The ethics of artificial intelligence: What talk data ?

Author : GARBA Moussa

Problem Statement

It is necessary to explore the full ethical, social and legal aspects of AI systems if we are to avoid unintended, negative consequences and risks arising from the implementation of AI in society.

software-based AI and intelligent robots (i.e. robots with an embedded AI) when exploring ethical issues.

Artificial intelligence refers to systems that can be designed to take cues from their environment and, based on those inputs, proceed to solve problems, assess risks, make predictions, and take actions. In the era predating powerful computers and big data, such systems were programmed by humans and followed rules of human invention, but advances in technology have led to the development of new approaches.

Objective

Objective: Exploring ethical data and finding homogeneous subgroups such that variables in the same group (clusters) are more similar to each other than the others. Based on this clustering, we can assess the global ethic index of Issuer.

- Aim 1: Clustering for Classification
- Aim 2: assessment the global ethic index of Issuer

Data

Who are the main producers?

What are they? What are the principles?

Navigate in this intense production

Open data available on Google Sheets – Licence Creative Commons BY-NC-SA

Digital Policies Frameworks Database Data Set(UCI Repository: https://docs.google.com/spreadsheets/d/1mU2brATV_fgd5MRGfT2ASOFepAI1pivwhGm0VCT2 (https://docs.google.com/spreadsheets/d/1mU2brATV_fgd5MRGfT2ASOFepAI1pivwhGm0VCT are used for this analysis.

Variables

Somes Attribute Information(Categorical & Numerical):

Categorical variable

- Issuer('Berkman Klein Center (University of Harvard)','Cyberjustice Laboratory', 'ETH Zurich','Fraunhofer, Institute for Intelligent Analysis and Information Systems IAIS', 'Handelsblatt Research Institute')
- Reference(Principled Artificial Intelligence','ACT Project - Projet AJC (Autonomisation des acteurs judiciaires par la Cyberjustice)','AI, the global landscape of ethics guidelines')
- Type ('Meta-analysis', 'Research project', 'Academic paper','Report/Study', 'Principles/Guidelines/Charters', 'Policy paper','Non binding instrument', 'Parliamentary proceeding','Binding instrument')
- Link('https://ssrn.com/abstract=3518482' (https://ssrn.com/abstract=3518482'), 'https://www.ajcact.org' (https://www.ajcact.org'), 'https://arxiv.org/ftp/arxiv/papers/1906/1906.11668.pdf','https://arxiv.org/abs/1809.03400' (https://arxiv.org/ftp/arxiv/papers/1906/1906.11668.pdf','https://arxiv.org/abs/1809.03400')
- Origin(Academia', 'Civil Society', 'International Organisation','Multistakeholder', 'National Authorities', 'Private Sector','Professional association', 'Think Tank')
- Source('United States', 'Canada', 'Switzerland', 'Germany', 'Slovenia','United Kingdom', 'Italy', 'China', 'Australia', 'Netherlands','Austria', 'France', 'Denmark', 'Sweden', 'Lithuania','EU - Article 29 Working Party', 'Council of Europe','EU - Council of the European Union', 'EU - European Union')
- CoE MS('No', 'Yes')
- Year (2018, 2019, 2020, 2015, 2017, 2014, 2011, 2016, 2010)
- Comments('Updated', 'New')
- Update('Data updated on 02 January 2021')

Numerical variable

- fundamental rights
- solidarity
- sustainability
- transparency
- explainability
- fairness
- freedom
- .....

[1]:

```
# This Python 3 environment comes with many helpful analytics libraries i
# It is defined by the kaggle/python Docker image: https://github.com/kag
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) wil

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) th
# You can also write temporary files to /kaggle/temp/, but they won't be
```

```
/kaggle/input/digital-policiers-frameworks/Digital Policies Frameworks.xlsx
/kaggle/input/digital-policies-framework-database/Digital_Policies_Frameworks
_Database.csv
```

[2]:

```
from sklearn.feature_extraction.text import CountVectorizer
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/ imp
from nltk.corpus import stopwords

from tqdm import tqdm
```

```
import os
import nltk
import spacy
import random
from spacy.util import compounding
from spacy.util import minibatch
import string

#Importing all the needed libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly as plt
from matplotlib.pyplot import xticks

from matplotlib import *
import sys
from pylab import *

%matplotlib inline
plt.rcParams['figure.figsize']=10,6
plt.rcParams['axes.grid']=True
plt.gray()

from io import BytesIO

import requests

## https://python-graph-gallery.com/wordcloud/


pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)


use_cuda = True
pd.set_option('display.max_columns', None)
```

```
<Figure size 720x432 with 0 Axes>
```

[3]:
```
import pandas as pd

#r = requests.get('https://docs.google.com/spreadsheets/d/1mU2brATV_fgd5M
#data = r.content
```

*Import dataset from Digital Policies Frameworks *

[4]:
```
### Import data from externe source
## link to datset source : https://docs.google.com/spreadsheets/d/1mU2brA
## https://inventory.algorithmwatch.org/
## https://www.europarl.europa.eu/RegData/etudes/STUD/2020/634452/EPRS_ST
## https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3518482
## https://www.nature.com/articles/s42256-019-0088-2
## https://blog.einstein.ai/frameworks-tool-kits-principles-and-oaths-oh-
## https://fra.europa.eu/en/project/2018/artificial-intelligence-big-data
## https://duckduckgo.com/

#dfs = pd.read_excel("/kaggle/input/digital-policiers-frameworks/Digital

dfs = pd.ExcelFile('/kaggle/input/digital-policiers-frameworks/Digital Po

## Data sources description
Database = pd.read_excel(dfs, 'Database')
```

[5]:
```
nrow, ncol = Database.shape
nrow, ncol
```

Out[5]: (405, 41)

[6]:
```
Database.head()
```

Out[6]:

	Issuer	Reference	Type	Link	Origin
0	Berkman Klein Center (University of Harvard)	Principled Artificial Intelligence	Meta-analysis	https://ssrn.com/abstract=3518482	Academia
1	Cyberjustice Laboratory	ACT Project - Projet AJC (Autonomisation des a...	Research project	https://www.ajcact.org	Academia
2	ETH Zurich	AI, the global landscape of ethics guidelines	Meta-analysis	https://arxiv.org/ftp/arxiv/papers/1906/1906.1...	Academia
3	ETH Zurich	A Moral Framework for Understanding of Fair ML...	Academic paper	https://arxiv.org/abs/1809.03400	Academia
4	Fraunhofer Institute for Intelligent Analysis ...	Trustworthy Use of Artificial Intelligence	Report/Study	https://www.iais.fraunhofer.de/content/dam/iai...	Academia

[7]:
```
#Database['fundamental rights'].unique()
```

[8]:
```
for col in Database.columns:
    print(col)
```

```
Issuer
Reference
Type
Link
Origin
Source
CoE MS
Year
Comments
Update
fundamental rights
human agency
human rights
non discrimination
non maleficence
rule of law
sustainable development
well being
accountability
autonomy
beneficence
democracy
dignity
diversity
explainability
fairness
freedom
inclusive
justice
liability
literacy
oversight
privacy
responsibility
robustness
safe
solidarity
sustainability
transparency
trust
trustworthy
```

Exploratory Data Analysis

Data Inspection

[9]:
```
#Define missing data function to identify the total number of missing dat
def missing_data(data):
    total = data.isnull().sum()
    percent = (data.isnull().sum()/data.isnull().count()*100)
    tt = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    types = []
    for col in data.columns:
        dtype = str(data[col].dtypes)
        types.append(dtype)
    tt['Types'] = types
    return(np.transpose(tt))
```

[10]:
```
missing_data(Database)
```

Out[10]:

	Issuer	Reference	Type	Link	Origin	Source	CoE MS	Year	Comments	Update	funda
Total	0	0	0	0	0	0	0	0	383	0	
Percent	0	0	0	0	0	0	0	0	94.5679	0	6
Types	object	object	object	object	object	object	object	int64	object	object	

[11]:
```
Database[Database.duplicated()== True]
```

Out[11]:

	Issuer	Reference	Type	Link	Origin	S
118	European Parliament	Comprehensive European industrial policy on ar...	Policy paper	https://oeil.secure.europarl.europa.eu/oeil/po...	International Organisation	Eur Parli

[12]:
```
Database.shape
```

Out[12]: (405, 41)

[13]:
```
Database.describe()
```

Out[13]:

	Year	fundamental rights	human agency	human rights	non discrimination	non maleficence	rule of law
count	405.000000	148.000000	148.000000	148.000000	148.000000	148.000000	148.000000
mean	2018.276543	0.000196	0.000017	0.001288	0.000113	0.000010	0.000143
std	1.284935	0.000579	0.000078	0.003047	0.000325	0.000072	0.000619
min	2010.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2018.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	2018.000000	0.000000	0.000000	0.000050	0.000000	0.000000	0.000000
75%	2019.000000	0.000026	0.000000	0.000946	0.000000	0.000000	0.000000
max	2020.000000	0.004255	0.000639	0.018059	0.002106	0.000745	0.006024

[14]:
```
Database.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 405 entries, 0 to 404
Data columns (total 41 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Issuer                  405 non-null    object 
 1   Reference               405 non-null    object 
 2   Type                    405 non-null    object 
 3   Link                    405 non-null    object 
 4   Origin                  405 non-null    object 
 5   Source                  405 non-null    object 
 6   CoE MS                  405 non-null    object 
 7   Year                    405 non-null    int64  
 8   Comments                22 non-null     object 
 9   Update                  405 non-null    object 
 10  fundamental rights      148 non-null    float64
 11  human agency            148 non-null    float64
 12  human rights            148 non-null    float64
 13  non discrimination      148 non-null    float64
 14  non maleficence         148 non-null    float64
 15  rule of law             148 non-null    float64
 16  sustainable development  148 non-null    float64
 17  well being              148 non-null    float64
 18  accountability          148 non-null    float64
 19  autonomy                148 non-null    float64
 20  beneficence             148 non-null    float64
 21  democracy               148 non-null    float64
 22  dignity                 148 non-null    float64
 23  diversity               148 non-null    float64
 24  explainability          148 non-null    float64
 25  fairness                148 non-null    float64
 26  freedom                 148 non-null    float64
 27  inclusive               148 non-null    float64
 28  justice                 148 non-null    float64
 29  liability               148 non-null    float64
 30  literacy                148 non-null    float64
 31  oversight               148 non-null    float64
 32  privacy                 148 non-null    float64
 33  responsibility          148 non-null    float64
 34  robustness              148 non-null    float64
 35  safe                    148 non-null    float64
 36  solidarity              148 non-null    float64
 37  sustainability          148 non-null    float64
 38  transparency            148 non-null    float64
 39  trust                   148 non-null    float64
 40  trustworthy             148 non-null    float64
dtypes: float64(31), int64(1), object(9)
memory usage: 129.9+ KB
```

[15]:
```
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from wordcloud import WordCloud

stopWords = stopwords.words('english')
tokenizer = RegexpTokenizer(r'\w+')

def get_wordcloud(series): #simple function to tokenize and plot a said c
    word_cloud = ''

    for job in series:
        tokens = tokenizer.tokenize(job)
        for token in tokens:
            if token not in stopWords:
                word_cloud += ''.join(token) + ' '

    #wordcloud = WordCloud(height=800,margin=1,max_words=500, colormap='S
    #wordcloud = WordCloud( width = 3000, height = 2000, random_state=1, 

    wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,
                          background_color='black', colormap='Set2', coll

    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)
    #plt.savefig('Plotly-World_Cloud.png')
```

Univariate Analysis

Categorical Variables

Grouped bar charts

Stacked bar charts

Issuer

[16]:
```
Database['Issuer'].unique()
```

Out[16]:
```
array(['Berkman Klein Center (University of Harvard)',
       'Cyberjustice Laboratory', 'ETH Zurich',
       'Fraunhofer Institute for Intelligent Analysis and Information System
s IAIS',
       'Handelsblatt Research Institute',
       'Institute of Criminology at the Faculty of Law of Ljubljana',
       'MIT', 'Montreal AI Ethics Institute',
       'National Academies of Science, Engineering & Medecine',
       'Oxford Internet Institute', 'PLOS Computational Biology',
       'Polytechnic University of Turin',
       'Research Center for Brain-inspired Intelligence, Institute of Automa
tion, Chinese Academy of Sciences',
       'Royal Australian and New Zealand College of Radiologists',
       'Special Interest Group on Artificial Intelligence (SIGAI), ICT Platf
orm Netherlands (IPN)',
       'Stanford Law School', 'Technical University of Delft',
       'The Alan Turing Institute', 'The Royal Society', 'TU Wien',
```

[17]:
```
len(Database['Issuer'].unique())
```

Out[17]:
```
298
```

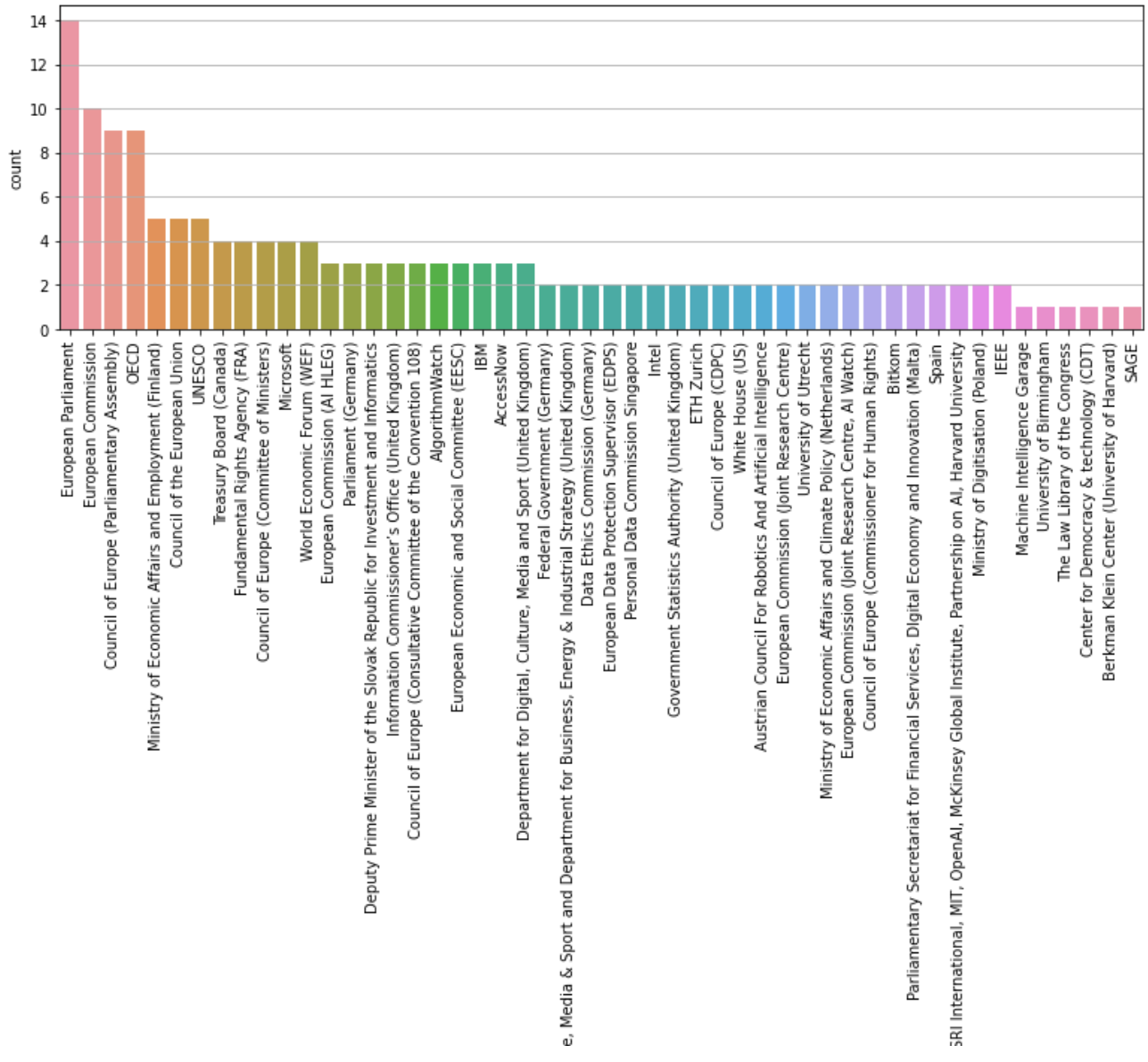
[18]:
```
Database['Issuer'].describe()
```

