In [1]:
```python
import pandas as pd
import numpy as np
import altair as alt
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

1. Clean and Tidy Raw Data

In [2]:
```python
# import data
raw_country = pd.read_csv('ESGCountry.csv').loc[:, 'Country Code']
country = raw_country.to_numpy()
raw = pd.read_csv('ESGData.csv')

# merge the raw data with the Topics of Indicator Names
code = pd.read_csv('ESGSeries.csv').loc[:, ['Topic', 'Indicator Name']]
code['Topic'] = code['Topic'].str.split(':', expand = True)[0]
raw_merge = pd.merge(raw, code, how = 'left', on = 'Indicator Name')

# select the countries we need
raw_select = raw_merge.loc[raw_merge['Country Code'].isin(country)]

# tidy up the raw data
raw1 = raw_merge.loc[:, ['Country Name', 'Country Code', 'Indicator Name', 'Topic', '2019', '2020']]
raw1['Topic'] = raw1['Topic'].fillna('Social')
raw1 = raw1.sort_values('Topic')
```

In [3]:
```python
# melt and pivot the columns
clean = raw1.drop(
    columns = 'Topic'
).melt(
    id_vars = ['Country Name', 'Country Code', 'Indicator Name'],
    var_name = 'Year',
    value_name = 'level'
).pivot(
    index = ['Country Name', 'Country Code', 'Year'],
    columns = ['Indicator Name'],
    values = 'level'
)

# select the columns that has missing less than 8%
clean2 = clean.loc[:, (clean.isna().mean() <= 0.08)]
```

In [4]:
```python
# Rename the column names
col_names = {
        'Forest area (% of land area)': 'fore_area',
        'Adjusted savings: net forest depletion (% of GNI)': 'fore_dep',
        'Adjusted savings: natural resources depletion (% of GNI)': 'natu_res_dep',
        'Population density (people per sq. km of land area)': 'pop_denst',
        'Ratio of female to male labor force participation rate (%) (modeled ILO estimate)' : 'rate_labor',
        'GDP growth (annual %)': 'gdp_grow',
        'Unemployment, total (% of total labor force) (modeled ILO estimate)': 'unemp_rate',
        'Life expectancy at birth, total (years)': 'life_exp',
        'Access to electricity (% of population)': 'acce_electr',
        'Mortality rate, under-5 (per 1,000 live births)': 'mortal_rate',
        'Access to clean fuels and technologies for cooking (% of population)': 'acce_fuel_tech',
        'Population ages 65 and above (% of total population)': 'pop_65',
        'Fertility rate, total (births per woman)': 'ferti_rate',
        'Proportion of seats held by women in national parliaments (%)': 'parliment_women_seat'
    }

# selected the needed variables and fill the missing values with 0
data = clean2.rename(
    columns = col_names
).fillna(value = 0).reset_index()

# log the population density and convert the mortality rate to percentage
data['log_pop_denst'] = np.log10(data['pop_denst'])
