[321]:

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

port numpy as np # linear algebra
port 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) will li

port os
r 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/) that g
You can also write temporary files to /kaggle/temp/, but they won't be save
```

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

[322]:

```
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()

## 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>
```

[]:

[323]:
```
from io import BytesIO

import requests
import pandas as pd

#r = requests.get('https://docs.google.com/spreadsheets/d/1mU2brATV_fgd5MR
#data = r.content
```

[324]:
```
#df = pd.read_csv(BytesIO(data), index_col=0)
```

*Import dataset from Digital Policies Frameworks *

[325]:
```
### 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')
```

[326]:
```
Database.head()
```

Out[326]:

	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

[327]:
```
Database.shape
```

Out[327]: (405, 41)

[328]:
```
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
```

columns in dataset

columns in dataset

- 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

[329]:

```
#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))
```

[330]:
```
missing_data(Database)
```

Out[330]:

	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	

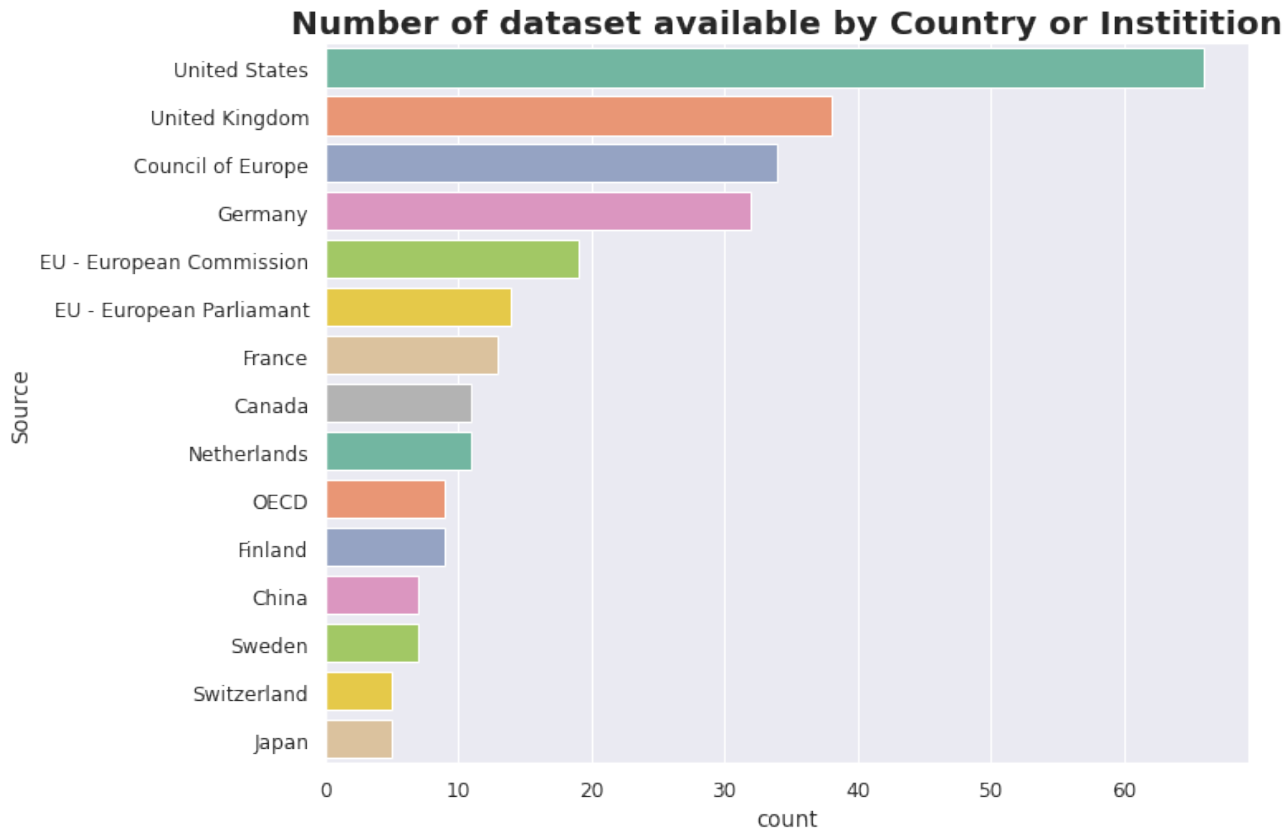
[331]:
```
Database[Database.duplicated()== True]
```

Out[331]:

	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

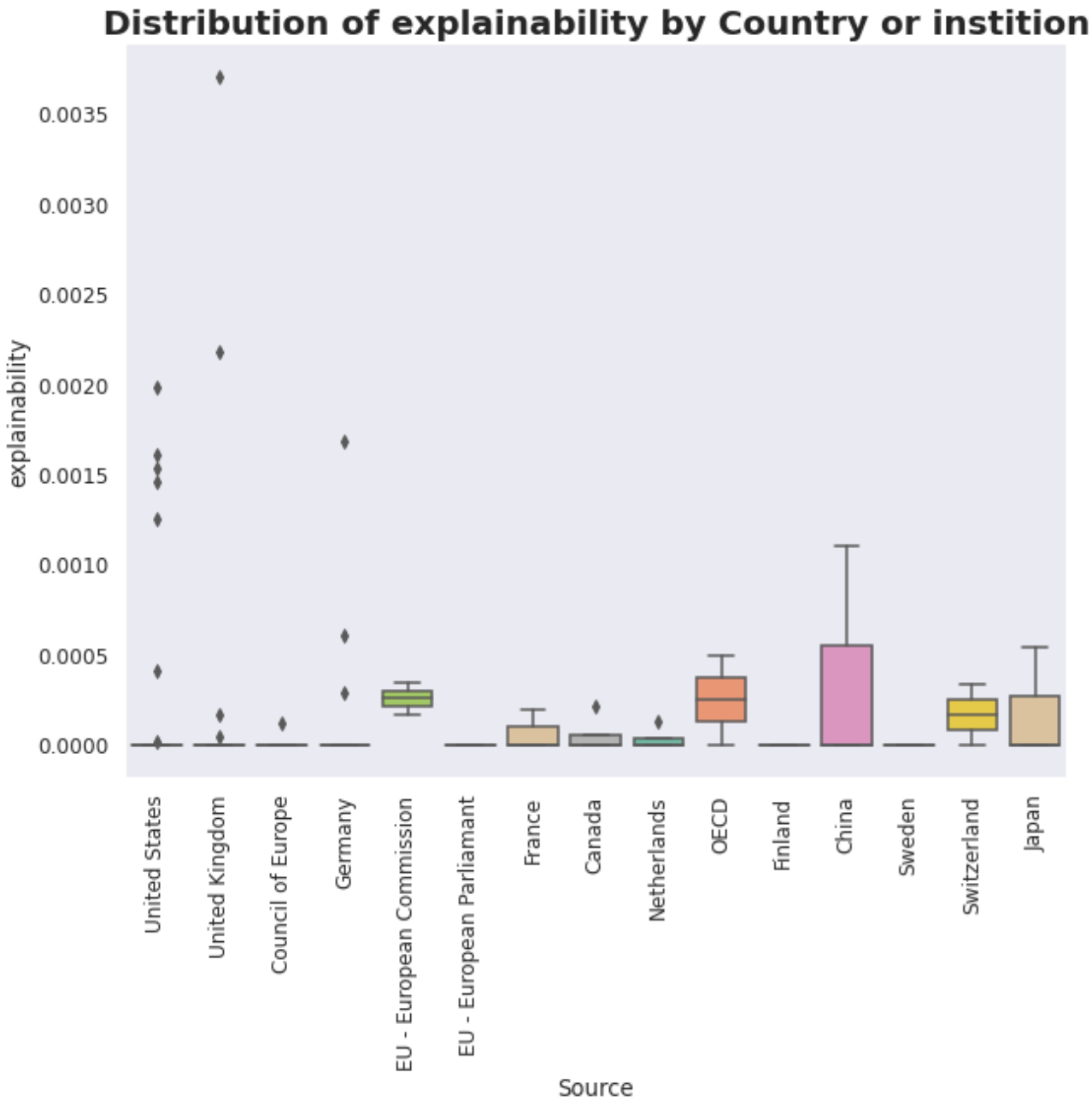
[332]:

```
#Source

ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Number of dataset available by Country or Instititon', fontwe
ax = sns.countplot(y = 'Source', data = Database, order = Database['Sourc
plt.savefig('dataset.png')
```



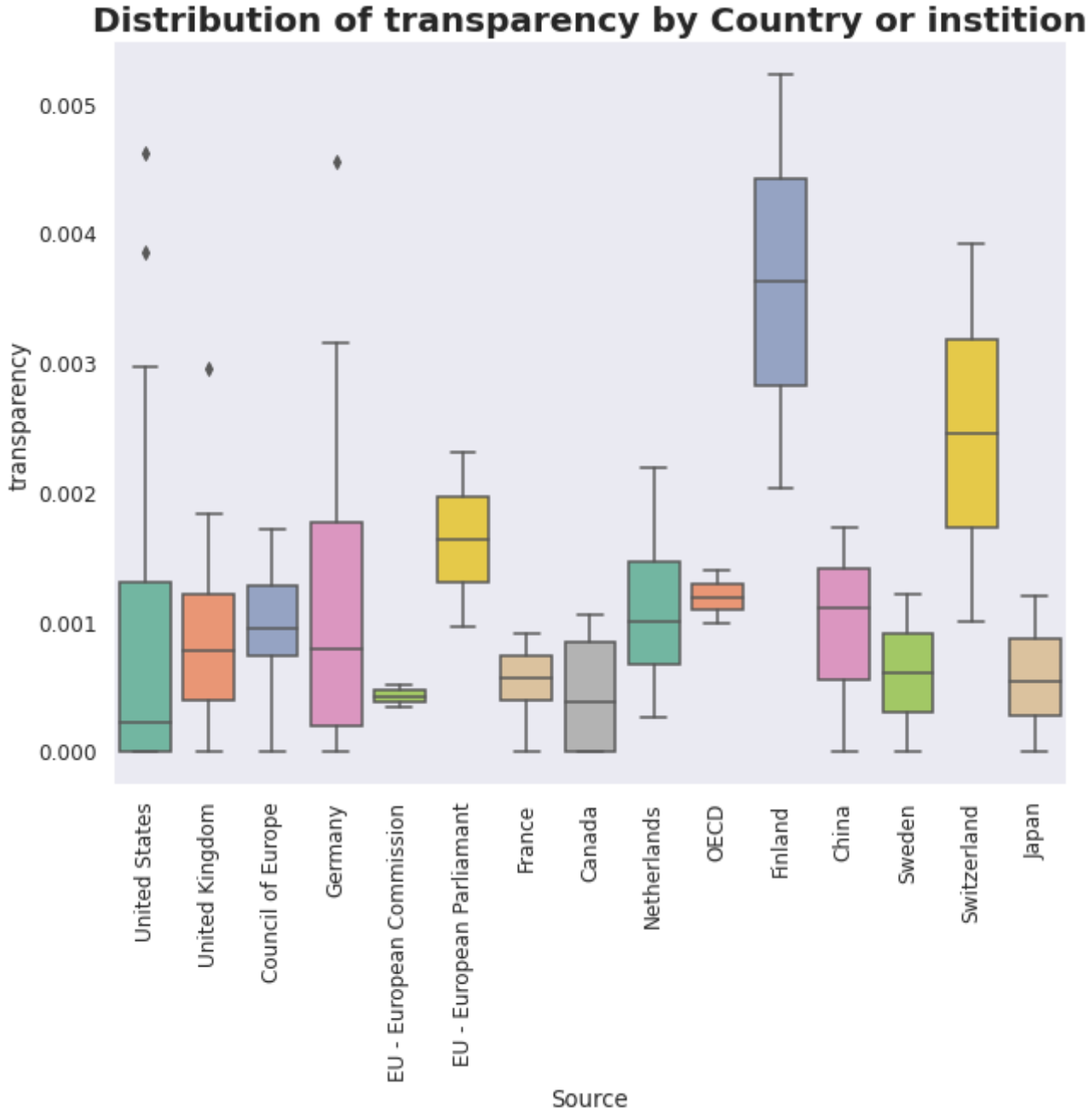
[333]:

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

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



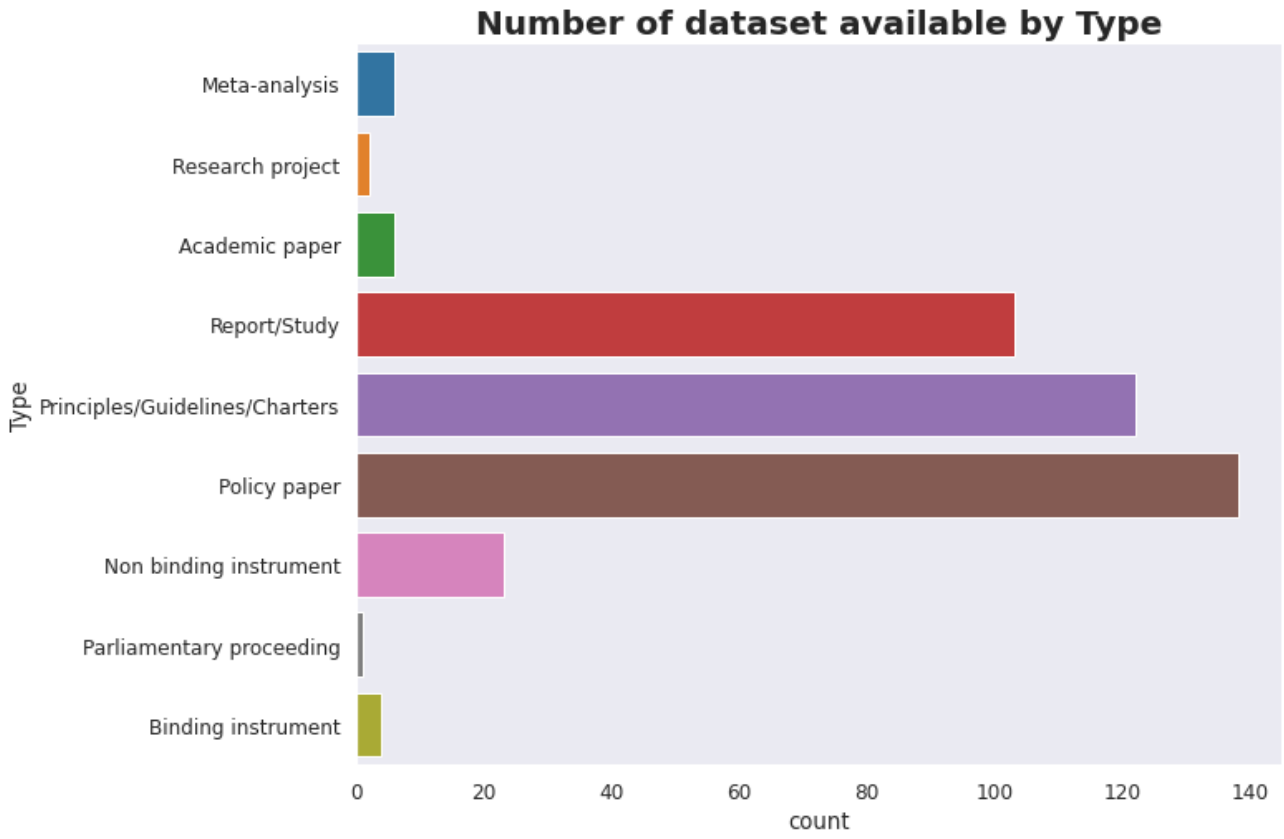
[334]:

```
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')
```



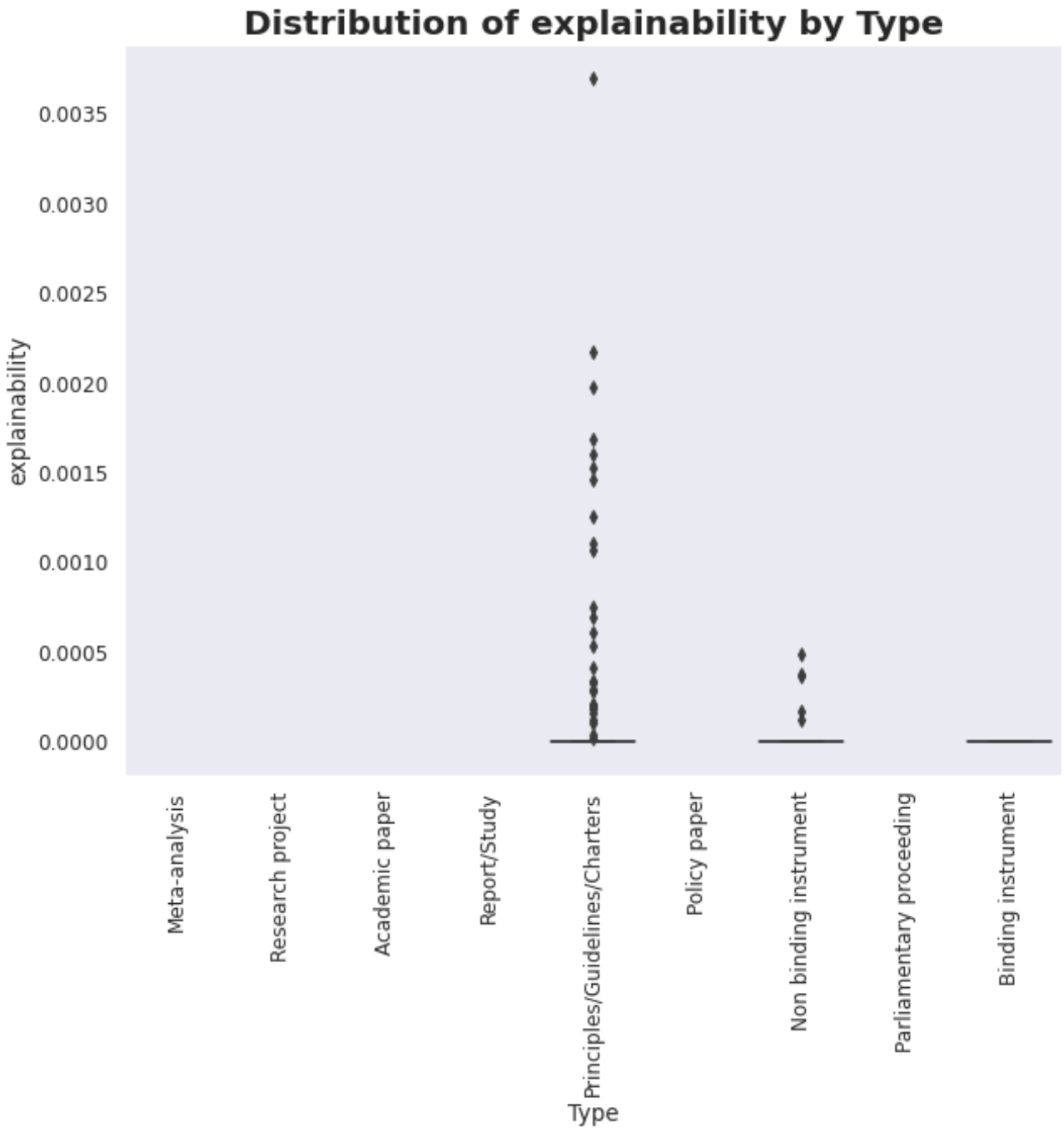
[335]:

```
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')
```



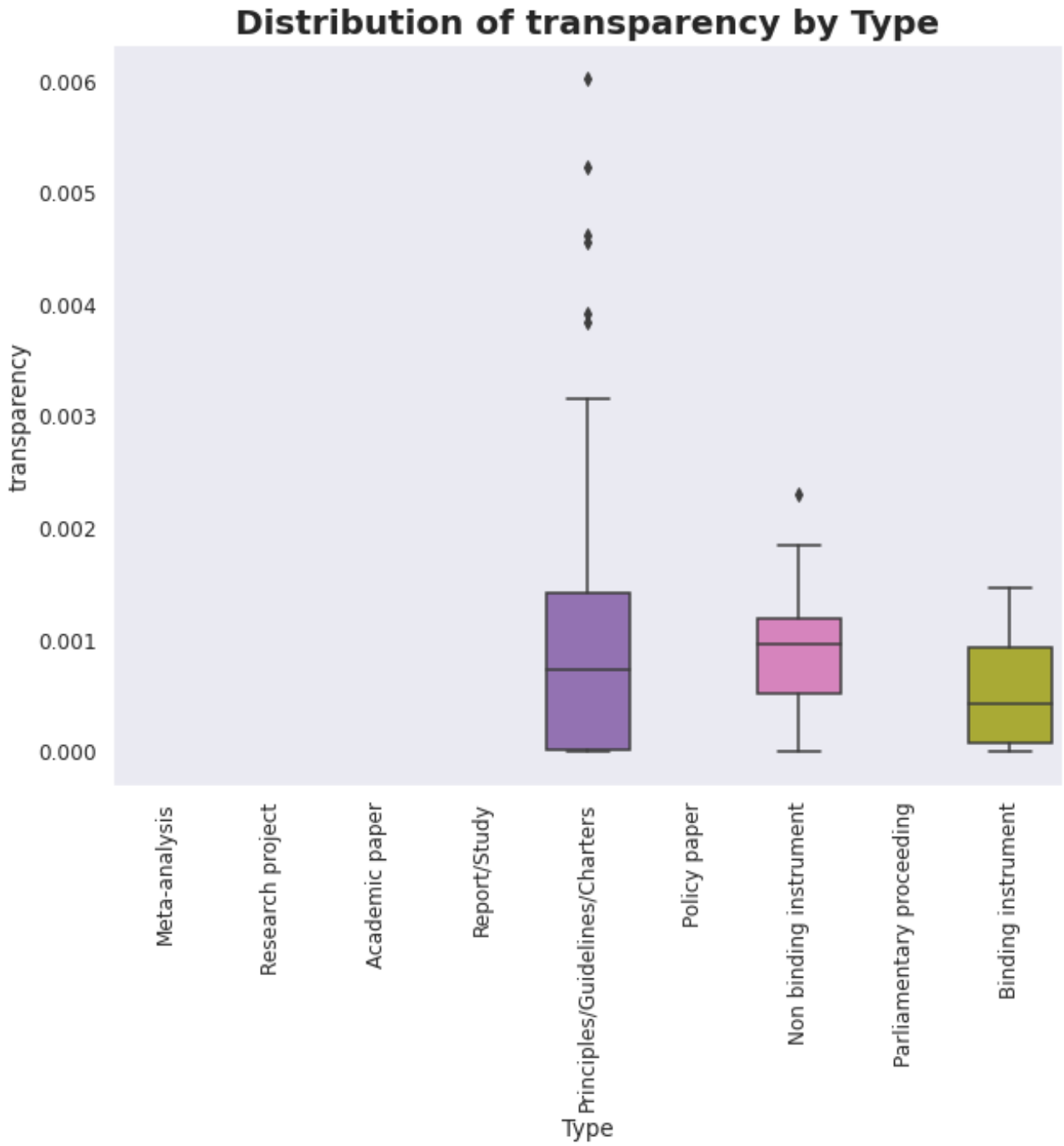
[336]:

```
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')
```



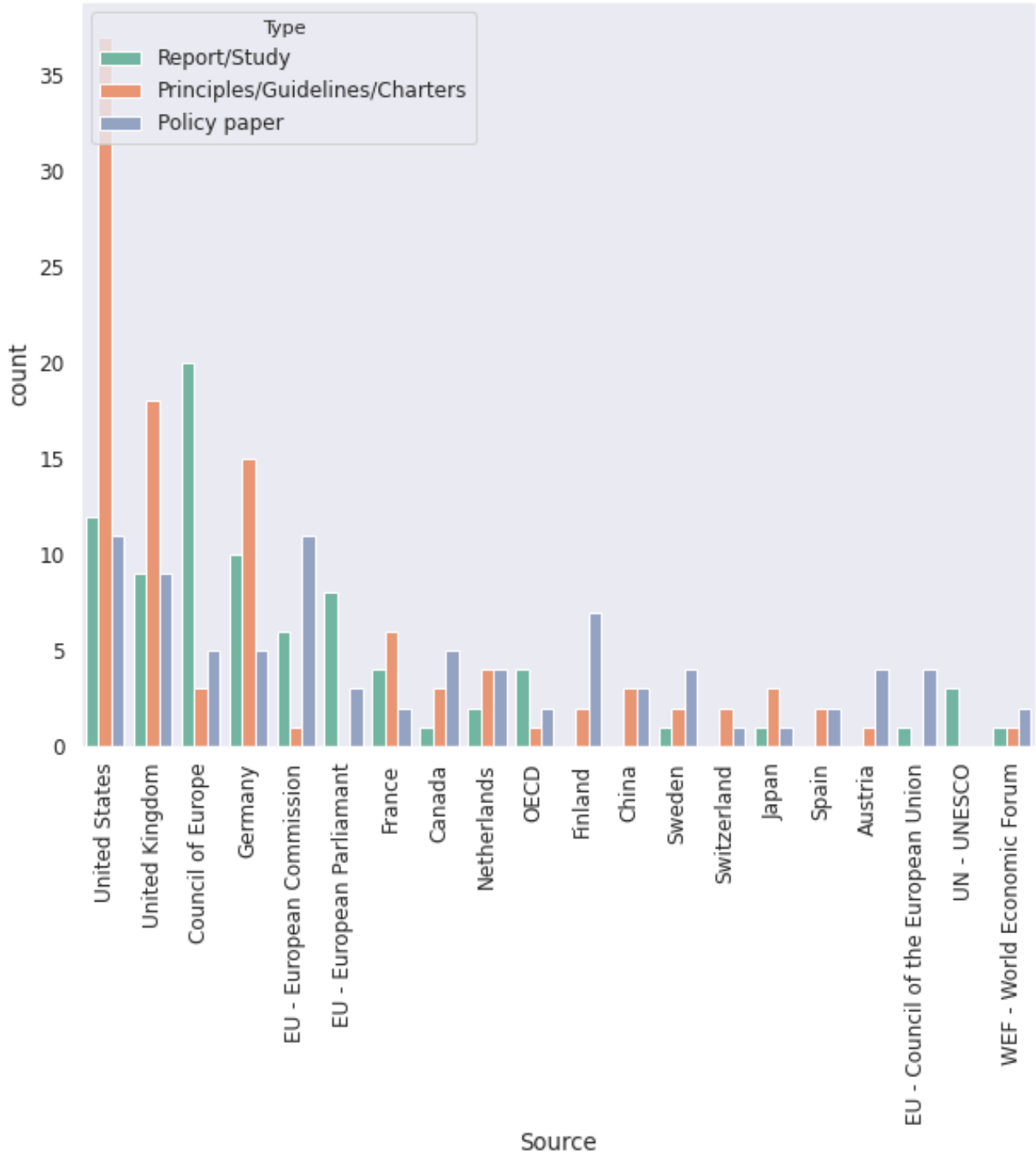
[337]:

