exercise_plotting_part2

October 10, 2025

1 Wealth and Democracy: Plotting, Part 2

In these expercises, we will be working with country-year level data from the World Development Indicators (published by the World Bank) to understand the relationship between wealth and democratic institutions.

This WDI data includes both countries' GDP per capita (a measure of wealth) and Polity IV scores (a measure of how democratic a country is – countries with higher scores are liberal democracies countries with low scores are autocratic).

Note by "liberal," we mean reflecting the values of traditional liberalism with a small-`l`, not Liberal in the sense of being associated with the US Democratic party.

1.0.1 Exercise 1

Load the World Development Indicator data from here.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns

world_data = pd.read_csv(
    "https://raw.githubusercontent.com/nickeubank/practicaldatascience/master/
 ↪Example_Data/world-small.csv"
)
```

1.0.2 Exercise 2

Let's begin analyzing this data by estimating a simple linear model ("ordinary least squares") of the relationship between democracy scores (`polityIV`) and GDP per capita (`gdppcap08`). Polity Scores are the outcome we care about, so make it the dependent variable in your model. We will do so using the `statsmodel` package, which we'll discuss in detail later is this course. For the moment, just use this code:

```
import statsmodels.formula.api as smf
results = smf.ols('polityIV ~ gdppcap08',
                  data=wdi).fit()
print(results.summary())
```

[2]:
```
results = smf.ols("polityIV ~ gdppcap08", data=world_data).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                polityIV   R-squared:                       0.047
Model:                            OLS   Adj. R-squared:                  0.040
Method:                 Least Squares   F-statistic:                     6.981
Date:                Fri, 10 Oct 2025   Prob (F-statistic):            0.00915
Time:                        15:49:46   Log-Likelihood:                -475.14
No. Observations:                 145   AIC:                             954.3
Df Residuals:                     143   BIC:                             960.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     12.1354      0.721     16.841      0.000      10.711      13.560
gdppcap08   9.602e-05   3.63e-05      2.642      0.009    2.42e-05       0.000
==============================================================================
Omnibus:                       17.820   Durbin-Watson:                   1.939
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               20.393
Skew:                          -0.887   Prob(JB):                     3.73e-05
Kurtosis:                       2.524   Cond. No.                     2.67e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.67e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

1.0.3 Exercise 3

Based on the results of this analysis, what would you conclude about the relationship between `gdppcap08` and `polityIV`?

Write down your conclusions.

- The coefficient for gdppcap08 is 0.000096, which means that for every 1 unit (dollar, here) increase there is a 0.000096 increase in the democracy score. For easier interpretation, for every $10,000 increase, there is approximately a unit increase in the democracy score.
- The relationship is positive and statistically significant ($p = 0.009$), suggesting that wealthier countries tend to have higher levels of democracy.
- However, the R-squared value of 0.047 shows that GDP per capita alone explains only about 5% of the variation in democracy levels, which means that many other factors likely influence democracy scores.

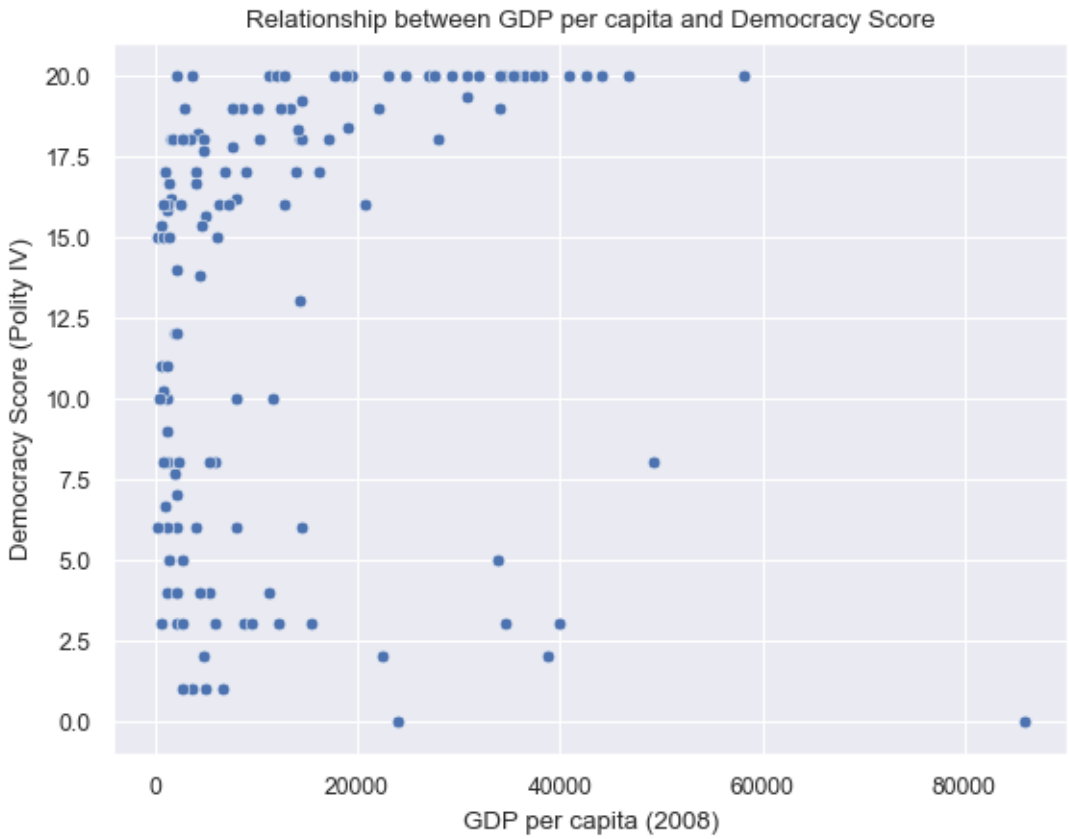
In conclusion: The results point to a positive connection between wealth and democracy in a country, but also show that democracy depends on more than just economic factors.