Out[18]:
```
count                    405
unique                   298
top       European Parliament
freq                      14
Name: Issuer, dtype: object
```

[19]:
```
Database['Issuer'].nunique()
```

Out[19]: 298

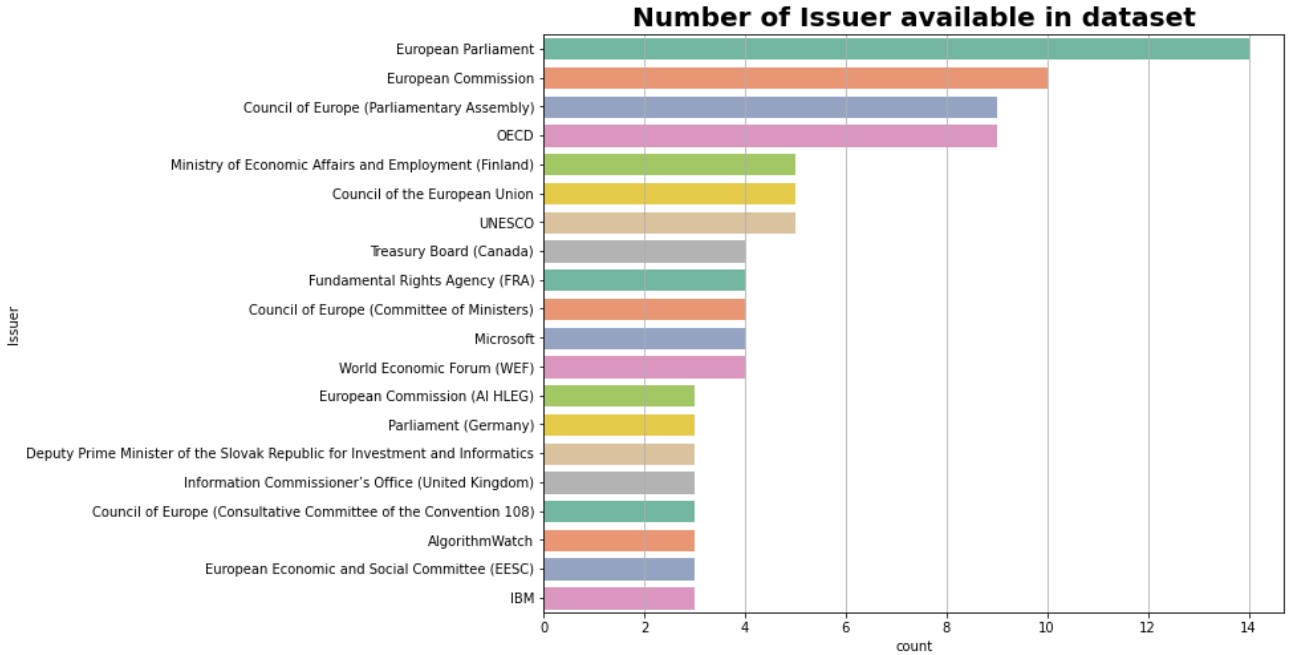
[20]:
```
#plt.figure(figsize=(13, 4))
#http://stackoverflow.com/questions/32891211/limit-the-number-of-groups-s
#sns.countplot(Database.Issuer.dropna(), order = Database.Issuer.value_co
#sns.violinplot(data=df, x='', y='')


plt.figure(figsize=(13, 4))
sns.countplot(Database.Issuer.dropna(), order = Database.Issuer.value_cou
plt.xticks(rotation=90);
```



OK,this is too much to inspect, let's see only the top 50. But the shape is very nice\smooth, looks like an exponential decay.

[21]:

```
#Source

ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Number of Issuer available in dataset', fontweight='bold', fon
ax = sns.countplot(y = 'Issuer', data = Database, order = Database['Issue
plt.savefig('Issuer.png')
```

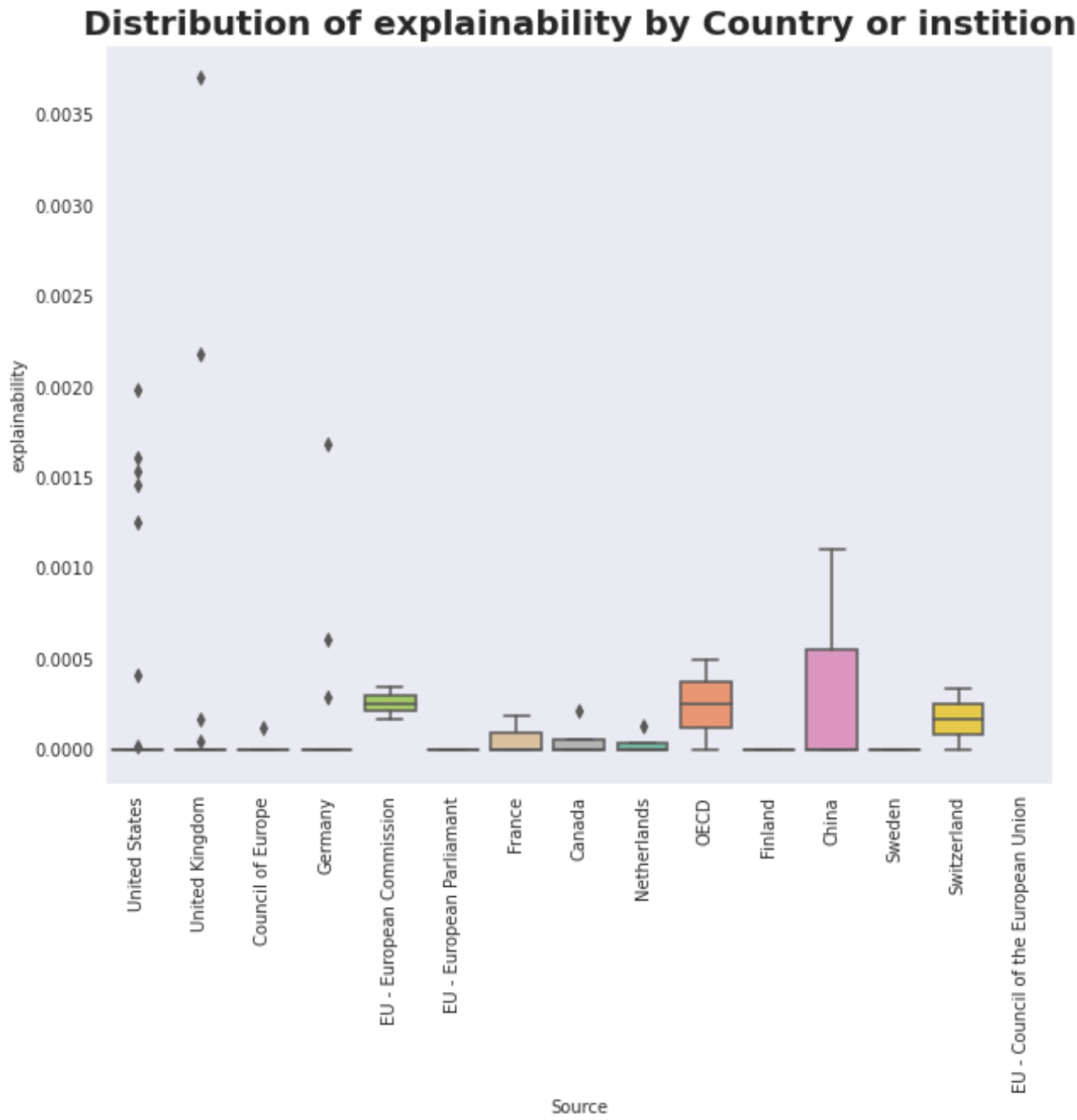


[22]:
```
Database['Issuer'].isnull().sum()/ nrow

#Let's multiple by 100 and keep only 1 decimal places
(Database['Issuer'].isnull().sum()/ nrow).round(3)*100
```

Out[22]: 0.0

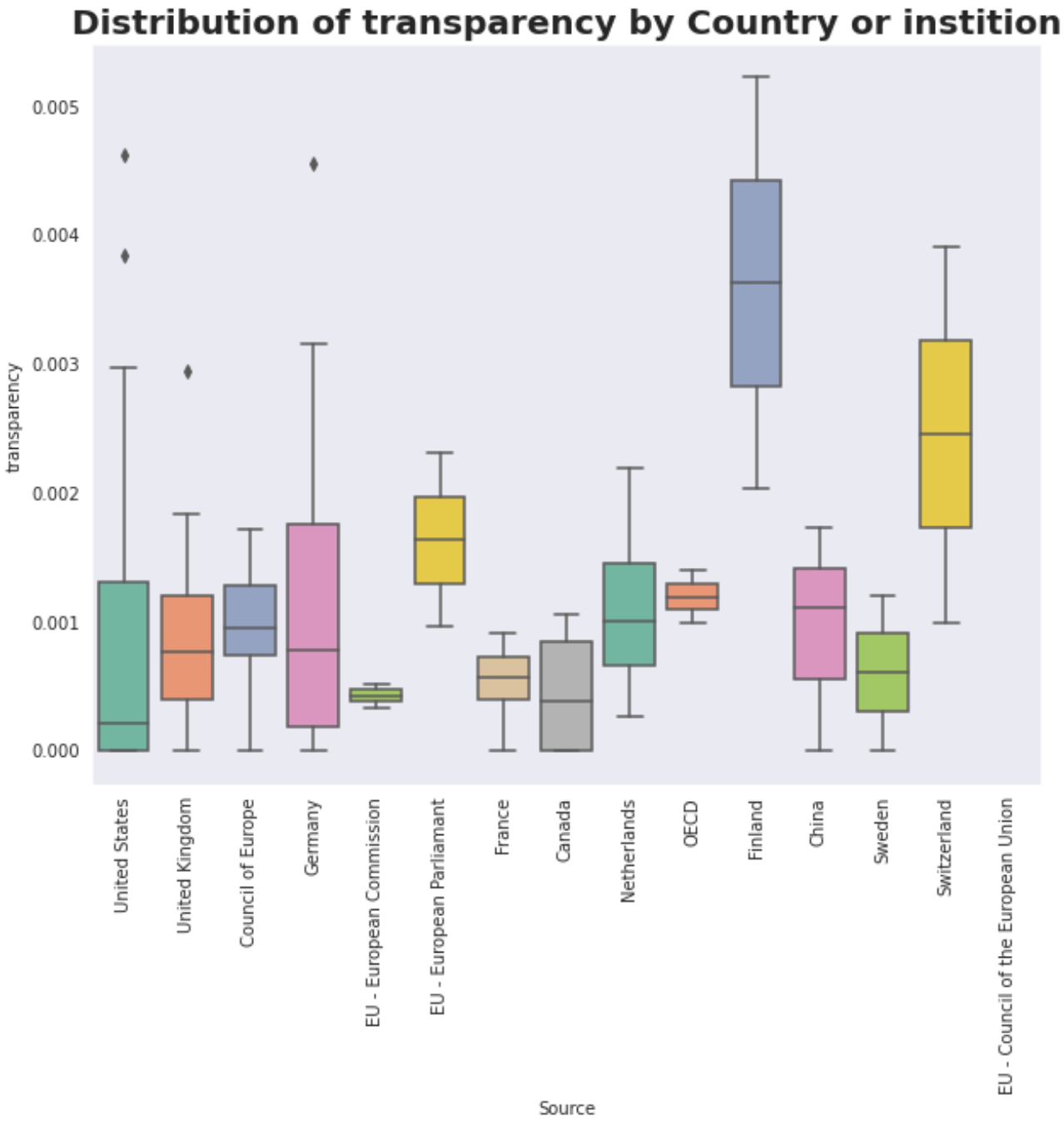
[23]:

```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Distribution of explainability by Country or institution', fontwe

ax = sns.boxplot(x='Source',y='explainability',data=Database,order = Datab
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.savefig('histor.png')
```



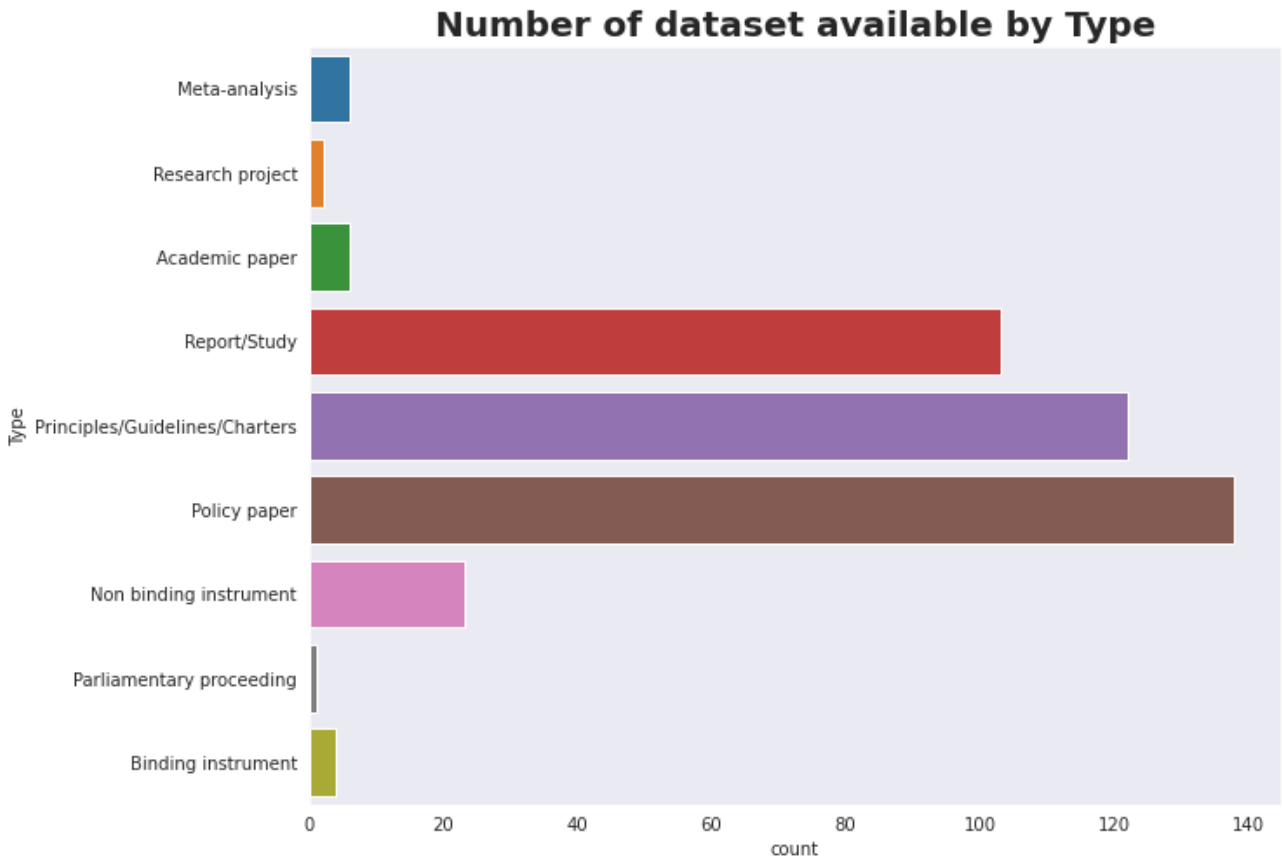
[24]:

```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Distribution of transparency by Country or institution', fontwei

ax = sns.boxplot(x='Source',y='transparency',data=Database,order = Databa
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.savefig('histo2.png')
```



