the mammals we have the sparse vegetation (0.34) and the temp_slope (-0.31). For insects cropland(0.25) and wetland(-0.42). For the amphibians shrubland(0.58) and temp_slope(-0.47). For the birds the protected area 1a(0.42) and the sulfur hexafluoride(-0.5) and for the reptiles the grassland(0.26) and gain percentage (-0.38). Looking at those values and referring back to one of our initial questions, we can't say that there are any high correlations, so there are no characteristics we observed that highly influence the threatened species per country, but wehave some attributes that contribute to our targets. As we divided the features into four groups (protected areas, land cover, temperature and greenhouse gases), we can see that for every group we have at least some variables that contributed to the prediction of our targets, so we decide to keep those four groups for our model building process and start with the full model.

6.3 Relative Number of Threatened Species

6.3.1 Modeling the data

The target values that we are intersted in are the relative number of threatened species per group and the relative numbers of the trend. For this we create a model for each target value und capture the RMSE and MAE for each model. As the goal is to see the predicitve power of the models we do not perform exhaustive hyperparameter optimization but only run a Grid Search by model and use the model with the best observed RMSE. The best params per target are reported.

For the evaluation we use Leave on Out CV for the model because we are limited in the number of observations (countries)

[172]:
```
data = pd.concat([
        ds_climate.set_index('Country'),
        ds_ghg.set_index('Country'),
        ds_land_cover.set_index('Country'),
        ds_protected_areas.set_index('Country'),
        ds_threatened_by_group.set_index('Country'),
        ds_trend_by_group],
        join='inner',
        axis=1)
data.shape
```

[172]:
```
(42, 43)
```

[173]:
```
data.columns
```

[173]:
```
Index(['temp_slope', 'gain_percentage', 'temp_difference', 'CH4', 'CO2', 'HFC',
       'N2O', 'NF3', 'PFC', 'SF6', 'Artificial surfaces', 'Bare area',
       'Cropland', 'Grassland', 'Inland water', 'Shrubland',
       'Sparse vegetation', 'Tree cover', 'Wetland', 'Year',
       'perc_area_protected_all_obj', 'perc_area_protected_obj_2',
       'perc_area_protected_obj_3', 'perc_area_protected_obj_4',
       'perc_area_protected_obj_1a', 'perc_area_protected_obj_1b',
       'perc_area_protected_no_obj', 'perc_area_protected_obj_5',
       'perc_area_protected_obj_6', 'total_threatened', 'reptiles_threatened',
       'mammals_threatened', 'amphibians_threatened', 'insects_threatened',
       'birds_threatened', 'reptiles_resident', 'mammals_resident',
       'amphibians_resident', 'insects_resident', 'birds_resident',
       'decreasing_trend', 'increasing_trend', 'stable_trend'],
      dtype='object')
```

[174]:
```
data.describe()
```

[174]:
```
       temp_slope  gain_percentage  temp_difference        CH4        CO2  \
count   42.000000        42.000000        42.000000  42.000000  42.000000
mean     0.032865        10.864795         0.987359   1.284412   7.478007
std      0.013226        12.305045         0.448411   1.194647   4.017188
min      0.006156       -24.178355         0.072246   0.236113   1.278950
25%      0.021700         3.832235         0.659961   0.750174   4.848182
50%      0.034182         9.547614         0.943211   0.893499   6.846338
75%      0.038667        17.897105         1.382112   1.261372   9.078078
max      0.058883        43.855657         1.852533   7.017958  16.642911

             HFC        N2O        NF3        PFC        SF6  …  \
count  42.000000  42.000000  42.000000  42.000000  42.000000  …
```

```
mean    0.164214   0.586226  -0.666490  -0.154093  -0.036179  …
std     0.297642   0.349903   0.477372   0.384327   0.219119  …
min    -1.000000   0.092906  -1.000000  -1.000000  -1.000000  …
25%     0.109285   0.345235  -1.000000   0.000021   0.002429  …
50%     0.176891   0.481730  -1.000000   0.003712   0.007283  …
75%     0.322812   0.849686   0.000009   0.012341   0.010464  …
max     0.550778   1.554079   0.002234   0.216726   0.128989  …

       perc_area_protected_obj_6  total_threatened  reptiles_threatened  \
count                  42.000000         42.000000            42.000000
mean                    0.015333          0.090214             0.113638
std                     0.027411          0.059176             0.150635
min                     0.000000          0.016700             0.000000
25%                     0.000000          0.050675             0.000000
50%                     0.000850          0.071400             0.077000
75%                     0.015525          0.124150             0.158400
max                     0.106300          0.357500             0.750000

       mammals_threatened  amphibians_threatened  insects_threatened  \
count           42.000000              42.000000           42.000000
mean             0.101226               0.127510            0.093860
std              0.069002               0.179915            0.059323
min              0.000000               0.000000            0.000000
25%              0.051750               0.000000            0.065525
50%              0.089700               0.032150            0.086850
75%              0.142075               0.209625            0.103925
max              0.294000               0.750000            0.304300

       birds_threatened  decreasing_trend  increasing_trend  stable_trend
count         42.000000         42.000000         42.000000     42.000000
mean           0.058579          0.325095          0.110214      0.310885
std            0.042883          0.039627          0.046089      0.060754
min            0.016000          0.195591          0.015253      0.175182
25%            0.038250          0.313731          0.080191      0.278904
50%            0.044300          0.335039          0.121524      0.302402
75%            0.064075          0.349774          0.144481      0.323285
max            0.290000          0.372725          0.233577      0.485924

[8 rows x 38 columns]
```

