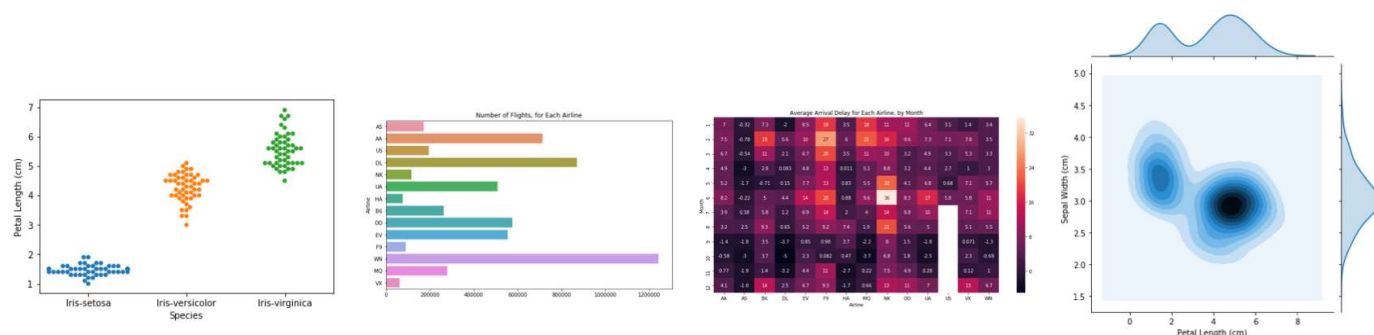


Welcome to Data Visualization!

In this hands-on micro-course, you'll learn how to take your data visualizations to the next level with [seaborn](#), a powerful but easy-to-use data visualization tool. To use seaborn, you'll also learn a bit about how to write code in Python, a popular programming language. That said,

- the micro-course is aimed at those with no prior programming experience, and
- each chart uses short and simple code, making seaborn much faster and easier to use than many other data visualization tools (*such as Excel, for instance*).

So, if you've never written a line of code, and you want to learn the **bare minimum** to start making faster, more attractive plots today, you're in the right place! To take a peek at some of the charts you'll make, check out the figures below.



Your coding environment

Take the time now to scroll quickly up and down this page. You'll notice that there are a lot of different types of information, including:

1. **text** (like the text you're reading right now!),
2. **code** (which is always contained inside a gray box called a **code cell**), and
3. **code output** (or the printed result from running code that always appears immediately below the corresponding code).

We refer to these pages as **Jupyter notebooks** (or, often just **notebooks**), and we'll work with them throughout the mini-course. Another example of a notebook can be found in the image below.

text → In this tutorial, you'll learn how to create advanced **scatter plots**.

code → **Set up the notebook**

As always, we begin by setting up the coding environment. (This code is hidden, but you can un-hide and re-hide it by clicking on the "Code" button immediately below this text, on the right.)

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
print("Setup Complete")
```

code output → Setup Complete

text → **Color-coded scatter plots**

We can use scatter plots to display the relationships between (not two, but...) three variables! One way of doing this is by color-coding the points. For instance, to understand how smoking affects the relationship between BMI and insurance costs, we can color-code the points by 'smoker', and plot the other two columns ('bmi', 'charges') on the axes.

code → In [6]: `sns.scatterplot(x=insurance_data['bmi'], y=insurance_data['charges'], hue=insurance_data['smoker'])`

code output → Out[6]: `<matplotlib.axes._subplots.AxesSubplot at 0x10cbd83c8>`



In the notebook you're reading now, we've already run all of the code for you. Soon, you will work with a notebook that allows you to write and run your own code!

Set up the notebook

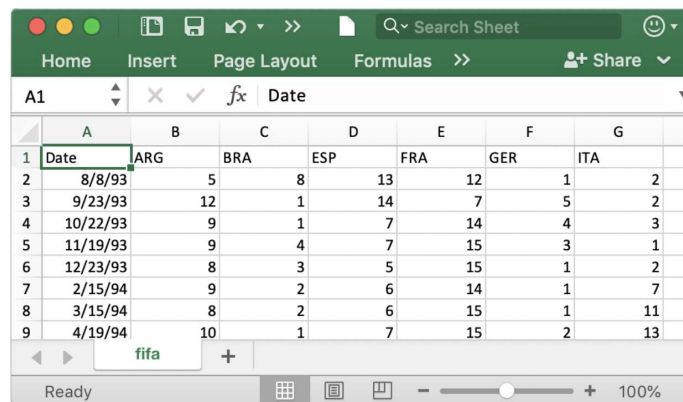
There are a few lines of code that you'll need to run at the top of every notebook to set up your coding environment. It's not important to understand these lines of code now, and so we won't go into the details just yet. (Notice that it returns as output: `Setup Complete`.)

```
In [1]: import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
print("Setup Complete")
```

Setup Complete

Load the data

In this notebook, we'll work with a dataset of historical FIFA rankings for six countries: Argentina (ARG), Brazil (BRA), Spain (ESP), France (FRA), Germany (GER), and Italy (ITA). The dataset is stored as a CSV file (short for [comma-separated values file](#)). Opening the CSV file in Excel shows a row for each date, along with a column for each country.



	A	B	C	D	E	F	G
	Date	ARG	BRA	ESP	FRA	GER	ITA
1	8/8/93	5	8	13	12	1	2
2	9/23/93	12	1	14	7	5	2
3	10/22/93	9	1	7	14	4	3
4	11/19/93	9	4	7	15	3	1
5	12/23/93	8	3	5	15	1	2
6	2/15/94	9	2	6	14	1	7
7	3/15/94	8	2	6	15	1	11
8	4/19/94	10	1	7	15	2	13

To load the data into the notebook, we'll use two distinct steps, implemented in the code cell below as follows:

- begin by specifying the location (or [filepath](#)) where the dataset can be accessed, and then
- use the filepath to load the contents of the dataset into the notebook.

In [2]:

```
# Path of the file to read
fifa_filepath = "../input/fifa.csv"

# Read the file into a variable fifa_data
fifa_data = pd.read_csv(fifa_filepath, index_col="Date", parse_dates=True)
```

```
# Path of the file to read
fifa_filepath = "../input/fifa.csv"

# Read the file into a variable fifa_data
fifa_data = pd.read_csv(fifa_filepath, index_col="Date", parse_dates=True)
```

Note that the code cell above has **four** different lines.

Comments

Two of the lines are preceded by a pound sign (#) and contain text that appears faded and italicized.

Both of these lines are completely ignored by the computer when the code is run, and they only appear here so that any human who reads the code can quickly understand it. We refer to these two lines as **comments**, and it's good practice to include them to make sure that your code is readily interpretable.

Executable code

The other two lines are **executable code**, or code that is run by the computer (*in this case, to find and load the dataset*).

The first line sets the value of `fifa_filepath` to the location where the dataset can be accessed. In this case, we've provided the filepath for you (in quotation marks). *Note that the **comment** immediately above this line of **executable code** provides a quick description of what it does!*

The second line sets the value of `fifa_data` to contain all of the information in the dataset. This is done with `pd.read_csv`. It is immediately followed by three different pieces of text (underlined in the image above) that are enclosed in parentheses and separated by commas. These are used to customize the behavior when the dataset is loaded into the notebook:

- `fifa_filepath` - The filepath for the dataset always needs to be provided first.
- `index_col="Date"` - When we load the dataset, we want each entry in the first column to denote a different row. To do this, we set the value of `index_col` to the name of the first column ("Date" , found in cell A1 of the file when it's opened in Excel).
- `parse_dates=True` - This tells the notebook to understand the each row label as a date (as opposed to a number or other text with a different meaning).

These details will make more sense soon, when you have a chance to load your own dataset in a hands-on exercise.

For now, it's important to remember that the end result of running both lines of code is that we can now access the dataset from the notebook by using `fifa_data` .

By the way, you might have noticed that these lines of code don't have any output (whereas the lines of code you ran earlier in the notebook returned `Setup Complete` as output). This is expected behavior -- not all code will return output, and this code is a prime example!

Examine the data

Now, we'll take a quick look at the dataset in `fifa_data`, to make sure that it loaded properly.

We print the *first* five rows of the dataset by writing one line of code as follows:

- begin with the variable containing the dataset (in this case, `fifa_data`), and then
- follow it with `.head()`.

```
In [3]: # Print the first 5 rows of the data
fifa_data.head()
```

Out[3]:

	ARG	BRA	ESP	FRA	GER	ITA
Date						
1993-08-08	5.0	8.0	13.0	12.0	1.0	2.0
1993-09-23	12.0	1.0	14.0	7.0	5.0	2.0
1993-10-22	9.0	1.0	7.0	14.0	4.0	3.0
1993-11-19	9.0	4.0	7.0	15.0	3.0	1.0
1993-12-23	8.0	3.0	5.0	15.0	1.0	2.0

Check now that the first five rows agree with the image of the dataset (*from when we saw what it would look like in Excel*) above.

Plot the data

In this micro-course, you'll learn about many different plot types. In many cases, you'll only need one line of code to make a chart!

For a sneak peak at what you'll learn, check out the code below that generates a line chart.

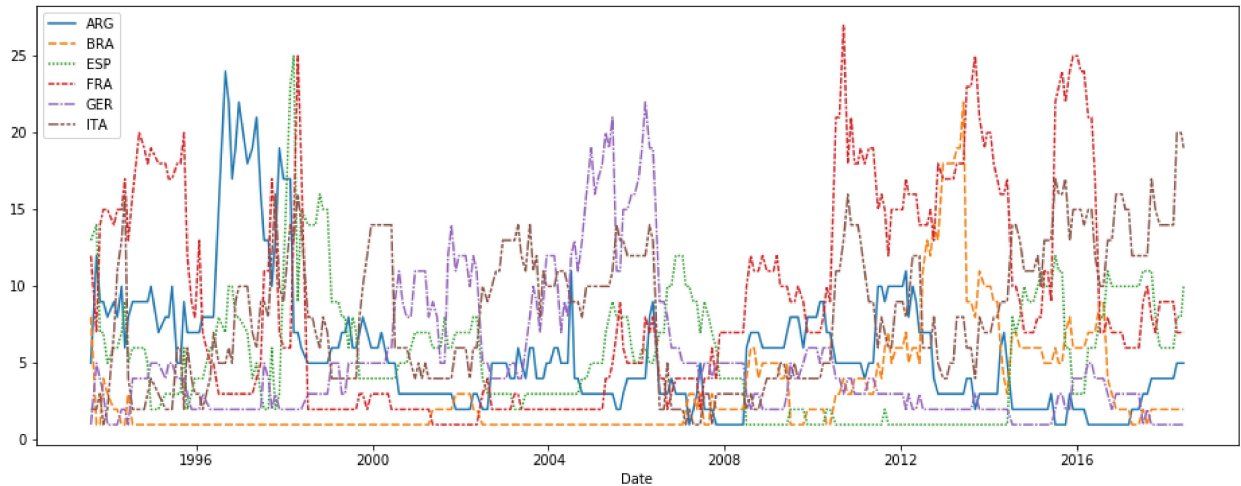
In [4]:

```
# Set the width and height of the figure
plt.figure(figsize=(16,6))

# Line chart showing how FIFA rankings evolved over time
sns.lineplot(data=fifa_data)
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f22bfac3fd0>
```



This code shouldn't make sense just yet, and you'll learn more about it in the upcoming tutorials. For now, continue to your first exercise, where you'll get a chance to experiment with the coding environment yourself!

What's next?

Write your first lines of code in the [first coding exercise](#)!