

# Case study: Plotting the Track of Hurricane Sandy

## Download

Download the icons for this session from the link below. Create a sub-directory called `training` inside your Metview home directory, `$HOME/metview`, and save the `.tar.gz` file there. Then, from Metview, navigate to this folder and right-click on the `tar.gz` icon; choose **Extract** to uncompress the files into their folder. Work from there. The main instructions are written for participants of the [Data analysis and visualisation using Metview](#) training course at ECMWF. Any files to be copied from the file system will also be downloadable from this page.

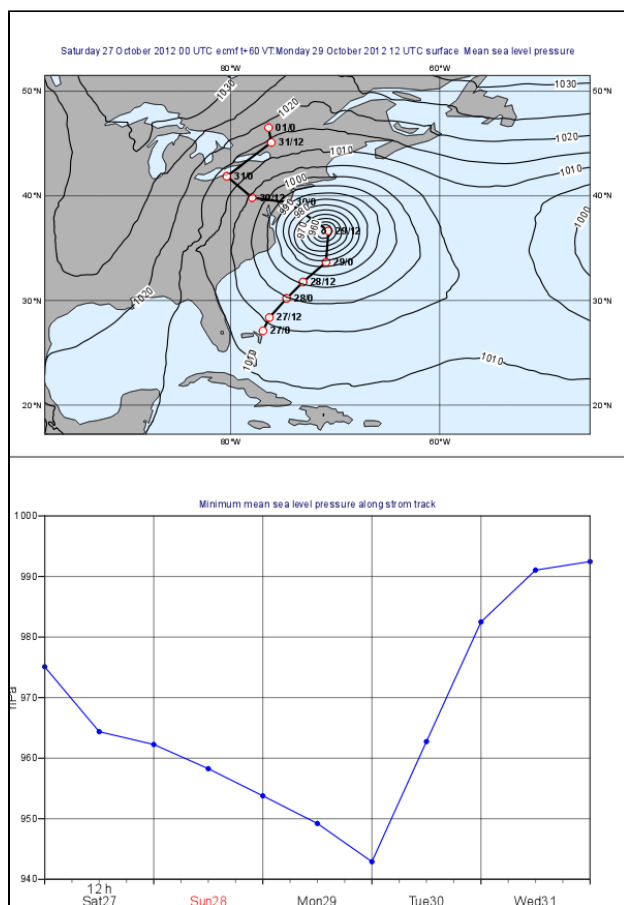
### File

### Modified

File sandy track.tar.gz

Mar 19, 2015 by Iain Russell

## Case description



In this exercise we will use Metview to produce the plots shown above:

- the plot at the top shows a map with mean sea level pressure forecast fields overlayed with the track of Hurricane Sandy.
- the plot at the bottom contains a graph chart showing the evolution of the minimum of the mean sea level pressure forecast along the storm track.

We will work both interactively with icons and write Macro code. First, we will create the two plots independently then align them together in the same page.



This exercise involves:

- reading and visualising GRIB and CSV data
- plotting text labels with [Symbol Plotting](#)
- plotting curves on maps and xy-views
- using the date functions in Macro
- performing fieldset computations in Macro

# Creating the map plot

## Setting the View

With a new *Geographical View* icon, set up a cylindrical projection with its area defined as


```
South/West/North/East: 17/-97/51/-45
```

Set up a new *Coastlines* icon with the following:

- the land coloured in *grey*
- the sea coloured as *#dcf0ff* (or your favourite shade of blue)

## Plotting the Mean Sea Level Pressure field

Plot the GRIB file *sandy\_msl.grib* into your view using a new *Contouring* icon. Plot black isolines with an interval of 5 hPa between them. Animate through the fields to see how the forecast is evolving.

 The fields you visualised were taken from the model run at 2012-10-27 0UTC and containing 12 hourly forecast steps from 0 to 120 hours.

## Plotting the storm track

The storm track data is stored in the CSV file called *sandy\_track.txt*. If you open this file you will see that it contains the date, time and geographical coordinates of the track points.

Create a new *Table Visualiser* icon and set it to visualise your CSV file:

- set the plot type to *geo points*
- select the columns storing the latitude and longitude by their *index* (column indexing start at 1)
- carefully specify the table delimiter and header information by setting
  - **table\_delimiter** to *whitespace* (enter a space character here)
  - **table\_combine\_delimiters** to *on* (it means we allow multiple whitespaces as separators)
  - **table\_header\_row** to *0* (it means do not have a header in the file)

Now drag your *Table Visualiser* icon the plot to overlay the pressure forecast with the track.

## Customising the storm track

The storm track in its current form needs some customisation. Use a *Graph Plotting* icon for it by setting the

- the track line to black and thick
- the track points to be white filled circles (their marker index is 15) with red outline.

## Plotting date/time labels onto the track

To finalise the track plot we will add the date/time labels to the track points. This could be done with a *Symbol Plotting* icon by specifying the list of labels we want to plot into the map. However, to make it reusable for other datasets as well, we will do it programmatically by using Metview *Macro*.

Create a new *Macro* and edit it. First, read the CSV file in with the *Table Reader*:

```
tbl = read_table(  
  table_delimiter      :      " ",  
  table_combine_delimiters :      "on",  
  table_header_row      :      0,  
  table_filename       :      "sandy_track.txt"  
)
```

 As you can see we specified the table delimiters exactly in the same way as we did for the *Table Visualiser* icon.

