10. Lectura de datos desde archivos .csv

Archivos csv

Con frecuencia deberemos "cargar" datos almacenados en archivos externos para utilizarlos en cálculos o programas realizados en Matlab/Octave. En la sección *"Descargas"* de la web de la asignatura se encuentran accesibles una colección de archivos en formato `.csv` con datos de diversas variables medidas durante una campaña oceanográfica. El formato `.csv` (*"comma separated values"*) es un formato de texto estándar para el intercambio de datos. Cada fila de un archivo `.csv` corresponde a los valores de distintas variables; dichos valores se encuentran separados por comas; el símbolo del separador decimal es el punto; y las variables tipo texto se recogen entre comillas (dobles o simples).

Prácticamente todos los programas para el tratamiento de hojas de cálculo (Excel, libreOffice Calc, …) son capaces de leer un archivo `.csv` y mostrarlo como hoja de cálculo. A su vez, si hemos rellenado una hoja de cálculo con datos utilizando uno de estos programas es siempre posible exportar los datos a formato `.csv`.

⊕ Los datos que vamos a utilizar como ejemplo han sido medidos con boyas ARGO. En concreto, los archivos *temperatura.csv* y *salinidad.csv* contienen valores de temperatura y salinidad obtenidos a 100 profundidades distintas, medidos en los 109 puntos de muestreo que se muestran en la figura al margen.

Cada uno de estos archivos contiene 100 filas y 109 columnas. Cada columna corresponde a una posición longitud-latitud y cada fila corresponde a una profundidad. El archivo *depth.csv* contiene también 100 filas y 109 columnas que indican la profundidad a que se ha medido cada uno de los valores de temperatura y salinidad de los otros dos archivos. Los archivos *latitud.csv* y *longitud.csv* contienen, cada uno, 109 filas y una columna. La combinación de cada valor de longitud y latitud (de 1 a 109) especifica la localización de cada punto de muestreo. A su vez, el archivo *time.csv* tiene también 109 filas (un valor por fila) que especifican el momento en que la boya comenzó a tomar datos en cada punto.

Si abrimos, por ejemplo, el archivo *temperatura.csv* con LibreOffice (o Excel) veremos los datos de temperatura dispuestos como una hoja de cálculo:

	A	B	C	D	E	F	G	H	I
1	23.2	23.26	23.32	23.3	23.22	23.21	23.17	23.17	23.16
2	23.2	23.26	23.32	23.3	23.23	23.21	23.17	23.17	23.16
3	23.19	23.24	23.32	23.3	23.24	23.21	23.17	23.18	23.16
4	23.19	23.22	23.3	23.29	23.24	23.21	23.18	23.18	23.16
5	23.19	23.22	23.27	23.29	23.24	23.21	23.18	23.17	23.16
6	23.19	23.21	23.24	23.28	23.24	23.21	23.18	23.18	23.16
7	23.19	23.2	23.22	23.27	23.24	23.21	23.18	23.18	23.16
8	23.19	23.2	23.22	23.26	23.24	23.21	23.19	23.18	23.16
9	23.19	23.2	23.22	23.26	23.24	23.21	23.19	23.18	23.16
10	23.19	23.2	23.21	23.25	23.24	23.22	23.19	23.18	23.16
11	23.19	23.2	23.21	23.25	23.24	23.22	23.19	23.18	23.16
12	23.19	23.2	23.21	23.23	23.24	23.22	23.19	23.18	23.16
13	23.19	23.2	23.21	23.22	23.24	23.21	23.18	23.18	23.16
14	23.19	23.2	23.21	23.21	23.24	23.22	23.19	23.18	23.16
15	23.19	23.19	23.21	23.21	23.24	23.22	23.19	23.18	23.16
16	23.19	23.19	23.21	23.2	23.23	23.22	23.19	23.18	23.16
17	23.19	23.19	23.2	23.2	23.23	23.21	23.2	23.18	23.16
18	23.19	23.19	23.2	23.2	23.22	23.21	23.2	23.19	23.16
19	23.19	23.19	23.2	23.2	23.2	23.22	23.2	23.18	23.17
20	23.19	23.19	23.2	23.2	23.2	23.22	23.2	23.18	23.17
21	23.19	23.19	23.2	23.2	23.19	23.22	23.2	23.18	23.17
22	23.19	23.19	23.2	23.19	23.19	23.22	23.2	23.18	23.17
23	23.2	23.19	23.2	23.19	23.18	23.22	23.19	23.18	23.17
24	23.2	23.19	23.19	23.18	23.18	23.22	23.2	23.18	23.17
25	23.2	23.19	23.19	23.17	23.16	23.2	23.2	23.18	23.17
26	23.2	23.19	23.19	23.17	23.14	23.19	23.2	23.18	23.17
27	23.2	23.19	23.19	23.17	23.13	23.17	23.2	23.18	23.17
28	23.2	23.19	23.19	23.16	23.13	23.17	23.19	23.18	23.17

Si abrimos el mismo archivo con el bloc de notas veremos que sus valores están simplemente separados por comas (el símbolo decimal es el punto):

```
23.2,23.26,23.32,23.3,23.22,23.21,23.17,23.17,23.16,23.16,23.15,23.18,23.25,23.2
9,23.28,23.25,23.22,23.2,23.17,23.17,23.17,23.15,23.16,23.19,23.29,23.31,23.29,2
3.26,23.23,23.23,23.23,23.2,23.17,23.15,23.12,23.13,23.15,23.26,23.24,23.38,23.4
5,23.34,23.24,23.2,23.18,23.17,23.17,23.14,23.16,23.18,23.29,23.41,23.42,23.36,2
3.35,23.3,23.29,23.26,23.24,23.24,23.22,23.2,23.13,23.14,23.17,23.22,23.2,23.17,
23.18,23.18,23.18,23.16,23.17,23.17,NaN,23.2,23.21,23.19,23.18,23.15,23.17,23.16
,23.16,23.13,23.15,23.15,23.16,23.17,23.15,23.15,23.13,23.12,23.11,23.08,23.09,2
3.07,23.09,23.2,23.2,23.54,23.29,23.25,23.18,23.14,23.13,23.11,23.11,23.12,23.16
23.2,23.26,23.32,23.3,23.23,23.21,23.17,23.17,23.16,23.16,23.15,23.18,23.24,23.2
9,23.28,23.25,23.22,23.2,23.18,23.18,23.17,23.16,23.16,23.19,23.27,23.31,23.28,2
3.26,23.23,23.24,23.22,23.2,23.18,23.15,23.12,23.13,23.15,23.27,23.19,23.23,23.4
3,23.33,23.24,23.2,23.18,23.17,23.17,23.14,23.16,23.18,23.28,23.33,23.33,23.3,23
.32,23.3,23.28,23.26,23.24,23.24,23.22,23.2,23.14,23.14,23.17,23.21,23.2,23.18,2
3.18,23.18,23.18,23.16,23.17,23.17,NaN,23.2,23.21,23.19,23.18,23.15,23.17,23.16,
23.16,23.14,23.14,23.15,23.17,23.16,23.15,23.14,23.13,23.12,23.11,23.1,23.09,23.
08,23.09,23.17,23.17,23.4,23.29,23.2,23.17,23.14,23.13,23.12,23.11,23.12,23.15
23.19,23.24,23.32,23.3,23.24,23.21,23.17,23.18,23.16,23.17,23.15,23.18,23.24,23.
29,23.27,23.25,23.22,23.21,23.19,23.18,23.17,23.17,23.16,23.19,23.23,23.3,23.27,
23.26,23.24,23.24,23.22,23.2,23.18,23.15,23.13,23.13,23.15,23.26,23.19,23.2,23.3
,23.27,23.24,23.2,23.17,23.17,23.17,23.15,23.16,23.18,23.23,23.23,23.22,23.24,23
.33,23.3,23.29,23.26,23.25,23.24,23.22,23.21,23.16,23.17,23.19,23.2,23.2,23.18,2
3.19,23.19,23.18,23.16,23.17,23.17,NaN,23.2,23.21,23.2,23.18,23.16,23.17,23.16,2
3.16,23.14,23.14,23.15,23.17,23.16,23.15,23.14,23.13,23.12,23.11,23.1,23.09,23.0
8,23.09,23.14,23.16,23.32,23.28,23.18,23.17,23.14,23.13,23.12,23.12,23.12,23.14
23.19,23.22,23.3,23.29,23.24,23.21,23.18,23.18,23.16,23.17,23.15,23.18,23.24,23.
29,23.28,23.25,23.23,23.2,23.19,23.18,23.17,23.17,23.16,23.18,23.2,23.3,23.26,23
.26,23.24,23.24,23.22,23.2,23.18,23.15,23.13,23.13,23.15,23.25,23.18,23.18,23.22
```

Nótese que en algunos lugares aparece escrito `NaN`. Esta expresión es un acrónimo de *"Not a Number"* indicando que en ese punto no se pudo medir la temperatura, y por tanto falta el valor numérico correspondiente. En tales casos es equivalente escribir `NaN` en el archivo, o dejar la casilla en blanco. Escribir otra cosa daría lugar a errores de lectura del archivo.

Lectura de archivos csv con `csvread`

Para leer estos archivos en Matlab en primer lugar hemos de colocarnos en la carpeta que los contiene. Supongamos que los archivos anteriores están en una carpeta llamada `datos`, que a su vez está dentro de la carpeta `c:\users\Mis documentos`. Nos colocamos en primer lugar en esta carpeta mediante:

⊕

```
>> cd "c:\users\Mis documentos"
```

(o bien nos situamos en ella utilizando el navegador que aparece a la izquierda en la pestaña "HOME" de la aplicación de matlab)

Ahor, para leer los datos contenidos en los archivos utilizamos la función `csvread()`, asignando el contenido de cada archivo a una variable:

⊕

```
>> T=csvread("datos/temperatura.csv");
 S=csvread("datos/salinidad.csv");
```

```
lon=csvread("datos/longitud.csv");
lat=csvread("datos/latitud.csv");
z=csvread("datos/depth.csv");
```

De esta forma hemos creado las matrices T, S, lon, lat, y Z, que contienen los datos que se han leido desde los archivos csv originales. Podemos ver, por ejemplo, la dimensión (el tamaño) de la matriz T donde se han cargado los datos de temperatura:

```
>> size(T)
```

```
ans =

   100   109
```

Como señalábamos más arriba, cada una de las 109 columnas de T corresponde a una localización de la boya en un día concreto, y cada fila corresponde a una profundidad. Así, por ejemplo, la columna 15 de la matriz T correspondería a las temperaturas medidas por la boya en la longitud:

```
>> lon(15)
```

```
ans = -38.484
```

y en la latitud:

```
>> lat(15)
```

```
ans =  25.154
```

Podemos combinar en una única matriz los datos de profundidad (`z`) con los de temperatura (`T`) en esa posición:

```
>> zt=[z(:,15),T(:,15)]
```

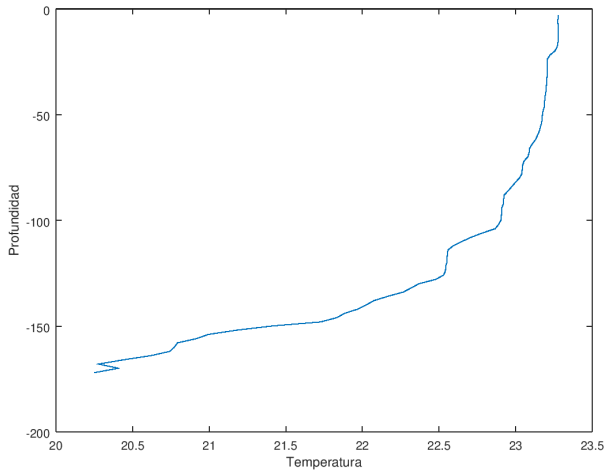
```
zt =

    2.9000   23.2790
    3.9000   23.2770
    5.9000   23.2750
    7.9000   23.2770
    9.9000   23.2780
   11.9000   23.2780
   13.9000   23.2780
   15.9000   23.2760
   17.9000   23.2710
   19.9000   23.2560
```

```
 21.9000    23.2230
 23.9000    23.2060
 25.9000    23.2070
 28.0000    23.2070
 29.9000    23.2060
 31.9000    23.2050
 33.9000    23.2030
 35.9000    23.2010
 37.9000    23.1990
 39.9000    23.1960
 41.9000    23.1910
 43.9000    23.1890
 45.9000    23.1890
 47.9000    23.1800
 49.9000    23.1750
 51.9000    23.1720
 53.9000    23.1700
 55.9000    23.1610
 57.9000    23.1540
 59.9000    23.1420
 61.9000    23.1290
 63.9000    23.1080
 65.9000    23.0910
 67.9000    23.0880
 69.9000    23.0800
 71.9000    23.0550
 73.9000    23.0460
 75.9000    23.0430
 77.9000    23.0410
 79.9000    23.0250
 81.9000    23.0000
 83.9000    22.9760
 85.9000    22.9530
 87.9000    22.9260
 89.9000    22.9220
 91.9000    22.9210
 93.9000    22.9110
 95.9000    22.9080
 97.9000    22.9070
 99.9000    22.9040
101.9000    22.8890
103.9000    22.8670
105.9000    22.7850
107.9000    22.7120
109.9000    22.6510
111.9000    22.5950
113.9000    22.5590
115.9000    22.5540
117.9000    22.5530
119.9000    22.5510
121.9000    22.5430
123.9000    22.5420
125.9000    22.5290
127.9000    22.4780
129.9000    22.3700
```

```
  131.9000    22.3190
  133.9000    22.2660
  135.9000    22.1650
  137.9000    22.0760
  139.9000    22.0260
  141.9000    21.9720
  143.9000    21.8850
  146.0000    21.8320
  148.0000    21.7270
  149.9000    21.4130
  151.9000    21.1790
  153.9000    20.9950
  155.9000    20.9170
  157.9000    20.7930
  159.9000    20.7730
  161.9000    20.7440
  163.9000    20.6220
  165.9000    20.4420
  167.9000    20.2770
  169.9000    20.4070
  172.0000    20.2500
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
       NaN         NaN
```

`z` es la profundidad en metros y `T` es la temperatura (en ºC) medida a esa profundidad; como podemos apreciar, a partir de 172 metros no hay datos disponibles. En matlab es sencillo hacer una representación gráfica de estos datos. El código siguiente representa la temperatura en el eje X y la profundidad en el eje Y (nótese que cambiamos el signo del vector `z`, de forma que la profundidad se mida “hacia abajo”):

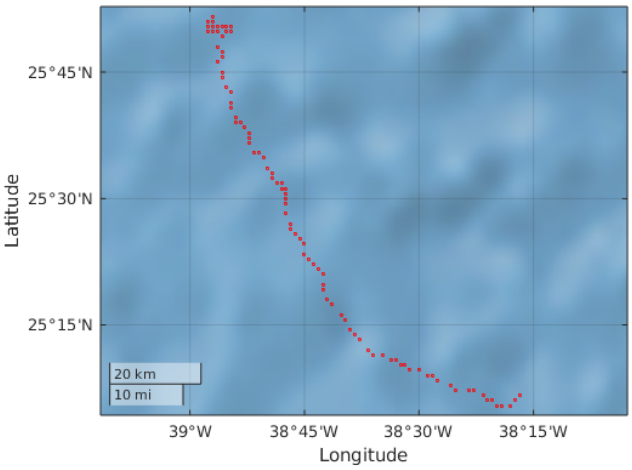
```
>> plot(T(:,15),-z(:,15))
 xlabel("Temperatura")
 ylabel("Profundidad")
```



En esta gráfica podemos apreciar claramente que a medida que aumenta la profundidad disminuye la temperatura.

Matlab ofrece también la posibilidad de dibujar mapas. Para ello es preciso descargar el *Mapping Toolbox* (paquete de aplicaciones para representación geográfica) y al menos un mapa base. La descarga de ambos componentes puede hacerse directamente desde la aplicación matlab (al menos a partir de la versión R2017b): dentro de la pestaña “HOME” pinchamos en Add-Ons -> Get Add-Ons y en el buscador de la ventana que aparece escribimos “basemap”. Para que funcione tenemos que estar conectados a internet, en cuyo caso el servidor de Mathworks nos presentará diversas aplicaciones y complementos, entre los que se encuentran el *Mapping Toolbox* y los mapas base que necesitamos. Basta pinchar en el nombre de la aplicación o mapa que queramos descargar para que matlab nos ofrezca la opción de descargar e instalar. Matlab tiene cinco mapas base (*landcover, colorterrain, grayterrain, grayland* y *bluegreen*). Podemos descargarlos los cinco (tarda bastante) o uno solo de ellos. El mapa que veremos a continuación usa como mapa base el *landcover*. Una vez descargados e instalados el *Mapping Toolbox* y el mapa base, la siguiente sintaxis dibuja los 109 puntos de muestreo *(lat,lon)* en los que la boya ha medido la temperatura y la salinidad a distintas profundidades:

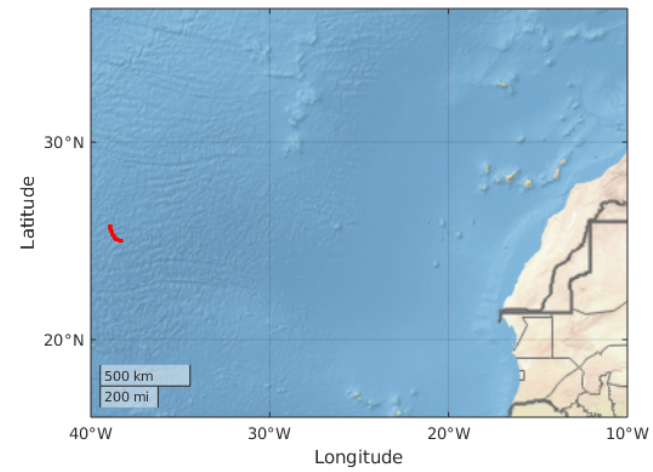
```
>> geoscatter(lat, lon, 3, 'red')
 geobasemap('landcover')
```



En la función `geoscatter` hemos especificado los vectores `lat` y `lon` con las latitudes y longitudes respectivas de los puntos de muestreo. El valor 3 es simplemente el tamaño de cada punto, y `'red'` es el color que usamos para la representación. Por defecto matlab centra el mapa en la región en la que se observan los puntos.

Si queremos ampliar el mapa anterior para situar la región de muestreo y poder hacernos una mejor idea de donde se encuentra, podemos cambiar los límites de latitud y longitud con los que se ha trazado el mapa, de forma que se incluyan las islas Canarias y parte de la costa africana. Para ello basta ejecutar la linea siguiente a continuación de las anteriores.

```
>> geolimits([25,28],[-40,-10])
```



Observamos, pues, que los puntos de muestreo se encuentran al este y ligeramente al sur de las islas Canarias.

Tratamiento de los valores perdidos

Como señalábamos más arriba, en los archivos de datos de temperatura y salinidad faltan valores que han sido sustituidos por el código de valor perdido `NaN`. Estos valores pueden no haberse medido por fallos del sensor, por fallos de la batería de la boya o por cualquier otra causa. La presencia de valores perdidos en un conjunto de datos generalmente es causa de problemas. Si quisiéramos, por ejemplo, calcular la temperatura media en la columna de agua correspondiente al punto de muestreo número 15, podríamos utilizar la función `mean`:

```
>> mean(T(:,15))
```

```
ans = NaN
```

pero, cotras omo vemos, la función nos devuelve `NaN` (*Not a Number*). Ello se debe a que la presencia de valores perdidos en el conjunto de datos impide calcular la media directamente (como hay valores que no se conocen, matlab no puede calcular la media usando dichos valores). Esto ocurre con muchas funciones. Por ejemplo, si quisiéramos calcular la media sumando los valores y dividiendo por el número total de valores válidos (no perdidos) y aplicamos para ello la función `sum`, obtenemos el mismo resultado que antes:

```
>> sum(T(:,15))
```

```
ans = NaN
```

Para poder calcular la media (o la suma) en presencia de valores perdidos, hay que utilizar funciones que ignoren dichos valores y calculen la media (o la suma) con los valores no perdidos. Las funciones `nanmean` y `nansum` hacen exactamente eso:

```
>> nanmean(T(:,15))
```

```
ans =

   22.5908
```

```
>> nansum(T(:,15))
```

```
ans =

   1.9428e+03
```

Si quisiéramos saber cuántos valores perdidos hay en el vector `T(:,15)` podemos utilizar la función `isnan()`. Esta función vale 1 para los valores perdidos y 0 en otro caso. Si, por ejemplo consideramos el vector:

```
>> v=[1 2 3 NaN 4 5 NaN];
```

al aplicar la función `isnan()` obtenemos:

```
>> isnan(v)
```

```
ans =

  0  0  0  1  0  0  1
```

es decir, un 1 donde había `NaN`s y un 0 donde no había `NaN`s. Por tanto, si queremos contar el número de valores perdidos en un vector, bastará con hacer:

```
>> sum(isnan(v))
```

```
ans =  2
```

y, análogamente, si queremos contar el número de valores no perdidos:

```
>> sum(~isnan(v))
```

```
ans =  5
```

