In [9]:
```
import pandas as pd
```

In [10]:
```
import numpy as np
```

In [12]:
```
# Reading the CSV files, skipping metadata rows
df_0_14 = pd.read_csv("C:/Users/aryas/OneDrive/Documents/World Bank Data/0-14 age grou
df_15_64 = pd.read_csv("C:/Users/aryas/OneDrive/Documents/World Bank Data/15-64 age.cs
df_65_above = pd.read_csv("C:\\Users\\aryas\\OneDrive\\Documents\\World Bank Data\\65
```

In [4]:
```
print(df_0_14)
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
0	Aruba	ABW	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	43.131043	42.949419	42.852732	42.661157	42.359159	41.936664	...	19.348756	19.045505	18.799607	18.571721	18.334859	18.069771	17.767339	17.351022	16.799407	16.240782
1	Africa Eastern and Southern	AFE	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	44.200016	44.285569	44.381992	44.495098	44.609810	44.728859	...	42.933684	42.777333	42.594160	42.423994	42.255251	42.052776	41.832090	41.598105	41.362729	41.122828
2	Afghanistan	AFG	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	41.627186	41.695303	41.769167	41.885377	42.059389	42.314133	...	46.867621	46.231538	45.792106	45.520967	45.118616	44.708445	44.291352	43.807912	43.424543	43.130634
3	Africa Western and Central	AFW	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	41.322373	41.417194	41.529034	41.696118	41.944426	42.168576	...	44.266677	44.231527	44.155163	44.042022	43.893535	43.721214	43.531411	43.319347	43.095027	42.852345
4	Angola	AGO	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	42.236698	42.728464	43.256801	43.811211	44.416202	44.971958	...	45.756383	45.764164	45.759548	45.718586	45.641093	45.545945	45.425588	45.306602	45.179105	45.015131
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	43.933394	44.370671	44.687577	44.819432	44.700448	44.401636	...	26.690248	26.485297	25.964487	25.162776	24.434338	23.816523	23.247349	22.638645	22.001860	21.329705
262	Yemen, Rep.	YEM	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	40.736166	40.747765	40.773133	40.875174	41.072802	41.368906	...	42.375582	42.027499	41.702829	41.407766	41.118413	40.820397				
263	South Africa	ZAF	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	42.452166	42.301710	42.224522	42.123209	42.075672	42.097008	...	28.423750	28.394889	28.275352	28.365854	28.586096	28.681530				
264	Zambia	ZMB	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	46.099247	46.303423	46.517781	46.773843	47.037066	47.253934	...	46.039495	45.789022	45.500445	45.189872	44.857299	44.496183				
265	Zimbabwe	ZWE	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	47.315655	47.663873	48.052887	48.509669	48.733595	48.674055	...	43.370440	43.172363	42.872670	42.517883	42.152298	41.798795				

```
262  40.503437  40.188908  39.872107  39.497243
263  28.745269  28.739672  28.672962  28.550796
264  44.099904  43.680373  43.262363  42.856553
265  41.468973  41.156085  40.893305  40.634003

[266 rows x 67 columns]
```

In [5]:
```
print(df_15_64)
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
0	Aruba	ABW	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	54.495678	54.588701	54.585630	54.674206	54.873448	55.181477	...	69.795393	69.620827	69.352325	69.027743	68.681269	68.320569	67.962515	67.712003	67.657094	67.617079
1	Africa Eastern and Southern	AFE	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	52.827416	52.760490	52.682676	52.586510	52.485520	52.376628	...	54.104827	54.234396	54.390358	54.530686	54.666078	54.834273	55.021545	55.235200	55.471932	55.721996
2	Afghanistan	AFG	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	55.539784	55.487023	55.431779	55.335655	55.181682	54.946620	...	50.729095	51.359300	51.802078	52.080044	52.481567	52.884462	53.290894	53.775135	54.171399	54.475027
3	Africa Western and Central	AFW	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	55.375946	55.275243	55.149793	54.971579	54.716344	54.490265	...	52.790894	52.851132	52.945271	53.064443	53.215431	53.386094	53.570061	53.783697	54.021827	54.271367
4	Angola	AGO	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	54.683257	54.177239	53.645570	53.091408	52.490711	51.943488	...	51.670046	51.679061	51.693635	51.735579	51.809850	51.894521	51.998415	52.103423	52.226425	52.384814
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	51.591830	51.221073	50.951144	50.844076	50.967615	51.258194	...	65.777830	65.688609	65.918197	66.454202	66.878794	67.166434	67.362981	67.672114	68.087006	68.477873
262	Yemen, Rep.	YEM	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	55.477807	55.501457	55.511203	55.448269	55.290068	55.031122	...	54.828845	55.176310	55.510796	55.821782	56.124579	56.433893				
263	South Africa	ZAF	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	54.167016	54.334620	54.424701	54.529505	54.571067	54.534850	...	66.343729	66.257397	66.292438	66.063901	65.676570	65.464346				
264	Zambia	ZMB	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	51.205645	51.003323	50.783529	50.517437	50.244703	50.019266	...	52.269049	52.533154	52.826394	53.135408	53.461350	53.810028				
265	Zimbabwe	ZWE	Population ages 15-64 (% of total population)	SP.POP.1564.TO.ZS	49.348844	49.006719	48.628252	48.185370	47.977551	48.055265	...	53.634994	53.799451	54.029704	54.314703	54.614584	54.907846				

```
262  56.759831  57.092707  57.438854  57.840528
263  65.304692  65.259793  65.354907  65.556214
264  54.187869  54.589767  54.998707  55.395954
265  55.185247  55.467653  55.743353  56.044152

[266 rows x 67 columns]
```

In [6]:
```
print(df_65_above)
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Aruba	ABW	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	2.373279	2.461880	2.561637	2.664637	2.767393	2.881859	...
1	Africa Eastern and Southern	AFE	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	2.972568	2.953941	2.935332	2.918392	2.904670	2.894513	...
2	Afghanistan	AFG	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	2.833029	2.817674	2.799055	2.778968	2.758929	2.739247	...
3	Africa Western and Central	AFW	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.301681	3.307563	3.321173	3.332303	3.339230	3.341159	...
4	Angola	AGO	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.080044	3.094297	3.097629	3.097381	3.093087	3.084555	...
..	...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	4.474777	4.408256	4.361278	4.336492	4.331937	4.340169	...
262	Yemen, Rep.	YEM	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.786027	3.750778	3.715664	3.676558	3.637130	3.599972	...
263	South Africa	ZAF	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.380818	3.363670	3.350777	3.347286	3.353262	3.368142	...
264	Zambia	ZMB	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	2.695108	2.693254	2.698690	2.708720	2.718231	2.726800	...
265	Zimbabwe	ZWE	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.335501	3.329409	3.318861	3.304961	3.288854	3.270680	...

	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
0	10.855851	11.333668	11.848068	12.400536	12.983872	13.609660	14.270146	14.936975	15.543499	16.142139
1	2.961488	2.988271	3.015482	3.045321	3.078671	3.112951	3.146365	3.166695	3.165339	3.155176
2	2.403285	2.409162	2.405816	2.398990	2.399818	2.407093	2.417754	2.416953	2.404058	2.394340
3	2.942429	2.917341	2.899566	2.893535	2.891034	2.892693	2.898529	2.896957	2.883146	2.876289
4	2.573570	2.556775	2.546817	2.545834	2.549057	2.559535	2.575997	2.589975	2.594470	2.600056
..	...	...	...	...	...	...	...	...	...	...
261	7.531922	7.826094	8.117316	8.383022	8.686868	9.017042	9.389670	9.689241	9.911135	10.192422

```
262   2.736732   2.718385   2.689038   2.662229
263   5.950038   6.000535   5.972131   5.892991
264   1.712227   1.729860   1.738930   1.747493
265   3.345781   3.376262   3.363343   3.321845

[266 rows x 67 columns]
```

In [13]:
```
df_combined = pd.concat([df_0_14, df_15_64, df_65_above])
```

In [14]:
```
print(df_combined )
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
0	Aruba	ABW	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	43.131043	42.949419	42.852732	42.661157	42.359159	41.936664	...	19.348756	19.045505	18.799607	18.571721	18.334859	18.069771	17.767339	17.351022	16.799407	16.240782
1	Africa Eastern and Southern	AFE	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	44.200016	44.285569	44.381992	44.495098	44.609810	44.728859	...	42.933684	42.777333	42.594160	42.423994	42.255251	42.052776	41.832090	41.598105	41.362729	41.122828
2	Afghanistan	AFG	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	41.627186	41.695303	41.769167	41.885377	42.059389	42.314133	...	46.867621	46.231538	45.792106	45.520967	45.118616	44.708445	44.291352	43.807912	43.424543	43.130634
3	Africa Western and Central	AFW	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	41.322373	41.417194	41.529034	41.696118	41.944426	42.168576	...	44.266677	44.231527	44.155163	44.042022	43.893535	43.721214	43.531411	43.319347	43.095027	42.852345
4	Angola	AGO	Population ages 0-14 (% of total population)	SP.POP.0014.TO.ZS	42.236698	42.728464	43.256801	43.811211	44.416202	44.971958	...	45.756383	45.764164	45.759548	45.718586	45.641093	45.545945	45.425588	45.306602	45.179105	45.015131
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
261	Kosovo	XKX	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	4.474777	4.408256	4.361278	4.336492	4.331937	4.340169	...	7.531922	7.826094	8.117316	8.383022	8.686868	9.017042	9.389670	9.689241	9.911135	10.192422
262	Yemen, Rep.	YEM	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.786027	3.750778	3.715664	3.676558	3.637130	3.599972	...	2.795573	2.796191	2.786375	2.770451	2.757008	2.745709				
263	South Africa	ZAF	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.380818	3.363670	3.350777	3.347286	3.353262	3.368142	...	5.232520	5.347714	5.432211	5.570245	5.737334	5.854124				
264	Zambia	ZMB	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	2.695108	2.693254	2.698690	2.708720	2.718231	2.726800	...	1.691457	1.677824	1.673161	1.674720	1.681351	1.693790				
265	Zimbabwe	ZWE	Population ages 65 and above (% of total popul...	SP.POP.65UP.TO.ZS	3.335501	3.329409	3.318861	3.304961	3.288854	3.270680	...	2.994566	3.028186	3.097626	3.167414	3.233118	3.293359				

```
262   2.736732   2.718385   2.689038   2.662229
263   5.950038   6.000535   5.972131   5.892991
264   1.712227   1.729860   1.738930   1.747493
265   3.345781   3.376262   3.363343   3.321845

[798 rows x 67 columns]
```

In [8]:
```
import matplotlib.pyplot as plt
```

In [10]:
```
print(df_combined.columns)
```

```
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022'],
      dtype='object')
```

In [11]:
```
print(df_combined.columns)
```

```
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022'],
      dtype='object')
```

In [12]:
```
df_combined.shape
```

Out[12]:
```
(798, 67)
```

In [13]:
```
df_combined.columns
```

Out[13]:
```
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022'],
      dtype='object')
```

In [14]:
```
last_year = df_combined.columns[df_combined.columns.str.isnumeric()][-1]
```

In [15]:
```
selected_columns = df_combined.columns[4:df_combined.columns.get_loc(last_year) + 1]
```

In [16]:
```
df_combined.describe()
```

Out[16]:

	1960	1961	1962	1963	1964	1965	1966	1967
count	795.000000	795.000000	795.000000	795.000000	795.000000	795.000000	795.000000	795.000000
mean	33.333333	33.333333	33.333333	33.333333	33.333333	33.333333	33.333333	33.333333
std	21.934068	21.898832	21.876102	21.856506	21.836975	21.817738	21.800284	21.788258
min	1.094746	1.087430	1.069956	1.045335	1.015901	0.984709	0.958864	0.940156
25%	5.883750	5.838678	5.845071	5.800939	5.721881	5.873718	6.013141	6.173385
50%	40.890044	41.417194	41.362898	41.508021	41.743148	41.984757	42.331521	42.493260
75%	52.525813	52.313362	52.028684	51.935161	51.735457	51.750847	51.614954	51.534132
max	68.088093	68.251146	68.475944	68.711671	68.971677	69.213469	69.246471	69.229039

8 rows × 63 columns

In [17]:
```
selected_columns_data = df_combined[selected_columns]
selected_columns_data.describe()
```

Out[17]:

	1960	1961	1962	1963	1964	1965	1966	1967
count	795.000000	795.000000	795.000000	795.000000	795.000000	795.000000	795.000000	795.000000
mean	33.333333	33.333333	33.333333	33.333333	33.333333	33.333333	33.333333	33.333333
std	21.934068	21.898832	21.876102	21.856506	21.836975	21.817738	21.800284	21.788258
min	1.094746	1.087430	1.069956	1.045335	1.015901	0.984709	0.958864	0.940156
25%	5.883750	5.838678	5.845071	5.800939	5.721881	5.873718	6.013141	6.173385
50%	40.890044	41.417194	41.362898	41.508021	41.743148	41.984757	42.331521	42.493260
75%	52.525813	52.313362	52.028684	51.935161	51.735457	51.750847	51.614954	51.534132
max	68.088093	68.251146	68.475944	68.711671	68.971677	69.213469	69.246471	69.229039

8 rows × 63 columns

In [18]:
```
df_combined['Country Name'].unique()
```

Out[18]:
```
array(['Aruba', 'Africa Eastern and Southern', 'Afghanistan',
       'Africa Western and Central', 'Angola', 'Albania', 'Andorra',
       'Arab World', 'United Arab Emirates', 'Argentina', 'Armenia',
       'American Samoa', 'Antigua and Barbuda', 'Australia', 'Austria',
       'Azerbaijan', 'Burundi', 'Belgium', 'Benin', 'Burkina Faso',
       'Bangladesh', 'Bulgaria', 'Bahrain', 'Bahamas, The',
       'Bosnia and Herzegovina', 'Belarus', 'Belize', 'Bermuda',
       'Bolivia', 'Brazil', 'Barbados', 'Brunei Darussalam', 'Bhutan',
       'Botswana', 'Central African Republic', 'Canada',
       'Central Europe and the Baltics', 'Switzerland', 'Channel Islands',
       'Chile', 'China', "Cote d'Ivoire", 'Cameroon', 'Congo, Dem. Rep.',
       'Congo, Rep.', 'Colombia', 'Comoros', 'Cabo Verde', 'Costa Rica',
       'Caribbean small states', 'Cuba', 'Curacao', 'Cayman Islands',
       'Cyprus', 'Czechia', 'Germany', 'Djibouti', 'Dominica', 'Denmark',
       'Dominican Republic', 'Algeria',
       'East Asia & Pacific (excluding high income)',
       'Early-demographic dividend', 'East Asia & Pacific',
       'Europe & Central Asia (excluding high income)',
       'Europe & Central Asia', 'Ecuador', 'Egypt, Arab Rep.',
       'Euro area', 'Eritrea', 'Spain', 'Estonia', 'Ethiopia',
       'European Union', 'Fragile and conflict affected situations',
       'Finland', 'Fiji', 'France', 'Faroe Islands',
       'Micronesia, Fed. Sts.', 'Gabon', 'United Kingdom', 'Georgia',
       'Ghana', 'Gibraltar', 'Guinea', 'Gambia, The', 'Guinea-Bissau',
       'Equatorial Guinea', 'Greece', 'Grenada', 'Greenland', 'Guatemala',
       'Guam', 'Guyana', 'High income', 'Hong Kong SAR, China',
       'Honduras', 'Heavily indebted poor countries (HIPC)', 'Croatia',
       'Haiti', 'Hungary', 'IBRD only', 'IDA & IBRD total', 'IDA total',
       'IDA blend', 'Indonesia', 'IDA only', 'Isle of Man', 'India',
       'Not classified', 'Ireland', 'Iran, Islamic Rep.', 'Iraq',
       'Iceland', 'Israel', 'Italy', 'Jamaica', 'Jordan', 'Japan',
       'Kazakhstan', 'Kenya', 'Kyrgyz Republic', 'Cambodia', 'Kiribati',
       'St. Kitts and Nevis', 'Korea, Rep.', 'Kuwait',
       'Latin America & Caribbean (excluding high income)', 'Lao PDR',
       'Lebanon', 'Liberia', 'Libya', 'St. Lucia',
       'Latin America & Caribbean',
       'Least developed countries: UN classification', 'Low income',
       'Liechtenstein', 'Sri Lanka', 'Lower middle income',
       'Low & middle income', 'Lesotho', 'Late-demographic dividend',
       'Lithuania', 'Luxembourg', 'Latvia', 'Macao SAR, China',
       'St. Martin (French part)', 'Morocco', 'Monaco', 'Moldova',
       'Madagascar', 'Maldives', 'Middle East & North Africa', 'Mexico',
       'Marshall Islands', 'Middle income', 'North Macedonia', 'Mali',
       'Malta', 'Myanmar',
       'Middle East & North Africa (excluding high income)', 'Montenegro',
       'Mongolia', 'Northern Mariana Islands', 'Mozambique', 'Mauritania',
       'Mauritius', 'Malawi', 'Malaysia', 'North America', 'Namibia',
       'New Caledonia', 'Niger', 'Nigeria', 'Nicaragua', 'Netherlands',
       'Norway', 'Nepal', 'Nauru', 'New Zealand', 'OECD members', 'Oman',
       'Other small states', 'Pakistan', 'Panama', 'Peru', 'Philippines',
       'Palau', 'Papua New Guinea', 'Poland', 'Pre-demographic dividend',
       'Puerto Rico', "Korea, Dem. People's Rep.", 'Portugal', 'Paraguay',
       'West Bank and Gaza', 'Pacific island small states',
       'Post-demographic dividend', 'French Polynesia', 'Qatar',
       'Romania', 'Russian Federation', 'Rwanda', 'South Asia',
       'Saudi Arabia', 'Sudan', 'Senegal', 'Singapore', 'Solomon Islands',
       'Sierra Leone', 'El Salvador', 'San Marino', 'Somalia', 'Serbia',
       'Sub-Saharan Africa (excluding high income)', 'South Sudan',
       'Sub-Saharan Africa', 'Small states', 'Sao Tome and Principe',
       'Suriname', 'Slovak Republic', 'Slovenia', 'Sweden', 'Eswatini',
```

```
       'Sint Maarten (Dutch part)', 'Seychelles', 'Syrian Arab Republic',
       'Turks and Caicos Islands', 'Chad',
       'East Asia & Pacific (IDA & IBRD countries)',
       'Europe & Central Asia (IDA & IBRD countries)', 'Togo', 'Thailand',
       'Tajikistan', 'Turkmenistan',
       'Latin America & the Caribbean (IDA & IBRD countries)',
       'Timor-Leste', 'Middle East & North Africa (IDA & IBRD countries)',
       'Tonga', 'South Asia (IDA & IBRD)',
       'Sub-Saharan Africa (IDA & IBRD countries)', 'Trinidad and Tobago',
       'Tunisia', 'Turkiye', 'Tuvalu', 'Tanzania', 'Uganda', 'Ukraine',
       'Upper middle income', 'Uruguay', 'United States', 'Uzbekistan',
       'St. Vincent and the Grenadines', 'Venezuela, RB',
       'British Virgin Islands', 'Virgin Islands (U.S.)', 'Viet Nam',
       'Vanuatu', 'World', 'Samoa', 'Kosovo', 'Yemen, Rep.',
       'South Africa', 'Zambia', 'Zimbabwe'], dtype=object)
```

In [19]:
```
df_combined['Country Code'].unique()
```

Out[19]:
```
array(['ABW', 'AFE', 'AFG', 'AFW', 'AGO', 'ALB', 'AND', 'ARB', 'ARE',
       'ARG', 'ARM', 'ASM', 'ATG', 'AUS', 'AUT', 'AZE', 'BDI', 'BEL',
       'BEN', 'BFA', 'BGD', 'BGR', 'BHR', 'BHS', 'BIH', 'BLR', 'BLZ',
       'BMU', 'BOL', 'BRA', 'BRB', 'BRN', 'BTN', 'BWA', 'CAF', 'CAN',
       'CEB', 'CHE', 'CHI', 'CHL', 'CHN', 'CIV', 'CMR', 'COD', 'COG',
       'COL', 'COM', 'CPV', 'CRI', 'CSS', 'CUB', 'CUW', 'CYM', 'CYP',
       'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'EAP', 'EAR',
       'EAS', 'ECA', 'ECS', 'ECU', 'EGY', 'EMU', 'ERI', 'ESP', 'EST',
       'ETH', 'EUU', 'FCS', 'FIN', 'FJI', 'FRA', 'FRO', 'FSM', 'GAB',
       'GBR', 'GEO', 'GHA', 'GIB', 'GIN', 'GMB', 'GNB', 'GNQ', 'GRC',
       'GRD', 'GRL', 'GTM', 'GUM', 'GUY', 'HIC', 'HKG', 'HND', 'HPC',
       'HRV', 'HTI', 'HUN', 'IBD', 'IBT', 'IDA', 'IDB', 'IDN', 'IDX',
       'IMN', 'IND', 'INX', 'IRL', 'IRN', 'IRQ', 'ISL', 'ISR', 'ITA',
       'JAM', 'JOR', 'JPN', 'KAZ', 'KEN', 'KGZ', 'KHM', 'KIR', 'KNA',
       'KOR', 'KWT', 'LAC', 'LAO', 'LBN', 'LBR', 'LBY', 'LCA', 'LCN',
       'LDC', 'LIC', 'LIE', 'LKA', 'LMC', 'LMY', 'LSO', 'LTE', 'LTU',
       'LUX', 'LVA', 'MAC', 'MAF', 'MAR', 'MCO', 'MDA', 'MDG', 'MDV',
       'MEA', 'MEX', 'MHL', 'MIC', 'MKD', 'MLI', 'MLT', 'MMR', 'MNA',
       'MNE', 'MNG', 'MNP', 'MOZ', 'MRT', 'MUS', 'MWI', 'MYS', 'NAC',
       'NAM', 'NCL', 'NER', 'NGA', 'NIC', 'NLD', 'NOR', 'NPL', 'NRU',
       'NZL', 'OED', 'OMN', 'OSS', 'PAK', 'PAN', 'PER', 'PHL', 'PLW',
       'PNG', 'POL', 'PRE', 'PRI', 'PRK', 'PRT', 'PRY', 'PSE', 'PSS',
       'PST', 'PYF', 'QAT', 'ROU', 'RUS', 'RWA', 'SAS', 'SAU', 'SDN',
       'SEN', 'SGP', 'SLB', 'SLE', 'SLV', 'SMR', 'SOM', 'SRB', 'SSA',
       'SSD', 'SSF', 'SST', 'STP', 'SUR', 'SVK', 'SVN', 'SWE', 'SWZ',
       'SXM', 'SYC', 'SYR', 'TCA', 'TCD', 'TEA', 'TEC', 'TGO', 'THA',
       'TJK', 'TKM', 'TLA', 'TLS', 'TMN', 'TON', 'TSA', 'TSS', 'TTO',
       'TUN', 'TUR', 'TUV', 'TZA', 'UGA', 'UKR', 'UMC', 'URY', 'USA',
       'UZB', 'VCT', 'VEN', 'VGB', 'VIR', 'VNM', 'VUT', 'WLD', 'WSM',
       'XKX', 'YEM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)
```

In [20]:
```
df_combined['Indicator Name'].unique()
```

Out[20]:
```
array(['Population ages 0-14 (% of total population)',
       'Population ages 15-64 (% of total population)',
       'Population ages 65 and above (% of total population)'],
      dtype=object)
```

In [21]:
```
df_combined['Indicator Code'].unique()
```

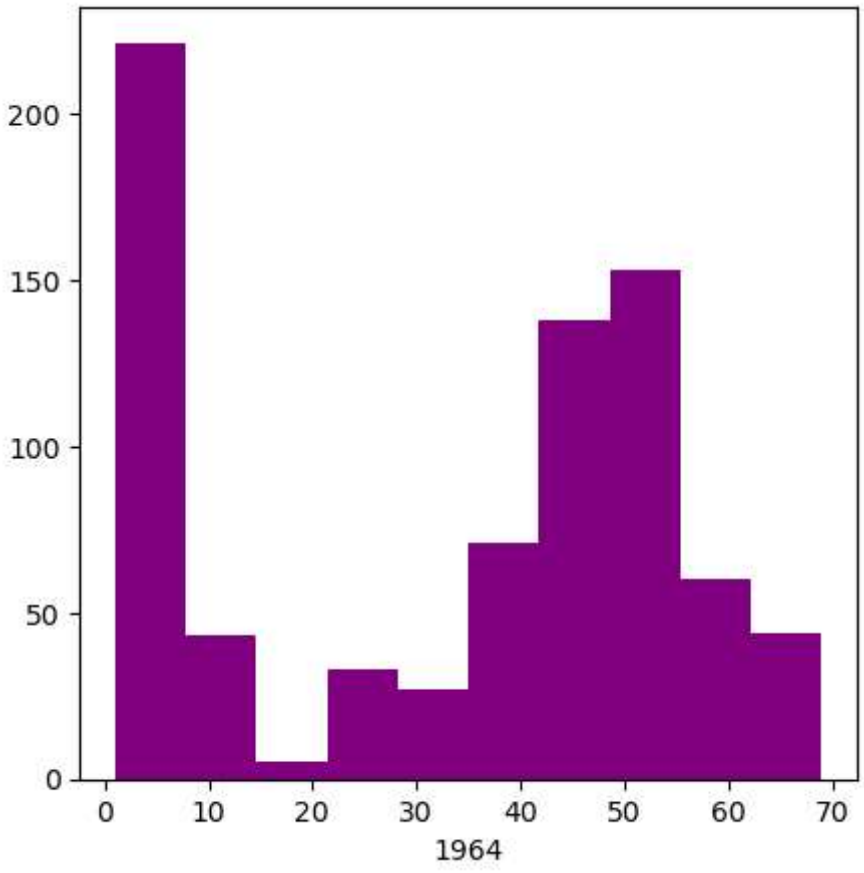
Out[21]:
```
array(['SP.POP.0014.TO.ZS', 'SP.POP.1564.TO.ZS', 'SP.POP.65UP.TO.ZS'],
      dtype=object)
```

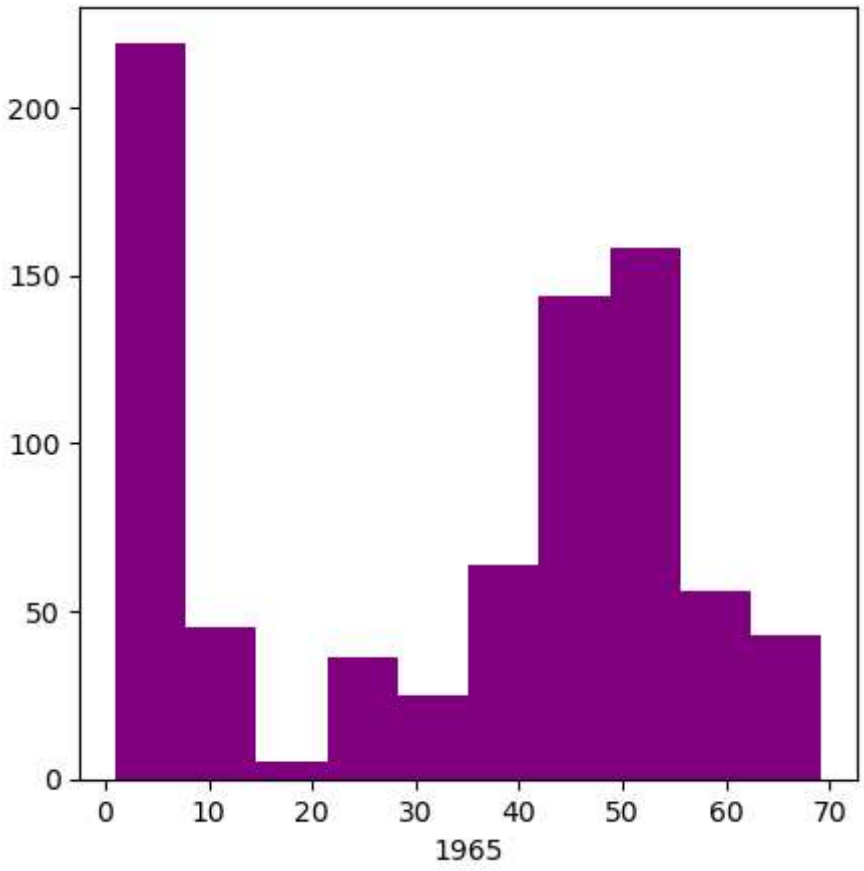
In [22]:
```
df_combined.drop(['Indicator Name','Indicator Code','Country Code'],axis=1,inplace=Tru
```

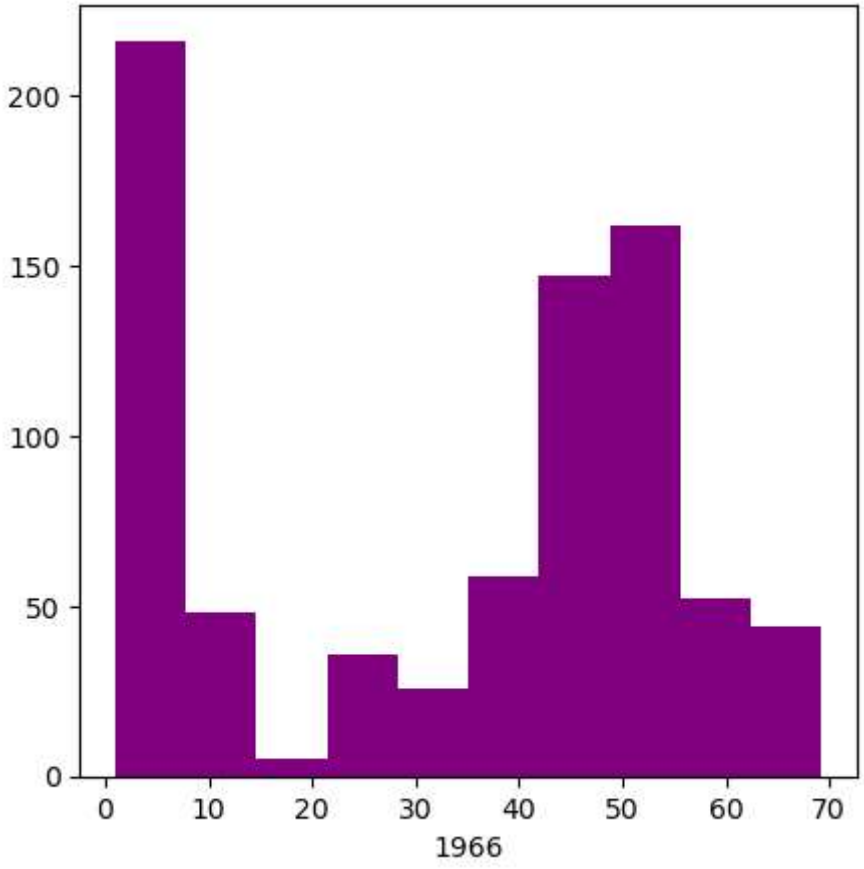
In [23]:
```
selected_columns_data.columns
```

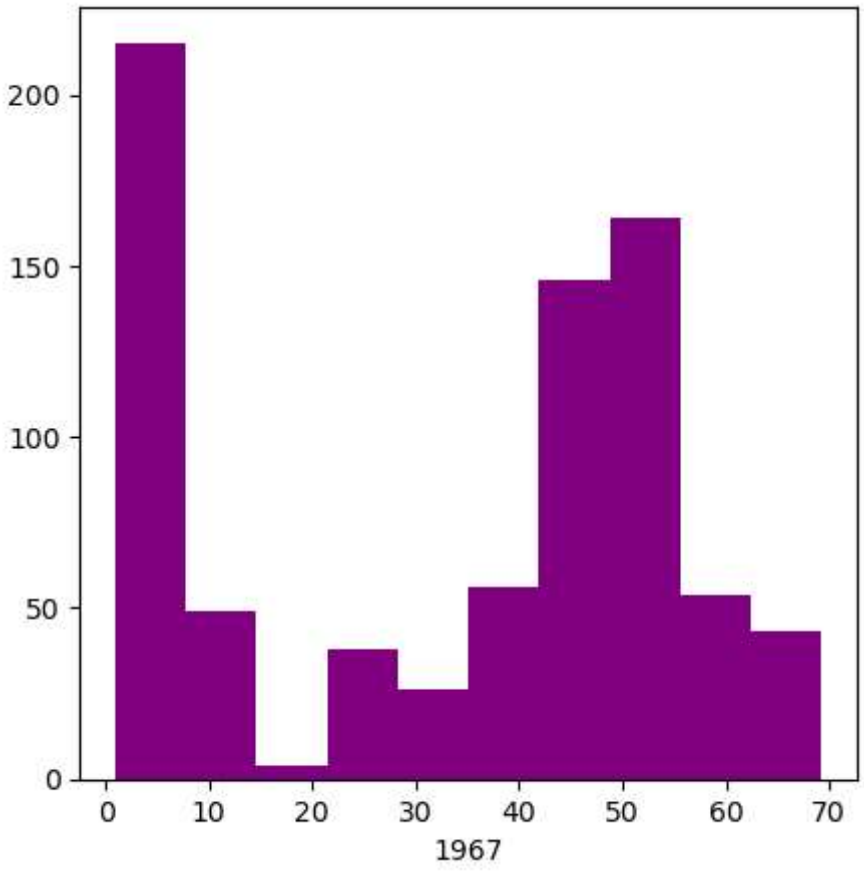
Out[23]:
```
Index(['1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022'],
      dtype='object')
```