[175]:
```
# are there any missing numbers
data.isna().any().sum()
```

[175]:
```
0
```

6.3.2 kNN

[176]:
```
def knn_by_target(data):
    cv_results = {}

    # iterate all tartget veriables
    target_columns = [col for col in data.columns if col.endswith('threatened')␣
↪or col.endswith('trend')]
    for target in target_columns:
        y = data[target]
        X = data.drop(columns=target_columns)

        # create pipeline for Model
        svr = Pipeline([
            ('scaling', StandardScaler()),
            ('knn', KNeighborsRegressor())
        ])

        # define grid search parameters
        params = {
            'knn__n_neighbors': [1, 2, 3 , 4 , 5],
            'knn__weights': ['uniform', 'distance']
        }

        svm_grid_search = GridSearchCV(
            svr,
            cv=LeaveOneOut(),
            param_grid=params,
            scoring=['neg_root_mean_squared_error', 'neg_mean_absolute_error'],
            refit='neg_root_mean_squared_error')

        svm_grid_search.fit(X, y)

        cv_results[target] = svm_grid_search.cv_results_

    return cv_results

knn_cv_results = knn_by_target(data)
```

[177]:
```
knn_best_scores, knn_best_params = results_by_target(data, knn_cv_results)
knn_best_scores
```

[177]:

	RMSE	RMSE_var	MAE	MAE_var
total_threatened	0.040616	0.001951	0.040616	0.001951
reptiles_threatened	0.087460	0.016539	0.087460	0.016539
mammals_threatened	0.048436	0.001609	0.048436	0.001609
amphibians_threatened	0.120676	0.021226	0.120676	0.021226

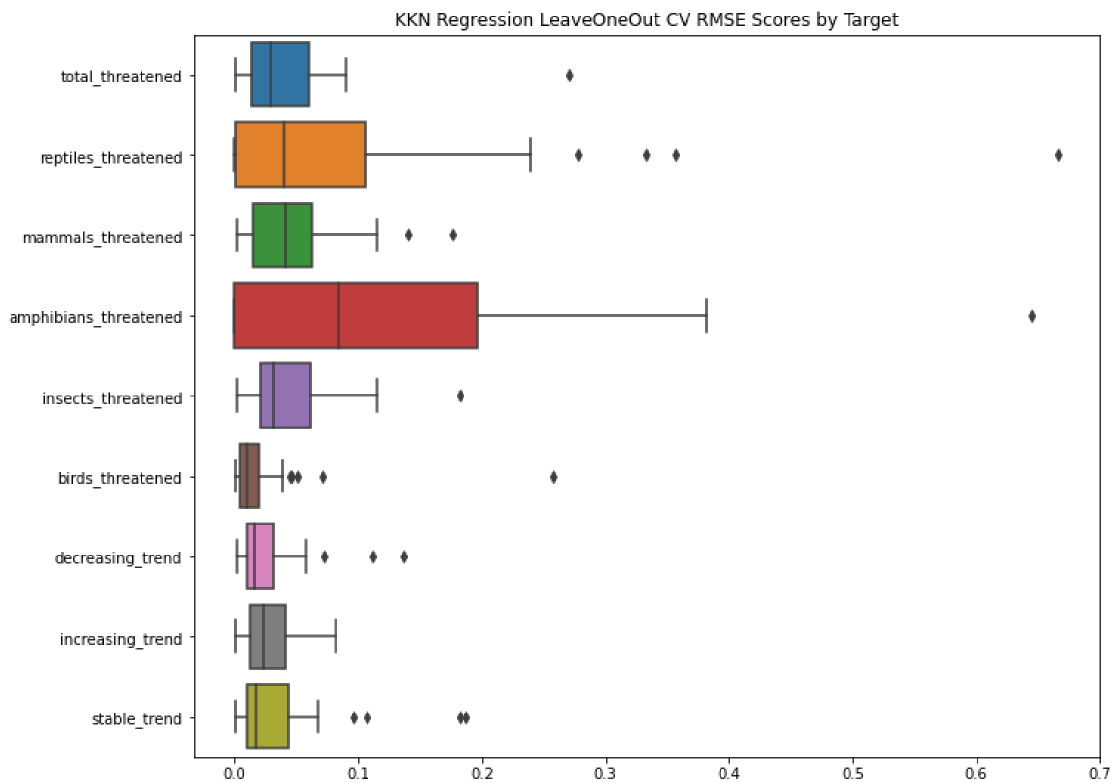
```
insects_threatened     0.045319  0.001476  0.045319  0.001476
birds_threatened       0.021192  0.001600  0.021192  0.001600
decreasing_trend       0.026372  0.000721  0.026372  0.000721
increasing_trend       0.029172  0.000523  0.029172  0.000523
stable_trend           0.033851  0.001694  0.033851  0.001694
```

[178]:
```
knn_best_params
```

[178]:
```
                                       neg_root_mean_squared_error  \
target                 index
total_threatened       n_neighbors                               3
                       weights                            distance
reptiles_threatened    n_neighbors                               1
                       weights                             uniform
mammals_threatened     n_neighbors                               5
                       weights                            distance
amphibians_threatened n_neighbors                                1
                       weights                            distance
insects_threatened     n_neighbors                               3
                       weights                            distance
birds_threatened       n_neighbors                               2
                       weights                            distance
decreasing_trend       n_neighbors                               3
                       weights                             uniform
increasing_trend       n_neighbors                               1
                       weights                             uniform
stable_trend           n_neighbors                               3
                       weights                            distance

                                       neg_mean_absolute_error
target                 index
total_threatened       n_neighbors                           3
                       weights                        distance
reptiles_threatened    n_neighbors                           1
                       weights                         uniform
mammals_threatened     n_neighbors                           5
                       weights                        distance
amphibians_threatened n_neighbors                            1
                       weights                        distance
insects_threatened     n_neighbors                           3
                       weights                        distance
birds_threatened       n_neighbors                           2
                       weights                        distance
decreasing_trend       n_neighbors                           3
                       weights                         uniform
increasing_trend       n_neighbors                           1
                       weights                         uniform
```

```
stable_trend          n_neighbors                    3
                      weights                 distance
```

[179]:
```
# get data
vis_knn_scores = extract_cv_scores(data, knn_cv_results,␣
 ↪'neg_root_mean_squared_error')

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))
ax.set_title('KKN Regression LeaveOneOut CV RMSE Scores by Target')

sns.boxplot(data=vis_knn_scores, orient='h')
plt.show()
```



6.3.3 Support Vector Regression

We select support vector regression mainly because they are: - Effective in high dimensional spaces. - Still effective in cases where number of dimensions is greater than the number of samples.

