analyze_data_scopes

August 8, 2025

[ ]:
```
import sys
sys.path.append("..")
from base.dataset_loader import CategoricalLoader, FinancialLoader, ScopeLoader
from datasources.loaders import RegionLoader

from datasources.local import LocalDatasource

import pathlib

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from base import OxariDataManager
from datasources.core import DefaultDataManager,␣
 ↪PreviousScopeFeaturesDataManager
from datasources.online import S3Datasource

sns.set_palette('viridis')
```

[ ]:
```
cwd = pathlib.Path(__file__).parent
DATA = pd.read_csv(cwd.parent/'model-data/input/scopes.csv')
DATA
```

[ ]:
```
        key_year key_ticker        meta_name  … tg_numc_scope_1
tg_numc_scope_2 tg_numc_scope_3
0         2018.0   1U1.XFRA           1&1 AG  …           943.0
1412.0             NaN
1         2018.0   DRI.XWBO           1&1 AG  …           943.0
1412.0             NaN
2         2018.0   1U1.XDUS           1&1 AG  …           943.0
1412.0             NaN
3         2018.0   1U1.XMUN           1&1 AG  …           943.0
1412.0             NaN
4         2018.0   1U1.XSTU           1&1 AG  …           943.0
1412.0             NaN
```

```
…          …         …                …  …              …
…             …
168994    2020.0   TKA.XBER  thyssenkrupp AG  …      21800000.0
1300000.0            NaN
168995    2020.0   TKA.XDUS  thyssenkrupp AG  …      21800000.0
1300000.0            NaN
168996    2020.0   TKA.XHAN  thyssenkrupp AG  …      21800000.0
1300000.0            NaN
168997    2020.0   TKA.XMUN  thyssenkrupp AG  …      21800000.0
1300000.0            NaN
168998    2020.0   TKA.XSTU  thyssenkrupp AG  …      21800000.0
1300000.0            NaN

[168999 rows x 14 columns]
```