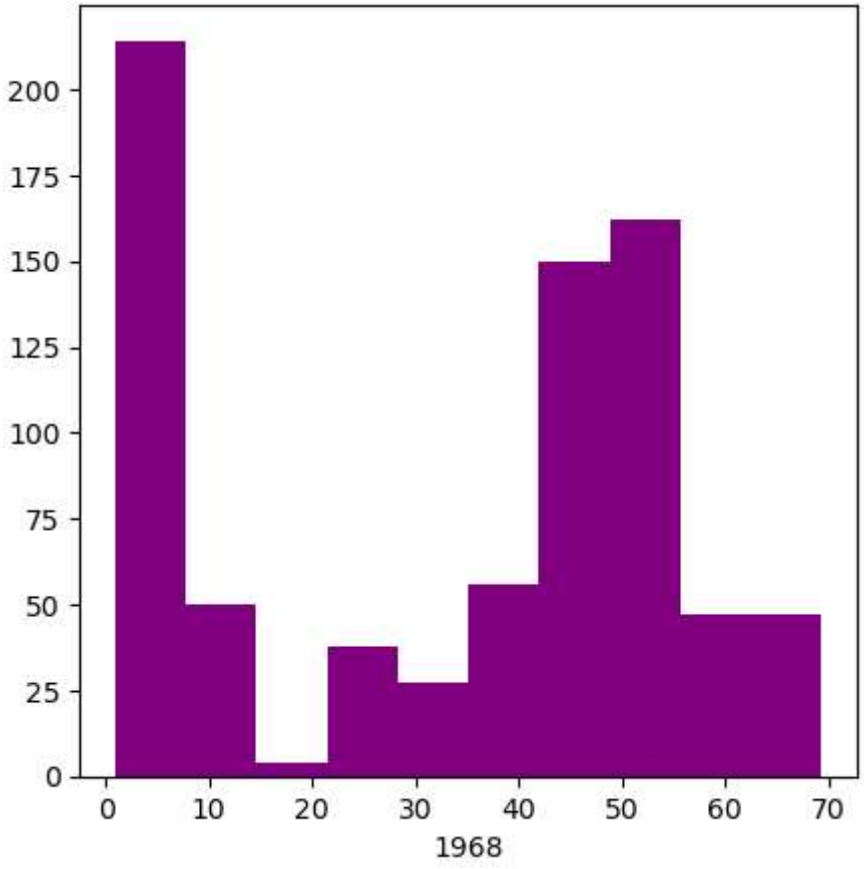
In [24]:
```
import matplotlib.pyplot as plt
cols = selected_columns_data.columns[4:67].tolist()

for i in cols:
    fig = plt.figure(figsize=(5, 5))
    plt.hist(selected_columns_data[i], color='purple', bins=10)
    plt.xlabel(i)
    plt.show()
```

