universidad de león

APUNTES DATA VISUALIZATION

Seaborn a powerful but easy-to-use data visualization tool

```
import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Line Charts

Once the dataset is loaded into the notebook, we need only one line of code to make a line chart!

```
sns.lineplot(data=spotify_data)
```

sns.lineplot tells the notebook that we want to create a line chart.

⇒ Every command that you learn about in this course will start with *sns*, which indicates that the command comes from the seaborn package.

⇒ data=spotify_data selects the data that will be used to create the chart.

Sometimes there are additional details we'd like to modify, like the size of the figure and the title of the chart.

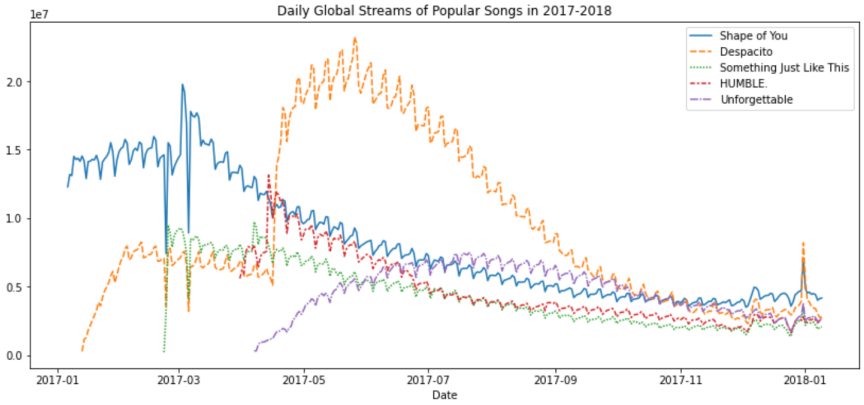
```
# Set the width and height of the figure
plt.figure(figsize=(14,6))

# Add title
plt.title("Daily Global Streams of Popular Songs in 2017-2018”)
```

Plot a subset of the data

```
# Line chart showing daily global streams of 'Shape of You'
sns.lineplot(data=spotify_data['Shape of You'], label="Shape of You”)

