



# Advanced Analysis

This set of tutorials introduces some more advanced analysis concepts, suited for advanced Debrief users or those using niche analysis capabilities.

---

[www.debrief.info](http://www.debrief.info)

---

Ian Mayo



# Contents

## Contents

<b>4 Advanced Analysis</b>	<b>2</b>
4.1 Synchronizing Video, Audio, and Images	2
4.1.1 Reviewing Files and Loading Video File	2
4.1.2 Controlling Plot from Media Player	5
4.1.3 Controlling Video from Time Controller	8
4.1.4 Using Image Viewer	8
4.2 Plot in Unit Centric View	14
4.2.1 Loading and Assigning Tracks	15
4.2.2 Using Snail Paint Mode	17
4.2.3 Experimenting with Local Grid	20
4.2.4 Experimenting with Range Rings	21
4.3 Boost Performance Via Lightweight Tracks	23
4.3.1 Inspecting Lightweight Track Data	23
4.3.2 Opening Lightweight Track	25
4.3.3 Converting Lightweight Track to Normal Track	31
4.3.4 Assigning Lightweight Tracks as Primary Track and Secondary Track	33
4.3.5 Creating Range Plot of Multiple Lightweight Tracks against Primary Track	35
4.3.6 Converting Normal Track to Lightweight Track	37
4.4 Paste REP from Clipboard	39
4.4.1 Pasting REP Data from Text Editor	43
4.4.2 Pasting REP Data from MS Excel File	47
4.4.3 Working with Annotations	50
4.5 Conclusion	54
<b>5 Scripting in Debrief</b>	<b>55</b>
5.1 Scripting Perspective Overview	55
5.2 Opening Scripting Perspective	56
5.3 Setting Path for Scripts Folder	58
5.4 Understanding Scripting Perspective UI	61
5.5 Recording Scripts	64
5.6 Using Debrief Objects in Scripting	69
5.7 Creating Scripts	73
5.7.1 Creating Script in File	73
5.7.2 Configuring Script to Create Command Button	79
5.7.3 Creating Keyboard Shortcut for Script	80

5.8 Debugging in the Debug Perspective	81
5.9 Conclusion	86

## 4 Advanced Analysis

This chapter contains guidance on a series of more advanced capabilities within **Debrief**. New **Debrief** analysts only really need to work through these sections if they've been advised that they will need to use them.

### 4.1 Synchronizing Video, Audio, and Images

When you are analyzing tracks, you might want to analyse a video feed synced with the track plot so as to add more context to the tracks that are being analysed.

Imagine that a helicopter is recording what it sees during an exercise. The video footage will be useful in the analysis of the exercise, but while you can view it (say in Windows Media Player) it is not time-synced to what is happening in **Debrief**. This feature will sync the video with **Debrief**, so as you change the time, in the **Time Controller**, the video will jump to the same time. And inversely, jumping to a point in the video moves the time slider, in the **Time Controller**, to that time. Furthermore, playing either one of them will cause the other to play as well.

On the other hand, you may want to analyze a folder of photographs taken during an exercise. This feature will display the photo nearest to the time in the **Time Controller**.

So, in this section, we will learn the following tasks:

- [Reviewing and Loading Video File](#)
- [Controlling Plot from Media Player](#)
- [Controlling Video from Time Controller](#)
- [Using Image Viewer](#)

#### 4.1.1 Reviewing Files and Loading Video File

Before you synchronize a video feed with the track plot, you need to check whether the required files are available in the respective folders.

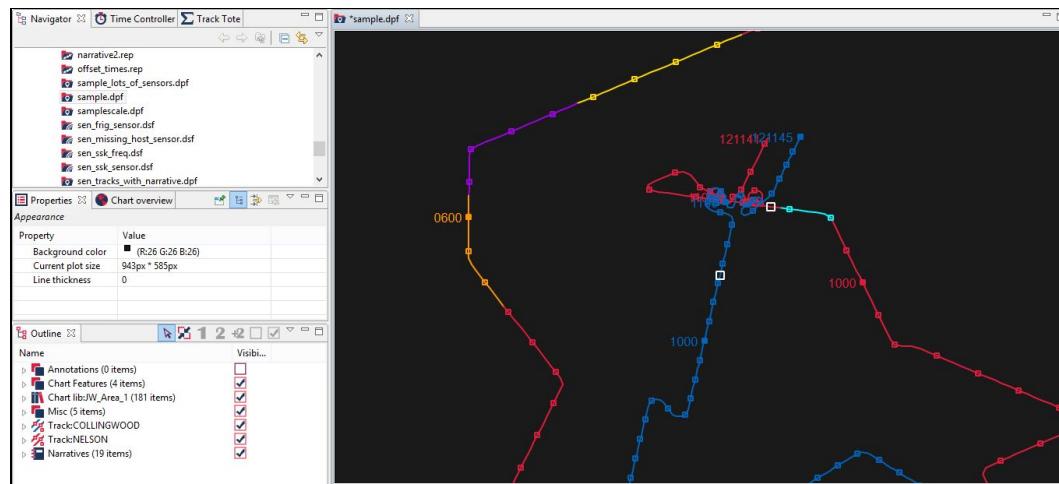
1. Navigate to **Debrief** folder on your system and then **sample\_data > other\_formats** folder.
2. Check the folder contents.
  - This folder should contain **19951212\_102956\_1sec.avi** and **19951212\_103415.avi** files. Note that filenames of both the video files start with a DTG (Date Time Group). This is the timestamp for the start of the recording of the video.

- This folder should contain a sub-folder, **TimedImages** which contains JPEG files. Note that once again the name of each file is a DTG (Date Time Group).
3. Just to verify that the video files are standard files, open both the video files in Windows Media Player.



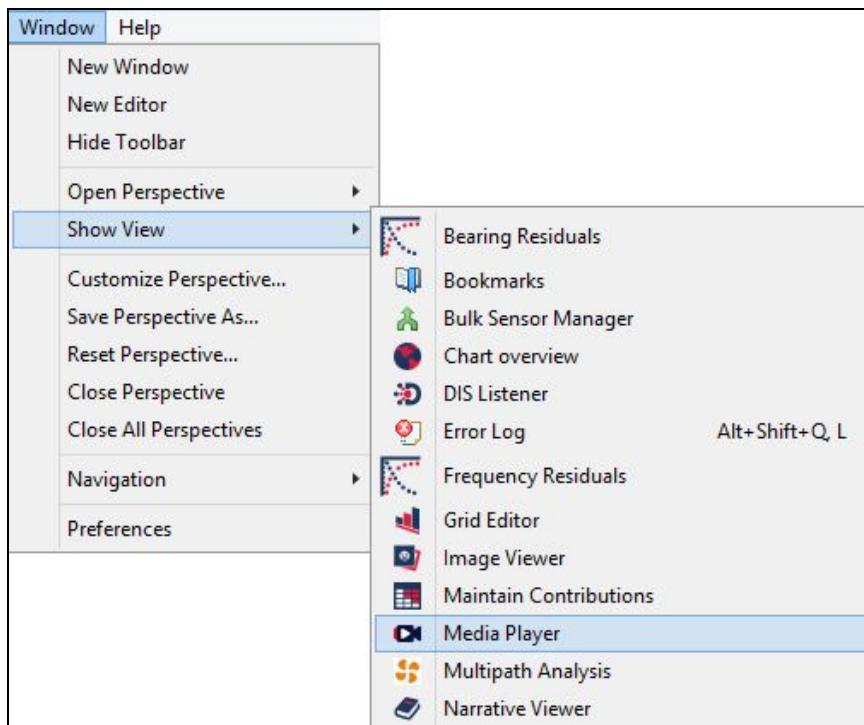
Once you have reviewed the files, it is time to open the required scenario.