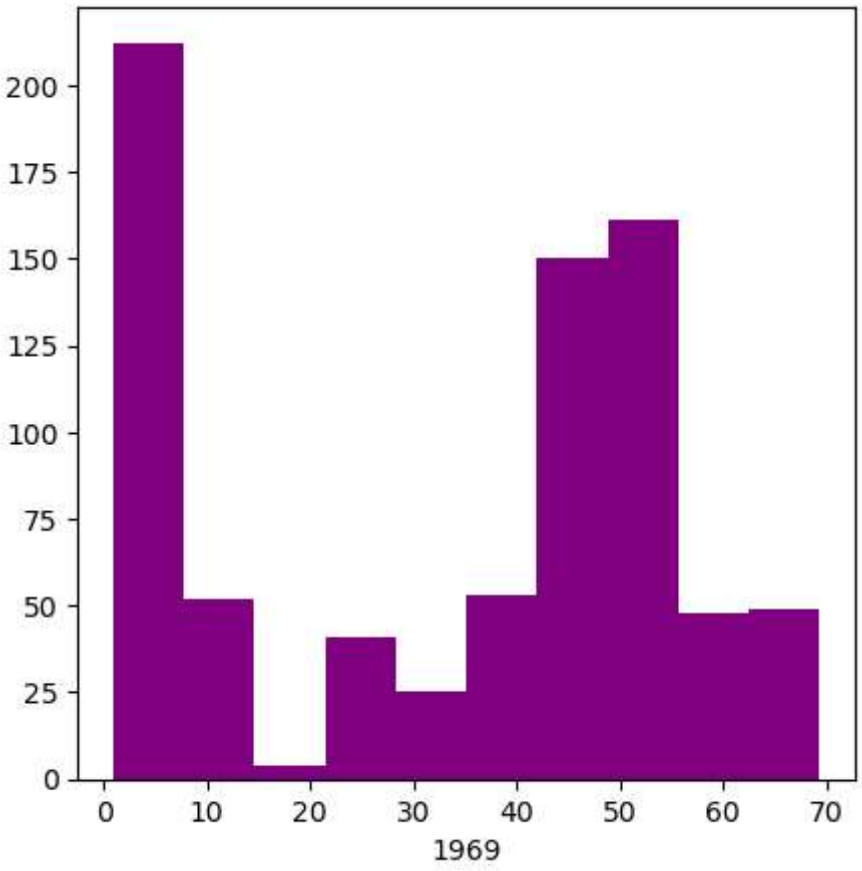


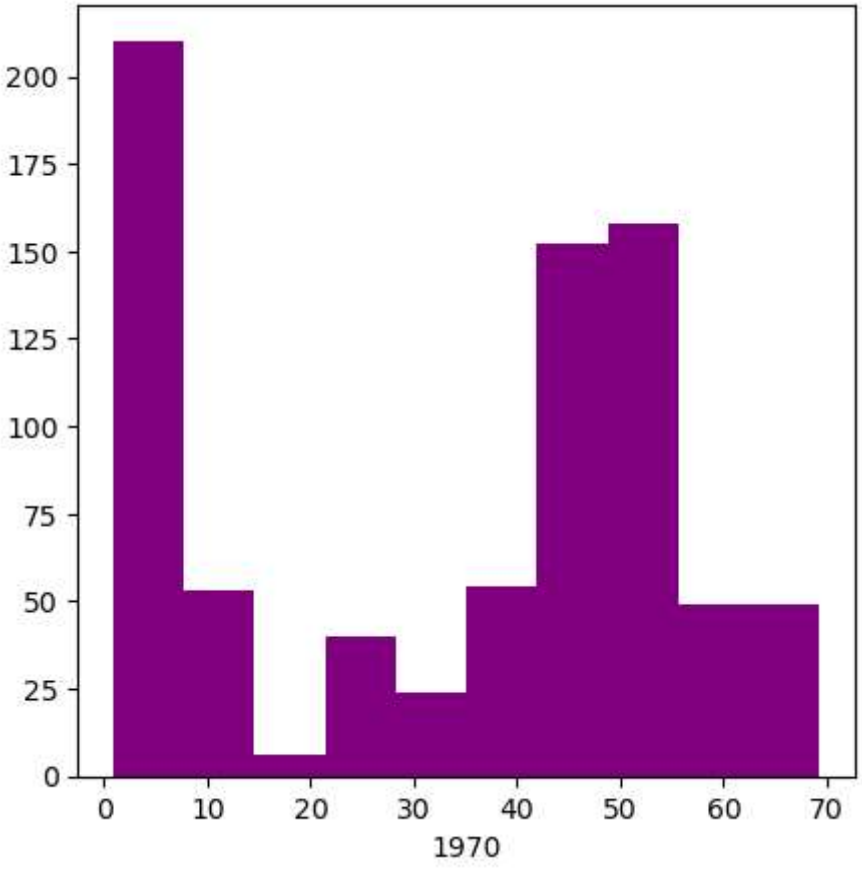

200
150
100
50
0
0
10
20
30
40
50
60
70
1965

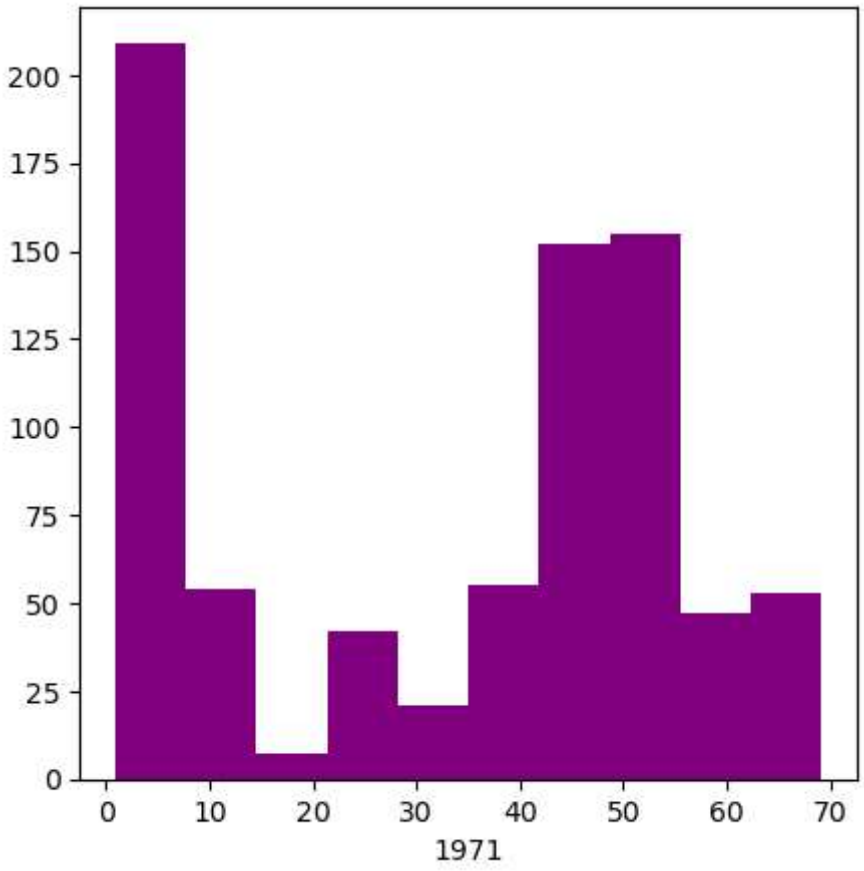

200
150
100
50
0
0
10
20
30
40
50
60
70
1966

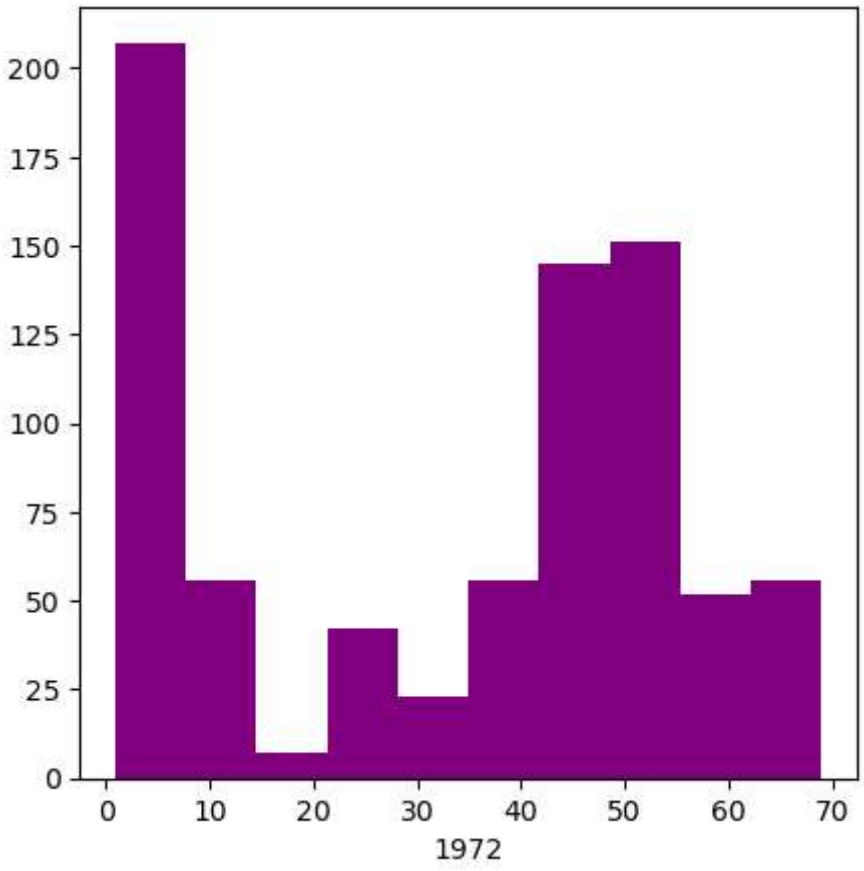

200
150
100
50
0
0
10
20
30
40
50
60
70
1967

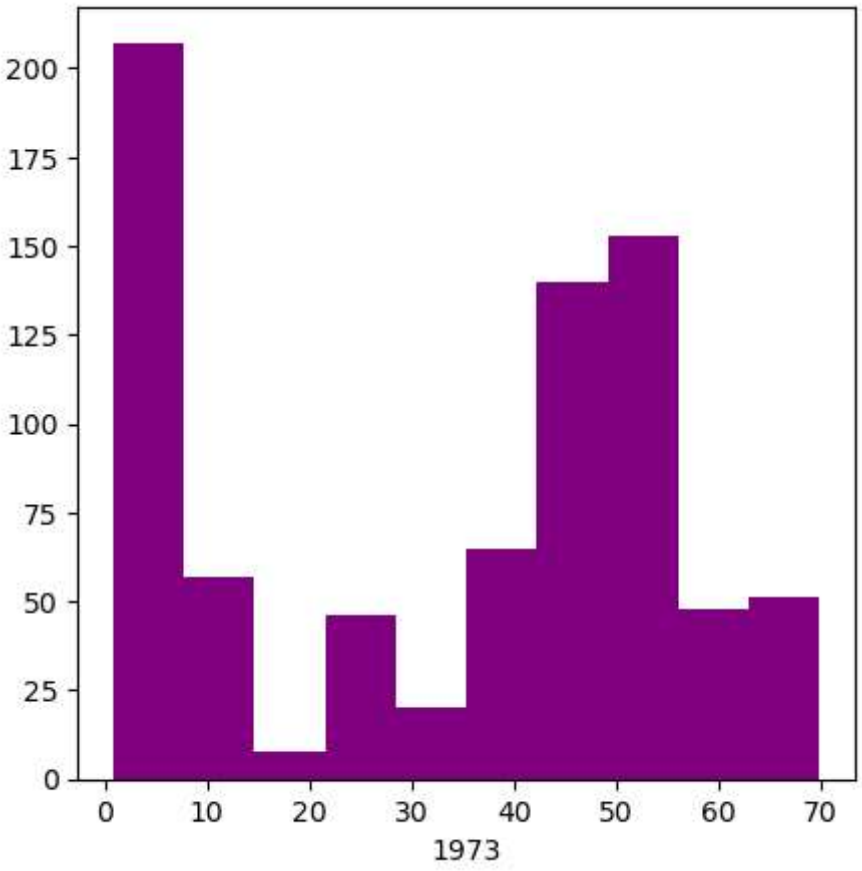

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1968

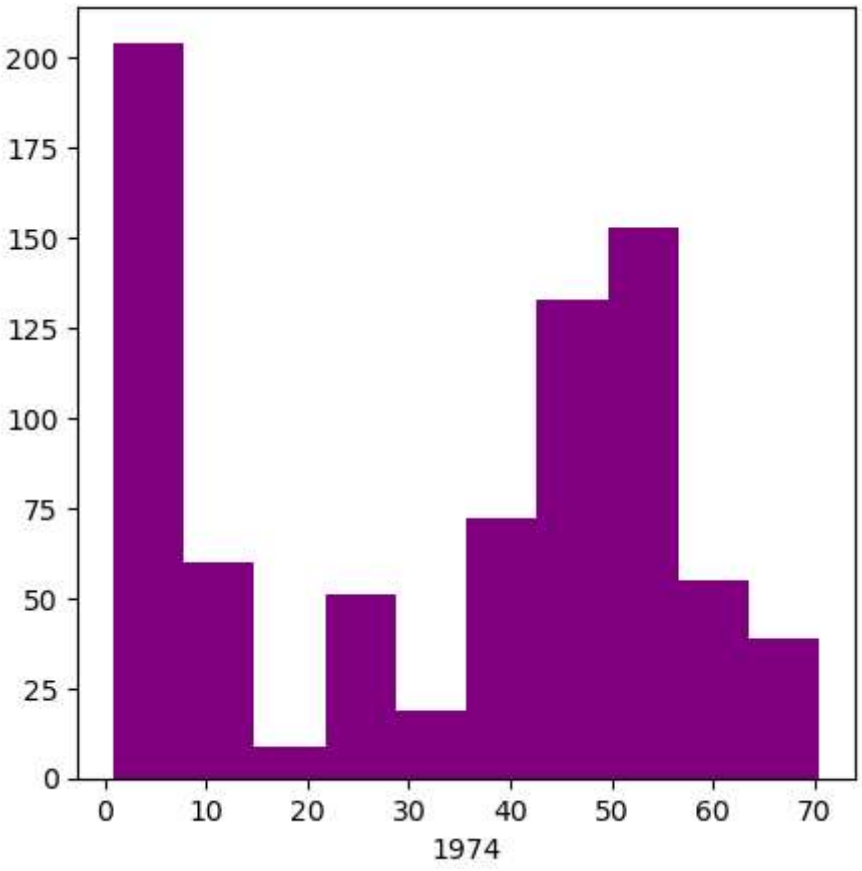

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1969

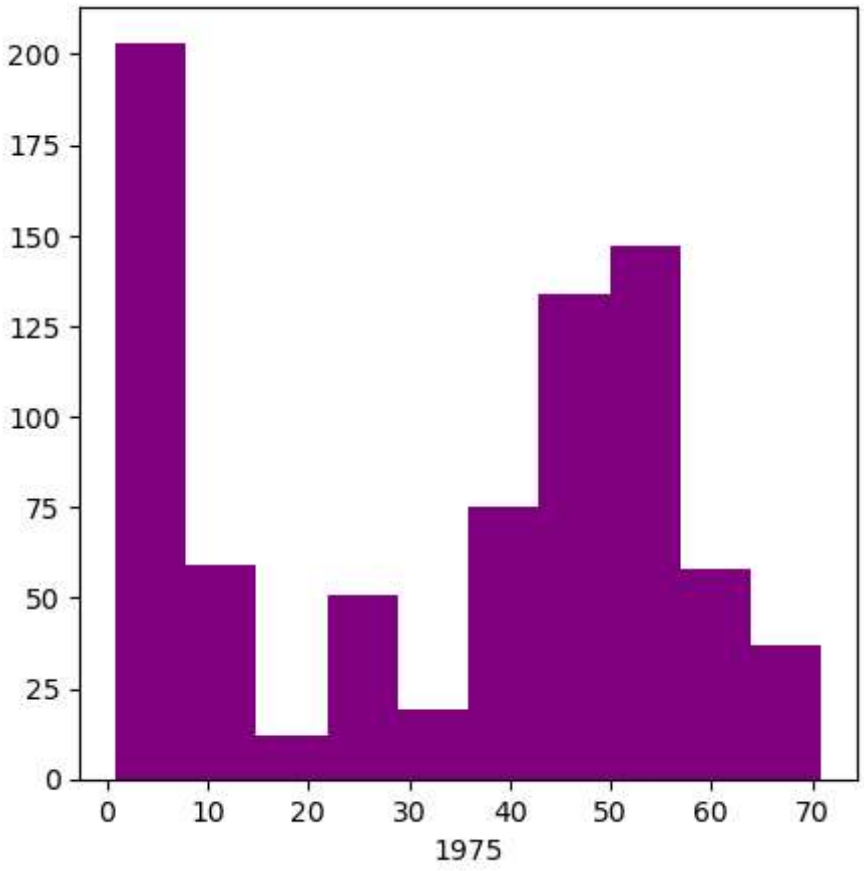

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1970

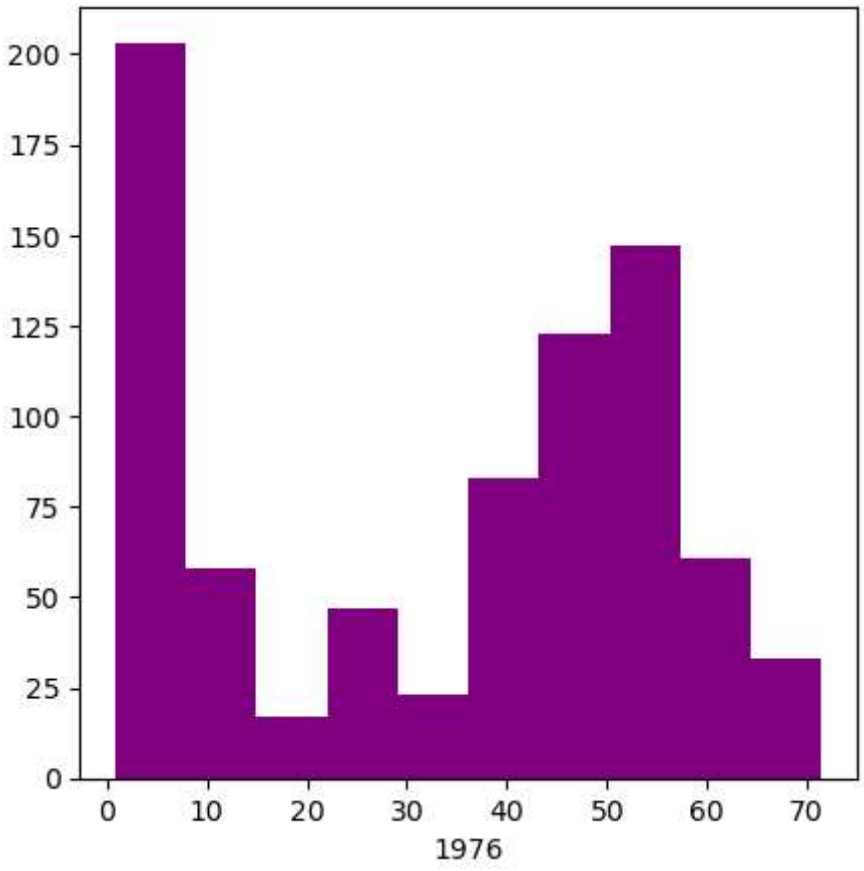

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1971

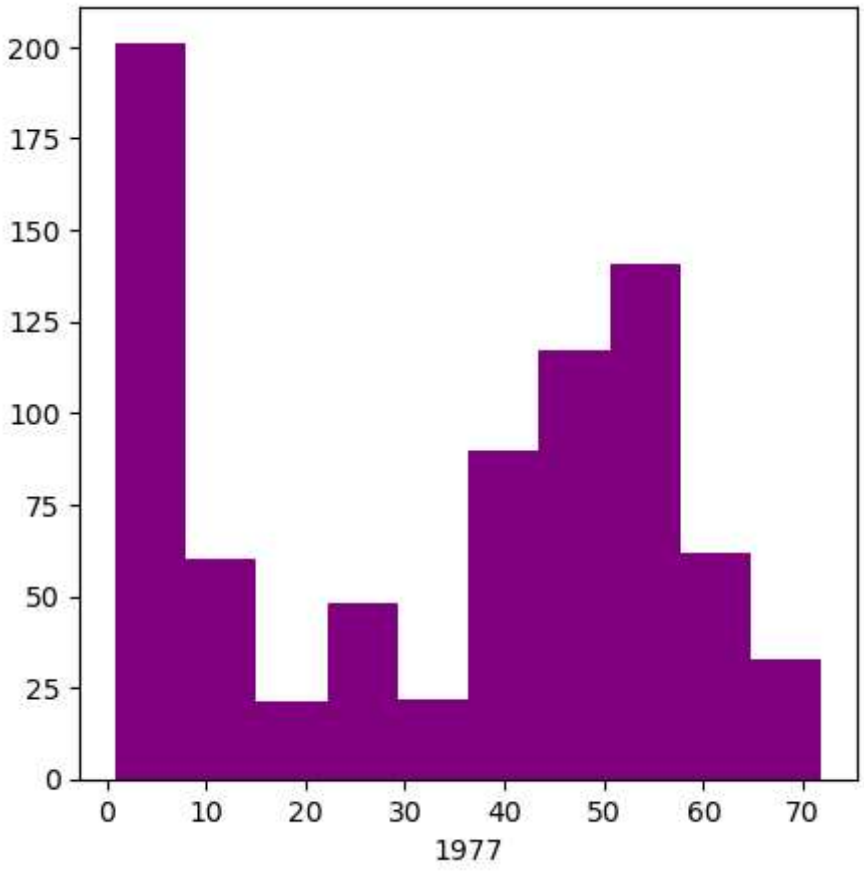

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1972

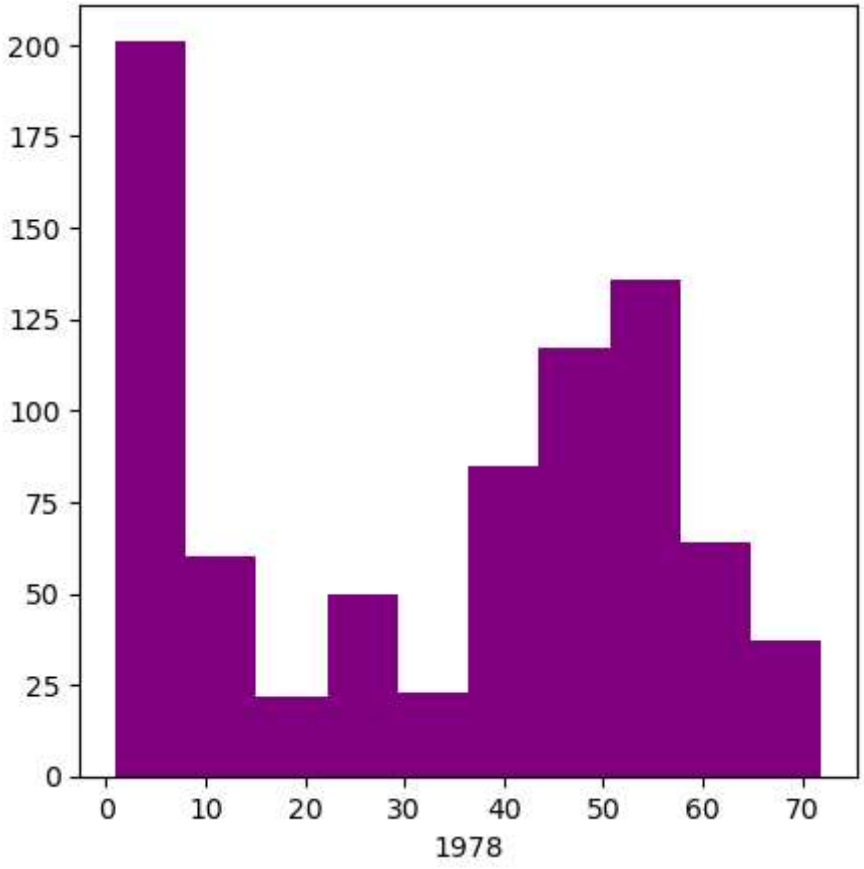

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1973

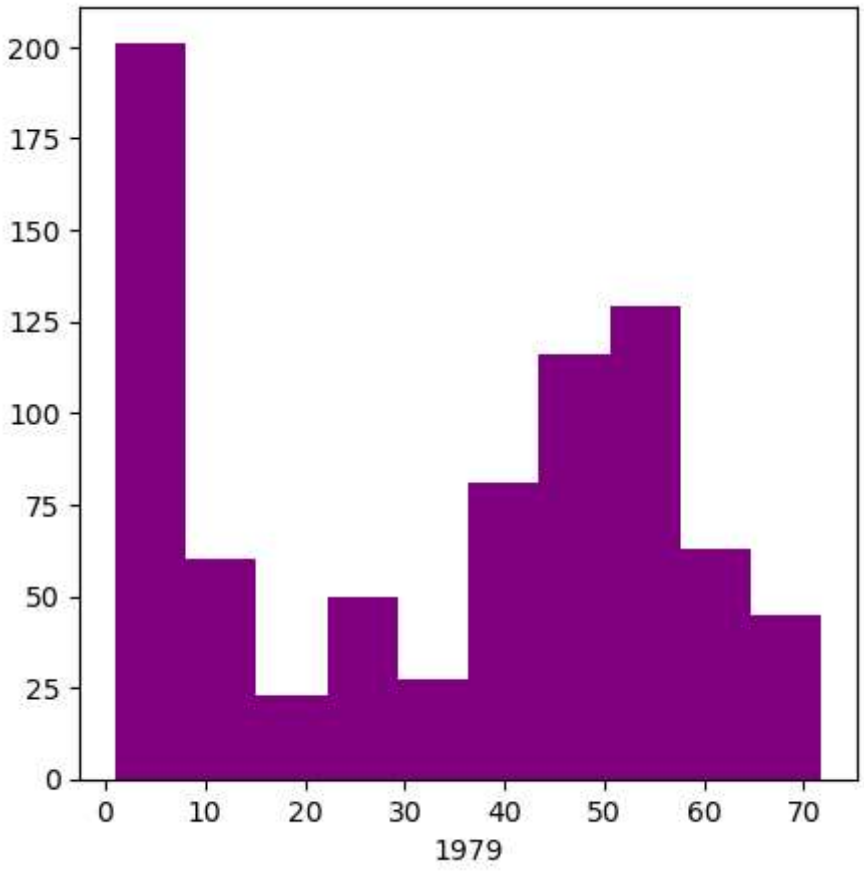

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1974

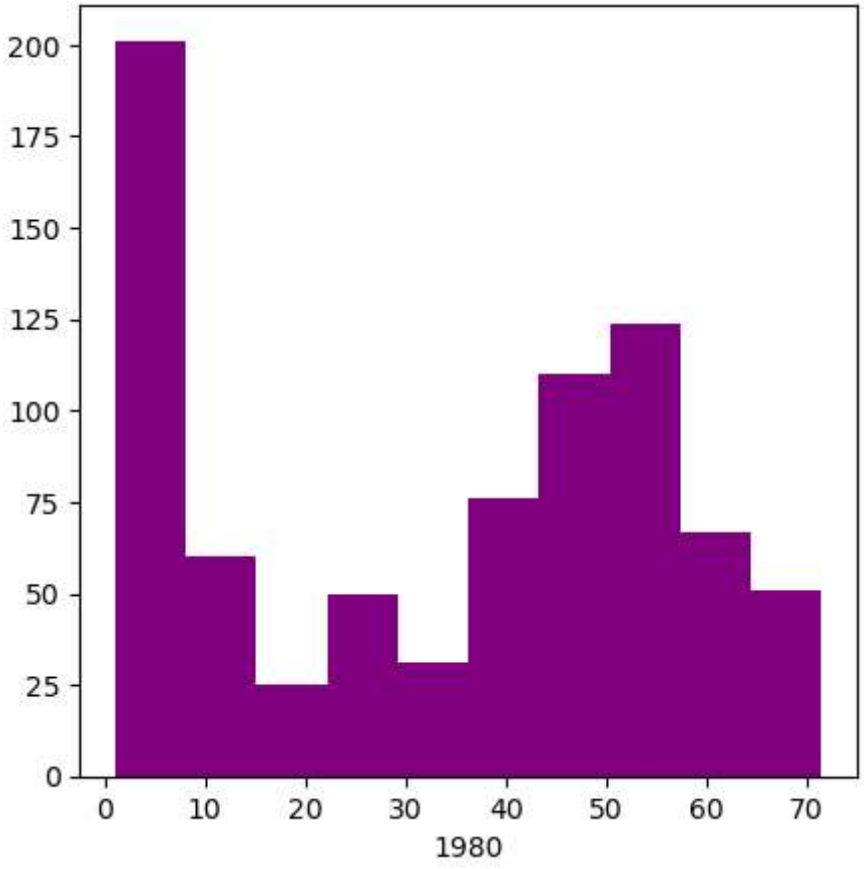

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1975

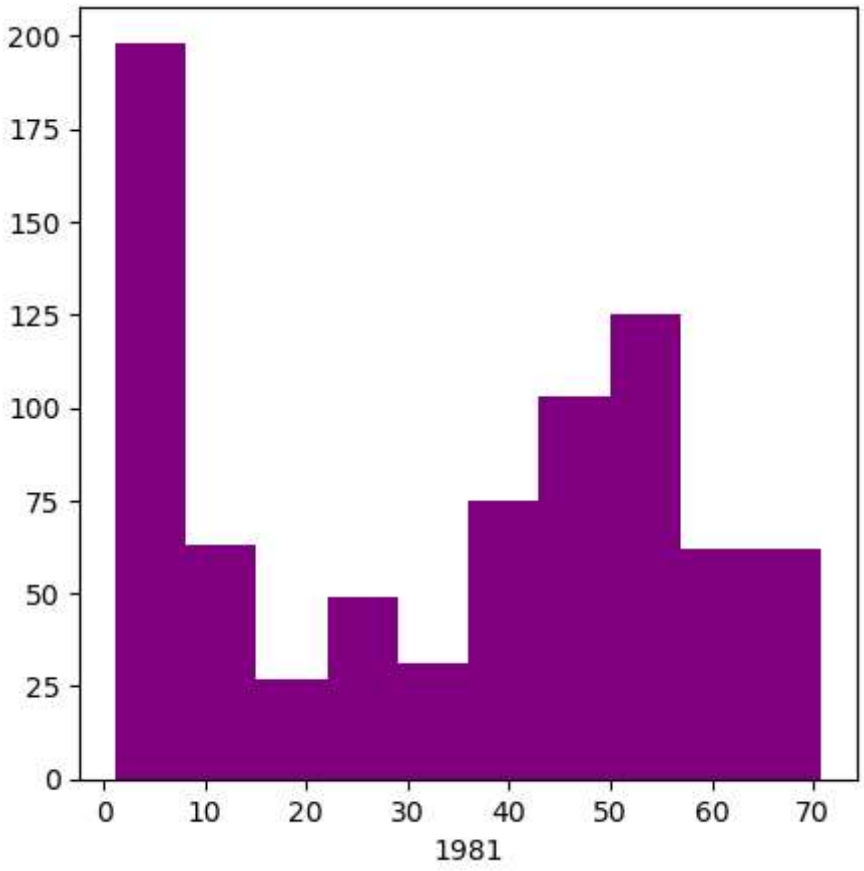

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1976

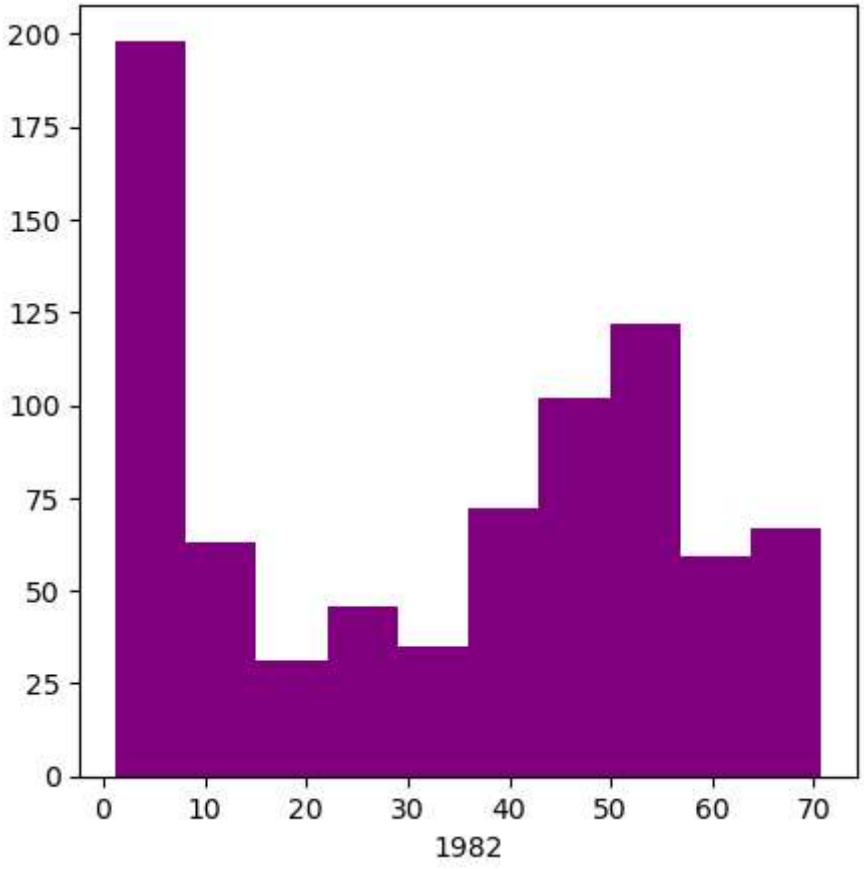

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1977

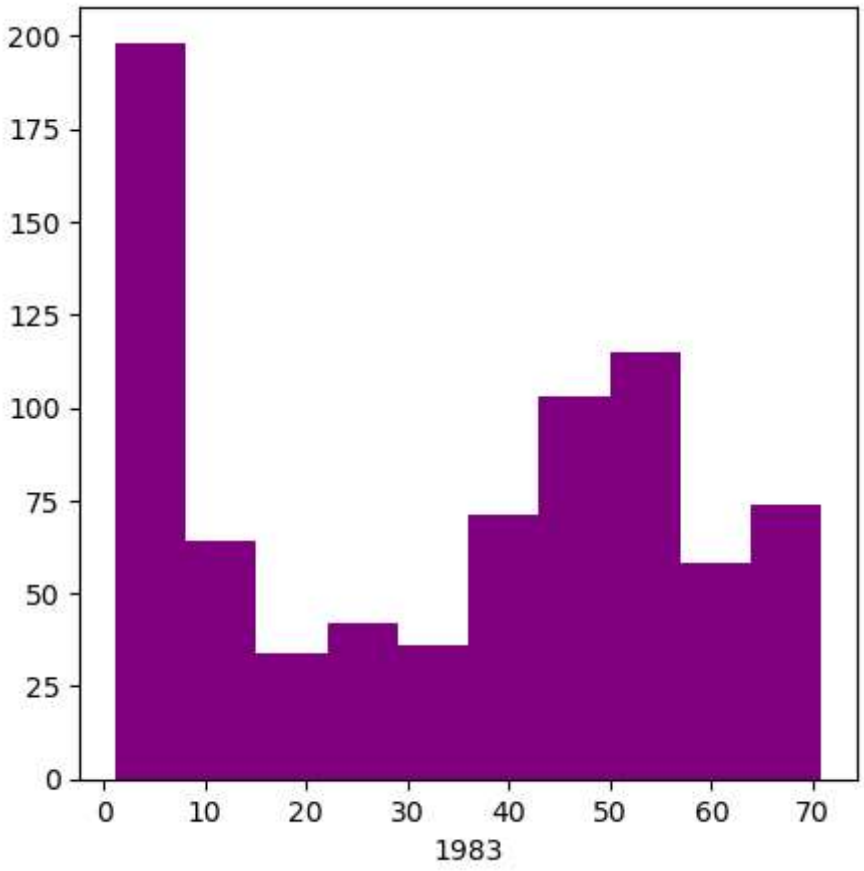

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1978

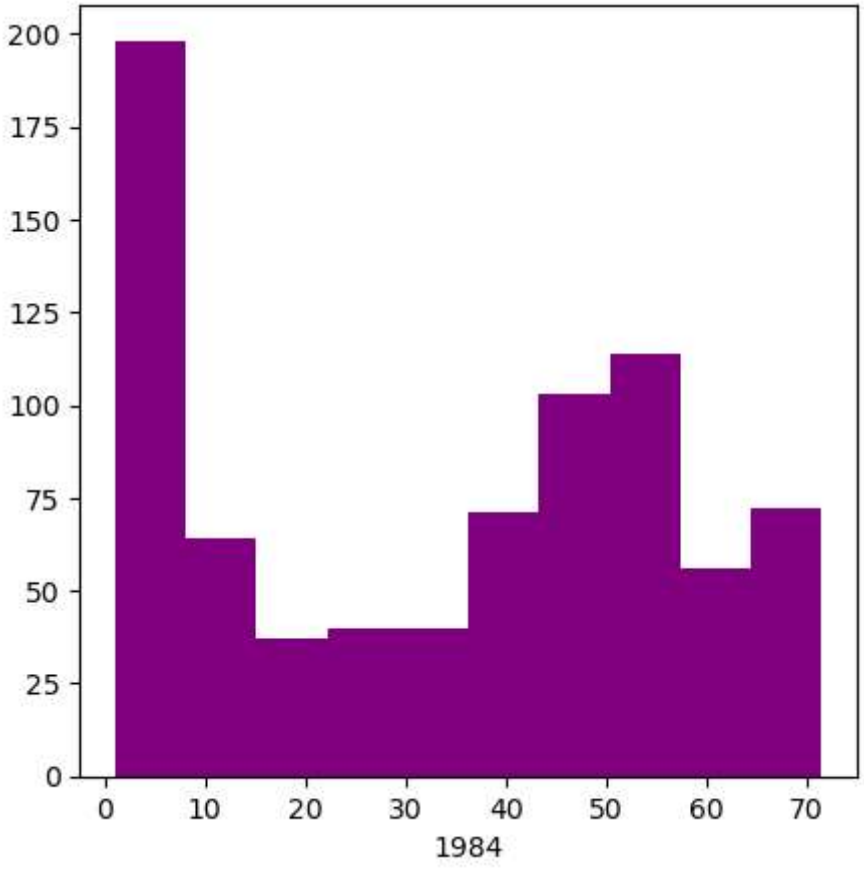

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1979

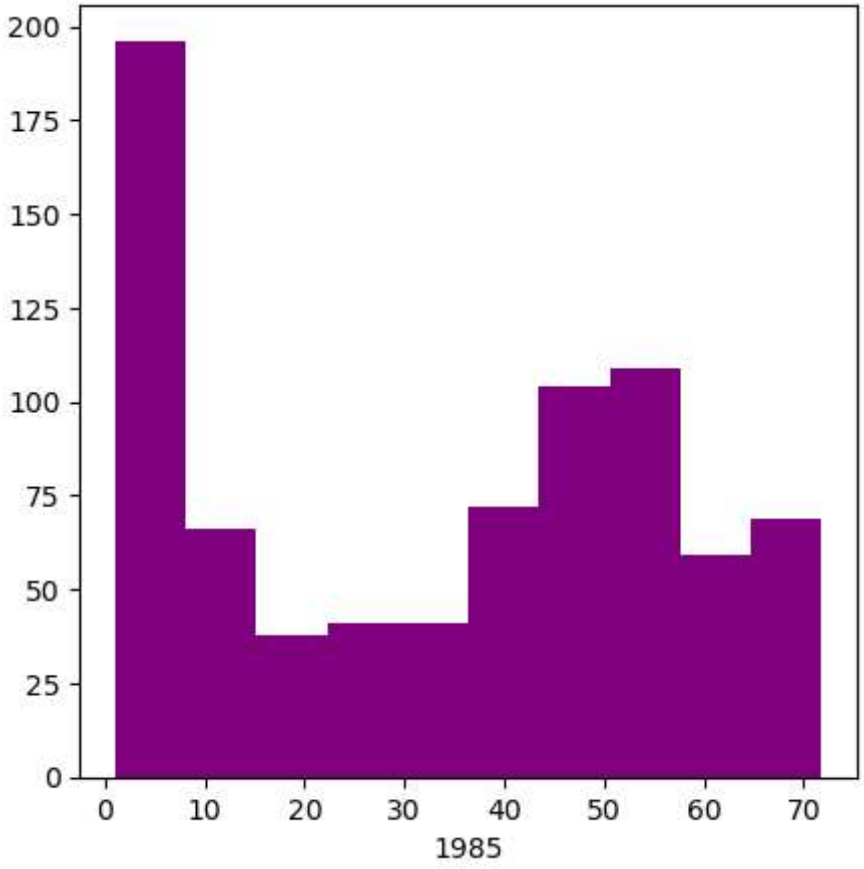

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1980

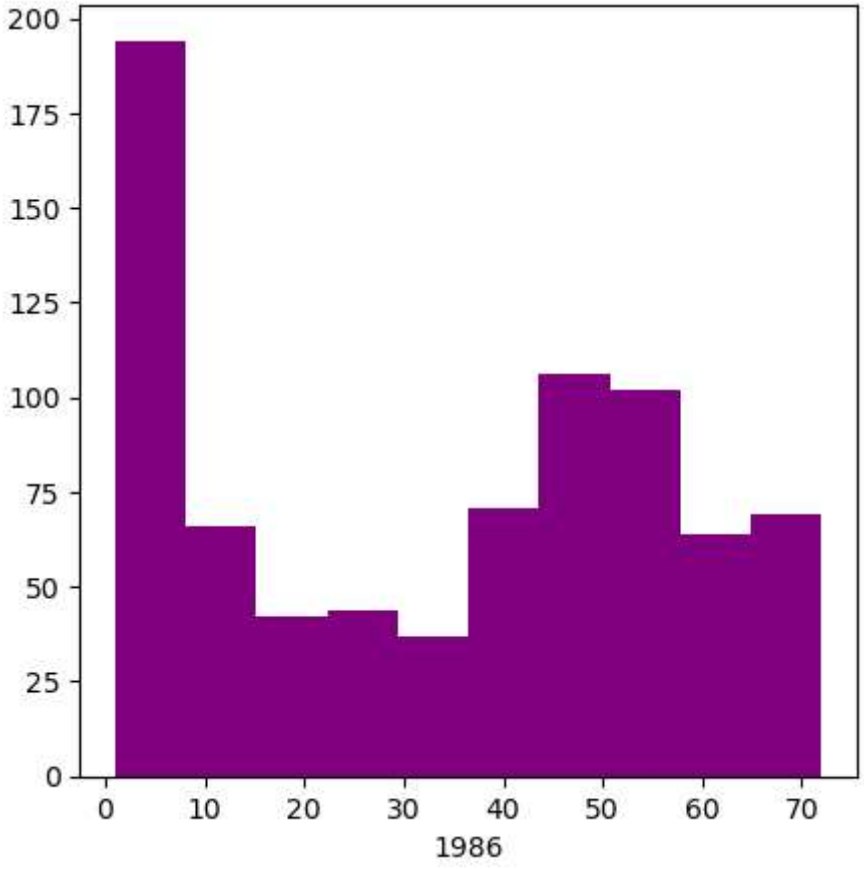

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1981

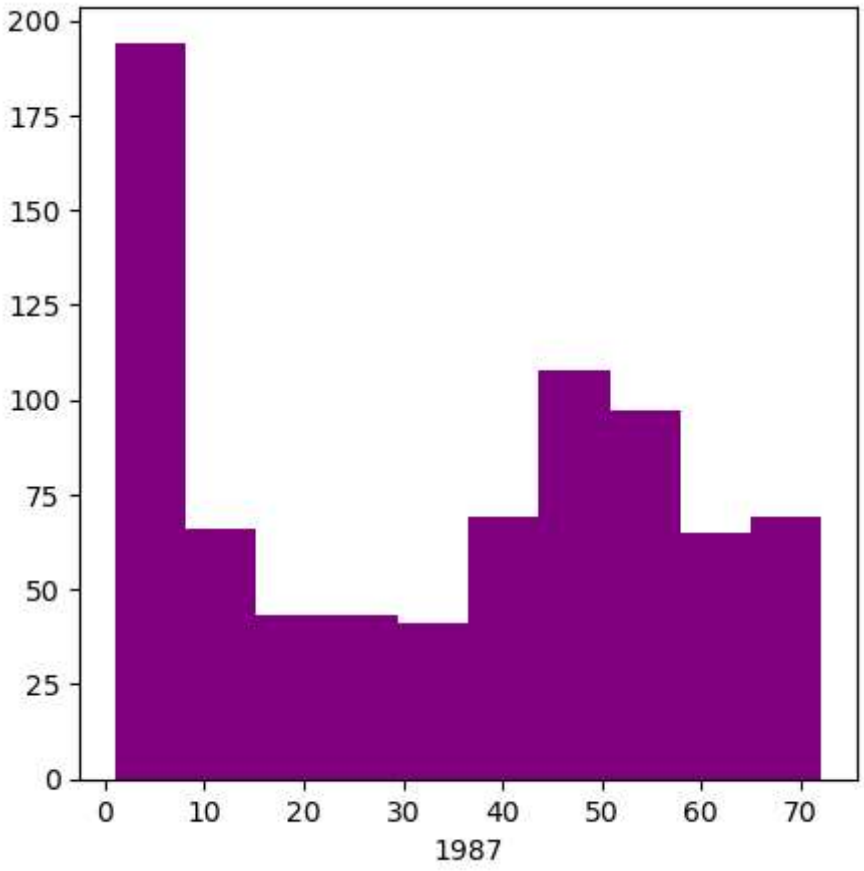

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1982

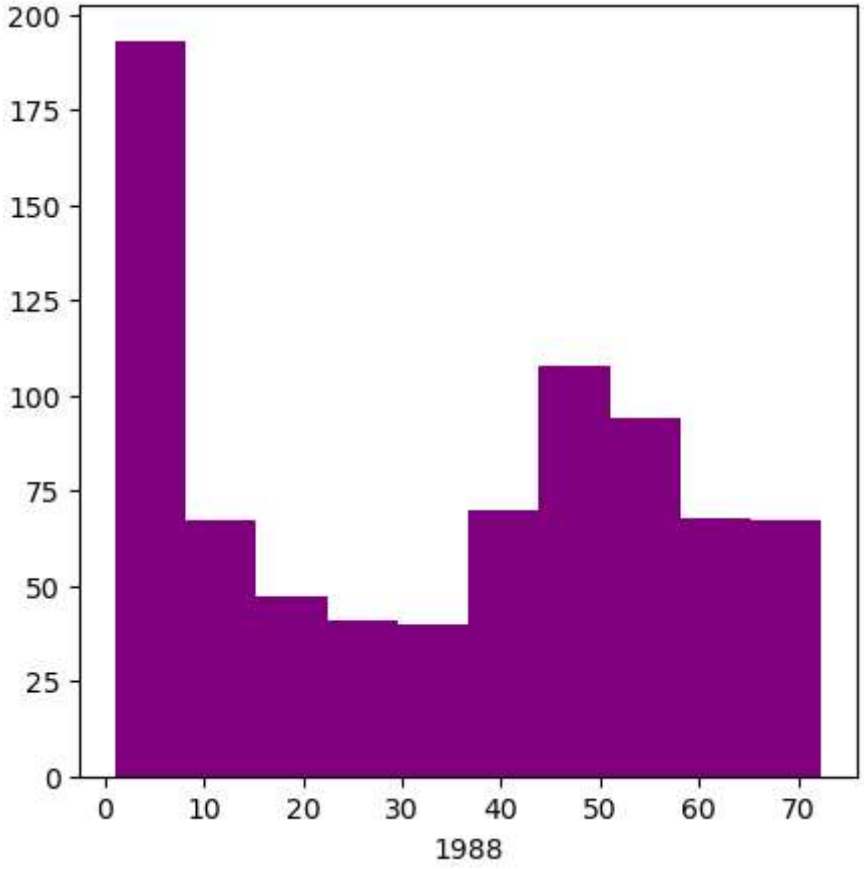

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1983

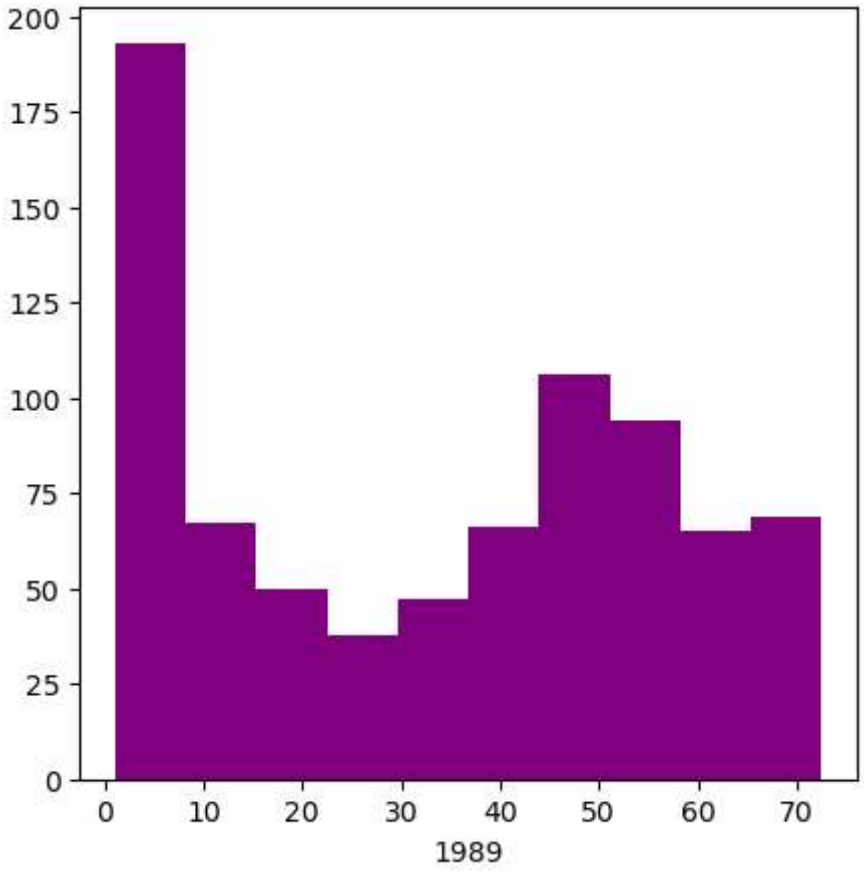