data = data.drop(columns = ['pop_denst', 'ferti_rate'])
data['mortal_rate'] = data['mortal_rate']/1000
data.head()
```

Out[4]:

Indicator Name	Country Name	Country Code	Year	acce_fuel_tech	acce_electr	fore_area	gdp_grow	life_exp	mortal_rate	pop_65	parliment_women_seat	rate_labor	unemp_rate	log_pop_denst
0	Afghanistan	AFG	2019	31.9	97.699997	1.850994	3.911603	64.833	0.0601	2.615794	27.868852	30.009880	11.217	1.765441
1	Afghanistan	AFG	2020	33.2	97.699997	1.850994	-2.351101	65.173	0.0580	2.649070	27.016129	24.685878	11.710	1.775446
2	Albania	ALB	2019	80.7	100.000000	28.791971	2.113420	78.573	0.0097	14.202631	29.508197	77.829119	11.470	2.017732
3	Albania	ALB	2020	81.3	100.000000	28.791971	-3.955398	78.686	0.0098	14.704581	29.508197	75.857448	13.329	2.015239
4	Algeria	DZA	2019	99.7	99.500000	0.814110	1.000000	76.880	0.0233	6.552778	25.757576	24.877860	10.513	1.257109

2. Exploratory analysis

Heat Map

In [5]:
```python
# find the correlation between each variable
corr_mx = data.drop(columns = ['Country Name', 'Country Code', 'Year']).corr()
corr_mx
```

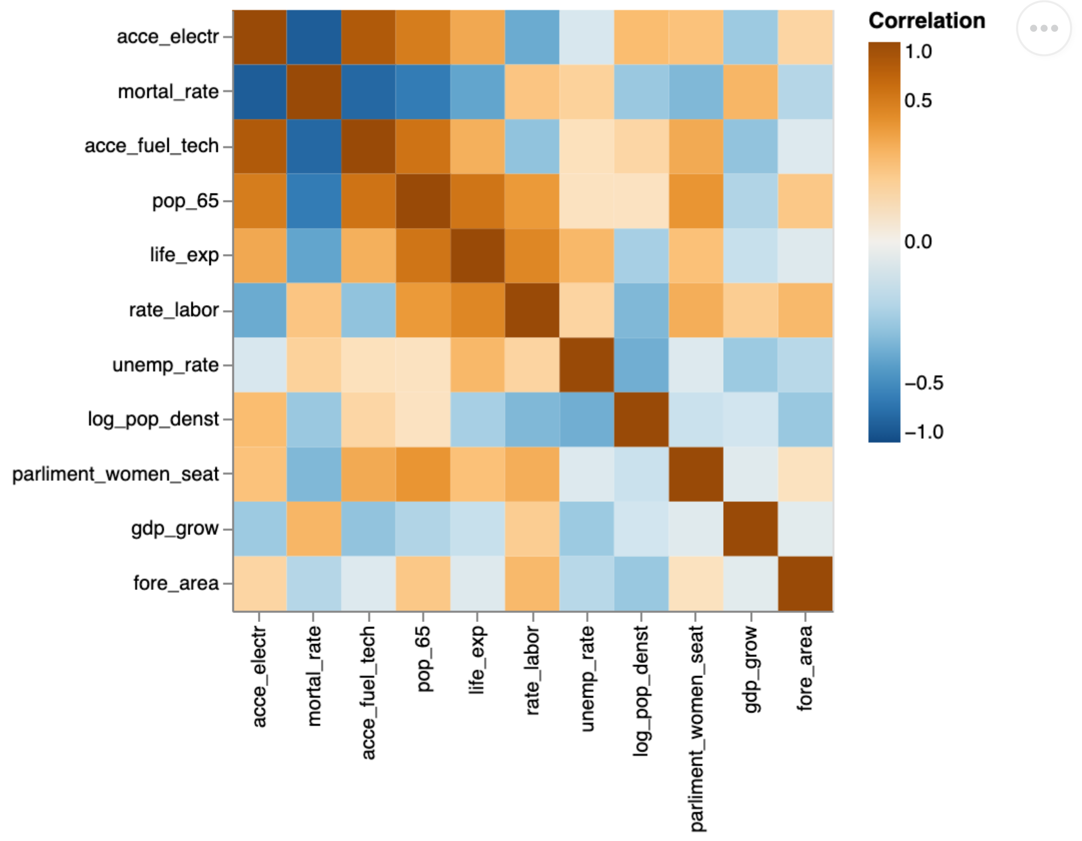
Out[5]:

Indicator Name Indicator Name	acce_fuel_tech	acce_electr	fore_area	gdp_grow	life_exp	mortal_rate	pop_65	parliment_women_seat	rate_labor	unemp_rate	log_pop_denst
acce_fuel_tech	1.000000	0.805142	-0.017538	-0.189063	0.221254	-0.756640	0.563688	0.246315	-0.190538	0.037127	0.067167
acce_electr	0.805142	1.000000	0.070259	-0.160301	0.252749	-0.842890	0.498690	0.145019	-0.308571	-0.026707	0.168424
fore_area	-0.017538	0.070259	1.000000	-0.009624	-0.015225	-0.096213	0.120457	0.032211	0.182658	-0.088003	-0.170028
gdp_grow	-0.189063	-0.160301	-0.009624	1.000000	-0.057843	0.195370	-0.105106	-0.013033	0.099554	-0.161315	-0.040207
life_exp	0.221254	0.252749	-0.015225	-0.057843	1.000000	-0.342652	0.552923	0.150092	0.431192	0.185842	-0.130302
mortal_rate	-0.756640	-0.842890	-0.096213	0.195370	-0.342652	1.000000	-0.610863	-0.239346	0.130139	0.082735	-0.167586
pop_65	0.563688	0.498690	0.120457	-0.105106	0.552923	-0.610863	1.000000	0.347114	0.324953	0.030698	0.029666
parliment_women_seat	0.246315	0.145019	0.032211	-0.013033	0.150092	-0.239346	0.347114	1.000000	0.229315	-0.017977	-0.052417
rate_labor	-0.190538	-0.308571	0.182658	0.099554	0.431192	0.130139	0.324953	0.229315	1.000000	0.074020	-0.239964
unemp_rate	0.037127	-0.026707	-0.088003	-0.161315	0.185842	0.082735	0.030698	-0.017977	0.074020	1.000000	-0.289614
log_pop_denst	0.067167	0.168424	-0.170028	-0.040207	-0.130302	-0.167586	0.029666	-0.052417	-0.239964	-0.289614	1.000000

In [6]:
```python
# melt corr_mx
corr_mx_long = corr_mx.reset_index().rename(
    columns = {'Indicator Name': 'row'}
).melt(
    id_vars = 'row',
    var_name = 'col',
    value_name = 'Correlation'
)
```