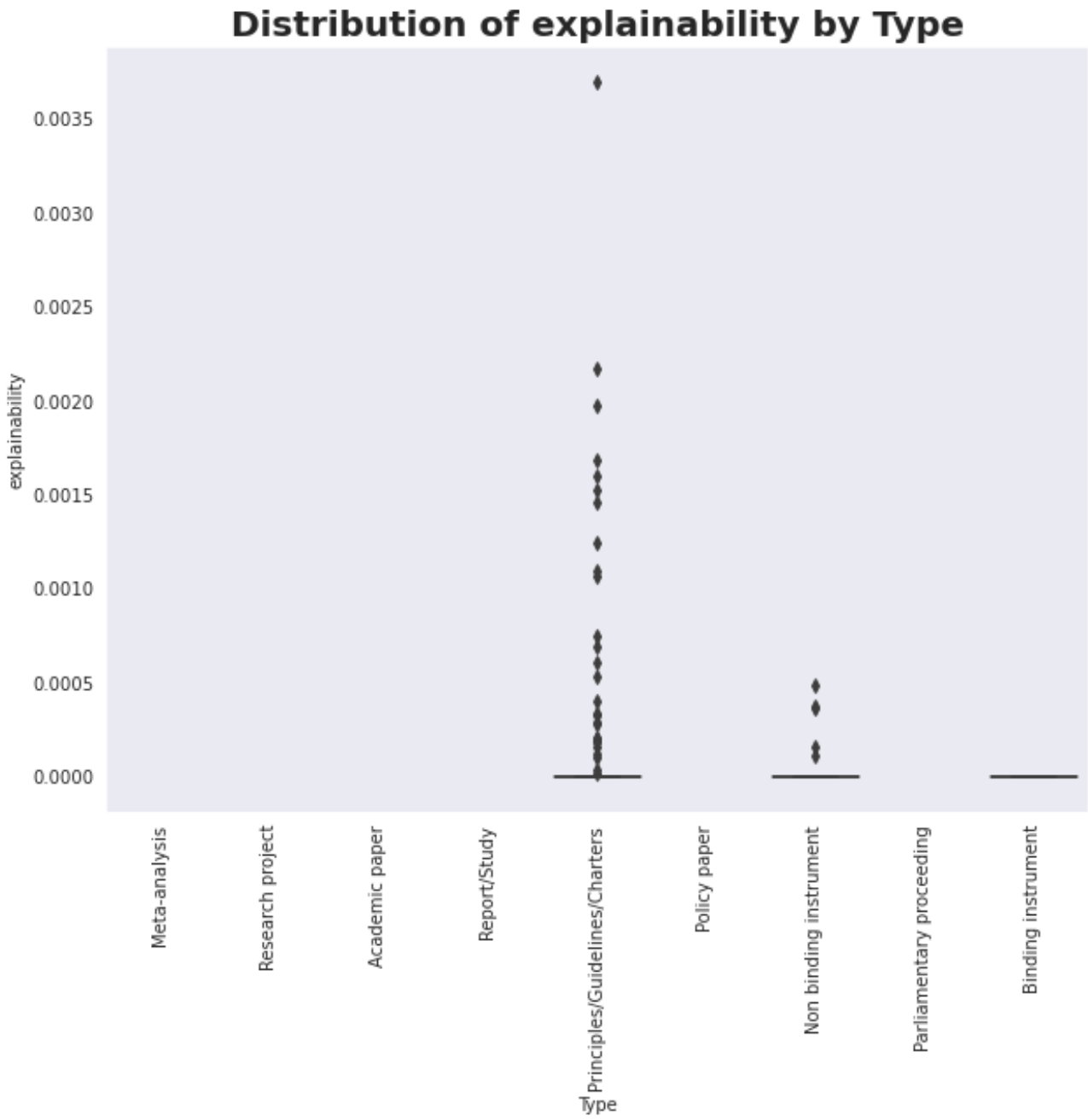
[25]:

```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Number of dataset available by Type', fontweight='bold', fonts
ax = sns.countplot(y = 'Type', data = Database, palette='tab10')
plt.savefig('Type.png')
```



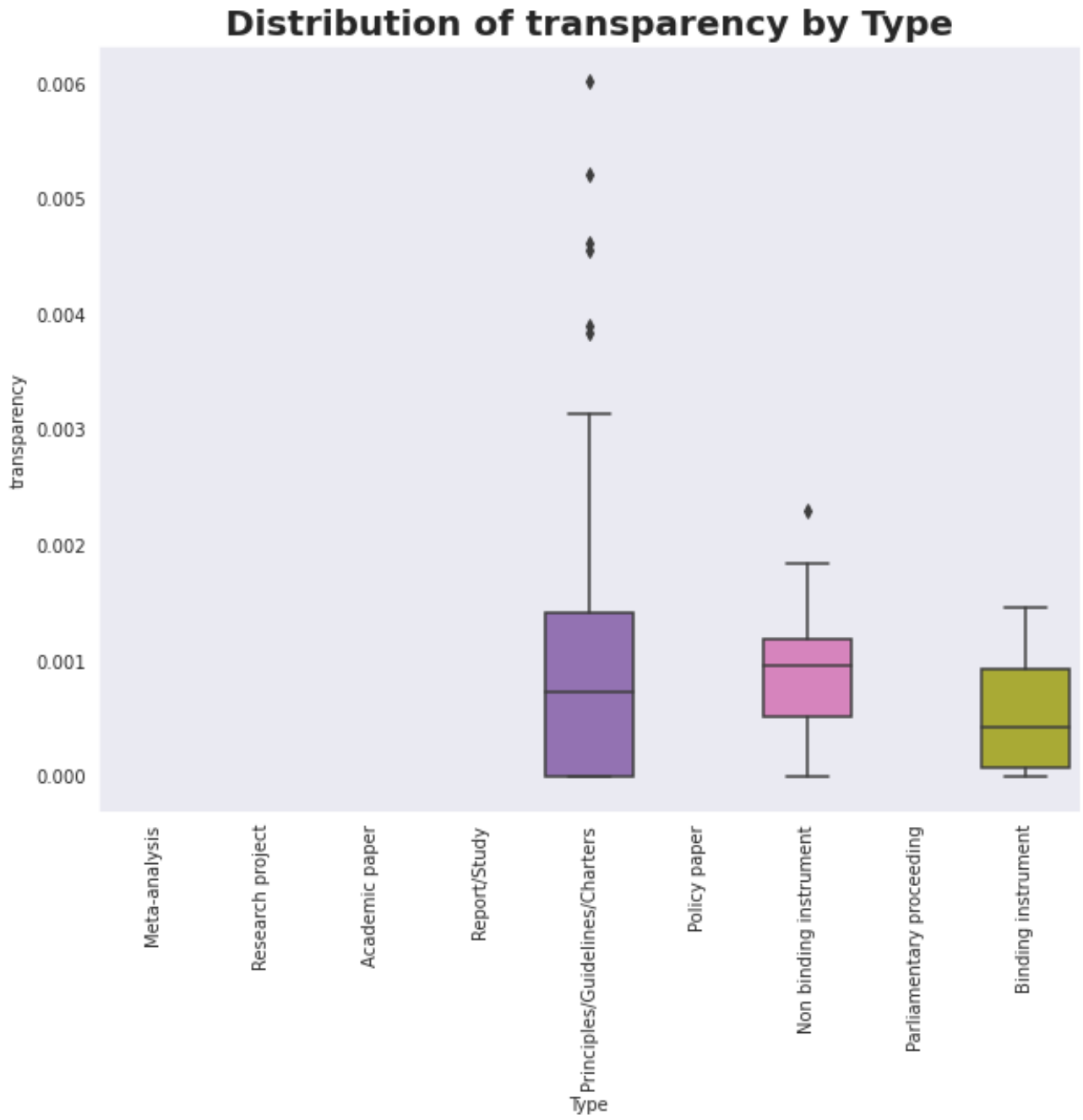
[26]:

```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Distribution of explainability by Type', fontweight='bold', fo

ax = sns.boxplot(x='Type',y='explainability',data=Database,palette='tab10
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.savefig('explainability_type.png')
```



[27]:

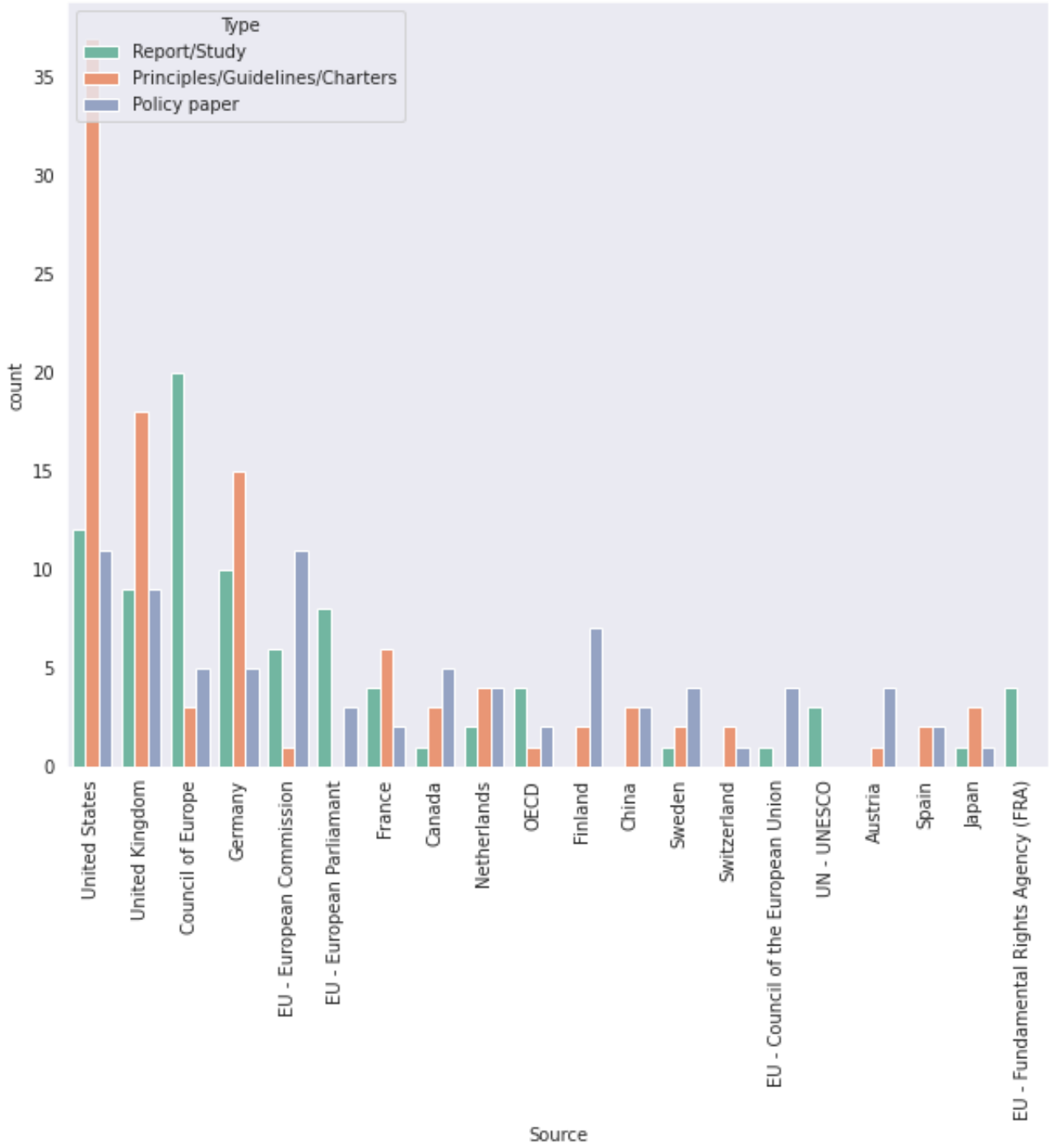
```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Distribution of transparency by Type', fontweight='bold', font

ax = sns.boxplot(x='Type',y='transparency',data=Database,palette='tab10')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.savefig('transparency_type.png')
```



[ ]:

[28]:

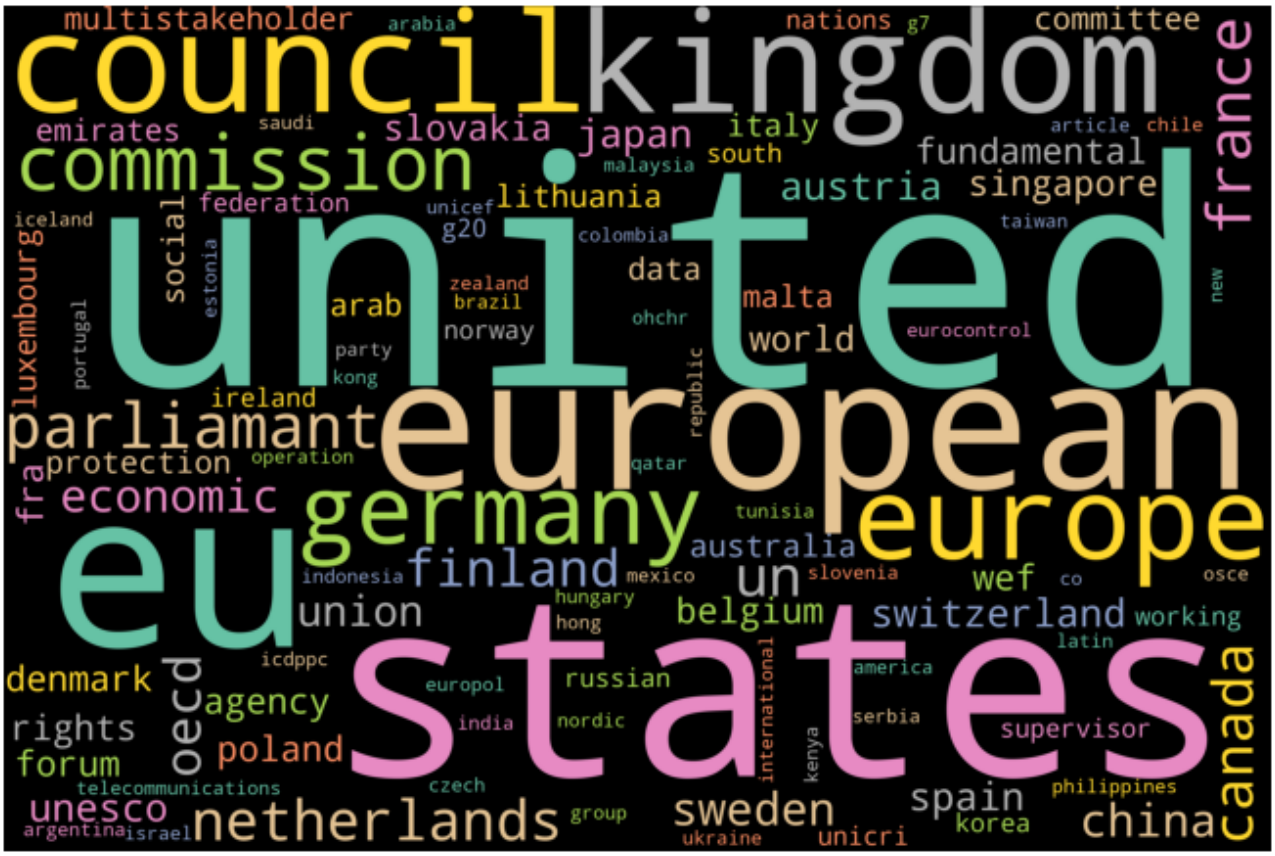
```
plus100 = Database['Type'].map(Database['Type'].value_counts()) > 100 # M

ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
ax = sns.countplot(x='Source',hue='Type',data=Database[plus100],order = D
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.savefig('Source_type.png')
```



[ ]:

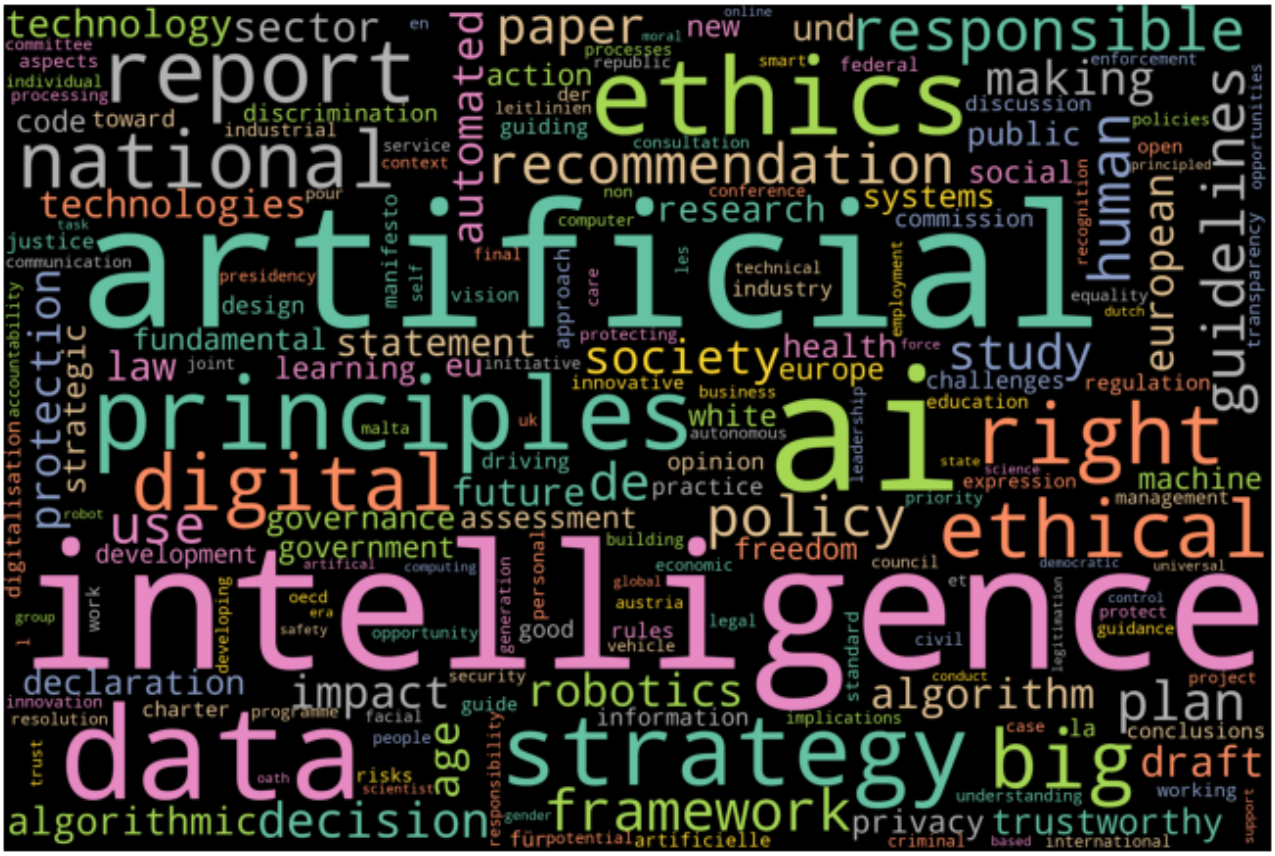
[ ]:

[29]:
```
Source = Database['Source'].apply(lambda x: x.lower())

plt_source = get_wordcloud(Source)

plt.savefig('plt_source.png')
```



[30]:

```
Reference = Database['Reference'].apply(lambda x: x.lower())

plt_Reference = get_wordcloud(Reference)

plt.savefig('plt_Reference.png')
```



[31]:

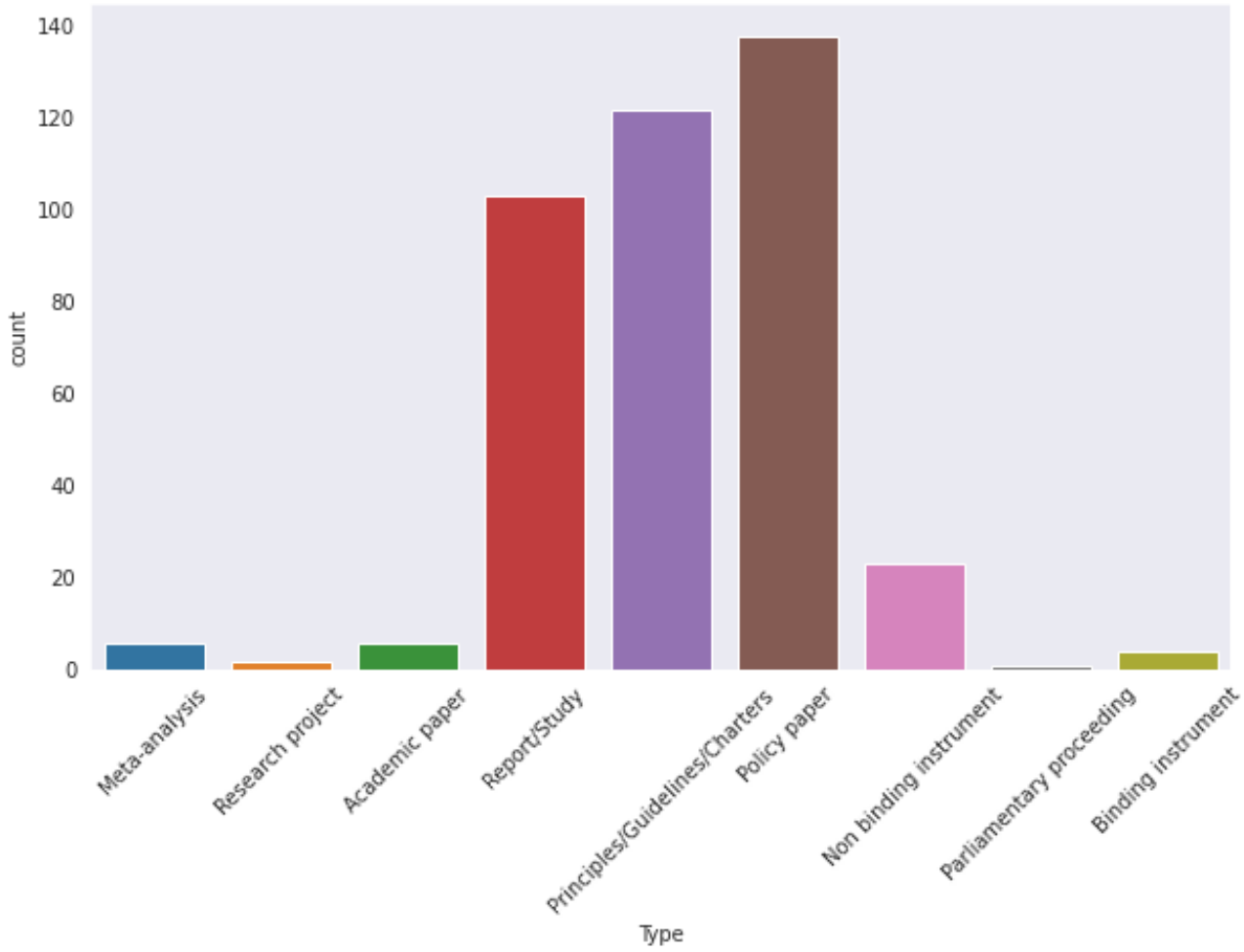
```
Origin = Database['Origin'].apply(lambda x: x.lower())

plt_origin = get_wordcloud(Origin)

plt.savefig('plt_origin.png')
```

[32]:

```
Issuer = Database['Issuer'].apply(lambda x: x.lower())

plt_Issuer = get_wordcloud(Issuer)

plt.savefig('plt_Issuer.png')
```

[33]:

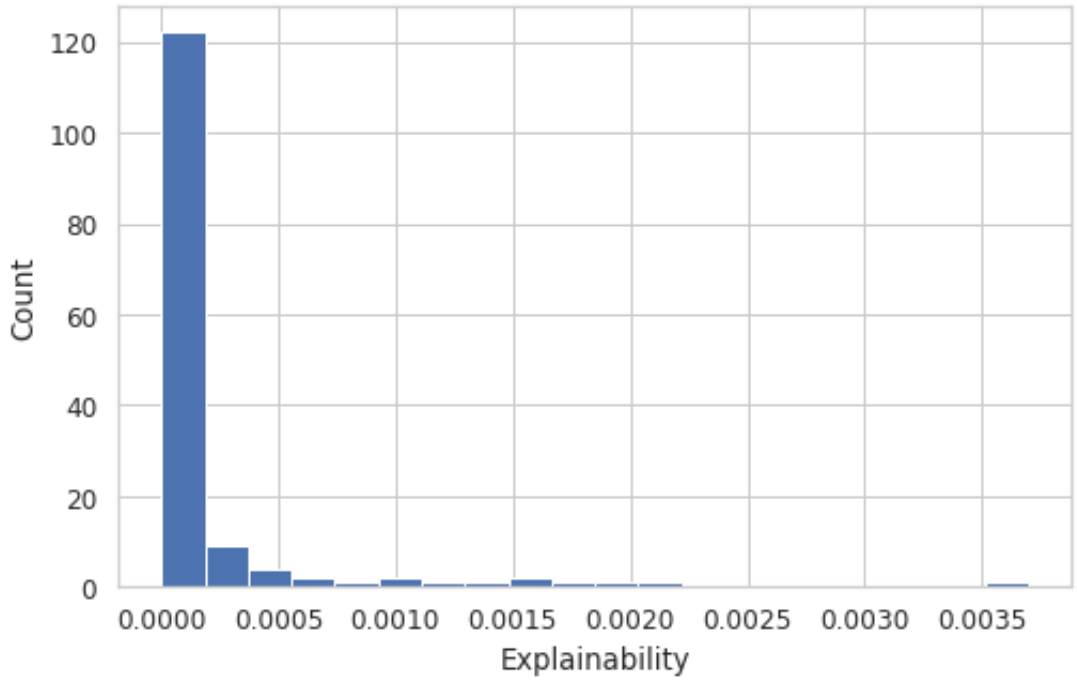
```
ax = sns.countplot(Database['Type'])
ax.set_xticklabels(ax.get_xticklabels(),rotation=45)
plt.savefig('Type1.png')
```



[34]:
```
numerical = ['fundamental rights','human agency','human rights','non disc

'rule of law','sustainable development','well being','accountability','au
 ,'dignity' ,'diversity','explainability','fairness','freedom','inclusive
,'privacy','responsibility','robustness','safe','solidarity','sustainabil

]
categorical = ['Issuer','Reference','Type','Link','Origin','Source','CoE
]

Database = Database[numerical + categorical]
Database.shape
```

Out[34]: (405, 40)

[35]:
```
sns.set(style='whitegrid', palette="deep", font_scale=1.1, rc={"figure.fi
sns.distplot(
    Database['explainability'], norm_hist=False, kde=False, bins=20, hist
).set(xlabel='Explainability', ylabel='Count');
```

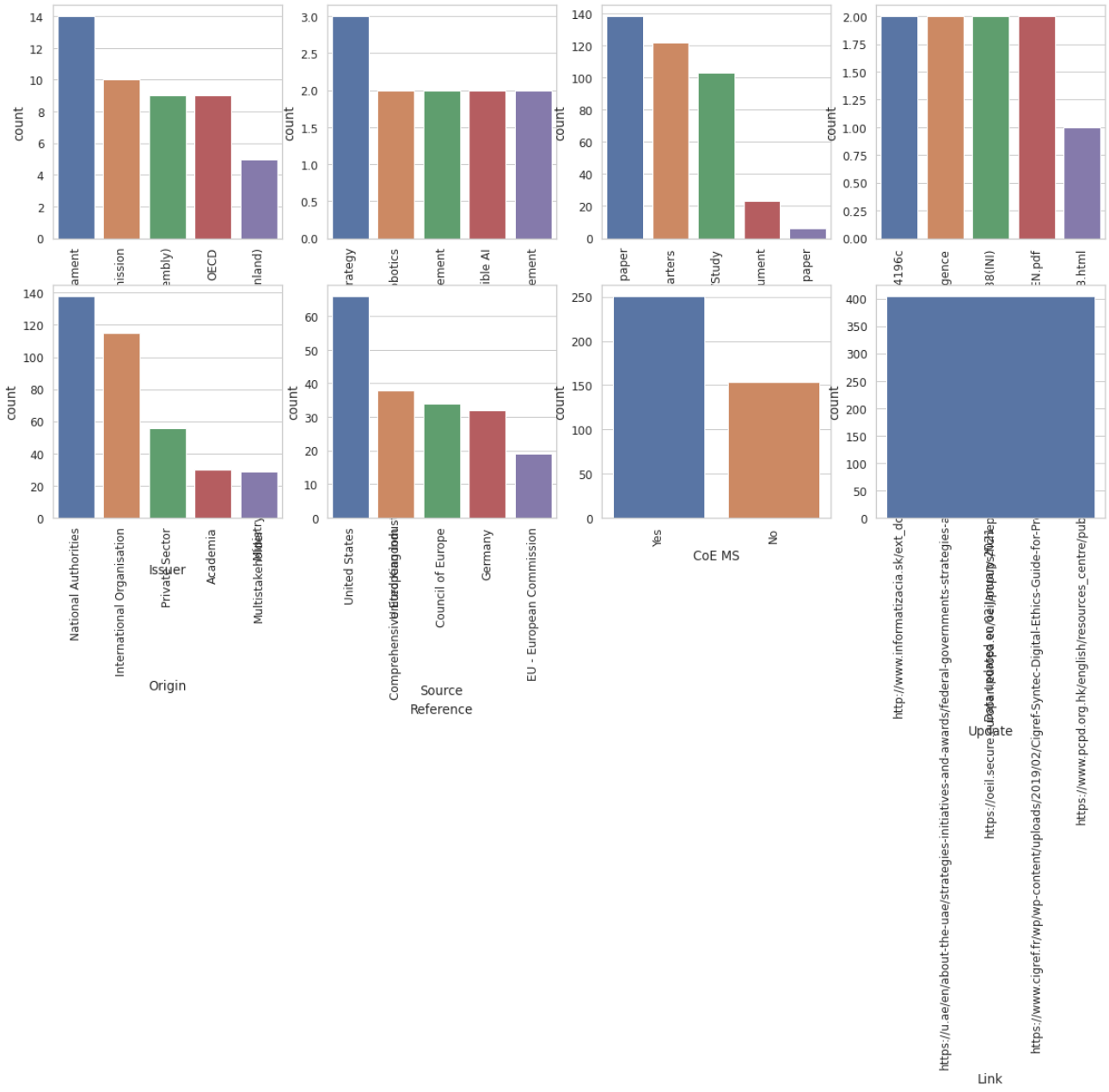


[36]:

```
# Database[numerical].plot.barh(stacked=True);
```

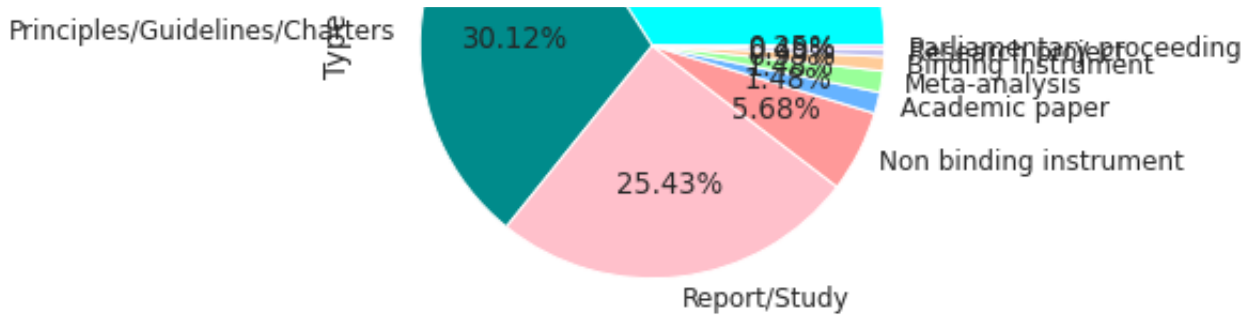
[37]:

```
fig, ax = plt.subplots(2, 4, figsize=(20, 10))
for variable, subplot in zip(categorical, ax.flatten()):
    sns.countplot(Database[variable],order = Database[variable].value_cou
    for label in subplot.get_xticklabels():
        label.set_rotation(90)
```

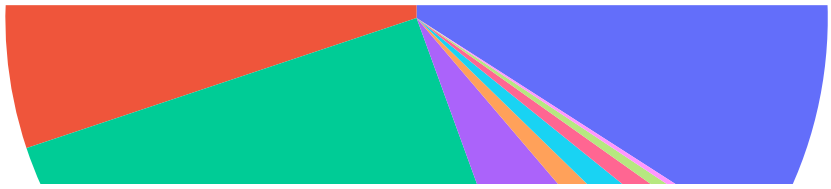


[38]:
```
# Plotly Libraris
import plotly.express as px
import plotly.graph_objects as go

Database['Type'].value_counts().plot.pie(autopct='%2.2f%%', colors = ['cy
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbf04938250>



[39]:
```
import plotly.express as px

fig = px.pie(Database['Type'], values=Database['Type'].value_counts().val
fig.update_traces(hoverinfo='label+percent', textinfo='value')
fig.show()
```



[40]:
```
#sns.catplot(x = 'Type',hue = 'Source',data = Database,order = Database['
#sns.catplot(x = 'Type',col = 'Source',data = Database, order = Database[
```

[41]:
```
dfs = pd.ExcelFile('/kaggle/input/digital-policiers-frameworks/Digital Po