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1984

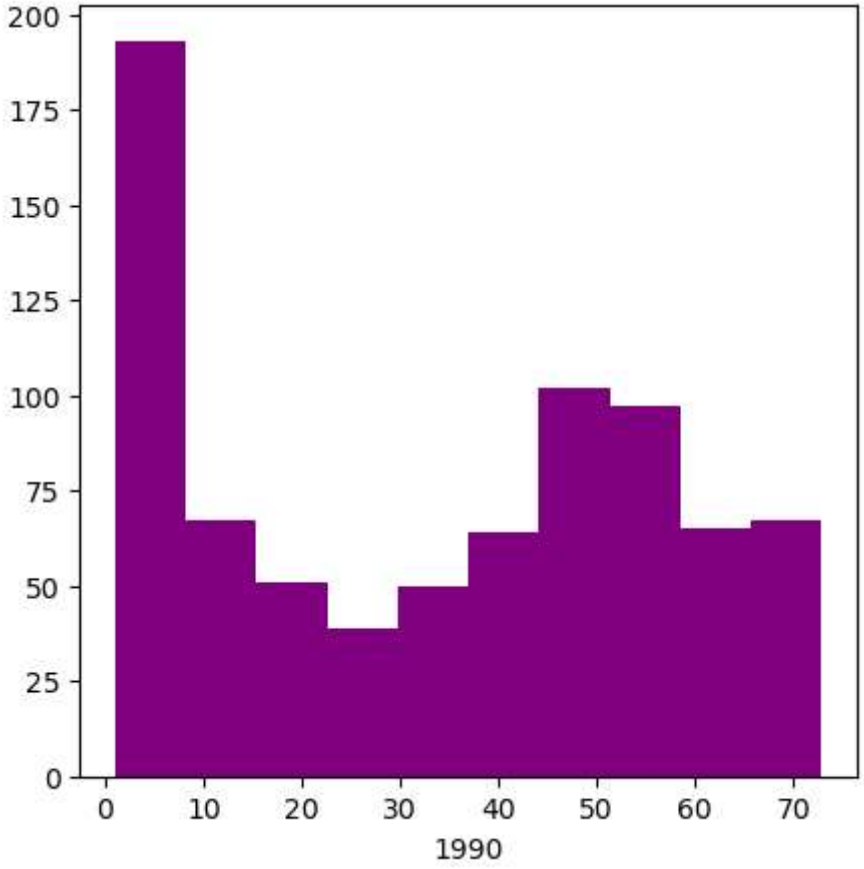

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1985

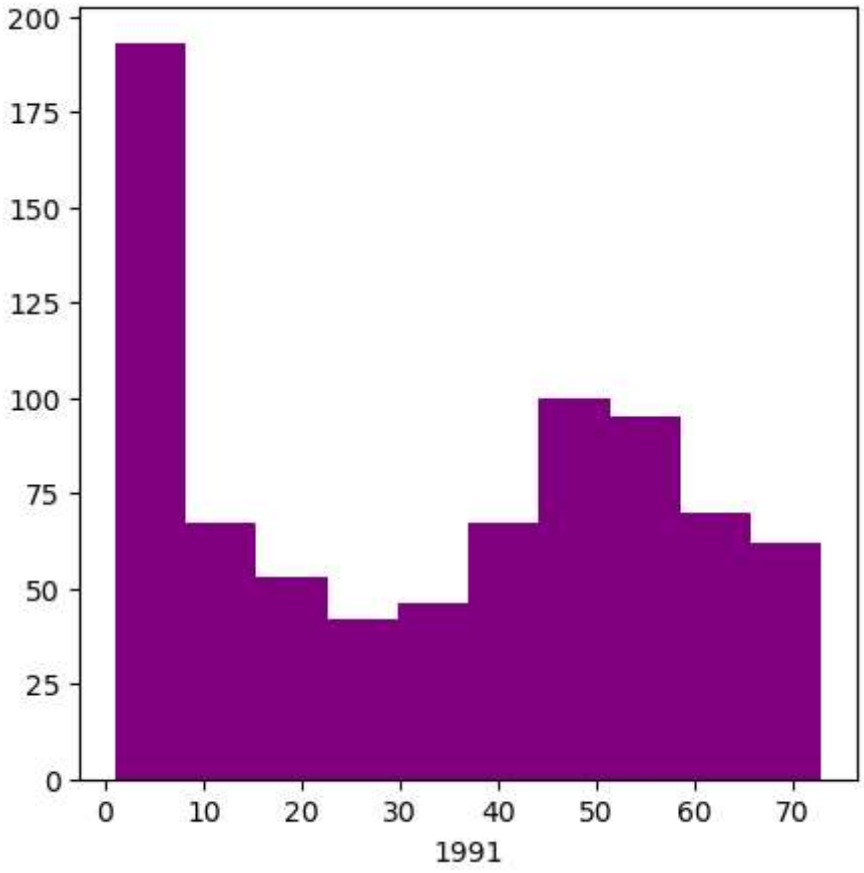

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1986

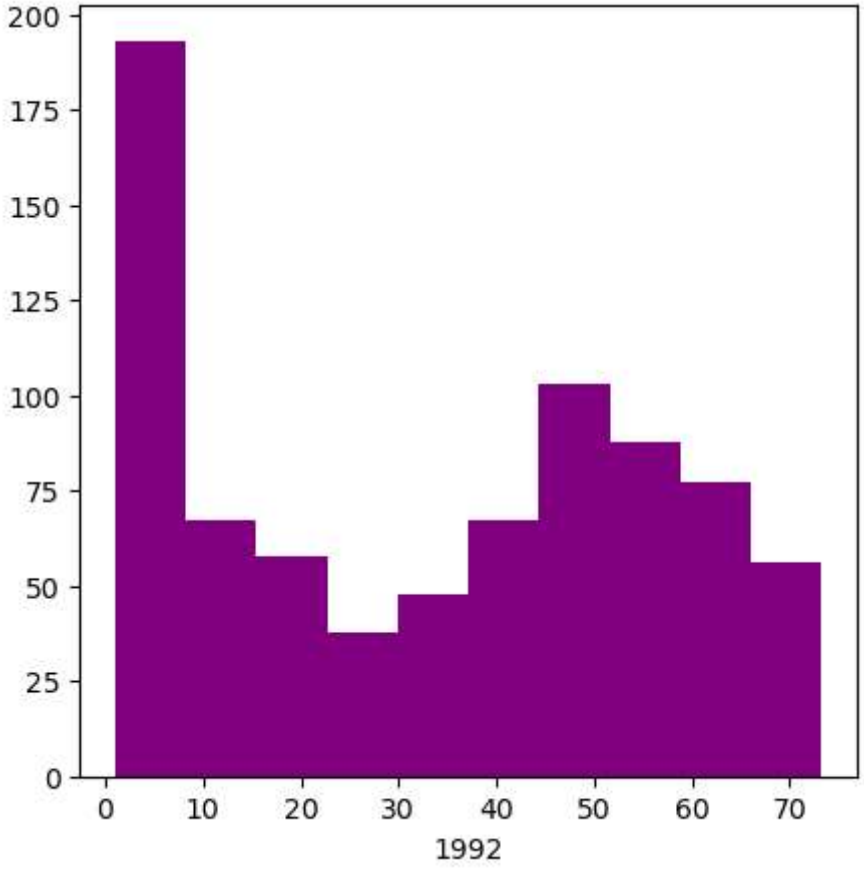

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1987

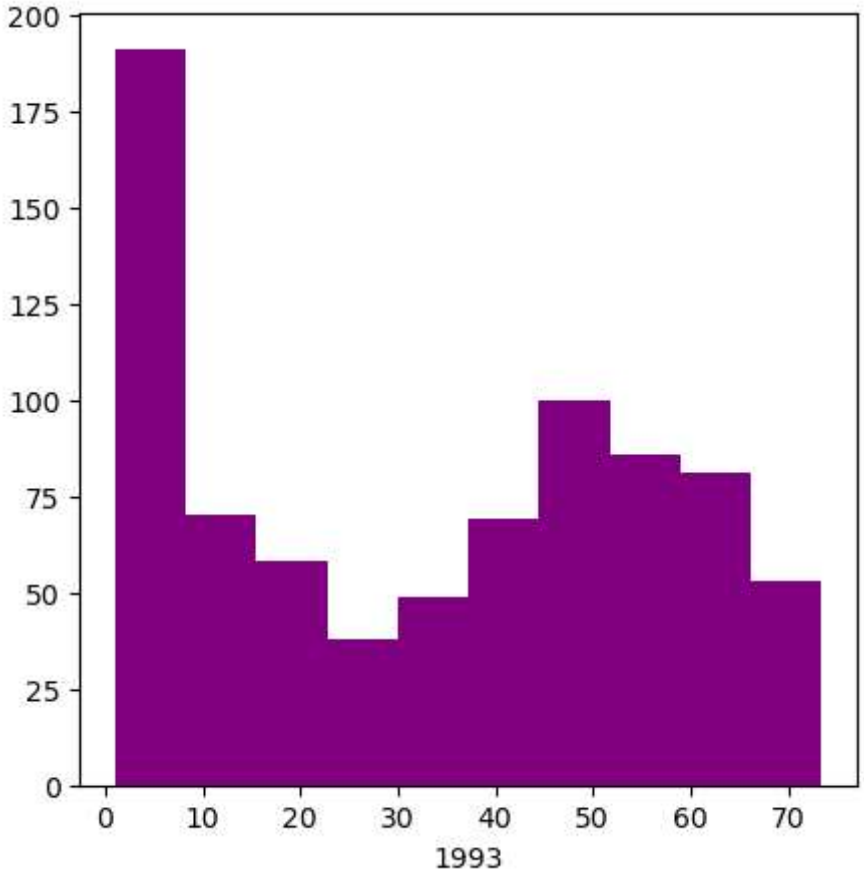

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1988

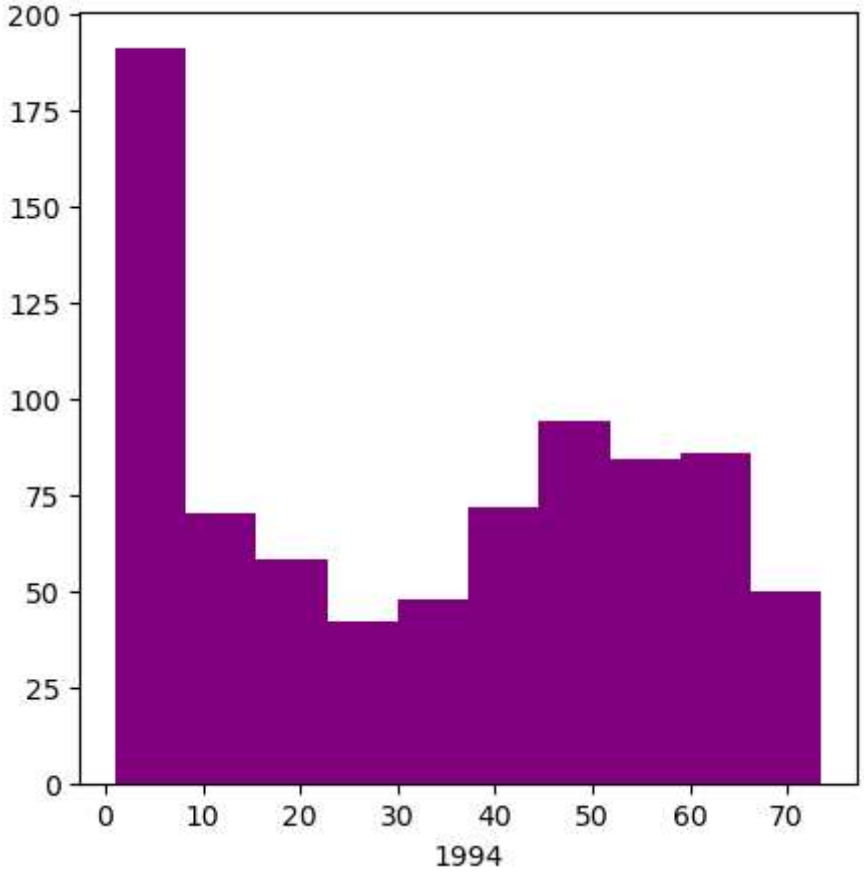

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1989

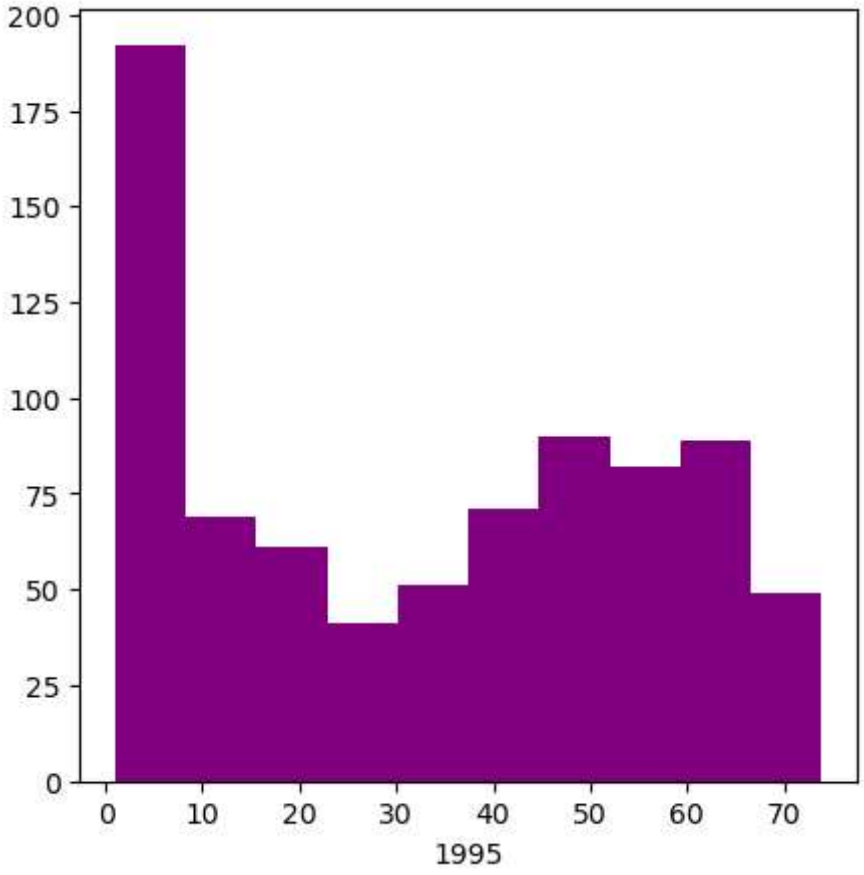

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1990

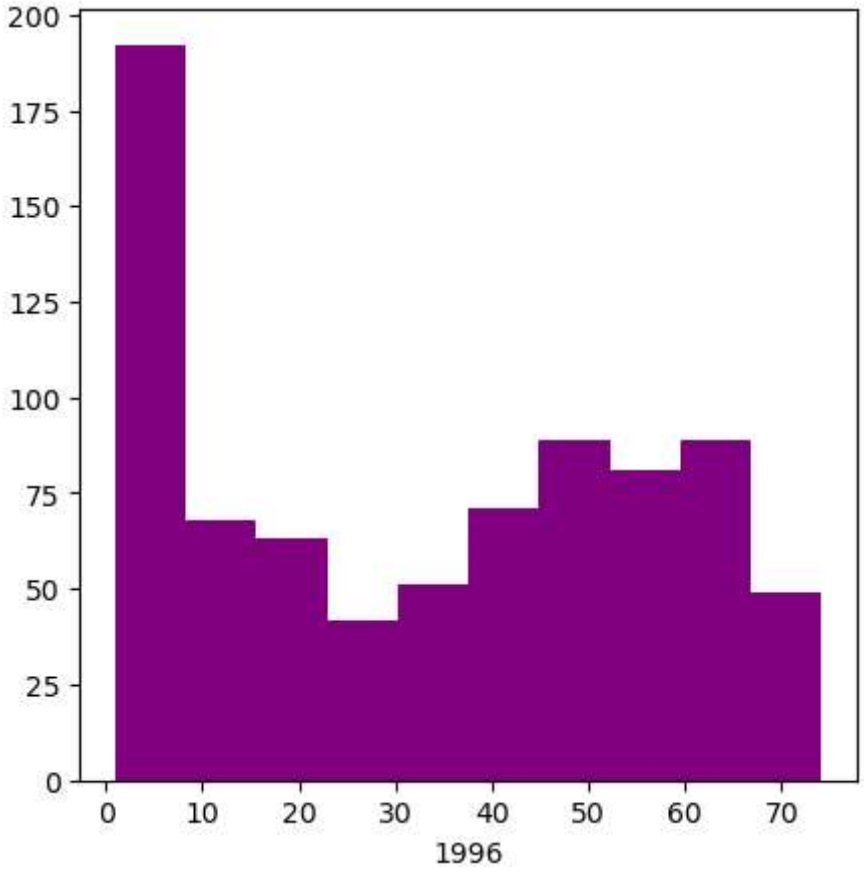

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1991

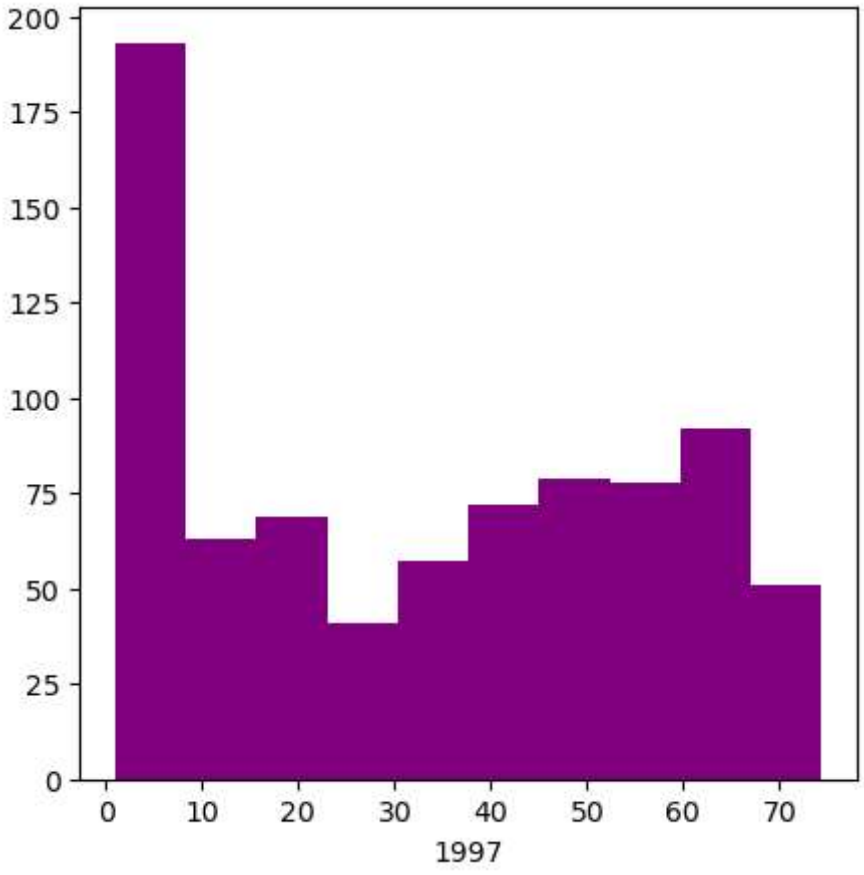

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1992

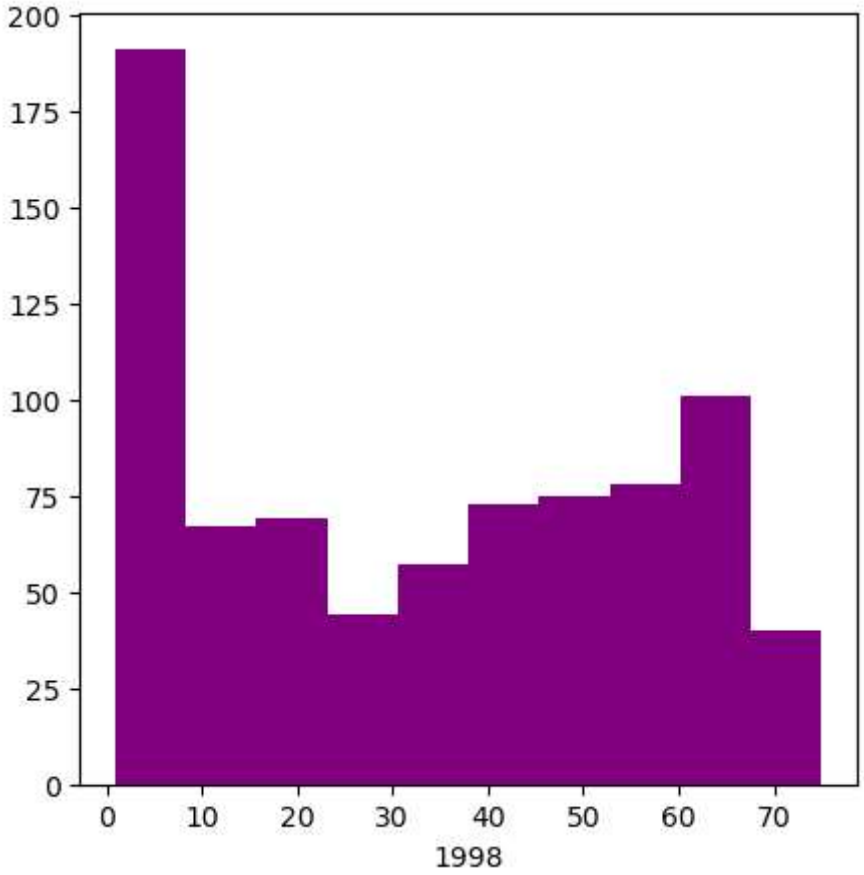

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1993

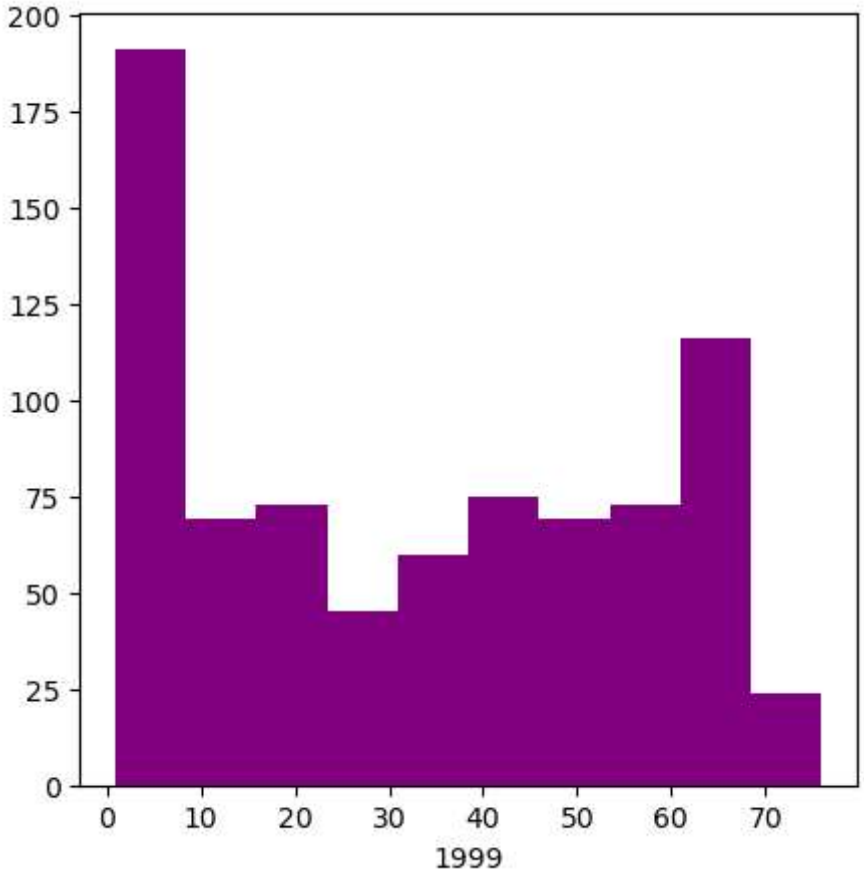

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1994

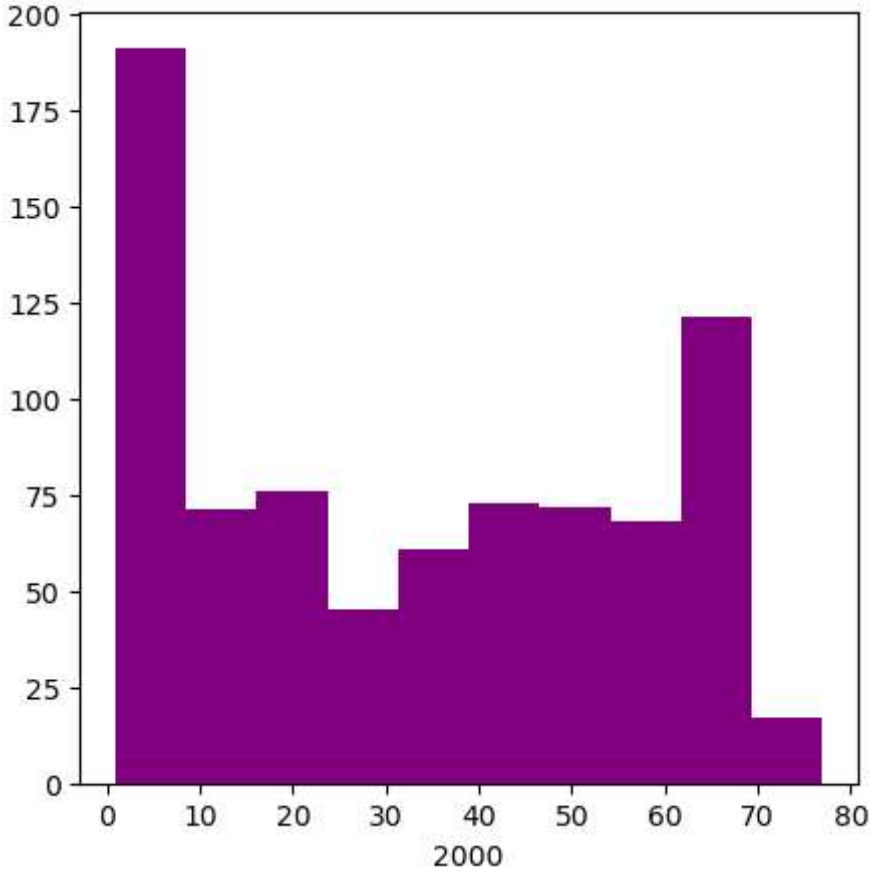

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1995

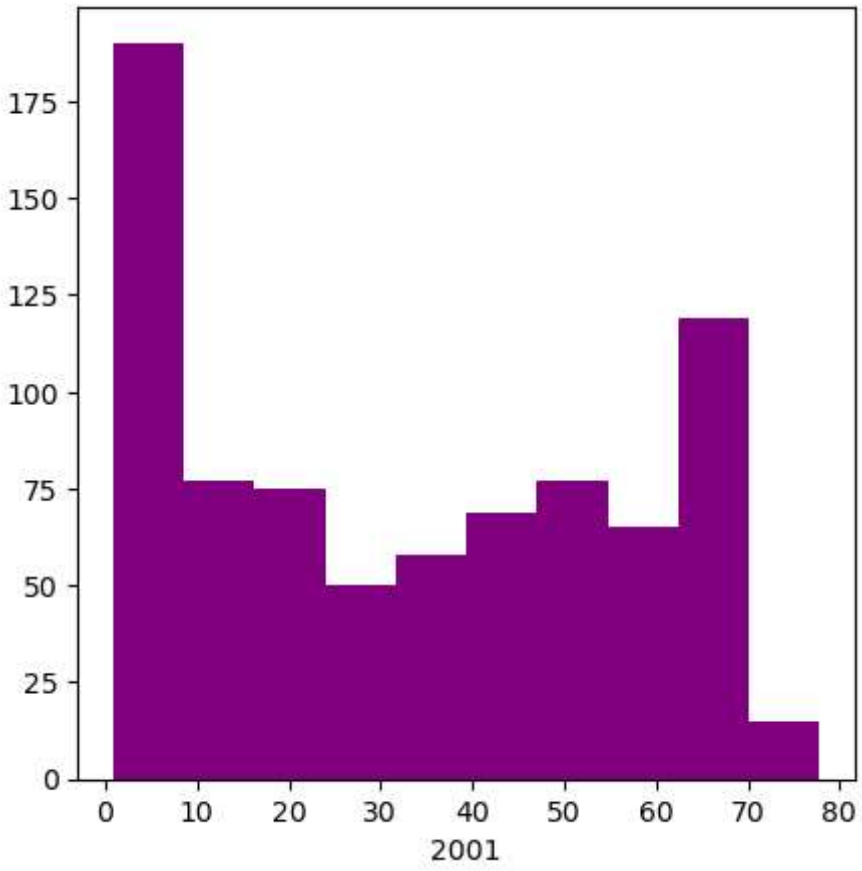

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1996

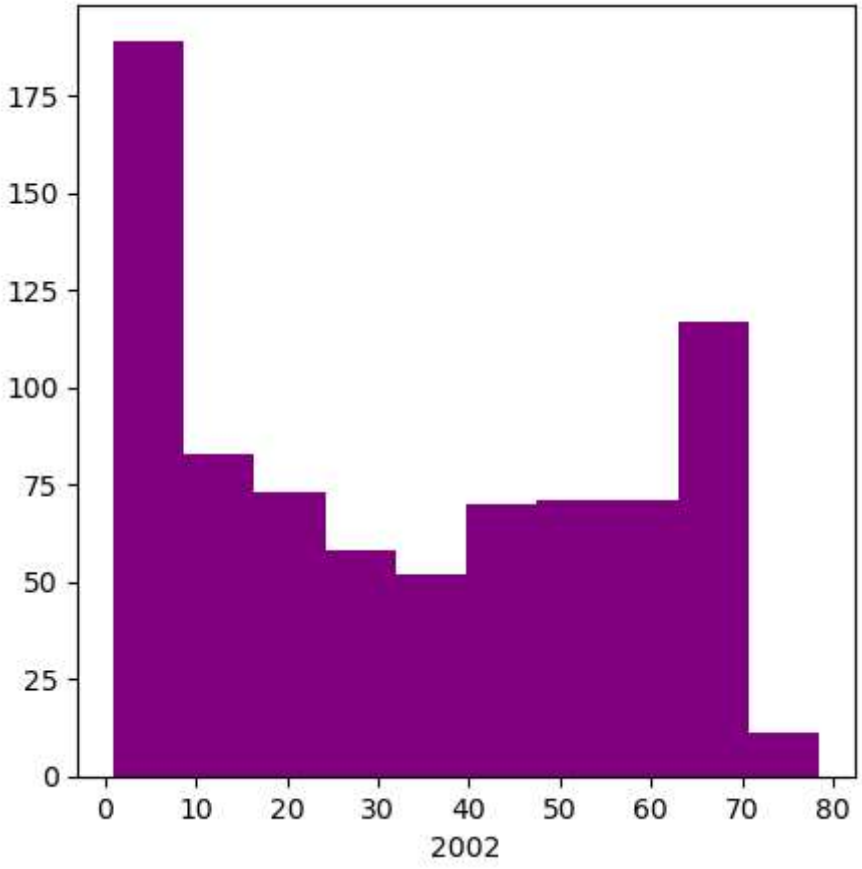

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1997

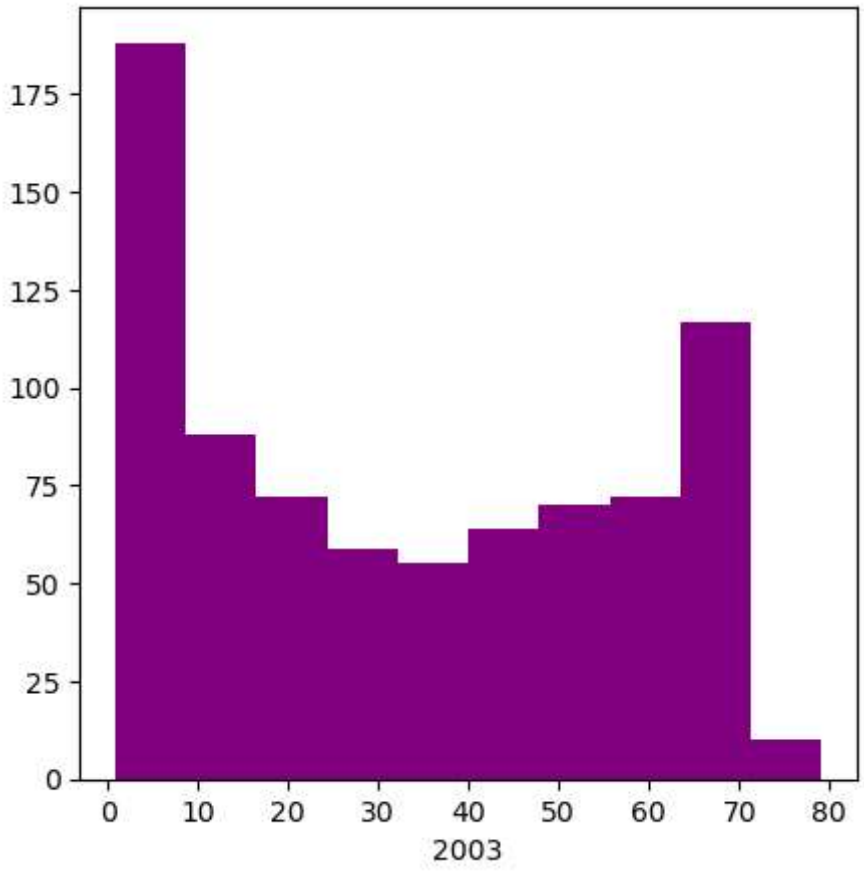

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1998

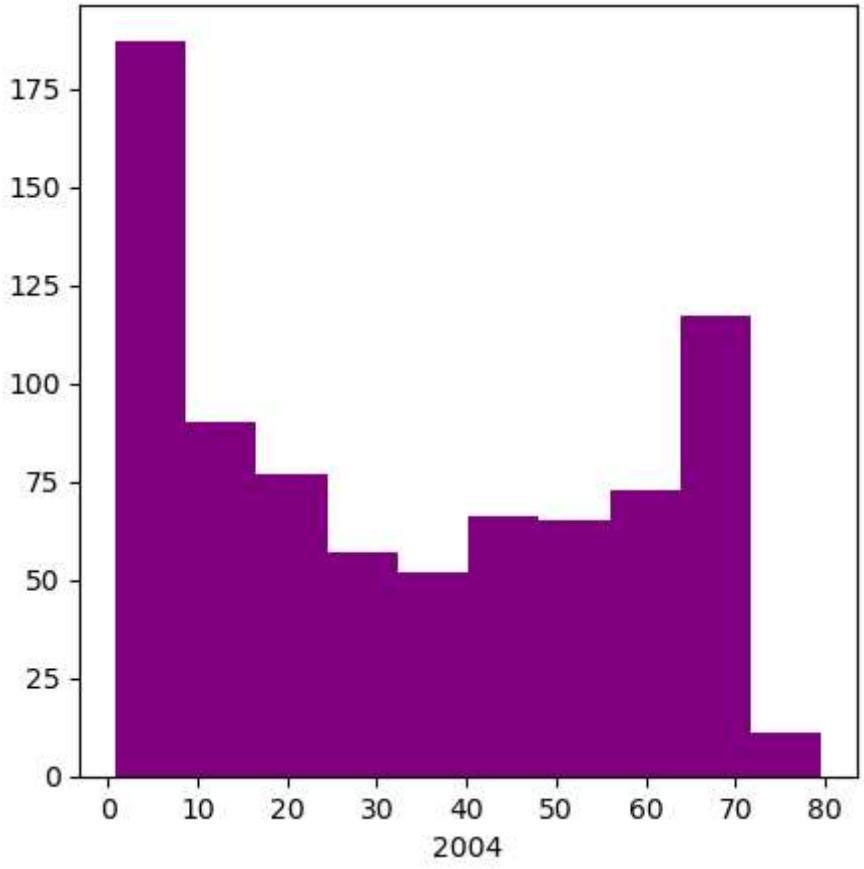

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
1999

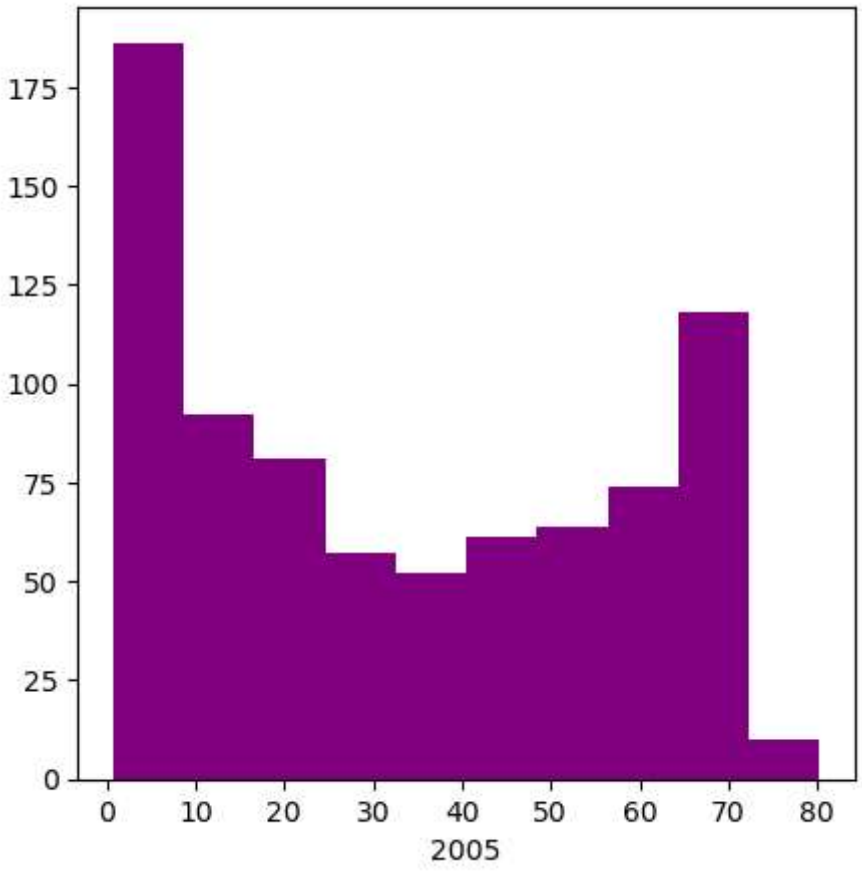

200
175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
2000

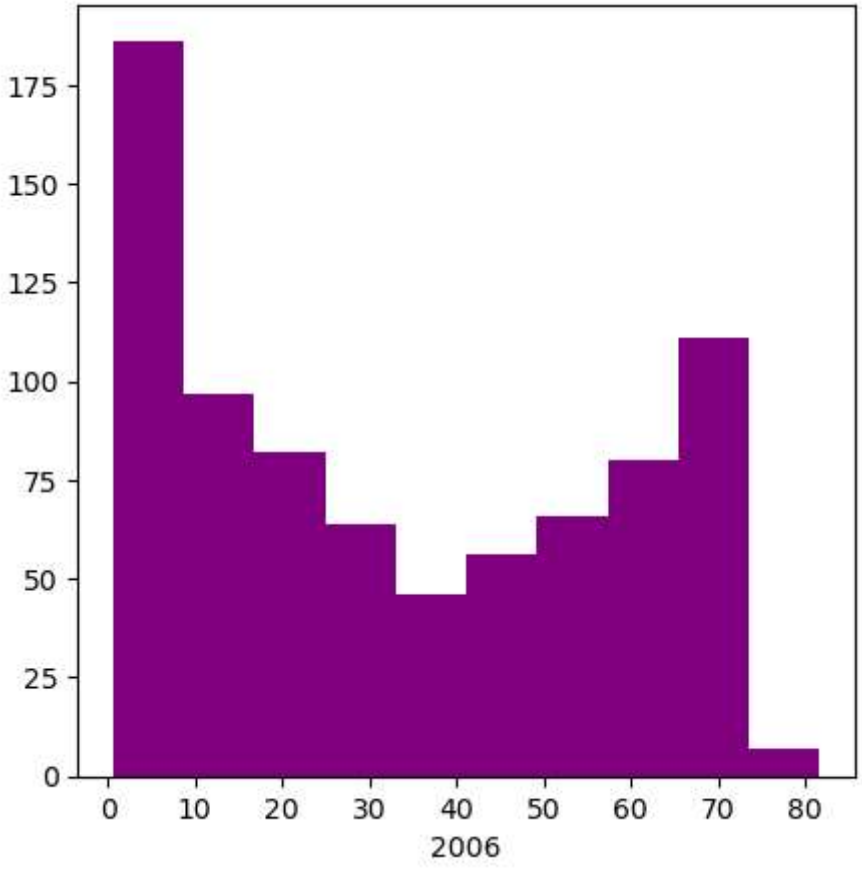

175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
2001

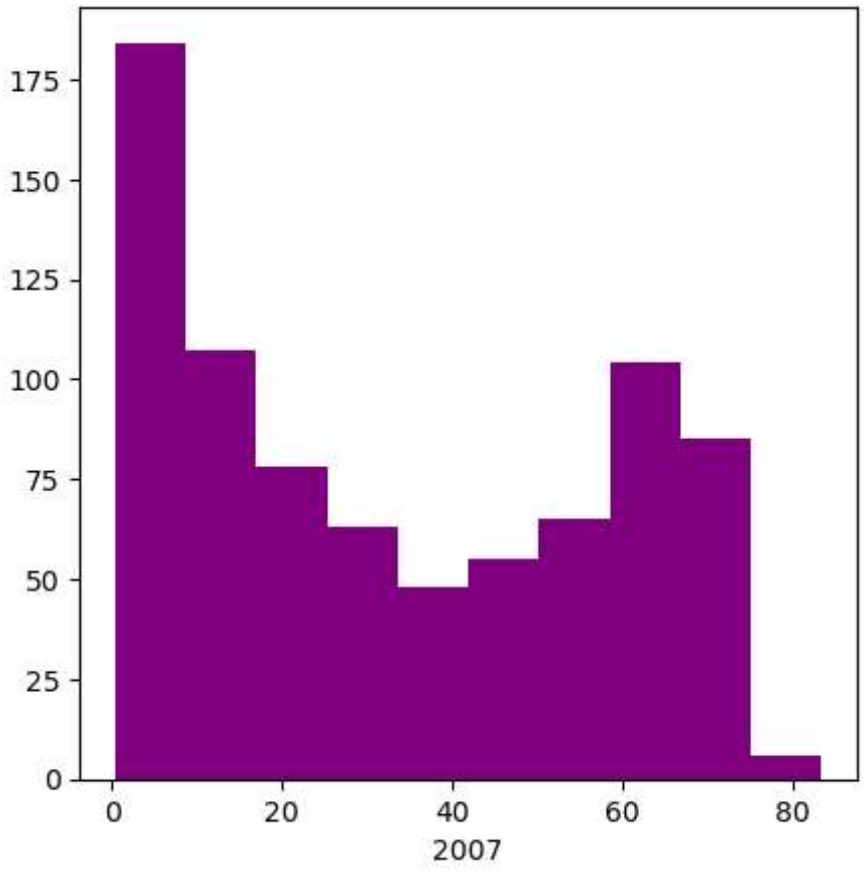

175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
2002

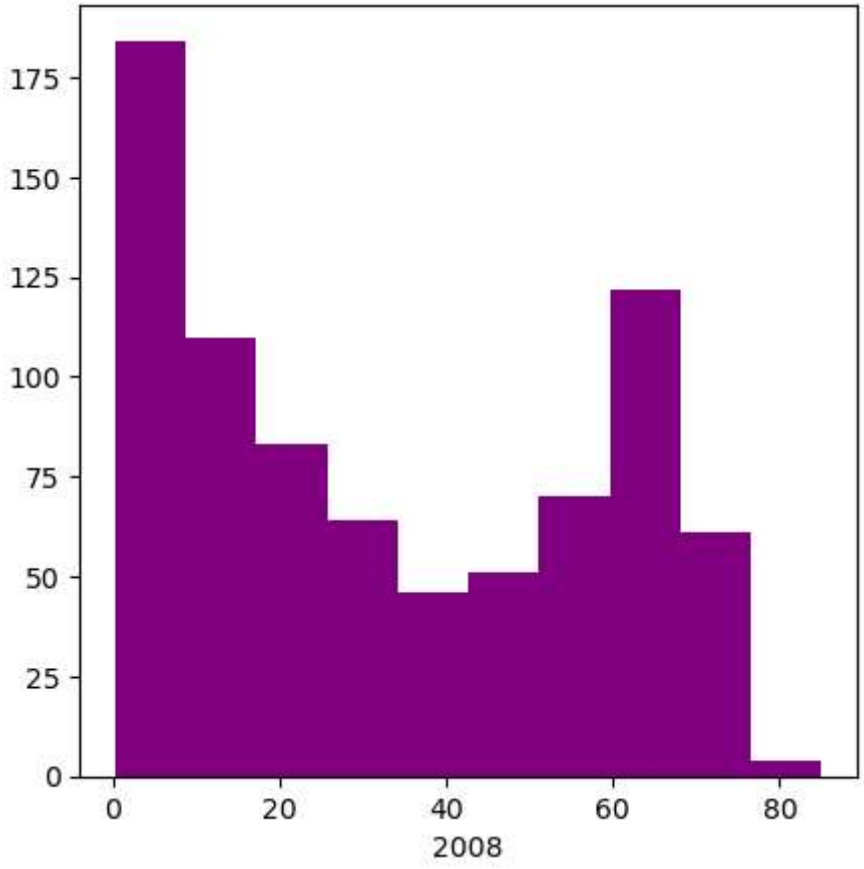

