Procesamiento y Análisis de Datos Astronómicos

2.- Importando datos

R. Gil-Hutton

Marzo 2020

Actividades:

Práctica 1:

- Elegir un archivo de datos astronómicos (encolumnados o no) de no menos de 300-400 líneas, aproximadamente.
- Crear una función de Python que lea ese archivo y lo guarde en un array de Numpy o en una lista.
- Crear una función de Python que guarde en un archivo el array o la lista generada en el punto anterior.

Archivo de texto:

Como ejemplo de archivo de texto a leer usaré **apollo.dat** que lista parámetros orbitales y físicos encolumnados para un grupo de asteroides Apollo:

Open | apollo.dat ~/cursos/proc-datos/actividades/practica-1 | Save

```
16.9 0.15 20171213 119.679500 31.384860 88.005039 22.824829 0.82684276 1.0780769
15.60 0.15 20171213 316.936752 276.893312 337.203701 13.337335 0.33542004 1.2453014
14.23 0.15 20171213 95.597375 127.163691 274.268409 9.380242 0.43612100 1.3675021
16.25 0.09 20171213 310.910083 285.968729 35.636866 6.355259 0.55970045 1.4701844
15.54 0.15 20171213 168.565724 268.010681 346.471912 18.397379 0.60637385 2.2593407
14.85 0.15 20171213 78.867376 325.639757 6.637127 22.210535 0.61437870 1.4609314
16.84 0.15 20171213 293.000353 325.255711 212.928009 16.095200 0.46692837 1.0800616
12.4 0.15 20171213 123.757904 293.091876 63.497998 41.202128 0.53853309 1.8933602
15.2 0.15 20171213 298.130265 267.796427 356.897731 39.833206 0.65026319 1.7759563
17.3 0.15 20171213 201.204050 55.314904 33.102871 9.432972 0.34940531 1.0779005
18.8 0.15 20171213 350.412963 43.479346 349.624853 1.325666 0.76480009 1.8736249
16.0 0.15 20171213 196.642118 61.545840 94.364683 64.003687 0.29914687 1.2901322
17.94 0.15 20171213 64.467168 290.878111 191.216524 23.061198 0.50290175 1.5996644
15.25 0.15 20171213 286.408988 98.265612 74.990303 2.522412 0.71294747 2.1749605
13.87 0.15 20171213 129.474452 209.351113 27.560807 11.558864 0.83769098 2.1598643
14.5 0.15 20171213 27.255705 146.008601 169.312155 24.453231 0.65351045 2.4101608
15.38 0.15 20171213 263.180502 254.068409 129.767464 20.928830 0.35428913 1.4041766
```

Plain Text | Tab Width: 8 | Ln 1, Col 1 | INS

Archivo de texto:

Recordar que el IO siempre se hace mediante strings. Los pasos a seguir para leer el archivo son:

- crear una lista vacía.
- abrir el archivo para lectura.
- leer línea por línea hasta el EOF (cadena nula).
- agregar cada línea leida a la lista.
- cerrar el archivo.

Archivo de texto:

```
In [2]: lista=[]

In [3]: f=open('apollo.dat','r')

In [4]: linea=f.readline()

In [5]: while(linea != ''):
   ...:     lista.append(linea)
   ...:     linea=f.readline()
   ...:

In [6]: f.close()

In [7]: lista[:2]
Out[7]:
['16.9 0.15 20171213 119.679500 31.384860 88.005039 22.824829 0.82684276 1.07807
69\n',
 '15.60 0.15 20171213 316.936752 276.893312 337.203701 13.337335 0.33542004 1.24
53014\n']

In [8]:
```

Archivo de texto:

Recordar que el IO siempre se hace mediante strings. Los pasos a seguir para leer el archivo son:

- crear una lista vacía.
- abrir el archivo para lectura.
- leer línea por línea hasta el EOF (cadena nula).
- sacar el retorno de carro.
- convertir los valores a reales.
- agregar cada línea leida a la lista.
- cerrar el archivo.

Archivo de texto:

```
In [8]: lista=[]

In [9]: f=open('apollo.dat','r')

In [10]: linea=f.readline()

In [11]: while(linea != ''):
    ...:     linea=linea.rstrip('\n')
    ...:     ss=linea.split(' ')
    ...:     ll=[]
    ...:     for ii in ss:
    ...:         ll.append(float(ii))
    ...:     lista.append(ll)
    ...:     linea=f.readline()
    ...:

In [12]: f.close()

In [13]: lista[:2]
Out[13]:
[[16.9,
  0.15,
  20171213.0,
  119.6795,
  31.38486,
  88.005039,
  22.824829,
  0.82684276,
  1.0780769],
 [15.6,
```

Archivo de texto:

Recordar que el IO siempre se hace mediante strings. Los pasos a seguir para leer el archivo son:

- crear una lista vacía.
- abrir el archivo para lectura.
- leer línea por línea hasta el EOF (cadena nula).
- sacar el retorno de carro.
- convertir los valores a reales.
- agregar cada línea leida a la lista.
- cerrar el archivo.
- convertir la lista a un array.

Archivo de texto:

```
In [18]: val=np.array(lista)

In [19]: np.shape(val)
Out[19]: (9239, 9)

In [20]: val[0,:]
Out[20]:
array([1.6900000e+01, 1.5000000e-01, 2.0171213e+07, 1.1967950e+02,
       3.1384860e+01, 8.8005039e+01, 2.2824829e+01, 8.2684276e-01,
       1.0780769e+00])

In [21]:
```

Archivo de texto:

Recordar que el IO siempre se hace mediante strings. Los pasos a seguir para guardar los datos en un archivo son:

- abrir el archivo para escritura.
- escribir los valores de cada línea convertidos a strings.
- cerrar el archivo.

Archivo de texto:

```
In [21]: f=open('apollo.out','w')

In [22]: for ii in range(len(val[:,0])):
    ...:     f.write('{:f} {:f} {:f} {:f} {:f} {:f} {:f} {:f} {:f}\n'.format(val
    ...: [ii,0],val[ii,1],val[ii,2],val[ii,3],val[ii,4],val[ii,5],val[ii,6],val[
    ...: ii,7],val[ii,8]))
    ...:

In [23]: f.close()

In [24]:
```

Archivo de texto:

También es posible operar directamente sobre una lista leyendo desde o escribiendo hacia un archivo todas sus líneas utilizando los métodos .readlines() y .writelines(), respectivamente.

```
In [30]: f=open('apollo.dat','r')

In [31]: lista=f.readlines()

In [32]: f.close()

In [33]: lista[:2]
Out[33]:
['16.9 0.15 20171213 119.679500 31.384860 88.005039 22.824829 0.82684276 1.07807
69\n',
 '15.60 0.15 20171213 316.936752 276.893312 337.203701 13.337335 0.33542004 1.24
53014\n']

In [34]: f=open('apollo.out','w')

In [35]: f.writelines(lista)

In [36]: f.close()

In [37]:
```

Archivo de texto:

Numpy tiene las funciones np.loadtxt() y np.savetxt() que permiten leer y escribir desde y hacia un archivo siempre que todos los datos sean del mismo tipo y el archivo este encolumnado. Ambas funciones tienen parámetros que configuran cómo se lee o escribe el archivo.

```
In [38]: val=np.loadtxt('apollo.dat')

In [39]: np.shape(val)
Out[39]: (9239, 9)

In [40]: val[0,:]
Out[40]:
array([1.6900000e+01, 1.5000000e-01, 2.0171213e+07, 1.1967950e+02,
       3.1384860e+01, 8.8005039e+01, 2.2824829e+01, 8.2684276e-01,
       1.0780769e+00])

In [41]: np.savetxt('apollo.ele',val)

In [42]:
```

Archivo de texto:

Este es un ejemplo del uso de np.loadtxt() y np.savetxt() empleando algunos de los posibles parámetros de esas funciones:

```
In [43]: val=np.loadtxt('apollo.dat',usecols=(8,7,6,0))

In [44]: val[0,:]
Out[44]: array([ 1.0780769 ,  0.82684276, 22.824829  , 16.9       ])

In [45]: np.savetxt('apollo-aeih.dat',val,fmt='%10.7f  %10.8f  %09.6f  %5.2f')

In [46]:
```

Archivo de texto:

Si el archivo tiene tipos de datos mezclados la lectura debe ser más cuidadosa. Por ejemplo, las dos primeras líneas del archivo de elementos orbitales **astorb.dat** mezcla caracteres, enteros y reales:

```
     1 Ceres              L.H. Wasserman   3.34  0.12 0.72 848.4 G?       0   0
 0   0   0   0 79742 6588 20190805  98.535703  73.785341  80.305586 10.593787 0.
07628185   2.76987472 20190731 1.6E-02 -8.6E-05 20190814 2.0E-02 20200826 2.5E-0
2 20270108 2.5E-02 20270108
     2 Pallas             L.H. Wasserman   4.13  0.11 0.66 498.1 m        0   0
 0   0   0  10 72368 7882 20190805  81.027581 310.077864 173.075342 34.833820 0.
23030130   2.77271698 20190731 1.2E-02 -4.5E-05 20190814 1.3E-02 20200705 3.4E-0
2 20280301 3.4E-02 20280301
```

que contiene el número, designación, quién calculó la órbita, parámetros H y G, (B-V), el diámetro, la taxonomía de IRAS, seis códigos enteros, el arco orbital en días, el número de observaciones, la época (YYYYMMDD), anomalía media, argumento del perihelio, longitud del nodo ascendente, inclinación, excentricidad, semieje mayor, fecha y las incertezas del cálculo.

Archivo de texto:

En este archivo si algún dato no se conoce el campo se rellena con espacios:

```
     1 Ceres              L.H. Wasserman   3.34  0.12 0.72 848.4 G?      0   0
 0   0   0   0 79742 6588 20190805  98.535703  73.785341  80.305586 10.593787 0.
07628185   2.76987472 20190731 1.6E-02 -8.6E-05 20190814 2.0E-02 20200826 2.5E-0
2 20270108 2.5E-02 20270108
     2 Pallas             L.H. Wasserman   4.13  0.11 0.66 498.1 m       0   0
 0   0   0  10 72368 7882 20190805  81.027581 310.077864 173.075342 34.833820 0.
23030130   2.77271698 20190731 1.2E-02 -4.5E-05 20190814 1.3E-02 20200705 3.4E-0
2 20280301 3.4E-02 20280301
```

```
       4571 T-3           E. Bowell       17.41  0.15                    0   0
 4   0   0   3    11   11 20190805 125.012445 276.733594  69.383976  5.339595 0.
29316663   2.54450458 20080411 4.0E+04 -4.5E+01 20190814 6.9E+04 20200302 2.0E+0
5 20261128 3.5E+02 20261127
```

lo que imposibilita leer por columnas como hasta ahora y requiere operar sobre cada línea como un string para extraer los datos.

Archivo de texto:

Extracción de asteroides Aten, Apollo y Amor de **astorb.dat**:

```
# para leer astorb.dat
#
lat=[]
lap=[]
lam=[]
f=open('astorb.dat','r')
linea=f.readline()
obj=1

while(linea != ''):
   hh=float(linea[42:47])
   gg=float(linea[47:53])
   arc=int(linea[94:100])
   obs=int(linea[100:105])
   mm=float(linea[114:125])
   per=float(linea[125:136])
   nod=float(linea[136:147])
   inc=float(linea[147:157])
   exc=float(linea[157:168])
   aa=float(linea[168:181])
   qp=aa*(1.-exc)
   qa=aa*(1.+exc)
   num=[obj,aa,exc,inc,nod,per,mm,hh,gg,arc,obs]

   if((aa < 1.)and(qa > 0.983)):
     lat.append(num)
   elif((aa > 1.)and(qp < 1.017)):
     lap.append(num)
   elif((aa > 1.017)and(qp > 1.017)and(qp < 1.3)):
     lam.append(num)

   linea=f.readline()
   obj+=1

f.close()
```

Función Open:

El segundo parámetro en la función open indica el modo en que el archivo se abrirá. Las opciones son 'r' para lectura (el default), 'w' para escritura, 'x' para abrir un archivo de forma exclusiva (si ya existe devuelve error), 'a' para escritura pero agregando al final del archivo, 'b' para modo binario, 't' para modo texto (el default) y '+' para abrir el archivo para lectura y escritura .

- en modo texto el archivo se abre y devuelve o recibe información en strings.
- en modo binario el archivo se abre y devuelve o recibe información en bytes.
- open también acepta otros parámetros (como buffering, encoding, errors, newline, etc.) pero que usualmente no es necesario modificarlos.

Función Open:

- f = open('xxx.txt') es igual a f = open('xxx,txt','rt')
- f = open('xxx.txt','r') es igual a f = open('xxx,txt','rt')
- f = open('xxx.txt','w') es igual a f = open('xxx,txt','wt')
- f = open('xxx.jpg','rb')
- f = open('xxx.jpg','wb')
- f = open('xxx.jpg','w+') y f = open('xxx.jpg','w+b') truncan el archivo.
- f = open('xxx.jpg','r+') y f = open('xxx.jpg','r+b') no truncan el archivo.

Función Open:

Una vez abierto el archivo con la función open() y asignado a un objeto (f en este caso), es necesario cerrar el archivo al terminar de operar con él. Para cerrarlo:

- se utiliza el método .close() del objeto abierto.
- se utiliza el comando with... as ...: que cierra el objeto al terminar el bloque.

```
In [9]: f=open('apollo-aeih.dat','r')

In [10]: apo=f.readlines()

In [11]: f.close()

In [12]: with open('apollo-aeih.dat','r') as f:
    ...:     apo1=f.readlines()
    ...:

In [13]: apo[0]
Out[13]: '1.078076899999999894e+00 8.268427600000000099e-01 2.282482900000000114
e+01 1.689999999999999858e+01\n'

In [14]: apo1[0]
Out[14]: '1.078076899999999894e+00 8.268427600000000099e-01 2.282482900000000114
e+01 1.689999999999999858e+01\n'

In [15]:
```

Lectura y escritura:

Para leer un archivo se utilizan tres métodos: .read(), .readline() y .readlines():

- .read(size=-1) lee un número size de bytes. Si no se pasa un parámetro, es None o size=-1 lee todo el archivo.
- .readline(size=-1) lee como máximo un número size de bytes de la línea. Si sobrepasa el EOL sigue desde el inicio. Si no se pasa un parámetro, es None o size=-1 lee toda la línea.
- .readlines() lee todas las líneas restantes y las devuelve en una lista.

Lectura y escritura:

Para escribir a un archivo se utilizan dos métodos: .write(), y .writelines():

- .write(string) escribe la string al archivo.
- .writelines(lista) escribe lalista de caracteres a un archivo. No agrega EOL asi que deben estar incluidos en la lista para separar líneas.
- existen módulos standard que permiten leer y escribir diferentes tipos de archivos, como tarfile y zipfile para leer archivos tar y zip respectivamente.

Formato de strings:

Cuando se quiere guardar datos en un archivo de texto se deben convertir los valores a string. Para ello se pueden usar formato para determinar como se guardará. El especificador en un format de string es la parte a la derecha de los dos puntos en cada identificador de variable. Por ejemplo:

```
aaa="Mi nombre es {:s},tengo {:d} años y mido {:4.2f} metros".format('José',30,1.8)
```

y el formato del especificador es:

```
format_spec ::=  [[fill]align][sign][#][0][width][,][.precision][type]
fill        ::=  <a character other than '}'>
align       ::=  "<" | ">" | "=" | "^"
sign        ::=  "+" | "-" | " "
width       ::=  integer
precision   ::=  integer
type        ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"
```

Otros módulos y librerías:

- scipy librería con módulos para estadística, álgebra lineal, etc.
- astropy.io.fits o pyfits para leer y operar con imágenes en formato FITS.
- librerias como pandas que permite manejar datos tabulados inhomogéneos.
- otros módulos disponibles en PyPI, por ejemplo:
  - pyPDF2 que permite leer y operar sobre archivos pdf.
  - pyFlux que permite trabajar y analizar series temporales.
  - cartopy que permite producir mapas y procesar datos geoespaciales.
  - Keras para construir redes neuronales.

Funciones:

Funciones utilizadas:

open()	float()	range()	len()
np.array()	np.shape()	np.loadtxt()	np.savetxt()
.readlines()	.writelines()	.rstrip()	.split()
.append()	.readline()	.read()	.close()
.format()	.write()		

Funciones útiles:

np.ones()	np.zeros()	np.eye()	np.diag()
np.arange()	np.linspace()	np.logspace()	np.where()
np.mgrid()	np.ogrid()	np.sort()	np.size()
np.unique()	.tolist()	.tobytes()	.extend()

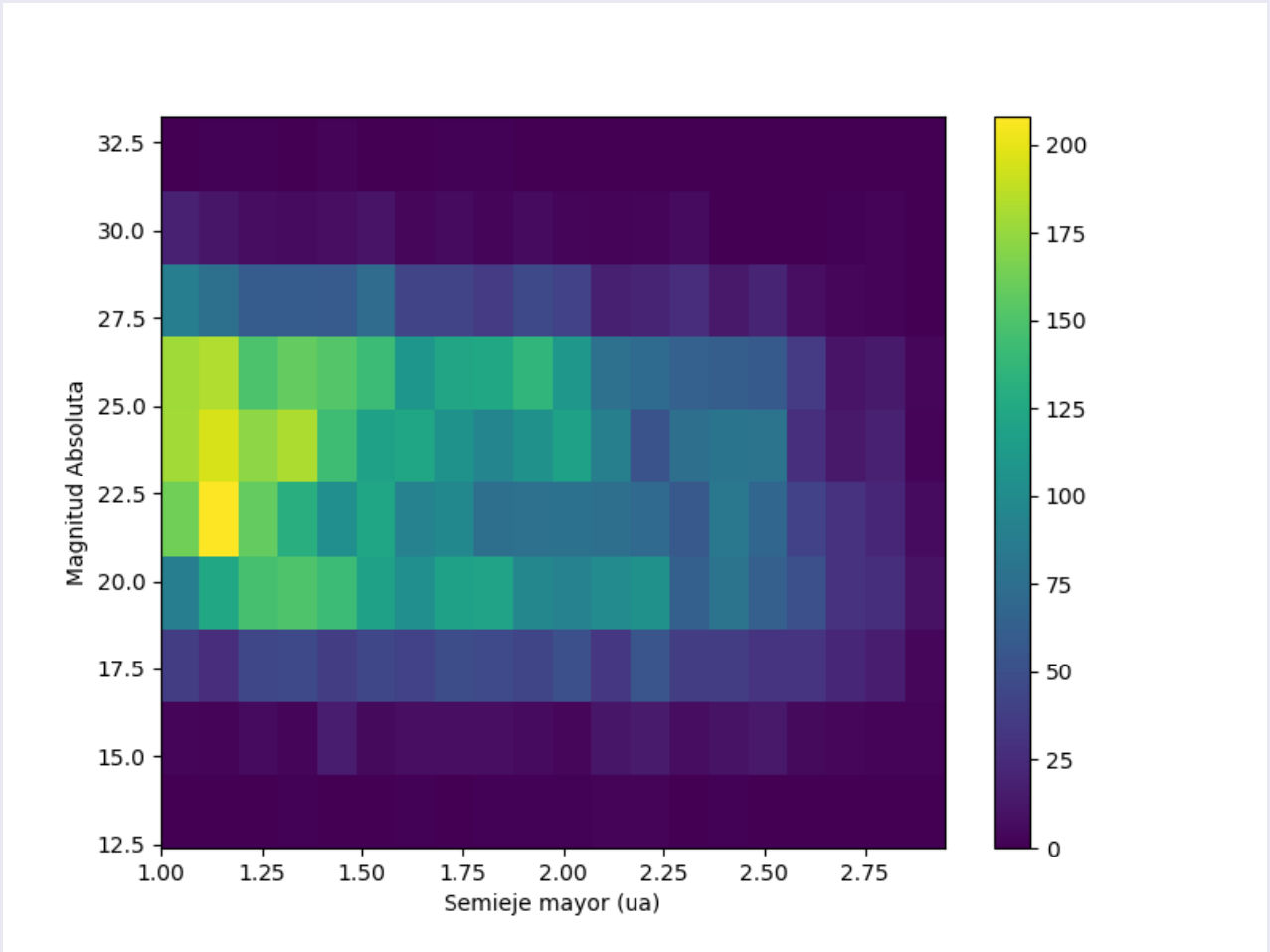
Actividades:

Práctica 2:

- Leer los datos del archivo elegido en la práctica anterior, graficar una columna cualquiera contra otra agregando nombre a los ejes (por ejemplo, Vmag vs. distancia).
- Sobre la figura anterior, graficar en otro color lo mismo pero seleccionando solo los casos que cumplan una cierta condición usando datos de una tercera columna (por ejemplo, Vmag vs. distancia pero para $(B - V) > 0{,}5$).
- Si quisiera representar la primera figura en un histograma bidimensional, cómo lo puedo hacer con matplotlib (ver ejemplo)?

Actividades:

Práctica 2 (cont.):



Actividades:

Práctica 2 (cont.):

- para hacer un histograma, cuáles son las diferencias entre usar plt.bar() y plt.hist()?.
- Cuántos colores hay disponibles en plt.plot() para graficar?
- Cuántos tipos de líneas y marcadores hay disponibles en plt.plot() para graficar?

Entrega

Para la próxima clase

Por consultas: ricardo.gil-hutton@conicet.gov.ar
Grupo de Ciencias Planetarias - CUIM 2