Asimismo, si queremos seleccionar sólo aquellos valores de `v` que no están perdidos, podemos ejecutar:

```
>> v(~isnan(v))
```

```
ans =

   1   2   3   4   5
```

(podemos leer esta última expresión como que le estamos pidiendo a matlab: *"muéstrame los valores*

de `v` tales que no están perdidos")

En el caso de nuestro vector de temperaturas `T(:,15)`, el número de valores perdidos que contiene es:

```
>> sum(isnan(T(:,15)))
```

```
ans =  14
```

y el número de valores no perdidos:

```
>> sum(~isnan(T(:,15)))
```

```
ans =  86
```

Combinando todo lo anterior, si quisiéramos calcular la media de `T(:,15)` sin utilizar las funciones `nanmean()` ni `nansum()` podemos utilizar la siguiente sintaxis:

```
>> Temp= T(:,15);            % Valores de temperatura en el punto 15
 noPerdidos=~isnan(Temp);  % Vector que identifica los valores no perdidos
 n=sum(noPerdidos);        % Total de valores no perdidos
 media=sum(Temp(noPerdidos))/n
```

```
media =  22.591
```

Lectura de archivos csv con `readtable`

Para leer el archivo *time.csv* hemos de proceder de manera diferente. Si lo abrimos con LibreOffice o Excel vemos que su contenido son fechas y horas:

	A	B
1	2013-02-18 12:28:44	
2	2013-02-18 14:34:01	
3	2013-02-18 16:59:21	
4	2013-02-18 19:25:21	
5	2013-02-18 21:54:55	
6	2013-02-19 00:18:42	
7	2013-02-19 02:23:02	
8	2013-02-19 04:27:01	
9	2013-02-19 06:36:58	
10	2013-02-19 08:43:33	
11	2013-02-19 11:15:33	
12	2013-02-19 13:15:51	
13	2013-02-19 15:41:59	
14	2013-02-19 17:48:14	
15	2013-02-19 19:34:56	
16	2013-02-19 21:43:29	
17	2013-02-19 23:30:37	
18	2013-02-20 01:45:57	
19	2013-02-20 03:31:01	
20	2013-02-20 05:12:26	
21	2013-02-20 06:54:10	
22	2013-02-20 09:28:25	

En particular, podemos observar que aparecen los símbolos "-" y ":" como separadores para marcar las fechas y las horas. La función `csvread` es capaz de leer ficheros que contengan solamente números; si el archivo contiene algún otro símbolo como en este caso, matlab nos devuelve un error:

```
>> file=fopen("datos/time.csv")
 time=csvread(file);
```

```
Error using dlmread (line 147)
Mismatch between file and format character vector.
Trouble reading 'Numeric' field from file (row number 1, field number 1) ==>
"2013-02-18 12:28:44"\n

Error in csvread (line 48)
    m=dlmread(filename, ',', r, c);
```

Para leer archivos que contengan caracteres no numéricos podemos utilizar la función `readtable`:

⊕

```
>> fh=readtable("datos/time.csv");
```

Cuando se usa esta función, el objeto que se crea (`fh` en nuestro caso) es un objeto de clase `table`, que tiene algunas particularidades específicas de esta clase (en esencia una tabla en matlab es un contenedor con variables de distinta naturaleza ordenadas a modo de hoja de cálculo). Para convertir una tabla en una matriz estándar usamos la función `table2array`:

```
>> FechaHora=table2array(fh);
```

Como ventaja adicional, la función `table2array` identifica que la variable *FechaHora* que hemos creado es de clase `datetime`, que es la clase específica que le permite a matlab manejar adecuadamente variables de fecha y hora. Si le pedimos a matlab que nos muestre los diez primeros valores de esta variable obtenemos la salida siguiente:

```
>> FechaHora(1:10)
```

```
ans =

  10×1 datetime array

   2013-02-18 12:28:44
   2013-02-18 14:34:01
   2013-02-18 16:59:21
   2013-02-18 19:25:21
   2013-02-18 21:54:55
   2013-02-19 00:18:42
   2013-02-19 02:23:02
   2013-02-19 04:27:01
   2013-02-19 06:36:58
   2013-02-19 08:43:33
```

y en particular podemos comprobar que la fecha y hora a que se tomaron los datos en el punto 15 de muestreo fue:

```
>> FechaHora(15)
```

```
ans =

  10×1 datetime array

   2013-02-19 19:34:56
```