[ ]:
```
df_scopes = DATA
df_scopes["grp_scope_1"] = None
df_scopes["log_scope_1"] = None
df_scopes.loc[df_scopes["tg_numc_scope_1"].isna(), ["grp_scope_1"]] = "Not␣
 ↪reported"
df_scopes.loc[df_scopes["tg_numc_scope_1"] == 0, ["grp_scope_1"]] = "Zero␣
 ↪Emissions"
df_scopes.loc[df_scopes["tg_numc_scope_1"] < 0, ["grp_scope_1"]] = "Impossible"
df_scopes.loc[df_scopes["tg_numc_scope_1"].between(0, 1, inclusive='right'),␣
 ↪["grp_scope_1"]] = "Weird"
df_scopes.loc[df_scopes["tg_numc_scope_1"] > 1, ["grp_scope_1"]] = "Emittor"
df_scopes["log_scope_1"] = np.log(df_scopes["tg_numc_scope_1"])
indices = df_scopes["tg_numc_scope_1"] > 0
df_scopes
```

```
c:\Users\User\Workspace\work_oxari\architectura\.venv\Lib\site-
packages\pandas\core\arraylike.py:402: RuntimeWarning: divide by zero
encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

[ ]:
```
        key_year key_ticker        meta_name  … tg_numc_scope_3 grp_scope_1
log_scope_1
0         2018.0   1U1.XFRA           1&1 AG  …              NaN     Emittor
6.849066
1         2018.0   DRI.XWBO           1&1 AG  …              NaN     Emittor
6.849066
2         2018.0   1U1.XDUS           1&1 AG  …              NaN     Emittor
6.849066
3         2018.0   1U1.XMUN           1&1 AG  …              NaN     Emittor
6.849066
4         2018.0   1U1.XSTU           1&1 AG  …              NaN     Emittor
6.849066
```

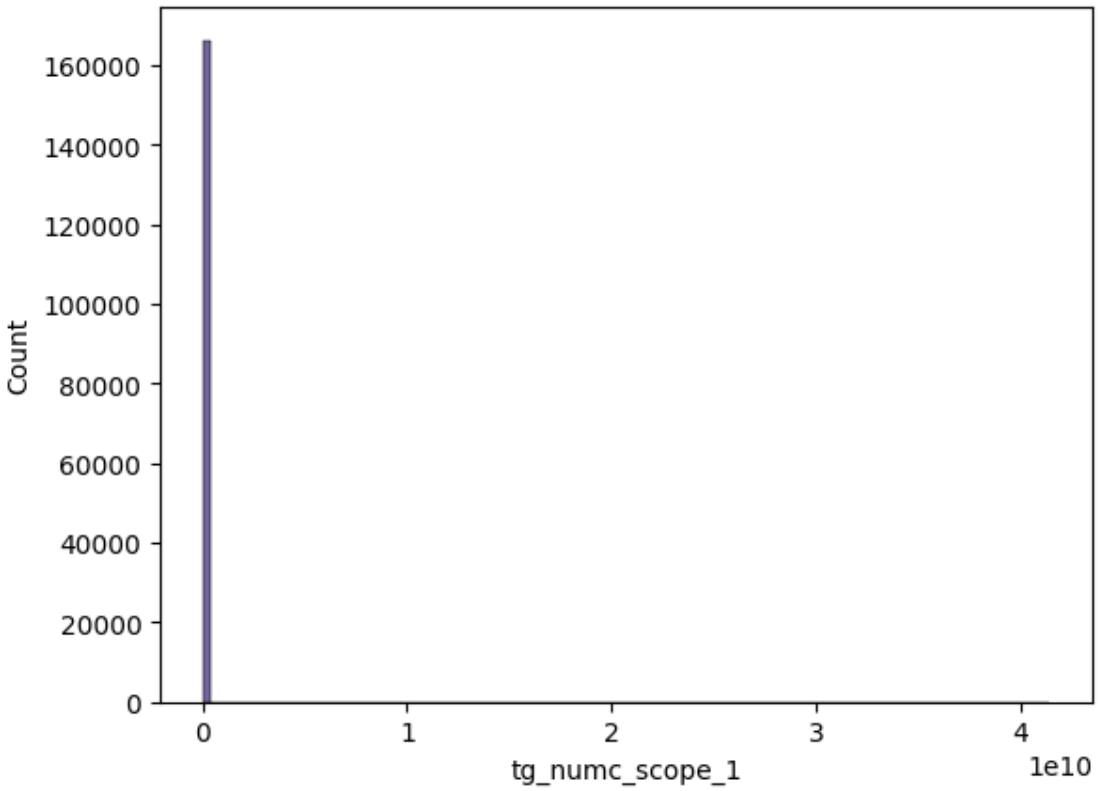
```
…          …         …                …  …              …          …
…
168994    2020.0   TKA.XBER  thyssenkrupp AG  …            NaN     Emittor
16.897421
168995    2020.0   TKA.XDUS  thyssenkrupp AG  …            NaN     Emittor
16.897421
168996    2020.0   TKA.XHAN  thyssenkrupp AG  …            NaN     Emittor
16.897421
168997    2020.0   TKA.XMUN  thyssenkrupp AG  …            NaN     Emittor
16.897421
168998    2020.0   TKA.XSTU  thyssenkrupp AG  …            NaN     Emittor
16.897421