```
# construct plot
alt.Chart(corr_mx_long).mark_rect().encode(
    x = alt.X('col', title = '', sort = {'field': 'Correlation', 'order': 'ascending'}),
    y = alt.Y('row', title = '', sort = {'field': 'Correlation', 'order': 'ascending'}),
    color = alt.Color('Correlation',
    scale = alt.Scale(scheme = 'blueorange', domain = (-1, 1), type = 'sqrt'),
    legend = alt.Legend(tickCount = 5))
).properties(width = 300, height = 300)
```

Out[6]:



Visualization of Summary Statistics

In [7]:

```
# provide the summary of each column variables
# Summary of 2019 data
data_summary19 = data[data.Year == '2019'].drop(
    columns = ['Country Name', 'Country Code', 'Year']
).aggregate(
    ['mean', 'std']
).transpose().reset_index()

# Summary of 2020 data
data_summary20 = data[data.Year == '2020'].drop(
    columns = ['Country Name', 'Country Code', 'Year']
).aggregate(
    ['mean', 'std']
).transpose().reset_index()

# concat two years summary data together
data_summary = pd.concat([data_summary19, data_summary20], keys=['2019', '2020']).reset_index().drop(columns = 'level_1')
plot_df = data_summary.rename(columns = {'Indicator Name': 'ESG Variables', 'level_0': 'Year'})
plot_df.head()
```

Out[7]:

	Year	ESG Variables	mean	std
0	2019	acce_fuel_tech	65.664854	37.181829
1	2019	acce_electr	84.758702	24.148638
2	2019	fore_area	32.074655	22.725592
3	2019	gdp_grow	2.901895	2.847891
4	2019	life_exp	69.615350	15.508656

In [8]:

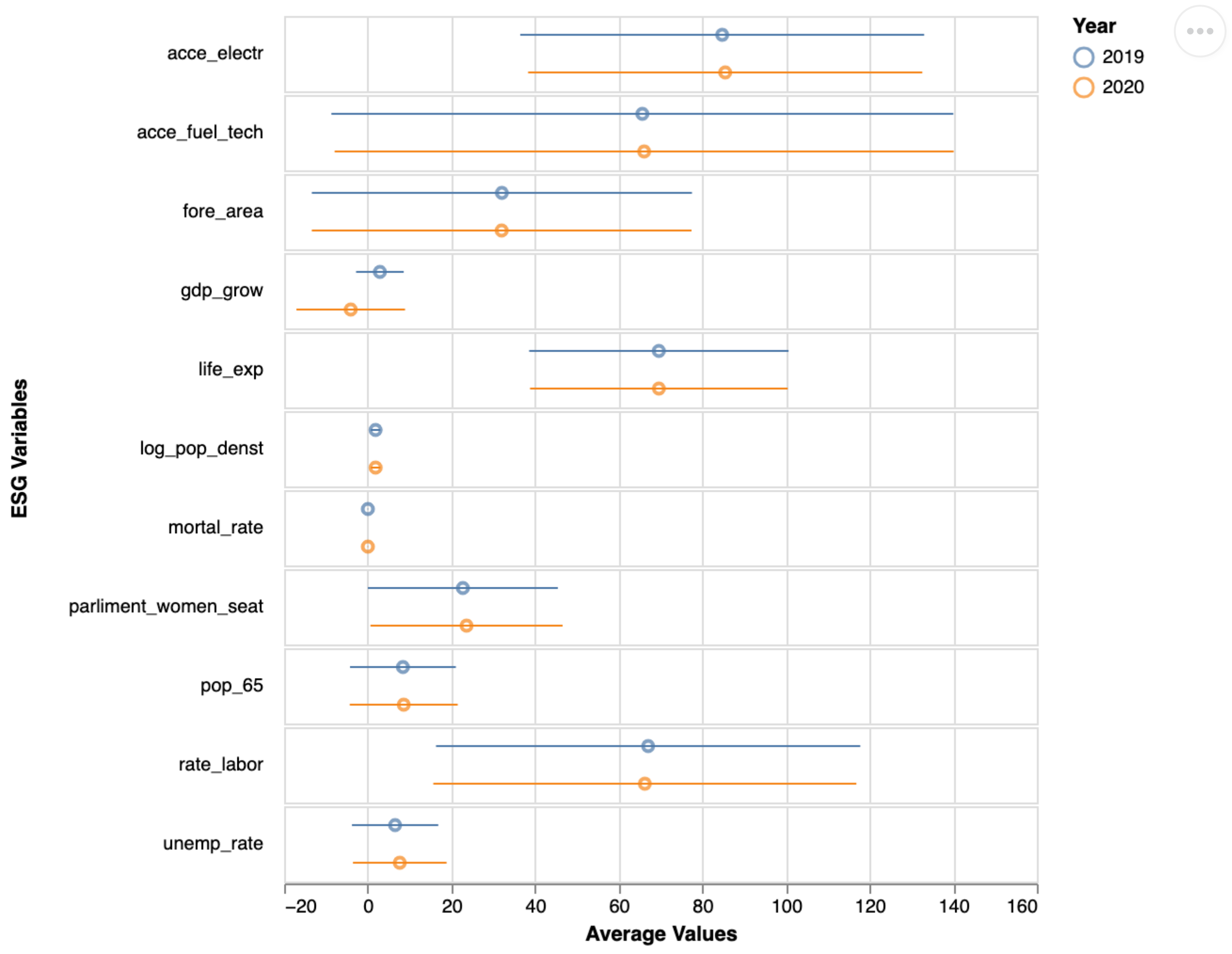
```
# visualizing summary statistics
points = alt.Chart(plot_df).mark_point().encode(
    x = alt.X('mean', title = 'Average Values'),
    y = alt.Y('Year', title = '', axis = None),
    color = alt.Color('Year', title = 'Year')
)

# variability about means
bars = alt.Chart(plot_df).transform_calculate(
    lwr = 'datum.mean - 2*datum.std',
    upr = 'datum.mean + 2*datum.std'
).mark_errorbar().encode(
    x = alt.X('lwr:Q', title = 'Average Values'),
    x2 = 'upr:Q',
    y = alt.Y('Year', title = '', axis = None),
    color = alt.Color('Year', title = 'Year')
)

# layer
fig = (points + bars).facet(
    row = alt.Row('ESG Variables',
                  title = 'ESG Variables',
                  header = alt.Header(labelAngle = 0,
                                      labelAlign = 'left'))
).configure_facet(spacing = 0)

# display
fig
```

Out[8]:



Distribution Changes of GDP Growth in 2019 and 2020

Since the averages of GPD Growth changes from 2019 to 2020, we are expected to know how the GDP growth is distributed.

In [9]:
```
# distribution of gdp growth across each country in 2019
base = alt.Chart(data)

hist19 = base.transform_filter(
     alt.FieldEqualPredicate(
         field = 'Year', equal = '2019')
).transform_bin(
    as_ = 'bin',
    field = 'gdp_grow',
    bin = alt.Bin(step = 1)
).transform_aggregate(
    Count = 'count()',
    groupby = ['bin']
).transform_calculate(
        Density = 'datum.Count/(1*239)',
).mark_bar(opacity=0.8).encode(
    x = alt.X('bin:Q',
        title = 'GDP Growth in 2019',
        scale = alt.Scale(domain = (-15, 20))),
    y = 'Density:Q'
)

smooth19 = alt.Chart(data).transform_filter(
     alt.FieldEqualPredicate(
         field = 'Year', equal = '2019')
).transform_density(
    density = 'gdp_grow', # variable to smooth
    as_ = ['bin', 'Estimated Density'], # names of outputs
    bandwidth = 1, # how smooth?
    extent = [-15, 20], # domain on which the smooth is defined
    steps = 1000 # for plotting: number of points to generate for plotting line
).mark_line(color = 'black').encode(
    x = 'bin:Q',
    y = 'Estimated Density:Q'
)