4. In the **Navigator** view, drag the file **sample.dpf** and drop it into the plot editor. The **sample.dpf** tracks will display.

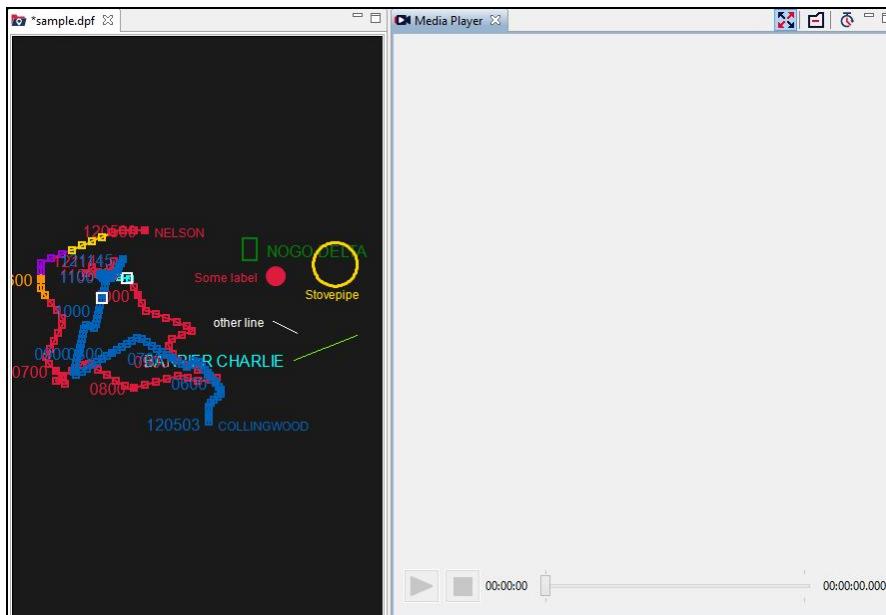


5. Navigate to **Window > Show View > Media Player**.





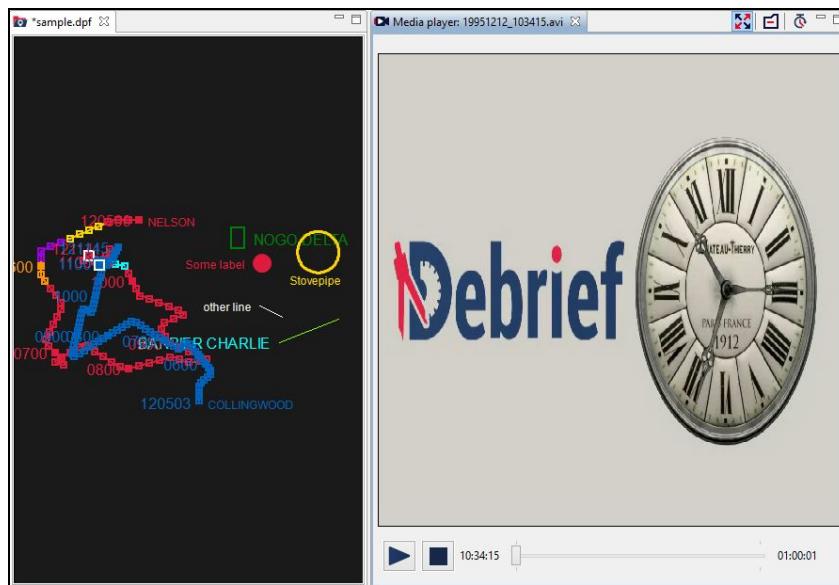
The **Media Player** will display to the right of the **Plot Editor**.



6. From the **Navigator** view, drag the **19951212\_103415.avi** file and drop it into the **Media Player**. The **19951212\_103415.avi** will display. Yes, it's a very dull piece of video, but it

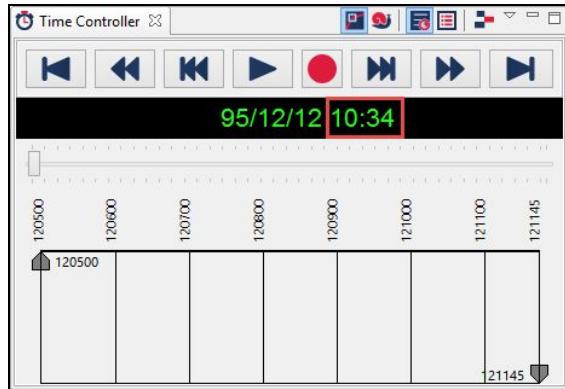


was created to indicate the current video time over a long period, but within a minimal file size.



#### 4.1.2 Controlling Plot from Media Player

In the **Time Controller** view, note that the slider control shows the start time as **10:34:15**. This is the start time of the video, which got extracted from the filename.

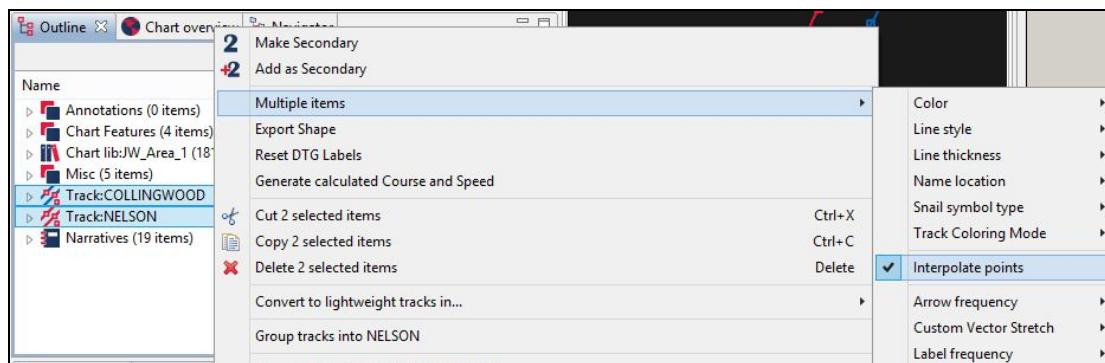


- Now toggle the **Stretch** button  on the toolbar of the **Media Player** and you will notice the video resize to its original aspect ratio.

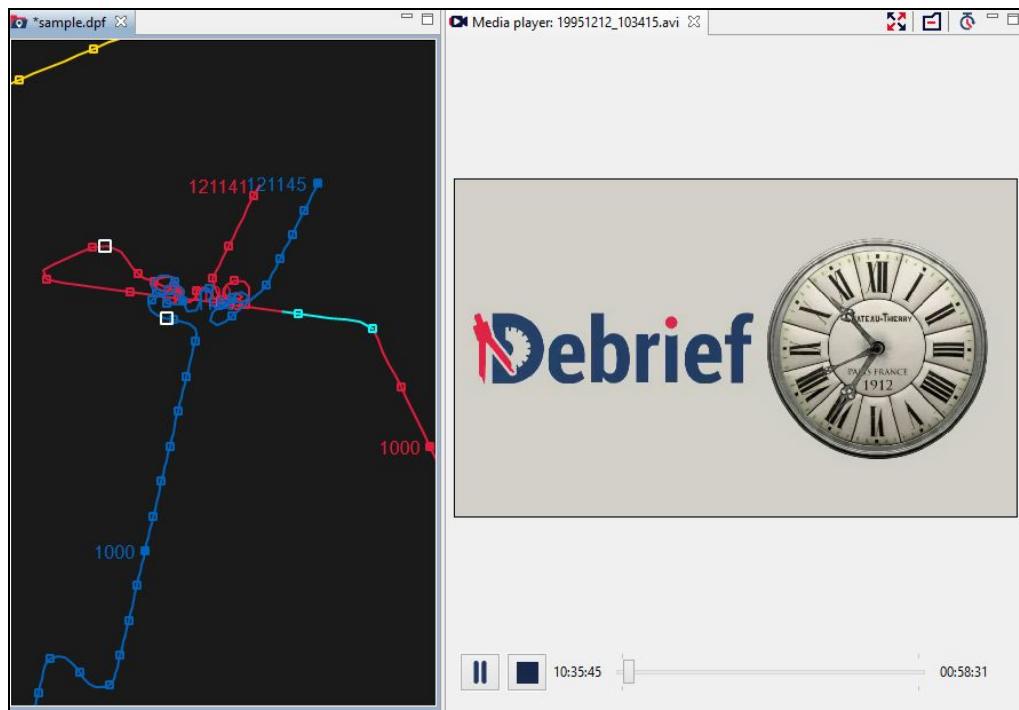




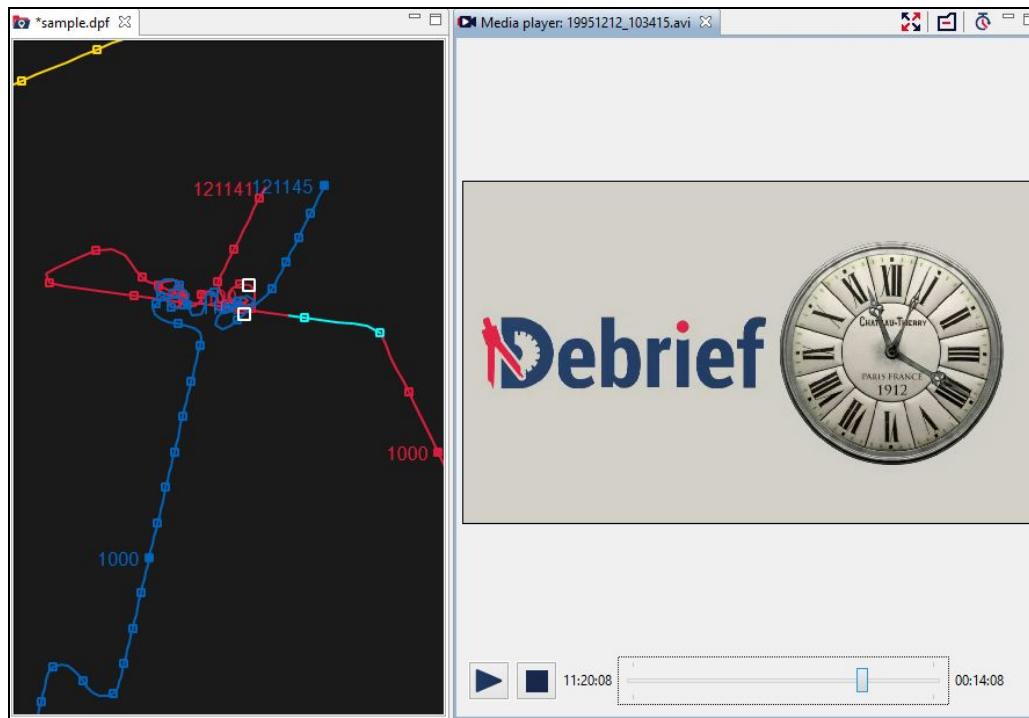
2. Click the **Play** button  on the video controller. You will notice that the video plays, but the track highlight does not play. This is because the plot contains data at one minute intervals, but the video is playing in a frame rate of several steps per second.
3. We can overcome this by instructing **Debrief** to interpolate vessel positions between the one-minute measurements. In the **Outline** view, click on **Track:COLLINGWOOD** and press **Ctrl** and then click on **Track:NELSON**. Now right click and select **Multiple items** and then tick **Interpolate points**.



4. Now, zoom in on the plot and observe the highlight markers moving forward.



5. Experiment with dragging the video slider. Observe the highlight markers update. An example is shown in the below screenshot.



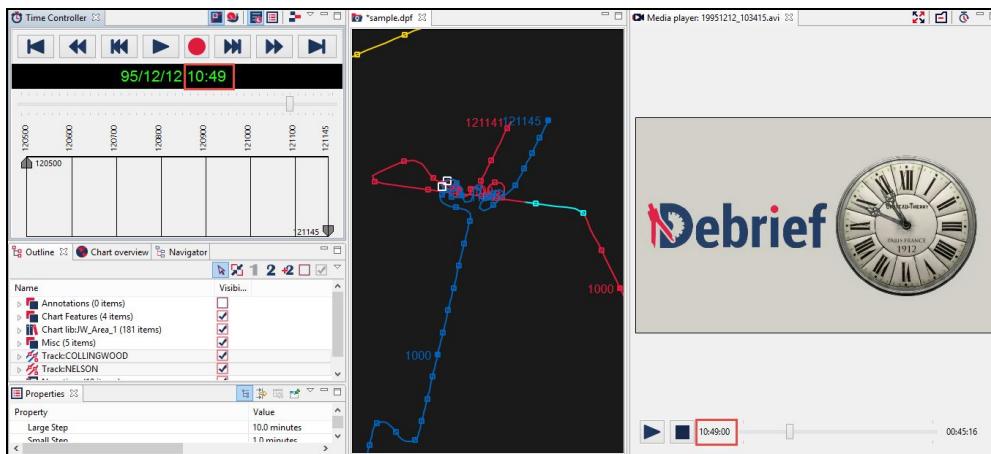
6. Press Stop button  in the video player.



#### 4.1.3 Controlling Video from Time Controller

After learning how to control video from **Media Player**, it is time to learn how to control it from **Time Controller**.

1. Look into the **Time Controller** and experiment with controlling video from it. Drag the slider in the **Time Controller**, and observe the video.



2. Observe that the video clock gets updated to match the time in the **Time Controller** view.



**Note:** The **Edit Start time** button  on the **Media Player** toolbar can be used to change the video start time if the filename doesn't contain a recognisable date-time value, or contains a time in the wrong timezone.

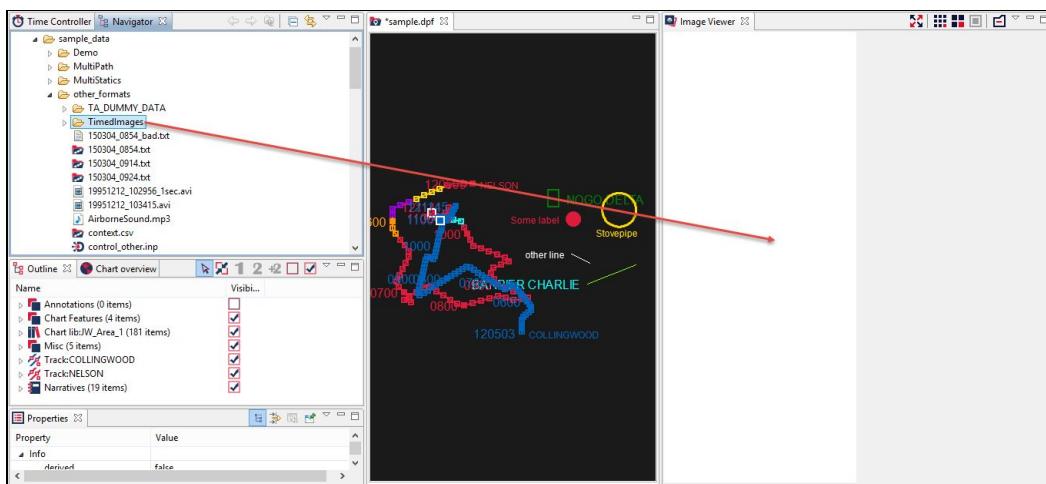
3. Close the **Media Player**.



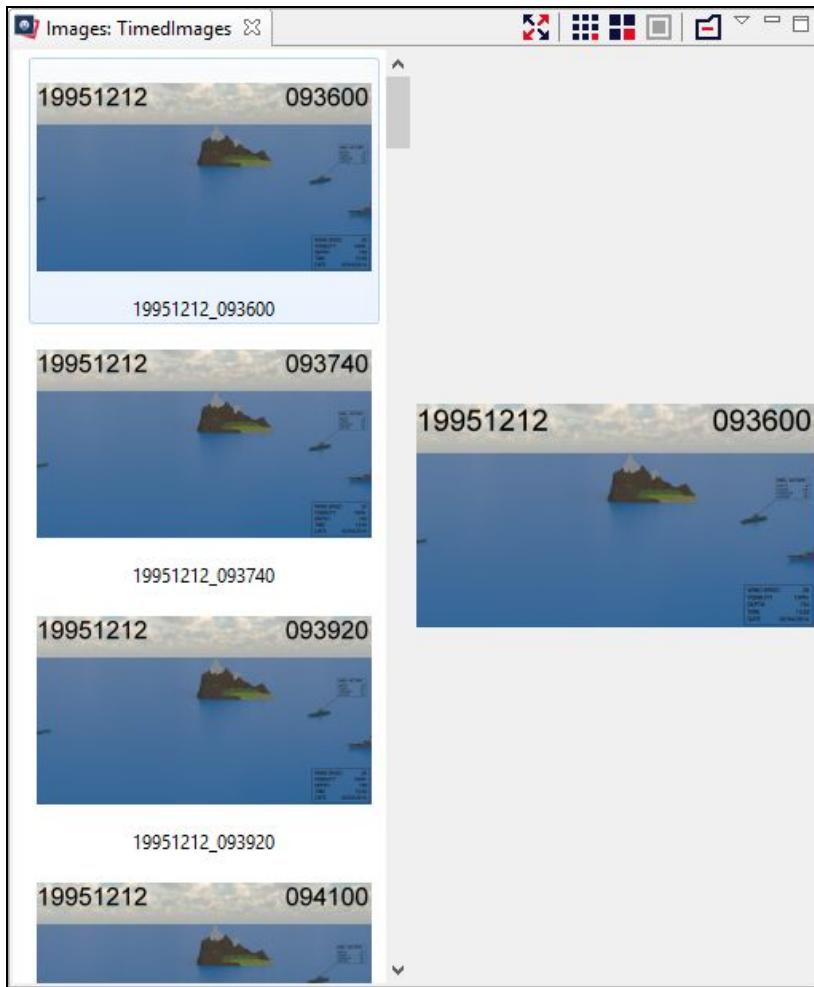
#### 4.1.4 Using Image Viewer

Now let us explore the **Image Viewer**.

1. Navigate to **Window > Show View > Image Viewer**.
  2. In the **Navigator** view, locate the folder **TimedImages** (it is placed under **sample\_data > other\_formats**). Drag and drop the folder **TimedImages** into the **Image Viewer**.



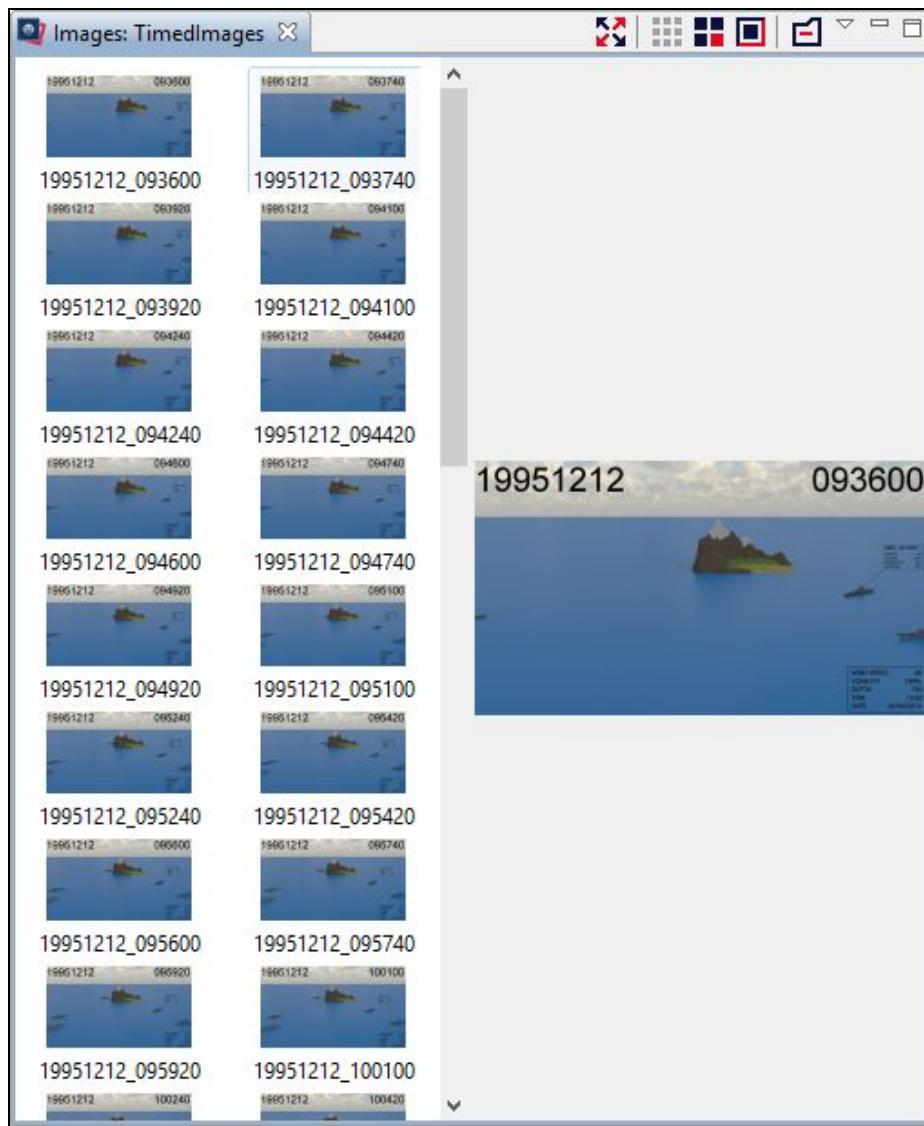
3. The images in the folder **TimedImages** will populate in the **Image Viewer**.



4. There are 3 options available to view the images.

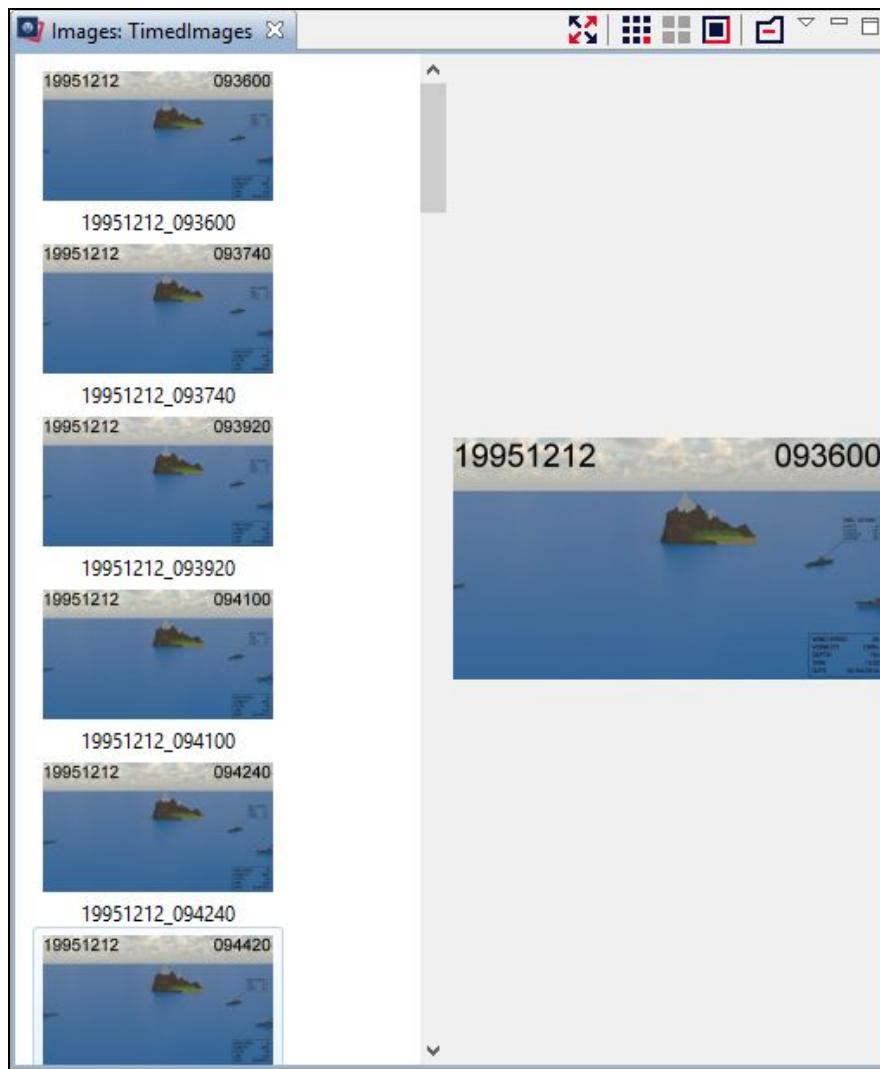
- a. Click on the **Small Icons** button , on the **Image Viewer** toolbar, to view the images of the folder as small icons.





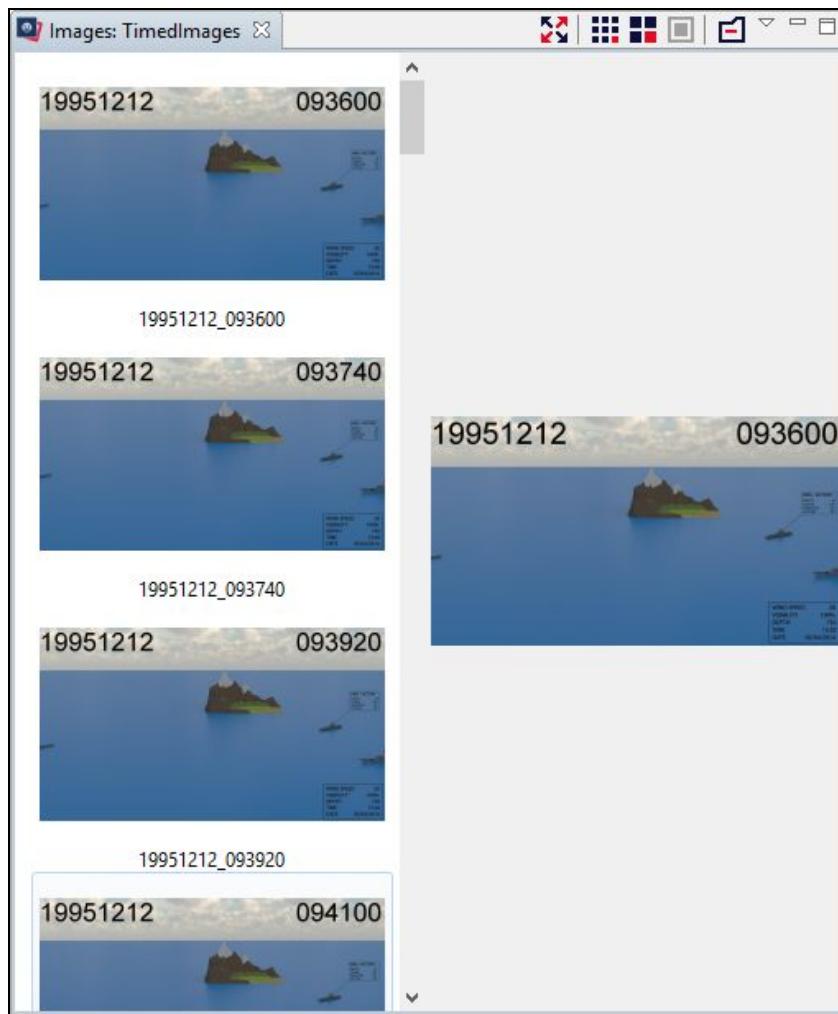
- b. Click on the **Medium Icons** button , on the **Image Viewer** toolbar, to view the images of the folder as medium sized icons.





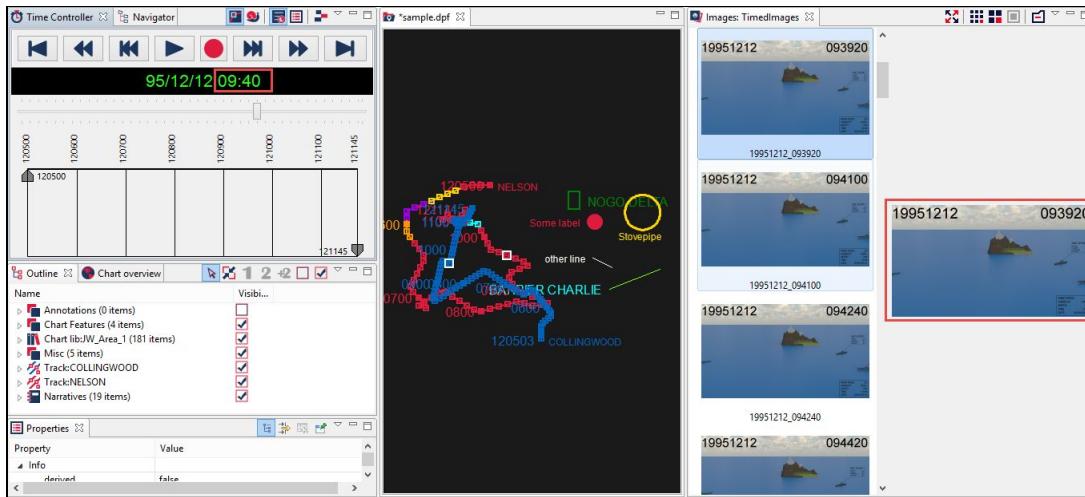
- c. Click on the **Large Icons** button , on the **Image Viewer** toolbar, to view the images of the folder as large sized icons.



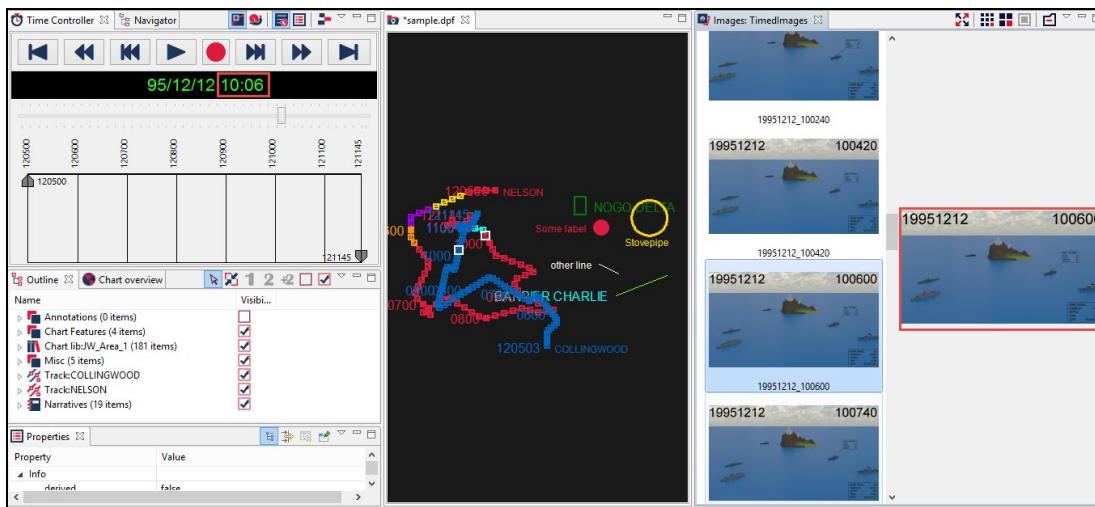


5. Now in the **Time Controller** view, move the slider to **09:40**. Observe that the selected image changes in the thumbnail pane and the image will display in full-size in the preview pane.





6. Scroll down the images and double-click on any one image. Observe that the time in the **Time Controller** updates to reflect this.



## 4.2 Plot in Unit Centric View

When analyzing tracks, you may be interested in the continuous range/bearing of one track to another. The **Unit Centric View** facilitates this exact need.

**Unit Centric View** is a plotting mode that shows the position of non-primary tracks relative to the primary track, at the point in time for the non-primary track measurement.

In this view, the primary track is represented as a single point in the centre of the screen, with other tracks represented as lines on the plot. For example, a non-primary track that maintains a constant

station at, say, 1km distance on a bearing of 135 degrees from the primary track, will be displayed as a single point at that range/distance. In this section you will learn about the **Unit Centric View** in detail, understand the **Snail Paint** mode of the **Unit Centric View**, experiment with the **Local Grid** and **Range Rings** features. For more information, refer to the following sections:

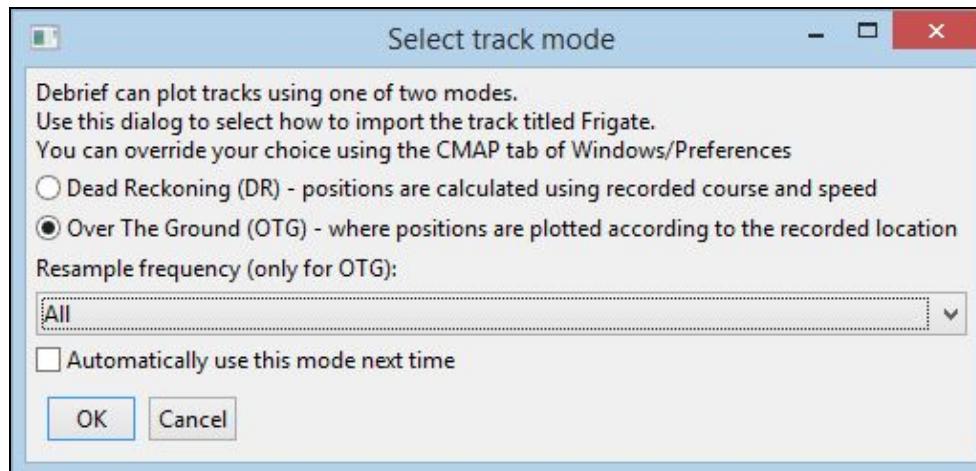
- [Loading and Assigning Tracks](#)
- [Using Snail Paint Mode](#)
- [Experimenting with Local Grid](#)
- [Experimenting with Range Rings](#)

#### 4.2.1 Loading and Assigning Tracks

To open the **Unit Centric View**, we first need to load a sample track and assign a primary track. For this tutorial, we will use the **sen\_tracks.rep** file.