[168999 rows x 16 columns]
```

[ ]:
```
df_scopes['grp_scope_1'].value_counts()
```

[ ]:
```
Emittor           166178
Not reported        1763
Zero Emissions       987
Weird                 71
Name: grp_scope_1, dtype: int64
```

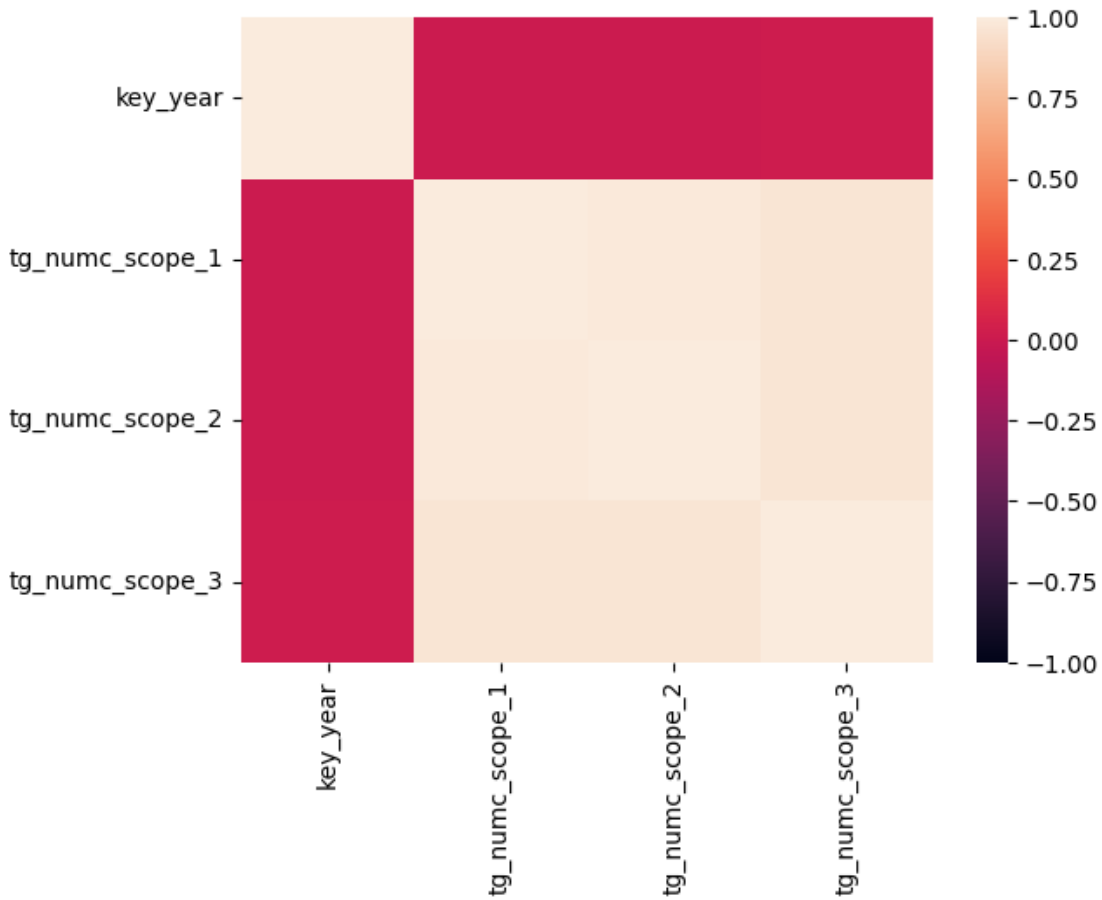
[ ]:
```
