Programación para *Data Science*

Unidad 8: Visualización de datos en Python

Introducción

En este Notebook encontraréis dos conjuntos de ejercicios: un primer conjunto de **ejercicios para practicar** y un segundo conjunto de **actividades evaluables** como PEC de la asignatura.

En cuanto al conjunto de ejercicios para practicar, éstos no puntúan para la PEC, pero os recomendamos que los intentéis resolver como parte del proceso de aprendizaje. Encontraréis ejemplos de posibles soluciones a los ejercicios al propio notebook, pero es importante que intentéis resolverlos vosotros antes de consultar las soluciones. Las soluciones os permitirán validar vuestras respuestas, así como ver alternativas de resolución de las actividades. También os animamos a preguntar cualquier duda que surja sobre la resolución de los **ejercicios para practicar** en el foro del aula.

En relación a las actividades evaluables, veréis que cada una de ellas tiene asociada una puntuación que indica el peso que tiene la actividad sobre la nota de la PEC. Adicionalmente, hay un ejercicio opcional, que no tiene puntuación dentro de la PEC, pero que se valora al final del semestre de cara a conceder las matrículas de honor y redondear las notas finales. Podéis sacar la máxima nota de la PEC sin necesidad de hacer este ejercicio! El objetivo de este ejercicio es que sirva como pequeño reto para los estudiantes que quieran profundizar en el contenido de la asignatura.

Además, veréis que todas las actividades tienen una etiqueta que indica los recursos necesarios para llevarla a cabo. Hay tres posibles etiquetas:

- `NM` **Sólo materiales**: las herramientas necesarias para realizar la actividad se pueden encontrar en los materiales de la asignatura.
- `EG` **Consulta externa guiada**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, pero el enunciado contiene indicaciones de dónde o cómo encontrar la información adicional necesaria para resolver la actividad.
- `EI` **Consulta externa independente**: la actividad puede requerir hacer uso de herramientas que no se encuentran en los materiales de la asignatura, y el enunciado puede no incluir la descripción de dónde o cómo encontrar esta información adicional. Será necesario que el estudiante busque esta información utilizando los recursos que se han explicado en la asignatura.

Es importante notar que estas etiquetas no indican el nivel de dificultad del ejercicio, sino únicamente la necesidad de consulta de documentación externa para su resolución. Además, recordad que las **etiquetas son informativas**, pero podréis consultar referencias externas en cualquier momento (aunque no se indique explícitamente) o puede ser que podáis hacer una actividad sin consultar ningún tipo de documentación. Por ejemplo, para resolver una actividad que sólo requiera los materiales de la asignatura, puedéis consultar referencias externas si queréis, ya sea tanto para ayudaros en la resolución como para ampliar el conocimiento!

En cuanto a la consulta de documentación externa en la resolución de los ejercicios, recordad **citar siempre la bibliografía utilizada** para resolver cada actividad.

Ejercicios para practicar

Los siguientes 3 ejercicios no puntúan para la PEC , pero os recomendamos que los intentéis resolver antes de pasar a los ejercicios propios de la PEC. También encontraréis las soluciones a estos ejercicios al final del Notebook.

Ejercicio 1

Cargad el conjunto de datos `pulitzer-circulation-data.csv` en un dataframe de Pandas. La fuente original de este conjunto de datos es el repositorio de datos de [FiveThirtyEight] (https://github.com/fivethirtyeight/data).

Cread un diagrama de dispersión que permita visualizar las muestras del conjunto de datos de pulitzer según las variables `Daily Circulation 2004` y `Daily Circulation 2013` . Incluid una recta con el ajuste lineal entre las dos variables. En cuanto a los detalles de visualización, limitad la visualización de ambos ejes en el intervalo (0, 1000000), utilice el estilo `whitegrid` de Seaborn, y dibuja los puntos y la línea en color verde.