```
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')
```



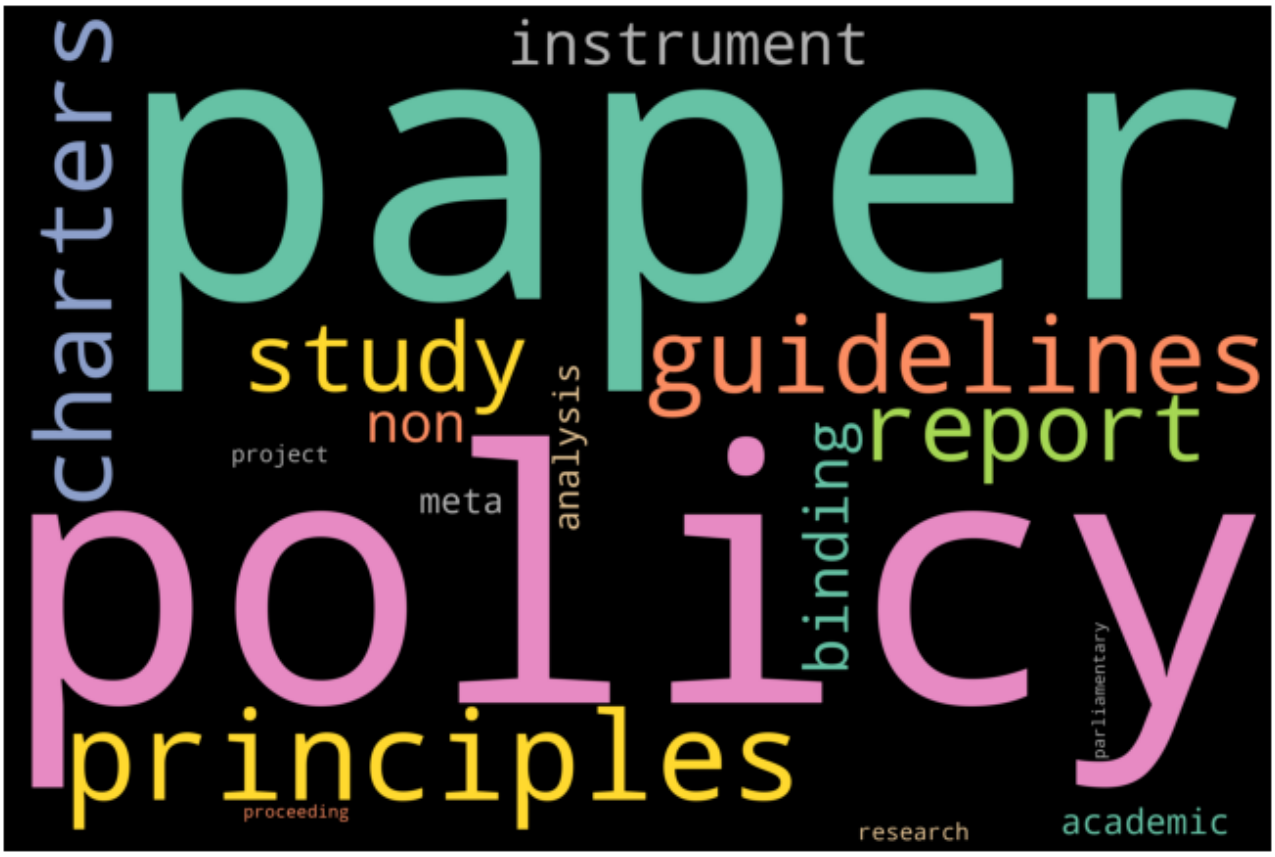
[ ]:

[338]:
```
lus100 = Database['Type'].map(Database['Type'].value_counts()) > 100 # More

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

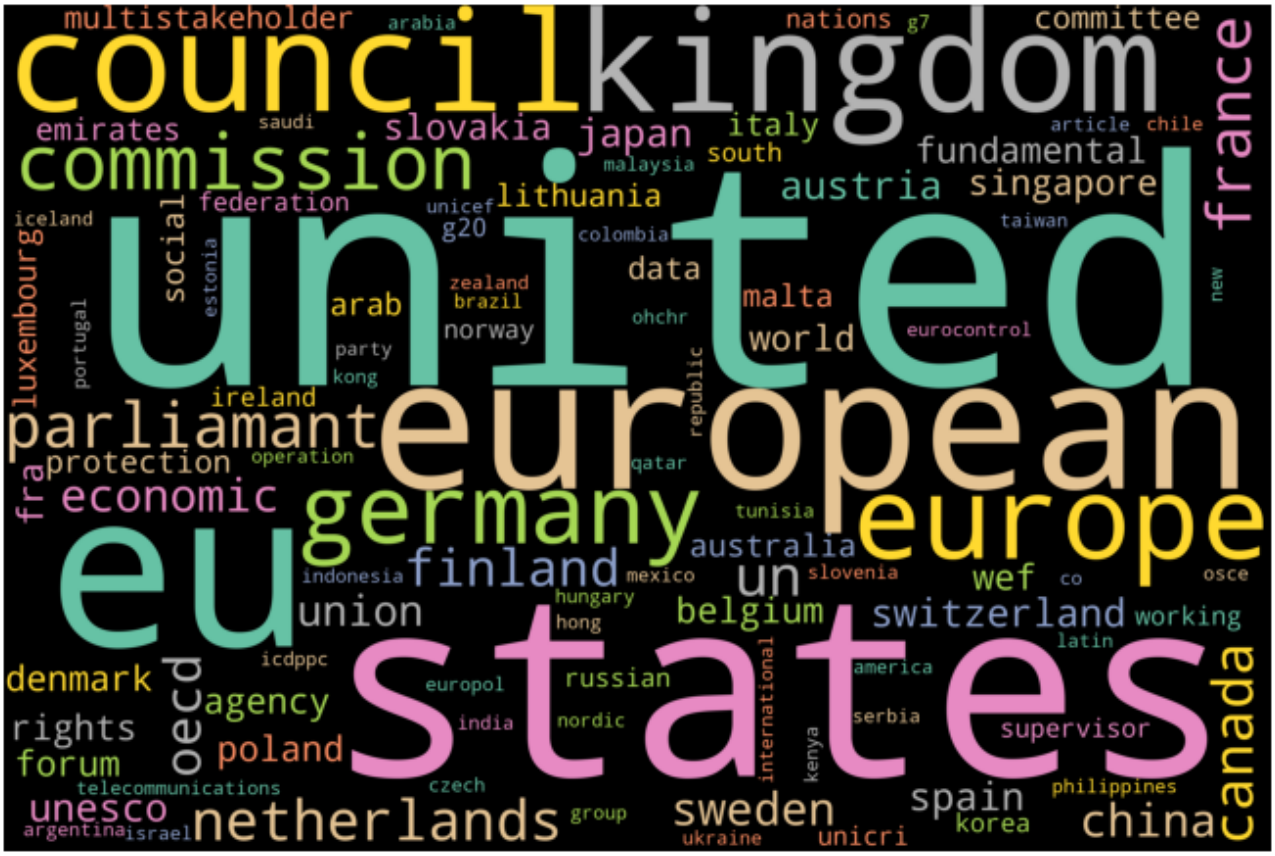


[339]:
```
ltk.tokenize import RegexpTokenizer
ltk.corpus import stopwords
ordcloud import WordCloud

rds = stopwords.words('english')
zer = RegexpTokenizer(r'\w+')

t_wordcloud(series): #simple function to tokenize and plot a said column
rd_cloud = ''

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

ordcloud = WordCloud(height=800,margin=1,max_words=500, colormap='Set1').ge
ordcloud = WordCloud( width = 3000, height = 2000, random_state=1, backgrou

rdcloud = WordCloud(width = 3000, height = 2000, random_state=1,
                    background_color='black', colormap='Set2', collocations

t.imshow(wordcloud)
t.axis("off")
t.tight_layout(pad = 0)
lt.savefig('Plotly-World_Cloud.png')
```

[340]:
```
Type = Database['Type'].apply(lambda x: x.lower())

plt_type = get_wordcloud(Type)

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

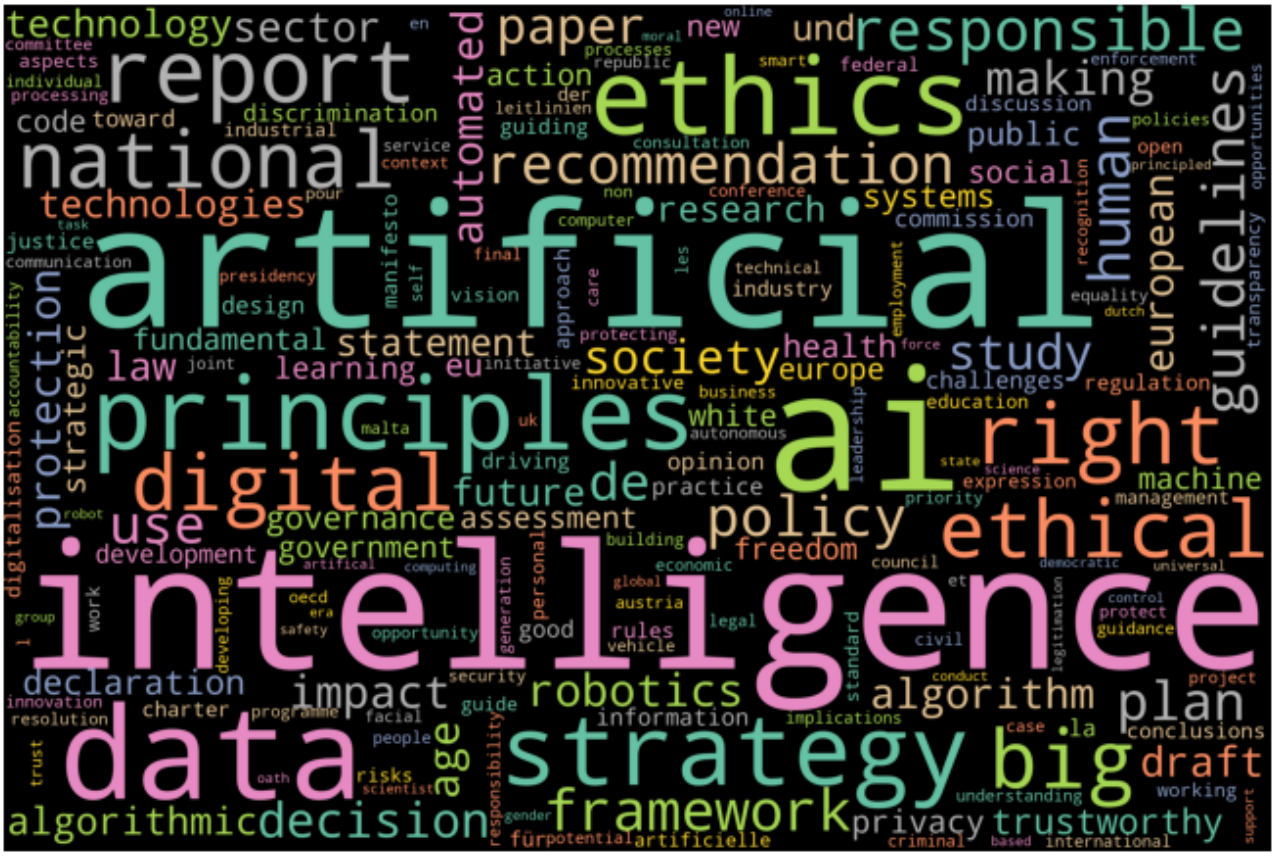


[341]:
```
Source = Database['Source'].apply(lambda x: x.lower())

plt_source = get_wordcloud(Source)

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



[342]:

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

plt_Reference = get_wordcloud(Reference)

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



[343]:

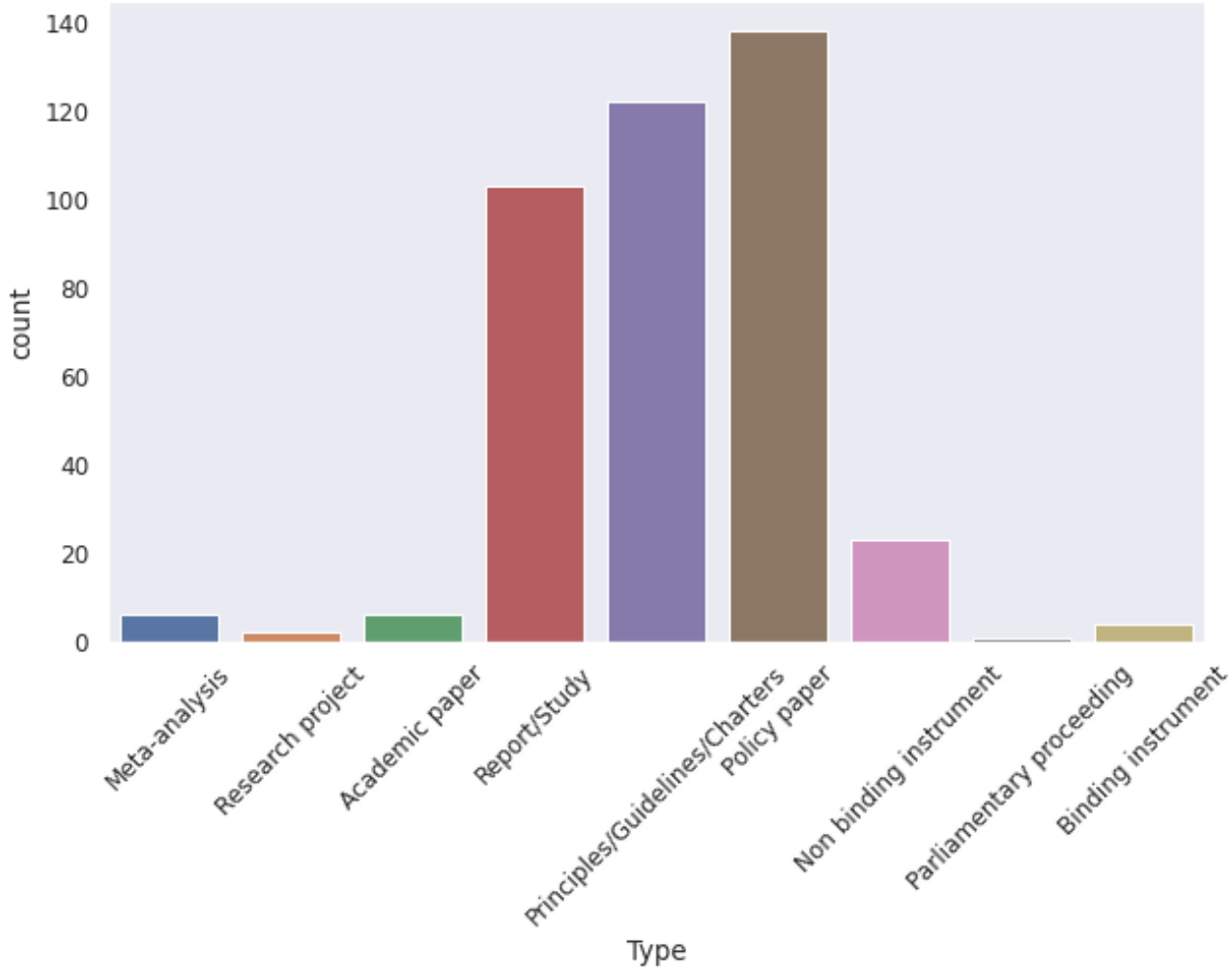
```
Origin = Database['Origin'].apply(lambda x: x.lower())

plt_origin = get_wordcloud(Origin)

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

[344]:

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

plt_Issuer = get_wordcloud(Issuer)

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

[345]:

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



[346]:
```
Database1
```

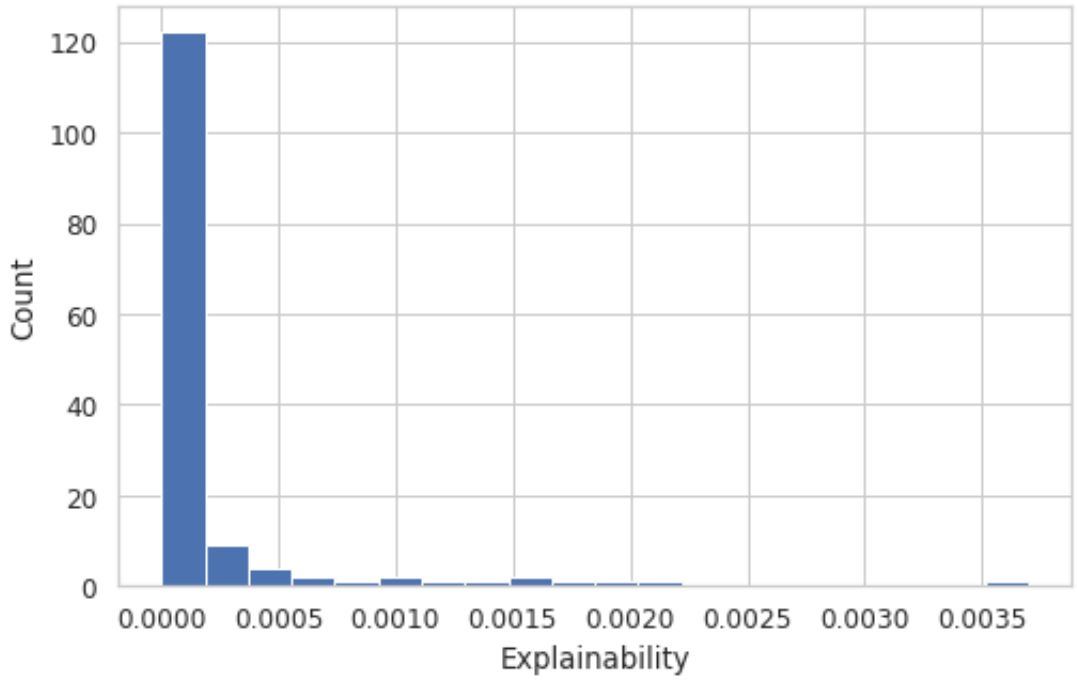
Out[346]:

	Issuer	Reference	Type	
9	National Academies of Science, Engineering & M...	Data Science Oath	Principles/Guidelines/Charters	https://www.nap.e
11	PLOS Computational Biology	Ten simple rules for responsible Big Data rese...	Principles/Guidelines/Charters	https://journals.pl
14	Royal Australian and New Zealand College of Ra...	RANZCR Ethical Principles for AI in Medicine -...	Principles/Guidelines/Charters	https://www.ranzcr.c

[347]:
```
merical = ['fundamental rights','human agency','human rights','non discrim

ule of law','sustainable development','well being','accountability','auton
'dignity' ,'diversity','explainability','fairness','freedom','inclusive','
privacy','responsibility','robustness','safe','solidarity','sustainability

tegorical = ['Issuer','Reference','Type','Link','Origin','Source','CoE MS'

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

Out[347]: (405, 40)

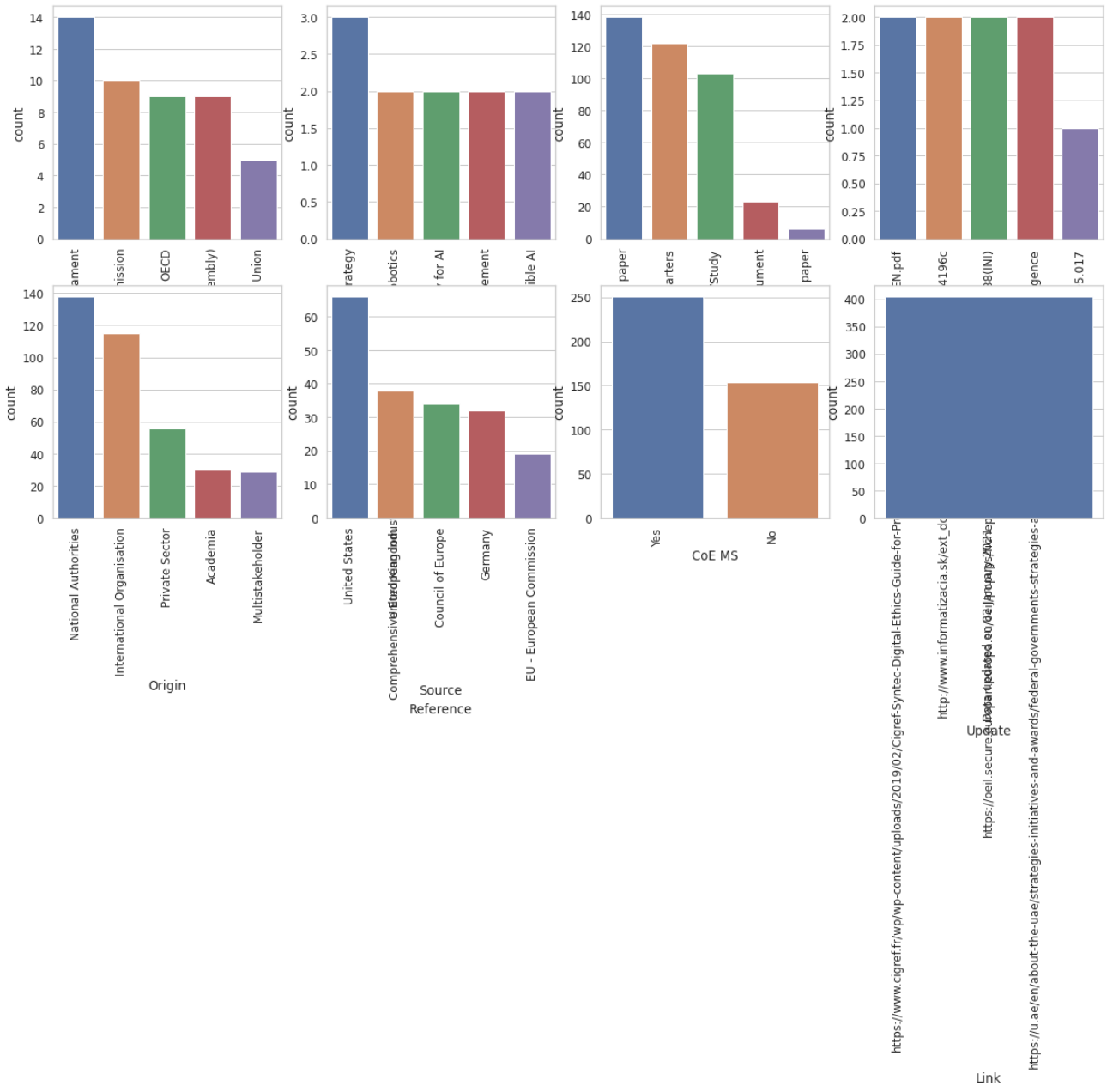
[348]:
```
sns.set(style='whitegrid', palette="deep", font_scale=1.1, rc={"figure.fi
sns.distplot(
    Database['explainability'], norm_hist=False, kde=False, bins=20, hist
).set(xlabel='Explainability', ylabel='Count');
```



[349]:
```
# Database[numerical].plot.barh(stacked=True);
```

[350]:

```
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)
```



[351]:
```
#mean=Database1['explainability'].mean()
max=Database1['explainability'].max()
min=Database1['explainability'].min()
```

[352]:
```
max
```

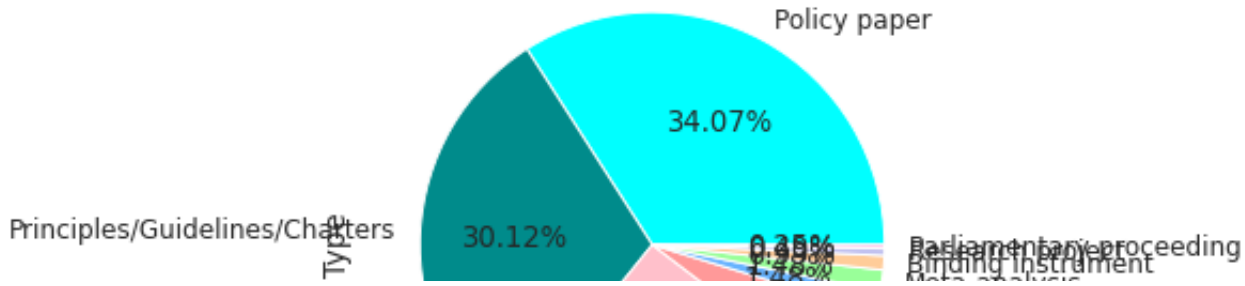
Out[352]: 0.003699137

[353]:
```
min
```

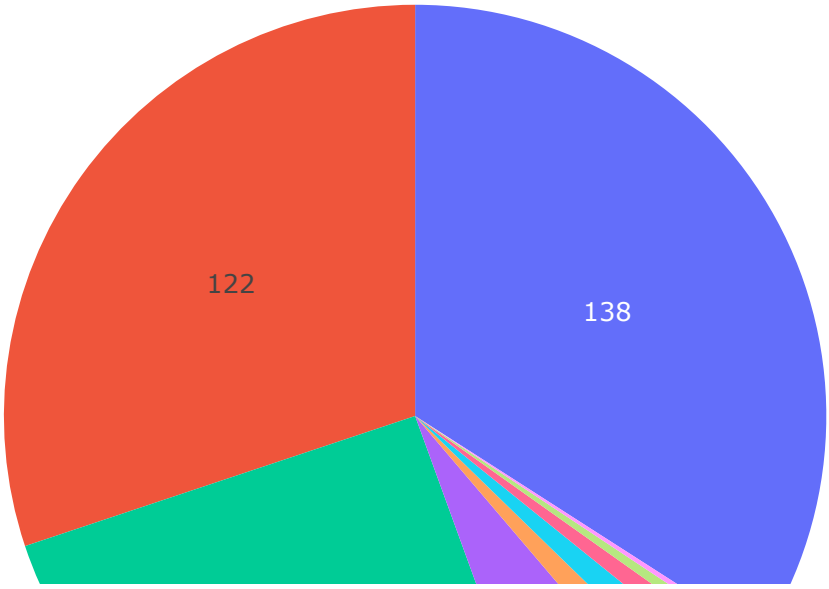
Out[353]: 0.0

[354]:
```
# Plotly Libraris
import plotly.express as px
import plotly.graph_objects as go

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

Out[354]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0a3a67ef90>



[355]:
```
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()
```



[356]:
```
#Database['Type']= Database['Type'].apply(str)

Database['Type'] = Database['Type'].fillna('').apply(str)
```

[357]:
```
dfs = pd.ExcelFile('/kaggle/input/digital-policiers-frameworks/Digital Po