sns.histplot(data=df_scopes[df_scopes["tg_numc_scope_1"] > 0],␣
 ↪x="tg_numc_scope_1", bins=100)
plt.show()
```



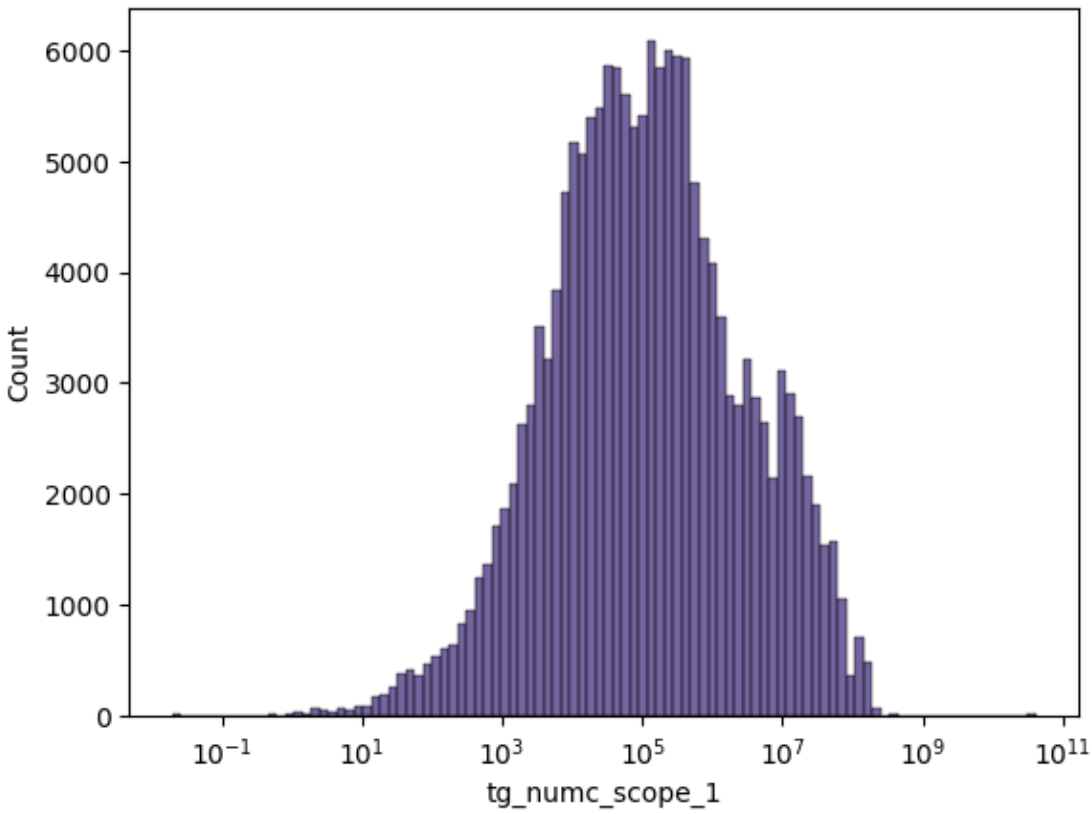
[ ]:
```
corrs =␣
 ↪df_scopes[["key_year","tg_numc_scope_1","tg_numc_scope_2","tg_numc_scope_3"]].
 ↪corr()
print(corrs)
sns.heatmap(corrs, vmin=-1, vmax=1)
plt.show()