Pista: podéis utilizar la función `jointplot` que hemos visto en el Notebook de explicación. Considerad qué tipo de gráfica, [de entre las que ofrece jointplot] (http://seaborn.pydata.org/generated/seaborn.jointplot.html), se ajusta a los requerimientos del enunciado.

In [1]:

```
# Respuesta
```

Ejercicio 2

Juego de Tronos es una serie de televisión basada en la famosa saga *Una Canción de Fuego y Hielo* de George RR Martin. Esta serie es conocida por sus complejos escenarios políticos y bélicos, así como las numerosas muertes de personajes.

En este ejercicio trabajamos con el dataset `battles.csv` que nos da información sobre las batallas que han tenido lugar a lo largo de *Juego de Tronos*.

Cread un diagrama de puntos (*scatter*) que permita comparar el tamaño (`attacker_size` vs `defender_size`) de los dos principales ejércitos involucrados en cada batalla. Utilizad la columna `attacker_outcome` para mostrar con diferentes colores el resultado de la batalla por el atacante.

Podéis utilizar la función [`lmplot`] (https://seaborn.pydata.org/generated/seaborn.lmplot.html) de Seaborn para generar la gráfica. Considerad qué atributos nos permiten crear este tipo de visualización.

Viendo la gráfica, podemos afirmar que el tamaño del ejército es determinante en el *outcome* de la batalla?

Nota: utilizad el estilo `whitegrid` de Seaborn.

In [2]:

```
# Respuesta
```

Ejercicio 3

En los reinos de Poniente, los Grandes Maestros consideran que todo el mundo debería tener acceso a una biblioteca. Desgraciadamente, después de una gran guerra la red de carreteras que comunican estos reinos ha quedado gravemente afectada. Los Grandes Maestros nos han contratado para resolver las siguientes cuestiones:

1. ¿Qué reinos han quedado sin poder acceder a una biblioteca?
2. ¿Cuál es el número mínimo de bibliotecas que se deberían construir para que todo el mundo vuelva a tener acceso a una biblioteca? (Si no podemos re-construir ninguna carretera)
3. ¿Cuál es el coste mínimo de re-construcción de carreteras a fin de que todo el mundo vuelva a tener acceso a una biblioteca? (Si no podemos construir ninguna biblioteca)
4. Viendo la respuesta de la pregunta 2 y 3, que sería lo más óptimo teniendo en cuenta que queremos minimizar los costes?

Para poder responder a estas preguntas, los Grandes Maestros nos facilitan la siguiente información:

- Disponemos de un grafo con el estado de la red de carreteras entre los reinos.
- Disponemos de un listado con las ciudades que disponen de una biblioteca.
- Una ciudad se considera que tiene acceso a una biblioteca si existe una biblioteca en la misma ciudad o si los habitantes pueden viajar por carretera hasta otra ciudad con biblioteca.
- El coste de construir una biblioteca es de 150.000 €.
- El coste de re-construir una carretera es de 80.000 €.

Nota: para resolver el ejercicio, **tenéis que generar una visualización del grafo que os permita responder a las cuatro preguntas planteadas**. Después, **recuerda responder a las cuatro preguntas!**

In [3]:

```
# Datos disponibles para el ejercicio

import networkx as nx

# Lista de carreteras transitables
carreteras_ok = [(1,3), (3,4), (2, 7), (2, 8), (5, 6), (9, 10)]

# Lista de carreteras cortadas
carreteras_ko = [(1, 2), (2, 4), (2, 5), (8, 9)]
```

```
# Graf G, que representa la red de carreteras entre las ciudades
G = nx.Graph()

G.add_edges_from(carreteras_ok)
G.add_edges_from(carreteras_ko)

# Lista de ciudades con biblioteca
ciudades_con_biblioteca = [3, 6, 9]
```

In [4]:

```
# Respuesta
```

Ejercicios para la PEC

A continuación encontraréis los **ejercicios y preguntas teóricas que debe completar en esta PEC** y que forman parte de la evaluación de esta unidad.

Ejercicio 1

Pokemon es una saga RPG muy famosa a nivel mundial. La misión de estos juegos es capturar y entrenar a los pokemon, unas criaturas que habitan en todo el continente, para hacerse con el título de maestro de la liga Pokemon. Los Pokemons son criaturas muy diversas, hay muchos tipos diferentes y algunos de ellos pueden evolucionar.

En este ejercicio exploraremos los Pokemons de la primera generación con el dataset `pokemon.csv` . NM

1. ¿Cuántos Pokemons legendarios hay? Cuántos Pokemons hay de cada tipo de evolución (*variable Stage*)?
2. ¿Cuál es el tipo más frecuente? Y el menos?
3. Un nuevo entrenador tiene que escoger entre 3 Pokemons iniciales (Bulbasaur, Charmander y Squirtle) y nos ha pedido que le ayudemos a decidir basándonos en las estadísticas de los pokemons de estos tres tipos (Grass, Fire y Water) . Si nos centramos sólo en las características de ataque y defensa, qué tipo deberíamos recomendar?

Representa las respuestas gráficamente.

Pista: podéis utilizar la función jointplot que hemos visto en el Notebook de teoría. Considerad qué tipo de gráfica, de entre las que ofrece jointplot, se ajusta a los requerimientos del enunciado.

(2.5 puntos)

In [5]:

```
# Respuesta
```

Exercici 2

La ciudad de Barcelona está realizando un estudio para ampliar las áreas verdes de la ciudad. Es por ello que el ayuntamiento de Barcelona requiere de nuestro servicio para construir un mapa **interactivo** que muestre los puntos donde hay árboles y palmeras en la ciudad.

Nos piden crear un mapa utilizando la librería *geoplotlib* que nos debe permitir ver la localización de cada árbol/palmera. EG

Nota 1: Puede obtener los datos del arbolado de zona de la ciudad de Barcelona en el portal Open Data BCN y cargar los datos a partir de un diccionario o un dataframe. Sino también puede cargar el conjunto de datos `arbrat_zona.csv` **(que os proporcionamos junto con el enunciado de la PEC)**.

Nota 2: Es importante que los tipos diferentes de zonas (arbol o palmera) tengan distinto color en el mapa.

(2.5 puntos)

In [6]:

```
# Respuesta
```

Ejercicio 3

Teniendo en cuenta los datos obtenidos en el ejercicio anterior, el ayuntamiento de Barcelona también nos pide llevar a cabo un análisis descriptivo de los árboles. NM

1. ¿Cuántos árboles y cuántas palmeras han plantado cada año? **Ver resultados separados para cada grupo (árbol y palmera).**
2. Adicionalmente, se nos pide generar una gráfica que muestre la **distribución** de los meses en los que se han plantado árboles y palmeras en los últimos 3 años (de 2017 2019).

Nota 1: La columna con la información de la fecha de plantación tiene missing. Por lo tanto, se deben seleccionar sólo aquellas filas con este dato.

Nota 2: Para obtener la información del mes y año a partir de la fecha, puede utilizar la función *DatetimeIndex* de la librería pandas.

Nota 3: La visualización de los datos debería ser atractiva y clara a fin de que los responsables del ayuntamiento puedan sacar conclusiones de ellas.

(2.5 puntos)

In [7]:

```
# Respuesta
```

Ejercicio 4

Juego de Tronos es conocida por las complicadas interacciones entre sus personajes. En este ejercicio trabajaremos con el conjunto de datos `book1.csv` que constituye una red de las relaciones entre los personajes del primer libro de *Juego de Tronos*. EG

Cada fila representa un link entre 2 personajes (Source y Target) y el valor (Weight) indica el número de veces que han interaccionado.

1. Representad gráficamente el grafo y pintad de diferente color cada comunidad detectada.
2. Cuál es el personaje más importante del primer libro de Juego de Tronos?

Nota: La importancia de un nodo dentro de una red se mide con el número de nodos a los que está conectado. Esta característica se conoce con el nombre de **degree_centrality**. Podéis ver este link para más información:

(2.5 puntos)

In [8]:

```
# Datos disponibles para el ejercicio

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx

# Cargamos los datos del fichero book1.csv en un dataframe
book1 = pd.read_csv('data/book1.csv')

# Creamos un objeto grafo vacío
G_book1 = nx.Graph()

# Iteramos para todas las filas del dataset
for _, edge in book1.iterrows():
    G_book1.add_edge(edge['Source'], edge['Target'], weight=edge['weight'])
```

In [9]:

```
# Respuesta
```

Ejercicio Opcional

Mediante la *degree centrality* hemos podido descubrir cuál es el personaje más importante del primer libro de *Juego de Tronos*. Sin embargo, esta saga consta de cinco volúmenes, por lo tanto, tiene sentido que la importancia de los personajes cambie a lo largo de

los cinco libros.

En este ejercicio estudiaremos la evolución de los personajes mediante la red de relaciones de los cinco libros. Trabajaremos con los conjuntos de datos `book1.csv`, `book2.csv`, `book3.csv`, `book4.csv` y `book5.csv` **(que os proporcionamos junto con el enunciado de la PEC)**. `EI`

- ¿Cuál es el personaje más importante de cada libro?
- Muestra gráficamente cómo varía la importancia de Eddard Stark, Arya Stark, Tyrion Lannister, Jon Snow y Cersei Lannister en cada libro.

In [10]:

```
# Respuesta
```

Soluciones a los ejercicios para practicar

Ejercicio 1

Cargad el conjunto de datos `pulitzer-circulation-data.csv` en un dataframe de Pandas. La fuente original de este conjunto de datos es el repositorio de datos de [FiveThirtyEight] (https://github.com/fivethirtyeight/data).

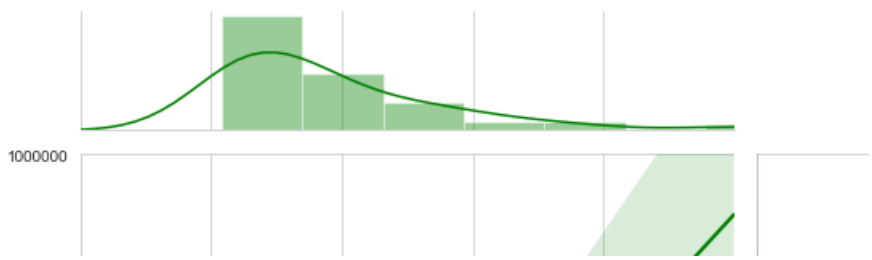
Cread un diagrama de dispersión que permita visualizar las muestras del conjunto de datos de pulitzer según las variables `Daily Circulation 2004` y `Daily Circulation 2013`. Incluid una recta con el ajuste lineal entre las dos variables. En cuanto a los detalles de visualización, limitad la visualización de ambos ejes en el intervalo (0, 1000000), utilice el estilo `whitegrid` de Seaborn, y dibuja los puntos y la línea en color verde.

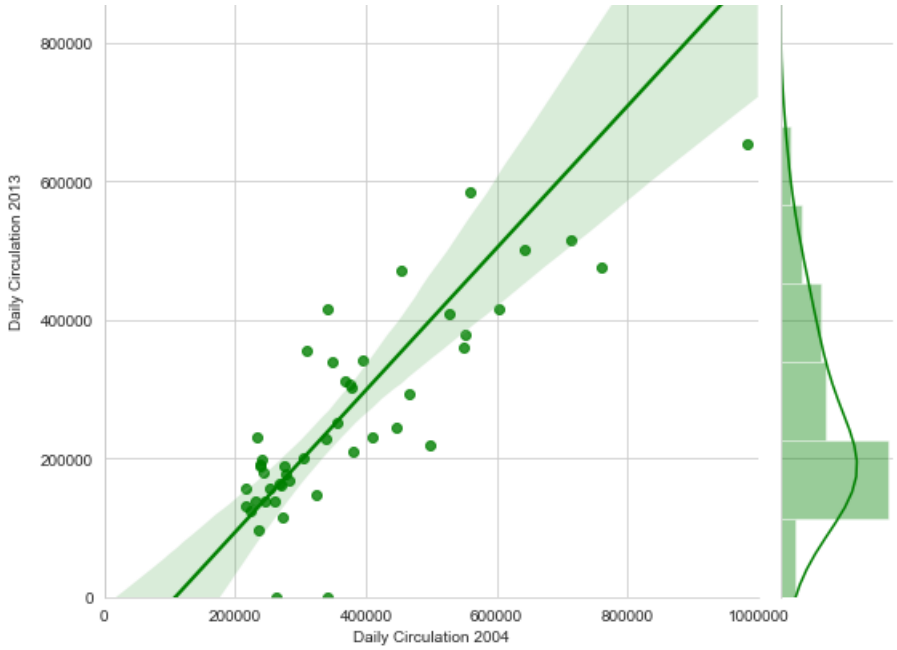
Pista: podéis utilizar la función `jointplot` que hemos visto en el Notebook de explicación. Considerad qué tipo de gráfica, [de entre las que ofrece jointplot] (http://seaborn.pydata.org/generated/seaborn.jointplot.html), se ajusta a los requerimientos del enunciado.

In [11]:

```
# Importamos las librerías
import pandas as pd
import seaborn as sns

# Mostramos las gráficas en el notebook
%matplotlib inline

# Indicamos que queremos utilizar el estilo "whitegrid" de Seaborn
sns.set_style("whitegrid")

# Cargamos los datos del fichero pulitzer-circulation-data.csv en un dataframe
data = pd.read_csv('data/pulitzer-circulation-data.csv')

# Generamos la gráfica de tipo "reg"
g = sns.jointplot("Daily Circulation 2004", "Daily Circulation 2013", data=data,
                  xlim=[0, 1000000], ylim=[0, 1000000],
                  kind="reg",
                  color="g", height=8)
```

```
/Users/alexandraabos/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tu
ple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index,
`arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```





Ejercicio 2

Juego de Tronos es una serie de televisión basada en la famosa saga *Una Canción de Fuego y Hielo* de George RR Martin. Esta serie es conocida por sus complejos escenarios políticos y bélicos, así como las numerosas muertes de personajes.

En este ejercicio trabajamos con el dataset `battles.csv` que nos da información sobre las batallas que han tenido lugar a lo largo de *Juego de Tronos*.

Cread un diagrama de puntos (*scatter*) que permita comparar el tamaño (`attacker_size` vs `defender_size`) de los dos principales ejércitos involucrados en cada batalla. Utilizad la columna `attacker_outcome` para mostrar con diferentes colores el resultado de la batalla por el atacante.

Podéis utilizar la función [`lmplot`] (https://seaborn.pydata.org/generated/seaborn.lmplot.html) de Seaborn para generar la gráfica. Considerad qué atributos nos permiten crear este tipo de visualización.

Viendo la gráfica, podemos afirmar que el tamaño del ejército es determinante en el *outcome* de la batalla?

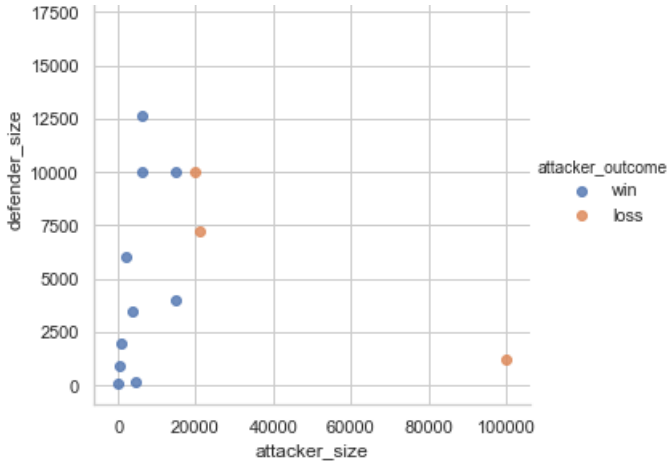
Nota: utilizad el estilo `whitegrid` de Seaborn.

In [12]:

```
# Importamos las librerías
import pandas as pd
import seaborn as sns

# Mostramos las gráficas en el notebook
%matplotlib inline

# Indicamos que queremos utilizar el estilo "whitegrid" de Seaborn
sns.set(style="whitegrid")

# Cargamos los datos del fichero got.csv en un dataframe
got_data = pd.read_csv('data/battles.csv')

# Eliminamos las muestras con valores NaN en las columnas attacker_size o defender_size
got_data = got_data.dropna(subset=["attacker_size", "defender_size"])

# Mostramos la gráfica
sns.lmplot(x="attacker_size", y="defender_size", hue="attacker_outcome", scatter=True, fit_reg=Fals
e, data=got_data)
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0xa1dd20390>
```



Respuesta

Veient la gràfica anterior, no podem afirmar que la mida de l'exèrcit és determinant en l'outcome de la batalla. Com podem observar, hi ha casos on l'atacant perd la batalla tot i la superioritat numèrica de les tropes i a l'inrevés, l'atacant guanya la batalla tot i la superioritat numèrica del bàndol contrari.

Ejercicio 3

En los reinos de Poniente, los Grandes Maestros consideran que todo el mundo debería tener acceso a una biblioteca. Desgraciadamente, después de una gran guerra la red de carreteras que comunican estos reinos ha quedado gravemente afectada. Los Grandes Maestros nos han contratado para resolver las siguientes cuestiones:

1. ¿Qué reinos han quedado sin poder acceder a una biblioteca?
2. ¿Cuál es el número mínimo de bibliotecas que se deberían construir para que todo el mundo vuelva a tener acceso a una biblioteca? (Si no podemos re-construir ninguna carretera)
3. ¿Cuál es el coste mínimo de re-construcción de carreteras a fin de que todo el mundo vuelva a tener acceso a una biblioteca? (Si no podemos construir ninguna biblioteca)
4. Viendo la respuesta de la pregunta 2 y 3, que sería lo más óptimo teniendo en cuenta que queremos minimizar los costes?

Para poder responder a estas preguntas, los Grandes Maestros nos facilitan la siguiente información:

- Disponemos de un grafo con el estado de la red de carreteras entre los reinos.
- Disponemos de un listado con las ciudades que disponen de una biblioteca.
- Una ciudad se considera que tiene acceso a una biblioteca si existe una biblioteca en la misma ciudad o si los habitantes pueden viajar por carretera hasta otra ciudad con biblioteca.
- El coste de construir una biblioteca es de 150.000 €.
- El coste de re-construir una carretera es de 80.000 €.

Nota: para resolver el ejercicio, **tenéis que generar una visualización del grafo que os permita responder a las cuatro preguntas planteadas**. Después, **recuerda responder a las cuatro preguntas!**

In [13]:

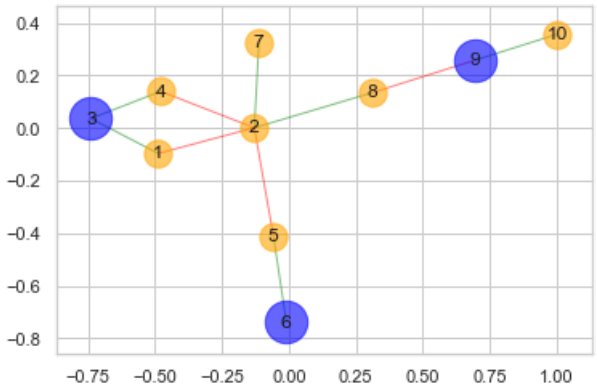
```
# Datos disponibles para el ejercicio

import networkx as nx

# Lista de carreteras transitables
carreteras_ok = [(1,3), (3,4), (2, 7), (2, 8), (5, 6), (9, 10)]

# Lista de carreteras cortadas
carreteras_ko = [(1, 2), (2, 4), (2, 5), (8, 9)]

# Graf G, que representa la red de carreteras entre las ciudades
G = nx.Graph()

G.add_edges_from(carreteras_ok)
G.add_edges_from(carreteras_ko)

# Lista de ciudades con biblioteca
ciudades_con_biblioteca = [3, 6, 9]
```

In [14]:

```
%matplotlib inline

# Calculamos la posición de los nodos usando el algoritmo spring
graph_pos = nx.spring_layout(G)

# Mostramos los nodos del grafo. Mostramos con un tamaño más grande y en azul las ciudades que tie
nen biblioteca
nx.draw_networkx_nodes(G, graph_pos, ciudades_con_biblioteca, node_size=700, node_color='blue', alp
ha = 0.6)

# Mostramos con un tamaño más pequeño y en naranja los otros nodos del grafo
ciudades_sin_biblioteca = list(set(G.nodes()) - set(ciudades_con_biblioteca))
nx.draw_networkx_nodes(G, graph_pos, ciudades_sin_biblioteca, node_size=300, node_color='orange', a
lpha = 0.6)

# Mostramos las etiquetas del nodos, especificando el tamaño y la fuente
nx.draw_networkx_labels(G, graph_pos, font_size=12, font_family='sans-serif')

# Mostramos las aristas del grafo
# Primero, mostramos con color verde las aristas que corresponden a las carreteras que aún son tra
nsitables
nx.draw_networkx_edges(G, graph_pos, edgelist=carreteras_ok, edge_color='green', alpha=0.5)

# Mostramos con color rojo las aristas que corresponden a las carreteras cortadas
nx.draw_networkx_edges(G, graph_pos, edgelist=carreteras_ko, edge_color='red', alpha=0.5)
```

Out[14]:

```
<matplotlib.collections.LineCollection at 0x1a1f474668>
```



Respuesta

Una vez construido el grafo, podemos contestar las preguntas que nos han hecho los Grandes Maestros.

- Qué reinos han quedado sin poder acceder a una biblioteca?

Los reinos que han quedado sin acceso a una biblioteca son: `[2, 7, 8]`.

- Cuál es el número mínimo de bibliotecas que se deberían construir para que todo el mundo vuelva a tener acceso a una biblioteca? (Si no podemos re-construir ninguna carretera)

Dado que los tres reinos que han quedado sin acceso a una biblioteca están conectados entre sí (aunque quedan carreteras que los unen), podemos construir una biblioteca en cualquiera de los tres para que los ciudadanos de estos reinos puedan volver a disfrutar de una biblioteca. Por tanto, el número mínimo de bibliotecas sería **1**.

- Cuál es el coste mínimo de re-construcción de carreteras a fin de que todo el mundo vuelva a tener acceso a una biblioteca? (Si no podemos construir ninguna biblioteca)

Siguiendo el mismo razonamiento anterior, si re-construimos la carretera que conectaba el reino 8 con el reino 9 (que tiene biblioteca), también estaremos habilitando el acceso a los reinos 2 y 7. Por tanto, el coste mínimo para re-construir carreteras sería **150.000 €**.

- Viendo la respuesta de la pregunta 2 y 3, que sería lo más óptimo teniendo en cuenta que queremos minimizar los costes?

En este caso, lo más óptimo sería re-construir una carretera (por ejemplo la que une los reinos 8 y 9). Esto costaría `80.000 €`, mientras que construir una biblioteca nos implica un coste de `150.000 €`.