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

Issuer

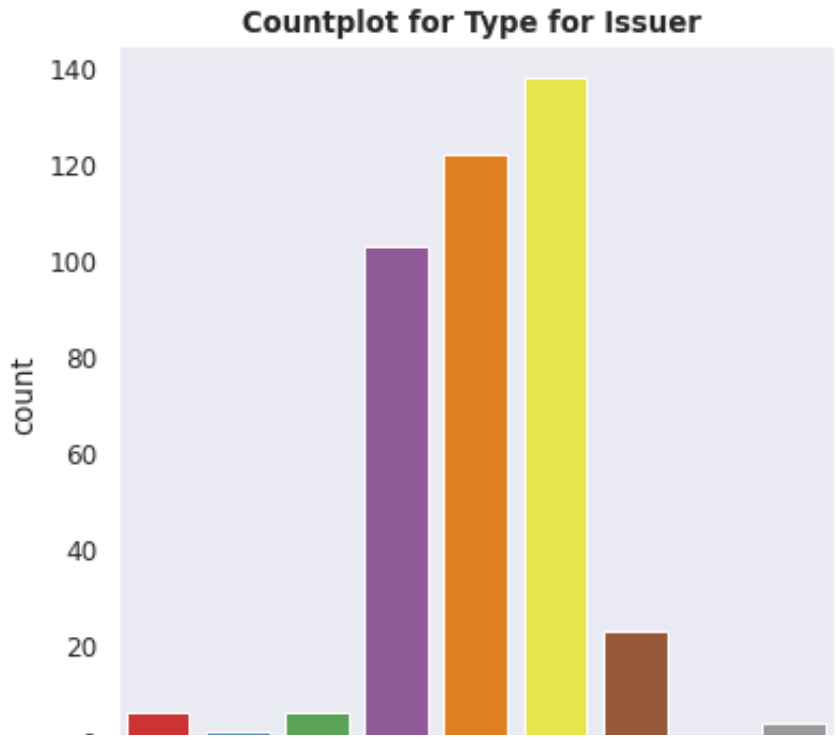
[358]:
```
#Database.head()

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

Out[358]:

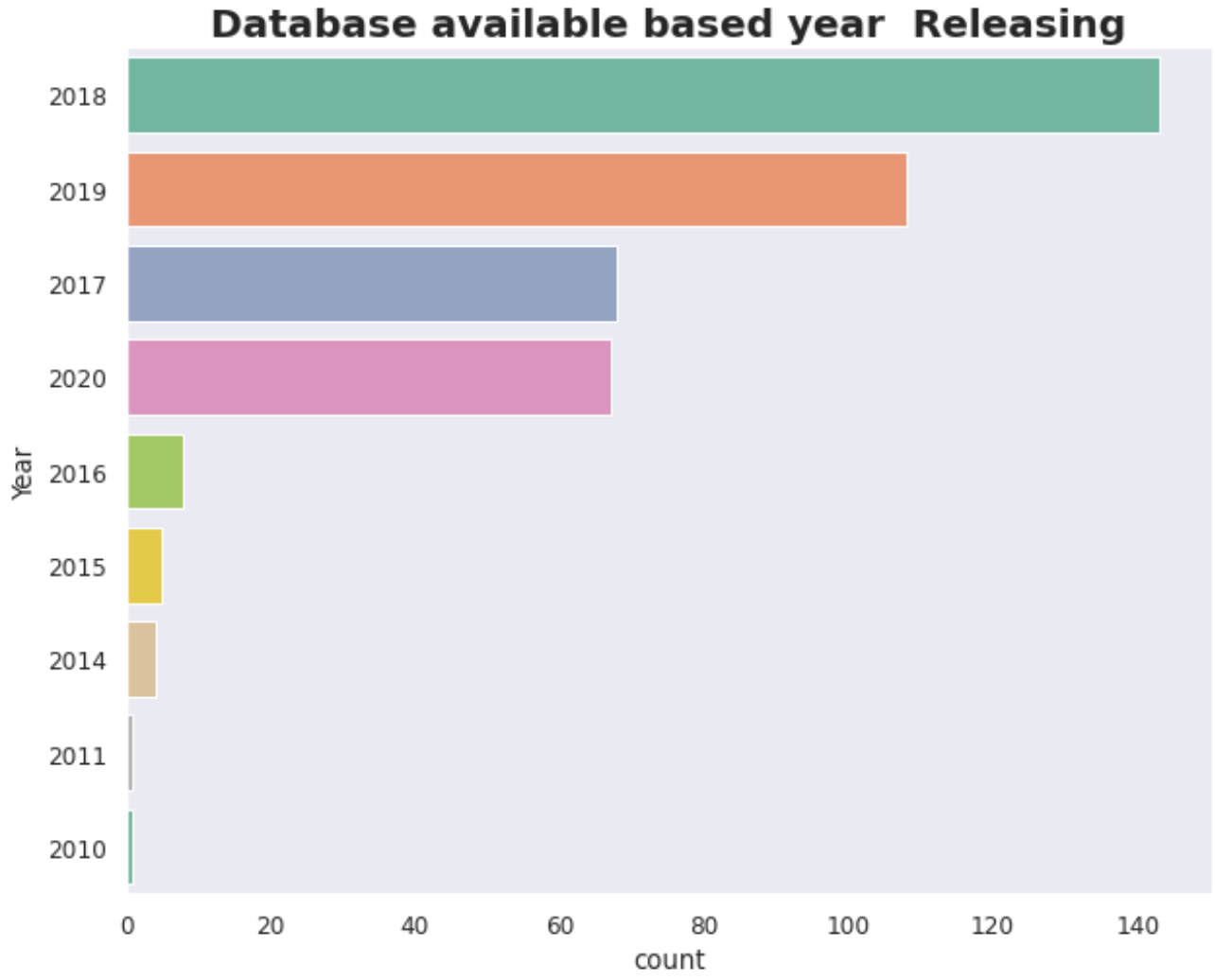
	Issuer	Reference	Type	Link	Origin
0	Berkman Klein Center (University of Harvard)	Principled Artificial Intelligence	Meta-analysis	https://ssrn.com/abstract=3518482	Academia
1	Cyberjustice Laboratory	ACT Project - Projet AJC (Autonomisation des a...	Research project	https://www.ajcact.org	Academia
2	ETH 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

[359]:
```
# 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[359]:
```
[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')]
```



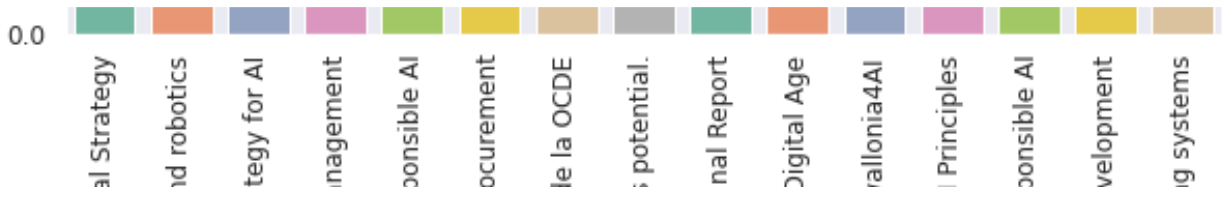
[360]:
```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title('Database available based year  Releasing', fontweight='bold',
ax = sns.countplot(y = 'Year', data = data, order = data['Year'].value_co
```

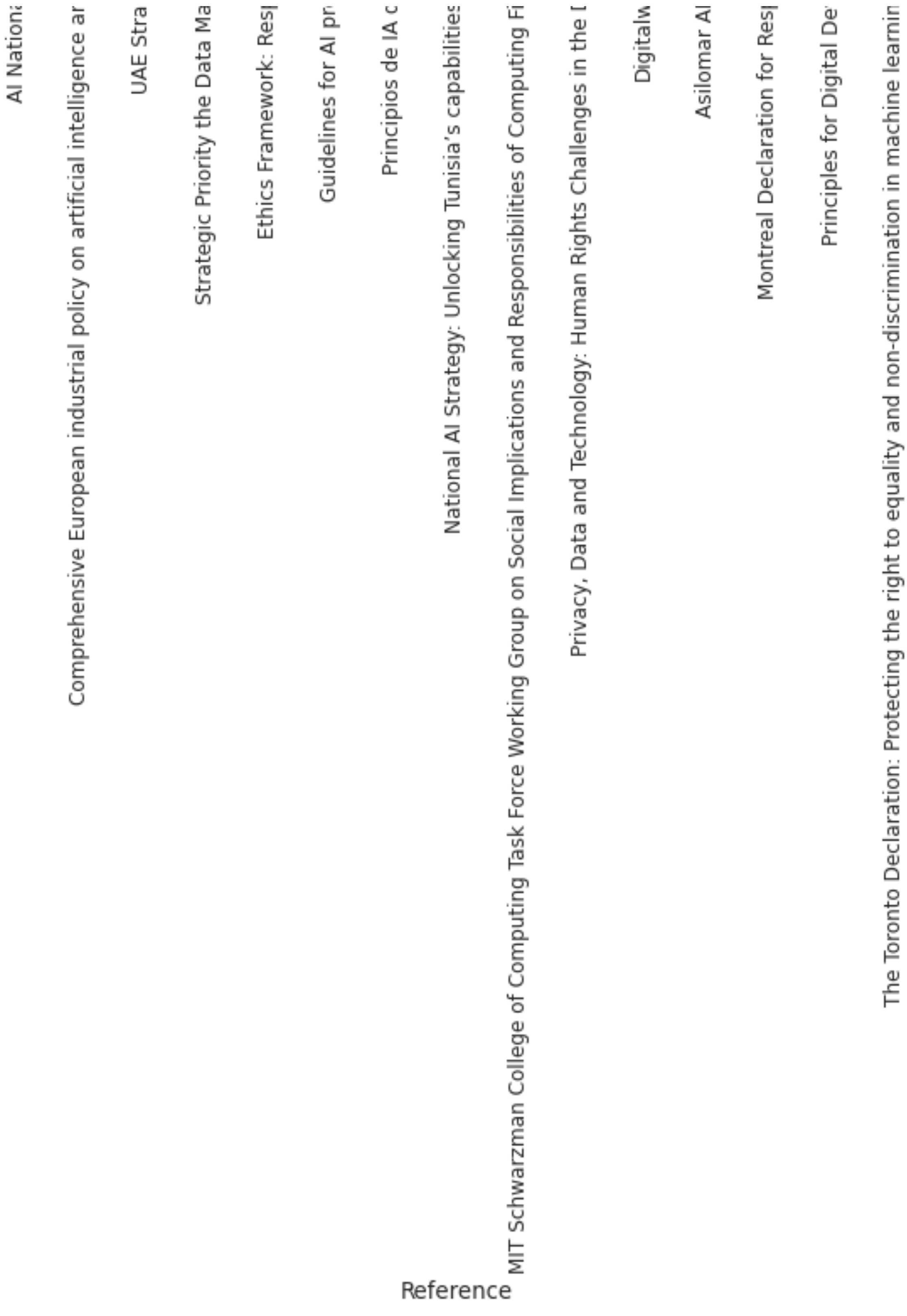


[361]:
```
ax = plt.subplots(figsize = (10, 8))
sns.set_style('dark')
plt.title(' Data issuer  Reference', fontweight='bold', fontsize=20)
ax = sns.countplot(x = 'Reference', data = data, palette = 'Set2', order
#ax.set_xticklabels(labels, rotation=45)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

Out[361]:[Text(0, 0, 'AT National Strategy')

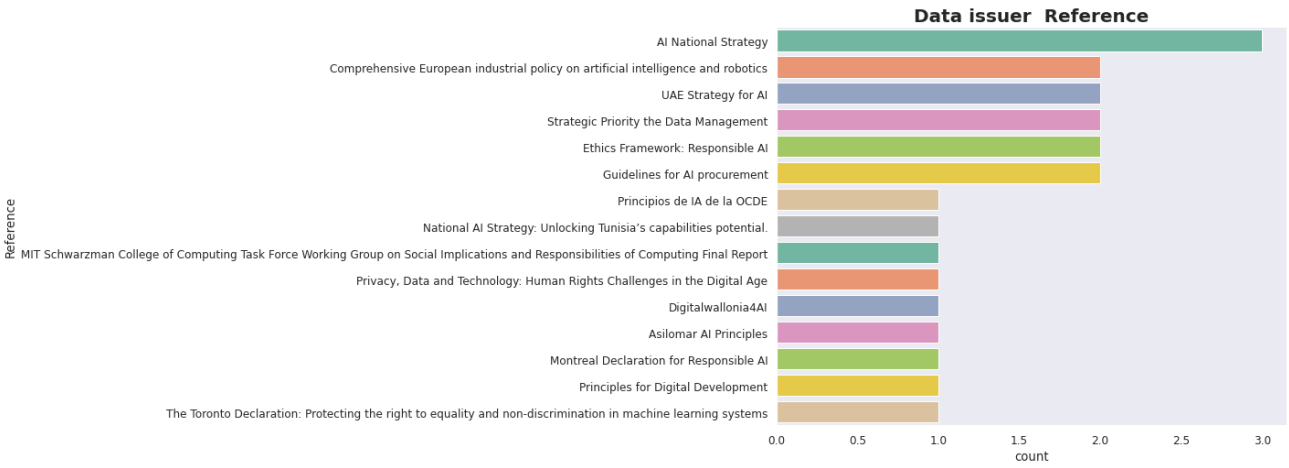
```
 Text(0, 0, 'Comprehensive European industrial policy on artificial intellige
nce and robotics'),
 Text(0, 0, 'UAE Strategy for AI'),
 Text(0, 0, 'Strategic Priority the Data Management'),
 Text(0, 0, 'Ethics Framework: Responsible AI'),
 Text(0, 0, 'Guidelines for AI procurement'),
 Text(0, 0, 'Principios de IA de la OCDE'),
 Text(0, 0, 'National AI Strategy: Unlocking Tunisia’s capabilities potential
.'),
 Text(0, 0, 'MIT Schwarzman College of Computing Task Force Working Group on
Social Implications and Responsibilities of Computing Final Report'),
 Text(0, 0, 'Privacy, Data and Technology: Human Rights Challenges in the Dig
ital Age'),
 Text(0, 0, 'Digitalwallonia4AI'),
 Text(0, 0, 'Asilomar AI Principles'),
 Text(0, 0, 'Montreal Declaration for Responsible AI'),
 Text(0, 0, 'Principles for Digital Development'),
 Text(0, 0, 'The Toronto Declaration: Protecting the right to equality and no
n-discrimination in machine learning systems')]
```




AI Nationa
Comprehensive European industrial policy on artificial intelligence ar
UAE Stra
Strategic Priority the Data Ma
Ethics Framework: Resp
Guidelines for AI pr
Principios de IA c
National AI Strategy: Unlocking Tunisia's capabilities
MIT Schwarzman College of Computing Task Force Working Group on Social Implications and Responsibilities of Computing Fi
Privacy, Data and Technology: Human Rights Challenges in the D
Digitalw
Asilomar AI
Montreal Declaration for Resp
Principles for Digital De
The Toronto Declaration: Protecting the right to equality and non-discrimination in machine learnin
Reference

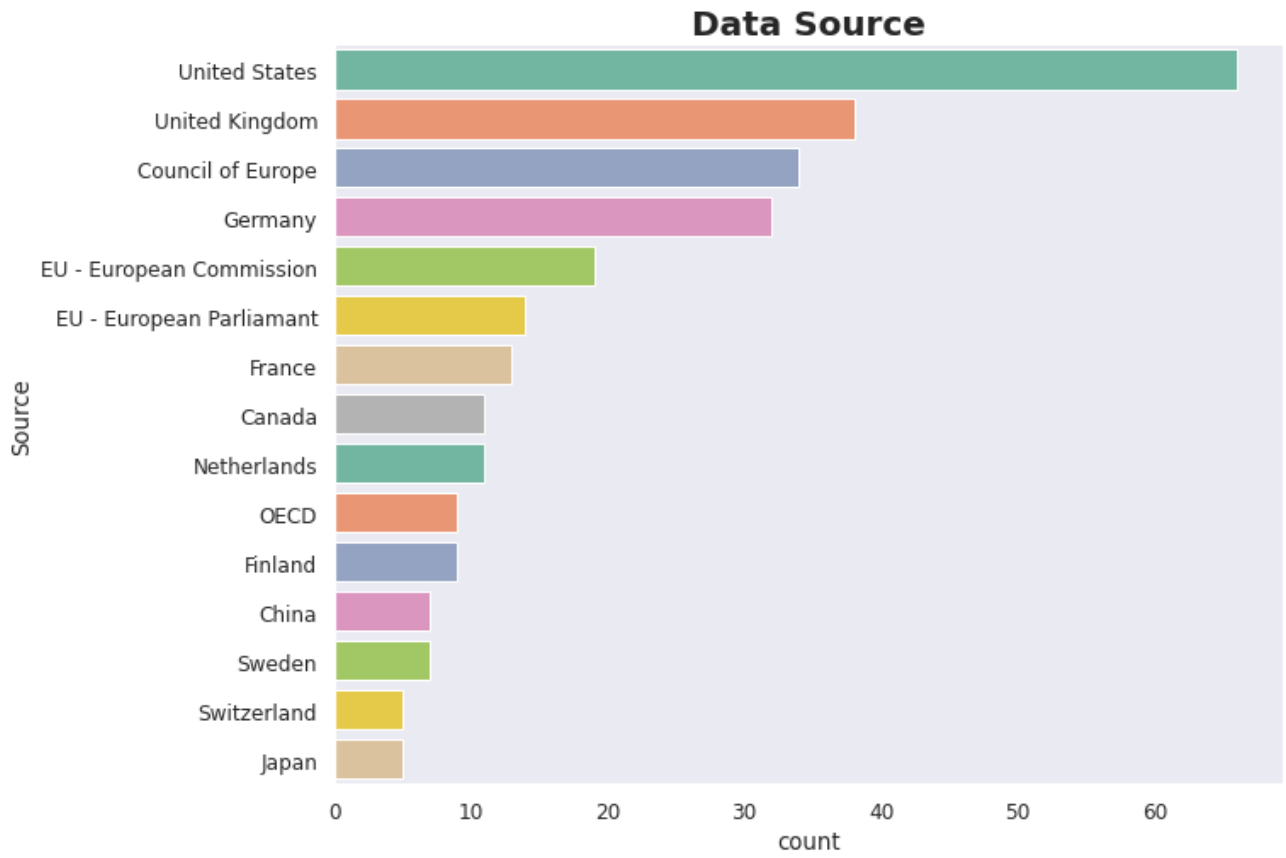
[362]:

```
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'
```



[363]:

```
#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
```

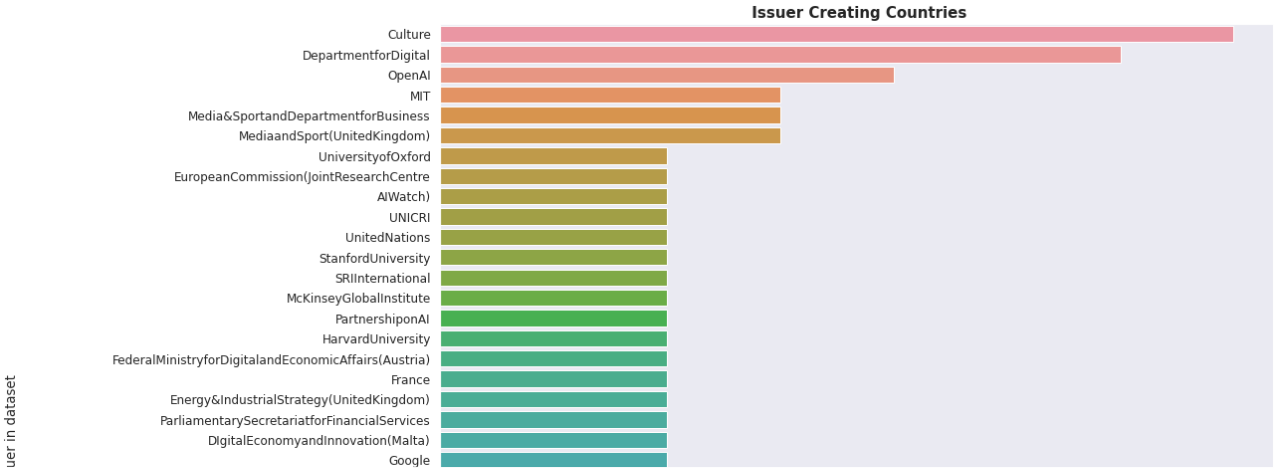


[364]:
```
# More Issuer content creating countries

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
```

[365]:
```
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
```

[366]:
```
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[366]: Text(0, 0.5, 'Issuer in dataset')

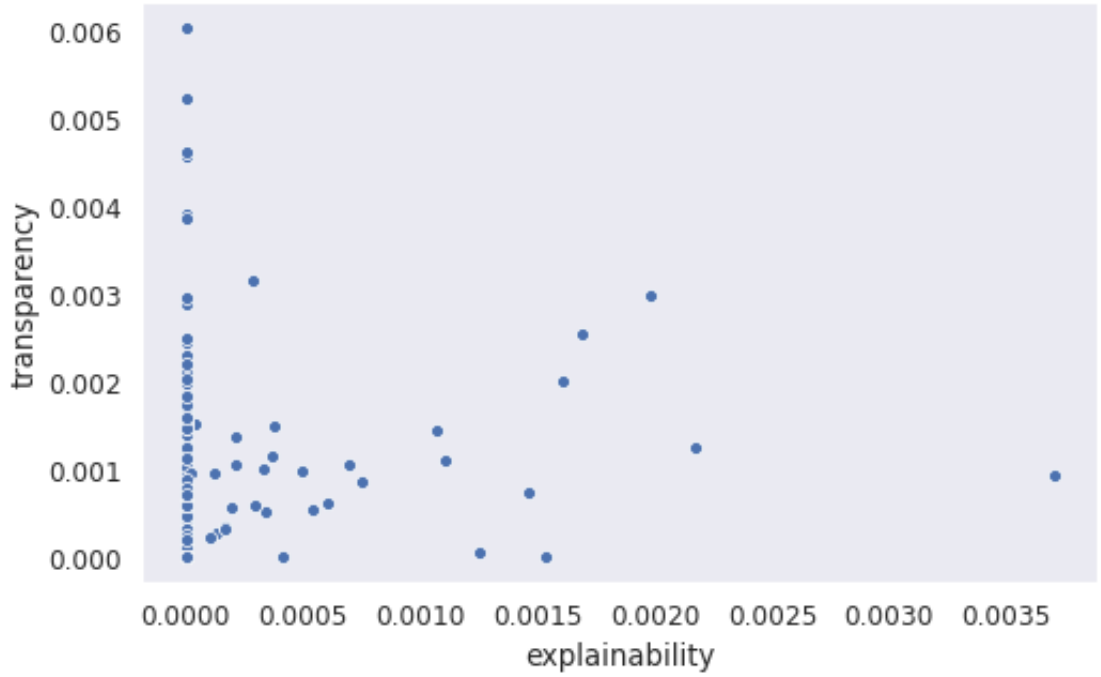


[367]:
```
data[data['Year'] == 2020].groupby('Type')['Year'].count()
```

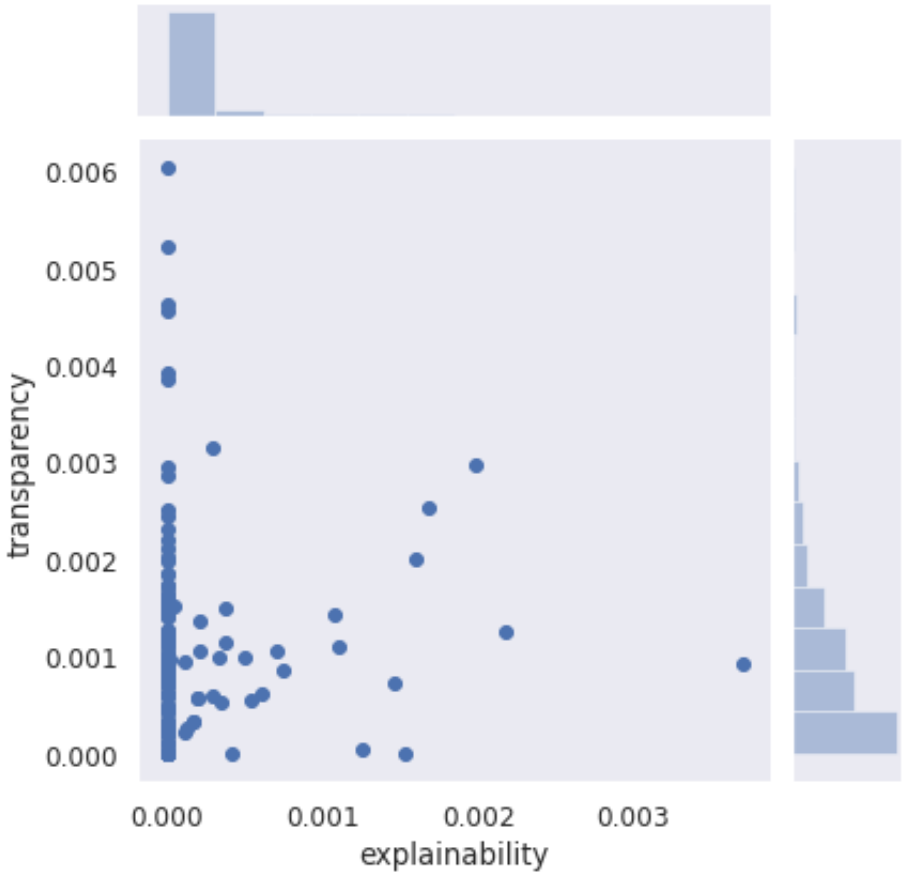
Out[367]:
```
Type
Binding instrument                1
Meta-analysis                     2
Non binding instrument            3
Policy paper                     21
Principles/Guidelines/Charters    9
Report/Study                     31
Name: Year, dtype: int64
```