1.0.4 Exercise 4

Now let's plot the relationship you just estimated statistically. First, use `seaborn.objects` to create a scatter plot of `polityIV` and `gdppcap08`. Include a title and label your axes (with formatted words, not variable names).

[3]:
```
sns.set_theme("paper", font_scale=1)
reg_plot = sns.scatterplot(data=world_data, x="gdppcap08", y="polityIV")
plt.title("Relationship between GDP per capita and Democracy Score")
plt.xlabel("GDP per capita (2008)")
plt.ylabel("Democracy Score (Polity IV)")
```

[3]:
```
Text(0, 0.5, 'Democracy Score (Polity IV)')
```



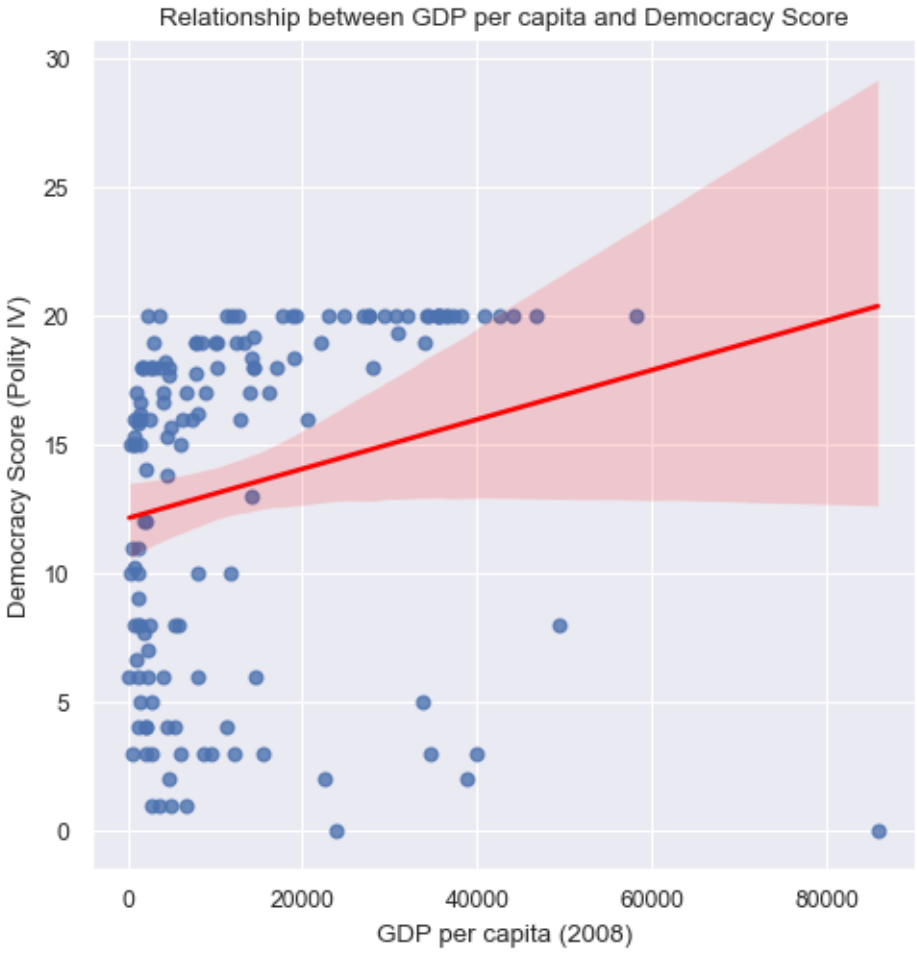
1.0.5 Exercise 5

Now overlay a linear regression (*not* a higher order polynomial, just linear) fit to the scatter plot.

Note: linear regression is *not* symmetric — regressing Polity IV on GDP per Capita does not give the same result as regressing GDP per Capita on Polity IV. That's because linear regression is designed to minimize the sum of squared errors between predicted values of the y-variable and the true values of the y-variable, so the y-variable is, in a sense, privileged.

Given that, for consistency be sure you make your plot reflect the regression you ran above.

[4]:
```
sns.lmplot(data=world_data, x="gdppcap08", y="polityIV", line_kws={"color":␣
↪"red"})
plt.title("Relationship between GDP per capita and Democracy Score")
plt.xlabel("GDP per capita (2008)")
plt.ylabel("Democracy Score (Polity IV)")
plt.show()
```



1.0.6 Exercise 6

Does it seem like the linear model you estimated fits the data well?

The linear model doesn't fit the data well for several reasons: - There's a large cluster of countries with low GDP (under $20,000) but a wide range of democracy scores, which a single line can't capture. - Outliers — especially high-GDP countries with low democracy scores - pull the line down, distorting the trend. The wide confidence interval also shows high uncertainty in the predictions.

Overall, the pattern looks non-linear, so a straight line oversimplifies the relationship between GDP and democracy.