175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
2003

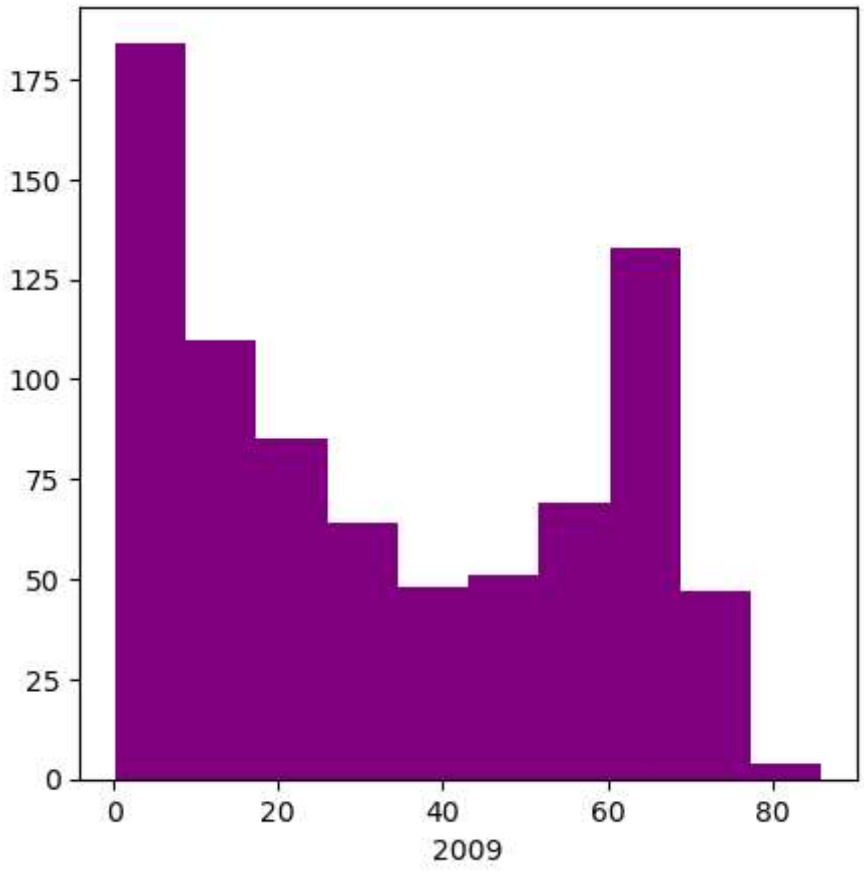

175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
2004

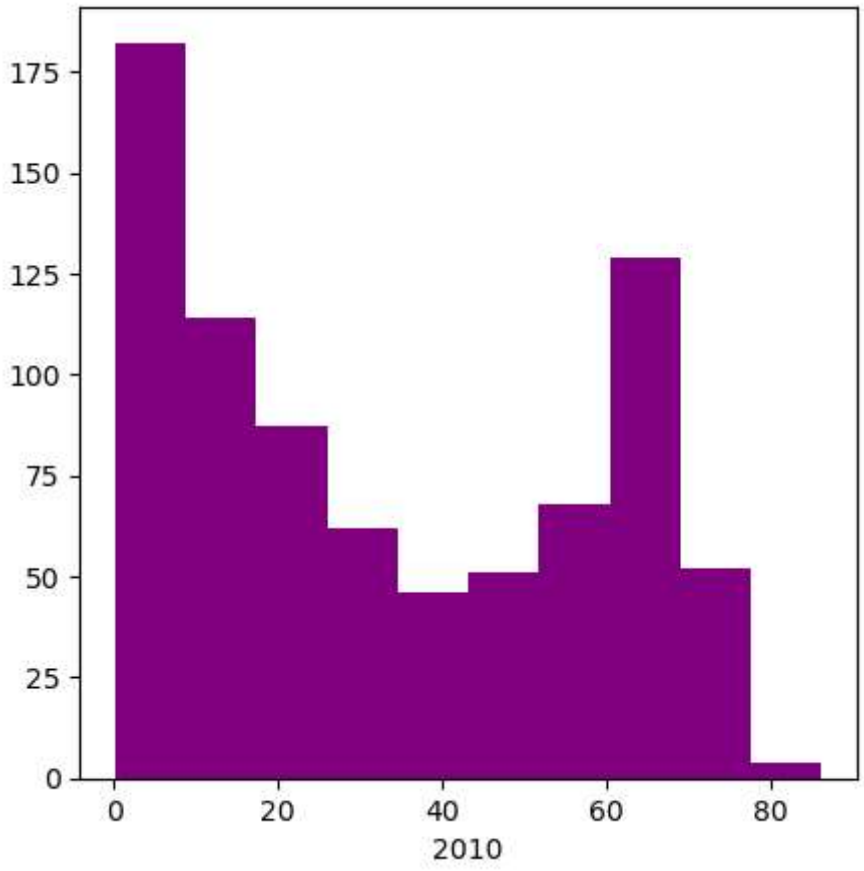

175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
2005

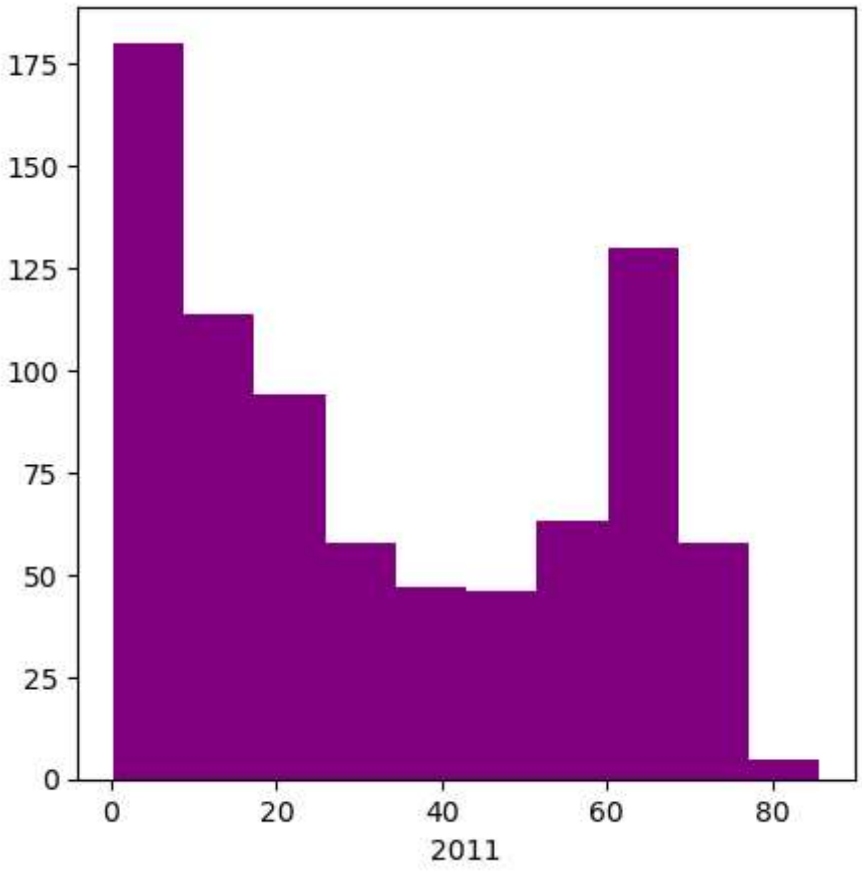

175
150
125
100
75
50
25
0
0
10
20
30
40
50
60
70
80
2006

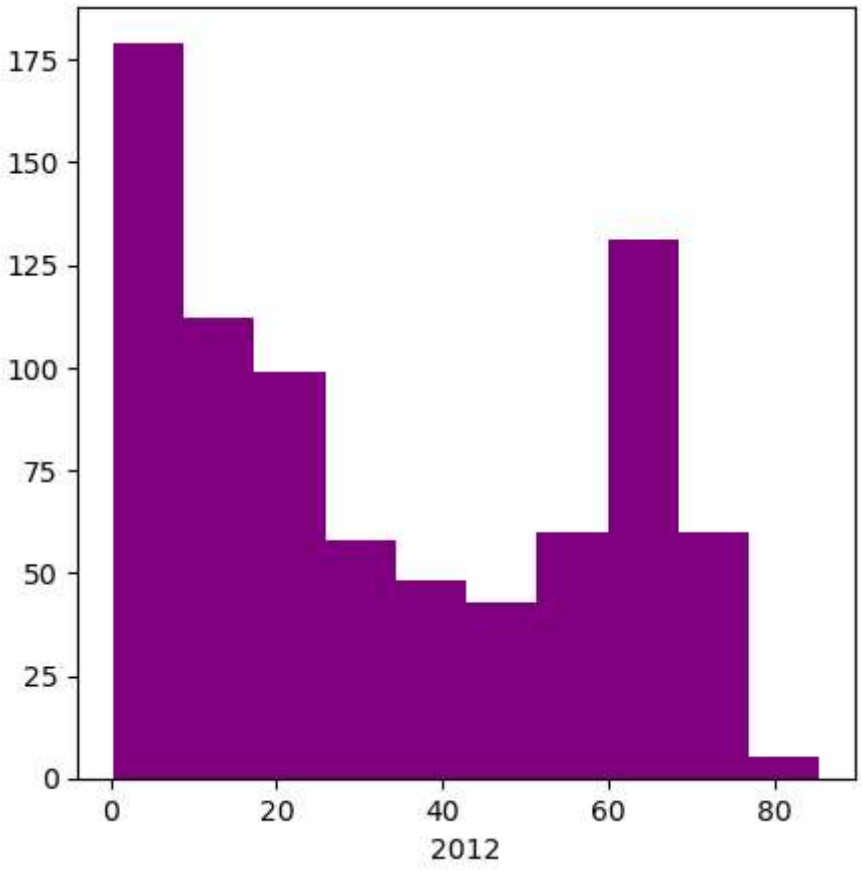

175
150
125
100
75
50
25
0
0
20
40
60
80
2007

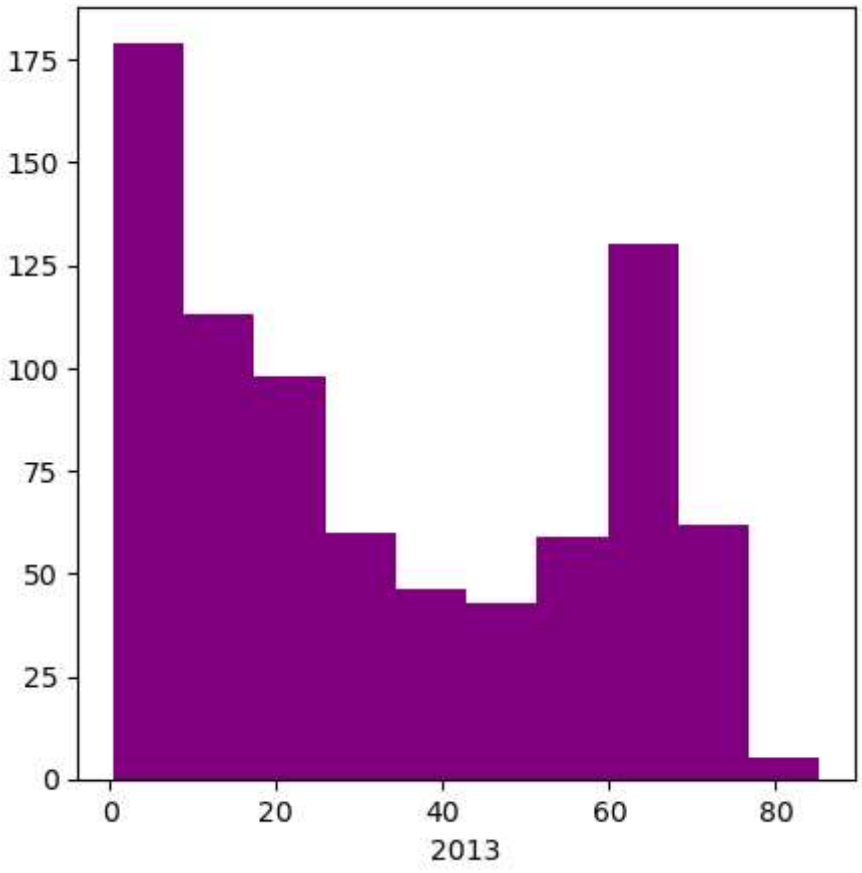

175
150
125
100
75
50
25
0
0
20
40
60
80
2008

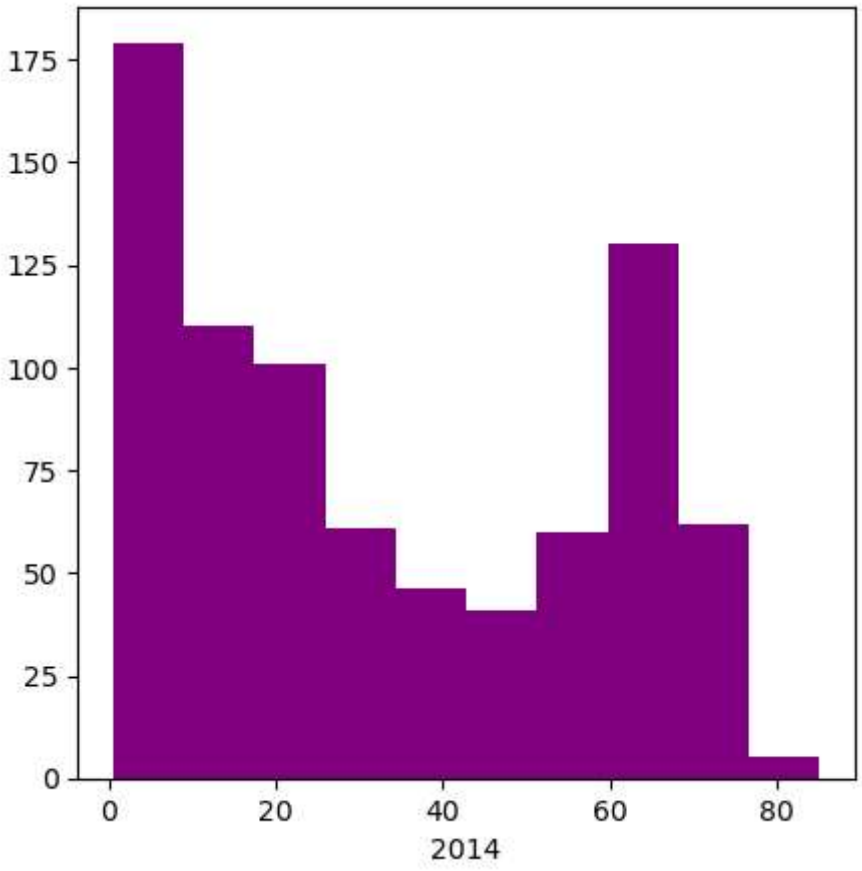

175
150
125
100
75
50
25
0
0
20
40
60
80
2009

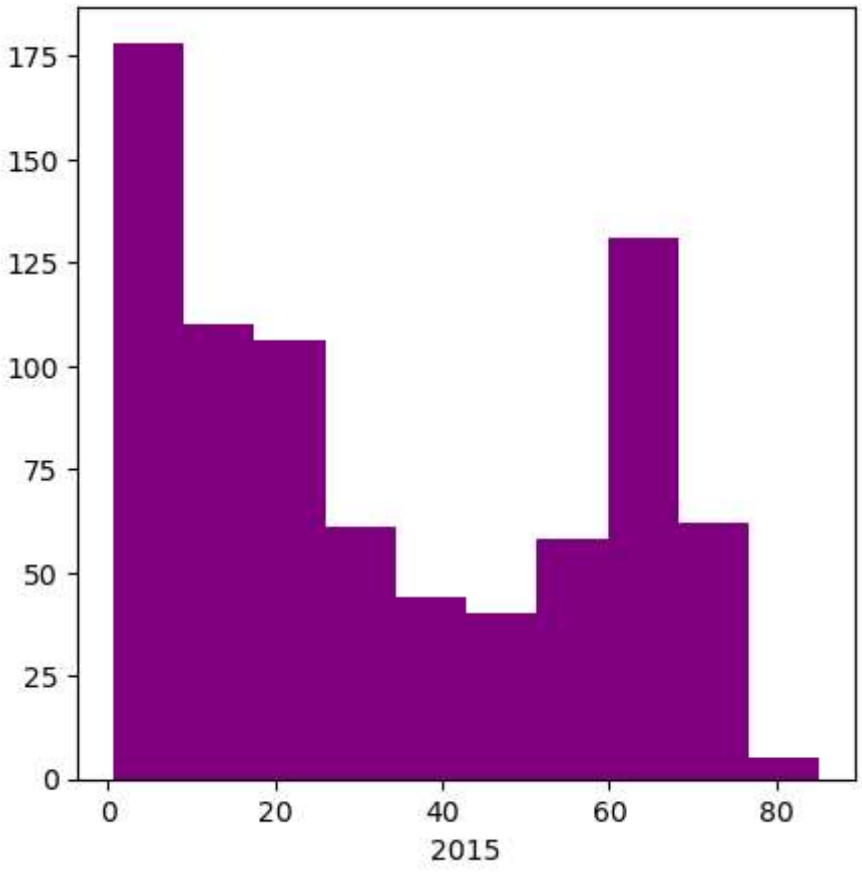

175
150
125
100
75
50
25
0
0
20
40
60
80
2010

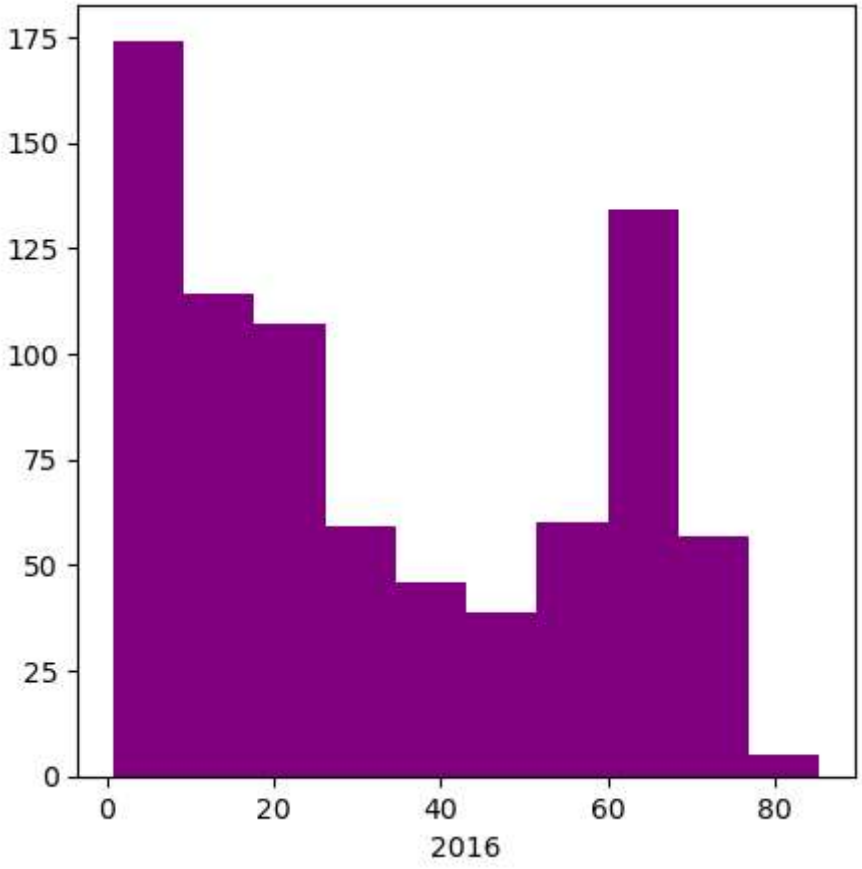

175
150
125
100
75
50
25
0
0
20
40
60
80
2011

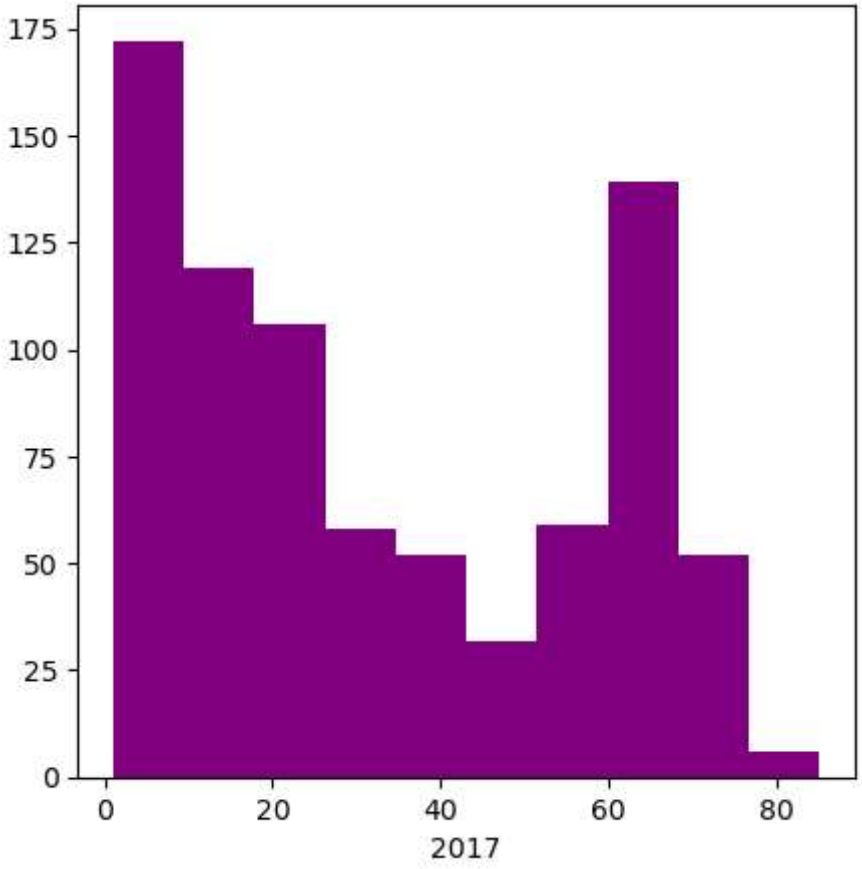

175
150
125
100
75
50
25
0
0
20
40
60
80
2012

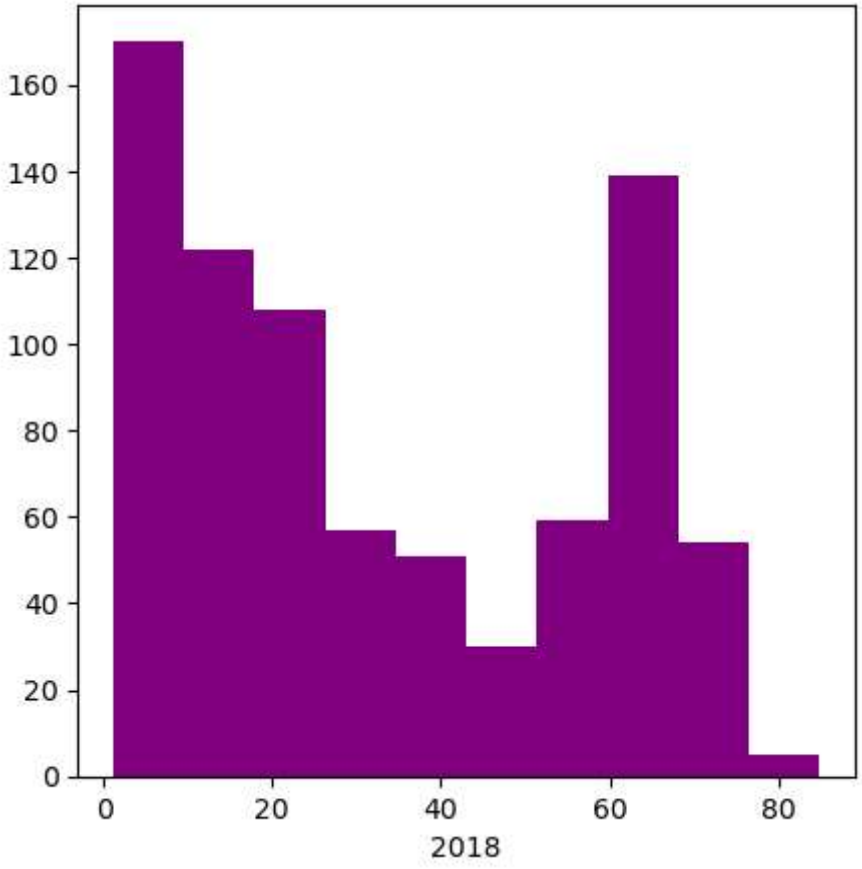

175
150
125
100
75
50
25
0
0
20
40
60
80
2013

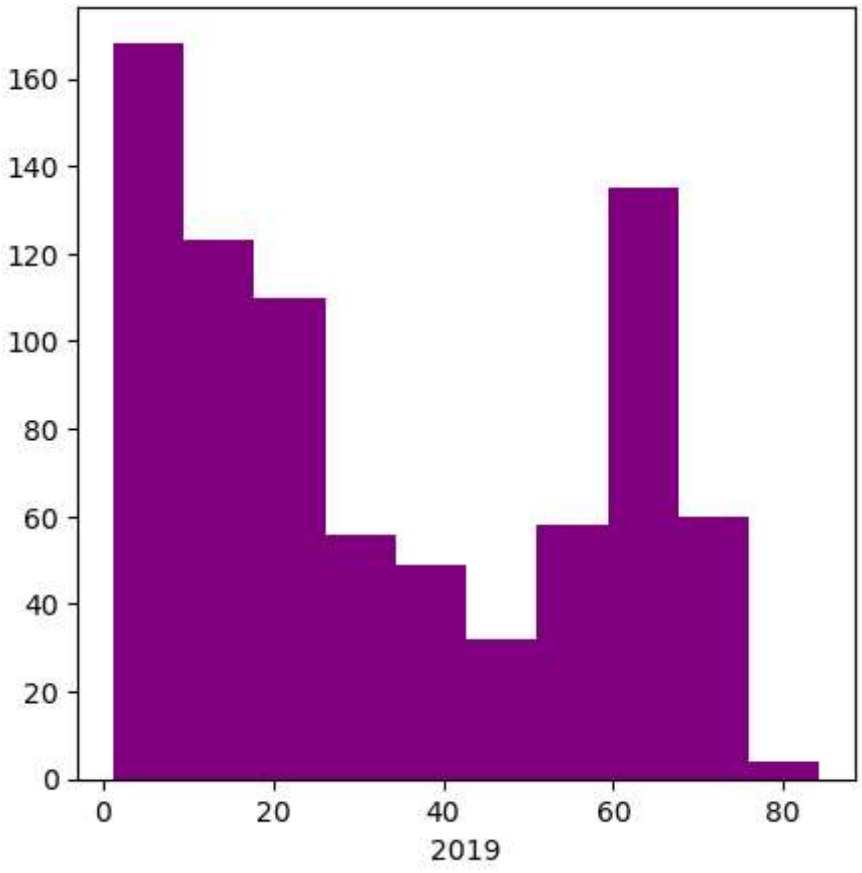

175
150
125
100
75
50
25
0
0
20
40
60
80
2014

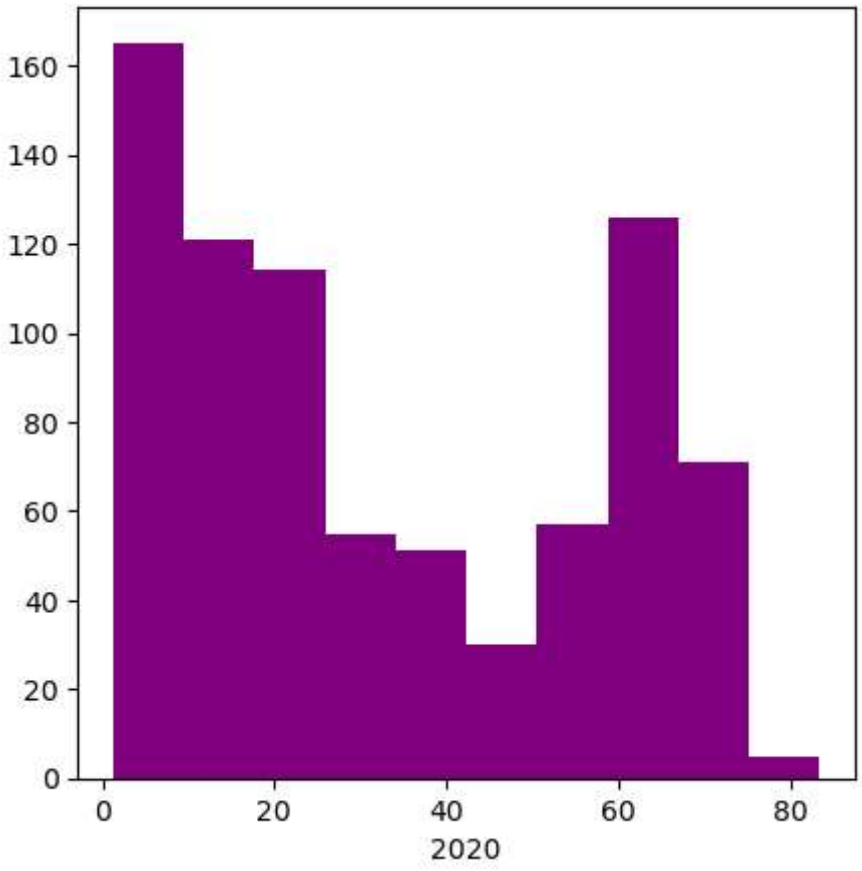

175
150
125
100
75
50
25
0
0
20
40
60
80
2015

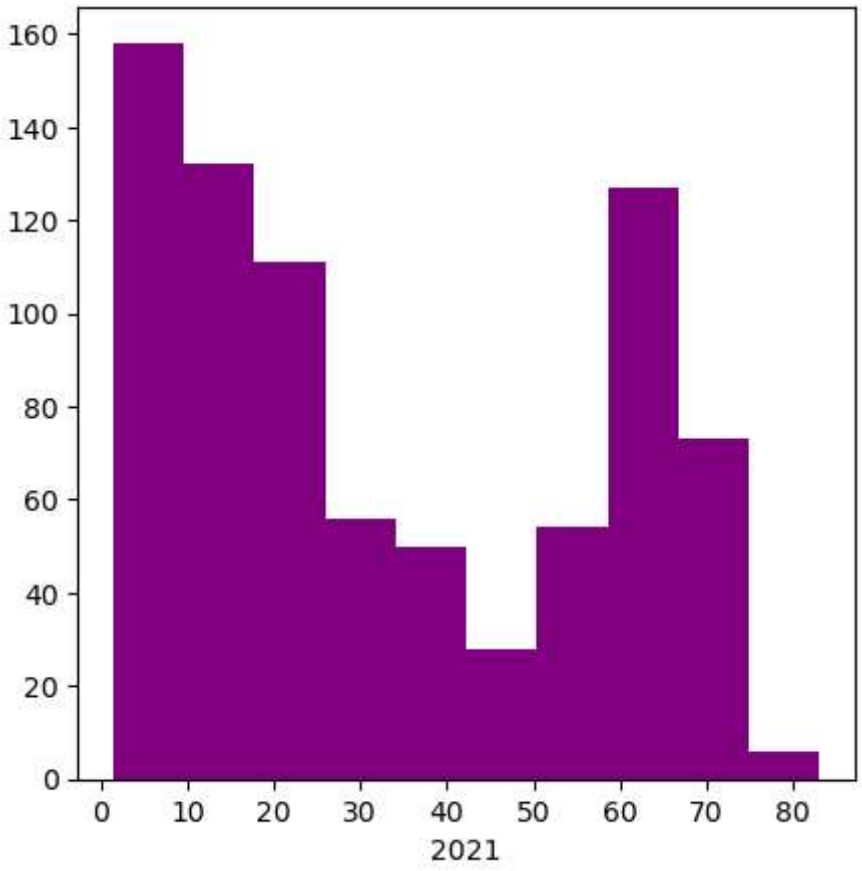

175
150
125
100
75
50
25
0
0
20
40
60
80
2016

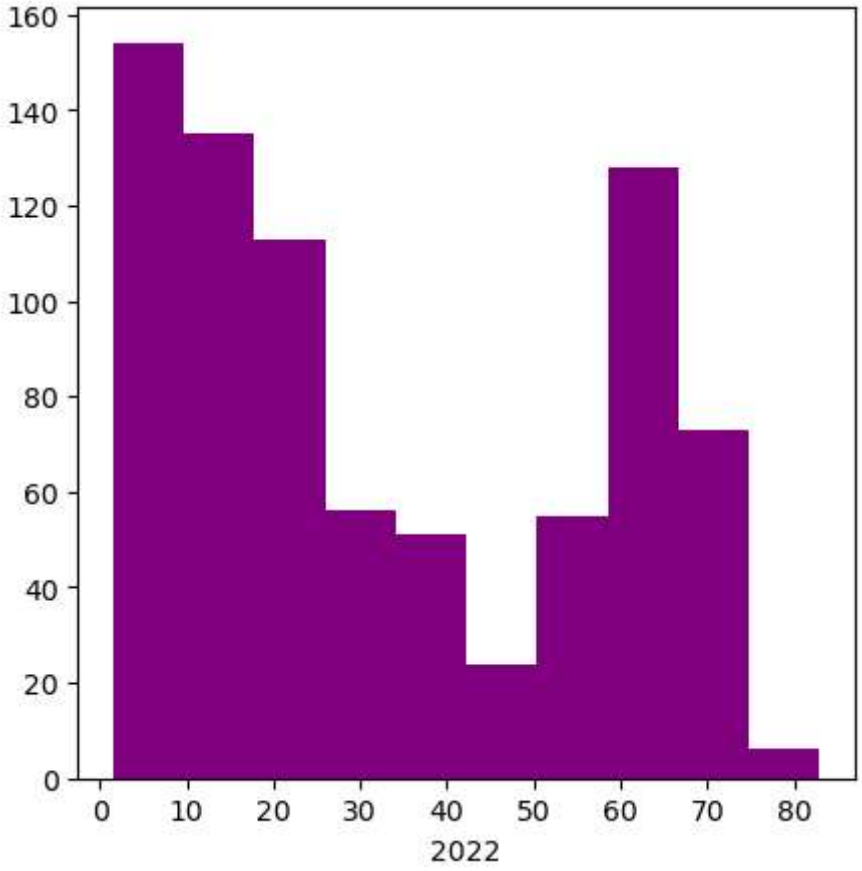

175
150
125
100
75
50
25
0
0
20
40
60
80
2017


160
140
120
100
80
60
40
20
0
0
20
40
60
80
2018


160
140
120
100
80
60
40
20
0
0
20
40
60
80
2019

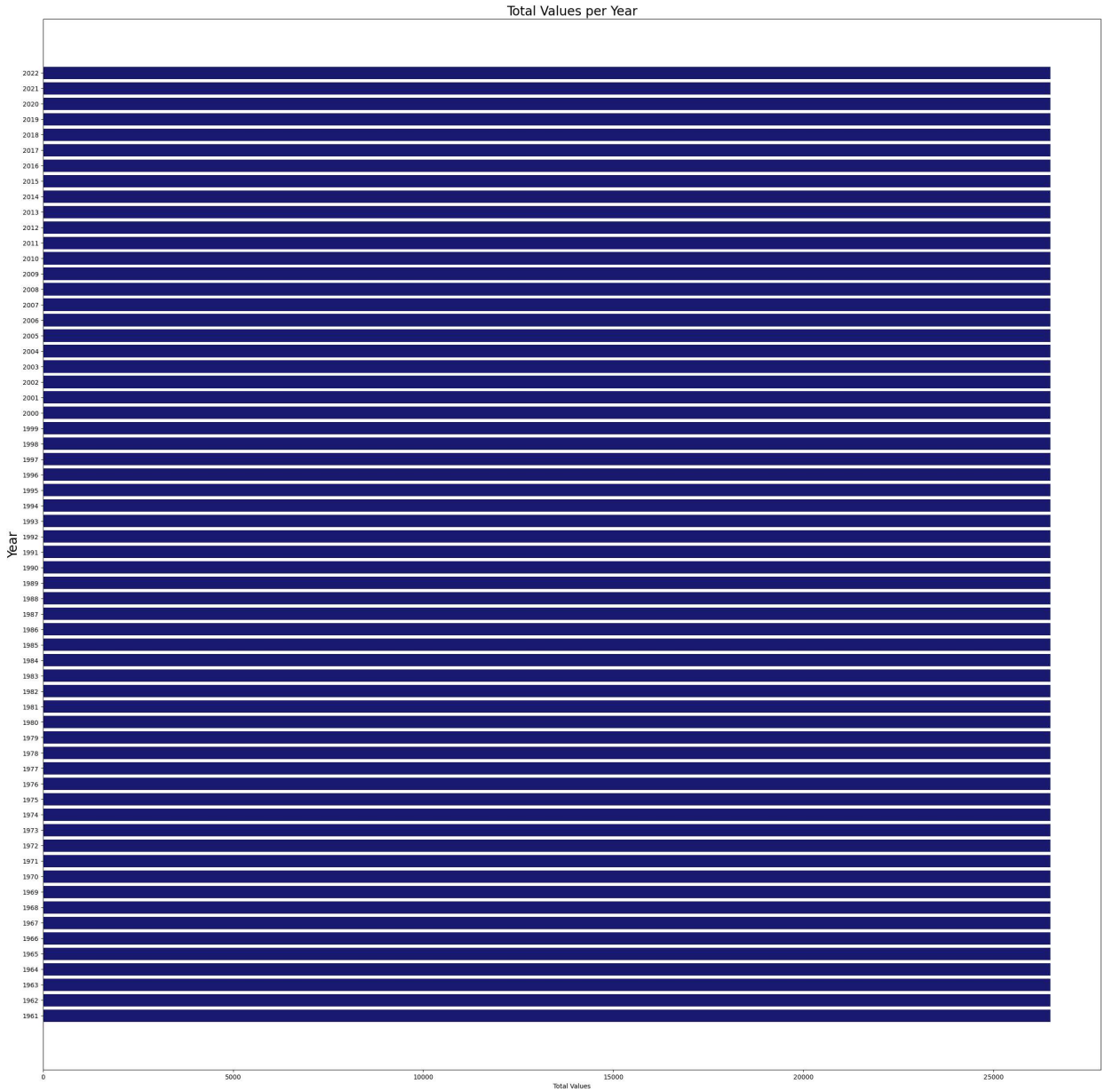

160
140
120
100
80
60
40
20
0
0
20
40
60
80
2020





In [25]:
```
years = selected_columns_data.columns[1:]

total_values = selected_columns_data[years].sum()
```

```
plt.figure(figsize=(30, 30))
plt.barh(years, total_values,color='#191970')
plt.xlabel('Total Values')
plt.ylabel('Year', size=20)
plt.title('Total Values per Year', size=20)
plt.show()
```



In [27]:
```
country_by_1960 =selected_columns_data.sort_values(by='1960').head(20)
country_by_1960
```

Out[27]:

	1960	1961	1962	1963	1964	1965	1966	1967	1968	196
189	1.094746	1.087430	1.069956	1.045335	1.015901	0.984709	0.958864	0.940156	0.923798	0.90881
173	1.122767	1.180629	1.236895	1.290489	1.340828	1.387669	1.431264	1.471433	1.508594	1.54392
93	1.284983	1.334732	1.380408	1.431883	1.484346	1.533861	1.580381	1.623742	1.653821	1.66870
208	1.627747	1.720915	1.840270	1.984493	2.144395	2.315480	2.502404	2.704916	2.926168	3.15429
91	1.975159	2.041274	2.109893	2.165894	2.213234	2.249669	2.314432	2.388262	2.446927	2.49623
247	2.121632	2.140215	2.165829	2.196639	2.231196	2.268459	2.307149	2.347236	2.388471	2.43036
127	2.221731	2.175324	2.115997	2.038911	1.965979	1.896116	1.857061	1.842195	1.822301	1.79841
166	2.238401	2.304787	2.367944	2.426669	2.480462	2.528838	2.571787	2.609875	2.643855	2.67511
254	2.278741	2.304910	2.331621	2.357689	2.384527	2.413394	2.445162	2.480030	2.518362	2.55838
8	2.304274	2.187482	2.081241	1.983676	1.888019	1.799806	1.716354	1.634505	1.569893	1.54761
22	2.338332	2.265071	2.194573	2.122680	2.043288	2.039510	2.122662	2.221642	2.331820	2.45459
56	2.340586	2.369764	2.397930	2.426026	2.452280	2.476752	2.498825	2.517166	2.535483	2.55668
41	2.371530	2.384485	2.395763	2.405175	2.412858	2.419456	2.425332	2.430607	2.435345	2.43992
0	2.373279	2.461880	2.561637	2.664637	2.767393	2.881859	3.016503	3.187924	3.405050	3.64823
32	2.377009	2.370189	2.365718	2.364027	2.361949	2.359907	2.360291	2.362174	2.365388	2.37127
246	2.379567	2.392069	2.404195	2.416018	2.427177	2.438334	2.449426	2.460282	2.472373	2.48872
92	2.407114	2.426318	2.448806	2.469248	2.486244	2.503077	2.517193	2.524840	2.527063	2.52732
106	2.445810	2.515145	2.582516	2.647445	2.711815	2.762680	2.815931	2.886494	2.958772	3.03032
175	2.474403	2.449325	2.428881	2.412156	2.398957	2.389717	2.384926	2.384113	2.386942	2.39282
237	2.526928	2.483415	2.462323	2.459801	2.470769	2.491246	2.523068	2.566565	2.612558	2.64974

20 rows × 63 columns

In [28]:

```
# Transpose the DataFrame
country_by_1960_t = df_combined.set_index('Country Name').T

# Plotting the data for each country
for country_name, data_values in country_by_1960_t.iterrows():
    fig = plt.figure(figsize=(10, 5))

    # Filter out non-numeric index values
    numeric_index = pd.to_numeric(data_values.index, errors='coerce')
    data_values = data_values[~numeric_index.isna()]

    # Convert the index (years) to numeric
    data_values.index = numeric_index.dropna()

    # Check if data_values is not empty before plotting
    if not data_values.empty:
        sns.barplot(x=data_values.index, y=data_values.values)
        plt.xlabel('Year')
```

```
        plt.ylabel('Data Values')
        plt.title(f"{country_name} - Data Values from 1960 to 2022")
        plt.xticks(rotation=90)
        plt.show()
    else:
        print(f"No data available for {country_name}")
```

```
No data available for 1960
No data available for 1961
No data available for 1962
No data available for 1963
No data available for 1964
No data available for 1965
No data available for 1966
No data available for 1967
No data available for 1968
No data available for 1969
No data available for 1970
No data available for 1971
No data available for 1972
No data available for 1973
No data available for 1974
No data available for 1975
No data available for 1976
No data available for 1977
No data available for 1978
No data available for 1979
No data available for 1980
No data available for 1981
No data available for 1982
No data available for 1983
No data available for 1984
No data available for 1985
No data available for 1986
No data available for 1987
No data available for 1988
No data available for 1989
No data available for 1990
No data available for 1991
No data available for 1992
No data available for 1993
No data available for 1994
No data available for 1995
No data available for 1996
No data available for 1997
No data available for 1998
No data available for 1999
No data available for 2000
No data available for 2001
No data available for 2002
No data available for 2003
No data available for 2004
No data available for 2005
No data available for 2006
No data available for 2007
No data available for 2008
No data available for 2009
No data available for 2010
No data available for 2011
No data available for 2012
No data available for 2013
No data available for 2014
No data available for 2015
No data available for 2016
No data available for 2017
No data available for 2018
No data available for 2019
```

```
No data available for 2020
No data available for 2021
No data available for 2022
```

```
C:\Users\aryas\AppData\Local\Temp\ipykernel_27884\2286694503.py:6: RuntimeWarning: Mo
re than 20 figures have been opened. Figures created through the pyplot interface (`m
atplotlib.pyplot.figure`) are retained until explicitly closed and may consume too mu
ch memory. (To control this warning, see the rcParam `figure.max_open_warning`). Cons
ider using `matplotlib.pyplot.close()`.
  fig = plt.figure(figsize=(10, 5))
```

```
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
```

```
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
```

In [29]:
```
country_by_2022 = df_combined.sort_values(by='2022').head(20)
country_by_2022
```

Out[29]:

	Country Name	1960	1961	1962	1963	1964	1965	1966	1967	
200	Qatar	3.397051	3.289662	3.165826	3.039254	2.912368	2.806030	2.709228	2.619460	2.54
247	Uganda	2.121632	2.140215	2.165829	2.196639	2.231196	2.268459	2.307149	2.347236	2.38
264	Zambia	2.695108	2.693254	2.698690	2.708720	2.718231	2.726800	2.735912	2.744504	2.75
8	United Arab Emirates	2.304274	2.187482	2.081241	1.983676	1.888019	1.799806	1.716354	1.634505	1.56
229	Chad	3.996434	3.980866	3.976728	3.978294	3.978302	3.971645	3.960409	3.946902	3.93
158	Mali	2.586177	2.586063	2.588271	2.596571	2.615815	2.645396	2.683787	2.730824	2.78
2	Afghanistan	2.833029	2.817674	2.799055	2.778968	2.758929	2.739247	2.721527	2.706206	2.69
41	Cote d'Ivoire	2.371530	2.384485	2.395763	2.405175	2.412858	2.419456	2.425332	2.430607	2.43
173	Niger	1.122767	1.180629	1.236895	1.290489	1.340828	1.387669	1.431264	1.471433	1.50
86	Gambia, The	2.738531	2.749503	2.759344	2.768248	2.776813	2.785747	2.794786	2.803982	2.81
179	Nauru	3.830205	3.556023	3.282497	3.020392	2.817030	2.593039	2.416743	2.256930	2.07
16	Burundi	3.177770	3.184761	3.196874	3.216324	3.230361	3.234851	3.239344	3.246579	3.25
34	Central African Republic	4.295843	4.257280	4.230038	4.209263	4.187160	4.160350	4.129781	4.096099	4.06
19	Burkina Faso	2.643459	2.692531	2.740016	2.787091	2.834256	2.882320	2.931898	2.983269	3.03
165	Mozambique	2.920088	2.903996	2.881284	2.856304	2.829830	2.802946	2.779160	2.759887	2.74
213	Somalia	3.169923	3.204763	3.234902	3.261362	3.284280	3.303593	3.320576	3.335596	3.34
4	Angola	3.080044	3.094297	3.097629	3.097381	3.093087	3.084555	3.071015	3.051170	3.02
168	Malawi	2.953648	2.937050	2.917781	2.897593	2.881594	2.870390	2.864812	2.866127	2.86
262	Yemen, Rep.	3.786027	3.750778	3.715664	3.676558	3.637130	3.599972	3.562423	3.524506	3.48
42	Cameroon	3.824815	3.836719	3.850304	3.865584	3.878139	3.888413	3.896722	3.902880	3.90

20 rows × 64 columns

In [30]:
```
print(selected_columns_data.columns)
```

```
Index(['1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022'],
      dtype='object')
```

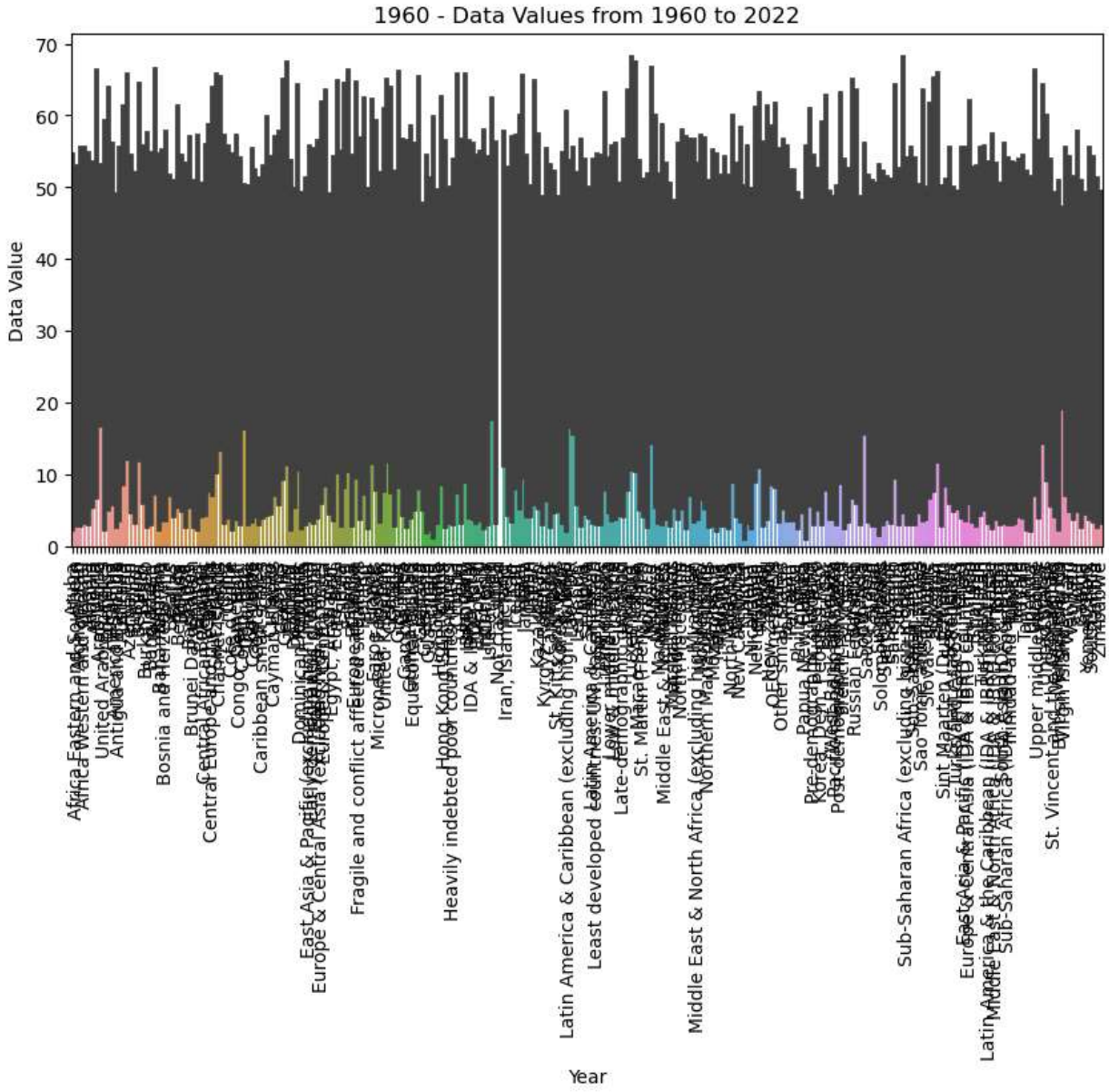
In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Assuming df_combined is your DataFrame containing the data

# Extracting the most recent 5 years
recent_years = df_combined.columns[-5:]

# Sorting the DataFrame by a specific column (e.g., '2022') and selecting the top 20
country_by_2022 = df_combined.sort_values(by='2022', ascending=False).head(20)

# Transposing the DataFrame
country_by_2022_t = country_by_2022.set_index('Country Name').T

# Plotting the data for each country
for country_name, data_values in country_by_2022_t.iterrows():
    fig = plt.figure(figsize=(10, 5))

    # Filter out non-year values from recent_years
    years_to_plot = [year for year in recent_years if str(year).isdigit()]

    # Plot the data using only the valid years present in years_to_plot
    sns.barplot(x=years_to_plot, y=data_values[years_to_plot])

    plt.xlabel('Year')
    plt.ylabel('Data Value')
    plt.title(f"{country_name} - Data Values for the Most Recent 5 Years")
    plt.xticks(rotation=90)
    plt.show()
```

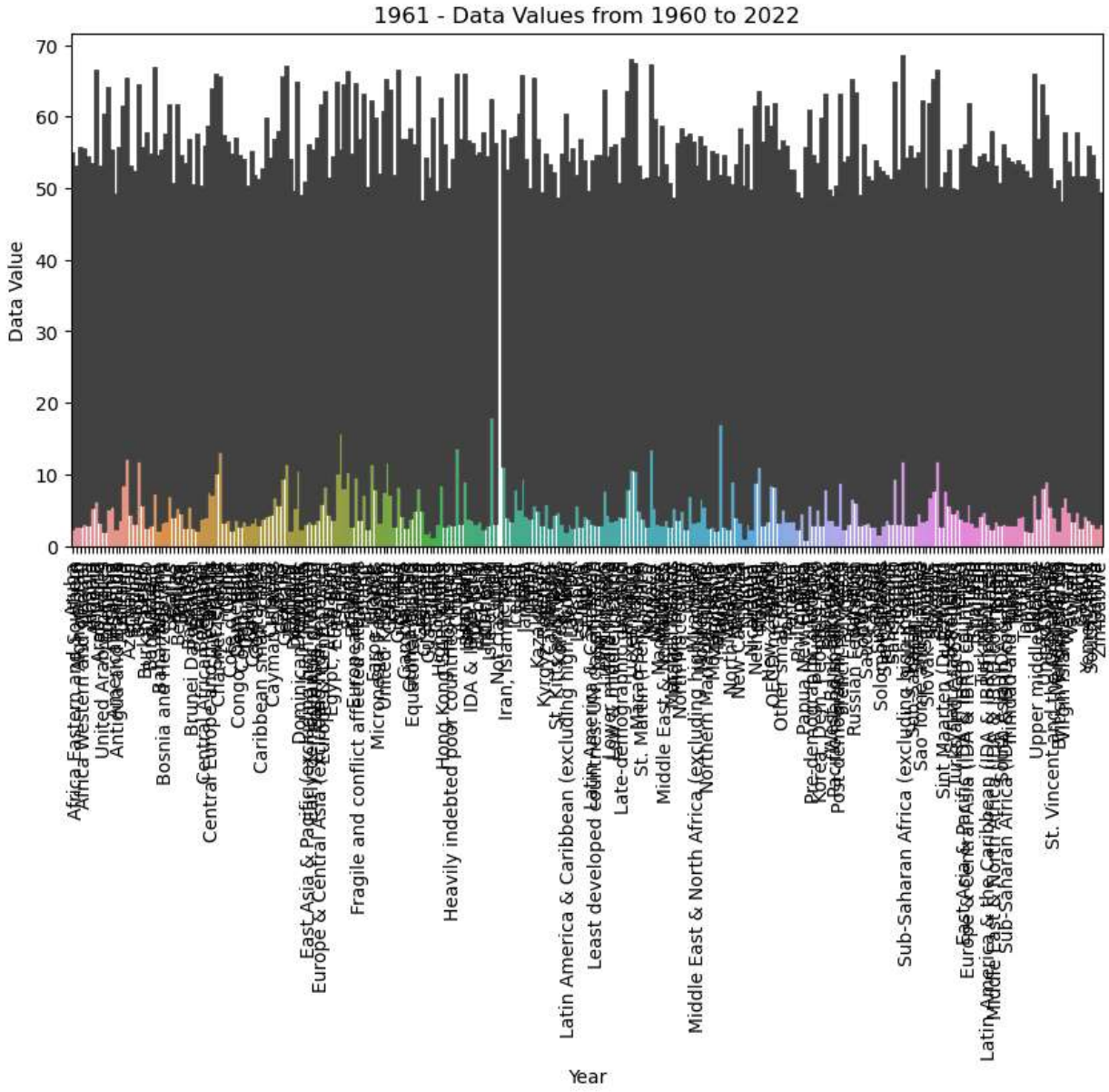
In [ ]:

```
country_by_2022_t = df_combined.set_index('Country Name').T

for country_name, data_values in country_by_2022_t.iterrows():
    fig = plt.figure(figsize=(10, 5))
    sns.barplot(x=data_values.index, y=data_values.values)
    plt.xlabel('Year')
    plt.ylabel('Data Value')
    plt.title(f"{country_name} - Data Values from 1960 to 2022")
    plt.xticks(rotation=90)
    plt.show()
```

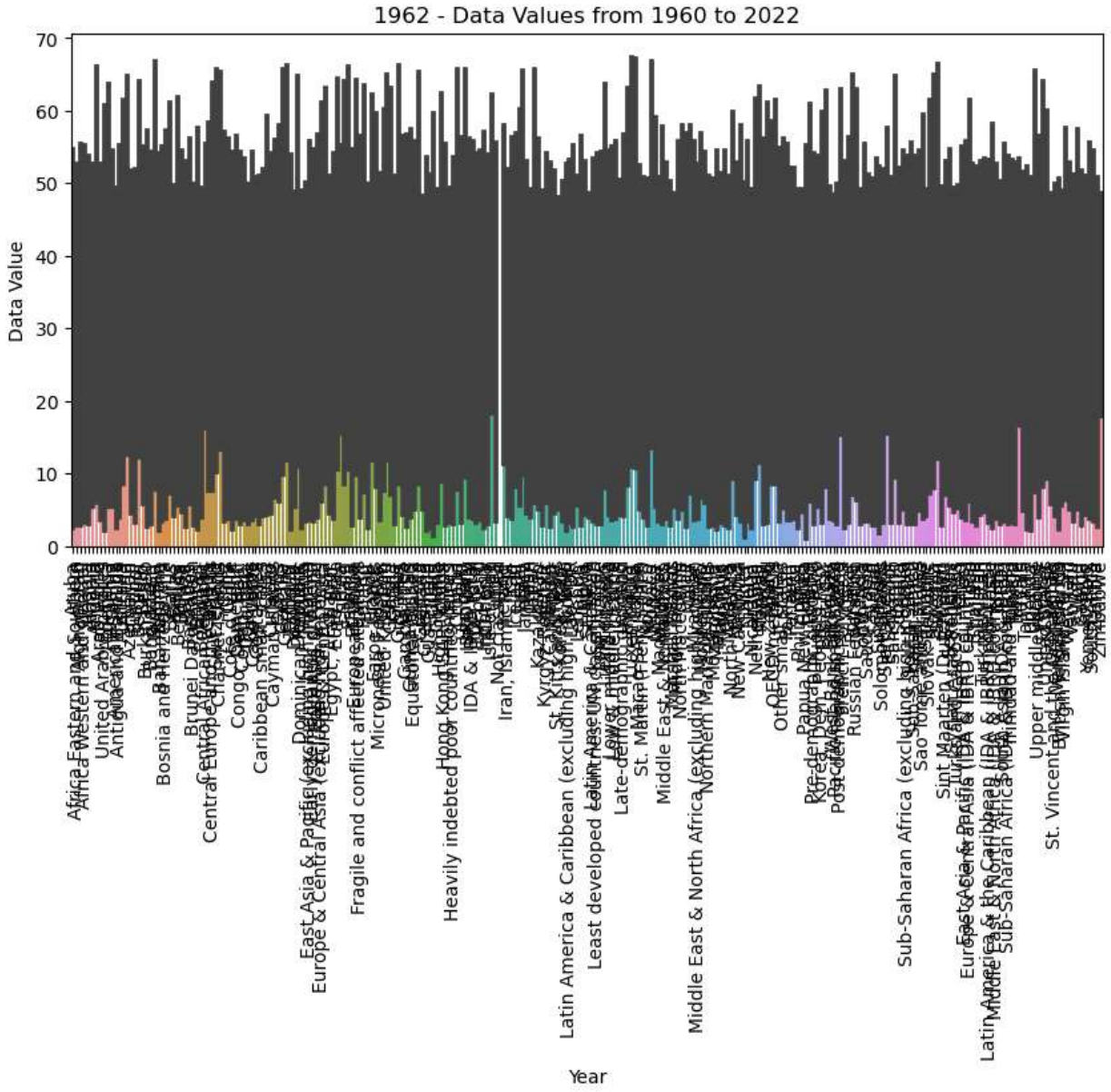
```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