In the code above, the object referenced by variable `tbl` contains all the columns from the CSV file. Now read the date and time (from the first two columns) into separate *vectors*:


```
val_date=values(tbl,1)
val_time=values(tbl,2)
```

Next, we build the list of labels. Each label is made up from a day and an hour part separated by a slash. We convert the date into a string and then take the last two characters to get the day. Use this loop to construct the list of labels:

```
labels=nil

for i=1 to count(val_date) do
  dPart = substring(string(val_date[i]),7,8)
  tPart = val_time[i]
  label = " " & dPart & "/" & tPart
  labels = labels & [label]
end for
```

Next, define a [Symbol Plotting](#) visual definition using the *text* mode.


 [Symbol Plotting](#) in *text* mode is used to plot string values to the positions of the dataset it is applied to. The rule is that the first string in the list defined by `symbol_text_list` goes to the first data position, the second one to the second position and so on.

The code we need to add is like this:

```
sym = msymb(
  symbol_type      :      "text",
  symbol_text_font_colour : "black",
  symbol_text_font_size: "0.3",
  symbol_text_font_style: "bold",
  symbol_text_list   :      labels
)
```

We finish the macro by returning our Visual Definition.

```
return sym
```

 By **returning** the visual definition our Macro behaves as if it were a real [Symbol Plotting](#) icon.

Save the Macro and drag it into the plot to see the labels appearing along the track.

## Creating the graph plot

### Setting the View

With a new [Cartesian View](#) icon, set up a view to cater for the graph showing the mean sea level pressure values in hPa units by setting

- the x-axis type to *date*
- the x axis minimum to *2012-10-27* and its maximum to *2012-10-31*
- the y-axis label to *hPa*
- the y-axis minimum value to *940* and its maximum to *1000*

### Computing the minimum pressure along the track

Since this task is fairly complex we will use a Macro for it. The idea goes like this:

- we read the track points from the CSV file
- we define a lat-lon box around each point
- we read the forecast mean sea level data for the box for the corresponding time
- we compute the minimum of the pressure in the box
- from these minimum values we can build the curve data to plot.

Create new *Macro* and edit it. First, read the CSV file in the very same way as before. However, this time, on top of date and time, we also need to read latitude and longitude into [vectors](#):

```
val_lon=values(tbl,3)
val_lat=values(tbl,4)
```

Next, read in the GRIB file containing the mean sea level forecast:

```
g=read("sandy_mslp.grib")
```

The curve data requires two lists: one for the dates and one for the values. First we initialise these lists:

```
trVal = nil
trDate = nil
```

Now the main part of the macro follows: we will loop through the track points and build the curve dataset. We will use a loop like this:

```
for i=1 to count(val_date) do
    ... your code will go here ...
end for
```

Within the loop first construct an area of e.g. 10 degrees wide centred on the current track point.



Remember an area is a list of South/West/North/East values. The coordinates of the current track point are `val_lat[i]` and `val_lon[i]`.

Next, read the forecast data for the current forecast step and the area you defined (supposing your area is called `wbox`):

```
p=read(
    data: g,
    step: (i-1)*12,
    area : wbox
)
```

Here we used the fact the forecasts steps are stored in hours units in the GRIB file.

Next, compute the minimum of the field in the subarea using the `minval()` macro function:

```
pmin=minvalue(p)
```

Finally, build the list for the values (scaling Pa units stored in the GRIB to hPa units):

```
trVal= trVal & [pmin/100]
```

And also build the list of dates:

```
dt = date(val_date[i]) + hour(val_time[i])
trDate = trDate & [dt]
```

Having finished the body of the loop the last step in our Macro is to define an [Input Visualiser](#) and return it. The code we need to add is like this:

```
vis = input_visualiser (
    input_x_type      : "date",
    input_date_x_values : trDate,
    input_y_values     : trVal
)

return vis
```



By **returning** the visualiser our Macro behaves as if it were an *Input Visualiser* icon.

Now visualise your *Cartesian View* icon and drag your Macro into it.

## Customising the graph

Customise the graph with a *Graph Plotting* icon by setting the

- the line thicker
- the points to be blue filled circles (their marker index is 15) with a reasonable size.

## Creating a title

Define a custom title as shown in the example plot with a new *Text Plotting* icon.

## Putting it all together

With a new *Display Window* icon design an A4 portrait layout with two views: your *Geographical View* icon should go top and your *Cartesian View* icon into the bottom. Now visualise this icon and populate the views with the data.

## Extra Work

### Adding new curves to the x-y plot

On top of the minimum pressure try to add the maximum and average pressure to the graph plot. Use a different colour to each curve and add a custom legend as well.

Hints:

- first, just try to add your *Graph Plotting* definition to the Macro. In the end return both the *Input Visualiser* and the *Graph Plotting* as a list like this

```
return [vis,graph]
```

If you visualise the Macro your *Graph Plotting* settings will be directly applied to the resulting curve.

- next, compute the maximum of the pressure (with the `maxvalue()` function) in the loop and store its values in another list. Build an input visualiser out of it (e.g. call it `vis_max`). Add a *Graph Plotting* for it (e.g. call it `graph_max`) using a different colour. In the end you need to return a longer list like this:

```
return [vis,graph,vis_max,graph_max]
```

- the average pressure curve (with the `average()` function) can be derived in a very similar manner
- add a *Legend* with *disjoint mode*. Set **legend\_text\_composition** to *user\_text\_only* and carefully set the **legend\_user\_lines** to provide a textual description to each curve in the legend. Add your legend to the back of the list you return from the Macro.

## Doing the whole task in Macro

Try to write a single Macro that is doing all the tasks in one go and directly produces the composite plot with the map and graph in the end