1.0.7 Exercise 7

Linear models impose a very strict functional form on the model they use: they try to draw a straight line through the data, no matter what.

Can you think of a (mathematical) transformation for your data that would make the data a little more sane?

Apply the transformation.

[5]:
```
# log transformation
world_data["log_gdppcap08"] = np.log(world_data["gdppcap08"])
```

1.0.8 Exercise 8

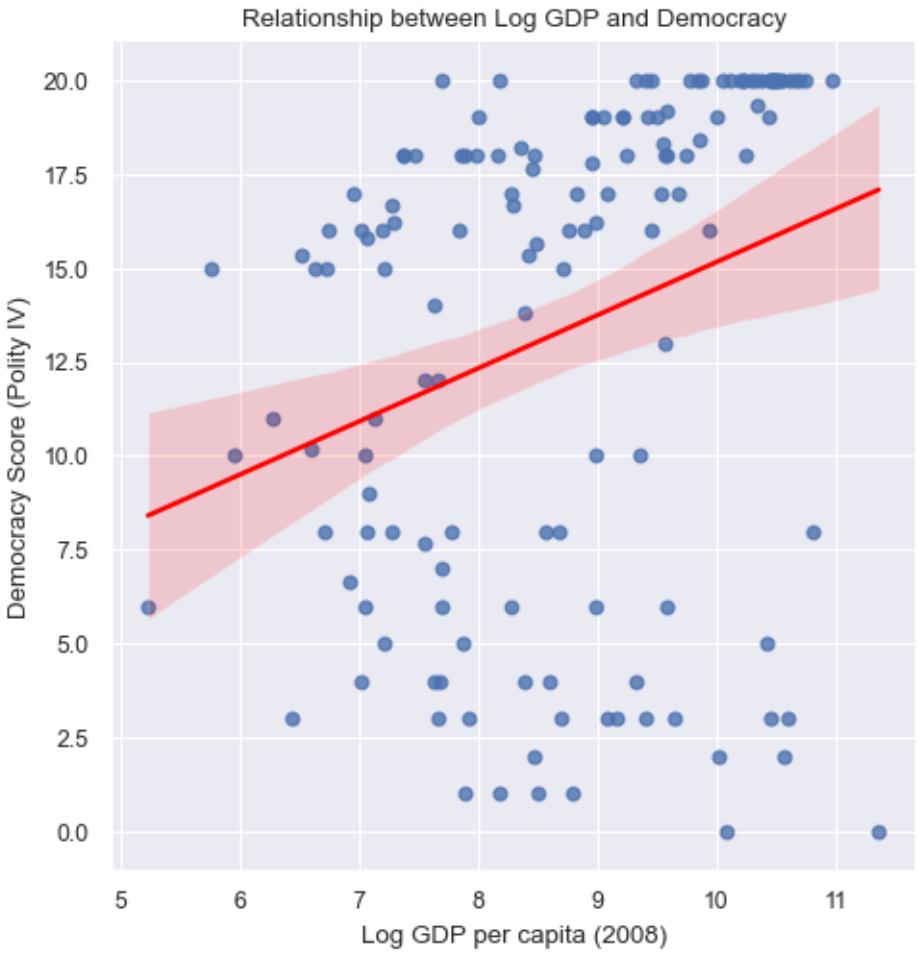
Once you've applied that transformation, let's re-fit our model.

Has you sense of the relationship changed at all?

[6]:
```
results_log = smf.ols("polityIV ~ log_gdppcap08", data=world_data).fit()

print(results_log.summary())

sns.lmplot(data=world_data, x="log_gdppcap08", y="polityIV", line_kws={"color":␣
↪"red"})
plt.title("Relationship between Log GDP and Democracy")
plt.xlabel("Log GDP per capita (2008)")
plt.ylabel("Democracy Score (Polity IV)")
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                polityIV   R-squared:                       0.083
Model:                            OLS   Adj. R-squared:                  0.077
Method:                 Least Squares   F-statistic:                     12.95
Date:                Fri, 10 Oct 2025   Prob (F-statistic):           0.000441
Time:                        15:49:46   Log-Likelihood:                -472.31
No. Observations:                 145   AIC:                             948.6
Df Residuals:                     143   BIC:                             954.6
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-
```

```
Intercept          0.9951       3.490       0.285       0.776       -5.903
7.893
log_gdppcap08      1.4163       0.394       3.598       0.000        0.638
2.194
==============================================================================
Omnibus:                       19.454   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               22.764
Skew:                          -0.940   Prob(JB):                     1.14e-05
Kurtosis:                       2.516   Cond. No.                         59.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```



Yes, my sense of the relationship has changed significantly.

The new model with log-transformed GDP has an R-squared of 0.083, which means it explains

8.3% of the variance in polity scores — almost doubling the improvement from the 4.7% of the original model. The relationship now appears much more linear and the fit is visibly better.

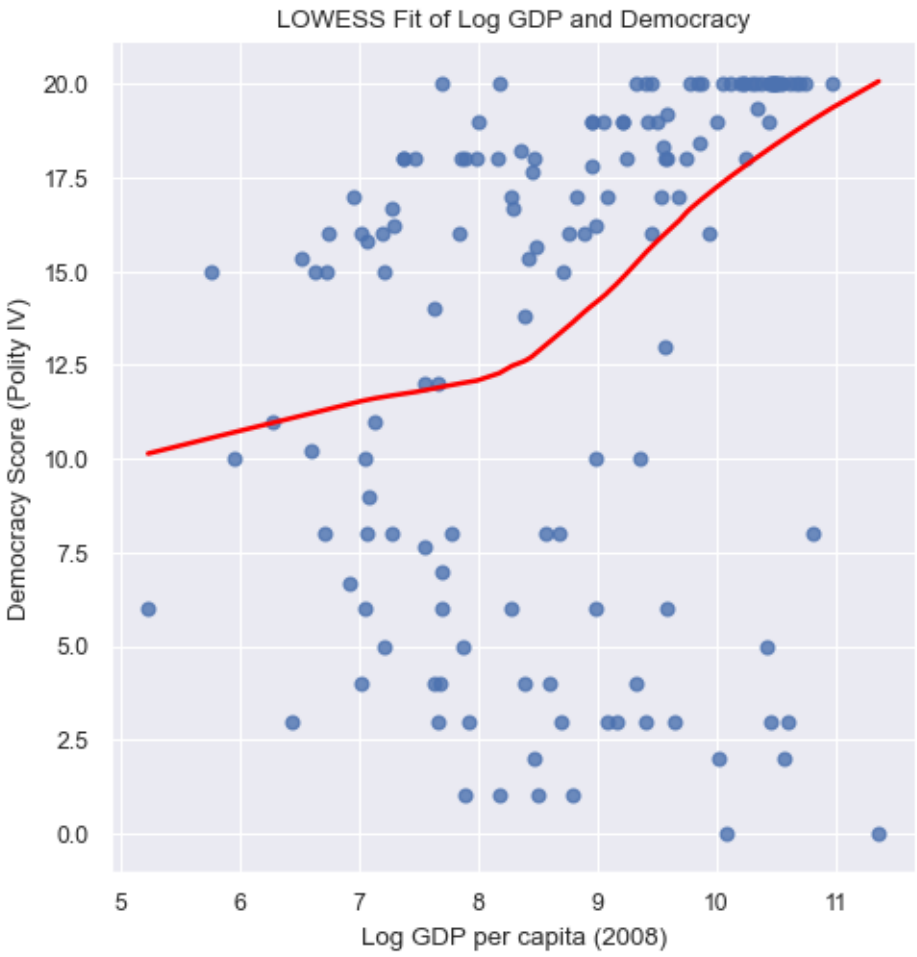
1.0.9 Exercise 9

When unsure of the most appropriate functional form for a model, it can be useful to fit a *non-parametric* model — one that does not impose a functional form as strictly.

Fit a lowess regression with confidence intervals using the `seaborn_objects_recipes` package.

`frac` is a keyword that manages how much lowess will smooth it's estimates across the data — higher values will be more smoothed, lower values with fit the data much tighter. Play with the values till you find something you think feels informative.

[7]:
```
sns.lmplot(
    data=world_data,
    x="log_gdppcap08",
    y="polityIV",
    lowess=True,  # Fit a LOWESS curve
    line_kws={"color": "red"},
)
plt.title("LOWESS Fit of Log GDP and Democracy")
plt.xlabel("Log GDP per capita (2008)")
plt.ylabel("Democracy Score (Polity IV)")
plt.show()
```



1.0.10 Exercise 10

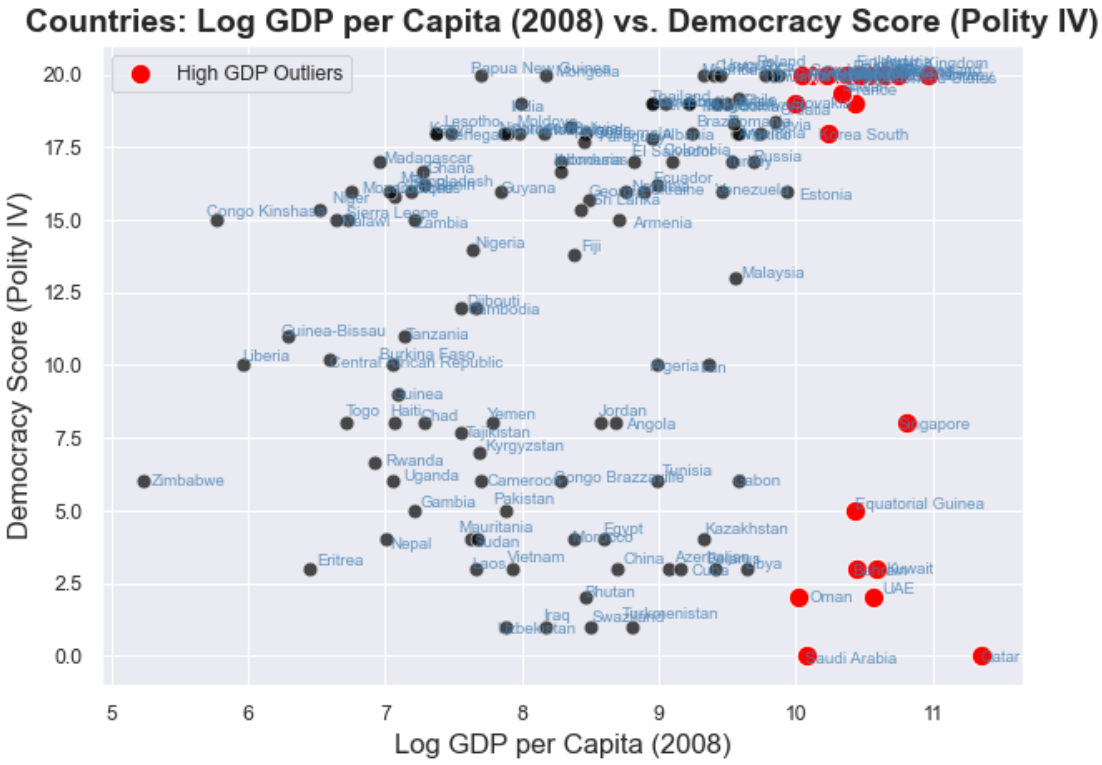
This does seem to fit the data better, but there seem to be quite a few outliers in the bottom right. Who is that? Add text labels to the points on your graph with country names.

Make sure the size of your text labels leaves them legible. There are enough points you won't be able to perfectly see all labels, but do your best.

You can also add `so.Jitter()` to "jitter" the location of labels (move them randomly so when several points share an exact coordinate, they'll be shifted a little).

[8]:
```
sns.scatterplot(
    data=world_data,
    x="log_gdppcap08",
    y="polityIV",
    s=40,
    alpha=0.7,
    color="black",
```

```
    edgecolor="white",
)

np.random.seed(61)
x_jitter = np.random.uniform(-0.1, 0.1, size=world_data.shape[0])
y_jitter = np.random.uniform(-0.3, 0.3, size=world_data.shape[0])

for i in range(world_data.shape[0]):
    plt.text(
        world_data.log_gdppcap08[i] + x_jitter[i],
        world_data.polityIV[i] + y_jitter[i],
        world_data.country[i],
        fontsize=7,
        color="steelblue",
        alpha=0.8,
    )

outliers = world_data[world_data["log_gdppcap08"] > 10]
sns.scatterplot(
    data=outliers,
    x="log_gdppcap08",
    y="polityIV",
    s=80,
    color="red",
    label="High GDP Outliers",
)

plt.title(
    "Countries: Log GDP per Capita (2008) vs. Democracy Score (Polity IV)",
    fontsize=14,
    weight="bold",
)
plt.xlabel("Log GDP per Capita (2008)", fontsize=12)
plt.ylabel("Democracy Score (Polity IV)", fontsize=12)
plt.legend()
plt.tight_layout()
plt.show()
```



1.0.11 Exercise 11

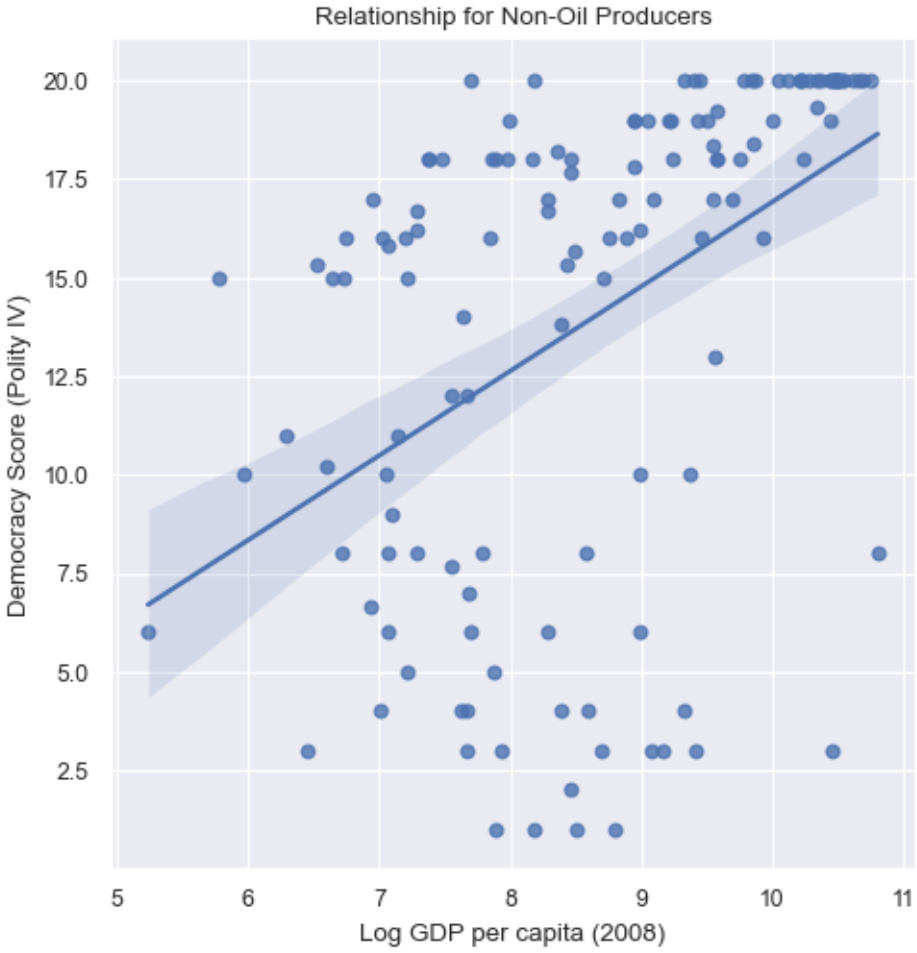
Interesting. It seems that there's are a lot of rich, undemocratic countries that all have something in common: they're oil-rich, small, Middle Eastern countries.

Let's see what happens if we exclude the ten countries with the highest per-capita oil production from our data: Qatar, Kuwait, Equatorial Guinea, United Arab Emirates, Norway, Saudi Arabia, Libya, Oman, Gabon, and Angola. (Note this was in 2007, and excludes very small countries!)

What does the relationship between Polity and GDP per capita look like for **non**-natural resource producers?

[9]:
```
oil_producers = [
    "Qatar",
    "Kuwait",
    "Equatorial Guinea",
    "UAE",
    "Norway",
    "Saudi Arabia",
    "Libya",
    "Oman",
    "Gabon",
    "Angola",
```

```
]

non_oil_data = world_data[~world_data["country"].isin(oil_producers)]

sns.lmplot(data=non_oil_data, x="log_gdppcap08", y="polityIV")
plt.title("Relationship for Non-Oil Producers")
plt.xlabel("Log GDP per capita (2008)")
plt.ylabel("Democracy Score (Polity IV)")
plt.show()
```



With the oil-rich countries removed, the positive relationship between wealth and democracy appears much stronger and more consistent than before.

1.0.12 Exercise 12

Let's make sure that you accurately identified all 10 of the oil producers. Write a line of code to count up how many big producers you have identified. If you do not get 10, can you figure out what you did wrong?

[10]:
```
count = world_data["country"].isin(oil_producers).sum()
print(f"Number of identified oil producers in the dataset: {count}")

missing_producers = [
    country for country in oil_producers if country not in␣
 ↪world_data["country"].values
]

print(f"Missing oil producer(s): {missing_producers}")
```

```
Number of identified oil producers in the dataset: 10
Missing oil producer(s): []
```

United Arab Emirates is missing in the world_data. Instead, it is spelt as UAE. Updating the oil_producers list below.

[11]:
```
oil_producers = [
    "Qatar",
    "Kuwait",
    "Equatorial Guinea",
    "UAE",
    "Norway",
    "Saudi Arabia",
    "Libya",
    "Oman",
    "Gabon",
    "Angola",
]
```

1.0.13 Exercise 13

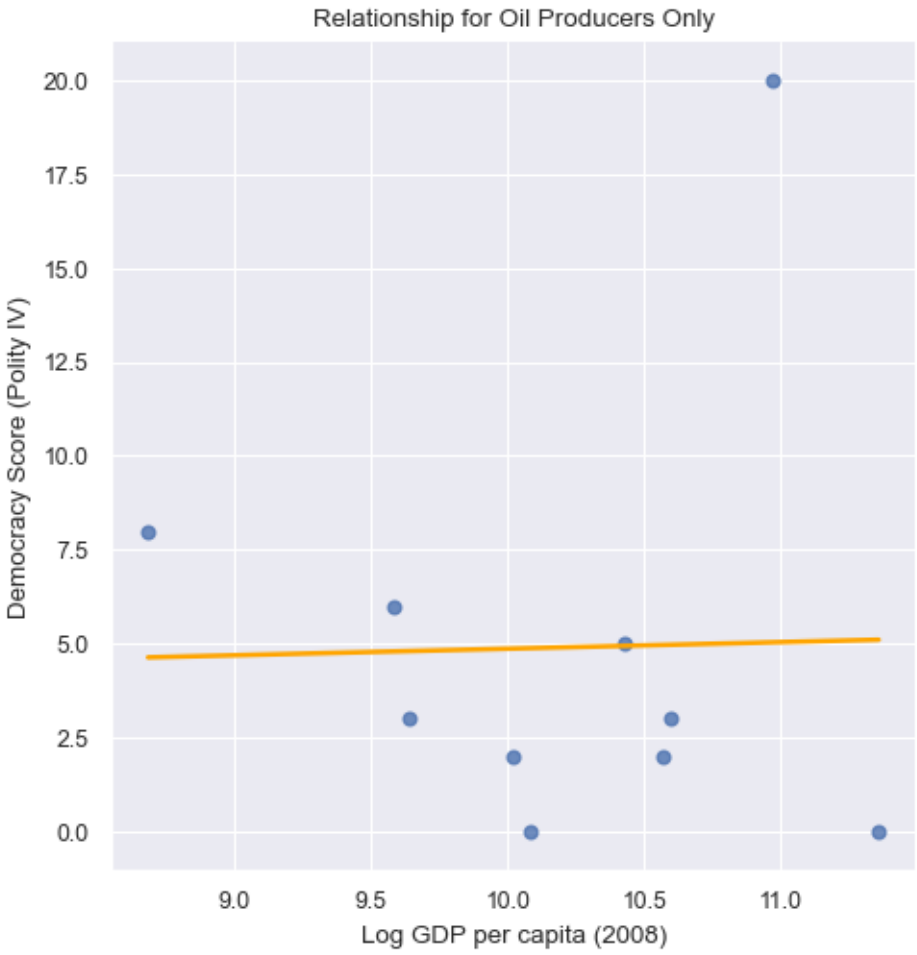
How does the relationship between GDP per capita and Polity look for the oil producers we dropped above?

(Note you can't add a confidence interval given there are only 10 observations here, so drop the CIs)

[12]:
```
model_all = smf.ols("polityIV ~ log_gdppcap08", data=world_data).fit()
model_no_oil = smf.ols("polityIV ~ log_gdppcap08", data=non_oil_data).fit()

coef_all = model_all.params["log_gdppcap08"]
coef_no_oil = model_no_oil.params["log_gdppcap08"]

oil_data = world_data[world_data["country"].isin(oil_producers)]

# Create the plot without a confidence interval
sns.lmplot(
    data=oil_data,
    x="log_gdppcap08",
```

```
    y="polityIV",
    ci=None,
    line_kws={"color": "orange"},
)
plt.title("Relationship for Oil Producers Only")
plt.xlabel("Log GDP per capita (2008)")
plt.ylabel("Democracy Score (Polity IV)")
```

[12]: `Text(20.056250000000006, 0.5, 'Democracy Score (Polity IV)')`



A notable difference in coefficients or R^2 suggests oil producers influence the estimated wealth-democracy relationship.

1.0.14 Exercise 12

Look back to your answer for Exercise 2. Do you still believe the result of your linear model? What did you learn from plotting. Write down your answers with your partner.

Looking back at the initial linear model from Exercise 2, I no longer believe it provides an accurate or useful summary of the relationship between wealth and democracy.

Initially, the statistical output suggested a weak but statistically significant positive relationship. However, plotting the data showed several issues that the simple regression model completely missed:

1. Non-Linearity: The relationship is clearly not linear. The log transformation in Exercise 7 showed that the relationship flattens out at higher GDPs, meaning a dollar increase in GDP has a much larger apparent effect on democracy for poor countries than for wealthy ones.
2. Outliers and Subgroups: Plotting revealed a distinct cluster of outliers—the oil-producing nations. These countries were both wealthy and autocratic, pulling the regression line down and weakening the overall observed positive trend. By treating them as a separate group, we discovered they have a negative relationship between wealth and democracy, which is a crucial insight hidden by the initial model.

What I learned from plotting is that statistical summaries alone can be misleading. A regression will always give you a "best-fit" line, but it's only by visualizing the data that you can determine if that line is a meaningful representation of the underlying patterns. Plotting is an essential diagnostic tool for identifying non-linearity, outliers, and hidden subgroups that can completely change the interpretation of the data.

1.0.15 Exercise 13

Finally, let's make a plot that color codes countries by whether they are big oil producers. Include separate linear regression fits for both groups.

[13]:
```
world_data["is_oil_producer"] = world_data["country"].isin(oil_producers)

sns.lmplot(
    data=world_data, x="log_gdppcap08", y="polityIV", hue="is_oil_producer",␣
 ↪ci=None
)
plt.title("Wealth & Democracy: Oil vs. Non-Oil Producers")
plt.xlabel("Log GDP per capita (2008)")
plt.ylabel("Democracy Score (Polity IV)")
```

[13]:
```
Text(40.40556400462965, 0.5, 'Democracy Score (Polity IV)')
```

1.1 Take-aways

One of our main jobs as data scientists is to *summarize* data. In fact, its such an obvious part of our jobs we often don't think about it very much. In reality, however, this is one of the most difficult things we do.

Summarization means taking rich, complex data and trying to tell readers about what is going on in that data using simple statistics. In the process of summarization, therefore, we must necessarily throw away much of the richness of the original data. When done well, this simplification makes data easier to understand, but only if we throw away the *right* data. You can *always* calulate the average value of a variable, or fit a linear model, but whether doing so generates a summary statistic that properly represents the essence of the data being studied depends on the data itself.

Plotting is one fo the best tools we have as data scientists for evaluating whether we are throwing away the *right* data. As we learned from Part 1 of this exercise, just looking at means and standard deviations can mask tremendous variation. Each of our example datasets looked the same when we examined our summary statistics, but they were all radically different when plotted.

Similarly, a simple linear model would "tell" us that if GDP per capita increases by $10,000, we

would expect Polity scores to increase by about 1 (i.e. the coefficent on the linear model was 9.602e-05). But when we plot the data, not only can we that the data is definitely *not* linear (and so that slope doesn't really mean anything), but we can also see that oil producing countries seem to defy the overall trend, and so should maybe be studied separately.

Moreover, we can see that if we just look at oil producers, there is no clear story: some are rich and democratic, while others are rich and autocratic (indeed, this observation is the foundation of some great research on the political consequences of resource wealth!)

So remember this: tools for summarizing data will always give you an answer, but it's up to you as a data scientist to make sure that the summaries you pass on to other people properly represent the data you're using. And there is perhaps no better way to do this than with plotting!

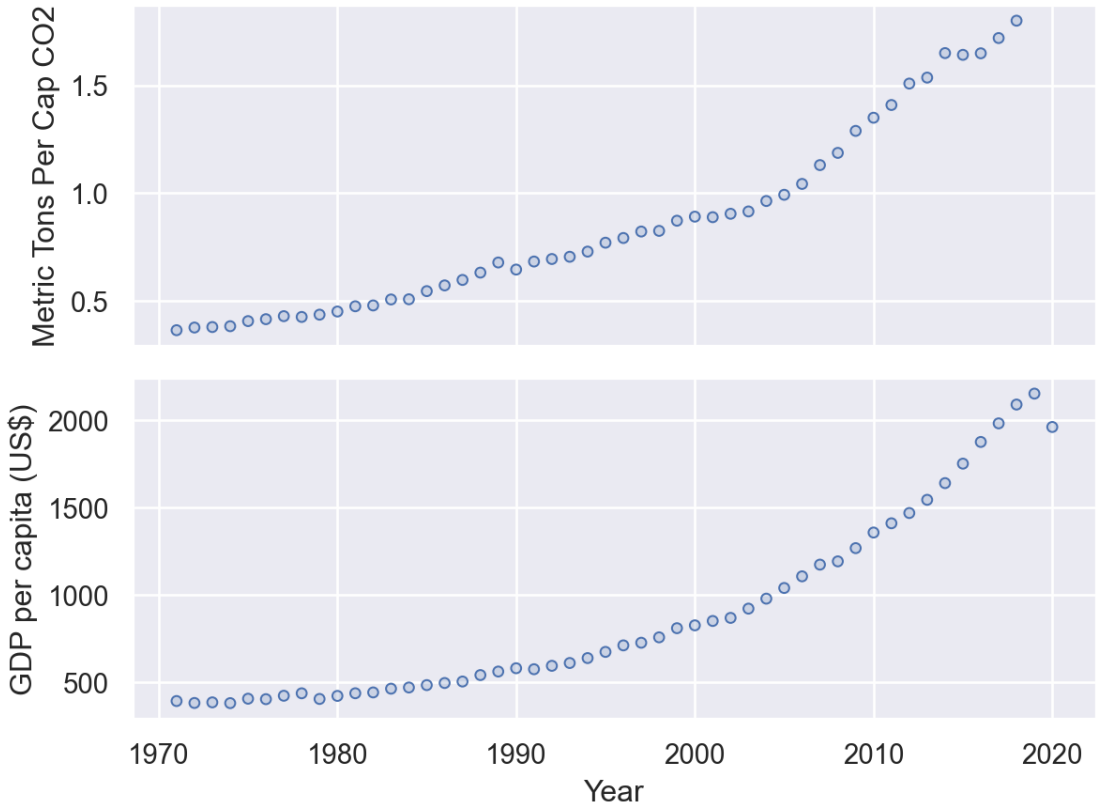
1.2 Overlaying Data Series with matplotlib

In our last plotting exercises, you were asked to make a paired plot in which different data series were plotted next to one another with a shared x-axis. Presumably that resulted in a figure that looked something like this:

[14]:
```
import pandas as pd
import numpy as np
import seaborn.objects as so
from matplotlib import style
import matplotlib.pyplot as plt
import warnings

warnings.simplefilter(action="ignore", category=FutureWarning)
nick_theme = {**style.library["seaborn-v0_8-whitegrid"]}
nick_theme.update({"font.sans-serif": ["Fira Sans", "Arial", "sans-serif"]})

pd.set_option("mode.copy_on_write", True)

wdi = pd.read_csv(
    "https://raw.githubusercontent.com/nickeubank/"
    "practicaldatascience/master/Example_Data/wdi_plotting.csv"
)

india = wdi[wdi["Country Name"] == "India"]

india = india.rename(
    columns={
        "CO2 emissions (metric tons per capita)": "Metric Tons Per Cap CO2",
        "GDP per capita (constant 2010 US$)": "GDP per capita (US$)",
    }
)
p = (
    so.Plot(
        india,
        x="Year",
```

```
    )
    .add(so.Dots())
    .pair(
        y=[
            "Metric Tons Per Cap CO2",
            "GDP per capita (US$)",
        ]
    )
)
p
```

[14]:



Often times, however, it's more interesting to directly overlay data series on the same plot to make a figure like this:

So let's do that here!

1.2.1 Exercise 14

Using the data from our last exercise (`exercise_plotting_part1`), let's do something like that!

Making this work will require two new tricks:

- using the `.twinx()` method from matplotlib, and

- using the `.on()` method from seaborn.objects.

How? Great question! I'm going to leave it to you to figure that out using the documentation for these methods. But here's a start — you can find the `.on()` method for seaborn.objects here, and the `.twinx()` for matplotlib method demonstrated here

Oh, and you may note use these two variables as your two. :)

Good luck!

Also, if you want to, feel free to add any extra bells and whistles as part of your exploration (like a legend, or colored y-axis labels).

[15]:
```
india = wdi[wdi["Country Name"] == "India"]
india = india.rename(
    columns={
        "CO2 emissions (metric tons per capita)": "CO2 Emissions (Metric Tons/
 ↪Capita)",
        "GDP per capita (constant 2010 US$)": "GDP per Capita (US$)",
    }
)

fig, ax1 = plt.subplots(figsize=(10, 6))

ax2 = ax1.twinx()

(
    so.Plot(india, x="Year", y="GDP per Capita (US$)")
    .add(so.Line(color="steelblue"), label="GDP per Capita")
    .on(ax1)
    .plot()
)

(
    so.Plot(india, x="Year", y="CO2 Emissions (Metric Tons/Capita)")
    .add(so.Line(color="firebrick"), label="CO2 Emissions")
    .on(ax2)
    .plot()
)

ax1.set_xlabel("Year", fontsize=12)
ax1.set_ylabel("GDP per Capita (US$)", fontsize=12, color="steelblue")
ax2.set_ylabel("CO2 Emissions (Metric Tons/Capita)", fontsize=12,␣
 ↪color="firebrick")
plt.title(
    "India: GDP per Capita vs. CO2 Emissions Over Time", fontsize=14,␣
 ↪weight="bold"
)
```

```
ax1.tick_params(axis="y", labelcolor="steelblue")
ax2.tick_params(axis="y", labelcolor="firebrick")

lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines1 + lines2, labels1 + labels2, loc="upper left")

plt.tight_layout()
```