Analyzing Relationships Between Numerical Variables

[368]:
```
## relationship between transparency and explainability

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

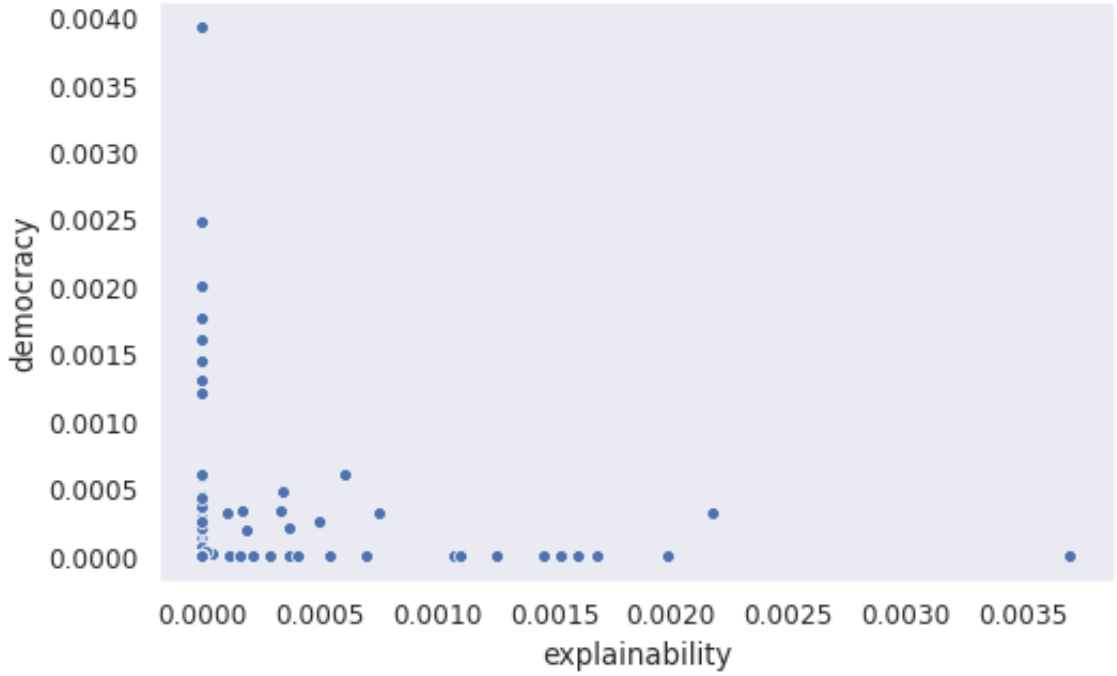


[369]:
```
sns.jointplot(x=data['explainability'], y=data['transparency']);
```

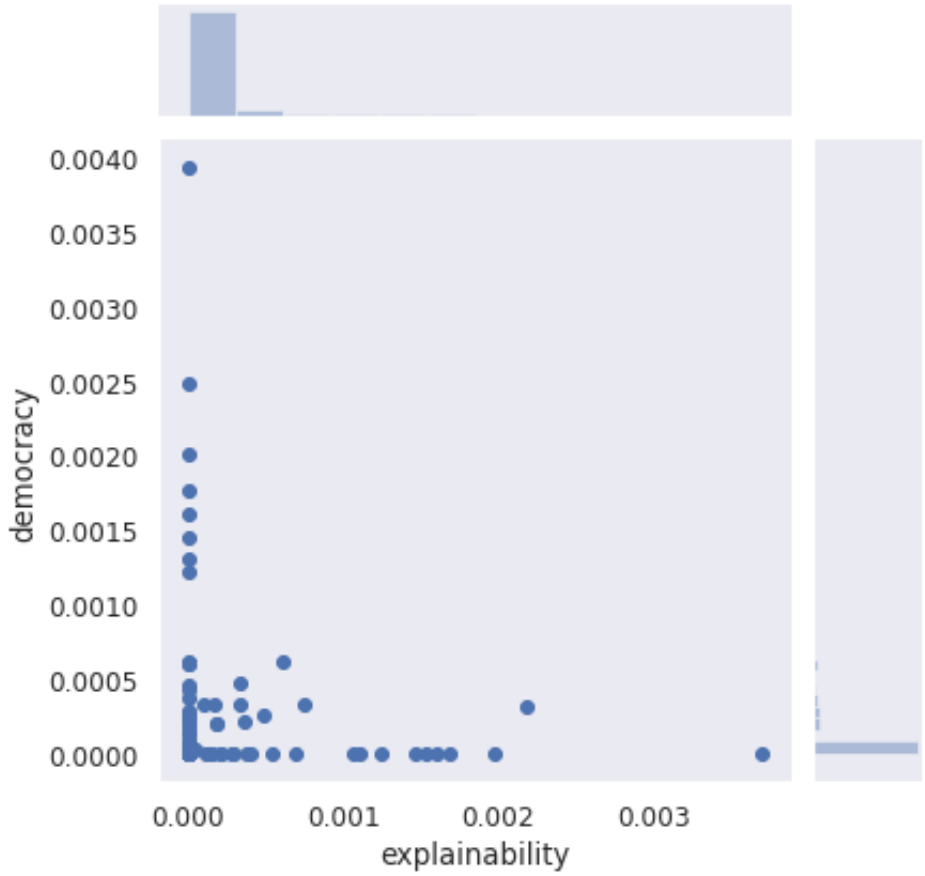


[370]:

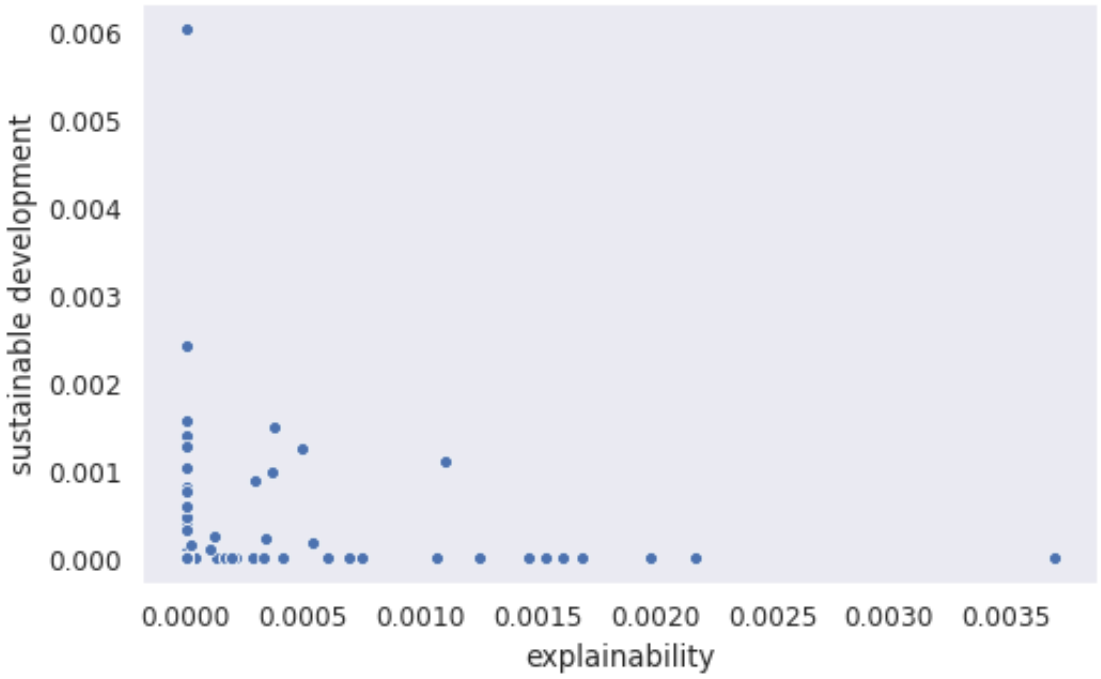
```
## relationship between democracy and explainability

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



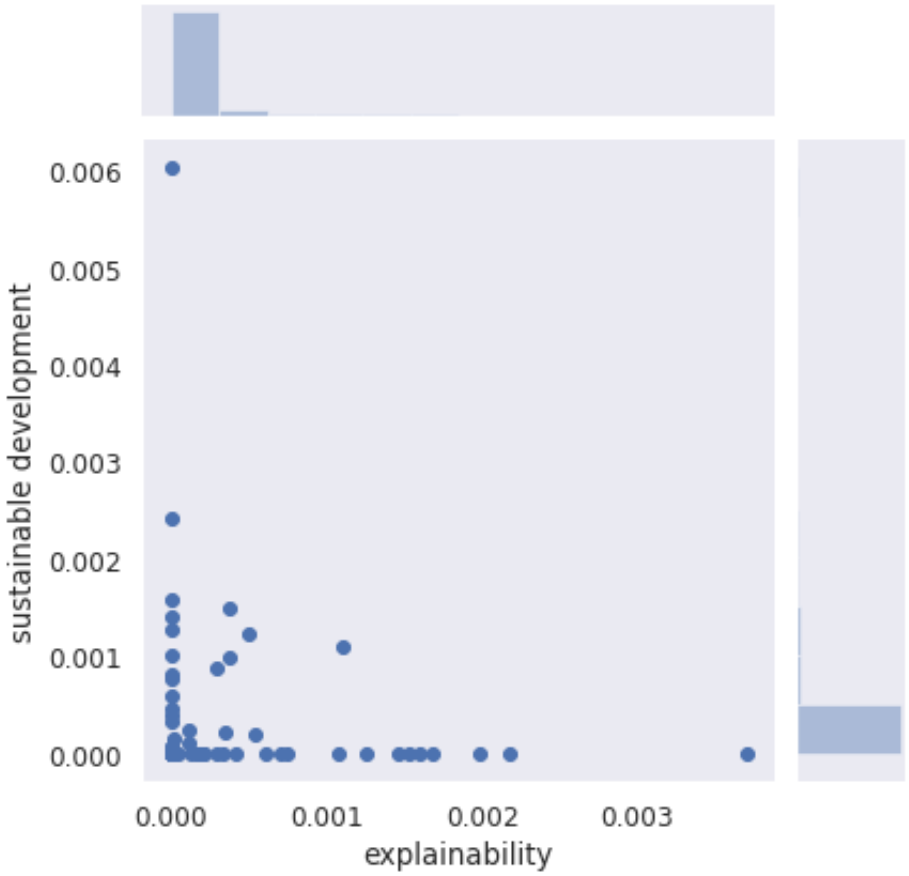
[371]:
```
sns.jointplot(x=data['explainability'], y=data['democracy']);
```



[372]:
```
## relationship between sustainable development and explainability

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

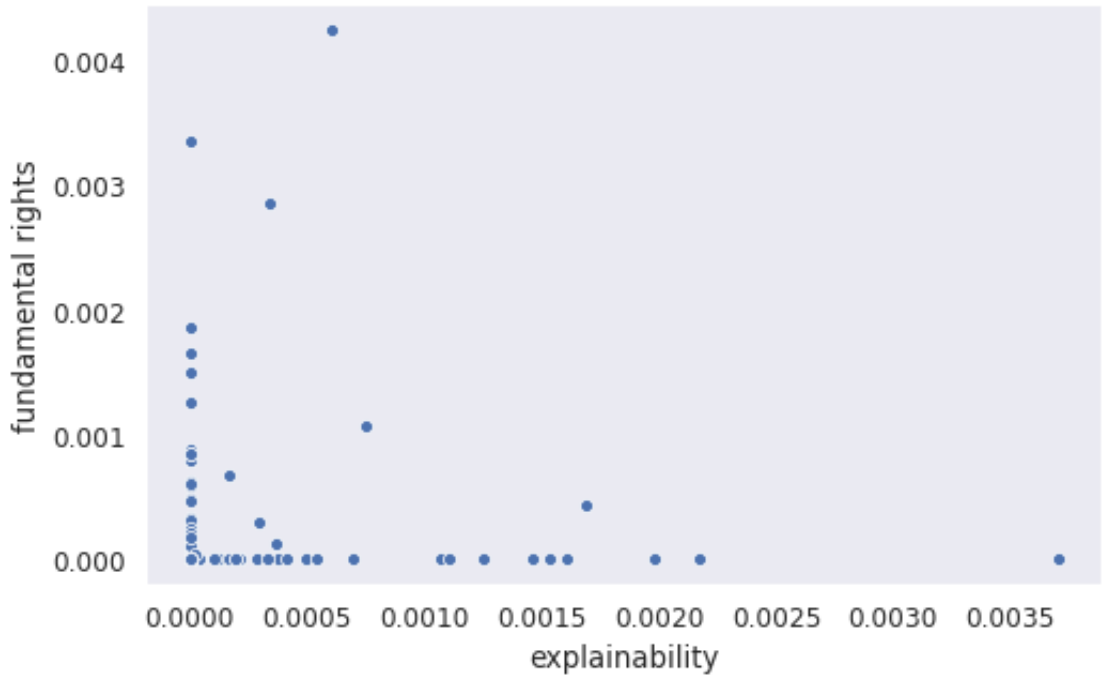


[373]:
```
sns.jointplot(x=data['explainability'], y=data['sustainable development'])
```

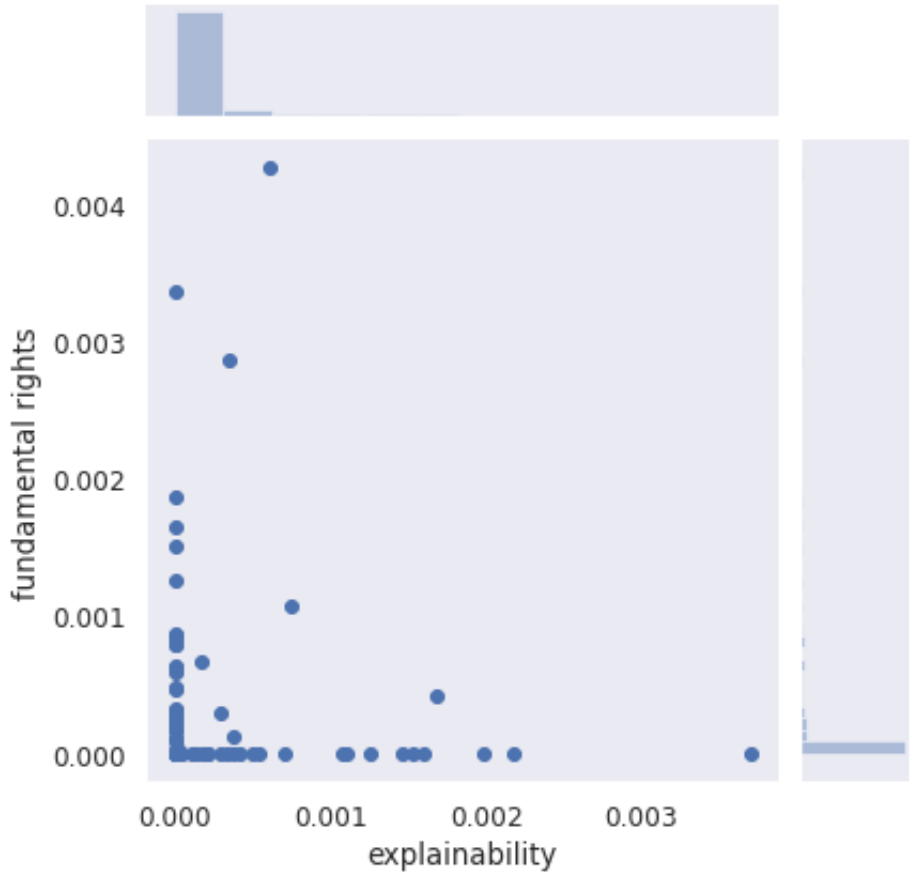


[374]:

```
## relationship between fundamental rights and explainability

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



[375]:
```
sns.jointplot(x=data['explainability'], y=data['fundamental rights']);
```



[376]:
```
#sns.pairplot(Database[numerical], kind="scatter")
#plt.show()
```

Analyzing Relationships Between Numerical and Categorical Variables

[377]:
```
pd.crosstab(Database.Type, Database.Source)
```

Out[377]:

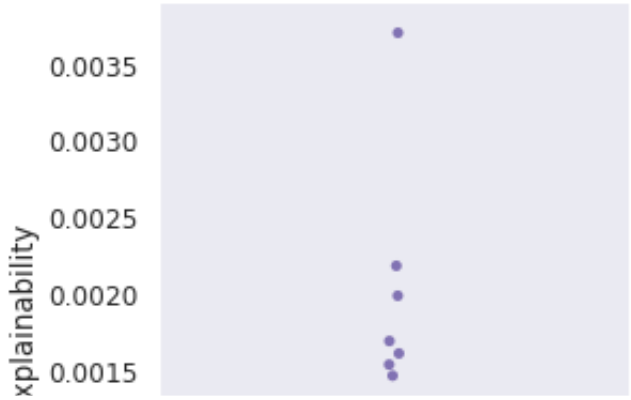
Source	Argentina	Australia	Austria	Belgium	Brazil	Canada	Chile	China
Type								
Academic paper	0	0	0	0	0	0	0	0
Binding instrument	0	0	0	0	0	1	0	0
Meta-analysis	0	0	0	0	0	0	0	1
Non binding instrument	0	0	0	0	0	0	0	0
Parliamentary proceeding	0	0	0	0	0	0	0	0
Policy paper	1	0	4	3	1	5	1	3
Principles/Guidelines/Charters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