[180]:
```
def svr_by_target(data):
    cv_results = {}
```

```
    # iterate all tartget veriables
    target_columns = [col for col in data.columns if col.endswith('threatened')␣
 ↪or col.endswith('trend')]
    for target in target_columns:
        y = data[target]
        X = data.drop(columns=target_columns)

        # create pipeline for Model
        svr = Pipeline([
            ('scaling', StandardScaler()),
            ('svr', SVR())
        ])

        # define grid search parameters
        params = {
            'svr__C': [0.1, 0.3, 0.5, 0.8, 1, 2, 5],
            'svr__kernel': ['linear', 'poly', 'rbf', 'sigmoid']
        }

        svm_grid_search = GridSearchCV(
            svr,
            cv=LeaveOneOut(),
            param_grid=params,
            scoring=['neg_root_mean_squared_error', 'neg_mean_absolute_error'],
            refit='neg_root_mean_squared_error')

        svm_grid_search.fit(X, y)

        cv_results[target] = svm_grid_search.cv_results_

    return cv_results

svr_cv_results = svr_by_target(data)
```

[181]:
```
svr_best_scores, svr_best_params = results_by_target(data, svr_cv_results)
svr_best_scores
```

[181]:

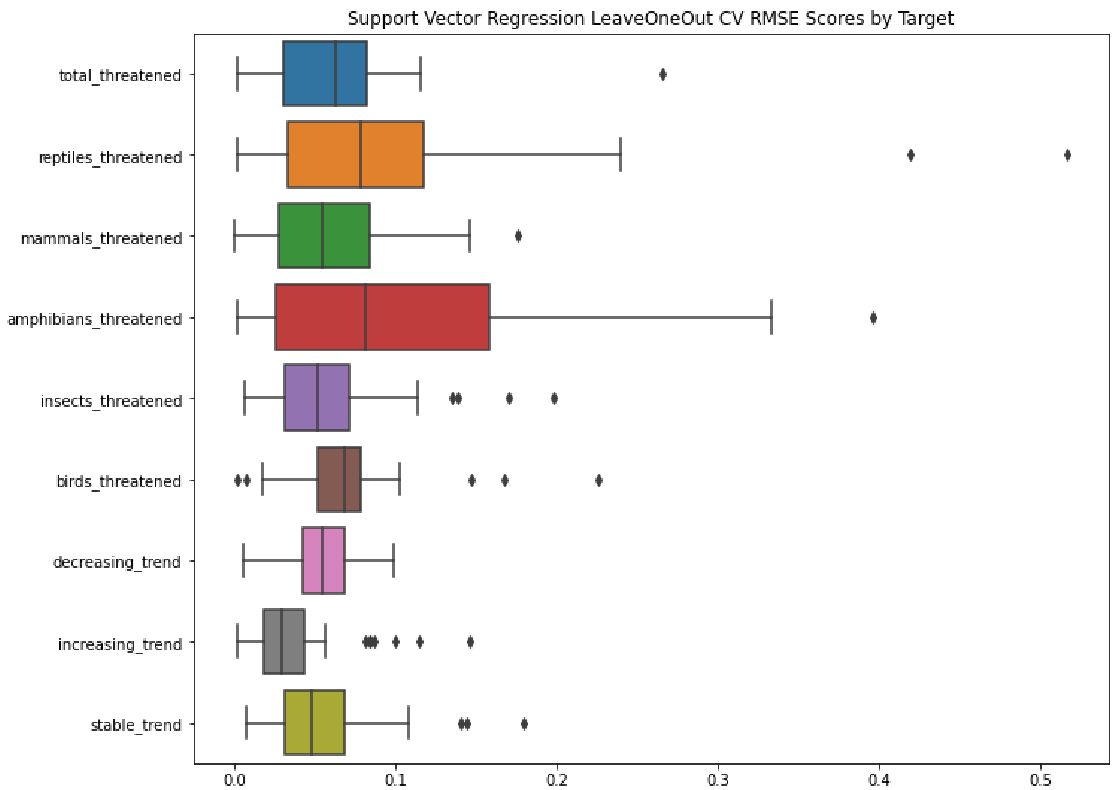
	RMSE	RMSE_var	MAE	MAE_var
total_threatened	0.059965	0.002021	0.059965	0.002021
reptiles_threatened	0.097390	0.009939	0.097390	0.009939
mammals_threatened	0.057326	0.001631	0.057326	0.001631
amphibians_threatened	0.102224	0.008497	0.102224	0.008497
insects_threatened	0.058216	0.001783	0.058216	0.001783
birds_threatened	0.071157	0.001529	0.071157	0.001529
decreasing_trend	0.052288	0.000523	0.052288	0.000523
increasing_trend	0.037430	0.001038	0.037430	0.001038
stable_trend	0.055964	0.001343	0.055964	0.001343

