**FUNDAMENTOS DE PROGRAMACIÓN**

# Ejercicio de Laboratorio: D'Hondt

**Autor:** José A. Troyano. **Revisores:** Mariano González, Fermín Cruz, Belén Vega. **Última modificación:** 30/12/2020

En este proyecto trabajaremos con datos de escrutinios electorales. Estos datos son públicos y podemos descargarlos desde muchas fuentes. Por ejemplo, las ediciones digitales de los periódicos son una buena opción. Los ficheros de entrada estarán en formato CSV y contendrán, en cada línea, el número de votos que ha obtenido un partido en una determinada provincia. Estas son las primeras líneas de un fichero de entrada:

```
Provincia,Partido,Votos
Almería,PP,99917
Almería,PSOE-A,88709
Almería,PODEMOS,29496
Almería,C's,25335
Almería,IULV-CA,11300
Almería,UPyD,4822
```

Además de los datos de escrutinio, también es necesario conocer el número de escaños asignados a cada provincia. Esta información también se encuentra en un fichero CSV. Estas son las primeras líneas de uno de estos ficheros:

```
Provincia,Escaños
Almería,12
Cádiz,15
Córdoba,12
Granada,13
```

Disponemos de conjuntos de datos correspondientes a:

- Elecciones autonómicas de Andalucía (2015 y 2018)
- Elecciones autonómicas de Cataluña (2017)
- Elecciones generales al gobierno de España (2015 y 2016)

A partir de estos datos, generaremos distintos informes y gráficos para poder analizar los resultados electorales desde distintos puntos de vista.

Por último, implementaremos el sistema D'Hondt que es uno de los más utilizados a la hora de asignar escaños a los partidos en función de los votos. Gracias a esta implementación podremos comparar, por ejemplo, el número de escaños que se obtienen con la asignación por provincias (sistema actual en España) frente a una asignación mediante circunscripción única (sumando los votos de todas las provincias). Estas son algunas de las salidas que generaremos:

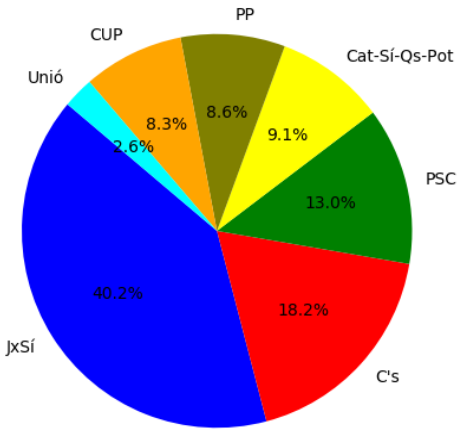


*Figura 1: diagrama de tarta con los votos por partido*

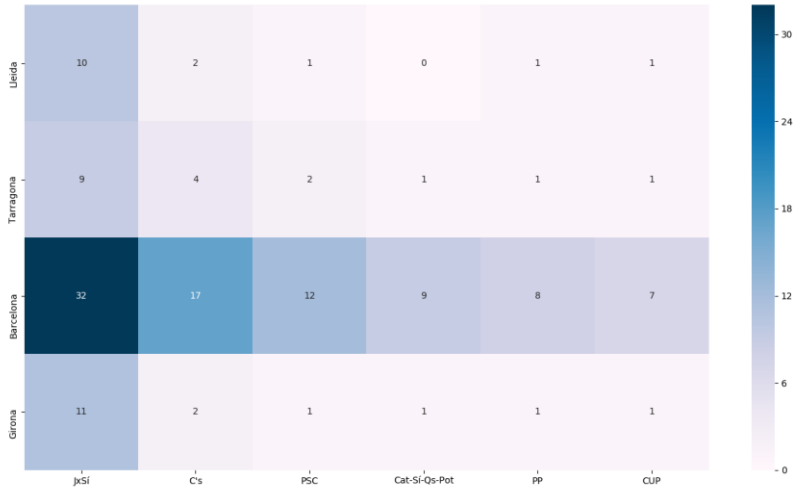


*Figura 2: mapa de calor con los acumulados por partido en cada provincia*

Para almacenar en Python la información de cada una de las líneas se usará la siguiente definición de namedtuple:

```
Votos = namedtuple('Votos', 'provincia,partido,votos')
```

Cree un fichero **dhondt.py** e incluya en él la definición del namedtuple anterior (recuerde que debe importar namedtuple del módulo collections para poder utilizarlo). A continuación, implemente las funciones que se le piden.

1. **lee_escrutinio**: recibe la ruta de un fichero de votos y otro de escaños, ambos en formato CSV codificados en UTF-8. La lectura del primer fichero devolverá una lista de tuplas de votos (provincia, partido, votos) de tipo (str, str, int). Por otro lado, la lectura del segundo fichero devolverá un diccionario con los escaños por provincia de tipo {str: int}.
2. **calcula_provincias**: recibe una lista de tuplas de registros de votos (provincia, partido, votos) de tipo (str, str, int) y devuelve el conjunto de provincias presentes en el escrutinio de tipo {str}.
3. **calcula_partidos:** recibe una lista de registros de votos (provincia, partido, votos) de tipo (str, str, int) y devuelve un conjunto {str} de partidos presentes en el escrutinio.
4. **calcula_diccionario_provincia:** recibe una lista de tuplas de votos (provincia, partido, votos) de tipo (str, str, int) y una provincia de tipo str, y devuelve un diccionario {str: int} con los votos por partido en la provincia dada como parámetro de entrada.
5. **calcula_diccionario_provincias:** recibe una lista de tuplas de votos (provincia, partido, votos) de tipo (str, str, int) y devuelve un diccionario 2D {str: {str: int}} con los votos por provincia y partido. Los diccionarios de dos dimensiones en Python son simplemente diccionarios de una dimensión cuyos valores son, a su vez, diccionarios. En este caso el diccionario de salida tiene la siguiente estructura:
   - clave: provincia -> str
   - valor: diccionario con el valor por partido para la provincia clave -> {str: int}
6. **totales_por_partido:** recibe un diccionario-2D de tipo {str: {str: int}} con los votos por provincia y partido y devuelve un diccionario {str: int} en el que las claves son partidos y los valores la suma de los valores asociados al partido en todas las provincias. En este ejercicio hará falta combinar dos

diccionarios sumando los valores de las claves que aparezcan en ambos. Una forma 'pitónica' de hacer esto es a través de la clase Counter que soporta el operador suma:

```
A = Counter({'a':1, 'b':2, 'c':3})
B = Counter({'b':3, 'c':4, 'd':5})
resultado = dict(A + B)
```

7. **genera_diagrama_tarta:** recibe un diccionario de tipo {str: int} con los valores por cada clave y un número *límite* de tipo int que indicará el número máximo de partidos a mostrar en el diagrama. Esta función generará un diagrama de tarta con los votos por partido (Figura 1). Se usarán las siguientes instrucciones matplotlib para generar el diagrama de tarta:

   ```
   plt.pie(valores, labels=claves, autopct='%1.1f%%', startangle=140)
   plt.axis('equal')
   plt.show()
   ```

   Donde *claves* se refiere a las claves del diccionario (en este caso partidos) ordenadas de mayor a menor según el valor (votos) asociado y *valores* se corresponde con los valores (votos) asociados en el diccionario a las claves (en el mismo orden que la lista de claves).

8. **genera_mapa_calor:** recibe un diccionario-2D con los votos por provincia y partido de tipo {str: {str: int}}, un número *limite_columnas* de tipo entero que indica el número máximo de columnas a mostrar y una variable *fmt* de tipo str con el formato de los valores de cada celda (enteros: 'd', reales: 'f'). Este método genera un mapa de calor con los acumulados por partido en cada provincia (Figura 2):
   - Las filas (provincias) se mostrarán por orden alfabético.
   - Las columnas (partidos) se mostrarán por orden descendente del acumulado de valores.

   Se utilizará el paquete seaborn para generar la gráfica. Seaborn es un *wrapper* sobre matplotlib, menos potente pero mucho más fácil de usar. Para instalar el paquete, abra una ventana de comandos de Anaconda (Anaconda Prompt) y ejecute el siguiente comando:

   ```
   pip install seaborn
   ```

   Para utilizarlo, coloque la siguiente línea al comienzo del módulo dhondt.py:

   ```
   import seaborn as sns
   ```

   Las instrucciones 'seaborn' que tendremos que ejecutar son:

   ```
   nombres_partidos = [partido[:10] for partido in partidos] # Acortar
   g = sns.heatmap(filas, annot=True, fmt=fmt, cmap="PuBu",
                   xticklabels=nombres_partidos, yticklabels=provincias)
   g.set_yticklabels(g.get_yticklabels(), rotation =0)
   g.set_xticklabels(g.get_xticklabels(), rotation =65)
   plt.show()
   ```

   Donde *filas* es una lista de vectores, uno por cada provincia:
   - El vector de cada provincia será una lista con los valores de la tabla para esa provincia y para cada partido.
   - Los partidos se ordenarán siempre de la misma forma para los vectores de todas las provincias.

9. **calcula_tabla_porcentajes:** recibe un diccionario-2D con los votos por provincia y partido de tipo {str: {str: int}} y devuelve otro diccionario-2D con los porcentajes de votos por provincia y partido de tipo {str: {str: float}}. Los porcentajes se calculan dividiendo el voto de cada partido por el total de los votos de la provincia.

10. **calcula_escaños_provincia:** recibe un diccionario con los votos para cada partido de tipo {str: int}, un número de escaños a repartir *total_escaños* de tipo int y un valor de tipo float *exclusión* que

indica el porcentaje mínimo de votos para entrar en el reparto de escaños. Este método devuelve un diccionario con el número de escaños por partido de tipo {str: int} según el método D'Hondt. Solo se tendrán en cuenta en el reparto de escaños aquellos partidos que superen el porcentaje de exclusión. En la legislación española ese porcentaje es el 3%, por esta razón se establece 0.03 como valor por defecto de este parámetro.

El método D'Hondt se basa en el cálculo de cocientes sucesivos para cada partido. En cada momento el cociente para un partido se obtiene con la siguiente fórmula:

cociente = votos / (diputados asignados + 1)

El sistema se basa en la asignación por rondas. En cada ronda (iteración) se asigna un diputado al partido que en ese momento tiene el mayor cociente, y se actualiza el cociente para dicho partido. Se puede encontrar una descripción más detallada, y también ejemplos, en el siguiente artículo de Wikipedia:

https://es.wikipedia.org/wiki/Sistema_d%27Hondt

Seguiremos el siguiente procedimiento:

1. Calcular un diccionario 'diputados' con los partidos como clave y con 0 diputados asignados a cada uno.
2. Calcular la variable 'umbral' con el número de votos mínimo para poder recibir escaños.
3. Calcular un diccionario cocientes con cada partido y el número de votos. Solo consideraremos los partidos que superen el umbral.
4. Para cada ronda (tantas veces como escaños hay en juego):
    - Buscar el partido ganador (el que tenga el cociente máximo).
    - Incrementamos en uno el número de diputados de ese partido ganador (diccionario 'diputados').
    - Actualizamos el cociente de ese partido (diccionario 'cocientes').

11. **calcula_tabla_escaños:** recibe un diccionario-2D con los votos por provincia y partido de tipo {str: {str: int}} (este diccionario se extrae del resultado de la función *calcula_escaños_provincia*), los números de escaños a repartir por cada provincia de tipo {str: int} y un parámetro *exclusión* de tipo float que indica el porcentaje mínimo de votos para entrar en el reparto de escaños. Esta función devuelve un diccionario-2D con los escaños por provincia y partido de tipo {str: {str: int}}.

Cree un fichero **dhondt_TEST.py.** Importe todas las funciones del módulo dhondt. Cargue los datos de los ficheros CSV y muestre en consola los datos leídos. Incluya llamadas a todas las funciones implementadas, mostrando los resultados en la consola.