[378]:
```
from scipy.stats import chi2_contingency
chi2_contingency(pd.crosstab(Database.Type, Database.Source))
```

Out[378]:
```
(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,
```

[379]:
```
sns.catplot(x="Type", y="explainability", data=Database)
```

Out[379]:<seaborn.axisgrid.FacetGrid at 0x7f0a3f264cd0>



[380]:
```
from sklearn.cluster import KMeans
```

[381]:
```
categorical = Database[categorical]
```

[382]:
```
categorical.describe()
```

Out[382]:

	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

[383]:
```
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
```

[384]:
```
categorical.isnull().sum()*100/categorical.shape[0]
```

Out[384]:
```
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

[385]:
```
categorical_copy = categorical.copy()
```

[386]:
```
from sklearn import preprocessing
from kmodes.kmodes import KModes
le = preprocessing.LabelEncoder()
categorical = categorical.apply(le.fit_transform)
categorical.head()
```

Out[386]:

	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

[387]:
```
#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
```

[388]:
```
# Predicted Clusters
fitClusters_cao
```

Out[388]:
```
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)
```

[389]:
```
clusterCentroidsDf = pd.DataFrame(km_cao.cluster_centroids_)
clusterCentroidsDf.columns = categorical.columns
```

[390]:
```
# Mode of the clusters
clusterCentroidsDf
```

Out[390]:

	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

[391]:
```
#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: 91, cost: 2280.0
Run 1, iteration: 2/100, moves: 3, cost: 2280.0
```

[392]:
```
# Predicted clusters
fitClusters_huang
```

Out[392]:
```
array([1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 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, 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, 1, 1, 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, 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, 1, 0,
       0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0], dtype=uint16)
```

Choosing K by comparing Cost against each K

[393]:
```
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
```

[394]:
```
y = np.array([i for i in range(1,5,1)])
plt.plot(y,cost)
```

Out[394]:
```
[<matplotlib.lines.Line2D at 0x7f0a3f10b250>]
```

[395]:
```
## 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
```

[396]:
```
fitClusters_cao
```

Out[396]:
```
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)
```

[397]:
```
#Combining the predicted clusters with the original DF
categorical = categorical_copy.reset_index()
```

[398]:
```
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)
```

[399]:
```
combinedDf.head()
```

Out[399]:

	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

[400]:
```
cluster_0 = combinedDf[combinedDf['cluster_predicted'] == 0]
cluster_1 = combinedDf[combinedDf['cluster_predicted'] == 1]
```

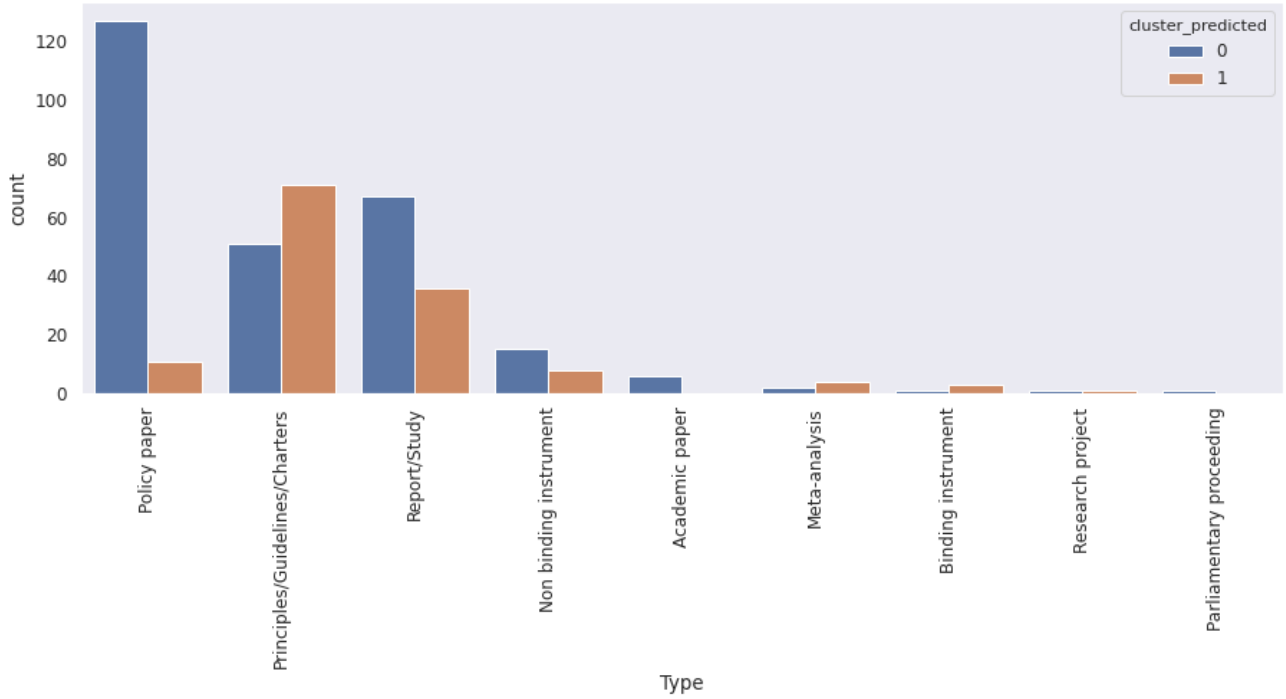
[401]:
```
cluster_0.info()
```

```
<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
```

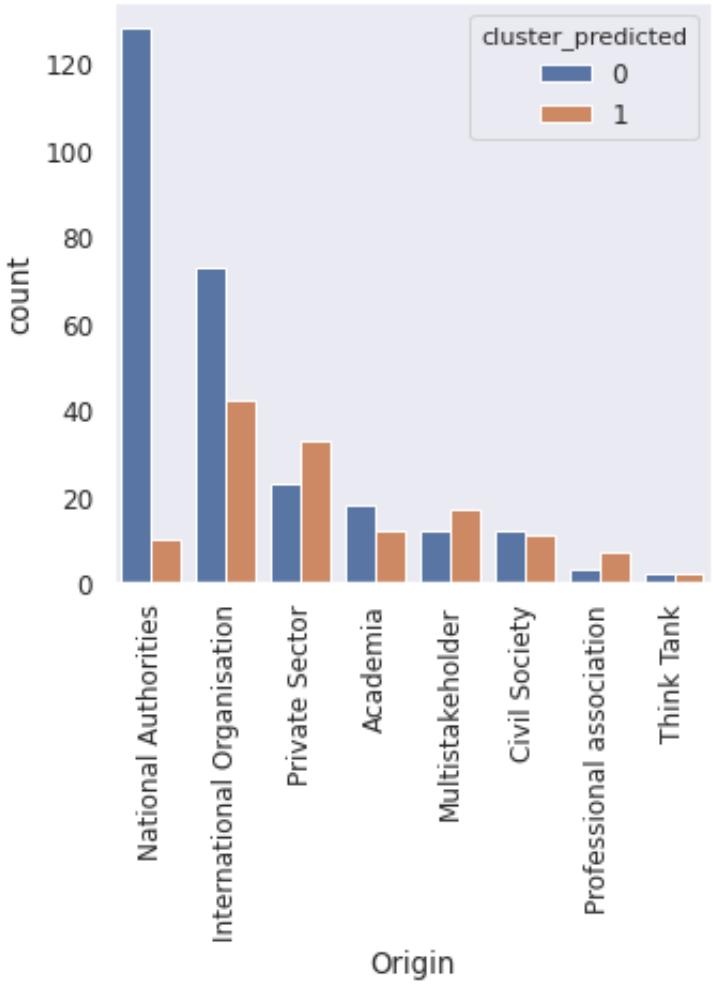
[402]:
```
cluster_1.info()
```

```
<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
```

[403]:
```
ubplots(figsize = (15,5))
sns.countplot(x=combinedDf['Type'],order=combinedDf['Type'].value_counts()
t_xticklabels(ax.get_xticklabels(), rotation=90)
how()
```

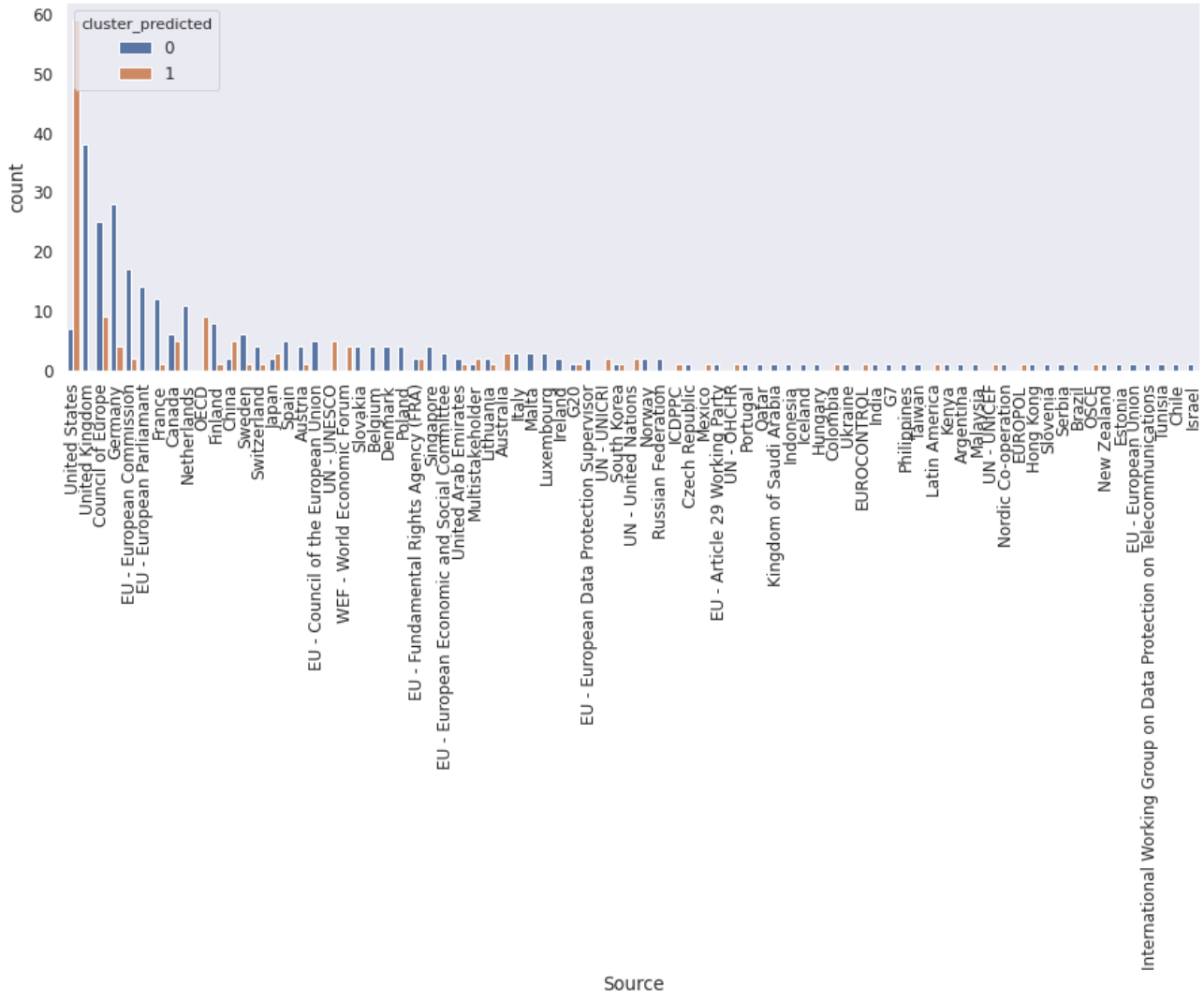


[404]:
```
plt.subplots(figsize = (5,5))
ax = sns.countplot(x=combinedDf['Origin'],order=combinedDf['Origin'].valu
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()
```

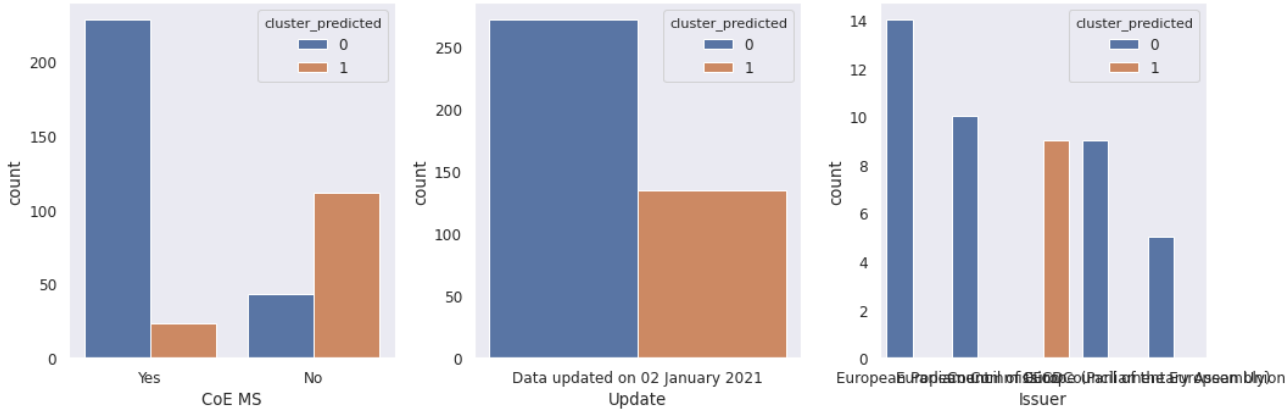


[405]:

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



[406]:
```
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()
```



[411]:
```
f, axs = plt.subplots(1,2,figsize = (15,5))
sns.countplot(x=combinedDf['Year'],order=combinedDf['Year'].value_counts(
plt.tight_layout()
plt.show()
```

[1]:
```
!ls
```

```
__notebook_source__.ipynb
```

[6]:
```
#pip install notebook-as-pdf
```

[]:

[]:

[]: