PSTAT 100 FINAL PROJECT (4)

December 14, 2023

Group Members: Brianna Grissom, Kristen Wu, Victor Lam, and Tiffany Ngo

```
[2]: # importing packages
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import altair as alt
     from sklearn.decomposition import PCA
     import statsmodels.api as sm
     from scipy.stats import t

     # make sure to uncomment this
     alt.renderers.enable('mimetype')
```

```
[2]: RendererRegistry.enable('mimetype')
```

0.0.1 I. Data Description

The World Happiness Report is an annual report that measures the state of global happiness through surveys. Citizens of each country answer a variety of questions aimed to gauge the overall quality of life, such as their perception of corruption within their nation's government, their freedom to make life choices, and how supported they feel in times of crises. Then, the national average is taken from these results for each variable.

- The **observational units** are ***countries***.
- The **variables** are ***Life Ladder, Log GDP per capita, Social Support, etc***.
- One **observation** is made for a country during ***each year*** from 2005 to 2022.

The following table is a description of each variable measured in the survey:

Variable	Variable Description	Type
Country	Name of Country	Categorical
Year	Year	Numeric
Life Ladder	Overall Happiness Score or Subjective Well-Being	Numeric
Log GDP per capita	Log GDP per capita	Numeric

Variable	Variable Description	Type
Social Support	Do you have someone to support you during times of need or trouble?	Numeric
HLE At Birth	Health Life Expectancy at Birth	Numeric
Freedom to Make Life Choices	Are you content with the life choices you're able to make?	Numeric
Generosity	Have you donated money to charity in the past month?	Numeric
Corruption Perception	How corrupted do you think the government is?	Numeric
Positive Affect	Laugh, enjoyment, & doing interesting things	Numeric
Negative Affect	Worry, sadness, & anger	Numeric

Now, let's take a closer look at the data itself. First, we'll change some column names for readability. Then, we'll take a look at the first few rows of the dataset.

[3]:
```
whr_2023 = pd.read_csv('data/whr-2023.csv')
whr_2023 = whr_2023.rename(columns = {'Country name': 'Country', 'year':'Year',
                                      'Social support': 'Social Support',
                                      'Healthy life expectancy at birth':'HLE␣
 ↪At Birth',
                                      'Freedom to make life choices': 'Freedom␣
 ↪to Make Life Choices',
                                      'Perceptions of corruption' : 'Corruption␣
 ↪Perception',
                                      'Positive affect': 'Positive Effect',␣
 ↪'Negative affect':'Negative Effect'})
whr_2023.head()
```

[3]:
```
       Country  Year  Life Ladder  Log GDP per capita  Social Support  \
0  Afghanistan  2008        3.724               7.350           0.451
1  Afghanistan  2009        4.402               7.509           0.552
2  Afghanistan  2010        4.758               7.614           0.539
3  Afghanistan  2011        3.832               7.581           0.521
4  Afghanistan  2012        3.783               7.661           0.521

   HLE At Birth  Freedom to Make Life Choices  Generosity  \
0          50.5                         0.718       0.168
1          50.8                         0.679       0.191
2          51.1                         0.600       0.121
3          51.4                         0.496       0.164
4          51.7                         0.531       0.238
```

```
   Corruption Perception  Positive Effect  Negative Effect
0                  0.882            0.414            0.258
1                  0.850            0.481            0.237
2                  0.707            0.517            0.275
3                  0.731            0.480            0.267
4                  0.776            0.614            0.268
```

[4]:
```
variables = ['Life Ladder', 'Log GDP per capita', 'Social Support',
                'HLE At Birth', 'Freedom to Make Life Choices',
                'Generosity', 'Corruption Perception',
                'Positive Effect', 'Negative Effect']
```

[5]:
```
whr_2023.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2199 entries, 0 to 2198
Data columns (total 11 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Country                       2199 non-null   object
 1   Year                          2199 non-null   int64
 2   Life Ladder                   2199 non-null   float64
 3   Log GDP per capita            2179 non-null   float64
 4   Social Support                2186 non-null   float64
 5   HLE At Birth                  2145 non-null   float64
 6   Freedom to Make Life Choices  2166 non-null   float64
 7   Generosity                    2126 non-null   float64
 8   Corruption Perception         2083 non-null   float64
 9   Positive Effect               2175 non-null   float64
 10  Negative Effect               2183 non-null   float64
dtypes: float64(9), int64(1), object(1)
memory usage: 189.1+ KB
```

[6]:
```
sorted((whr_2023["Year"]).unique())
```

[6]:
```
[2005,
 2006,
 2007,
 2008,
 2009,
 2010,
 2011,
 2012,
 2013,
 2014,
 2015,
 2016,
 2017,
 2018,
 2019,
 2020,
 2021,
 2022]
```

By taking a closer look at the data, we can see that the data values were recorded from the year 2005 to 2022 and has a total of 2,199 observations and 11 columns.

0.0.2 II. Question of Interest

We are interested in finding the largest drop in happiness from year to year, measured by a variety of variables such as Life Ladder, Log GDP per capita, Social Support, and so on. We will be utilizing various methods such as Principal Component Analysis and regression analysis to achieve this.

0.0.3 III. Data Analysis

First, let's start check for any missing values in the dataset.

[7]:
```
whr_2023.isna().sum()  # amount of missing data
```

[7]:
```
Country                        0
Year                           0
Life Ladder                    0
Log GDP per capita            20
Social Support                13
HLE At Birth                  54
Freedom to Make Life Choices  33
Generosity                    73
Corruption Perception        116
Positive Effect               24
Negative Effect               16
dtype: int64
```

There are quite a bit of missing values for most categories. Let's take a look at the total amount of observations by year to see if any year is missing a disproportionate amount of observations in comparison to the others.

[8]:
```
total_obsvn_by_year = whr_2023.groupby
whr_2023 = whr_2023.dropna()
observations_per_year = whr_2023.groupby('Year').size().reset_index(name='Total␣
 ↪Observations')
print(observations_per_year)
```

```
    Year  Total Observations
0   2005                   1
1   2006                  74
```

```
2    2007                    92
3    2008                   101
4    2009                   106
5    2010                   112
6    2011                   132
7    2012                   121
8    2013                   124
9    2014                   129
10   2015                   128
11   2016                   125
12   2017                   132
13   2018                   129
14   2019                   129
15   2020                   105
16   2021                   114
17   2022                   104
```

Since we only have data from one country in 2005, we will drop the year entirely to avoid skewing our results.

[9]:
```
whr_2023 = whr_2023.loc[whr_2023['Year']!=2005,:]
```

Now, we will decide how to handle our missing values. Imputing these values with a mean score or simply '0' may skew the outcome more severely in comparison to just dropping these values. Thus, we will drop all rows with at least one missing value.

