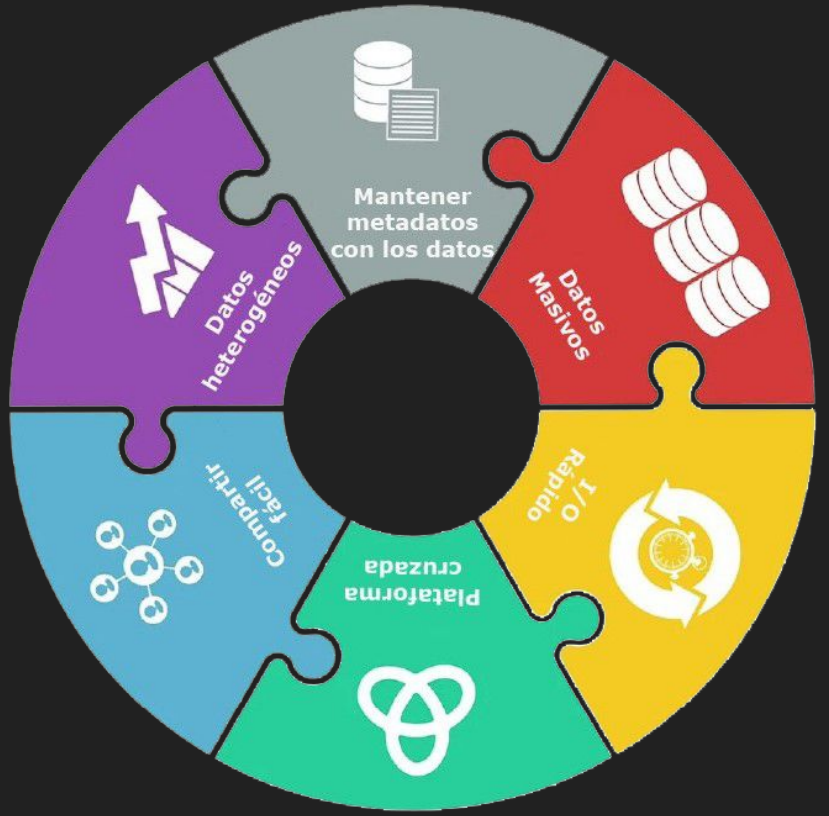
Hierarchical Data Formats Version 5 (HDF5)

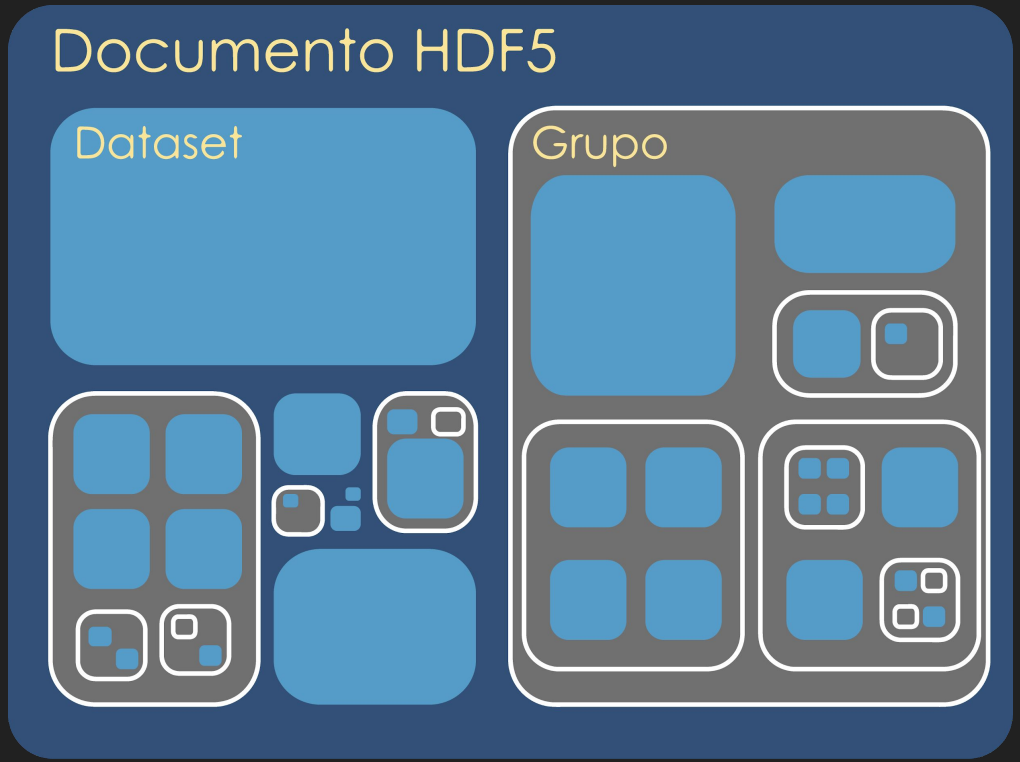
Alvarado Morán Óscar
Bermúdez Marbán Dante
Visualización de la Información
IIMAS, UNAM

¿Qué es?

"Es un formato de archivo de código abierto que admite datos grandes, complejos y heterogéneos. HDF5 utiliza una estructura similar a un 'directorio de archivos' que le permite organizar los datos dentro del archivo de muchas formas estructuradas diferentes, como lo haría con los archivos en su computadora. El formato HDF5 también permite la incrustación de metadatos haciéndolo autodescriptivo."[1]



Estructura



Grupo

↓

Diccionario

Dataset

↓

Numpy Array

¿Dónde se usa actualmente?

- Astronomía
- Dinámica de Fluidos Computacional
- Ciencias de la tierra
- Ingeniería
- Finanzas
- Genómica
- Medicina
- Física

Lectura

```
1.  # Leyendo el archivo.
2.  f = Lectura_hdf5(path_archivo =
    'NEONDSImagingSpectrometerData.h5')

3.  # Obtenemos e imprimimos una lista
    de los datasets contenidos en el
    archivo hdf5
4.  f.ver_estructura()
```

```
- Reflectance

- fwhm

- map info

+ spatialInfo

- wavelength
```

Lectura

```
+ model_weights
+   model_weights/dense
+    model_weights/dense/dense
-     model_weights/dense/dense/bias:0
-     model_weights/dense/dense/kernel:0
+   model_weights/dense_1
+    model_weights/dense_1/dense_1
-     model_weights/dense_1/dense_1/bias:0
-     model_weights/dense_1/dense_1/kernel:0
+   model_weights/dense_2
+    model_weights/dense_2/dense_2
-     model_weights/dense_2/dense_2/bias:0
-     model_weights/dense_2/dense_2/kernel:0
+   model_weights/dense_3
+    model_weights/dense_3/dense_3
-     model_weights/dense_3/dense_3/bias:0
-     model_weights/dense_3/dense_3/kernel:0
```

```
+ optimizer_weights
+   optimizer_weights/Adam
+    optimizer_weights/Adam/dense
+     optimizer_weights/Adam/dense/bias
-      optimizer_weights/Adam/dense/bias/m:0
-      optimizer_weights/Adam/dense/bias/v:0
+     optimizer_weights/Adam/dense/kernel
-      optimizer_weights/Adam/dense/kernel/m:0
-      optimizer_weights/Adam/dense/kernel/v:0
+    optimizer_weights/Adam/dense_1
+     optimizer_weights/Adam/dense_1/bias
-      optimizer_weights/Adam/dense_1/bias/m:0
-      optimizer_weights/Adam/dense_1/bias/v:0
+     optimizer_weights/Adam/dense_1/kernel
-      optimizer_weights/Adam/dense_1/kernel/m:0
-      optimizer_weights/Adam/dense_1/kernel/v:0
+    optimizer_weights/Adam/dense_2
+     optimizer_weights/Adam/dense_2/bias
-      optimizer_weights/Adam/dense_2/bias/m:0
-      optimizer_weights/Adam/dense_2/bias/v:0
+     optimizer_weights/Adam/dense_2/kernel
-      optimizer_weights/Adam/dense_2/kernel/m:0
-      optimizer_weights/Adam/dense_2/kernel/v:0
```

Lectura

```
1.  # Extrayendo datos de reflectancia del archivo hdf5
2.  reflectancia = f.obtener_dataset('Reflectance')
3.  print(reflectancia.shape)

4.  # Extrayendo un pixel de los datos
5.  datos_reflectancia = reflectancia[:,49,392]
6.  datos_reflectancia = datos_reflectancia.astype(float)
```

```
(426, 502, 477)
```

Lectura

```
1.  # Imprimiendo los atributos (metadatos):
2.  print("Atributos: ", f.obtener_atributos('/Reflectance'))
```

```
Atributos:  ['DIMENSION_LABELS', 'Description', 'Scale Factor', 'Unit', 'data ignore value', 'row_col_band']
```

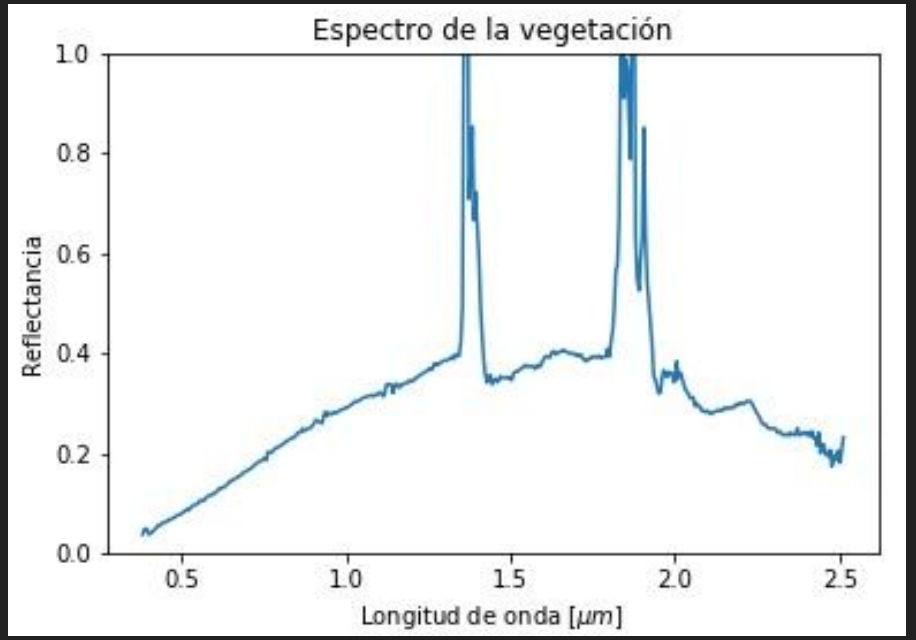
Lectura

```
1.  # Dividiendo los datos por el factor de escala
2.  factor_escala = f.obtener_contenido_atributos("/Reflectance","Scale
    Factor")
3.  datos_reflectancia /= factor_escala

4.  longitud_onda = f.obtener_dataset('wavelength')
5.  datos_longitud_onda = longitud_onda[:]
6.  # Transponer los datos para que la longitud de onda esté en una columna
7.  datos_longitud_onda = np.reshape(datos_longitud_onda, 426)
8.  f.cerrar() # cerramos
```

Usando datasets

```
1.  plt.plot(datos_longitud_onda,
    datos_reflectancia)
2.  plt.title("Espectro de la vegetación")
3.  plt.ylabel('Reflectancia')
4.  plt.ylim((0,1))
5.  plt.xlabel('Longitud de onda [$\mu m$]')
6.  plt.show()
```



Escritura

```
1.  #Escribiendo un nuevo archivo hdf5 conteniendo el espectro
2.  f = Escritura_hdf5("Espectro_vegetacion.h5")
3.  rdata = f.crear_dataset("Espectro_vegetacion", datos =
    datos_reflectancia)
4.  f.crear_atributo("Longitud_onda", data = datos_longitud_onda)
5.  f.cerrar()
```

Referencias

- https://www.hdfgroup.org/solutions/hdf5/
- [1]https://www.neonscience.org/about-hdf5
- http://docs.h5py.org/en/stable/