## Data sources description
data = pd.read_excel(dfs, 'Database')
```

Issuer

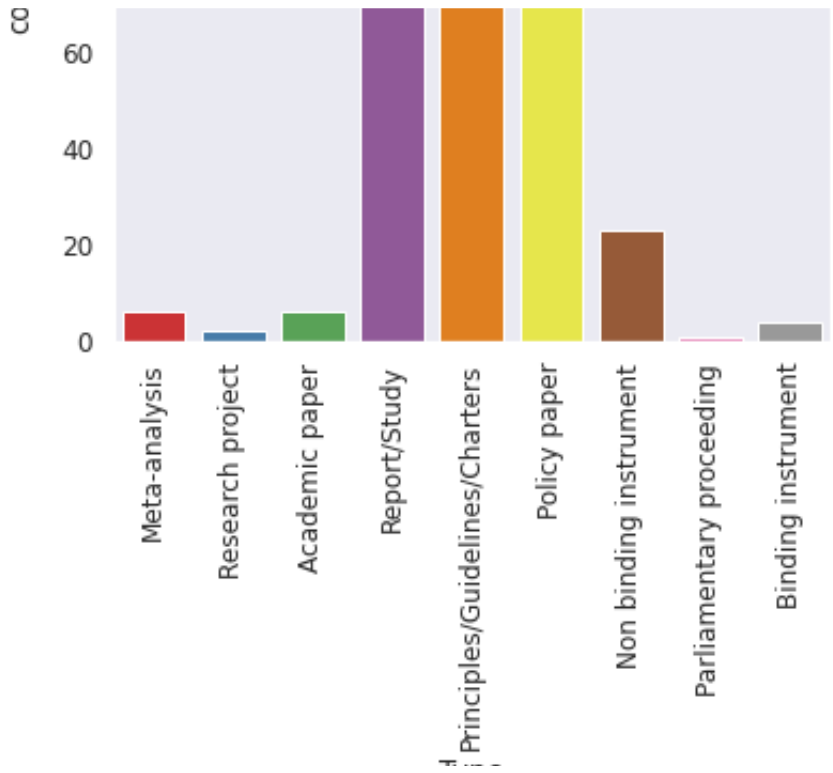
[42]:
```
#Database.head()

fill_data = data.fillna(' ')
fill_data.head()
```

Out[42]:

	Issuer	Reference	Type	Link	Origin
0	Berkman Klein Center (University of Harvard)	Principled Artificial Intelligence	Meta-analysis	https://ssrn.com/abstract=3518482	Academia
1	Cyberjustice Laboratory	ACT Project - Projet AJC (Autonomisation des a...	Research project	https://www.ajcact.org	Academia
2	ETH Zurich	AI, the global landscape of ethics guidelines	Meta-analysis	https://arxiv.org/ftp/arxiv/papers/1906/1906.1...	Academia
3	ETH Zurich	A Moral Framework for Understanding of Fair ML...	Academic paper	https://arxiv.org/abs/1809.03400	Academia
4	Fraunhofer Institute for Intelligent Analysis ...	Trustworthy Use of Artificial Intelligence	Report/Study	https://www.iais.fraunhofer.de/content/dam/iai...	Academia

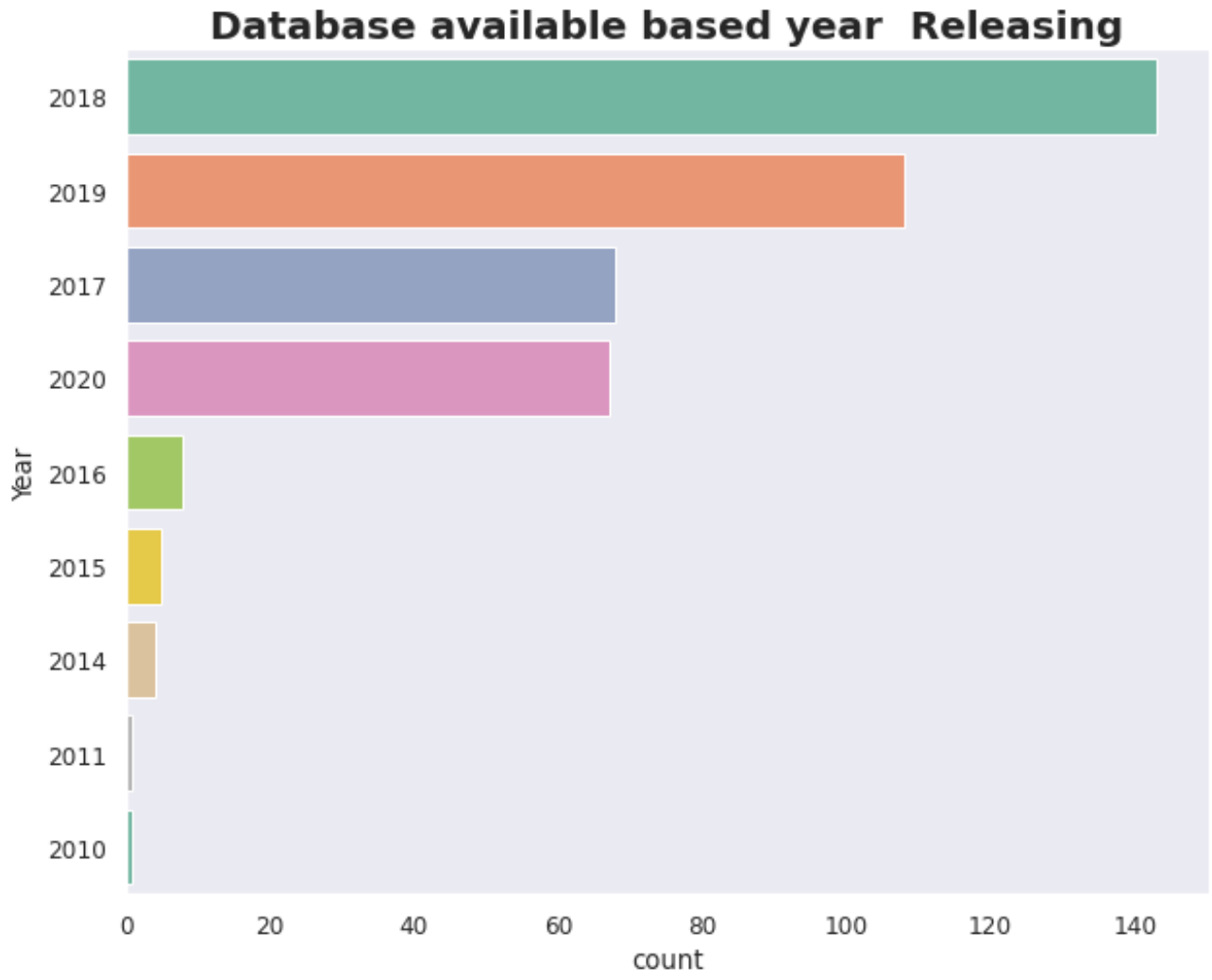
[43]:
```
# creating Countplot from Seaborn to show max available content in NETFLI

sns.set_style('dark')
ax = plt.subplots(figsize = (6, 6))
plt.title('Countplot for Type for Issuer ', fontweight='bold')
ax = sns.countplot(x = 'Type', data=data, palette='Set1')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

Out[43]:
```
[Text(0, 0, 'Meta-analysis'),
 Text(0, 0, 'Research project'),
 Text(0, 0, 'Academic paper'),
 Text(0, 0, 'Report/Study'),
 Text(0, 0, 'Principles/Guidelines/Charters'),
 Text(0, 0, 'Policy paper'),
 Text(0, 0, 'Non binding instrument'),
 Text(0, 0, 'Parliamentary proceeding'),
 Text(0, 0, 'Binding instrument')]
```



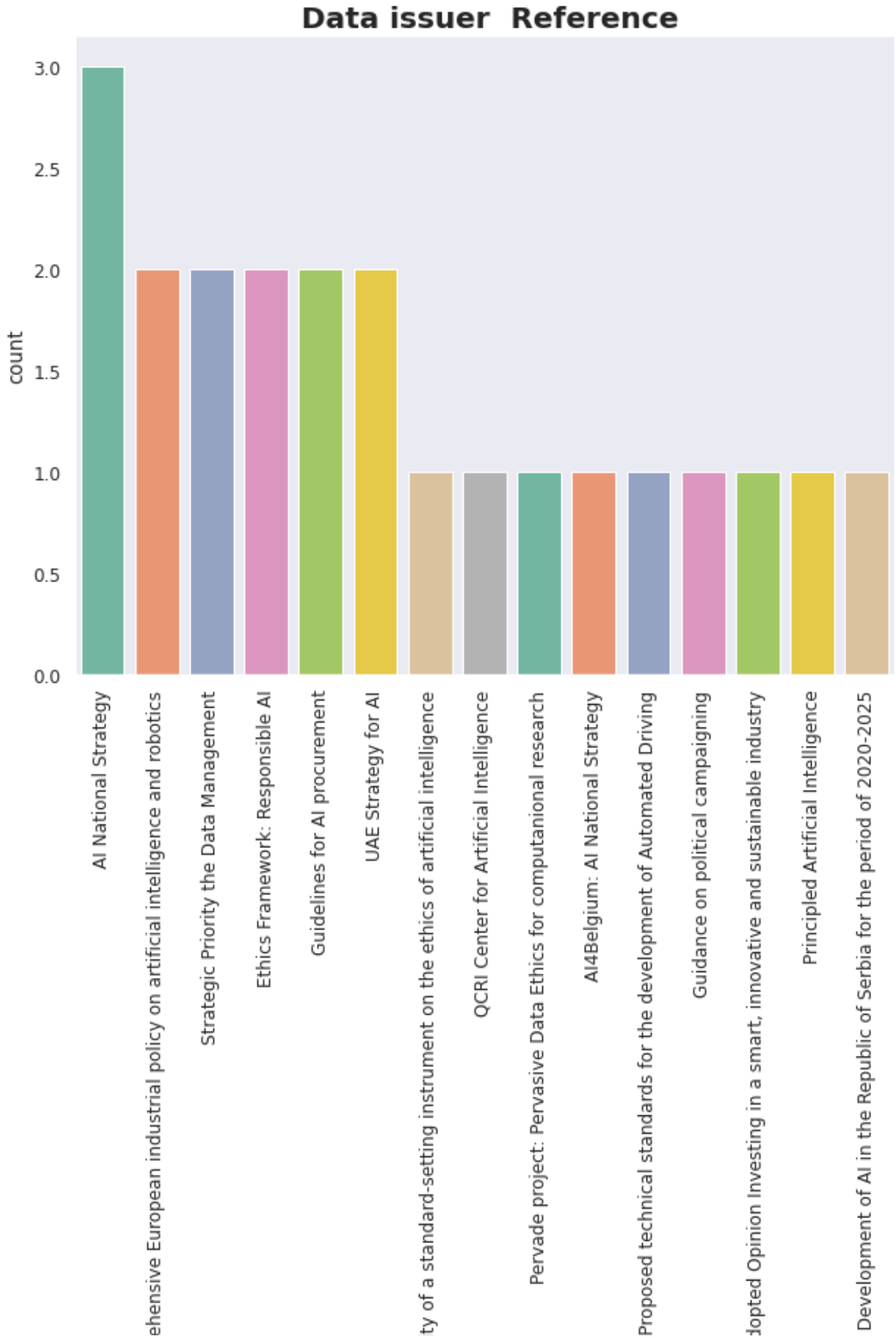
Type

[92]:
```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Database available based year  Releasing', fontweight='bold',
ax = sns.countplot(y = 'Year', data = data, order = data['Year'].value_co
plt.savefig('Year_release.png')
```



[45]:
```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title(' Data issuer  Reference', fontweight='bold', fontsize=20)
ax = sns.countplot(x = 'Reference', data = data, palette = 'Set2', order
#ax.set_xticklabels(labels, rotation=45)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

Out[45]:
```
[Text(0, 0, 'AI National Strategy'),
 Text(0, 0, 'Comprehensive European industrial policy on artificial intelligence and robotics'),
 Text(0, 0, 'Strategic Priority the Data Management'),
 Text(0, 0, 'Ethics Framework: Responsible AI'),
 Text(0, 0, 'Guidelines for AI procurement'),
 Text(0, 0, 'UAE Strategy for AI'),
 Text(0, 0, 'Preliminary study on the technical and legal aspects relating to the desirability of a standard-setting instrument on the ethics of artificial intelligence'),
 Text(0, 0, 'QCRI Center for Artificial Intelligence'),
 Text(0, 0, 'Pervade project: Pervasive Data Ethics for computanional research'),
 Text(0, 0, 'AI4Belgium: AI National Strategy'),
 Text(0, 0, 'Safety First for Automated Driving – Proposed technical standards for the development of Automated Driving'),
 Text(0, 0, 'Guidance on political campaigning'),
 Text(0, 0, 'Adopted Opinion Investing in a smart, innovative and sustainable industry'),
 Text(0, 0, 'Principled Artificial Intelligence'),
 Text(0, 0, 'Strategy for the Development of AI in the Republic of Serbia for the period of 2020-2025')]
```

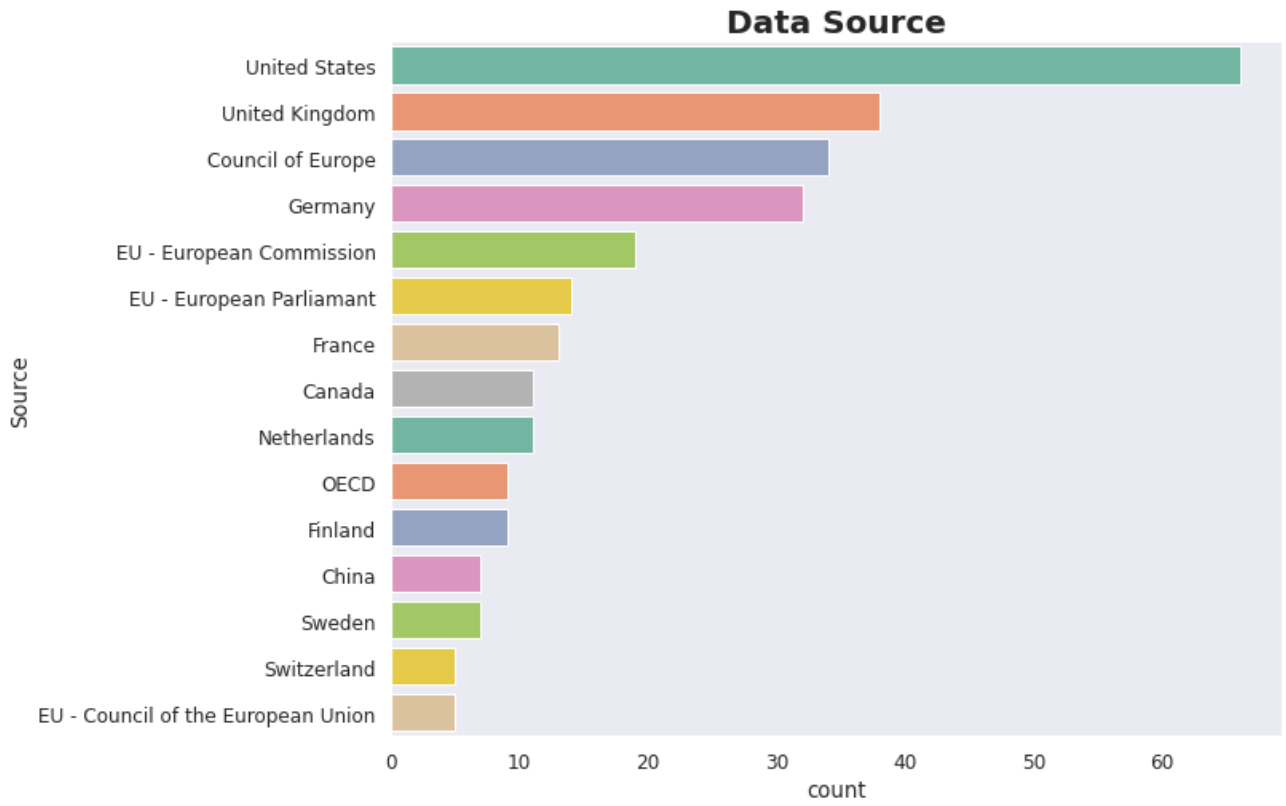

Data issuer Reference
count
3.0
2.5
2.0
1.5
1.0
0.5
0.0
AI National Strategy
ehensive European industrial policy on artificial intelligence and robotics
Strategic Priority the Data Management
Ethics Framework: Responsible AI
Guidelines for AI procurement
UAE Strategy for AI
ty of a standard-setting instrument on the ethics of artificial intelligence
QCRI Center for Artificial Intelligence
Pervade project: Pervasive Data Ethics for computanional research
AI4Belgium: AI National Strategy
Proposed technical standards for the development of Automated Driving
Guidance on political campaigning
dopted Opinion Investing in a smart, innovative and sustainable industry
Principled Artificial Intelligence
Development of AI in the Republic of Serbia for the period of 2020-2025

[46]:

```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Data issuer  Reference', fontweight='bold', fontsize=20)
#ax = sns.countplot(x = 'Reference', data = data, palette = 'Set2', order
ax = sns.countplot(y = 'Reference', data = data, order = data['Reference'
```

[47]:

```
#Source

ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Data Source', fontweight='bold', fontsize=20)
ax = sns.countplot(y = 'Source', data = data, order = data['Source'].valu
```

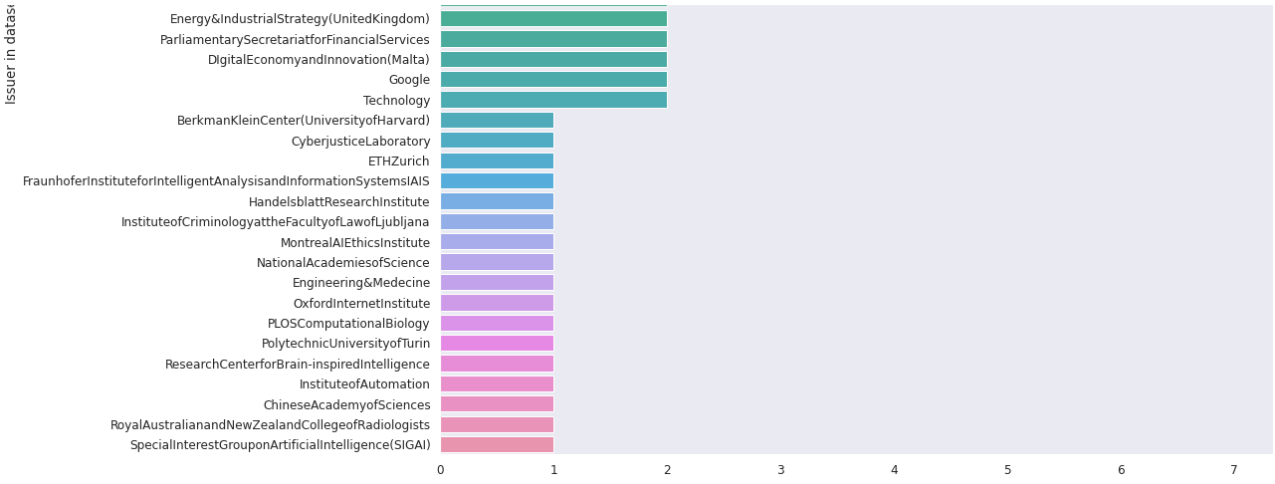


[48]:
```
# More Issuer content creating countries

countries = {}
data['Issuer'] = data['Issuer'].fillna('Unknown')

list_countries = list(data['Issuer'])

for i in list_countries:
    i = list(i.split(','))

    if len(i) is 1:
        if i in list(countries.keys()):
            countries[i] += 1
        else:
            countries[i[0]] = 1
    else:
        for j in i:
            if j in list(countries.keys()):
                countries[j] += 1
            else:
                countries[j] = 1
```

[49]:
```
final_countries = {}

for country, no in countries.items():
    country = country.replace(' ','')

    if country in list(final_countries.keys()):
        final_countries[country] += no
    else:
        final_countries[country] = no

final_countries = {k : v for k, v in sorted(final_countries.items(), key
```

[50]:
```
plt.figure(figsize = (15, 15))
plt.title(' Issuer Creating Countries', fontweight = 'bold', fontsize=15)

y_ver = list(final_countries.keys())
x_hor = list(final_countries.values())
sns.barplot( y = y_ver[0:40], x = x_hor[0:40])
plt.ylabel('Issuer in dataset')
```

Out[50]: Text(0, 0.5, 'Issuer in dataset')

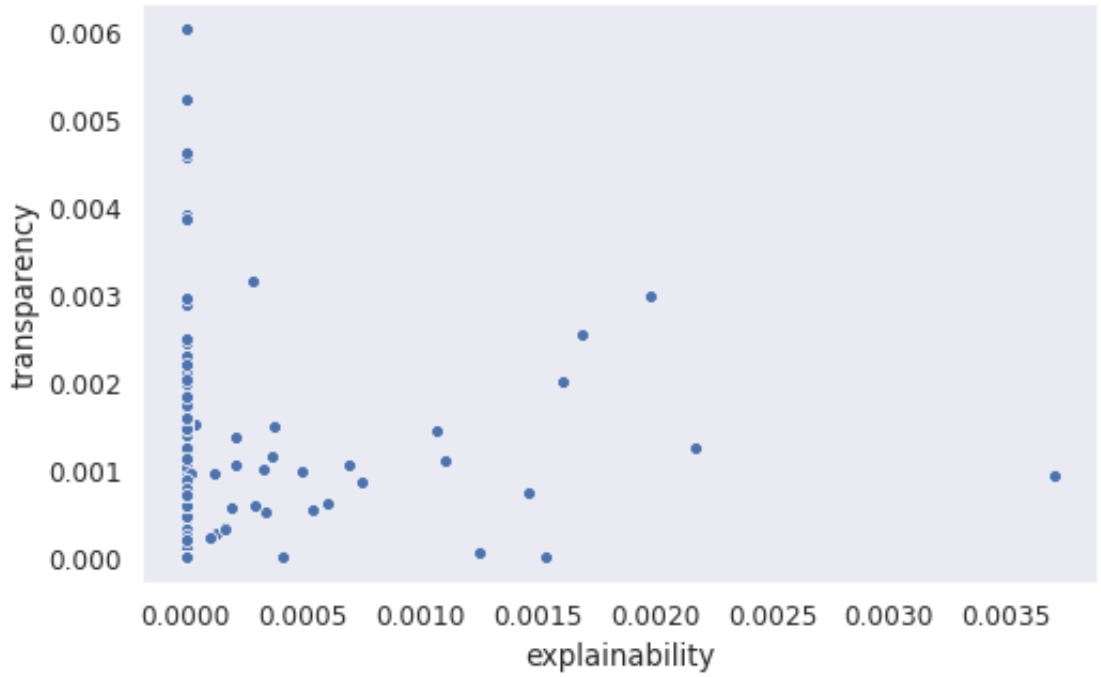


[51]:
```
data[data['Year'] == 2020].groupby('Type')['Year'].count()
```

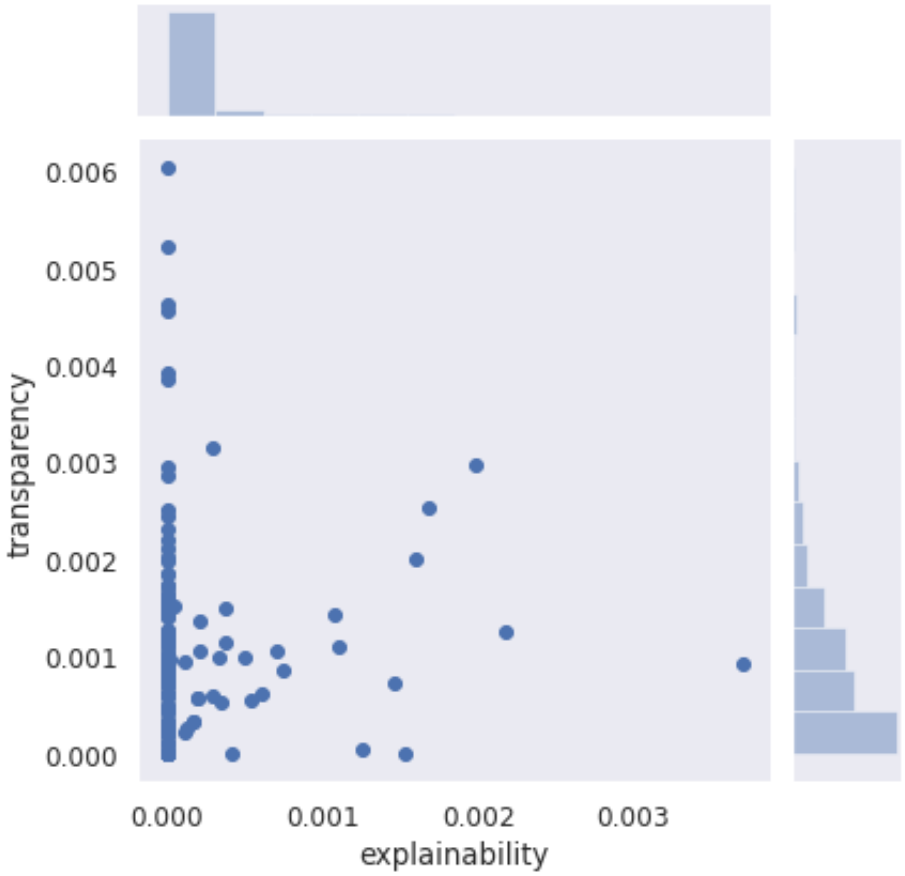
Out[51]:
```
Type
Binding instrument                1
Meta-analysis                     2
Non binding instrument            3
Policy paper                     21
Principles/Guidelines/Charters    9
Report/Study                     31
Name: Year, dtype: int64
```

Analyzing Relationships Between Numerical Variables

[93]:
```
## relationship between transparency and explainability

sns.scatterplot(x=data['explainability'], y=data['transparency']);
plt.savefig('explainability.png')
```

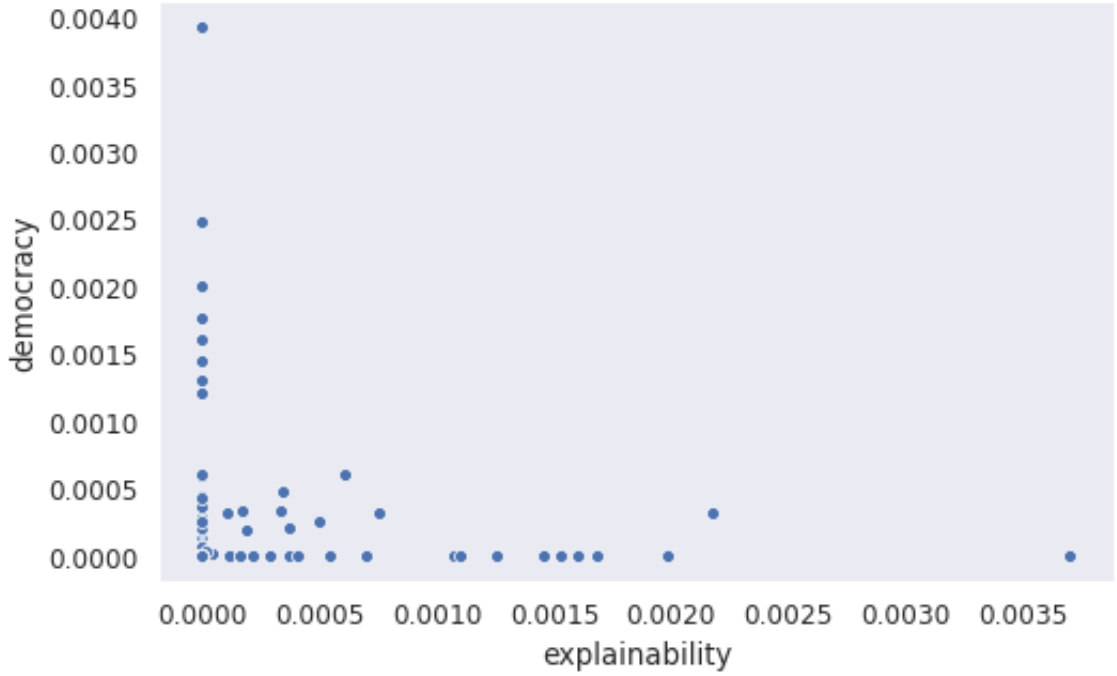


[53]:
```
sns.jointplot(x=data['explainability'], y=data['transparency']);
```

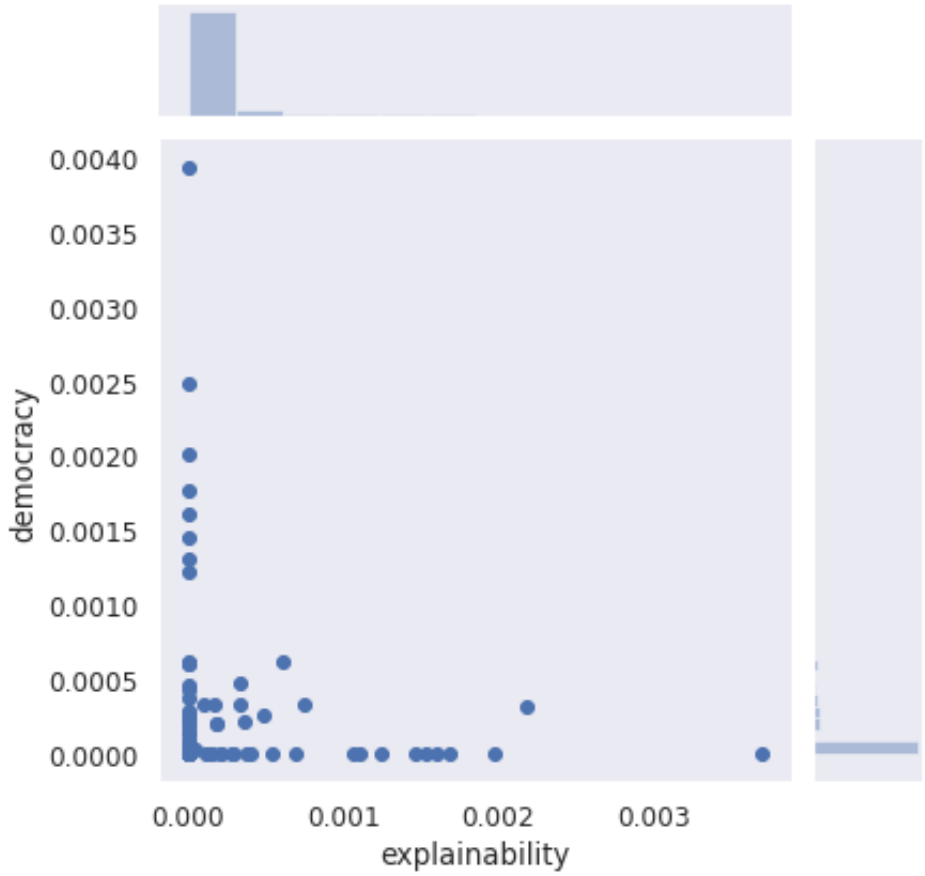


[54]:

```
## relationship between democracy and explainability

sns.scatterplot(x=data['explainability'], y=data['democracy']);
```

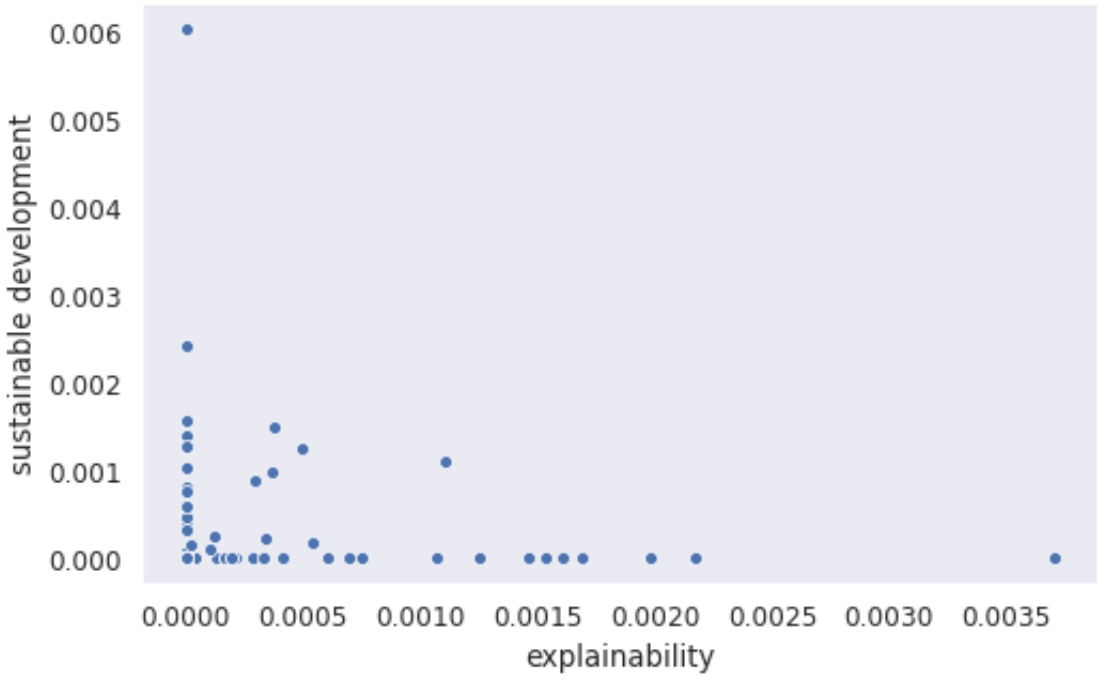


[55]:
```
sns.jointplot(x=data['explainability'], y=data['democracy']);
```

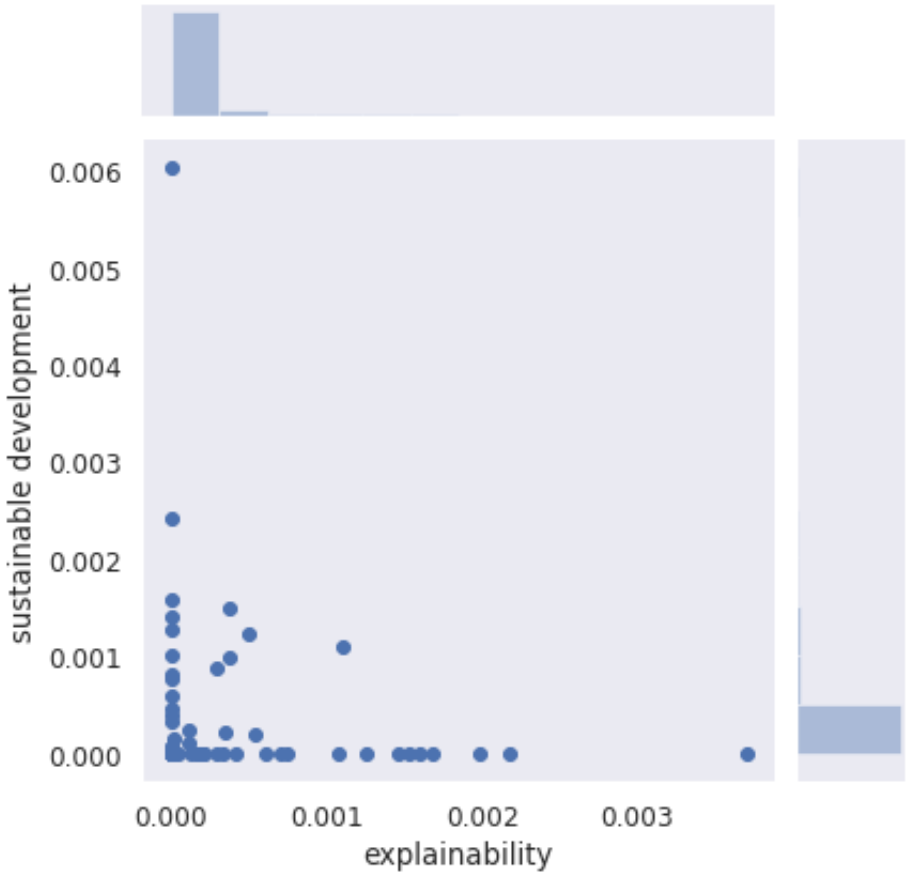


[56]:

```
## relationship between sustainable development and explainability

sns.scatterplot(x=data['explainability'], y=data['sustainable development
```

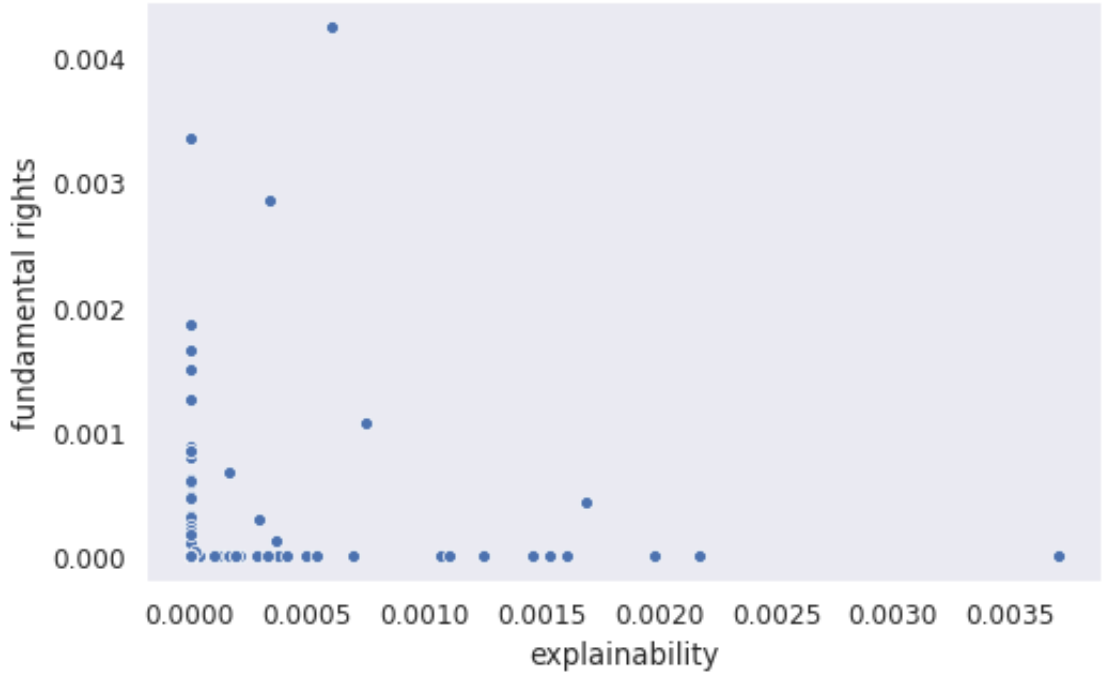


[57]:
```
sns.jointplot(x=data['explainability'], y=data['sustainable development'])
```

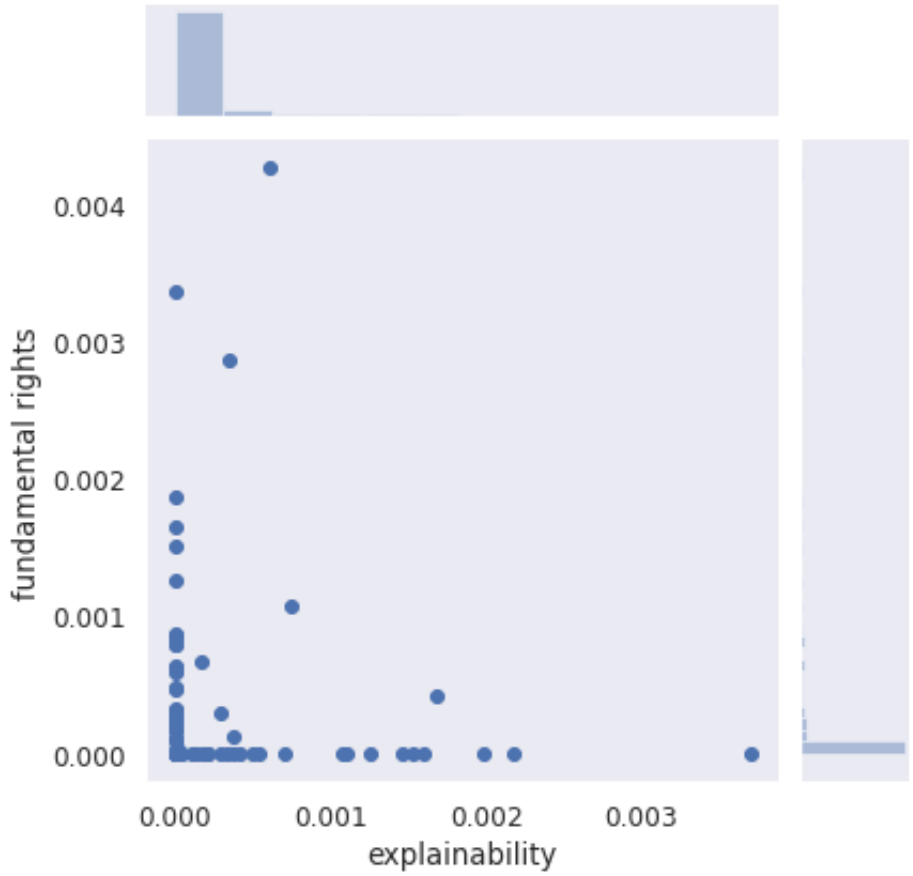


[58]:

```
## relationship between fundamental rights and explainability

sns.scatterplot(x=data['explainability'], y=data['fundamental rights']);
```



[59]:
```
sns.jointplot(x=data['explainability'], y=data['fundamental rights']);
```



[60]:
```
#sns.pairplot(Database[numerical], kind="scatter")
#plt.show()
```

Analyzing Relationships Between Numerical and Categorical Variables

[61]:
```
pd.crosstab(Database.Type, Database.Source)
```

Out[61]:

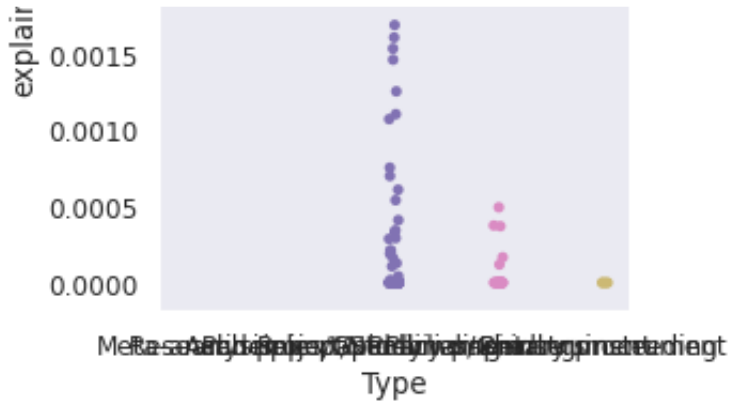
Source	Argentina	Australia	Austria	Belgium	Brazil	Canada	Chile	China
Type								
Academic paper	0	0	0	0	0	0	0	0
Binding instrument	0	0	0	0	0	1	0	0
Meta-analysis	0	0	0	0	0	0	0	1
Non binding instrument	0	0	0	0	0	0	0	0
Parliamentary proceeding	0	0	0	0	0	0	0	0
Policy paper	1	0	4	3	1	5	1	3
Principles/Guidelines/Charters	0	3	1	1	0	3	0	3
Report/Study	0	0	0	0	0	1	0	0
Research project	0	0	0	0	0	1	0	0

[62]:
```
from scipy.stats import chi2_contingency
chi2_contingency(pd.crosstab(Database.Type, Database.Source))
```

Out[62]:
```
(597.7809100308771,
 0.7684950281913201,
 624,
 array([[1.48148148e-02, 4.44444444e-02, 7.40740741e-02, 5.92592593e-02,
         1.48148148e-02, 1.62962963e-01, 1.48148148e-02, 1.03703704e-01,
         1.48148148e-02, 5.03703704e-01, 1.48148148e-02, 5.92592593e-02,
         1.48148148e-02, 7.40740741e-02, 2.81481481e-01, 2.96296296e-02,
         4.44444444e-02, 2.07407407e-01, 1.48148148e-02, 5.92592593e-02,
         1.48148148e-02, 1.48148148e-02, 1.48148148e-02, 1.33333333e-01,
         1.92592593e-01, 2.96296296e-02, 1.48148148e-02, 4.74074074e-01,
         1.48148148e-02, 1.48148148e-02, 1.48148148e-02, 1.48148148e-02,
         1.48148148e-02, 1.48148148e-02, 1.48148148e-02, 2.96296296e-02,
         1.48148148e-02, 4.44444444e-02, 7.40740741e-02, 1.48148148e-02,
         1.48148148e-02, 1.48148148e-02, 4.44444444e-02, 4.44444444e-02,
         1.48148148e-02, 4.44444444e-02, 1.48148148e-02, 4.44444444e-02,
         1.62962963e-01, 1.48148148e-02, 1.48148148e-02, 2.96296296e-02,
         1.33333333e-01, 1.48148148e-02, 1.48148148e-02, 5.92592593e-02,
```

[63]:
```
sns.catplot(x="Type", y="explainability", data=Database)
```

Out[63]: <seaborn.axisgrid.FacetGrid at 0x7fbefec73f90>



[64]:
```
from sklearn.cluster import KMeans
```

[65]:
```
categorical = Database[categorical]
```

[66]:
```
categorical.describe()
```

Out[66]:

	Year
count	405.000000
mean	2018.276543
std	1.284935
min	2010.000000
25%	2018.000000
50%	2018.000000
75%	2019.000000
max	2020.000000

[67]:
```
categorical.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 405 entries, 0 to 404
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Issuer     405 non-null    object
 1   Reference  405 non-null    object
 2   Type       405 non-null    object
 3   Link       405 non-null    object
 4   Origin     405 non-null    object
 5   Source     405 non-null    object
 6   CoE MS     405 non-null    object
 7   Update     405 non-null    object
 8   Year       405 non-null    int64 
dtypes: int64(1), object(8)
memory usage: 28.6+ KB
```

[68]:
```
categorical.isnull().sum()*100/categorical.shape[0]
```

Out[68]:
```
Issuer       0.0
Reference    0.0
Type         0.0
Link         0.0
Origin       0.0
Source       0.0
CoE MS       0.0
Update       0.0
Year         0.0
dtype: float64
```

Model Building K-modes

[69]:
```
categorical_copy = categorical.copy()
```

[70]:
```
from sklearn import preprocessing
from kmodes.kmodes import KModes
le = preprocessing.LabelEncoder()
categorical = categorical.apply(le.fit_transform)
categorical.head()
```

Out[70]:

	Issuer	Reference	Type	Link	Origin	Source	CoE MS	Update	Year
0	21	271	2	203	0	77	0	0	6
1	66	15	8	240	0	5	0	0	6
2	86	37	2	72	0	66	1	0	7
3	86	8	0	70	0	66	1	0	6
4	111	380	7	313	0	27	1	0	8

[71]:
```
#Using K-Mode with "Cao" initialization
km_cao = KModes(n_clusters=2, init = "Cao", n_init = 1, verbose=1)
fitClusters_cao = km_cao.fit_predict(categorical)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 15, cost: 2241.0
```

[72]:
```
# Predicted Clusters
fitClusters_cao
```

Out[72]:
```
array([1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=uint16)
```

[73]:
```
clusterCentroidsDf = pd.DataFrame(km_cao.cluster_centroids_)
clusterCentroidsDf.columns = categorical.columns
```

[74]:
```
# Mode of the clusters
clusterCentroidsDf
```

Out[74]:

	Issuer	Reference	Type	Link	Origin	Source	CoE MS	Update	Year
0	99	22	5	48	4	76	1	0	6
1	198	4	6	5	2	77	0	0	7

[75]:
```
#Using K-Mode with "Huang" initialization
km_huang = KModes(n_clusters=2, init = "Huang", n_init = 1, verbose=1)
fitClusters_huang = km_huang.fit_predict(categorical)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 58, cost: 2243.0
Run 1, iteration: 2/100, moves: 19, cost: 2243.0
```

[76]:
```
# Predicted clusters
fitClusters_huang
```

Out[76]:
```
array([1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=uint16)
```

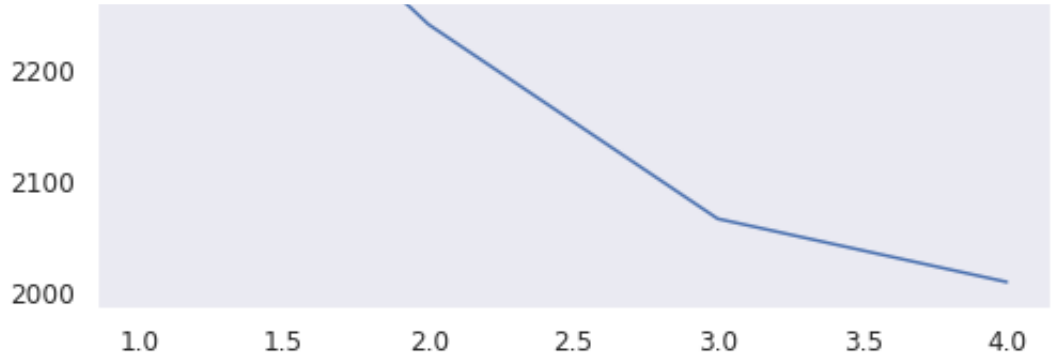
Choosing K by comparing Cost against each K

[77]:
```
cost = []
for num_clusters in list(range(1,5)):
    kmode = KModes(n_clusters=num_clusters, init = "Cao", n_init = 1, ver
    kmode.fit_predict(categorical)
    cost.append(kmode.cost_)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 0, cost: 2485.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 15, cost: 2241.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 39, cost: 2066.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 52, cost: 2009.0
Run 1, iteration: 2/100, moves: 1, cost: 2009.0
```

[78]:
```
y = np.array([i for i in range(1,5,1)])
plt.plot(y,cost)
```

Out[78]: [<matplotlib.lines.Line2D at 0x7fbefb89afd0>]



[79]:
```
## Choosing K=2