# sns.histplot(data=df_scopes[(df_scopes["tg_numc_scope_1"] > 0) &␣
 ↪(df_scopes["tg_numc_scope_1"] < 1e4)], x="tg_numc_scope_1", bins=100)
```

	key_year	tg_numc_scope_1	tg_numc_scope_2	tg_numc_scope_3
key_year	1.000000	0.003497	0.006625	0.009564
tg_numc_scope_1	0.003497	1.000000	0.991350	0.963769
tg_numc_scope_2	0.006625	0.991350	1.000000	0.967808
tg_numc_scope_3	0.009564	0.963769	0.967808	1.000000



[ ]:
```
sns.histplot(data=df_scopes[df_scopes["tg_numc_scope_1"] > 0],␣
  ↪x="tg_numc_scope_1", bins=100, log_scale=True)
plt.show()
```



[ ]:
```
df_scopes[df_scopes["grp_scope_1"] != "Zero Emissions"].groupby('key_year').
↪var()
```

```
<ipython-input-20-1e4630c51b55>:2: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.var is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  df_scopes[df_scopes["grp_scope_1"] != "Zero
Emissions"].groupby('key_year').var()
```

[ ]:

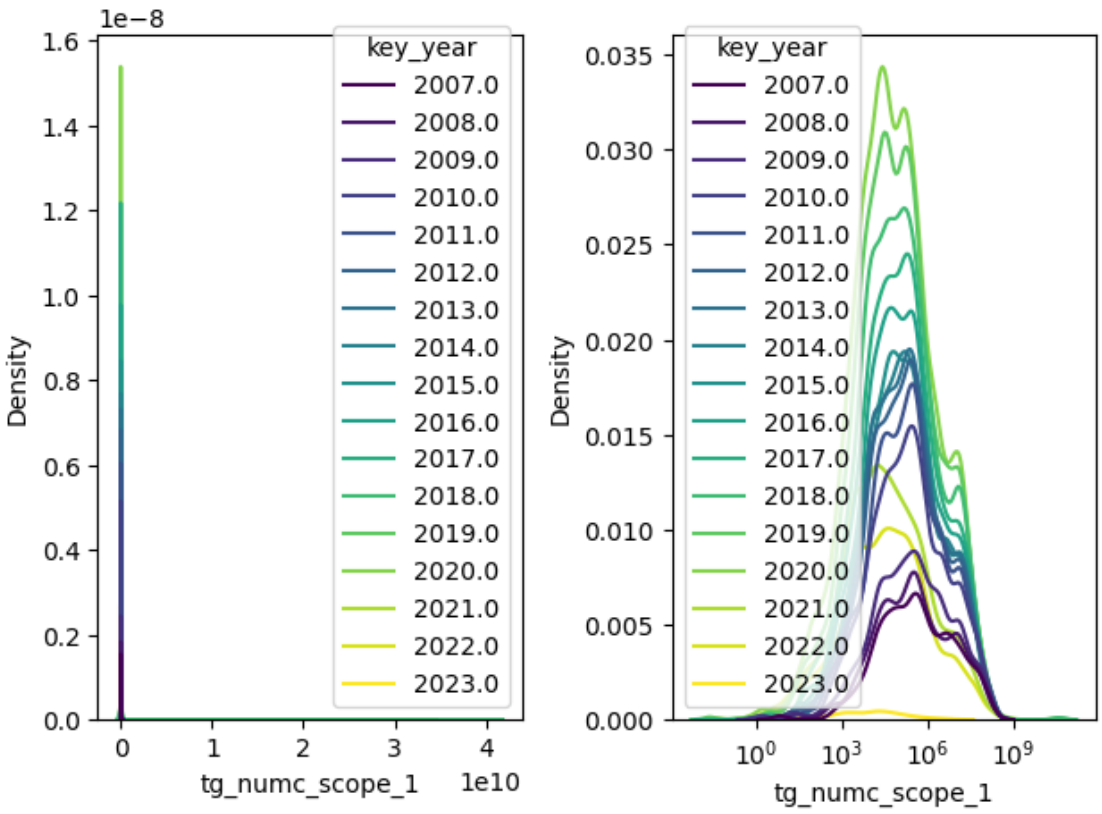
key_year	tg_numc_scope_1	tg_numc_scope_2	tg_numc_scope_3	log_scope_1
2007.0	5.144858e+14	1.314011e+13	1.198471e+16	8.859099
2008.0	4.713573e+14	1.338966e+13	5.127587e+15	8.774211
2009.0	4.638895e+14	1.094622e+13	6.749594e+15	9.602394
2010.0	3.345452e+14	6.402582e+12	6.337912e+15	9.411160
2011.0	3.296603e+14	8.403365e+12	6.205273e+15	9.104389
2012.0	3.150330e+14	4.238025e+12	5.562443e+15	9.159565
2013.0	3.264929e+14	4.318898e+12	6.189106e+15	9.222339
2014.0	2.857181e+14	4.219932e+12	7.126690e+15	9.146470
2015.0	2.645297e+14	4.420276e+12	7.562764e+16	9.292187

2016.0	2.638279e+14	1.528415e+18	6.059562e+15	9.271228
2017.0	2.223839e+14	5.417163e+12	1.097253e+16	9.547705
2018.0	8.730553e+17	1.111968e+20	3.698570e+18	10.058544
2019.0	5.219852e+17	1.004316e+20	3.413148e+18	10.587793
2020.0	3.046553e+14	6.328033e+12	2.936671e+16	10.897105
2021.0	1.625379e+14	8.566181e+12	3.026430e+17	12.014829
2022.0	3.237454e+14	9.313944e+12	4.874352e+15	12.180389
2023.0	2.407839e+11	2.670472e+11	3.804687e+13	7.021464

[ ]:
```
df_scopes[indices].groupby("key_year")
```

[ ]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000015C95521310>
```

[ ]:
```
# sns.set_palette('viridis')
fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
sns.kdeplot(data=df_scopes[df_scopes["tg_numc_scope_1"] > 0],␣
 ↪x='tg_numc_scope_1', hue="key_year", log_scale=False, ax=ax,␣
 ↪palette='viridis')
ax = fig.add_subplot(1, 2, 2)
sns.kdeplot(data=df_scopes[df_scopes["tg_numc_scope_1"] > 0],␣
 ↪x='tg_numc_scope_1', hue="key_year", log_scale=True, ax=ax,␣
 ↪palette='viridis')
fig.tight_layout()
plt.show()
```