[10]:
```
whr_2023 = whr_2023.dropna()
```

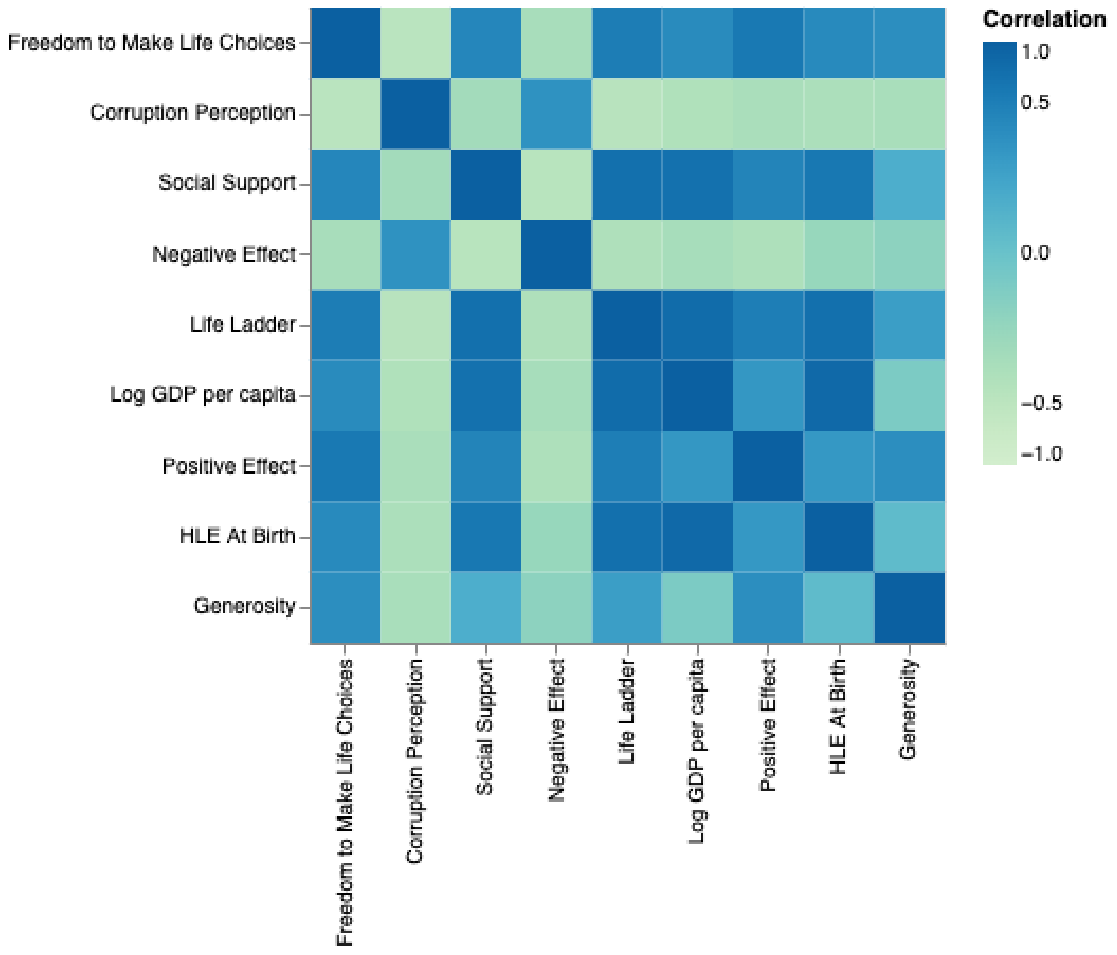
Now, we'll take a look at the heatmap to examine the correlation between the variables.

[11]:
```
x_mx = whr_2023.drop(columns = ['Country', 'Year'])
corr_mx = x_mx.corr()
corr_mx_long = corr_mx.reset_index().rename(
    columns = {'index': '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 = 'greenblue', # diverging␣
 ↪gradient
```

```
                                         domain = (-1, 1), # ensure white = 0
                                         type = 'sqrt'), # adjust gradient scale
                          legend = alt.Legend(tickCount = 5)) # add ticks to␣
 ↪colorbar at 0.5 for reference
).properties(width = 300, height = 300)
```

```
/opt/conda/lib/python3.11/site-packages/altair/utils/core.py:410: FutureWarning:
the convert_dtype parameter is deprecated and will be removed in a future
version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

[11]:

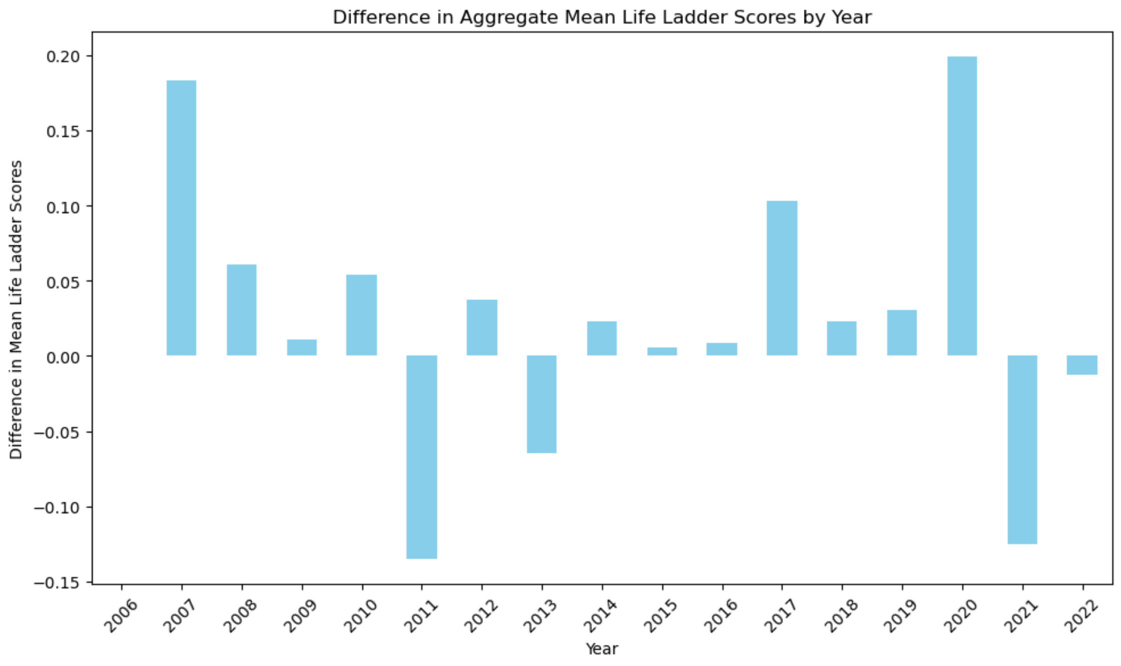


We can see that the variable `Freedom to Make Life Choices` is positively correlated with the variables `Social Support`, `Life Ladder`, `Log GDP`, `Positive Effect`, `HLE At Birth`, and `Generosity` and negatively correlated with the variables `Corruption Perception` and `Negative Effect`. Furthermore, we can see that `Corruption Perception` is positively correlated with `Negative Effect`.

Since we are trying to answer the question of which year had the biggest drop in happiness (where

Life Ladder is the overall happiness score), it would make sense to plot the differences in the aggregate mean Life Ladder scores from year to year (ie. the difference between 2007 and 2006). If the difference is POSITIVE, then the aggregate mean Life Ladder score from the latter year experienced an INCREASE in comparison to the year before it. If the difference is NEGATIVE, then the latter year experienced a DROP in the aggregate mean Life Ladder score in comparison to the prior year.

[12]:
```
mean_life_ladder_by_year = whr_2023.groupby('Year')['Life Ladder'].mean()

# differences in mean Life Ladder scores from one year to the next
life_ladder_diff = mean_life_ladder_by_year.diff()

# differences in mean Life Ladder scores over time
plt.figure(figsize=(10, 6))
life_ladder_diff.plot(kind='bar', color='skyblue')
plt.title('Difference in Aggregate Mean Life Ladder Scores by Year')
plt.xlabel('Year')
plt.ylabel('Difference in Mean Life Ladder Scores')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