km_cao = KModes(n_clusters=2, init = "Cao", n_init = 1, verbose=1)
fitClusters_cao = km_cao.fit_predict(categorical)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 15, cost: 2241.0
```

[80]:
```
fitClusters_cao
```

Out[80]:
```
array([1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=uint16)
```

[81]:
```
#Combining the predicted clusters with the original DF
categorical = categorical_copy.reset_index()
```

[82]:
```
clustersDf = pd.DataFrame(fitClusters_cao)
clustersDf.columns = ['cluster_predicted']
combinedDf = pd.concat([categorical, clustersDf], axis = 1).reset_index()
combinedDf = combinedDf.drop(['index', 'level_0'], axis = 1)
```

[83]:
```
combinedDf.head()
```

Out[83]:

	Issuer	Reference	Type	Link	Origin
0	Berkman Klein Center (University of Harvard)	Principled Artificial Intelligence	Meta-analysis	https://ssrn.com/abstract=3518482	Academia
1	Cyberjustice Laboratory	ACT Project - Projet AJC (Autonomisation des a...	Research project	https://www.ajcact.org	Academia
2	ETH Zurich	AI, the global landscape of ethics guidelines	Meta-analysis	https://arxiv.org/ftp/arxiv/papers/1906/1906.1...	Academia
3	ETH Zurich	A Moral Framework for Understanding of Fair ML...	Academic paper	https://arxiv.org/abs/1809.03400	Academia
4	Fraunhofer Institute for Intelligent Analysis ...	Trustworthy Use of Artificial Intelligence	Report/Study	https://www.iais.fraunhofer.de/content/dam/iai...	Academia

Cluster Identification

[84]:
```
cluster_0 = combinedDf[combinedDf['cluster_predicted'] == 0]
cluster_1 = combinedDf[combinedDf['cluster_predicted'] == 1]
```

[85]:

```
cluster_0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 271 entries, 1 to 404
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Issuer             271 non-null    object
 1   Reference          271 non-null    object
 2   Type               271 non-null    object
 3   Link               271 non-null    object
 4   Origin             271 non-null    object
 5   Source             271 non-null    object
 6   CoE MS             271 non-null    object
 7   Update             271 non-null    object
 8   Year               271 non-null    int64 
 9   cluster_predicted  271 non-null    uint16
dtypes: int64(1), object(8), uint16(1)
memory usage: 21.7+ KB
```

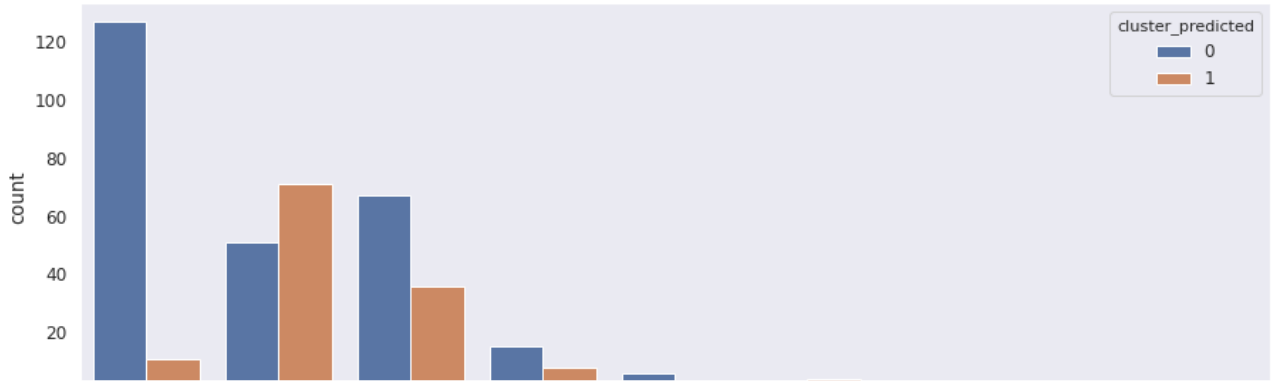
[86]:

```
cluster_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134 entries, 0 to 397
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Issuer             134 non-null    object
 1   Reference          134 non-null    object
 2   Type               134 non-null    object
 3   Link               134 non-null    object
 4   Origin             134 non-null    object
 5   Source             134 non-null    object
 6   CoE MS             134 non-null    object
 7   Update             134 non-null    object
 8   Year               134 non-null    int64 
 9   cluster_predicted  134 non-null    uint16
dtypes: int64(1), object(8), uint16(1)
memory usage: 10.7+ KB
```

[94]:

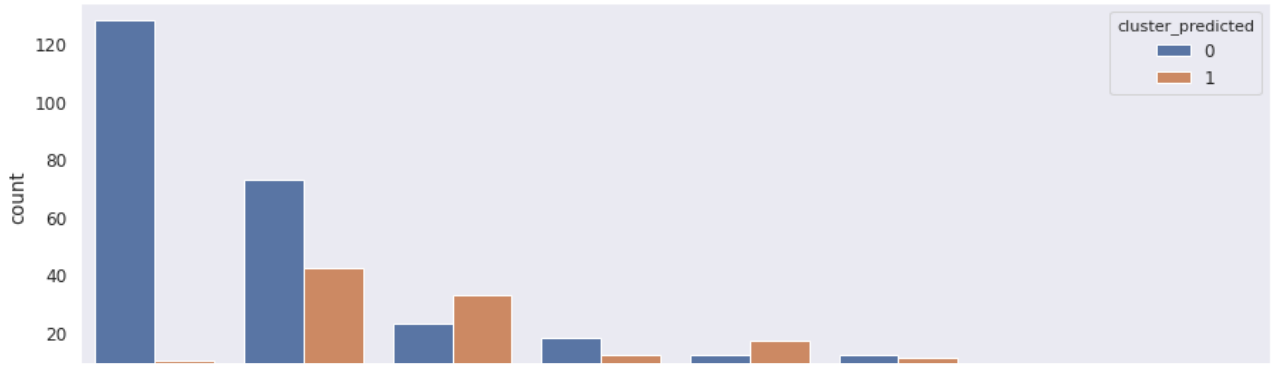
```
plt.subplots(figsize = (15,5))
ax = sns.countplot(x=combinedDf['Type'],order=combinedDf['Type'].value_co
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()
plt.savefig('cluster_type.png')
```




```
<Figure size 576x360 with 0 Axes>
```

[95]:

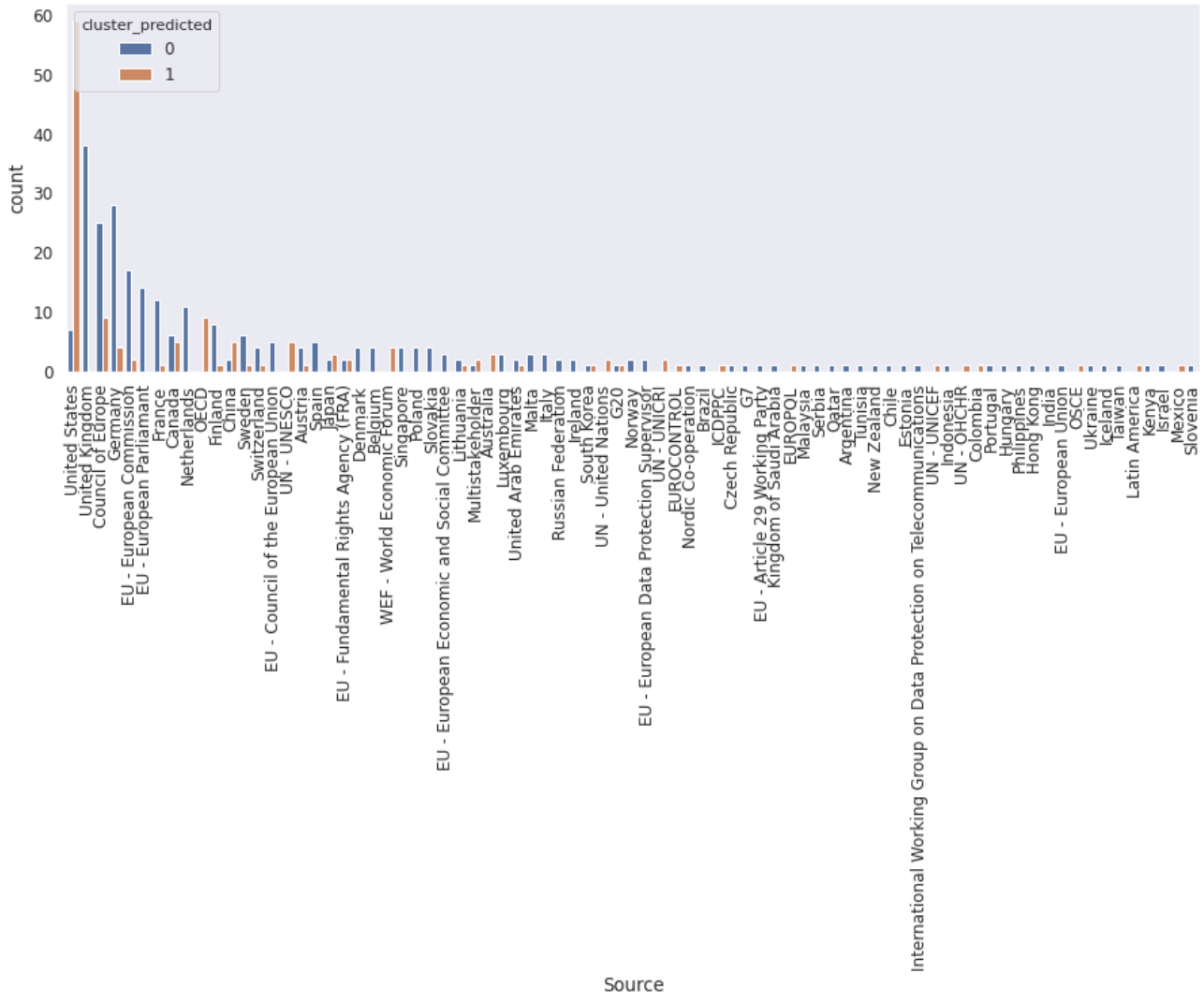
```
plt.subplots(figsize = (15,5))
ax = sns.countplot(x=combinedDf['Origin'],order=combinedDf['Origin'].valu
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()
plt.savefig('cluster_origin.png')
```



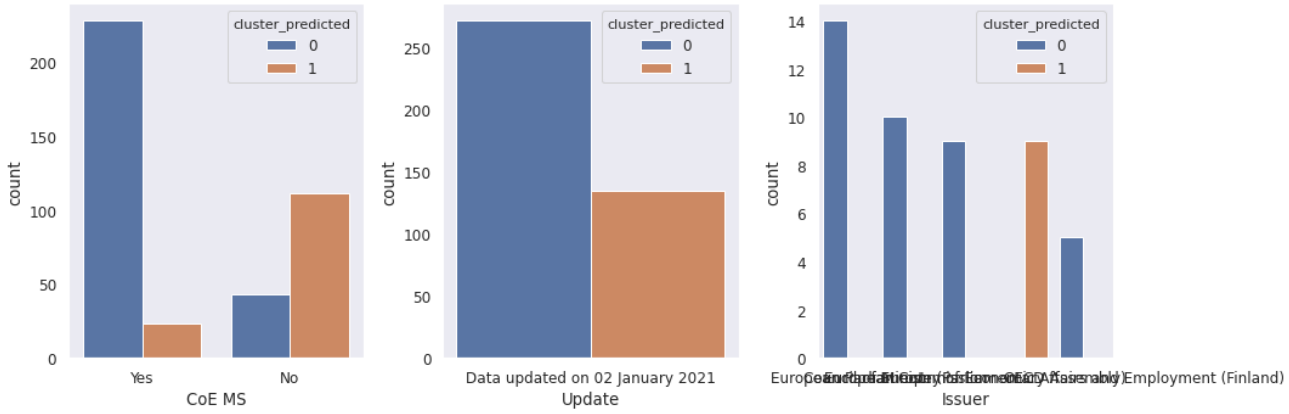

```
<Figure size 576x360 with 0 Axes>
```

[96]:

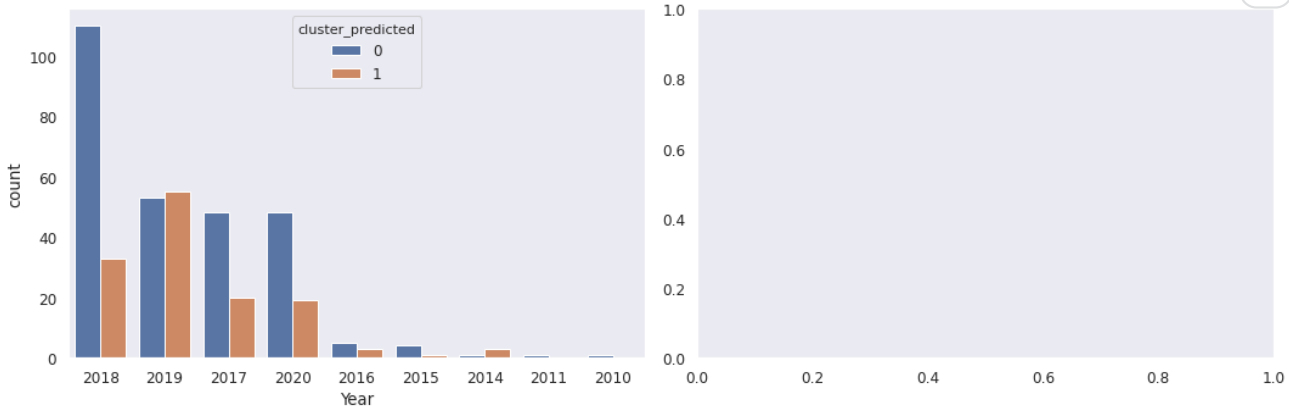
```
plt.subplots(figsize = (15,5))
ax = sns.countplot(x=combinedDf['Source'],order=combinedDf['Source'].valu
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.savefig('Cluster_source.png')
plt.show()
```



[90]:
```
f, axs = plt.subplots(1,3,figsize = (15,5))
sns.countplot(x=combinedDf['CoE MS'],order=combinedDf['CoE MS'].value_cou
sns.countplot(x=combinedDf['Update'],order=combinedDf['Update'].value_cou
sns.countplot(x=combinedDf['Issuer'],order=combinedDf['Issuer'].value_cou

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




```
f, axs = plt.subplots(1,2,figsize = (15,5))
sns.countplot(x=combinedDf['Year'],order=combinedDf['Year'].value_counts(
plt.tight_layout()
plt.show()
```



+ Code + Markdown