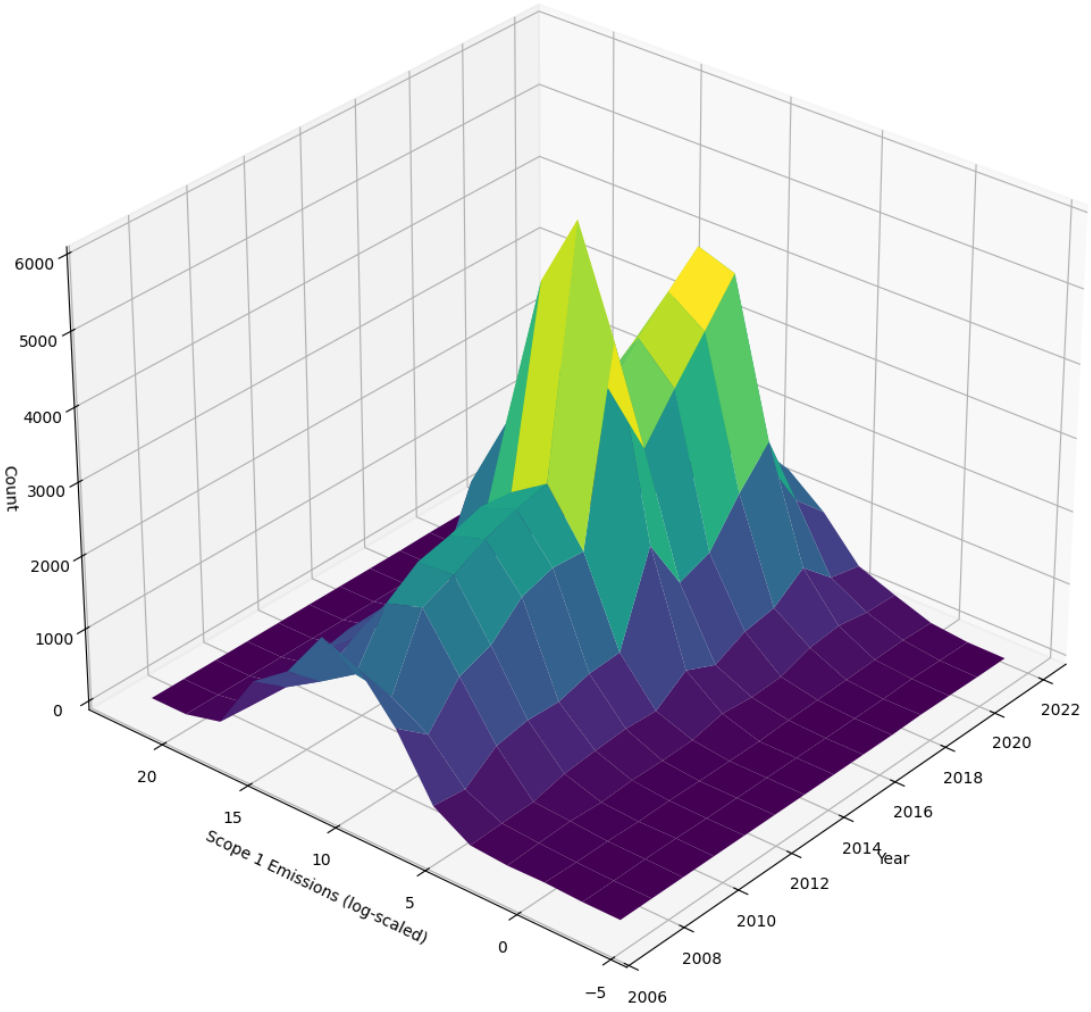
[ ]:
```
# sns.kdeplot(data=df_scopes.groupby(['isin']).mean(), x="scope_1")
num_bins = 14
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(projection='3d')

hist, xedges, yedges = np.histogram2d(df_scopes[indices]["key_year"],␣
 ↪df_scopes[indices]["log_scope_1"], bins=num_bins)

# Construct arrays for the anchor positions of the 16 bars.
xpos, ypos = np.meshgrid(xedges[:-1], yedges[:-1], indexing="ij")

ax.plot_surface(xpos, ypos, hist, cmap='viridis', edgecolor='none')
ax.view_init(30, 220)
ax.set_xlabel('Year')
ax.set_ylabel('Scope 1 Emissions (log-scaled)')
ax.set_zlabel('Count')

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



[ ]:
```
num_bins = 14
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(projection='3d')

hist, xedges, yedges = np.histogram2d(df_scopes[indices]["key_year"],␣
 ↪df_scopes[indices]["log_scope_1"], bins=num_bins)

for row, year in zip(hist, xedges):
    xs = [year] * num_bins
    ys = row
    zs = yedges[:-1]
    ax.plot(xs, ys, zs, zdir="y")
    # ax.bar(xs, ys, zs, zdir="y")
```

```
ax.set_xlabel('Year')
ax.set_ylabel('Scope 1 Emissions (log-scaled)')
ax.set_zlabel('Count')

# On the y axis let's only label the discrete values that we have data for.
ax.set_yticks(np.round(yedges))
ax.invert_xaxis()
# ax.set_xticks(np.round(xedges))
ax.view_init(20, 25)
fig.tight_layout()
plt.show()
```