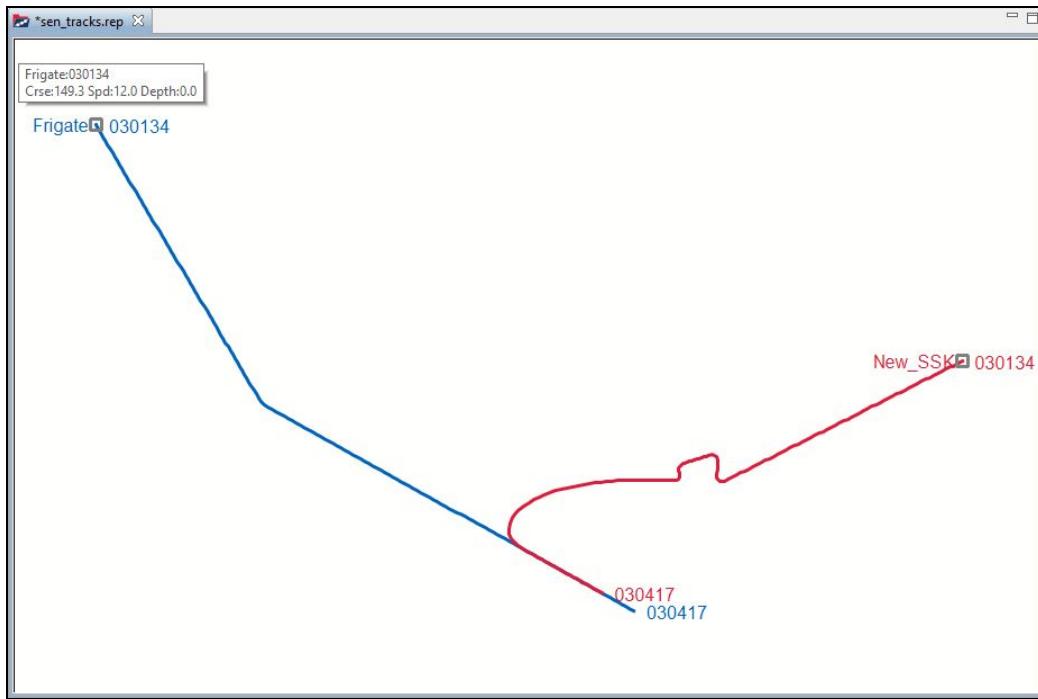
1. In the **Navigator** view, navigate to **sample\_data > SATC\_Test > sen\_tracks.rep**. 

2. Double click on the file **sen\_tracks.rep** to open it. The **Select track mode** dialog will display. 

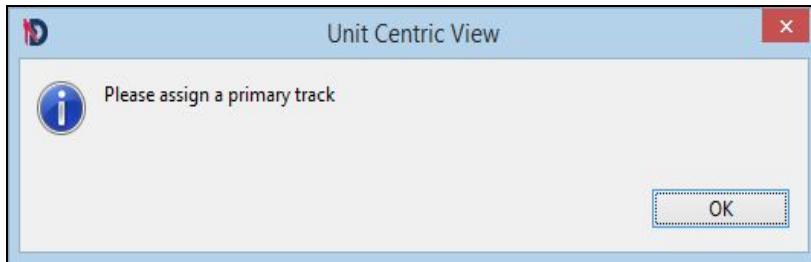


3. Select **Over The Ground (OTG)** option and click **OK**. The **sen\_tracks.rep** file will display 

in the plot editor. Click on the **Fit to window** button  to view the plot clearly. 

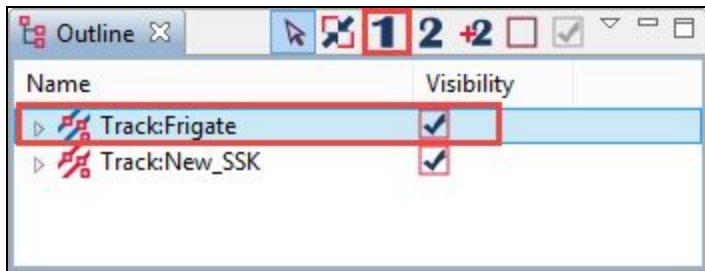


4. To open the **Unit Centric View**, navigate to **Window > Show View > Unit Centric View**.  
A dialog reminding you to assign the primary track will display.

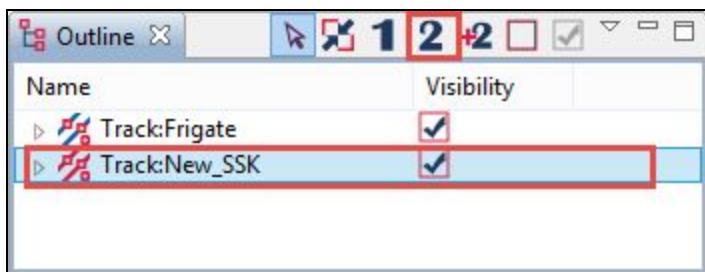


5. Click **OK**. Now we will assign a primary track in the **Outline** view.

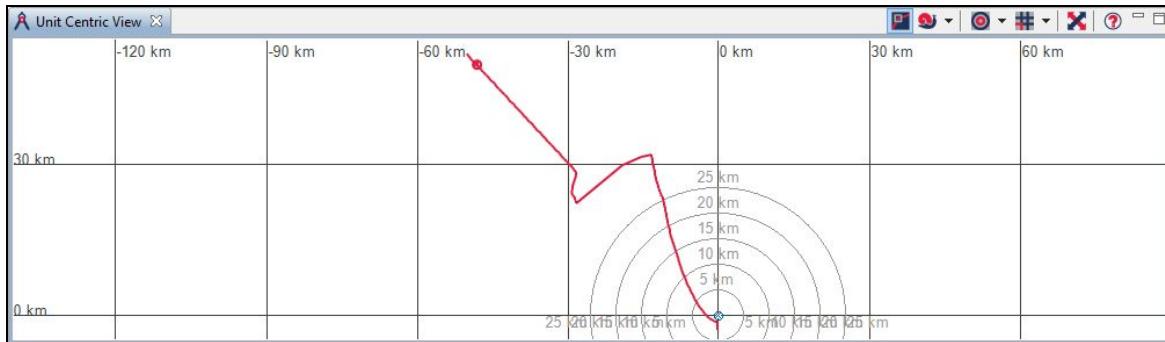
6. In the **Outline** view, click on **Track:Frigate** and then click on  on the toolbar of the **Outline** view.