gdp_plot19 = hist19 + smooth19
```

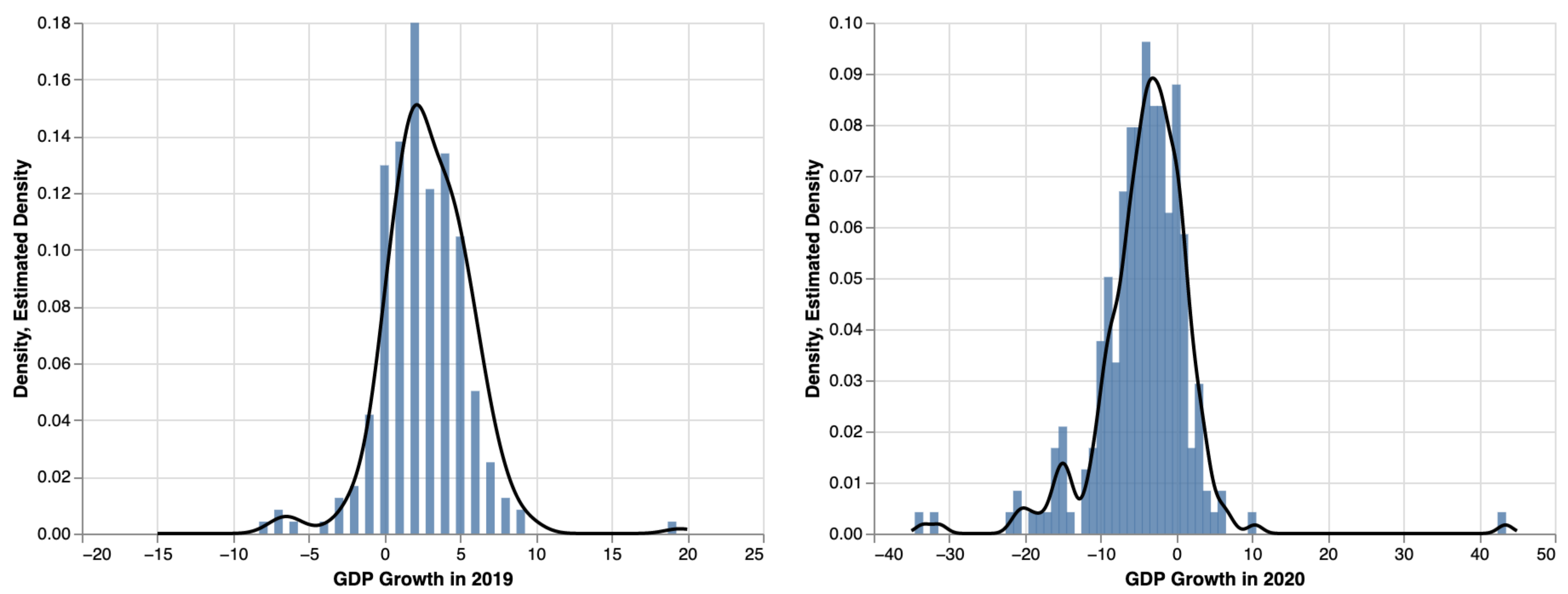
In [10]:
```
# distribution of gdp growth across each country in 2020
hist20 = base.transform_filter(
     alt.FieldEqualPredicate(
         field = 'Year', equal = '2020')
).transform_bin(
    as_ = 'bin',
    field = 'gdp_grow',
    bin = alt.Bin(step = 1)
).transform_aggregate(
    Count = 'count()',
    groupby = ['bin']
).transform_calculate(
        Density = 'datum.Count/(1*239)',
).mark_bar(opacity=0.8).encode(
    x = alt.X('bin:Q',
        title = 'GDP Growth in 2020',
        scale = alt.Scale(domain = (-35, 40))),
    y = 'Density:Q'
)

smooth20 = alt.Chart(data).transform_filter(
     alt.FieldEqualPredicate(
         field = 'Year', equal = '2020')
).transform_density(
    density = 'gdp_grow', # variable to smooth
    as_ = ['bin', 'Estimated Density'], # names of outputs
    bandwidth = 1, # how smooth?
    extent = [-35, 45], # domain on which the smooth is defined
    steps = 1000 # for plotting: number of points to generate for plotting line
).mark_line(color = 'black').encode(
    x = 'bin:Q',
    y = 'Estimated Density:Q'
)

gdp_plot20 = hist20 + smooth20
```

In [11]:
```
gdp_plot19 | gdp_plot20
```

Out[11]:

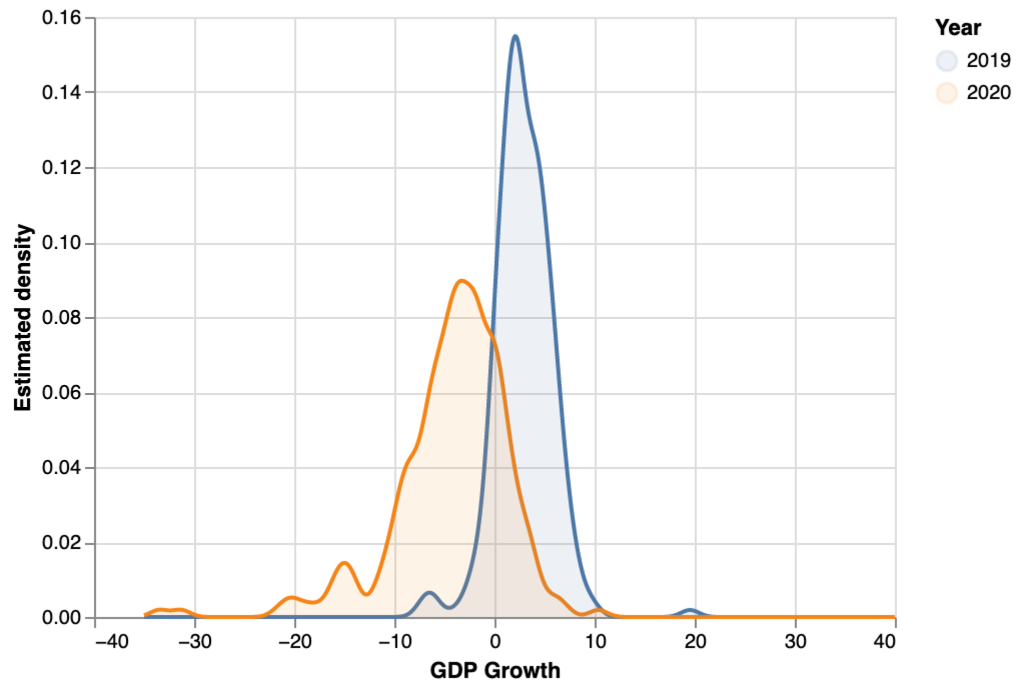


In [12]:
```
# Combine the distribution together
kdes = alt.Chart(data).transform_density(
    density = 'gdp_grow',
    groupby = ['Year'],
    as_ = ['GDP Growth', 'Estimated density'],
    bandwidth = 0.9,
    extent = [-35, 40],
    steps = 1000
).mark_line().encode(
    x = alt.X('GDP Growth:Q'),
    y = alt.Y('Estimated density:Q'),
    color = alt.Color('Year:N', legend = alt.Legend(title = 'Year'))
)

kdes + kdes.mark_area(order = False, opacity = 0.1)

# although gpd_grow decreases but it does not have much effect on the other variable
```

Out[12]:



3. Principal Components Analysis

We consider using PCA to analysis which ESG variable drives the variation more, is GDP Growth one of them?

In [13]:
```
# Normalize the dataset
pcdata_raw = data.set_index(['Country Name', 'Country Code', 'Year'])
pcdata = (pcdata_raw - pcdata_raw.mean())/pcdata_raw.std()