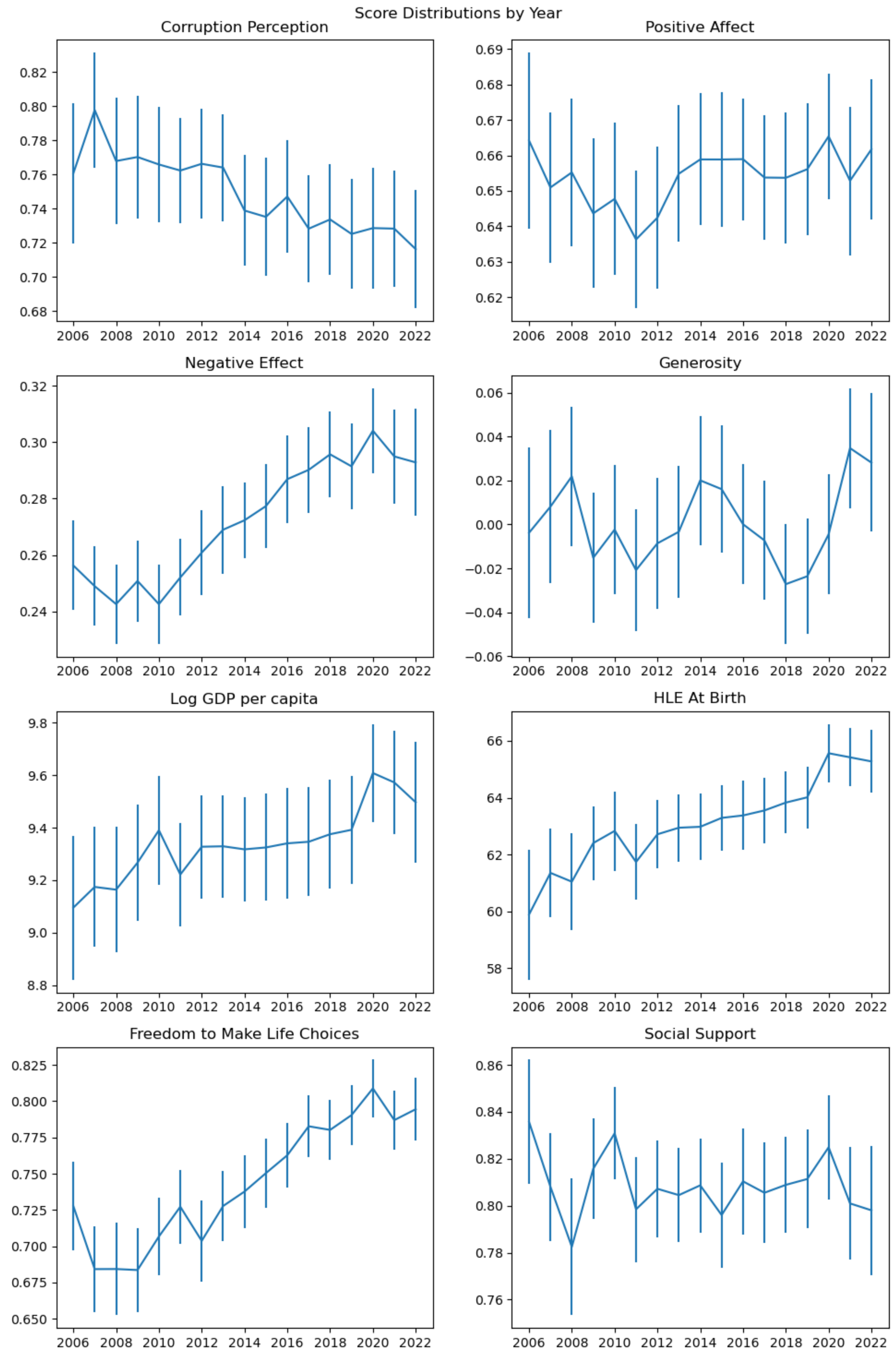


The biggest drops in Life Ladder occurs from 2010-2011 and 2020-2021.

[13]:
```
fig, axes = plt.subplots(4, 2, figsize = (10,15))
fig.suptitle('Confidence Intervals for Aggregate Mean Variables by Year')
```

```python
# Corruption Perception
a = whr_2023.groupby('Year')['Corruption Perception'].agg(['mean', 'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(a.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
axes[0,0].errorbar(x = a.index, y = a['mean'], yerr = y_errors)
axes[0,0].set_title('Corruption Perception')

# Positive Effect
b = whr_2023.groupby('Year')['Positive Effect'].agg(['mean', 'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(b.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
axes[0,1].errorbar(x = b.index, y = b['mean'], yerr = y_errors)
axes[0,1].set_title('Positive Affect')

## Negative Effect
c = whr_2023.groupby('Year')['Negative Effect'].agg(['mean', 'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(c.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
axes[1,0].errorbar(x = c.index, y = c['mean'], yerr = y_errors)
axes[1,0].set_title('Negative Effect')

## Generosity
d = whr_2023.groupby('Year')['Generosity'].agg(['mean', 'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(d.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
axes[1,1].errorbar(x = d.index, y = d['mean'], yerr = y_errors)
axes[1,1].set_title('Generosity')

## Log GDP per capita
e = whr_2023.groupby('Year')['Log GDP per capita'].agg(['mean', 'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(e.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
```

```
axes[2,0].errorbar(x = e.index, y = e['mean'], yerr = y_errors)
axes[2,0].set_title('Log GDP per capita')

## HLE At Birth
f = whr_2023.groupby('Year')['HLE At Birth'].agg(['mean', 'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(f.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
axes[2,1].errorbar(x = f.index, y = f['mean'], yerr = y_errors)
axes[2,1].set_title('HLE At Birth')


## Freedom to Make Life Choices
g = whr_2023.groupby('Year')['Freedom to Make Life Choices'].agg(['mean',␣
 ↪'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(g.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
axes[3,0].errorbar(x = g.index, y = g['mean'], yerr = y_errors)
axes[3,0].set_title('Freedom to Make Life Choices')

## Social Support
h = whr_2023.groupby('Year')['Social Support'].agg(['mean', 'std'])
y_errors = []
for yr in np.sort(whr_2023['Year'].unique()):
    year_df = whr_2023.loc[whr_2023['Year'] == yr, :]
    y_err = 1.96*(h.loc[yr, 'std'] / np.sqrt(len(year_df)))
    y_errors.append(y_err)
axes[3,1].errorbar(x = h.index, y = h['mean'], yerr = y_errors)
axes[3,1].set_title('Social Support')

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


Score Distributions by Year
Corruption Perception
Positive Affect
Negative Effect
Generosity
Log GDP per capita
HLE At Birth
Freedom to Make Life Choices
Social Support

We generated eight subplots displaying the aggregate mean and confidence intervals for the different variables across the years. This provides a visual representation the variability of the aggregate mean values for each predictor variable. The variances for each variable do not vary by much per year and `Generosity` has the most inconsistent pattern from year to year.

Do outliers impact our results? Let's plot the mean scores per year with outliers and without outliers.

[14]:
```
df = whr_2023.copy()

# Determine which values are outliers
outlier_dict = {}
# Loop through year
for yr in df['Year'].unique():
    outlier_dict[yr] = {}
    # Loop through the variables
    for col in variables:
        yr_df = df.loc[df['Year']==yr, col]
        IQR = np.quantile(a=yr_df, q=0.75) - np.quantile(a=yr_df, q=0.25)
        out_u = np.quantile(a=yr_df, q=0.75) + 1.5*IQR
        out_l = np.quantile(a=yr_df, q=0.25) - 1.5*IQR
        outliers = [i for i in yr_df if (i<=out_l) or (i>=out_u)]
        # Add the outlier value to the dictionary
        outlier_dict[yr][col] = outliers
```

[15]:
```
# Determine how many outliers each variable has
count_dict = {}
for col in variables:
    count_dict[col] = 0
for year in outlier_dict.keys():
    for col in variables:
        length = len(outlier_dict[year][col])
        count_dict[col] = count_dict[col] + length

pd.DataFrame({'Variable':count_dict.keys(), 'Outlier Count':count_dict.
 ↪values()}).sort_values(
    by='Outlier Count', ascending=False)
```

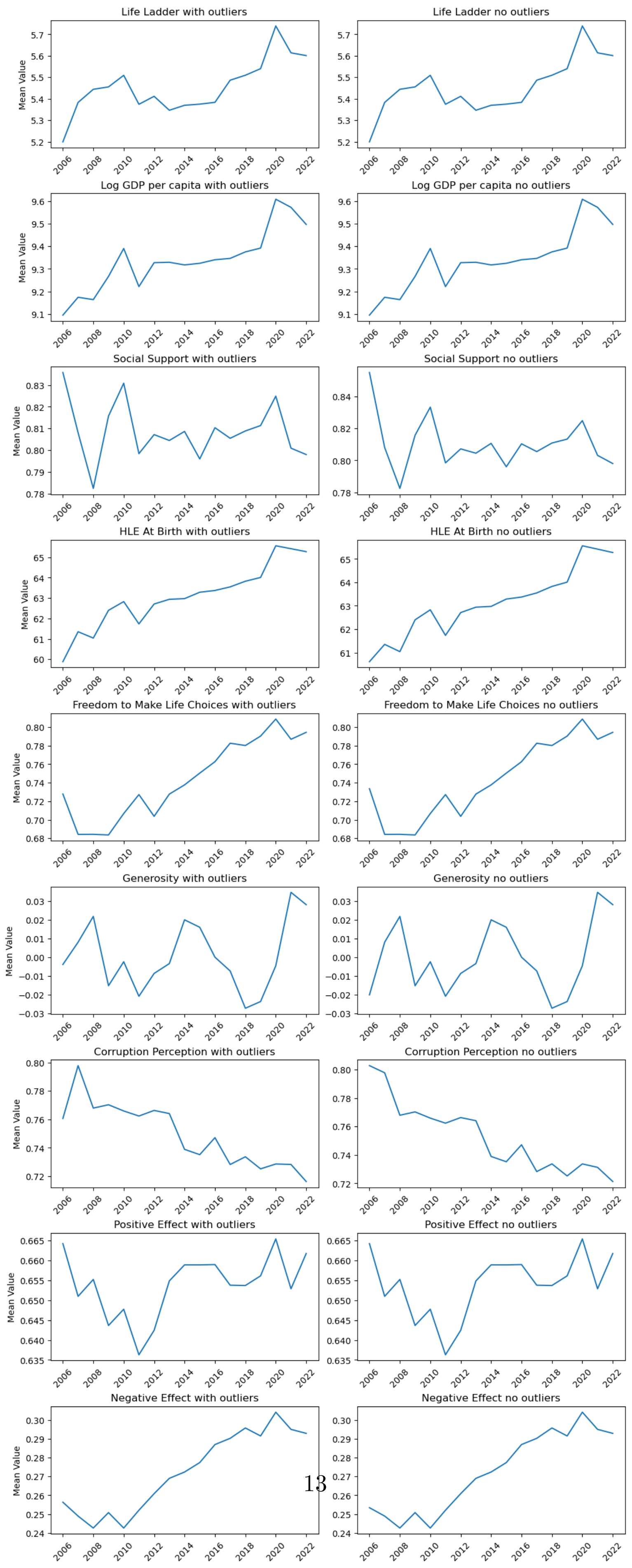
[15]:
```
                     Variable  Outlier Count
6        Corruption Perception            165
5                   Generosity             43
2               Social Support             38
8              Negative Effect             29
4  Freedom to Make Life Choices            23
3                 HLE At Birth             12
```

```
7                Positive Effect              8
0                    Life Ladder              5
1             Log GDP per capita              1
```

`Corruption Perception` has the largest amount of outliers. Will this impact the mean scores per year?

[16]:
```python
# Specify which values are outliers in the DataFrame
df2 = df.copy()
for col in variables:
    df2[f'{col} Outlier'] = None
for yr in df2['Year'].unique():
    yr_df = df2.loc[df2['Year']==yr, :]
    for col in variables:
        for idx in yr_df.index:
            value = df2.loc[idx, col]
            if value in outlier_dict[year][col]:
                df2.loc[idx, f'{col} Outlier'] = 'Yes'
            else:
                df2.loc[idx, f'{col} Outlier'] = 'No'
```

[17]:
```python
# Plot the means per year with and without outliers
fig, axes = plt.subplots(9, 2, figsize=(10,25), constrained_layout=True)
#fig.tight_layout()
labels = list(range(2006, 2023, 2))
rowidx = 0
for col in variables:
    no_outliers = df2.loc[df2[f'{col} Outlier'] == 'No', :]

    g_no_outliers = no_outliers.groupby('Year')[[col]].mean()
    g_with_outliers = df2.groupby('Year')[[col]].mean()

    axes[rowidx, 0].plot(g_with_outliers.index, g_with_outliers[col])
    axes[rowidx, 0].set_title(f'{col} with outliers')
    axes[rowidx, 0].set_xticks(list(range(2006, 2023, 2)), labels = labels,␣
 ↪rotation = 45)
    axes[rowidx, 0].set_ylabel('Mean Value')

    axes[rowidx, 1].plot(g_no_outliers.index, g_no_outliers[col])
    axes[rowidx, 1].set_title(f'{col} no outliers')
    axes[rowidx, 1].set_xticks(list(range(2006, 2023, 2)), labels = labels,␣
 ↪rotation = 45)

    rowidx = rowidx + 1
```


Life Ladder with outliers
Life Ladder no outliers
Log GDP per capita with outliers
Log GDP per capita no outliers
Social Support with outliers
Social Support no outliers
HLE At Birth with outliers
HLE At Birth no outliers
Freedom to Make Life Choices with outliers
Freedom to Make Life Choices no outliers
Generosity with outliers
Generosity no outliers
Corruption Perception with outliers
Corruption Perception no outliers
Positive Effect with outliers
Positive Effect no outliers
Negative Effect with outliers
Negative Effect no outliers
Mean Value
2006
2008
2010
2012
2014
2016
2018
2020
2022

Looking at the graphs, we can see that the outliers did not exude a significant change on the predictor variables. A small change is that the distribution for `Corruption Affect` no longer has a spike at 2007 when we remove outliers. Now, let's move on to our principal component analysis.

We will use PCA to determine changes in happiness throughout the years. First, let's explore what each principal component represents.

[18]:
```
X = whr_2023[whr_2023.columns[2:]]
pca = PCA(n_components=3)

# Standardize data
X = (X - X.mean()) / X.std()
pca.fit(X)

# Create DataFrame of the loadings by variable
pca_df = pd.DataFrame({'Variable': whr_2023.columns[2:], 'PC1 loadings': pca.
 ↪components_[0],
                       'PC2 loadings' : pca.components_[1], 'PC3 loadings':pca.
 ↪components_[2]})
pca_df
```

[18]:
```
                     Variable  PC1 loadings  PC2 loadings  PC3 loadings
0                 Life Ladder     -0.443662     -0.107574      0.076913
1            Log GDP per capita   -0.397550     -0.374878      0.138143
2              Social Support     -0.393562     -0.188921     -0.271400
3                HLE At Birth     -0.372270     -0.364808      0.287331
4  Freedom to Make Life Choices   -0.337158      0.338601      0.111930
5                  Generosity     -0.122047      0.568806      0.324577
6        Corruption Perception      0.276977     -0.270040     -0.269433
7             Positive Effect     -0.300836      0.386458     -0.198900
8             Negative Effect      0.236330     -0.136588      0.767301
```

[19]:
```
pcvars = pd.DataFrame(pca.explained_variance_ratio_)

# add component number as a new column
pcvars['Component'] = np.arange(1, 4)

# add cumulative variance explained as a new column
pcvars['Cumulative variance explained'] = pca.explained_variance_ratio_.cumsum()

pcvars = pcvars.rename(columns = {0:'Proportion of variance explained'})

pcvars = pcvars[['Component', 'Proportion of variance explained', 'Cumulative␣
 ↪variance explained']]
```

```
pcvars
```

[19]:

```
   Component  Proportion of variance explained  Cumulative variance explained
0          1                          0.467858                       0.467858
1          2                          0.168758                       0.636616
2          3                          0.104577                       0.741192
```

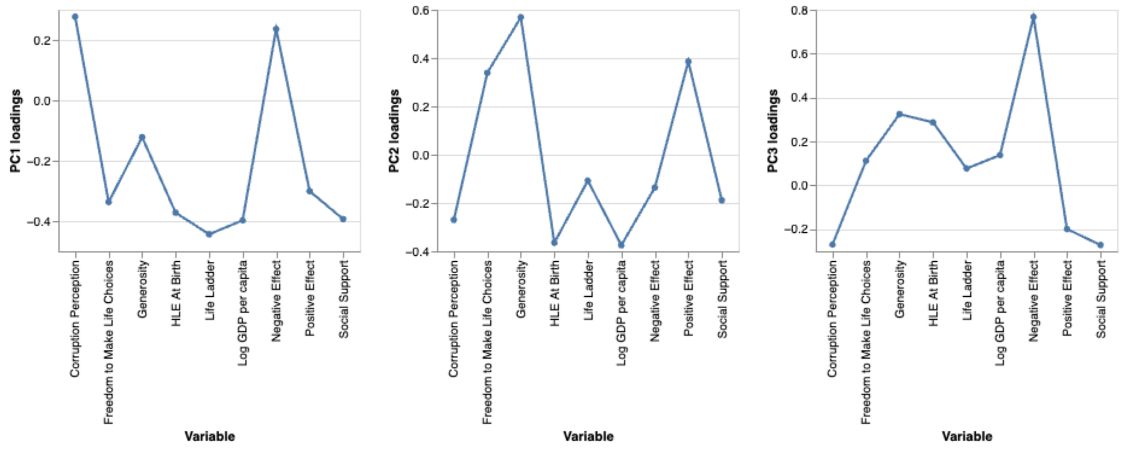
74% of the variance in this dataset is explained by the first 3 principal components.

[20]:

```
# Create plots of the loadings for each variable
pc1_plot = alt.Chart(pca_df).mark_line(point=True).encode(
            x=alt.X('Variable'), y=alt.Y('PC1 loadings')).properties(width=250,␣
 ↪height=200)

pc2_plot = alt.Chart(pca_df).mark_line(point=True).encode(
            x=alt.X('Variable'), y=alt.Y('PC2 loadings')).properties(width=250,␣
 ↪height=200)

pc3_plot = alt.Chart(pca_df).mark_line(point=True).encode(
            x=alt.X('Variable'), y=alt.Y('PC3 loadings')).properties(width=250,␣
 ↪height=200)

pc1_plot | pc2_plot | pc3_plot
```

```
/opt/conda/lib/python3.11/site-packages/altair/utils/core.py:410: FutureWarning:
the convert_dtype parameter is deprecated and will be removed in a future
version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

[20]:



The positive PC1 loading values for `Corruption` and `Negative Affect` and negative PC1 loading values for all other variables indicate that a higher PC1 score represents a higher frequency of the "bad" variables while a lower PC1 score represents a higher frequency of the "good" variables. The loadings for PC2 are not very helpful in attempting to determine happiness: PC2 has the strongest positive correlatation with `Generosity` and `Positive Affect`, but has the strongest negative correlation with `HLE at Birth` and `Log GDP per capita`. Thus, this principal component is not a good indicator of happiness; its loading values suggest that the scores will be hard to interpret with regard to happiness. We will not use PC2 to determine changes in happiness as a result. Lastly, PC3 is mostly a `Negative Affect` index. Thus, a high PC3 score corresponds to higher `Negative Affect`.

Next, we will compute the mean values for the PC1 and PC3 scores from 2007-2022 to see which year had the largest average INCREASE in the PC1 score (a high PC1 score is an indicator of a higher frequency of the "bad" variables and less of the "good" variables) and the largest average INCREASE in the PC3 score (a high PC3 score indicates high `Negative Affect`). To do so, we will find the mean PC1 and PC3 values grouped by year.

[21]:
```
# Get the PC1 and PC3 scores for the data
pca_scores = pca.transform(X)

# Add to DataFrame
df['PC1'] = pca_scores[:, 0]
df['PC3'] = pca_scores[:, 2]
```

[22]:
```
mean_std = df.groupby('Year')['PC1'].agg(['mean', 'std'])
mean_std
```

[22]:
```
          mean       std
Year
2006  0.256496  2.144205
2007  0.336692  1.880347
2008  0.329734  2.329809
2009  0.200744  1.890855
2010 -0.039290  2.030452
2011  0.249534  2.021176
2012  0.179738  2.046999
2013  0.123678  2.000886
2014  0.022453  2.051669
2015  0.022584  2.126533
2016 -0.009799  2.080850
2017 -0.092633  2.095466
2018 -0.092232  1.992078
2019 -0.185908  2.016058
2020 -0.506970  1.791947
2021 -0.330114  2.099733
2022 -0.343756  2.166980
```

[23]:
```
pd.DataFrame(mean_std['mean'].diff()).sort_values(by='mean', ascending=False)
```

[23]:
```
          mean
Year
2011  0.288824
2021  0.176856
2007  0.080196
2018  0.000401
2015  0.000131
2008 -0.006958
2022 -0.013642
2016 -0.032383
2013 -0.056060
2012 -0.069797
2017 -0.082834
2019 -0.093676
2014 -0.101224
2009 -0.128991
2010 -0.240034
2020 -0.321062
2006       NaN
```

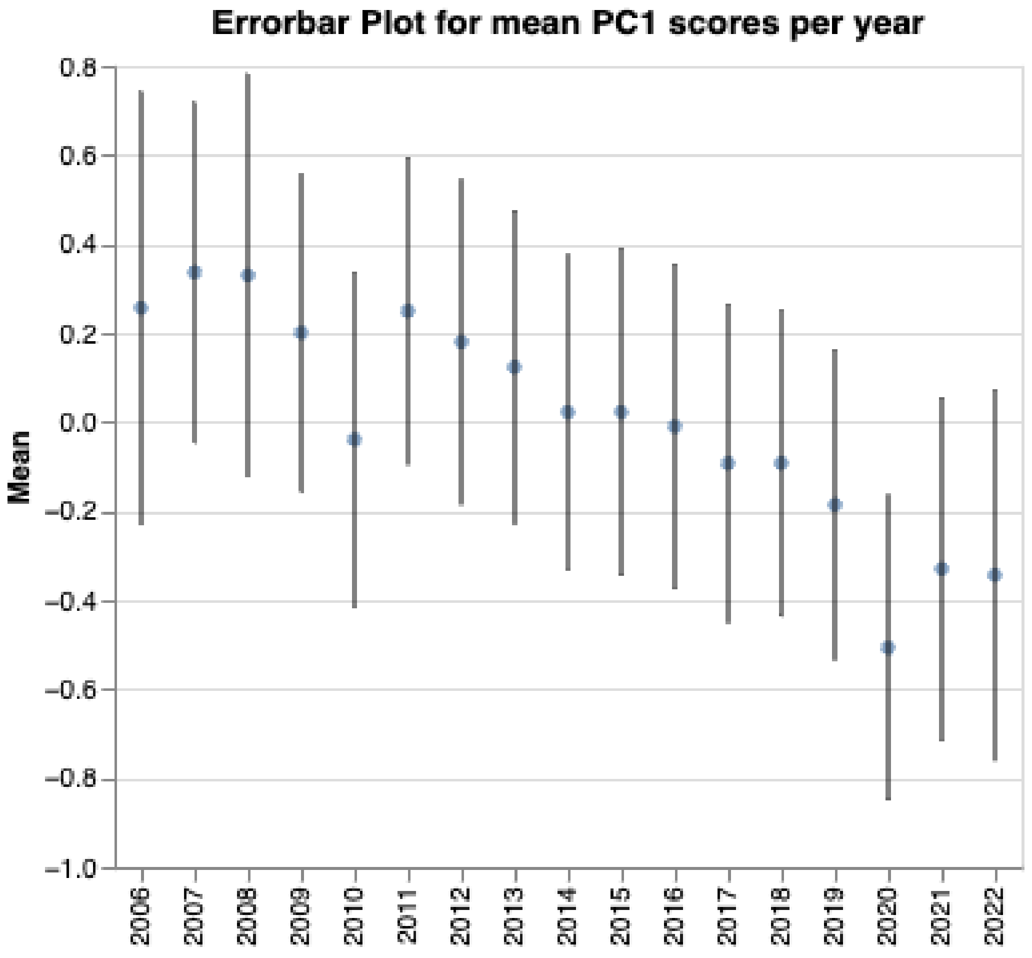
2010-2011 and 2020-2021 experience the the largest increase in PC1 score. This provides evidence for overall happiness decreasing from the former year to the latter.

Next, we will construct confidence intervals for the true mean PC1 score by year to determine if the shifts from 2010-2011 and 2020-2021 are significant.

[24]:
```
# Get the amount of records per year
counts = dict(df['Year'].value_counts())

# Construct confidence intervals
for year in counts.keys():
    # CI lower = mean - 1.96*(std/sqrt(n))
    ci_lower = mean_std.loc[year, 'mean'] - 1.96*(mean_std.loc[year, 'std'] /␣
 ↪np.sqrt(counts[year]))

    # CI upper = mean + 1.96*(std/sqrt(n))
    ci_upper = mean_std.loc[year, 'mean'] + 1.96*(mean_std.loc[year, 'std'] /␣
 ↪np.sqrt(counts[year]))
    mean_std.loc[year, 'ci_lower'] = ci_lower
    mean_std.loc[year, 'ci_upper'] = ci_upper

mean_std.reset_index(inplace=True)
# mean_std.rename(columns={'index':'yeareee'}, inplace=True)

mean_std['Year'] = mean_std['Year'].astype('str')

points = alt.Chart(mean_std).mark_circle().encode(
    x='Year:O', y='mean'
)
errorbars = alt.Chart(mean_std).mark_errorbar().encode(
    x=alt.X('Year:O', title=''), y=alt.Y('ci_lower', title='Mean'),
    y2='ci_upper'
)
(points + errorbars).properties(title='Errorbar Plot for mean PC1 scores per␣
 ↪year')
```

```
/opt/conda/lib/python3.11/site-packages/altair/utils/core.py:410: FutureWarning:
the convert_dtype parameter is deprecated and will be removed in a future
version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

[24]:



Again, we see an increase in the confidence interval of the true mean PC1 score from 2010 to 2011 and 2020 to 2021. However, since these confidence intervals overlap, we do not have strong evidence to say that the mean PC1 score, and hence unhappiness, has increased from 2010 to 2011 for the entire population.

We will now repeat the process for PC3.

[25]:
```
# Create groupby dataframe
mean_std = df.groupby('Year')['PC3'].agg(['mean', 'std'])
mean_std
```

[25]:
```
          mean       std
Year
2006 -0.453488  0.892845
2007 -0.417756  0.856360
2008 -0.364902  0.915035
2009 -0.353021  0.984738
2010 -0.381264  0.962729
2011 -0.293066  0.995625
2012 -0.189830  1.014099
2013 -0.095640  0.990685
2014  0.010954  0.883825
2015  0.105353  0.955095
2016  0.125870  0.922946
2017  0.219208  0.843137
2018  0.227691  0.879319
2019  0.218874  0.925032
2020  0.436002  0.747377
2021  0.473015  0.925027
2022  0.438496  0.968533
```

[26]:
```
# Calculate PC3 differences
pd.DataFrame(mean_std['mean'].diff()).sort_values(by='mean', ascending=False)
```

[26]:
```
          mean
Year
2020  0.217128
2014  0.106594
2012  0.103235
2015  0.094399
2013  0.094190
2017  0.093338
2011  0.088198
2008  0.052854
2021  0.037013
2007  0.035732
2016  0.020517
2009  0.011881
2018  0.008483
2019 -0.008816
2010 -0.028243
2022 -0.034519
```

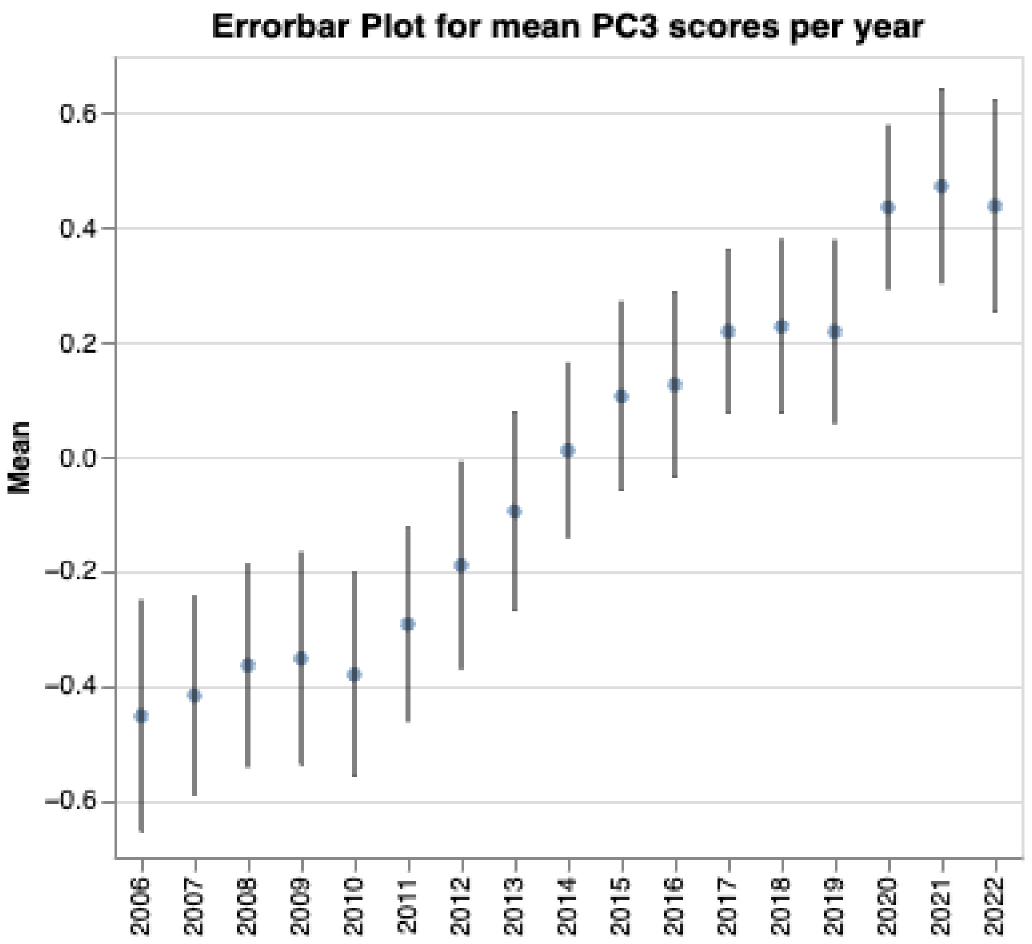
```
2006        NaN
```

2019-2020 experienced the largest increase in PC3 score by quite some margin. This suggests that Negative Affect went up by an abnormal amount during this time period. The second-largest increase in PC score was for 2013-2014, also suggesting that Negative Affect has increased.

[27]:
```
# Get the amount of records per year
counts = dict(df['Year'].value_counts())

# Construct confidence intervals
for year in counts.keys():
    # CI lower = mean - 1.96*(std/sqrt(n))
    ci_lower = mean_std.loc[year, 'mean'] - 1.96*(mean_std.loc[year, 'std'] /␣
 ↪np.sqrt(counts[year]))

    # CI upper = mean + 1.96*(std/sqrt(n))
    ci_upper = mean_std.loc[year, 'mean'] + 1.96*(mean_std.loc[year, 'std'] /␣
 ↪np.sqrt(counts[year]))
    mean_std.loc[year, 'ci_lower'] = ci_lower
    mean_std.loc[year, 'ci_upper'] = ci_upper

mean_std.reset_index(inplace=True)
# mean_std.rename(columns={'index':'yeareee'}, inplace=True)

mean_std['Year'] = mean_std['Year'].astype('str')

points = alt.Chart(mean_std).mark_circle().encode(
    x='Year:O', y='mean'
)
errorbars = alt.Chart(mean_std).mark_errorbar().encode(
    x=alt.X('Year:O', title=''), y=alt.Y('ci_lower', title='Mean'),
    y2='ci_upper'
)
(points + errorbars).properties(title='Errorbar Plot for mean PC3 scores per␣
 ↪year')
```

```
/opt/conda/lib/python3.11/site-packages/altair/utils/core.py:410: FutureWarning:
the convert_dtype parameter is deprecated and will be removed in a future
version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
  col = df[col_name].apply(to_list_if_array, convert_dtype=False)
```

[27]:

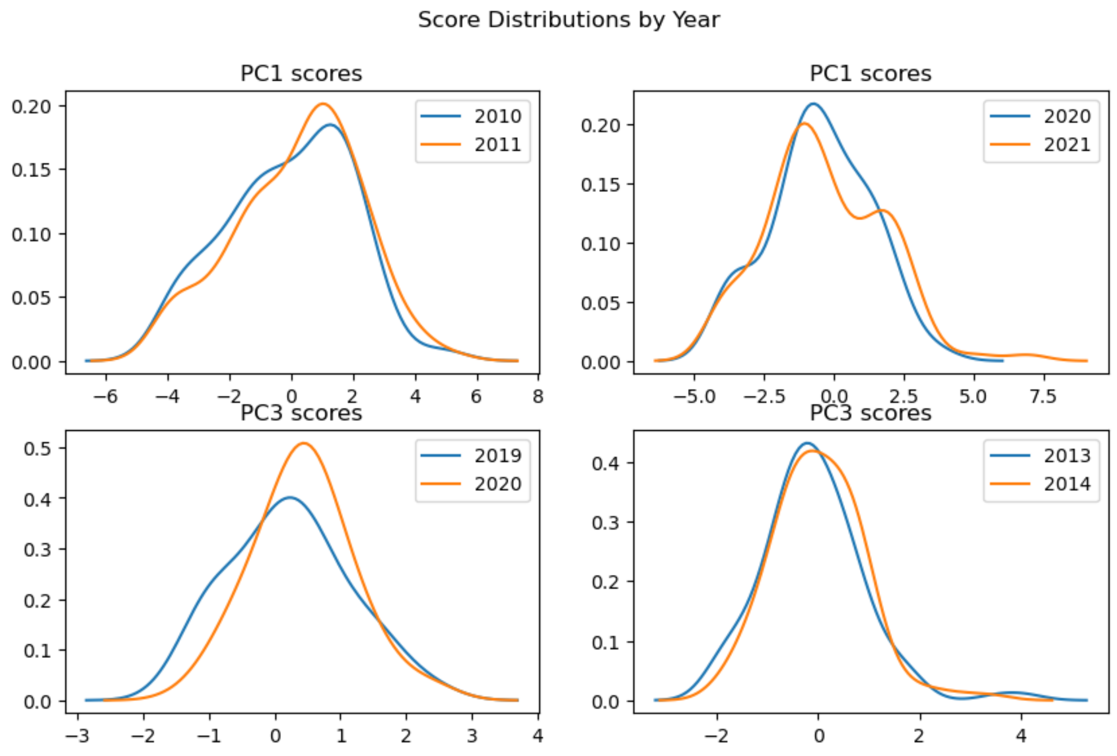


Even though there is an increase in the confidence interval for the true PC3 score from 2019 to 2020 and 2013-2014, the overlapping of the confidence intervals from one year to the next does not provide significant evidence to demonstrate that the PC3 score has increased for the whole population.

Next, let's examine the distributions of the PC1 scores for 2010-2011 and 2020-2021, and the distributions of the PC3 scores for 2019-2020 and 2013-2014.

[28]:
```
fig, axes = plt.subplots(2, 2, figsize=(10, 6))
fig.suptitle('Score Distributions by Year')
for year in [2010, 2011]:
    kde = sm.nonparametric.KDEUnivariate(df.loc[df['Year']==year, 'PC1'])
    kde.fit(kernel='gau', fft=False, bw=0.7)
    axes[0,0].plot(kde.support, kde.density)
axes[0,0].legend(['2010', '2011'])
axes[0,0].set_title('PC1 scores')
axes[0,0].set_ylabel('Density')

for year in [2020, 2021]:
    kde = sm.nonparametric.KDEUnivariate(df.loc[df['Year']==year, 'PC1'])
    kde.fit(kernel='gau', fft=False, bw=0.7)
    axes[0,1].plot(kde.support, kde.density)
axes[0,1].legend(['2020', '2021'])
axes[0,1].set_title('PC1 scores')

for year in [2019, 2020]:
    kde = sm.nonparametric.KDEUnivariate(df.loc[df['Year']==year, 'PC3'])
    kde.fit(kernel='gau', fft=False, bw=0.4)
    axes[1,0].plot(kde.support, kde.density)
axes[1,0].legend(['2019', '2020'])
axes[1,0].set_title('PC3 scores')
axes[1,0].set_ylabel('Density')

for year in [2013, 2014]:
    kde = sm.nonparametric.KDEUnivariate(df.loc[df['Year']==year, 'PC3'])
    kde.fit(kernel='gau', fft=False, bw=0.4)
    axes[1,1].plot(kde.support, kde.density)
axes[1,1].legend(['2013', '2014'])
axes[1,1].set_title('PC3 scores')
```

```
[28]: Text(0.5, 1.0, 'PC3 scores')
```



For the 2010 and 2011 PC1 distributions, the density at the left tail has decreased and the density at the right tail has increased, further suggesting that the PC1 distribution has increased from 2010 to 2011. The 2020 and 2021 curves experience a less obvious shift, suggesting that the increase may have been solely because of the density at the right tail increasing. The 2019 and 2020 curves have the most signficant shift of the four plots; the entire curve gets translated to the right from the former year to the latter. The 2013 and 2014 also experience a shift, but to a lesser extent.

Which variables contribute the most to the subjective happiness score? We will address this question by using linear regression. Since the `Life Ladder` variable seems to be a summary metric for happiness in general, we will consider this as a function of all of the other factors. It also makes intuitive sense that the other more specific variables have an impact on one's self-rated happiness. Hence, we will attempt to predict `Life Ladder` using all of the other happiness variables. The variables with associated parameters the farthest from 0 will indicate that those variables have the greatest impact on happiness. We will select the top three determinants of happiness to factor into our final decision of which year experienced the greatest drop in happiness.

[29]:
```
# Define predictors and response
X = df[df.columns[3:11]]
y = df['Life Ladder']
X = sm.add_constant(X)
model = sm.OLS(y, X)

# Fit OLS model
results = model.fit()

# Extract standard errors of the parameters
SEs = np.sqrt(results.cov_params().values.diagonal())

# Calculate the t-value for the confidence intervals
t_value = t.ppf(0.975, 1948)
```

[30]:
```
# Create DataFrame of the results and create 95% confidence intervals to␣
↪estimate the true parameter values
parameter_df = pd.DataFrame(results.params)
parameter_df.rename(columns={0:'parameter estimate'}, inplace=True)
parameter_df['standard error'] = SEs
parameter_df['ci_lower'] = parameter_df['parameter estimate'] -␣
↪t_value*parameter_df['standard error']
parameter_df['ci_upper'] = parameter_df['parameter estimate'] +␣
↪t_value*parameter_df['standard error']
parameter_df
```

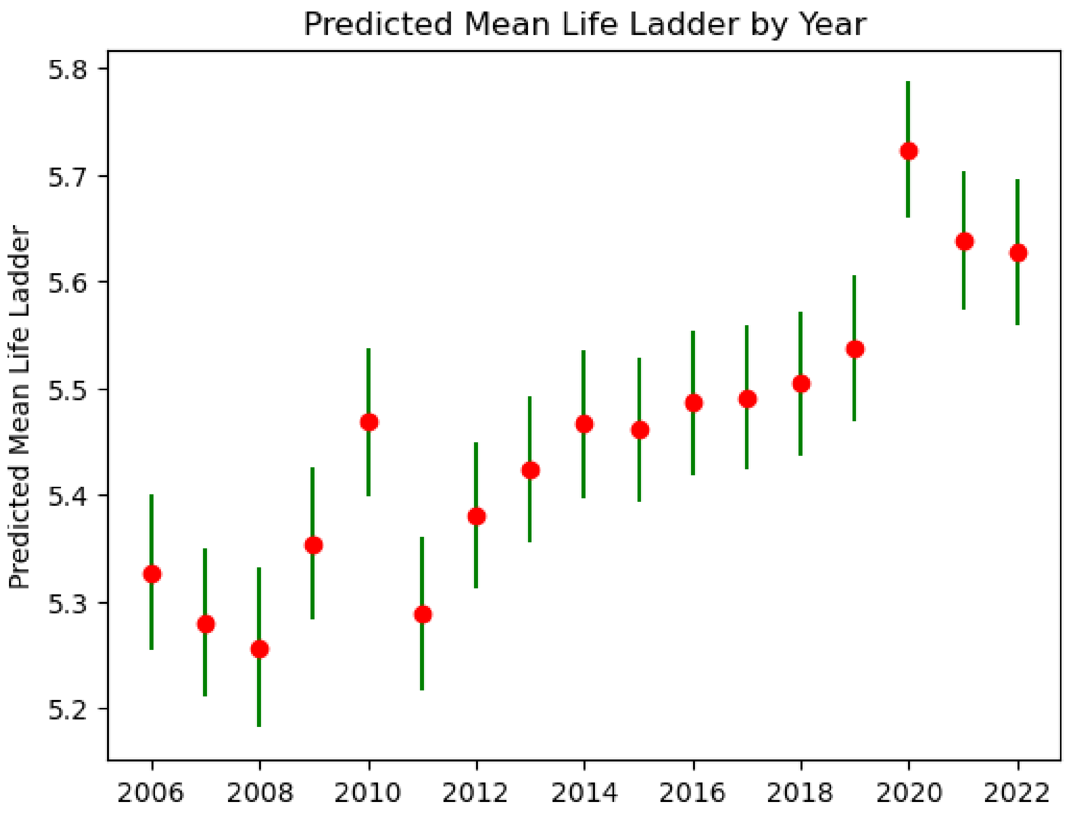
[30]:
```
                              parameter estimate  standard error  ci_lower  \
const                                  -2.734818        0.175777 -3.079549
Log GDP per capita                      0.397444        0.021753  0.354782
Social Support                          1.819139        0.161129  1.503136
HLE At Birth                            0.027569        0.003181  0.021331
Freedom to Make Life Choices            0.378036        0.120646  0.141427
```

```
Generosity                                  0.349565        0.083160  0.186473
Corruption Perception                      -0.696660        0.080673 -0.854873
Positive Effect                             2.312883        0.151023  2.016700
Negative Effect                            -0.015323        0.168823 -0.346416

                               ci_upper
const                         -2.390088
Log GDP per capita             0.440105
Social Support                 2.135142
HLE At Birth                   0.033808
Freedom to Make Life Choices   0.614645
Generosity                     0.512658
Corruption Perception         -0.538446
Positive Effect                2.609066
Negative Effect                0.315770
```

Based on the parameter estimates, `Positive Affect` has the greatest impact on `Life Ladder` (subjective happiness score) because the parameter corresponding to this variable is the greatest in absolute value. The regression suggests that a one unit increase in `Positive Affect` causes a 2 to 2.6 unit increase in `Life Ladder` when keeping all other variables constant. `Social Support` has the next largest impact on `Life Ladder`, causing a 1.5 to 2.1 unit increase in `Life Ladder` when keeping all other components constant. `Corruption Perception` has the third largest impact on `Life Ladder`, suggesting that a one unit increase results in a 0.53 to 0.85 decrease in `Life Ladder` when keeping all other variables constant. The very small parameter estimates for `HLE At Birth` and `Negative Affect` indicate that these variables have little, if any, impact on `Life Ladder`. Thus, these variables will be placed much lower on the priority list for determining which year has experienced the greatest drop in happiness.

Let's compare the predicted `Life Ladder` by year.

[31]:
```python
# Calculate predictions and create dataframe
predicted_life_ladder = results.get_prediction(X)
prediction_df = predicted_life_ladder.summary_frame()
# Specify the year for each prediction
prediction_df['Year'] = df['Year']

# Groupby year and calculate confidence intervals
g = prediction_df.groupby('Year')[['mean', 'mean_ci_lower', 'mean_ci_upper']].
  ↪mean()
g.reset_index(inplace=True)
g['yerr'] = g['mean'] - g['mean_ci_lower']

# Plot
plt.errorbar(x=g['Year'], y=g['mean'], yerr=g['yerr'], fmt='o', ecolor='green',␣
  ↪color='red')
plt.title('Predicted Mean Life Ladder by Year')
plt.ylabel('Predicted Mean Life Ladder')
plt.show()
```



The largest drop in predicted `Life Ladder` occured from 2010 to 2011. These are the only consecutive years in which the confidence intervals do not overlap for a drop in `Life Ladder`. This shows that `Life Ladder` or subjective happiness score has decreased throughout all countries from 2010 to 2011.

0.0.4 IV. Summary of Findings

After conducting our analysis, we found that 2010-2011 had the biggest overall drop in happiness. We attempted to determine which variables were most relevant in measuring a country's overall happiness. We also attempted to account for variability by utilizing confidence intervals to visualize the variability of each predictor. In doing so, we found that the predicted `Life Ladder` confidence interval was the only one of significance. Although, in our principal component analysis, we did not find reliable evidence that the mean PC1 score increased from 2010-2011, these years had the largest jump in PC1 in comparison to all of the other years by a significant margin. Despite the variable `Freedom to Make Life Choices` increasing, alongside `Corruption Perception` decreasing, the other predictor variables increased. We can also see that the predictor variable `Positive Affect` experienced a large drop from 2010 to 2011. This is significant because `Positive Affect` had the largest impact on subjective happiness scores through the regression. Similarly, the predictor variable `Social Support` had the second-largest drop for the years 2010-2011. This is important because the variable `Social Support` had the second-largest impact on subjective happiness scores.

Additionally, based on our principal component analysis we found that the years 2020-2021 also experienced significant drops in happiness. This drop can be attributed to the COVID-19 pandemic, causing job losses, a decrease in productivity, and an uncertain economic and social future.

It is also important to note that major influential countries are missing from this dataset. Let's grab the countries with the most null values and see if any countries have disproportionately missing values compared to others.

[32]:
```
whr_2023_no_null = pd.read_csv('data/whr-2023.csv')
whr_2023_no_null = whr_2023_no_null.rename(columns = {'Country name':␣
  ↪'Country', 'year':'Year',
                                      'Social support': 'Social Support',
                                      'Healthy life expectancy at birth':'HLE␣
  ↪At Birth',
                                      'Freedom to make life choices': 'Freedom␣
  ↪to Make Life Choices',
                                      'Perceptions of corruption' : 'Corruption␣
  ↪Perception',
                                      'Positive affect': 'Positive Effect',␣
  ↪'Negative affect':'Negative Effect'})
rows_w_null_any_column = whr_2023_no_null[whr_2023_no_null.isnull().any(axis=1)]
rows_w_null_any_column['Country'].value_counts().nlargest(10)
```

[32]:
```
Country
China                        16
Kosovo                       16
Saudi Arabia                 15
Jordan                       14
United Arab Emirates         12
Hong Kong S.A.R. of China    12
Taiwan Province of China     12
State of Palestine           11
Turkmenistan                 10
Kuwait                        8
Name: count, dtype: int64
```

Since China is missing at least one value in each row for every year, China is not included in our final dataset. The majority of data from other influential countries such as Saudi Arabia, the United Arab Emirates, Hong Kong S.A.R. of China, and Taiwan Province of China were also dropped from the dataset due to having a high number of null values. We feel the lack of data from these countries could potentially alter the results of our analysis because of their high populations. For example, China alone has a population of 1.4 billion people. This must be taken into consideration along with our results.