# Add label for horizontal axis
plt.xlabel(“Date")
```



Bar Charts and Heatmaps

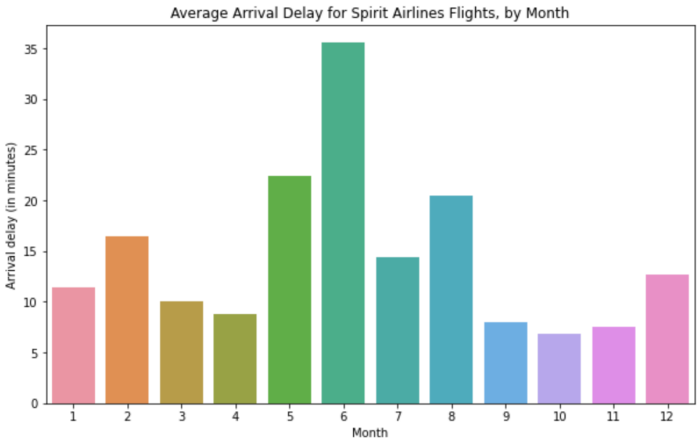
 # Bar chart showing average arrival delay for Spirit Airlines flights by month
sns.barplot(x=flight_data.index, y=flight_data['NK'])

It has three main components:

⇒ sns.barplot - This tells the notebook that we want to create a bar chart.

⇒ x=flight_data.index - This determines what to use on the horizontal axis. In this case, we have selected the column that **index**es the rows

⇒ y=flight_data['NK'] - This sets the column in the data that will be used to determine the height of each bar. In this case, we select the 'NK' column.



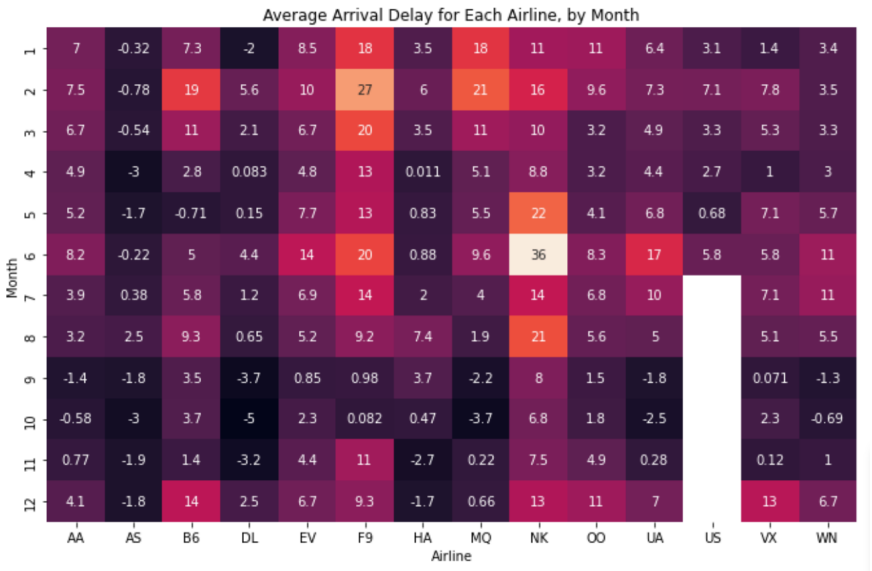
 # Heatmap showing average arrival delay for each airline by month
sns.heatmap(data=flight_data, annot=True)

This code has three main components:

⇒ sns.heatmap - This tells the notebook that we want to create a heatmap.

⇒ data=flight_data - This tells the notebook to use all of the entries in flight_data to create the heatmap.

⇒ annot=True - This ensures that the values for each cell appear on the chart.



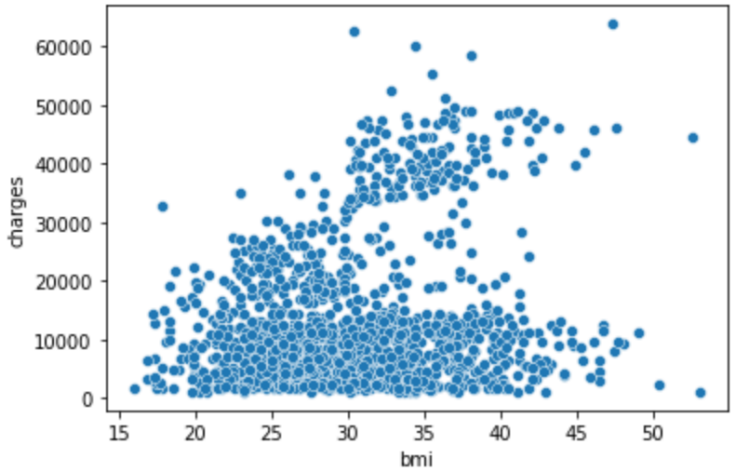
Scatter Plots

To create a simple **scatter plot**, we use the sns.scatterplot command and specify the values for:

⇒ the horizontal x-axis (x=insurance_data['bmi']).

⇒ the vertical y-axis (y=insurance_data[‘charges']).

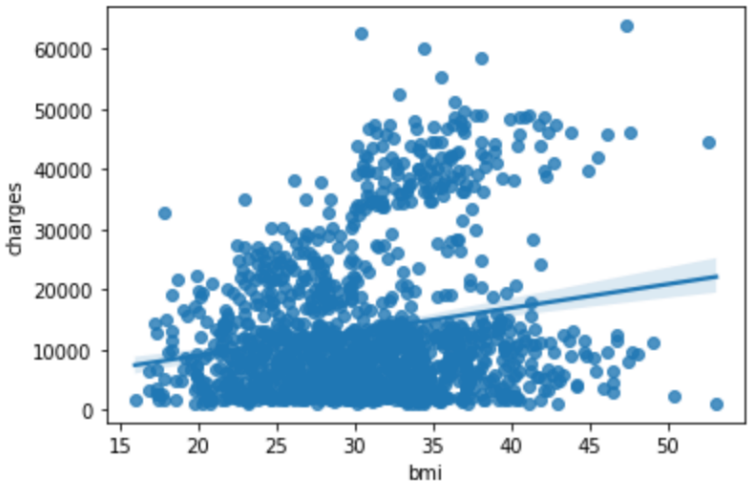
```
sns.scatterplot(x=insurance_data['bmi'], y=insurance_data['charges'])
```



The scatterplot above suggests that body mass index (BMI) and insurance charges are **positively correlated**, where customers with higher BMI typically also tend to pay more in insurance costs.

To double-check the strength of this relationship, you might like to add a **regression line**, or the line that best fits the data. We do this by changing the command to sns.regplot.

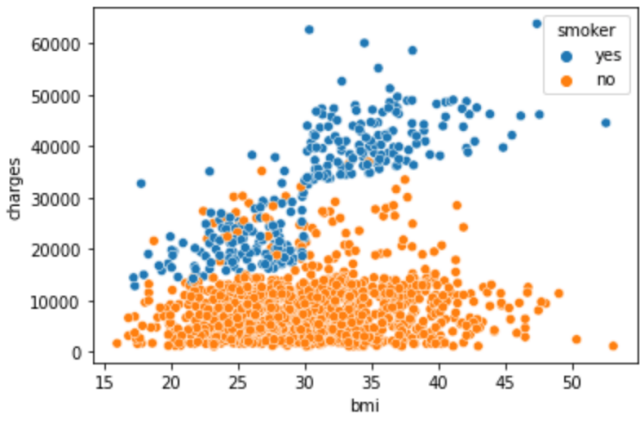
```
sns.regplot(x=insurance_data['bmi'], y=insurance_data['charges'])
```



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.

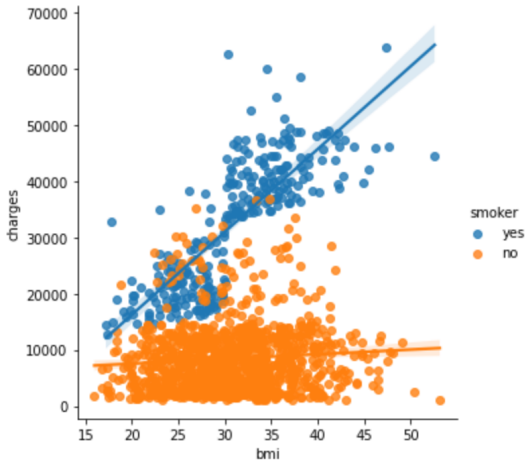
For instance, to understand how smoking affects the relationship between BMI and insurance costs, we can color-code the points by 'smoker', and plot the other two columns ('bmi', 'charges') on the axes.

```
sns.scatterplot(x=insurance_data['bmi'], y=insurance_data['charges'],
                hue=insurance_data['smoker'])
```



To further emphasize this fact, we can use the sns.lmplot command to add two regression lines, corresponding to smokers and nonsmokers.

```
sns.lmplot(x="bmi", y="charges", hue="smoker", data=insurance_data)
```



The sns.lmplot command above works slightly differently than the commands you have learned about so far:

⇒ Instead of setting x=insurance_data['bmi'] to select the 'bmi' column in insurance_data, we set x="bmi" to specify the name of the column only.

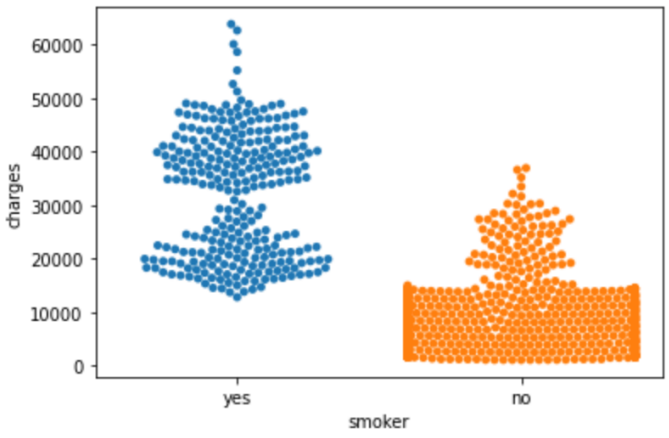
⇒ Similarly, y="charges" and hue="smoker" also contain the names of columns.

⇒ We specify the dataset with data=insurance_data.

Finally, there's one more plot that you'll learn about. Usually, we use scatter plots to highlight the relationship between two continuous variables. However, we can adapt the design of the scatter plot to feature a categorical variable (like "smoker") on one of the main axes.

We'll refer to this plot type as a **categorical scatter plot**, and we build it with the sns.swarmplot command.

```
sns.swarmplot(x=insurance_data['smoker'],
              y=insurance_data['charges'])
```



Among other things, this plot shows us that:

⇒ on average, non-smokers are charged less than smokers, and

⇒ the customers who pay the most are smokers; whereas the customers who pay the least are non-smokers.

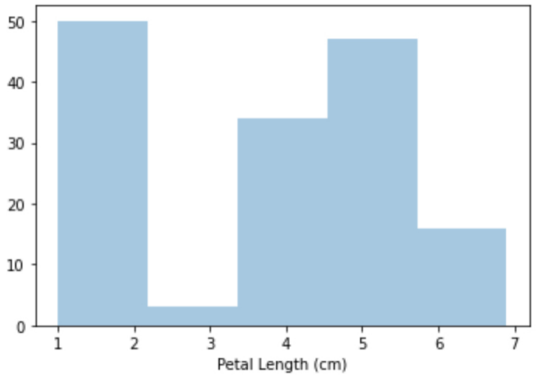
universidad de león

Distributions

Histograms¶

Say we would like to create a histogram to see how petal length varies in iris flowers. We can do this with the sns.distplot command.

```
# Histogram
sns.distplot(a=iris_data['Petal Length (cm)'], kde=False)
```



We customize the behavior of the command with two additional pieces of information:

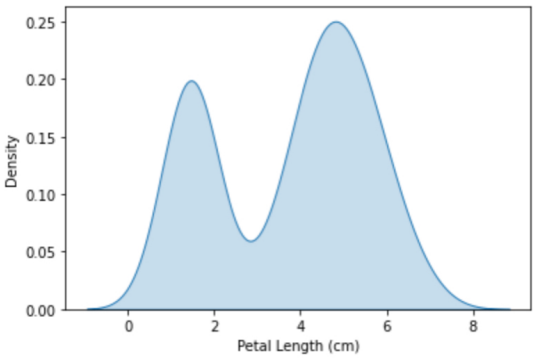
- ⇒ a= chooses the column we'd like to plot (in this case, we chose 'Petal Length (cm)').

- ⇒ kde=False is something we'll always provide when creating a histogram

Density plots¶

In case you're not familiar with KDE plots, you can think of it as a smoothed histogram.
To make a KDE plot, we use the sns.kdeplot command. Setting shade=True colors the area below the curve.

```
# KDE plot
sns.kdeplot(data=iris_data['Petal Length (cm)'], shade=True)
```



2D KDE plots¶

We're not restricted to a single column when creating a KDE plot. We can create a two-dimensional (2D) KDE plot with the sns.jointplot command.

```
# 2D KDE plot
sns.jointplot(x=iris_data['Petal Length (cm)'], y=iris_data['Sepal Width (cm)'],
kind="kde")
```

Color-coded plots¶

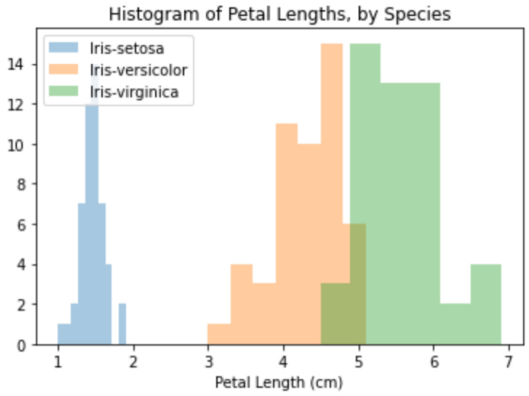
For the next part of the tutorial, we'll create plots to understand differences between the species. To accomplish this, we begin by breaking the dataset into three separate files, with one for each species.

```
# Paths of the files to read
iris_set_filepath = "../input/iris_setosa.csv"
iris_ver_filepath = "../input/iris_versicolor.csv"
iris_vir_filepath = "../input/iris_virginica.csv"

# Read the files into variables
iris_set_data = pd.read_csv(iris_set_filepath, index_col="Id")
iris_ver_data = pd.read_csv(iris_ver_filepath, index_col="Id")
iris_vir_data = pd.read_csv(iris_vir_filepath, index_col="Id")
```

In the code cell below, we create a different histogram for each species

```
# Histograms for each species
sns.distplot(a=iris_set_data['Petal Length (cm)'], label="Iris-setosa", kde=False)
sns.distplot(a=iris_ver_data['Petal Length (cm)'], label="Iris-versicolor",
kde=False)
sns.distplot(a=iris_vir_data['Petal Length (cm)'], label="Iris-virginica",
kde=False)
```



One interesting pattern that can be seen in plots is that the plants seem to belong to one of two groups, where Iris versicolor and Iris virginica seem to have similar values for petal length, while Iris setosa belongs in a category all by itself.

⇒ Trends - A trend is defined as a pattern of change.

sns.lineplot - Line charts are best to show trends over a period of time, and multiple lines can be used to show trends in more than one group.

⇒ Relationship - There are many different chart types that you can use to understand relationships between variables in your data.

sns.barplot - Bar charts are useful for comparing quantities corresponding to different groups.

sns.heatmap - Heatmaps can be used to find color-coded patterns in tables of numbers.

sns.scatterplot - Scatter plots show the relationship between two continuous variables; if color-coded, we can also show the relationship with a third categorical variable.

sns.regplot - Including a regression line in the scatter plot makes it easier to see any linear relationship between two variables.

sns.lmplot - This command is useful for drawing multiple regression lines, if the scatter plot contains multiple, color-coded groups.

sns.swarmplot - Categorical scatter plots show the relationship between a continuous variable and a categorical variable.

⇒ Distribution - We visualize distributions to show the possible values that we can expect to see in a variable, along with how likely they are.

sns.distplot - Histograms show the distribution of a single numerical variable.

sns.kdeplot - KDE plots (or 2D KDE plots) show an estimated, smooth distribution of a single numerical variable (or two numerical variables).

sns.jointplot - This command is useful for simultaneously displaying a 2D KDE plot with the corresponding KDE plots for each individual variable.