# Compute the Principal components
pca = PCA(n_components = pcdata.shape[1])
pca.fit(pcdata)
```

Out[13]: PCA(n_components=11)

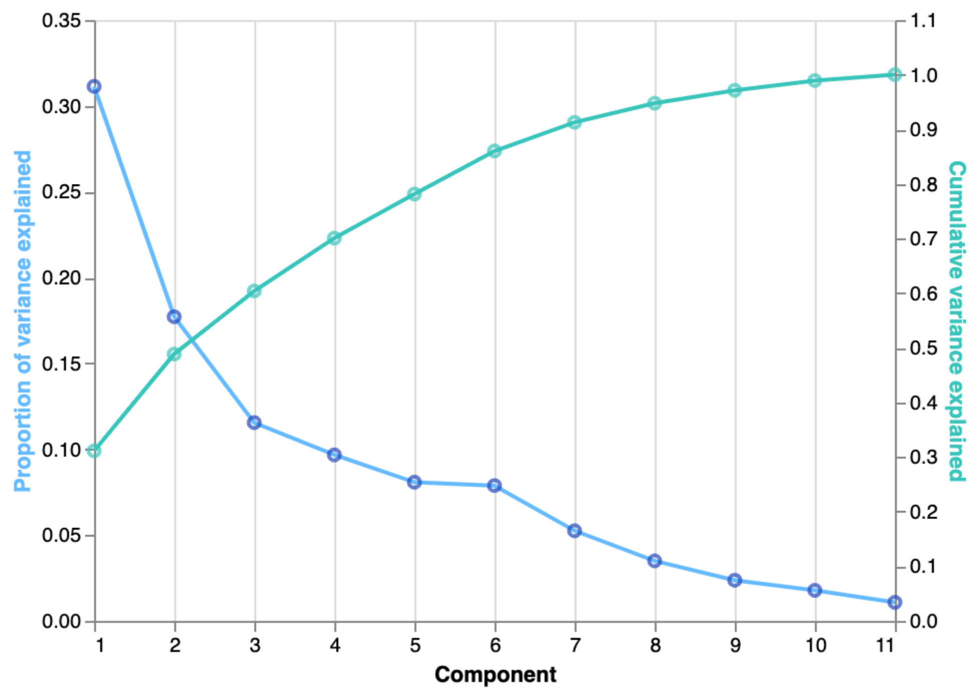
In [14]:
```
# store proportion of variance explained as a dataframe
pcvars = pd.DataFrame({'Proportion of variance explained': pca.explained_variance_ratio_})
# add component number as a new column
pcvars['Component'] = np.arange(1, 12)
# add cumulative variance explained as a new column
pcvars['Cumulative variance explained'] = np.cumsum(pcvars['Proportion of variance explained'])
# print
pcvars
```

Out[14]:

	Proportion of variance explained	Component	Cumulative variance explained
0	0.311327	1	0.311327
1	0.177151	2	0.488478
2	0.115490	3	0.603968
3	0.096713	4	0.700681
4	0.080778	5	0.781459
5	0.078817	6	0.860276
6	0.052480	7	0.912756
7	0.034957	8	0.947713
8	0.023678	9	0.971391
9	0.017808	10	0.989199
10	0.010801	11	1.000000

In [15]:
```
# encode component axis only as base layer
base = alt.Chart(pcvars).encode(x = 'Component')
# make a base layer for the proportion of variance explained
prop_var_base = base.encode(
    y = alt.Y('Proportion of variance explained', axis = alt.Axis(titleColor = '#66BBFF'))
)
# make a base layer for the cumulative variance explained
cum_var_base = base.encode(
    y = alt.Y('Cumulative variance explained', axis = alt.Axis(titleColor = '#39C5BB'))
)
# add points and lines to each base layer
prop_var = prop_var_base.mark_line(stroke = '#66BBFF') + prop_var_base.mark_point(color = '#2449BB')
cum_var = cum_var_base.mark_line(stroke = '#39C5BB') + cum_var_base.mark_point(color = '#39C5BB')
# layer the layers
var_explained_plot = alt.layer(prop_var, cum_var).resolve_scale(y = 'independent')
var_explained_plot
```

Out[15]:



In [16]:
```
# store the loadings as a data frame with appropriate names
loading_df = pd.DataFrame(pca.components_).transpose().rename(
    columns = {0: 'PC1', 1: 'PC2'}
).loc[:, ['PC1', 'PC2']]
# add a column with the taxon names
loading_df['ESG Variables'] = pcdata_raw.columns.values
# print
loading_df.head()
```

Out[16]:

	PC1	PC2	ESG Variables
0	-0.466300	0.146653	acce_fuel_tech
1	-0.469304	0.230413	acce_electr
2	-0.047138	-0.172484	fore_area
3	0.135179	-0.075788	gdp_grow
4	-0.269861	-0.415530	life_exp

In [17]:
```
# melt from wide to long
loading_plot_df = loading_df.melt(
```

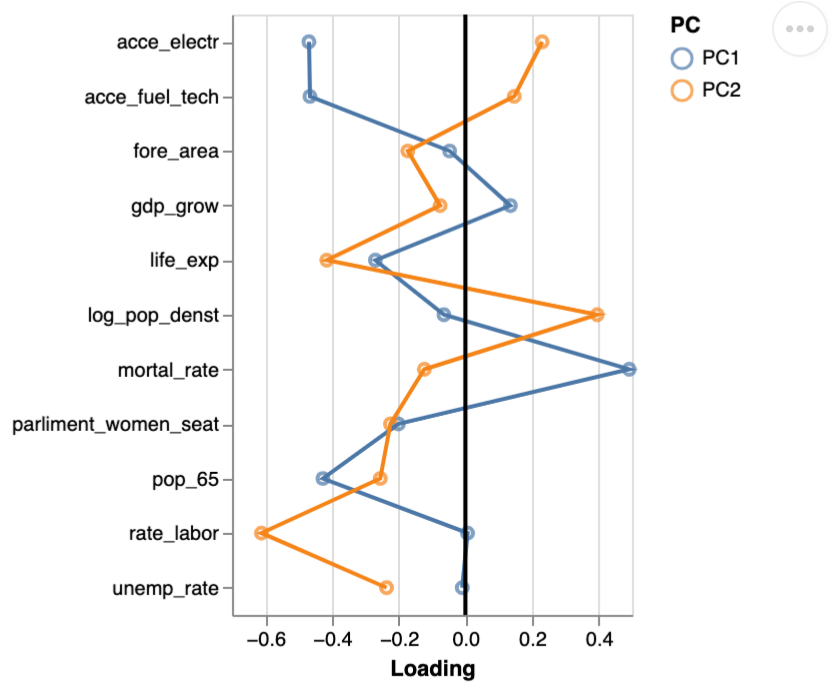
```
    id_vars = 'ESG Variables',
    var_name = 'PC',
    value_name = 'Loading'
)

# create base layer with encoding
base = alt.Chart(loading_plot_df).encode(
    y = alt.X('ESG Variables', title = ''),
    x = 'Loading',
    color = 'PC'
)

# store horizontal line at zero
rule = alt.Chart(pd.DataFrame({'Loading': 0}, index = [0])).mark_rule().encode(x = 'Loading', size = alt.value(2))

# layer points + lines + rule to construct loading plot
pca_loading_plot = base.mark_point() + base.mark_line() + rule
pca_loading_plot.properties(width = 200, height = 300)
```

Out[17]:



Some thoughts and analysis of the loading plots:

- Access to electricity and fuel, and motality rate have a strong influence on PC1. They are negative correlated, PC1 explain a higher access to electricity reflects a lower mortality rate (negative correlation between them)
- life expectancy and labor rate have a strong influence on PC2

In [18]:

```
# project pcdata onto first two components; store as data frame
projected_data = pd.DataFrame(pca.fit_transform(pcdata)).iloc[:, 0:2].rename(
    columns = {0: 'PC1', 1: 'PC2'}
)

# add index and reset
projected_data.index = pcdata.index
projected_data = projected_data.reset_index()

# print first four rows
projected_data.head()
```

Out[18]:

	Country Name	Country Code	Year	PC1	PC2
0	Afghanistan	AFG	2019	1.307875	0.921758
1	Afghanistan	AFG	2020	1.118395	1.134510
2	Albania	ALB	2019	-1.419153	-0.674074
3	Albania	ALB	2020	-1.598227	-0.655313
4	Algeria	DZA	2019	-0.687696	0.779908

In [19]:

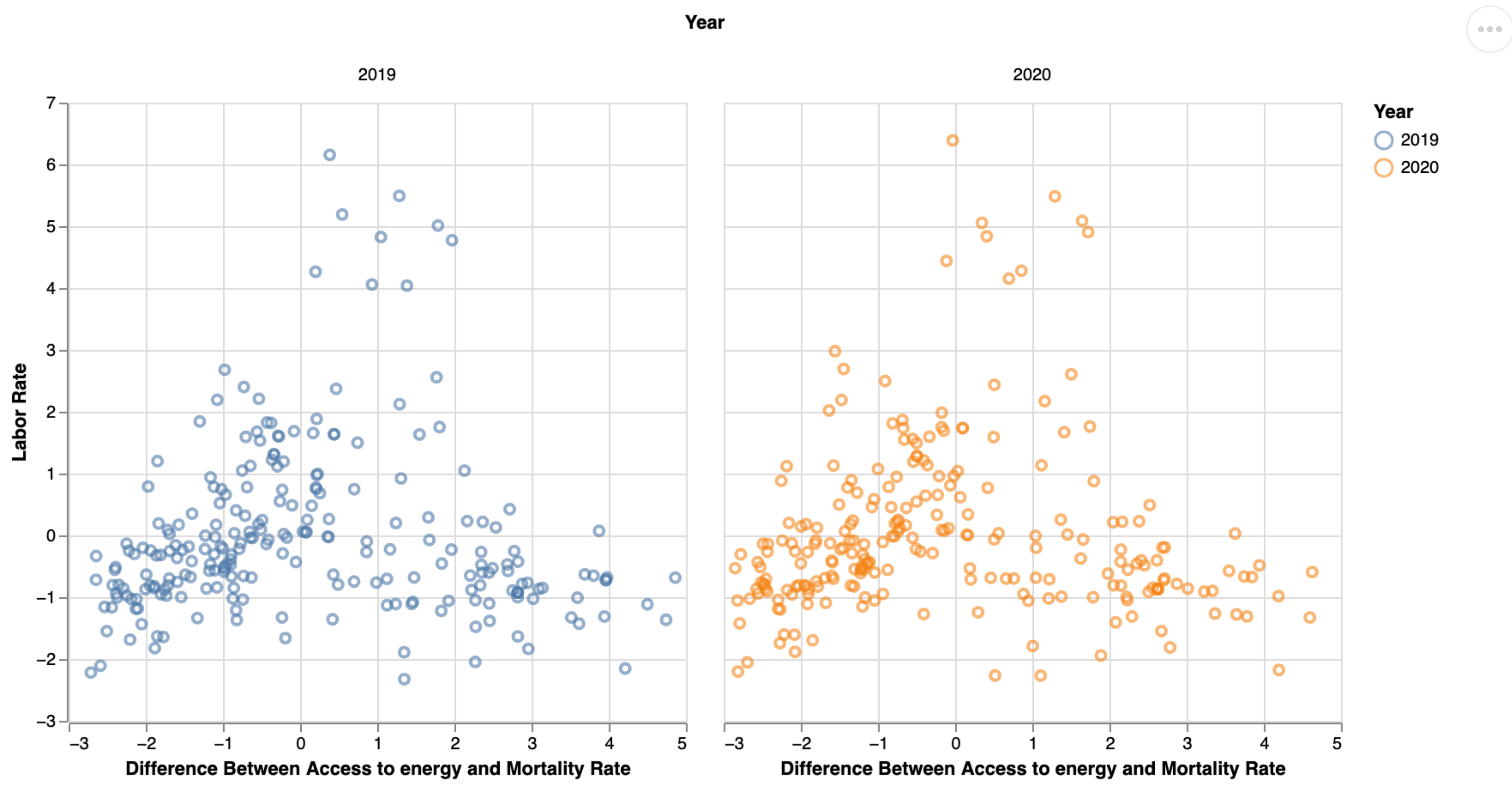
```
# Construct a scatterplot of PC1 and PC2 by Yea indicator

# base chart
base = alt.Chart(projected_data)

# data scatter
scatter = base.mark_point(opacity = 0.6).encode(
    x = alt.X('PC1:Q', title = 'Difference Between Access to energy and Mortality Rate'),
    y = alt.Y('PC2:Q', title = 'Labor Rate'),
    color = alt.Color('Year:N', title = 'Year')
)