[182]:
```
svr_best_params
```

[182]:
```
                                          neg_root_mean_squared_error  \
target                    index
total_threatened          C                                       0.1
                          kernel                                 poly
reptiles_threatened       C                                       0.1
                          kernel                              sigmoid
mammals_threatened        C                                       0.1
                          kernel                               linear
amphibians_threatened C                                           0.1
                          kernel                               linear
insects_threatened        C                                       0.1
                          kernel                              sigmoid
birds_threatened          C                                       0.3
                          kernel                              sigmoid
decreasing_trend          C                                       0.1
                          kernel                               linear
increasing_trend          C                                       0.1
                          kernel                              sigmoid
stable_trend              C                                       0.1
                          kernel                                  rbf

                                          neg_mean_absolute_error
target                    index
total_threatened          C                                   0.1
                          kernel                             poly
reptiles_threatened       C                                   0.1
                          kernel                          sigmoid
mammals_threatened        C                                   0.1
                          kernel                           linear
amphibians_threatened C                                       0.1
                          kernel                           linear
insects_threatened        C                                   0.1
                          kernel                          sigmoid
birds_threatened          C                                   0.3
                          kernel                          sigmoid
decreasing_trend          C                                   0.1
                          kernel                           linear
increasing_trend          C                                   0.1
                          kernel                          sigmoid
stable_trend              C                                   0.1
                          kernel                              rbf
```

[183]:
```
# get data
vis_knn_scores = extract_cv_scores(data, svr_cv_results,␣
 ↪'neg_root_mean_squared_error')
```

```
# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))
ax.set_title('Support Vector Regression LeaveOneOut CV RMSE Scores by Target')

sns.boxplot(data=vis_knn_scores, orient='h')
plt.show()
```



6.3.4 Random Forest Regression

We perform random forest regression here to predict the outcome of our relative threatened species for all groups combined and for each group separately, as well as the trends. As a random forest is a tree partitioning algorithm it does by nature not need any scaling of the data beforehand. After the results of our correlation analysis we start with the full model here. As the number of samples in our remaining data frame is quite limited, we use leave one out cross validation to measure the performance of our regressor rather than a train/test split.Because of the high dimension of the data, we take the square root of the total number of features to pick for every split. We also try different values for the number of decision trees created.

[184]:
```
complete_data = pd.concat([
        ds_climate.set_index('Country'),
        ds_ghg.set_index('Country'),
```

```
        ds_land_cover.set_index('Country'),
        ds_protected_areas.set_index('Country'),
        ds_threatened_by_group.set_index('Country'),
        ds_trend_by_group],
        join='inner',
        axis=1)
```

[185]:
```
rf_data = complete_data.copy()
rf_data.shape
```

[185]: (42, 43)

[186]:
```
def rf_predict_threatened_relative(data):
    cv_results = {}

    # iterate all target variables
    columns_threatened = [col for col in data.columns if col.
 ↪endswith('threatened') or col.endswith('trend')]
    for target in columns_threatened:
        y = data[target]
        X = data.drop(columns=columns_threatened)

        # n_estimator(number of trees) is the hyperparameter that we try to␣
 ↪optimize here
        n_estimators = [int(x) for x in np.linspace(start = 50, stop = 500, num␣
 ↪= 10)]
        grid = random_grid = {'n_estimators': n_estimators}
        rf=RandomForestRegressor(max_features = 'sqrt', random_state=0)
        rf_grid = GridSearchCV(estimator = rf, param_grid = random_grid,  cv =␣
 ↪LeaveOneOut(),  scoring=['neg_root_mean_squared_error',␣
 ↪'neg_mean_absolute_error'],
            refit='neg_root_mean_squared_error')
        rf_grid.fit(X,y)
        cv_results[target] = rf_grid.cv_results_

    return cv_results
rf_cv_results = rf_predict_threatened_relative(complete_data)
```

[187]:
```
rf_best_scores, rf_best_params = results_by_target(complete_data, rf_cv_results)
rf_best_scores
```

[187]:

	RMSE	RMSE_var	MAE	MAE_var
total_threatened	0.035622	0.001573	0.035622	0.001573
reptiles_threatened	0.087534	0.010062	0.087534	0.010062
mammals_threatened	0.043758	0.001634	0.043758	0.001634
amphibians_threatened	0.112768	0.013741	0.112768	0.013741
insects_threatened	0.039387	0.001652	0.039387	0.001652

```
birds_threatened       0.021773  0.001189  0.021773  0.001189
decreasing_trend       0.025722  0.000521  0.025722  0.000521
increasing_trend       0.027723  0.000481  0.027723  0.000481
stable_trend           0.035605  0.001769  0.035605  0.001769
```

[188]:
```
rf_data
```

[188]:
```
                 temp_slope  gain_percentage  temp_difference       CH4  \
Country
Argentina          0.014949         0.500525         0.072246  1.828861
Australia          0.017804         0.969423         0.214683  4.382540
Austria            0.038525        24.241043         1.498389  0.728541
Belgium            0.028670        12.044550         1.167659  0.688260
Brazil             0.033281         3.703937         0.929317  1.683181
Canada             0.037969       -24.178355         1.583367  2.466998
Chile              0.014697         9.319080         0.761799  0.767192
Colombia           0.020083         3.383376         0.833692  0.946694
Costa Rica         0.018108         2.447034         0.604383  0.820945
Czech Republic     0.047428        20.586324         1.598151  1.237946
Denmark            0.036901         9.699569         0.800735  1.272634
Estonia            0.037562         4.866007         0.301582  0.845256
Finland            0.049607        43.855657         0.946397  0.823286
France             0.026566         9.395659         1.010464  0.844018
Germany            0.034688        13.837223         1.201279  0.634896
Greece             0.050600        14.083415         1.852533  0.940815
Hungary            0.057306        17.752956         1.710039  0.744502
Iceland            0.042999        27.490863         0.691358  1.786166
India              0.023121         3.484826         0.843200  0.334763
Indonesia          0.013825         2.671550         0.696717  0.711852
Ireland            0.006156         3.196774         0.298907  2.879337
Israel             0.058883         7.233207         1.416450  0.831843
Italy              0.038715        12.180835         1.425786  0.712204
Japan              0.022342         5.224242         0.589587  0.236113
Korea              0.013117        12.654444         1.061609  0.519825
Latvia             0.035525         6.075607         0.406742  0.899544
Lithuania          0.037384         8.398839         0.601233  1.093803
Luxembourg         0.029807        12.243670         1.112161  0.966622
Mexico             0.020896         4.217127         0.878617  1.171375
Netherlands        0.032691        13.518773         1.279099  1.004478
New Zealand        0.029854        12.086625         1.156237  7.017958
Norway             0.038105        40.227732         0.649496  0.904485
Poland             0.048945        17.945155         1.421874  1.269181
Portugal           0.021486         3.690444         0.559011  0.887454
Russia             0.052953       -21.630035         1.168926  2.740886
Slovak Republic    0.055742        22.755623         1.667737  0.815535
Slovenia           0.042709        18.825714         1.604571  0.935319
Spain              0.027007         7.044876         0.940025  0.849984
```

Sweden	0.034578	22.476843	0.538604	0.430511
Switzerland	0.033786	19.951503	1.186882	0.568525
United Kingdom	0.019055	8.485272	0.722540	0.781754
United States	0.035890	19.363436	1.465005	1.939243

Country	CO2	HFC	N2O	NF3	PFC \
Argentina	4.710241	0.014370	1.002114	-1.000000	0.003744
Australia	16.642911	0.479420	0.804808	-1.000000	0.009443
Austria	7.549433	0.207606	0.398982	0.001868	0.003680
Belgium	8.787278	0.391962	0.500037	0.000057	0.011516
Brazil	2.566549	-1.000000	0.894033	-1.000000	-1.000000
Canada	15.826302	0.338527	1.023899	0.000003	0.016758
Chile	4.837817	0.157948	0.369279	-1.000000	0.000000
Colombia	1.691487	0.039670	0.478937	-1.000000	-1.000000
Costa Rica	1.651245	0.125102	0.224534	-1.000000	-1.000000
Czech Republic	9.825615	0.351586	0.571438	0.000293	0.000125
Denmark	6.260186	0.101028	0.936264	-1.000000	0.000001
Estonia	13.397323	0.174761	0.690017	-1.000000	0.000037
Finland	8.312781	0.213589	0.864645	-1.000000	0.000328
France	5.054056	0.238381	0.602349	0.000183	0.010156
Germany	9.110170	0.126482	0.428374	0.000142	0.003495
Greece	6.693887	0.550778	0.399490	-1.000000	0.012616
Hungary	5.080930	0.139033	0.497430	-1.000000	0.000081
Iceland	10.417635	0.474101	0.866050	-1.000000	0.216726
India	1.278950	0.000014	0.092906	-1.000000	0.000016
Indonesia	2.490887	-1.000000	0.216795	-1.000000	-1.000000
Ireland	7.989144	0.226551	1.431682	0.000272	0.010265
Israel	7.303624	0.485538	0.218293	-1.000000	0.019626
Italy	5.760918	0.274235	0.292855	0.000366	0.027428
Japan	8.981805	0.371611	0.158174	0.002234	0.027576
Korea	12.634908	0.187555	0.271585	-1.000000	0.041266
Latvia	4.078149	0.123763	0.972284	-1.000000	-1.000000
Lithuania	4.879276	0.203895	1.054900	0.000010	-1.000000
Luxembourg	15.738992	0.111259	0.515067	-1.000000	-1.000000
Mexico	4.149015	0.103972	0.338982	-1.000000	0.000000
Netherlands	9.295129	0.095269	0.484523	-1.000000	0.009460
New Zealand	7.180502	0.371743	1.554079	-1.000000	0.014819
Norway	8.248936	0.159356	0.442235	-1.000000	0.027880
Poland	8.791412	0.108627	0.575480	-1.000000	0.000295
Portugal	5.006162	0.331768	0.312796	-1.000000	0.001855
Russia	11.705637	0.295945	0.594723	0.000001	0.018861
Slovak Republic	6.625547	0.129025	0.385281	-1.000000	0.001428
Slovenia	6.998789	0.141656	0.363993	-1.000000	0.007532
Spain	5.770099	0.097526	0.394015	-1.000000	0.002791
Sweden	4.104698	0.101708	0.442609	-1.000000	0.006080
Switzerland	4.333901	0.179021	0.338256	0.000059	0.004186

```
United Kingdom    5.732622  0.197676  0.289148  0.000009  0.003866
United States    16.581362  0.474912  1.328153  0.001922  0.014156

                          SF6  …  insects_threatened  birds_threatened  \
Country                        …
Argentina        4.164567e-05  …              0.0744            0.0519
Australia        9.143932e-03  …              0.1403            0.0716
Austria          4.324131e-02  …              0.0867            0.0426
Belgium          8.337265e-03  …              0.0544            0.0303
Brazil          -1.000000e+00  …              0.0807            0.0914
Canada           8.348774e-03  …              0.0712            0.0426
Chile            1.498683e-02  …              0.0870            0.0774
Colombia         3.286490e-03  …              0.0872            0.0637
Costa Rica       4.014706e-04  …              0.0677            0.0328
Czech Republic   6.639765e-03  …              0.1050            0.0314
Denmark          1.276383e-02  …              0.0952            0.0327
Estonia          1.937250e-03  …              0.0319            0.0354
Finland          3.631567e-03  …              0.0625            0.0417
France           6.134144e-03  …              0.1178            0.0475
Germany          4.668272e-02  …              0.1007            0.0354
Greece           4.608477e-04  …              0.2154            0.0519
Hungary          1.039805e-02  …              0.0925            0.0455
Iceland          9.242406e-03  …              0.0000            0.0642
India            8.123604e-08  …              0.0433            0.0776
Indonesia       -1.000000e+00  …              0.0767            0.0987
Ireland          8.424516e-03  …              0.0000            0.0431
Israel           1.048581e-02  …              0.0677            0.0464
Italy            7.388493e-03  …              0.1936            0.0506
Japan            1.615652e-02  …              0.1312            0.1119
Korea            1.289891e-01  …              0.0723            0.0914
Latvia           5.470716e-03  …              0.0648            0.0410
Lithuania        2.287670e-03  …              0.0360            0.0412
Luxembourg       1.678099e-02  …              0.0297            0.0160
Mexico           1.608978e-03  …              0.0966            0.0615
Netherlands      7.178082e-03  …              0.0388            0.0369
New Zealand      3.011974e-03  …              0.3043            0.2900
Norway           1.063684e-02  …              0.0700            0.0431
Poland           2.795215e-03  …              0.1000            0.0378
Portugal         2.306341e-03  …              0.2344            0.0514
Russia           9.065043e-03  …              0.0951            0.0848
Slovak Republic  1.724508e-03  …              0.0913            0.0408
Slovenia         7.646192e-03  …              0.0474            0.0345
Spain            4.854703e-03  …              0.1442            0.0548
Sweden           3.154627e-03  …              0.0876            0.0396
Switzerland      1.846527e-02  …              0.1188            0.0316
United Kingdom   8.216791e-03  …              0.0737            0.0412
United States    1.814418e-02  …              0.1540            0.1044
```

	reptiles_resident	mammals_resident	amphibians_resident	\
Country				
Argentina	True	True	True	
Australia	True	True	True	
Austria	True	True	True	
Belgium	True	True	True	
Brazil	True	True	True	
Canada	True	True	True	
Chile	True	True	True	
Colombia	True	True	True	
Costa Rica	True	True	True	
Czech Republic	True	True	True	
Denmark	True	True	True	
Estonia	True	True	True	
Finland	True	True	True	
France	True	True	True	
Germany	True	True	True	
Greece	True	True	True	
Hungary	True	True	True	
Iceland	False	True	False	
India	True	True	True	
Indonesia	True	True	True	
Ireland	True	True	True	
Israel	True	True	True	
Italy	True	True	True	
Japan	True	True	True	
Korea	True	True	True	
Latvia	True	True	True	
Lithuania	True	True	True	
Luxembourg	True	True	True	
Mexico	True	True	True	
Netherlands	True	True	True	
New Zealand	True	True	True	
Norway	True	True	True	
Poland	True	True	True	
Portugal	True	True	True	
Russia	True	True	True	
Slovak Republic	True	True	True	
Slovenia	True	True	True	
Spain	True	True	True	
Sweden	True	True	True	
Switzerland	True	True	True	
United Kingdom	True	True	True	
United States	True	True	True	

	insects_resident	birds_resident	decreasing_trend	\

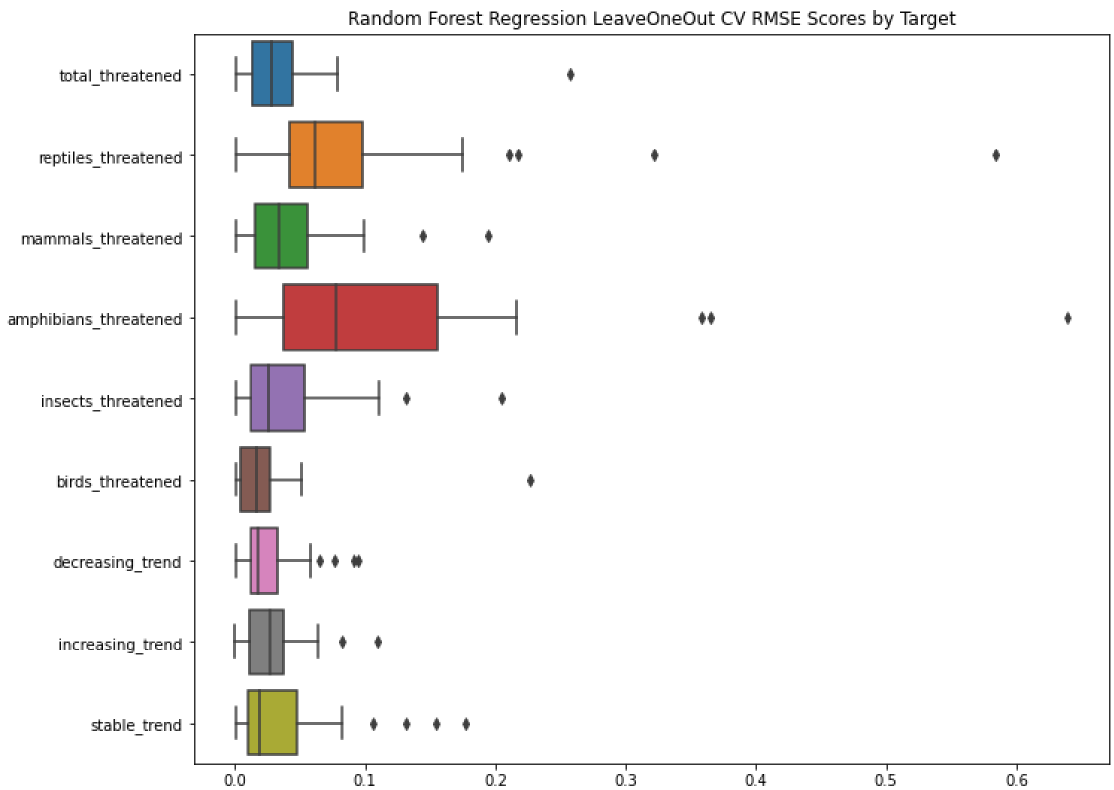
Country			
Argentina	True	True	0.274619
Australia	True	True	0.195591
Austria	True	True	0.342246
Belgium	True	True	0.332008
Brazil	True	True	0.315529
Canada	True	True	0.232601
Chile	True	True	0.302956
Colombia	True	True	0.372725
Costa Rica	True	True	0.327454
Czech Republic	True	True	0.344771
Denmark	True	True	0.329004
Estonia	True	True	0.371429
Finland	True	True	0.361798
France	True	True	0.303371
Germany	True	True	0.333795
Greece	True	True	0.289881
Hungary	True	True	0.346709
Iceland	True	True	0.350365
India	True	True	0.294961
Indonesia	True	True	0.351745
Ireland	True	True	0.355987
Israel	True	True	0.313131
Italy	True	True	0.300248
Japan	True	True	0.354331
Korea	True	True	0.367754
Latvia	True	True	0.356044
Lithuania	True	True	0.357968
Luxembourg	True	True	0.342618
Mexico	True	True	0.319672
Netherlands	True	True	0.331325
New Zealand	True	True	0.357542
Norway	True	True	0.336364
Poland	True	True	0.346154
Portugal	True	True	0.330786
Russia	True	True	0.326982
Slovak Republic	True	True	0.347484
Slovenia	True	True	0.340491
Spain	True	True	0.269043
Sweden	True	True	0.348000
Switzerland	True	True	0.326241
United Kingdom	True	True	0.336283
United States	True	True	0.215966