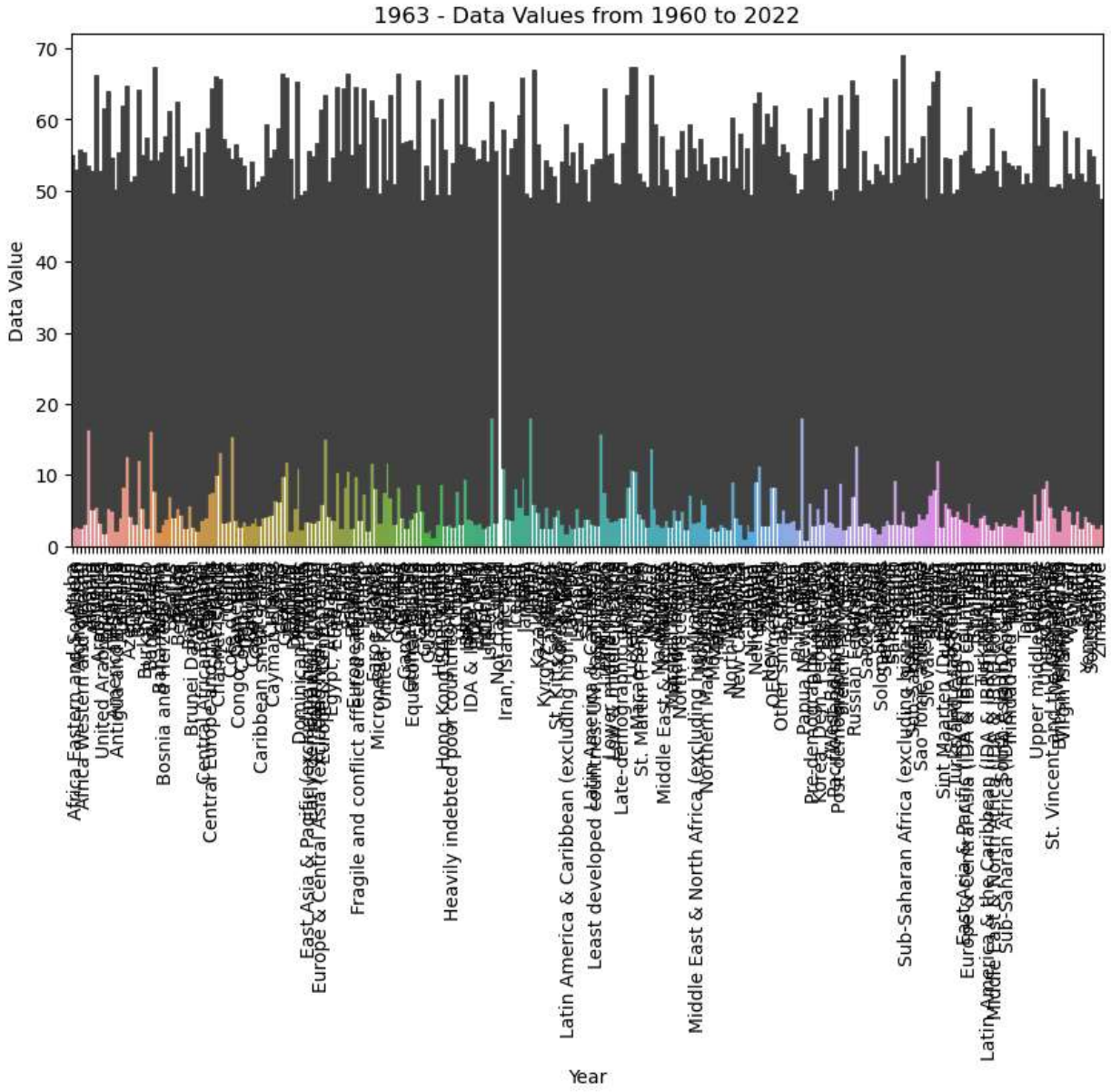

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



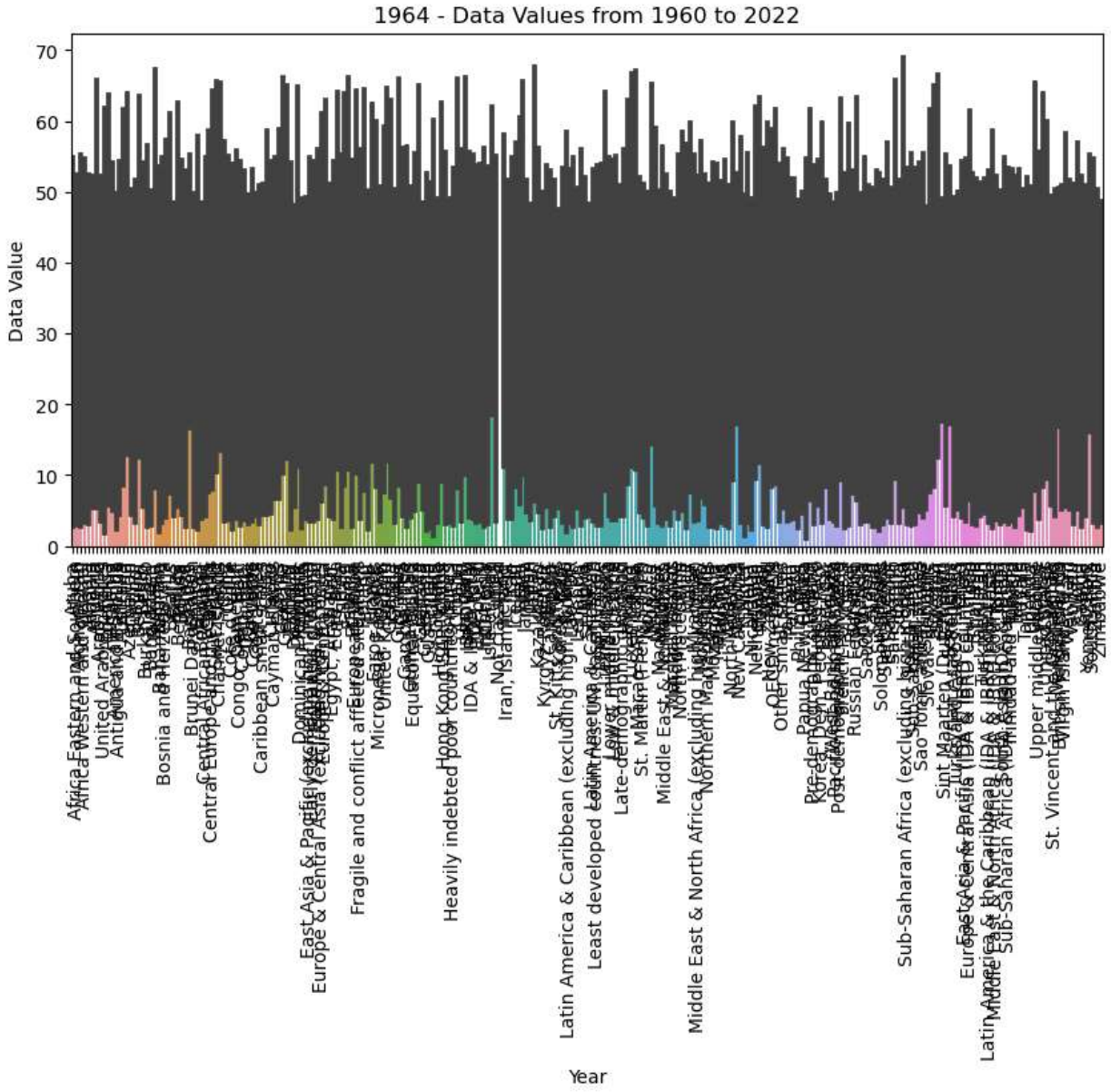

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



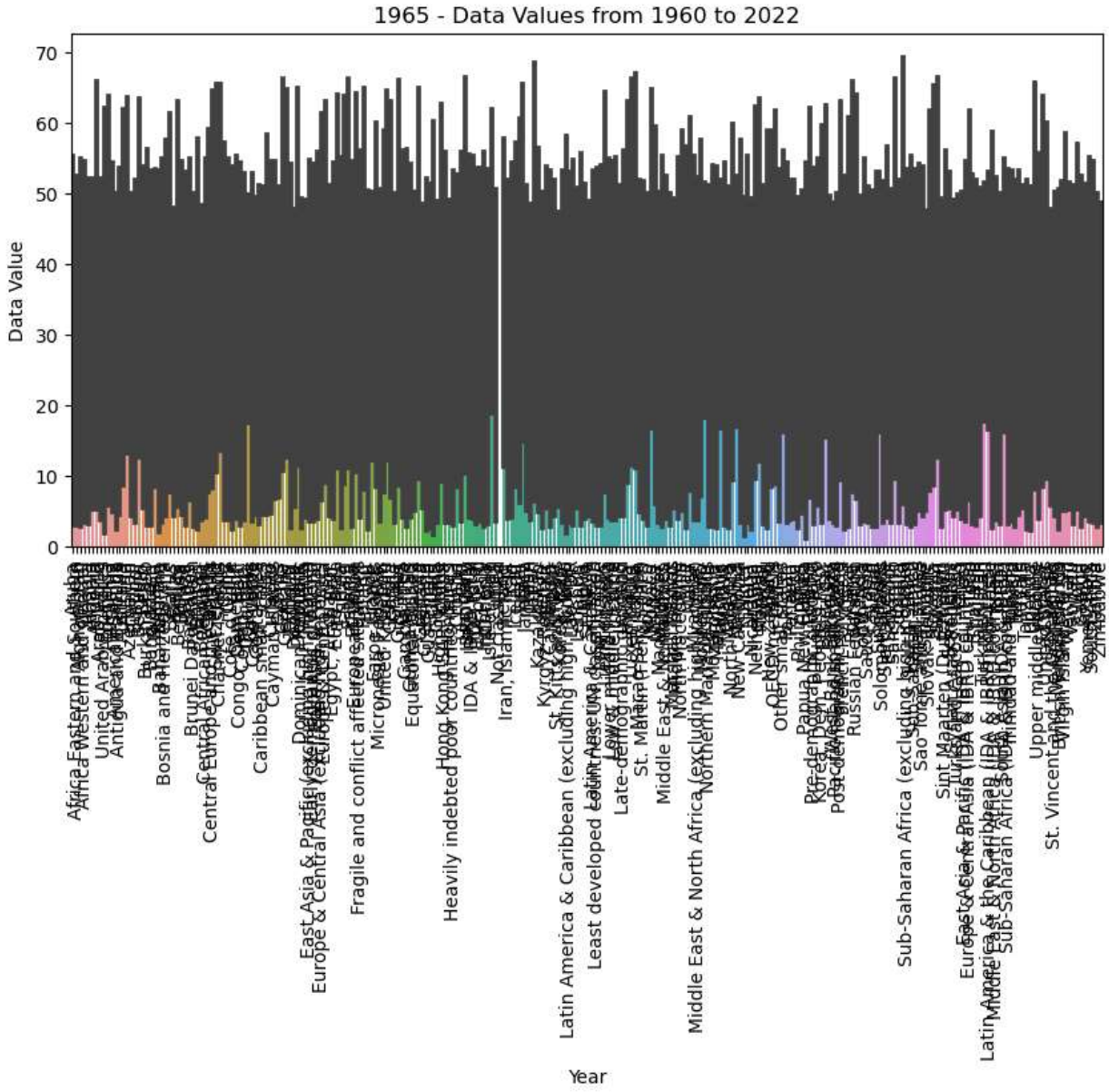

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



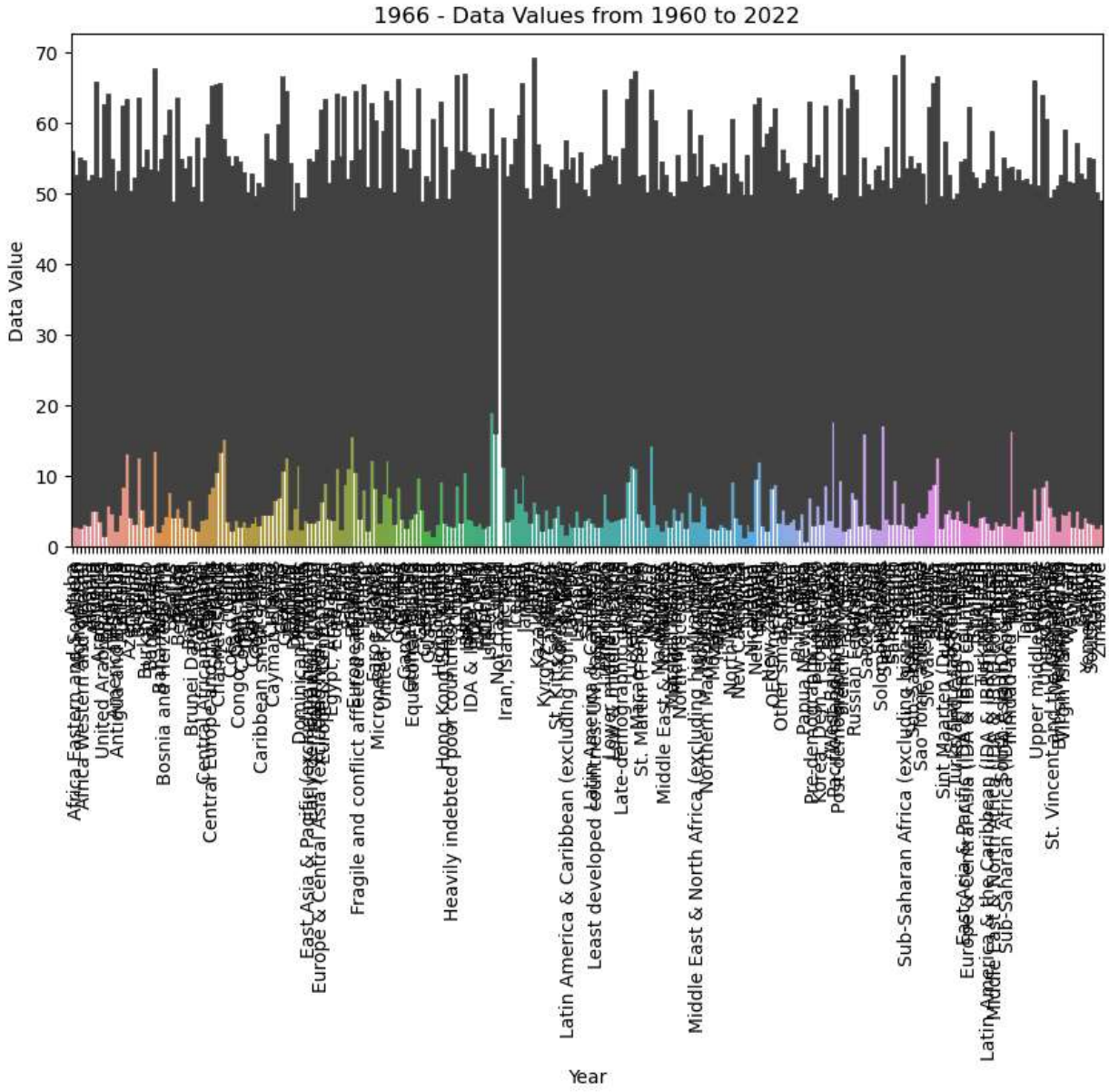

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



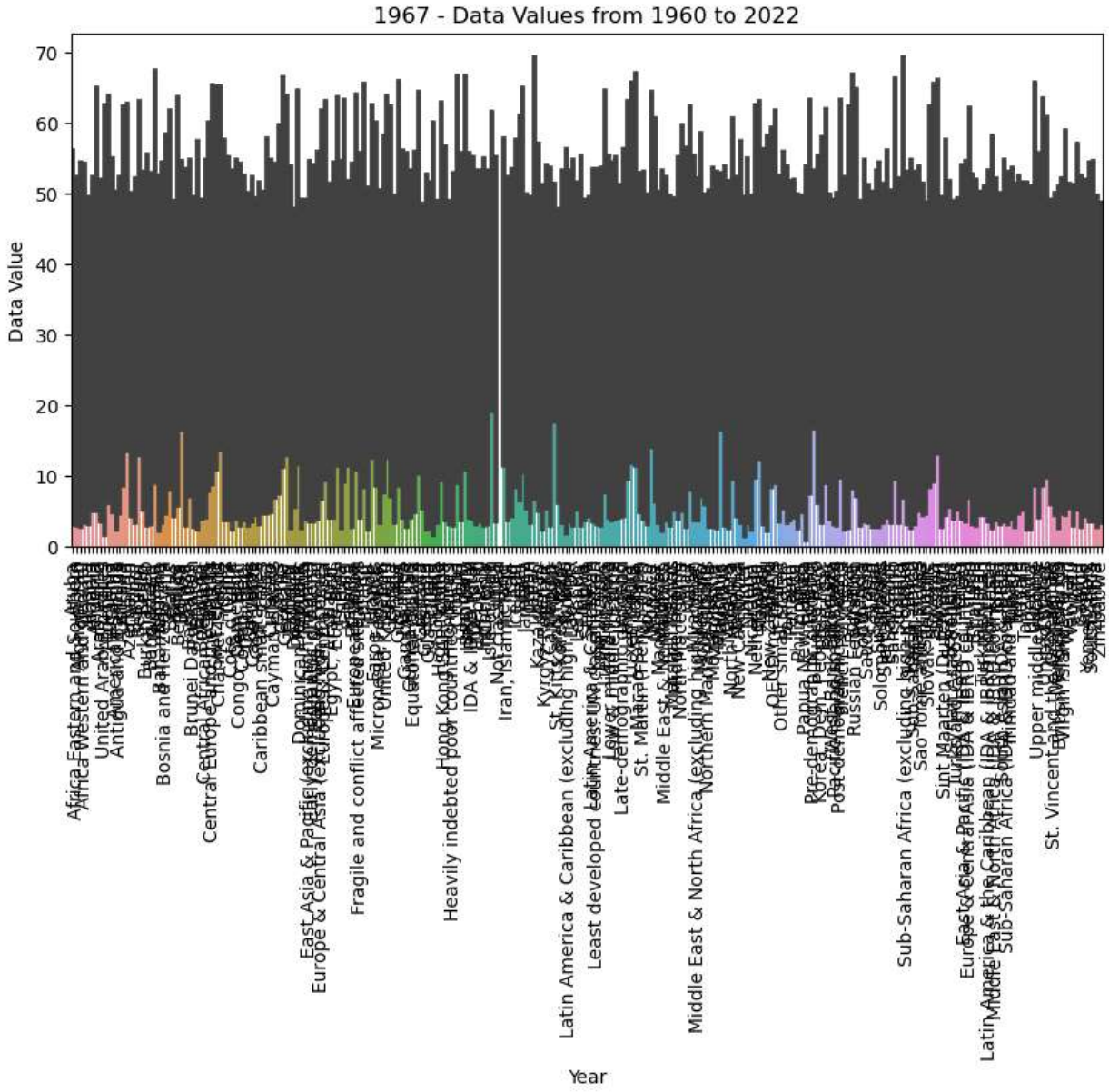

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



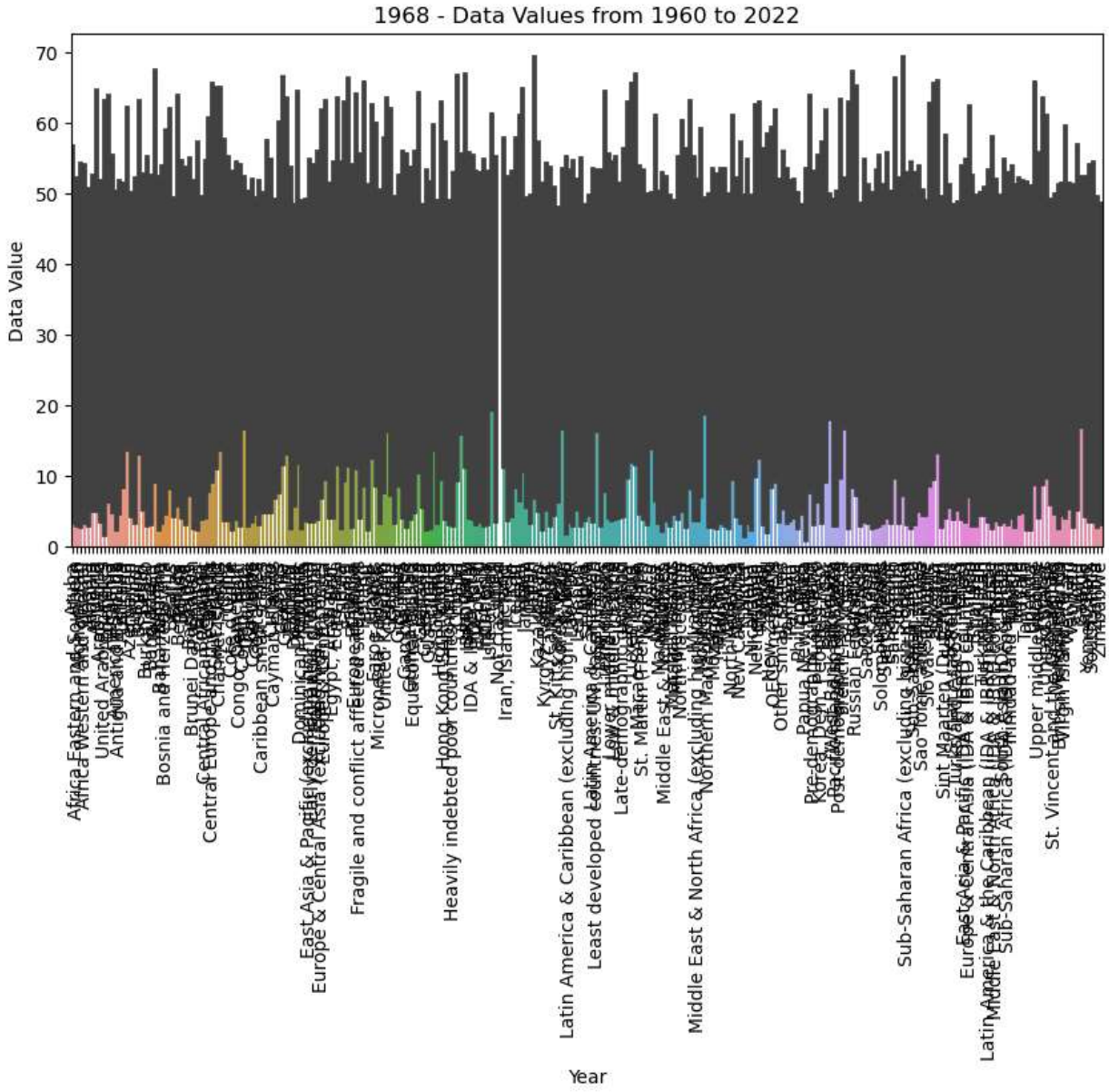

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



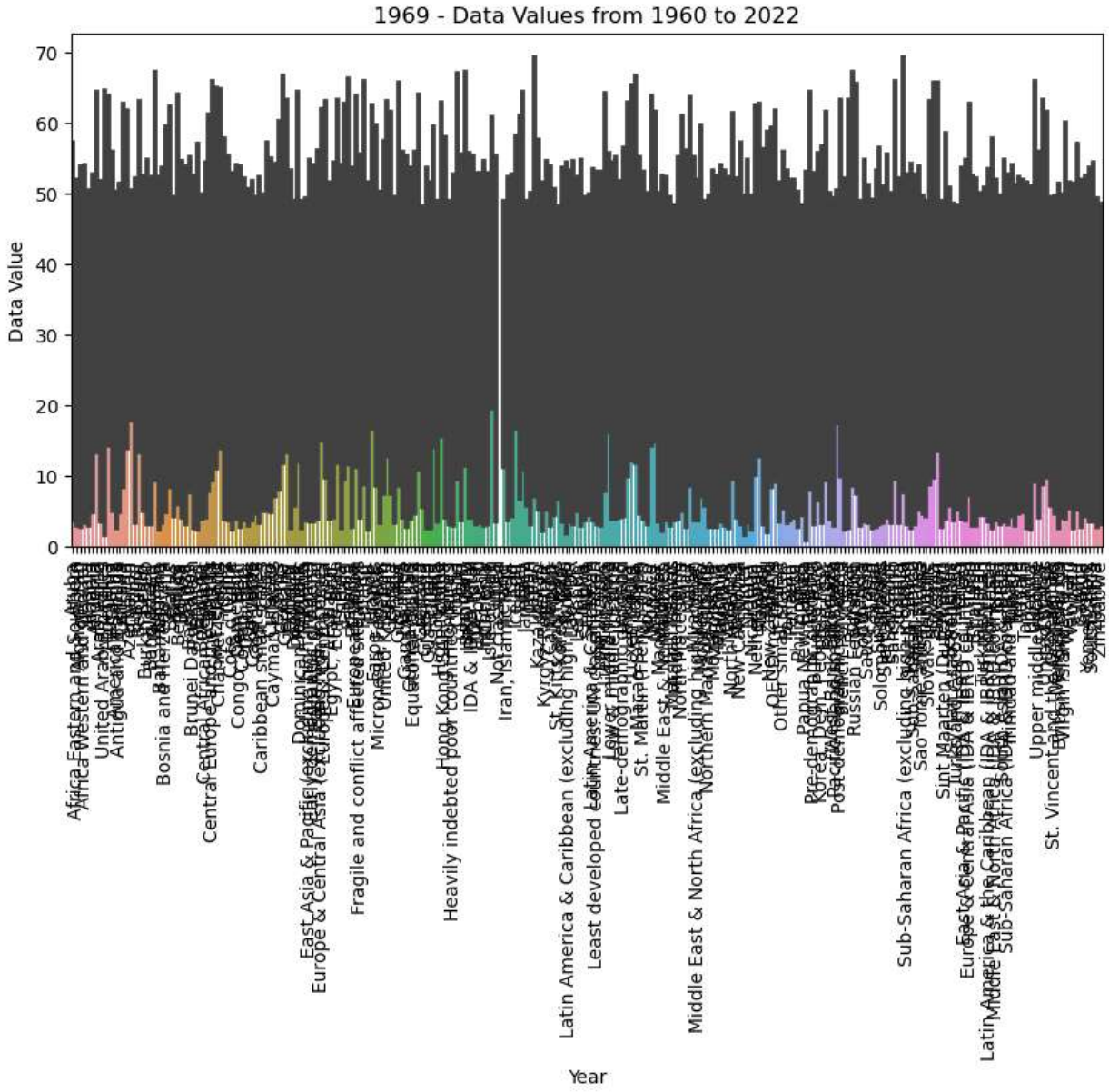

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



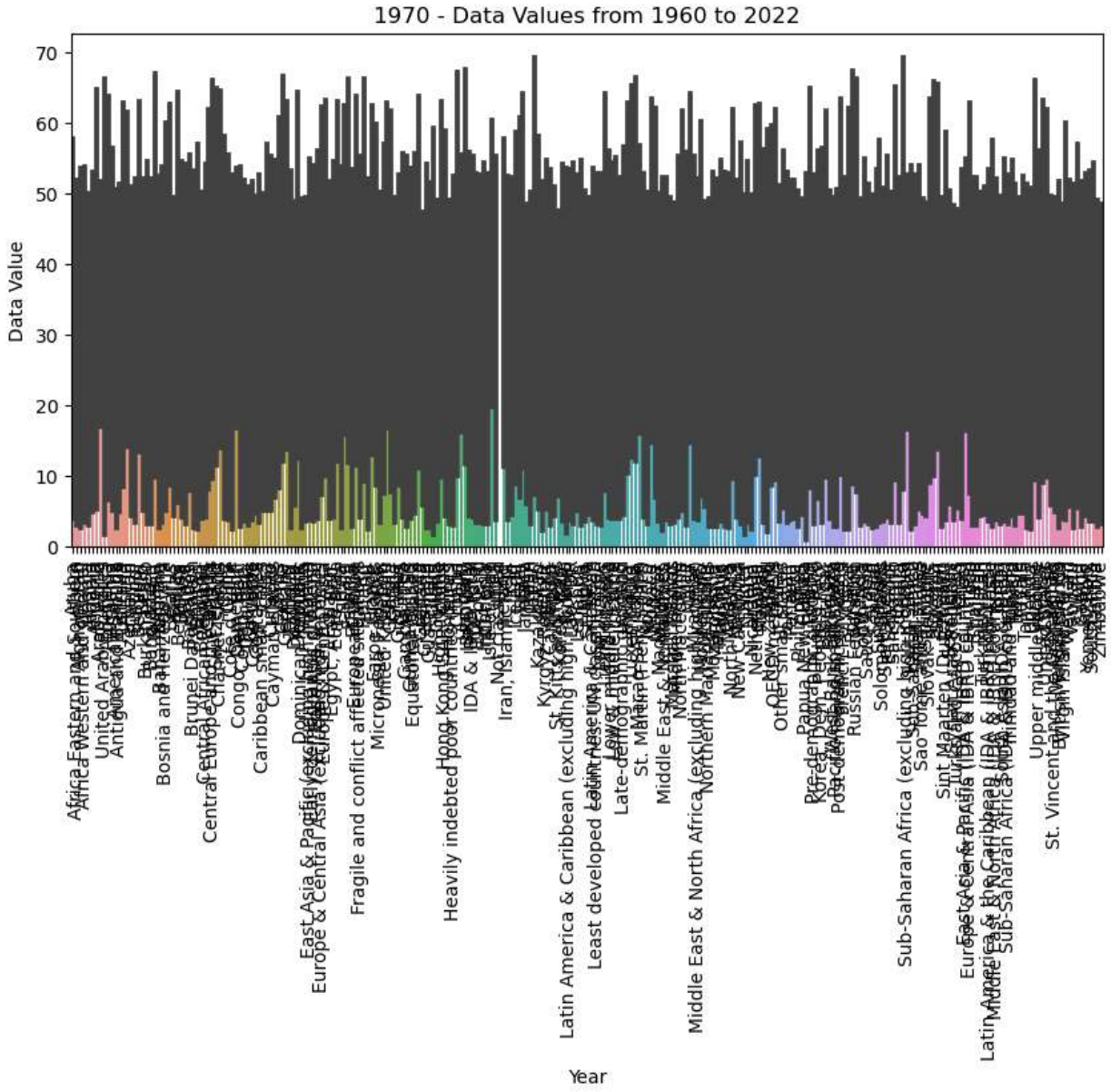

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



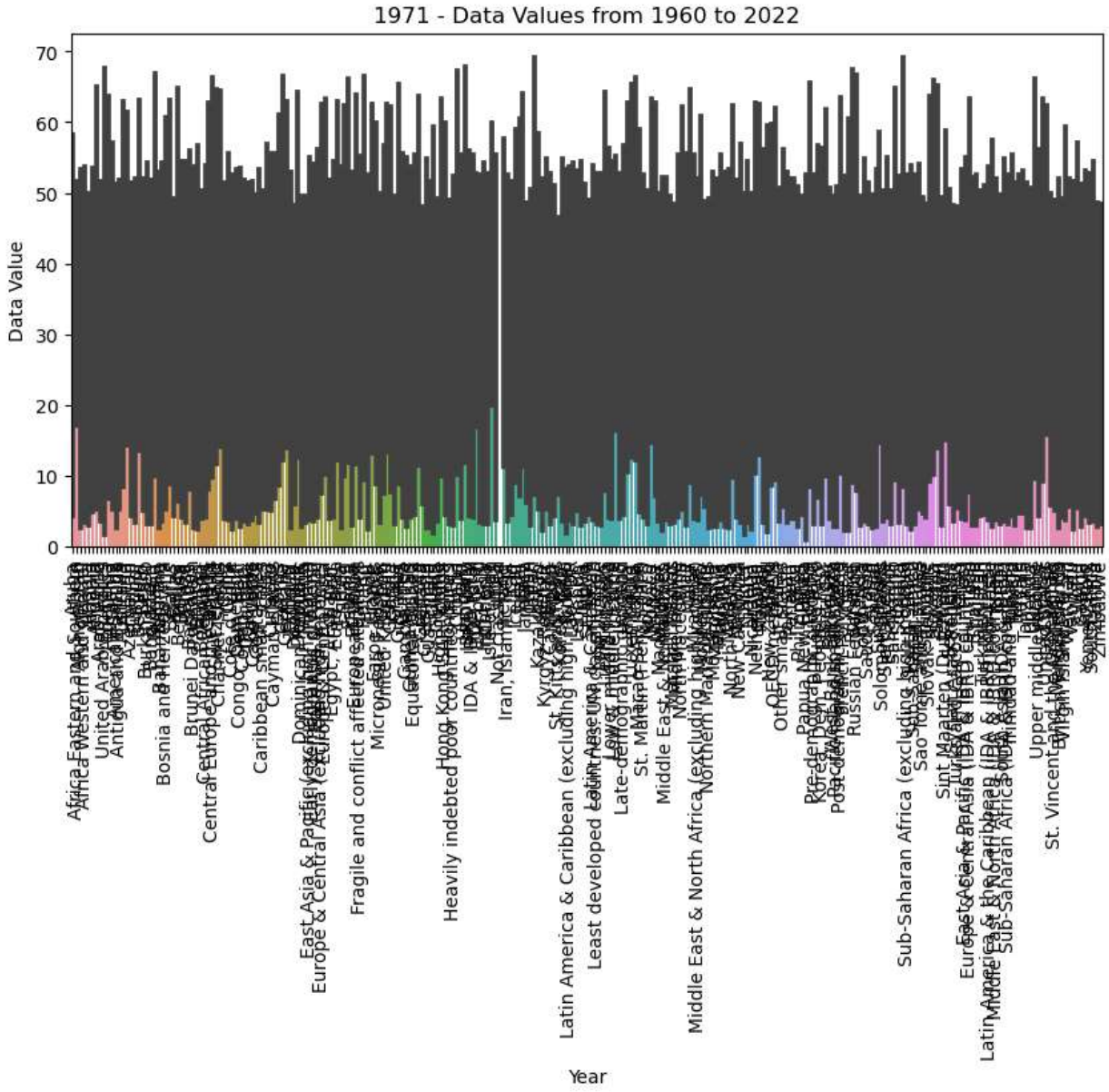

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



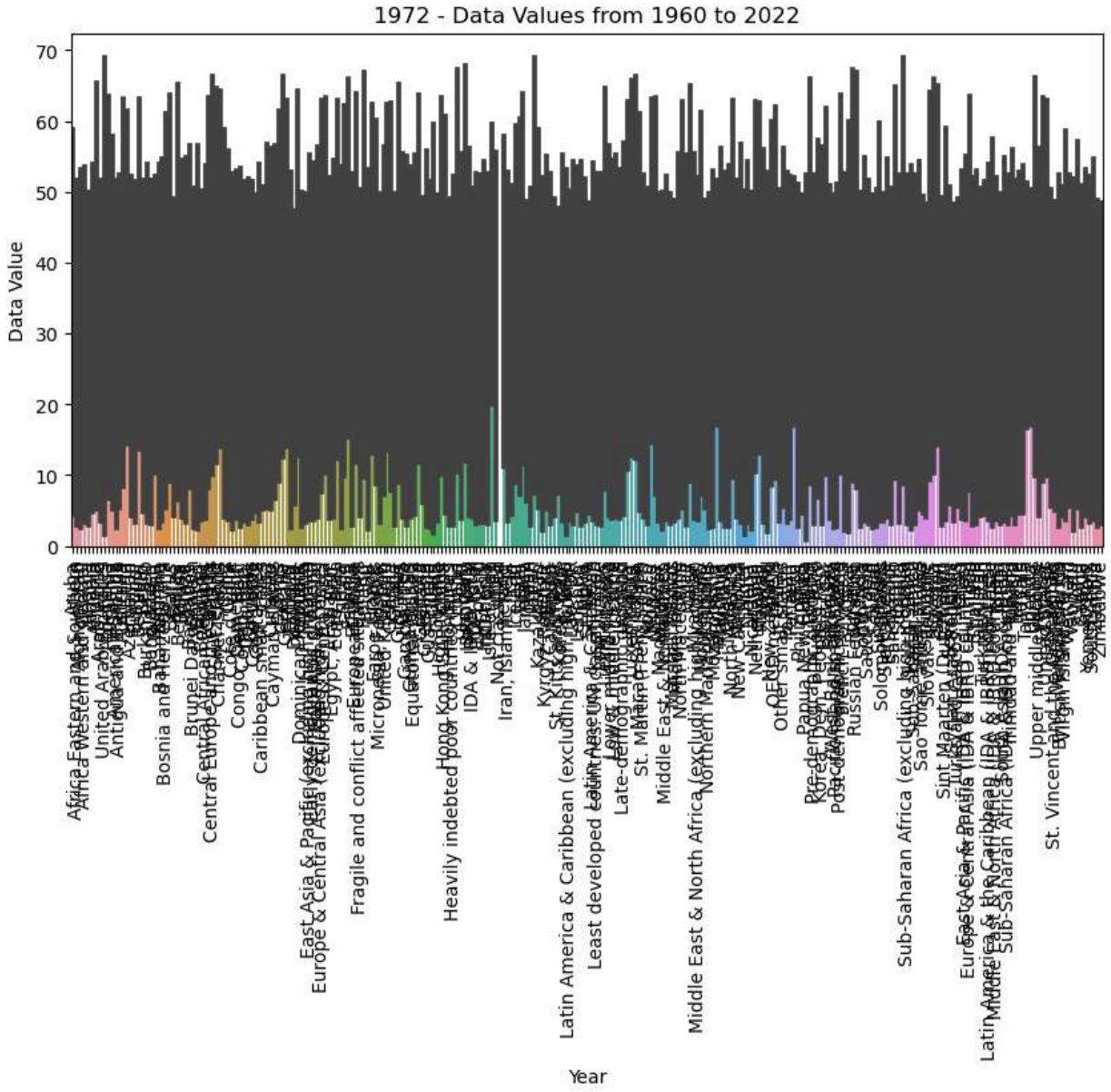

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



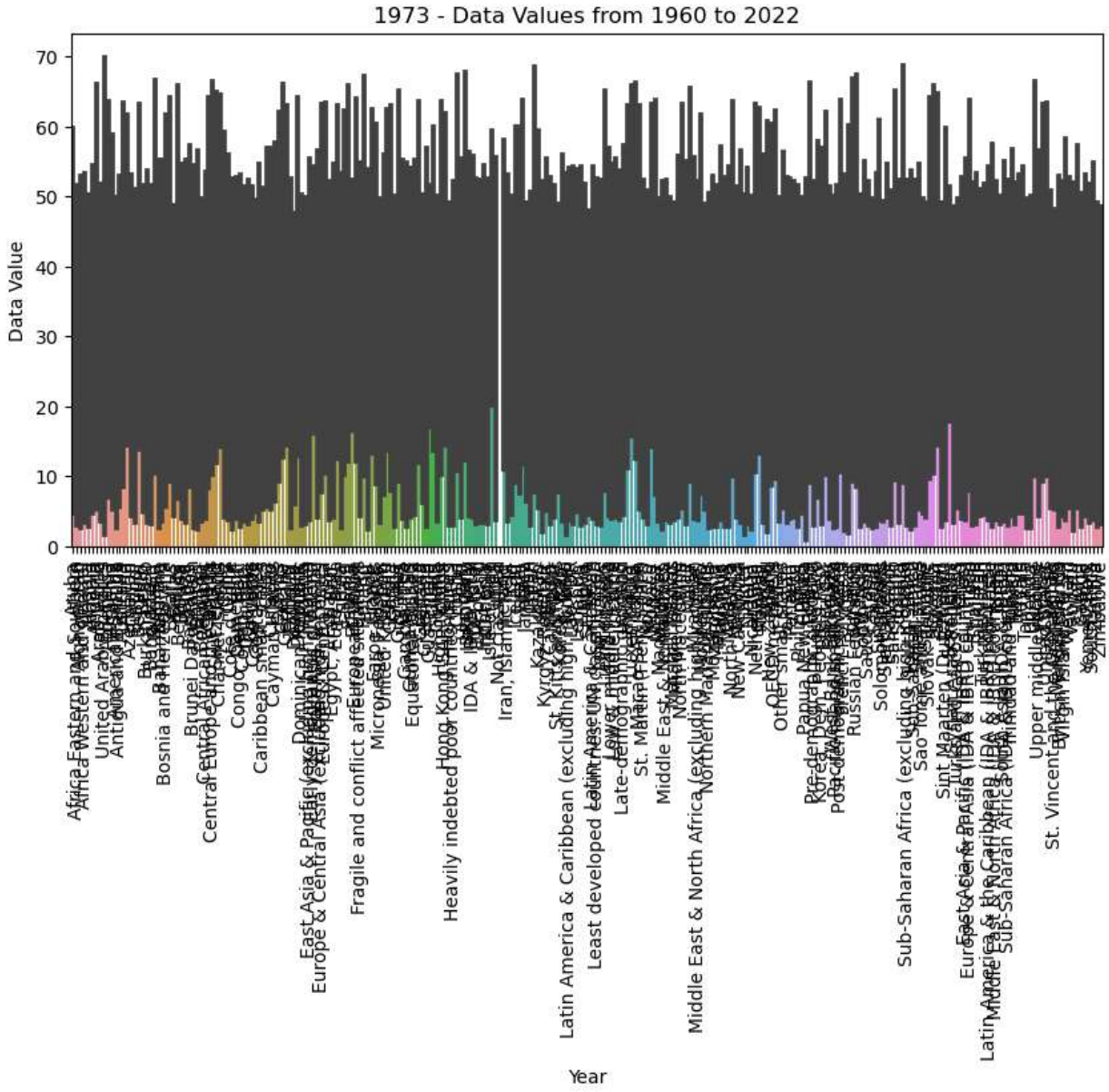

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



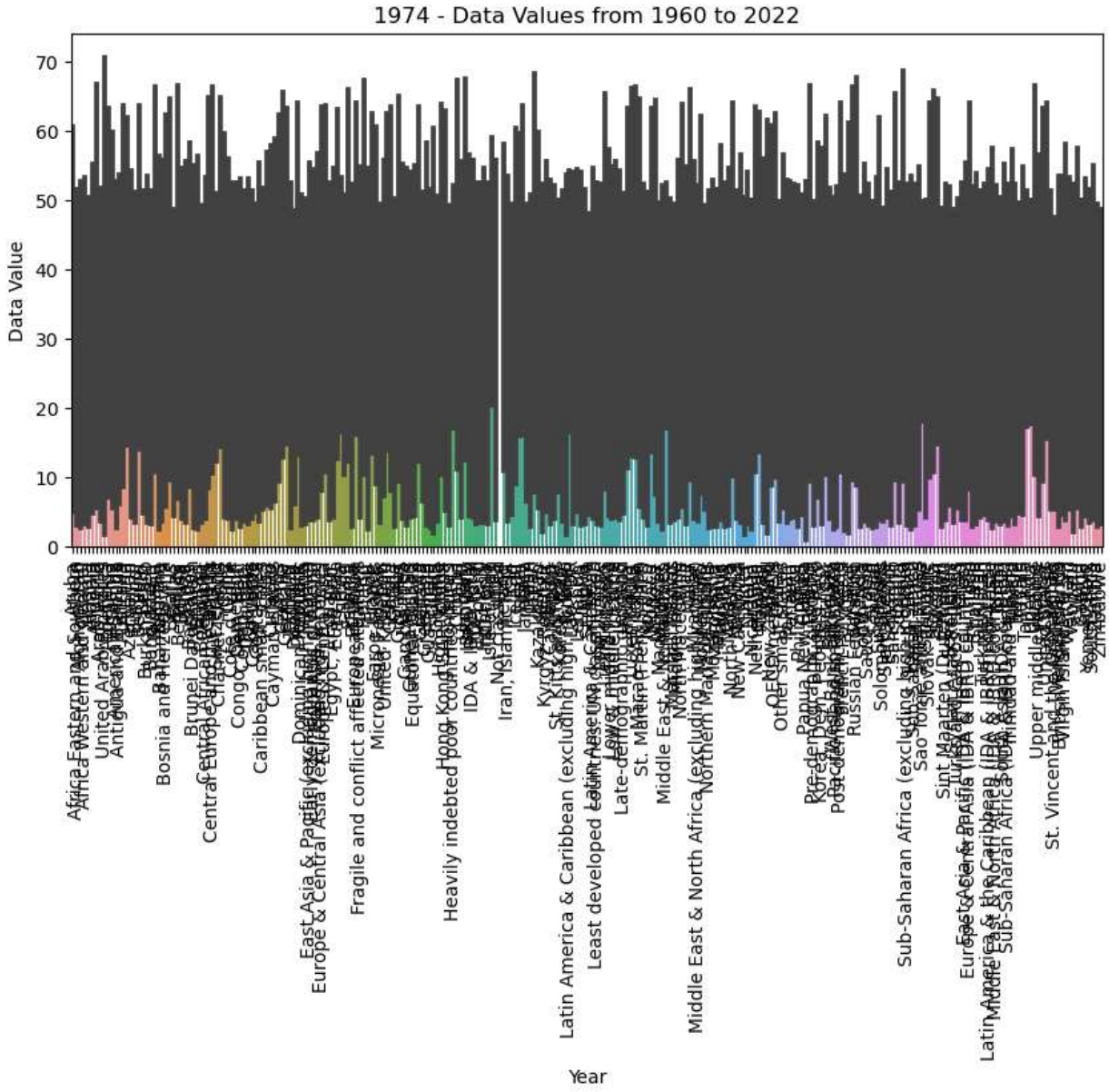

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



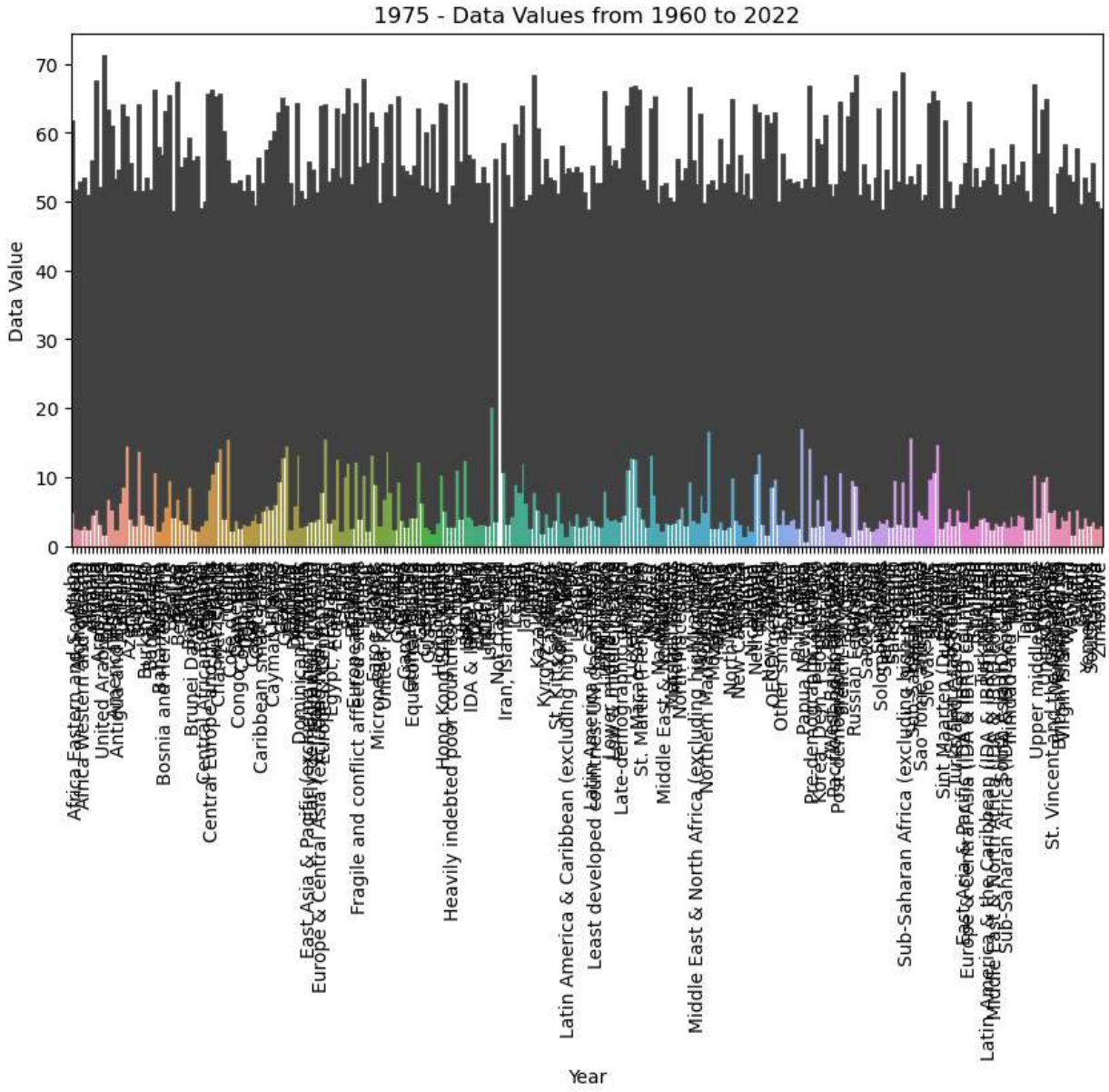

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



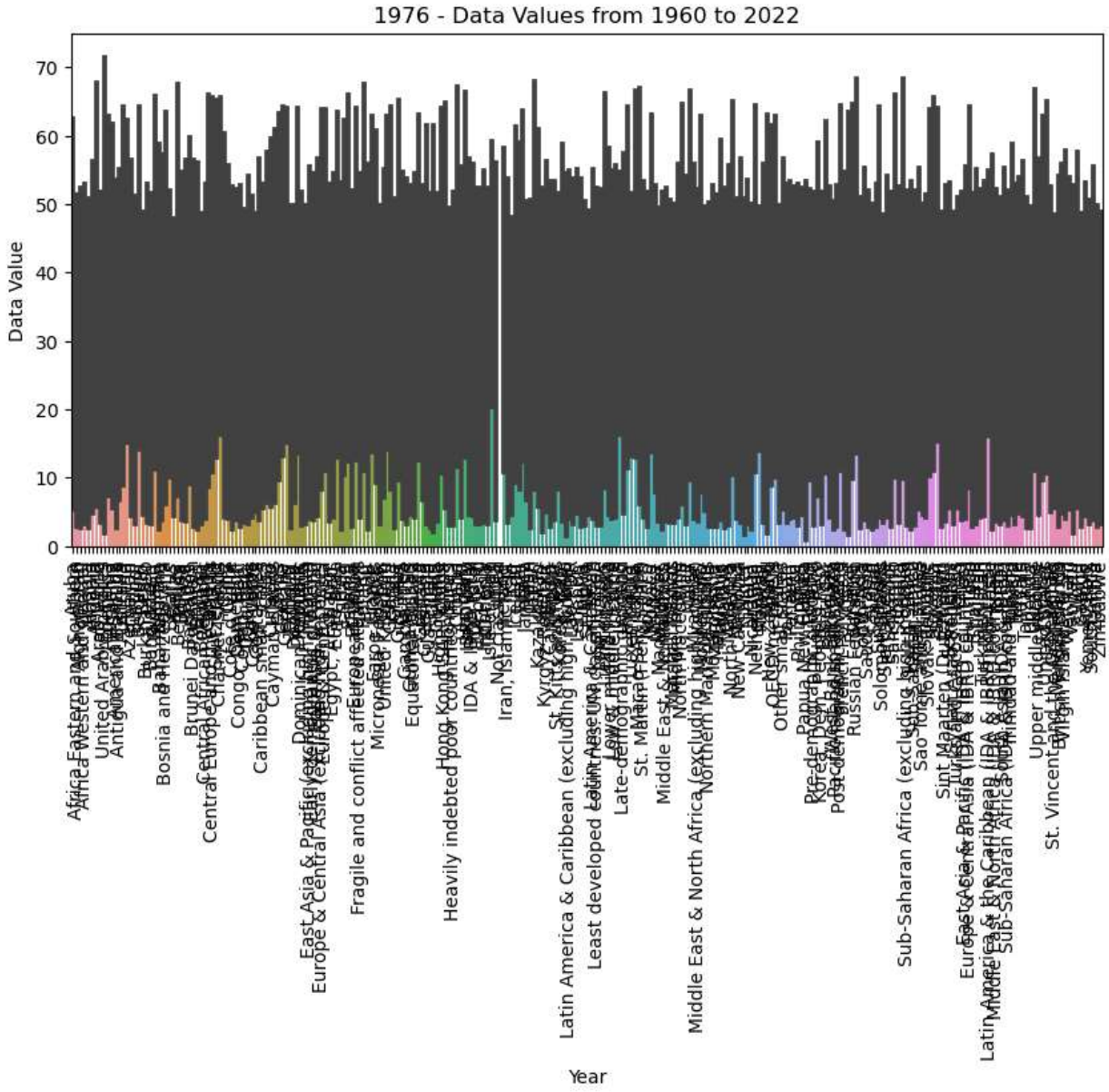

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



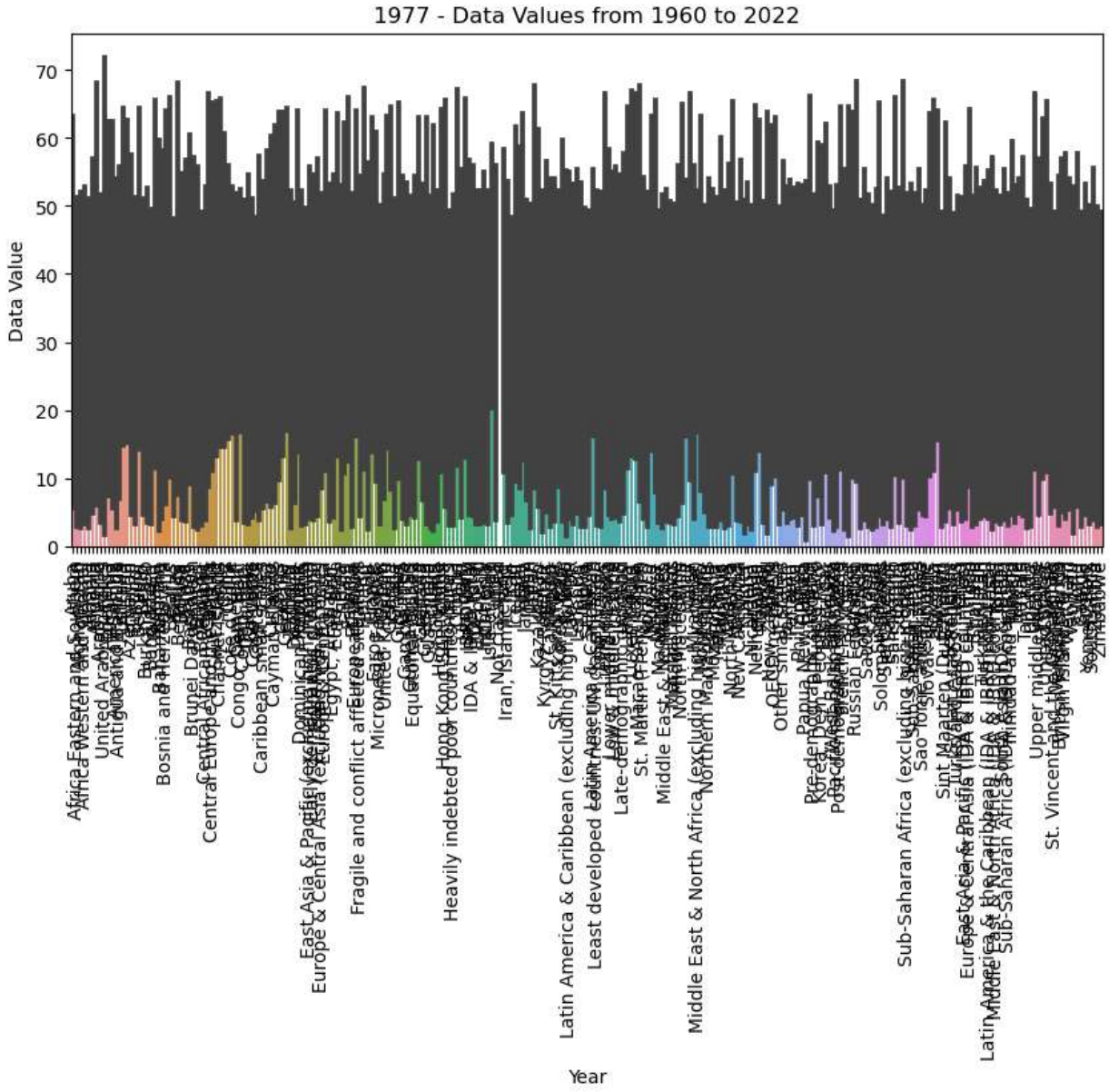

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



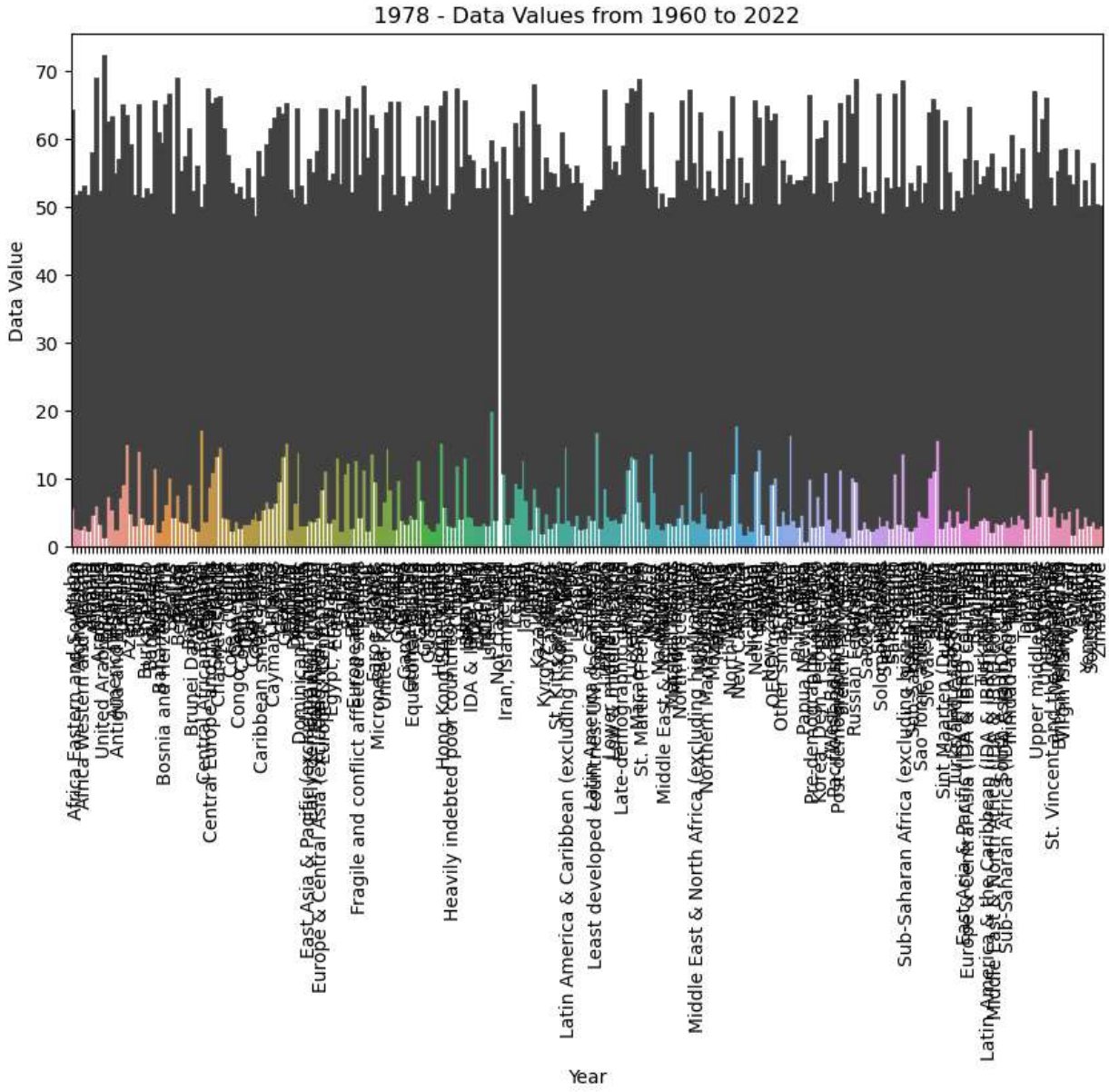

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



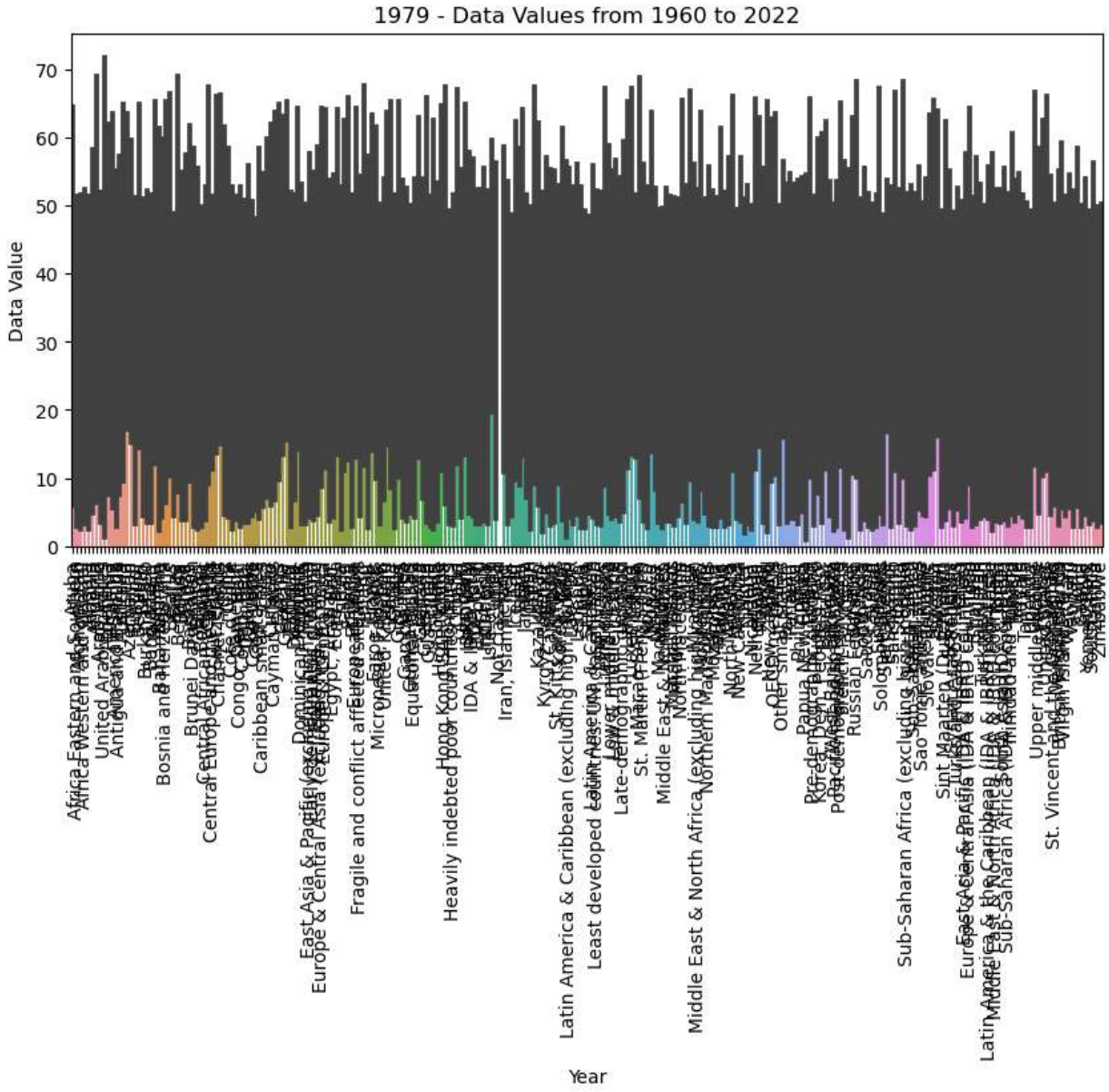

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



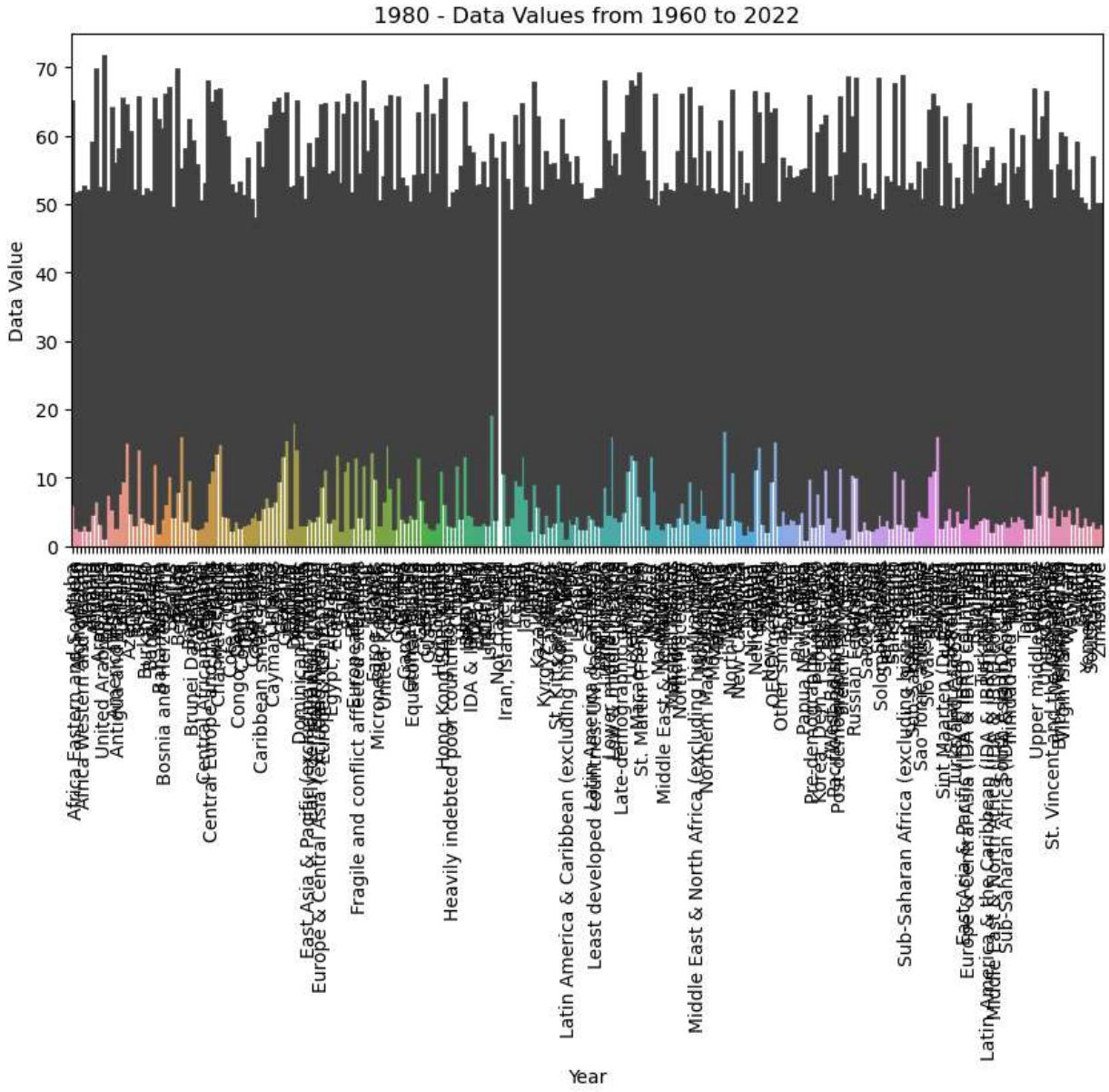

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



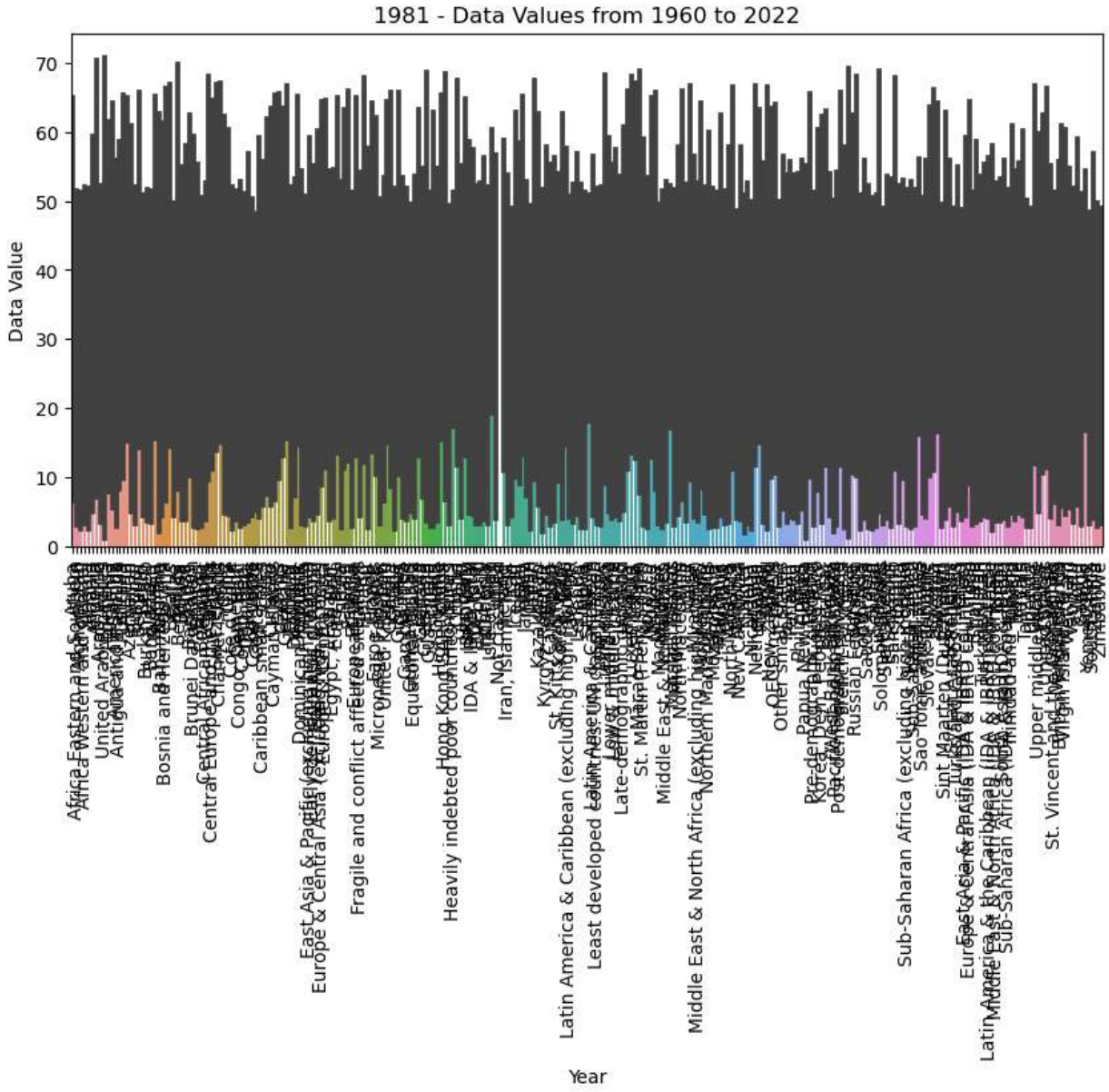

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



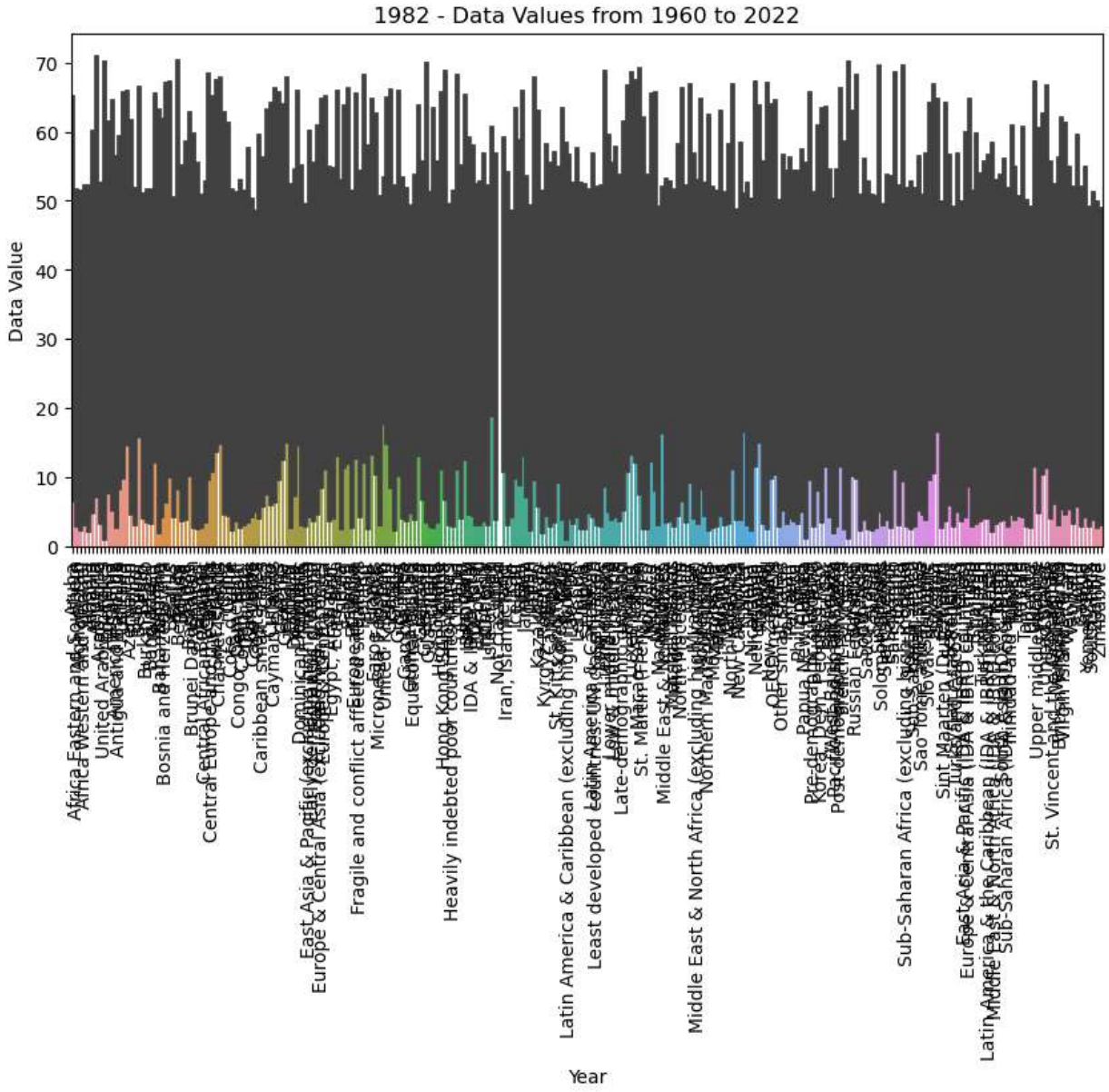

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



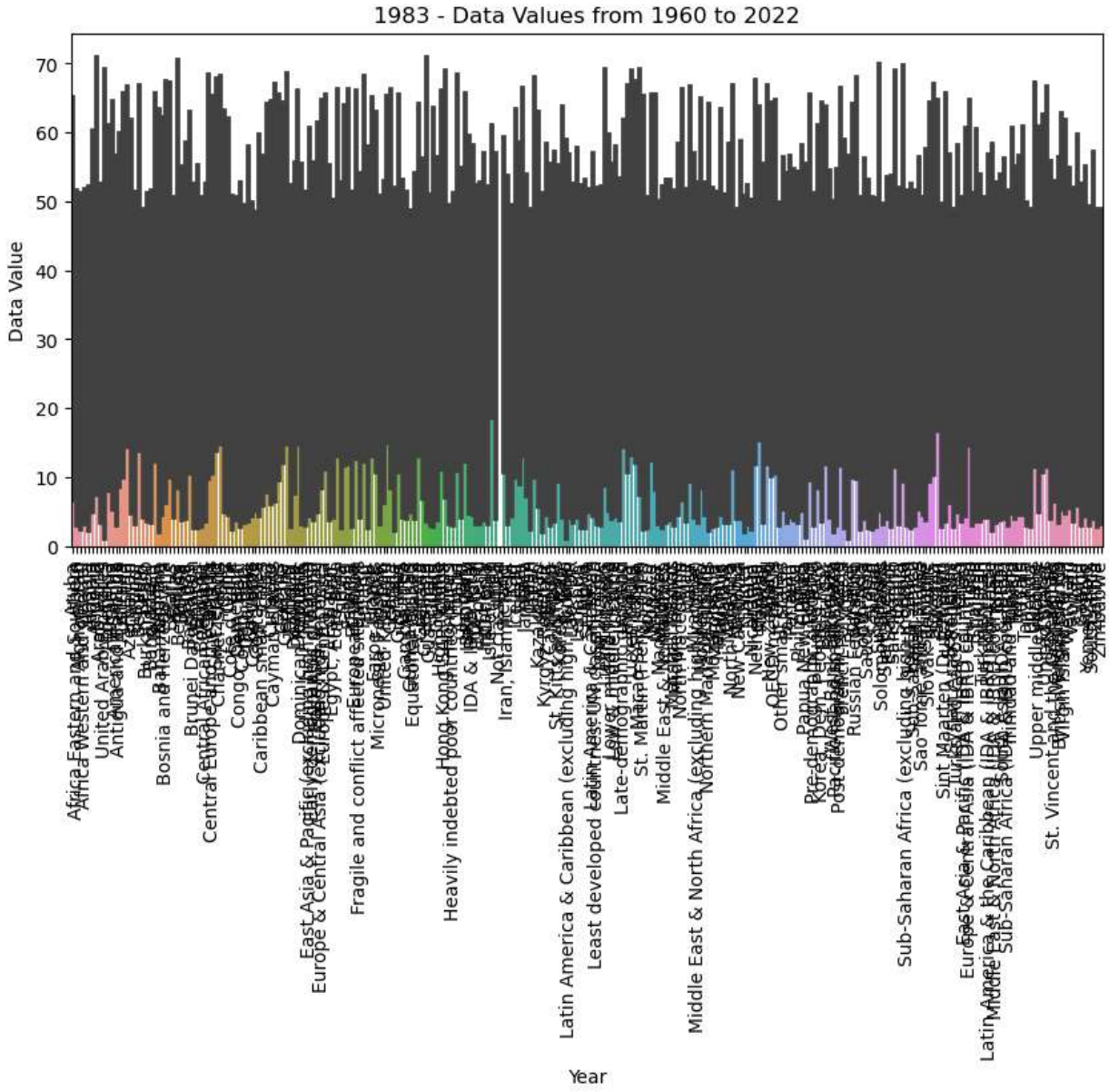

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```



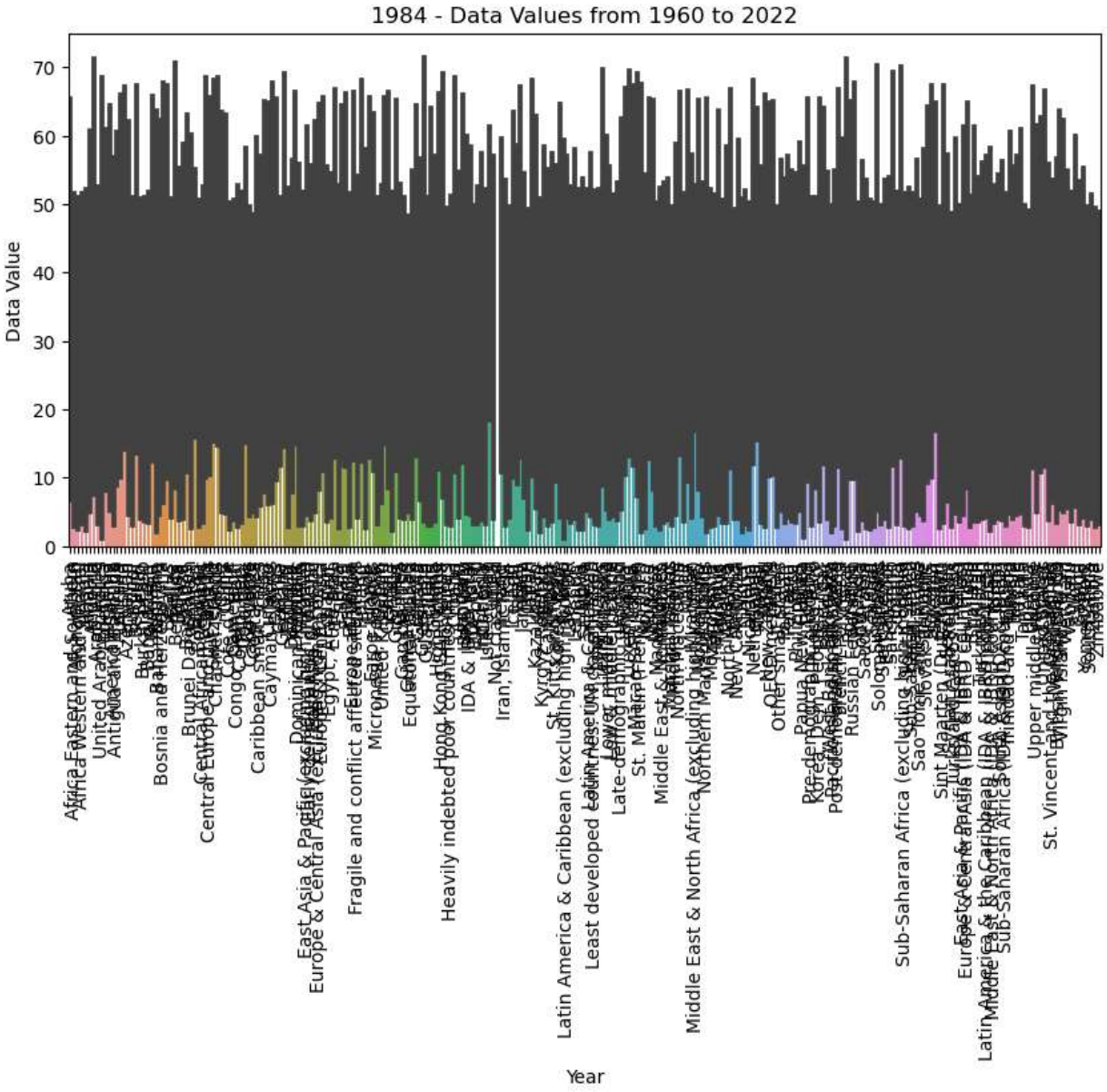

```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```




```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```




```
C:\Users\aryas\anaconda3\Lib\site-packages\seaborn\algorithms.py:98: RuntimeWarning:
Mean of empty slice
  boot_dist.append(f(*sample, **func_kwargs))
C:\Users\aryas\anaconda3\Lib\site-packages\numpy\lib\nanfunctions.py:1556: RuntimeWar
ning: All-NaN slice encountered
  return function_base._ureduce(a,
```


1984 - Data Values from 1960 to 2022
Data Value
0
10
20
30
40
50
60
70
Year