# show
scatter.properties(width = 360, height = 360).facet(column = 'Year')
```

Out[19]:



No significant change of patterns is observed.

In [20]:

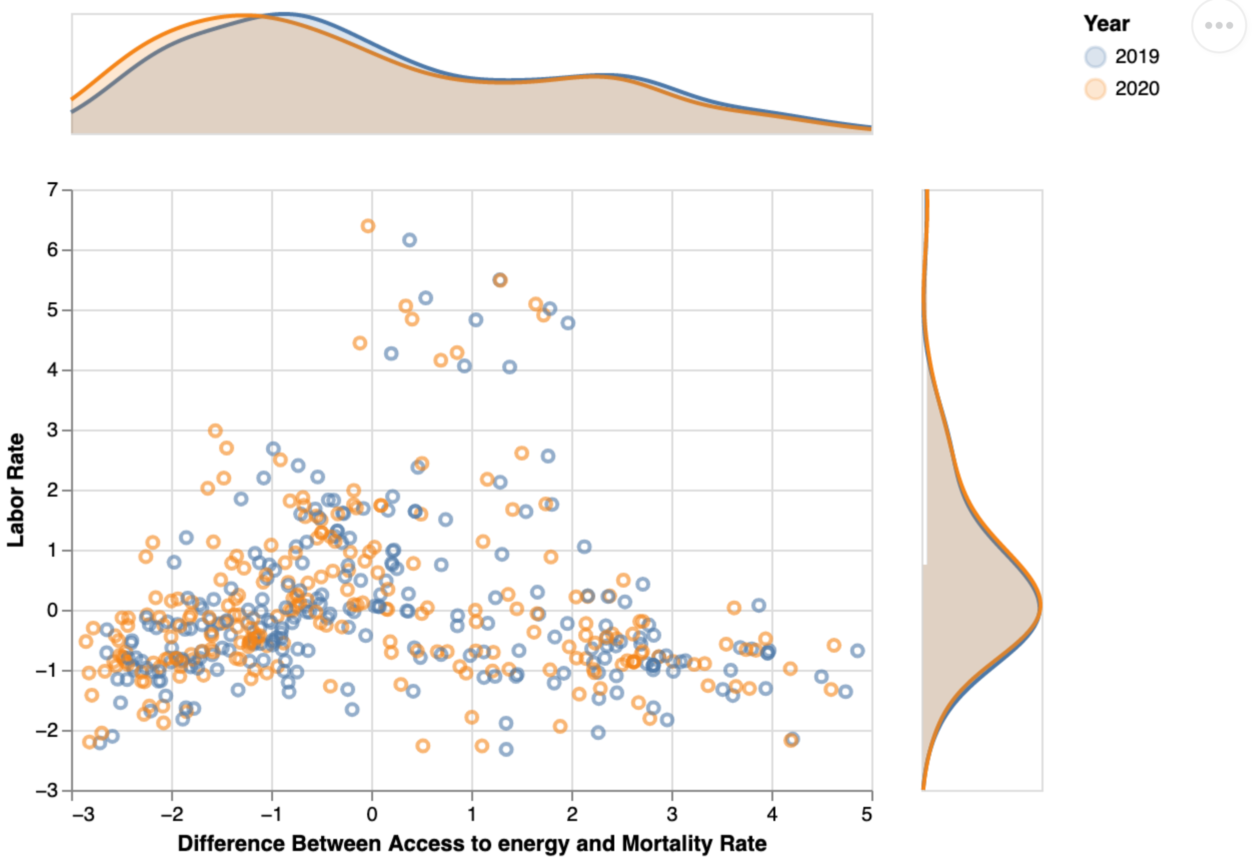
```
# construct upper panel (kdes for pc1)
top_panel = base.transform_density(
    density = 'PC1',
    groupby = ['Year'],
    as_ = ['PC1', 'Estimated density'],
    bandwidth = 0.5,
    extent = [-3, 5],
    steps = 1000
).mark_line(order = False).encode(
    x = alt.X('PC1:Q',
              title = '',
              axis = None),
    y = alt.Y('Estimated density:Q',
              title = '',
```

```
                  axis = None),
    color = alt.Color('Year:N', title = 'Year')
).properties(height = 60)

# construct side panel (kdes for pc2)
side_panel = base.transform_density(
    density = 'PC2',
    groupby = ['Year'],
    as_ = ['PC2', 'Estimated density'],
    bandwidth = 0.5,
    extent = [-3, 5],
    steps = 1000
).mark_line(order = False).encode(
    y = alt.Y('PC2:Q',
              title = '',
              axis = None),
    x = alt.X('Estimated density:Q',
              title = '',
              axis = None),
    color = alt.Color('Year:N', title = 'Year')
).properties(width = 60)

(top_panel + top_panel.mark_area(order = False, opacity = 0.2)) & (scatter | (side_panel + side_panel.mark_area(order = False, opacity = 0.2)))
```

Out[20]:



Is there any changes of variables before and after COVID that affect sustainability?
NO.

4. Sustainability Score Evaluation

In [21]:
```
data2019 = data[data.Year == '2019']
data2020 = data[data.Year == '2020']
```

- Variables evaluate in ascending rank: 'acce_fuel_tech', 'acce_electr', 'fore_area', 'gdp_grow', 'life_exp', 'parliment_women_seat', 'rate_labor', 'log_pop_denst'
- Variables evaluate in descending rank: 'mortal_rate', 'pop_65', 'unemp_rate'
- Higher score sum reflects a better sustainability

In [22]:
```
# 2019 score
a_score_labels = [1, 2, 3, 4, 5]
d_score_labels = [5, 4, 3, 2, 1]

conditions1 = [
    (data2019['acce_fuel_tech'] <= 20),
    (data2019['acce_fuel_tech'] > 20) & (data2019['acce_fuel_tech'] <= 40),
    (data2019['acce_fuel_tech'] > 40) & (data2019['acce_fuel_tech'] <= 60),
    (data2019['acce_fuel_tech'] > 60) & (data2019['acce_fuel_tech'] <= 80),
    (data2019['acce_fuel_tech'] > 80)
]

data2019['fuel_score'] = np.select(conditions1, a_score_labels)

conditions2 = [
    (data2019['acce_electr'] <= 20),
    (data2019['acce_electr'] > 20) & (data2019['acce_electr'] <= 40),
    (data2019['acce_electr'] > 40) & (data2019['acce_electr'] <= 60),
    (data2019['acce_electr'] > 60) & (data2019['acce_electr'] <= 80),
    (data2019['acce_electr'] > 80)
]
data2019['electr_score'] = np.select(conditions2, a_score_labels)



data2019['fore_score'] = pd.qcut(data2019.fore_area,
                              q = 5,
                              labels = a_score_labels)

data2019['gdp_score'] = pd.qcut(data2019.gdp_grow,
                              q = 5,
                              labels = a_score_labels)

data2019['life_score'] = pd.qcut(data2019.life_exp,
                              q = 5,
                              labels = a_score_labels)

data2019['mortal_score'] = pd.qcut(data2019.mortal_rate,
                              q = 5,
                              labels = d_score_labels)

data2019['pop65_score'] = pd.qcut(data2019.pop_65,
                              q = 5,
                              labels = d_score_labels)

data2019['par_w_score'] = pd.qcut(data2019.parliment_women_seat,
                              q = 5,
                              labels = a_score_labels)

data2019['labor_score'] = pd.qcut(data2019.rate_labor,
                              q = 5,
                              labels = a_score_labels)

data2019['unemp_score'] = pd.qcut(data2019.unemp_rate,
                              q = 5,
                              labels = d_score_labels)

data2019['popden_score'] = pd.qcut(data2019.log_pop_denst,
                              q = 5,
                              labels = a_score_labels)
```