	increasing_trend	stable_trend
Country		
Argentina	0.043458	0.396209

```
Australia               0.029391      0.407978
Austria                 0.108289      0.295455
Belgium                 0.145129      0.298211
Brazil                  0.028848      0.291864
Canada                  0.158425      0.479853
Chile                   0.070197      0.407635
Colombia                0.035376      0.329915
Costa Rica              0.077014      0.366255
Czech Republic          0.122549      0.290850
Denmark                 0.145022      0.307359
Estonia                 0.142857      0.323810
Finland                 0.139326      0.319101
France                  0.099251      0.275281
Germany                 0.120499      0.282548
Greece                  0.083865      0.232452
Hungary                 0.126806      0.277689
Iceland                 0.233577      0.175182
India                   0.036410      0.275837
Indonesia               0.015253      0.253293
Ireland                 0.155340      0.268608
Israel                  0.134199      0.314574
Italy                   0.085194      0.265509
Japan                   0.071991      0.321710
Korea                   0.086957      0.324275
Latvia                  0.149451      0.312088
Lithuania               0.157044      0.311778
Luxembourg              0.158774      0.342618
Mexico                  0.069987      0.394704
Netherlands             0.156627      0.303213
New Zealand             0.142458      0.231844
Norway                  0.154545      0.290909
Poland                  0.125418      0.302676
Portugal                0.099345      0.256550
Russia                  0.092988      0.317835
Slovak Republic         0.125786      0.286164
Slovenia                0.131902      0.305215
Spain                   0.078966      0.233403
Sweden                  0.142000      0.300000
Switzerland             0.107801      0.302128
United Kingdom          0.150442      0.298673
United States           0.090243      0.485924

[42 rows x 43 columns]
```

[189]:
```
# get data
vis_rf_scores = extract_cv_scores(complete_data, rf_cv_results,␣
↪'neg_root_mean_squared_error')
```

```
# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))
ax.set_title('Random Forest Regression LeaveOneOut CV RMSE Scores by Target')

sns.boxplot(data=vis_rf_scores, orient='h')
plt.show()
```



6.3.5 Value Ranges

As performance metric we used the RMSE and the MAE here. To get an understanding of what these values mean and how they can be interpreted, we take a look at our data again and look at the value ranges to then interpret the results.

[190]:
```
target_columns = [col for col in data.columns if col.endswith('threatened') or␣
↪col.endswith('trend')]
complete_data[target_columns].describe()
```

[190]:

	total_threatened	reptiles_threatened	mammals_threatened	\
count	42.000000	42.000000	42.000000	
mean	0.090214	0.113638	0.101226	
std	0.059176	0.150635	0.069002	

```
min              0.016700              0.000000             0.000000
25%              0.050675              0.000000             0.051750
50%              0.071400              0.077000             0.089700
75%              0.124150              0.158400             0.142075
max              0.357500              0.750000             0.294000

       amphibians_threatened  insects_threatened  birds_threatened  \
count              42.000000           42.000000         42.000000
mean                0.127510            0.093860          0.058579
std                 0.179915            0.059323          0.042883
min                 0.000000            0.000000          0.016000
25%                 0.000000            0.065525          0.038250
50%                 0.032150            0.086850          0.044300
75%                 0.209625            0.103925          0.064075
max                 0.750000            0.304300          0.290000

       decreasing_trend  increasing_trend  stable_trend
count         42.000000         42.000000     42.000000
mean           0.325095          0.110214      0.310885
std            0.039627          0.046089      0.060754
min            0.195591          0.015253      0.175182
25%            0.313731          0.080191      0.278904
50%            0.335039          0.121524      0.302402
75%            0.349774          0.144481      0.323285
max            0.372725          0.233577      0.485924
```