```
score2019 = data2019.drop(columns = ['acce_fuel_tech', 'acce_electr',
       'fore_area', 'gdp_grow', 'life_exp', 'mortal_rate', 'pop_65',
       'parliment_women_seat', 'rate_labor', 'unemp_rate', 'log_pop_denst'])

score2019['score_sum'] = score2019.drop(columns = ['Country Name', 'Country Code', 'Year']).sum(
    axis = 1)

score2019.head()
```

Out[22]:

Indicator Name	Country Name	Country Code	Year	fuel_score	electr_score	fore_score	gdp_score	life_score	mortal_score	pop65_score	par_w_score	labor_score	unemp_score	popden_score	score_sum
0	Afghanistan	AFG	2019	2	5	1	4	2	1	5	4	1	1	3	29
2	Albania	ALB	2019	5	5	3	2	5	4	2	4	3	1	4	38
4	Algeria	DZA	2019	5	5	1	2	4	3	3	4	1	1	1	30
6	Andorra	AND	2019	5	5	3	2	1	5	5	5	1	5	4	41
8	Angola	AGO	2019	3	3	5	1	1	1	5	4	5	2	2	32

In [23]:

```
# 2020 score
a_score_labels = [1, 2, 3, 4, 5]
d_score_labels = [5, 4, 3, 2, 1]

conditions3 = [
    (data2020['acce_fuel_tech'] <= 20),
    (data2020['acce_fuel_tech'] > 20) & (data2020['acce_fuel_tech'] <= 40),
    (data2020['acce_fuel_tech'] > 40) & (data2020['acce_fuel_tech'] <= 60),
    (data2020['acce_fuel_tech'] > 60) & (data2020['acce_fuel_tech'] <= 80),
    (data2020['acce_fuel_tech'] > 80)
]

data2020['fuel_score'] = np.select(conditions3, a_score_labels)

conditions4 = [
    (data2020['acce_electr'] <= 20),
    (data2020['acce_electr'] > 20) & (data2020['acce_electr'] <= 40),
    (data2020['acce_electr'] > 40) & (data2020['acce_electr'] <= 60),
    (data2020['acce_electr'] > 60) & (data2020['acce_electr'] <= 80),
    (data2020['acce_electr'] > 80)
]
data2020['electr_score'] = np.select(conditions4, a_score_labels)



data2020['fore_score'] = pd.qcut(data2020['fore_area'],
                              q = 5,
                              labels = a_score_labels)

data2020['gdp_score'] = pd.qcut(data2020.gdp_grow,
                              q = 5,
                              labels = a_score_labels)

data2020['life_score'] = pd.qcut(data2020.life_exp,
                              q = 5,
                              labels = a_score_labels)

data2020['mortal_score'] = pd.qcut(data2020.mortal_rate,
                              q = 5,
                              labels = d_score_labels)

data2020['pop65_score'] = pd.qcut(data2020.pop_65,
                              q = 5,
                              labels = d_score_labels)

data2020['par_w_score'] = pd.qcut(data2020.parliment_women_seat,
                              q = 5,
                              labels = a_score_labels)

data2020['labor_score'] = pd.qcut(data2020.rate_labor,
                              q = 5,
                              labels = a_score_labels)

data2020['unemp_score'] = pd.qcut(data2020.unemp_rate,
                              q = 5,
                              labels = d_score_labels)

data2020['popden_score'] = pd.qcut(data2020.log_pop_denst,
                              q = 5,
                              labels = a_score_labels)


score2020 = data2020.drop(columns = ['acce_fuel_tech', 'acce_electr',
       'fore_area', 'gdp_grow', 'life_exp', 'mortal_rate', 'pop_65',
       'parliment_women_seat', 'rate_labor', 'unemp_rate', 'log_pop_denst'])

score2020['score_sum'] = score2020.drop(columns = ['Country Name', 'Country Code', 'Year']).sum(
    axis = 1)

score2020.head()
```

Out[23]:

Indicator Name	Country Name	Country Code	Year	fuel_score	electr_score	fore_score	gdp_score	life_score	mortal_score	pop65_score	par_w_score	labor_score	unemp_score	popden_score	score_sum
1	Afghanistan	AFG	2020	2	5	1	3	2	1	5	4	1	1	3	28
3	Albania	ALB	2020	5	5	3	3	5	4	2	4	3	1	4	39
5	Algeria	DZA	2020	5	5	1	2	4	3	3	4	1	1	1	30
7	Andorra	AND	2020	5	5	3	1	1	5	5	5	1	5	4	40
9	Angola	AGO	2020	3	3	5	2	1	1	5	4	5	2	2	33

In [24]:

```
score2019[score2019.score_sum == score2019.score_sum.max()]
# Vietnam
```

Out[24]:

Indicator Name	Country Name	Country Code	Year	fuel_score	electr_score	fore_score	gdp_score	life_score	mortal_score	pop65_score	par_w_score	labor_score	unemp_score	popden_score	score_sum
468	Vietnam	VNM	2019	4	5	4	5	4	3	3	4	5	5	5	47

In [25]:

```
score2020[score2020.score_sum == score2020.score_sum.max()]
# Vietnam
```

Out[25]:

Indicator Name	Country Name	Country Code	Year	fuel_score	electr_score	fore_score	gdp_score	life_score	mortal_score	pop65_score	par_w_score	labor_score	unemp_score	popden_score	score_sum
259	Luxembourg	LUX	2020	5	5	4	4	5	5	2	4	5	3	5	47
469	Vietnam	VNM	2020	4	5	4	5	4	3	3	4	5	5	5	47