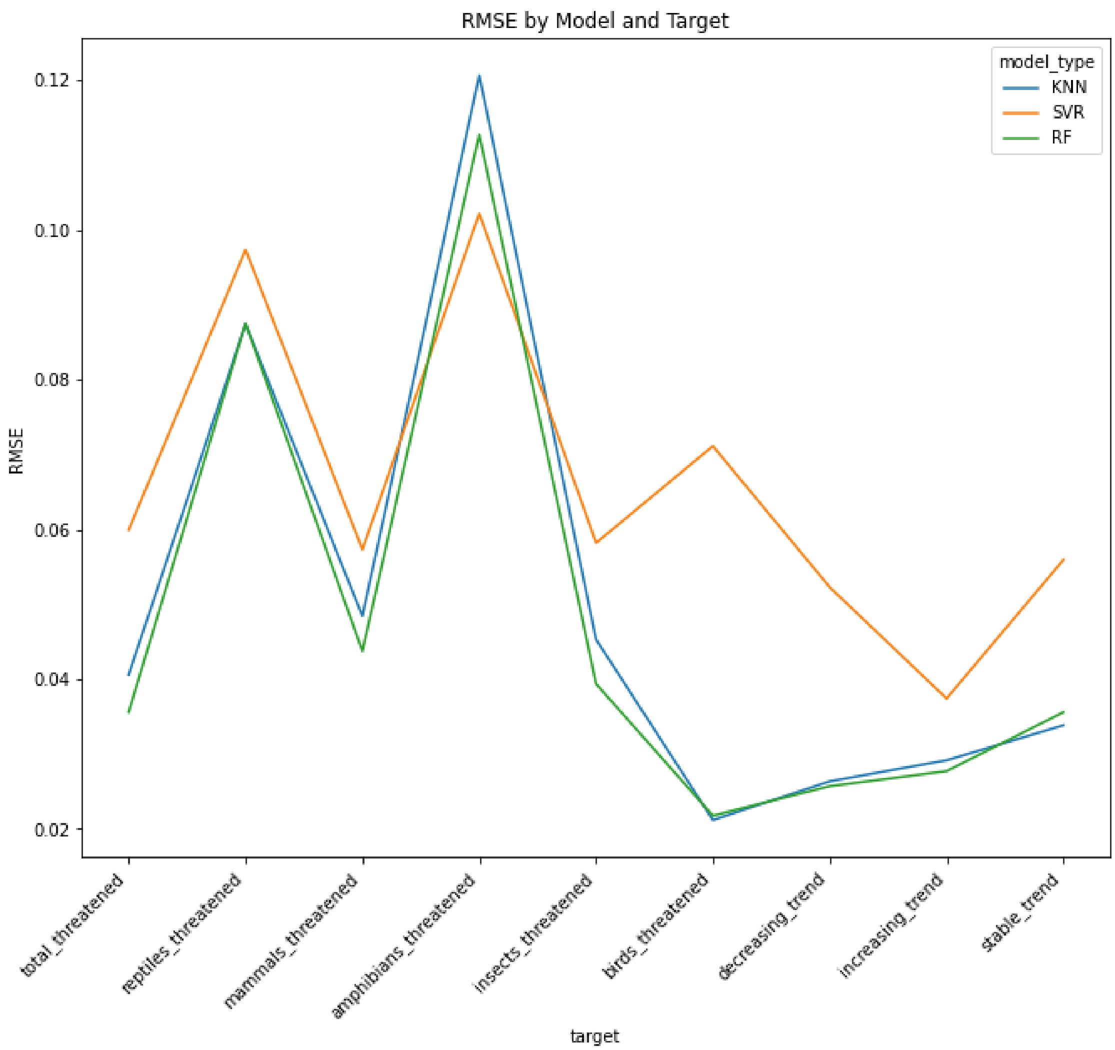
The values for the threatened species (all groups) range from 1.6%-35.8%, while the mean is at 9%. for the taxonomic groups of reptiles and amphibians we have quite high maximum values of 75%, but those are rather outliers. For the decreasing trend the value range is from 19.6%-37.3%, for the increasing 1.5%-23.3% and for the stable trends it is from 17.5%-48.6%.

6.3.6 Compare Models

To interpret our results after looking at the value ranges, we show the different scores for the RMSE across targets and models below.

[191]:
```
# prepare data for visualization
vis_knn = knn_best_scores.reset_index().rename(columns={'index': 'target'})
vis_knn['model_type'] = 'KNN'
vis_svr = svr_best_scores.reset_index().rename(columns={'index': 'target'})
vis_svr['model_type'] = 'SVR'
vis_rf = rf_best_scores.reset_index().rename(columns={'index': 'target'})
vis_rf['model_type'] = 'RF'
df =pd.concat([vis_knn, vis_svr, vis_rf])

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))
ax.set_title('RMSE by Model and Target')
```

```python
# create heatmap
sns.lineplot(data=df, x='target', y='RMSE', hue='model_type')
plt.xticks(rotation=45, ha='right')
plt.show()
```



Looking at the relative number of threatened species, we have our lowest RMSE score for the combination of all groups with 0.03 obtained by our random forest regressor. This means that we are on average 3% off with our prediction of the best model here. For the separate groups we can see highly different errors with the highest error across all models for the amphibians with minimum RMSE of 0.1 by our SVR. While the prediction for the total number of threatened species is overall okay looking at the range of the values, the performance for the amphibians is not so meaningful. It is also surprising that we obtained the best values for the combined groups, while predicting speficic groups was rather difficult. But the interpretation of this is limited by the fact, that the prediction with the chosen characteristics are all in all not optimal. Regarding the prediction of relative trends the results are relatively for all different implications of trends with RMSEs that are similiar for decreasing, stable and increasing (~0.03). Regarding the performance of our different models, we can see relatively similar values across all our targets for the kNN and the random forest regressors, while overall the SVR performed the worst, but it got the best predictions on amphibians, which are our overall worst results.

6.4 Conclusion

Although we integrated data from various sources and created features corresponding to environmental factors for each country we were not able to reliably predict any of our target variables. The main problem in our view is that species do not care for country borders. Thus, data on a single country cannot tell us much about the status of a species.

6.5 Task Sharing

Ahmadou Wagne

- Analysis and Preparation of Land Cover Data
- Correlation Analysis of Relative Threatened Species
- Random Forest Regression on Multiple Targets (Trends, Threatened Species)
- Interpretation of Results for Mentioned Targets (+Comparison of KNN, SVR and RF)

Markus Kiesel

- Web Scraping of IUCN Red List Data and Preparation of Data
- Protected Area by Management Objective Data Preparation
- KNN and SVM Regression on Multiple Targets (Trends, Threatened Species)

Matthias Hofmaier

- Assessment and Analysis of World Bank Climate Data, Feature Extraction
- Data Merging and Correlation Analysis
- KNN, SVM and RF Regression for Trends by Group and Country

Michael Hermann-Hubler

- Data Set Greenhouse Gases
- Data Analysis Missing Trends
- Combining all Notebooks into one Notebook

[ ]:
```

```