7. In the **Outline** view, click on **Track>New\_SSK** and then click on **2** on the toolbar of the **Outline** view.



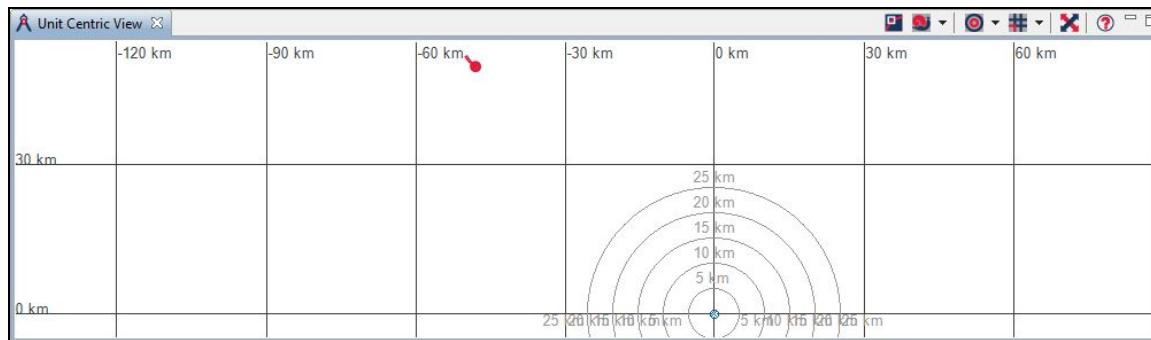
8. The **Unit Centric View** will update to display the primary and secondary tracks. Observe that **Track:Frigate** is the primary track and it is **Track>New\_SSK** for which a line is shown in the **Unit Centric View**.



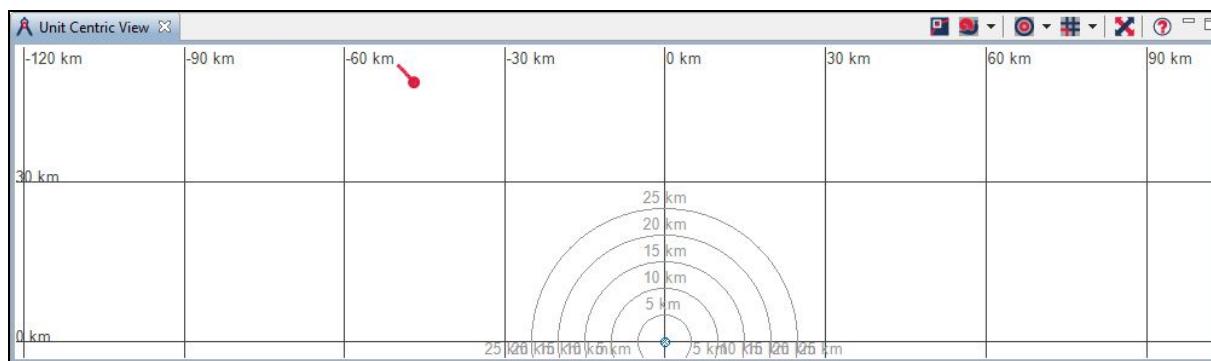
#### 4.2.2 Using Snail Paint Mode

The **Snail Paint** mode is used for analysis tasks when you want to concentrate on specific activities around a specific time, without the clutter of the remaining track data.

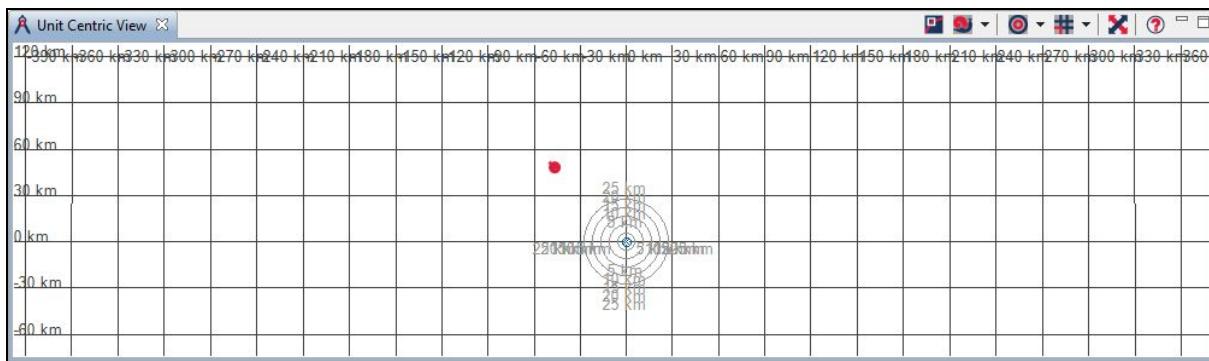
- Let us view the **Unit Centric View** in the **Snail Paint** mode. Click on the **Snail Painter** button  on the **Unit Centric View** toolbar.



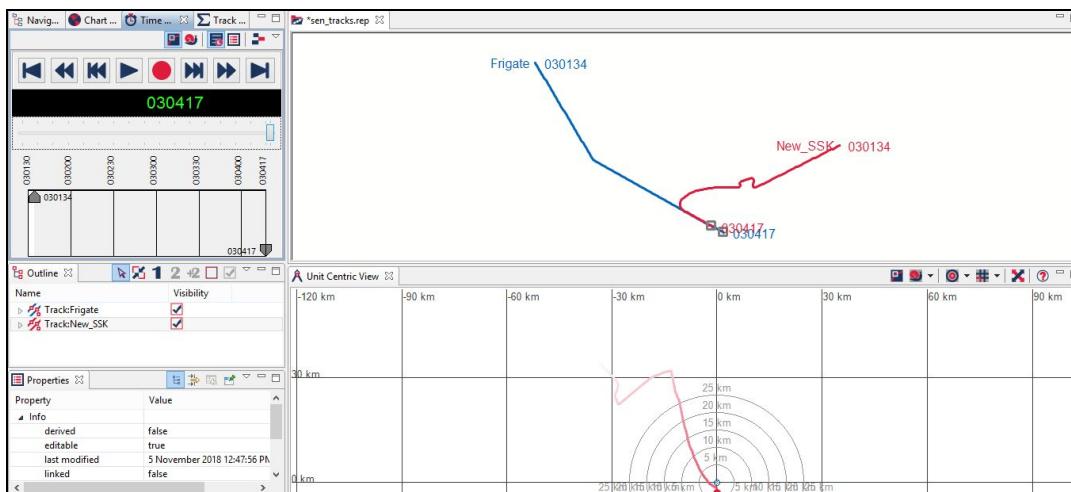
- Now, experiment with the **Time Controller** view by dragging the slider and observe how the line representing the **Track:New\_SS** changes.
- Experiment with zooming in on the **Unit Centric View**. Use the mouse, click and drag on the plot area from top-left to bottom-right and observe the changes.



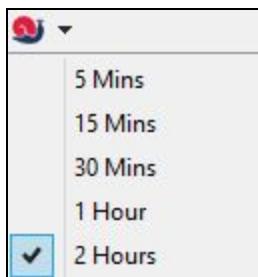
- Experiment with zooming out by clicking and dragging from bottom-right to top-left and observe the changes.



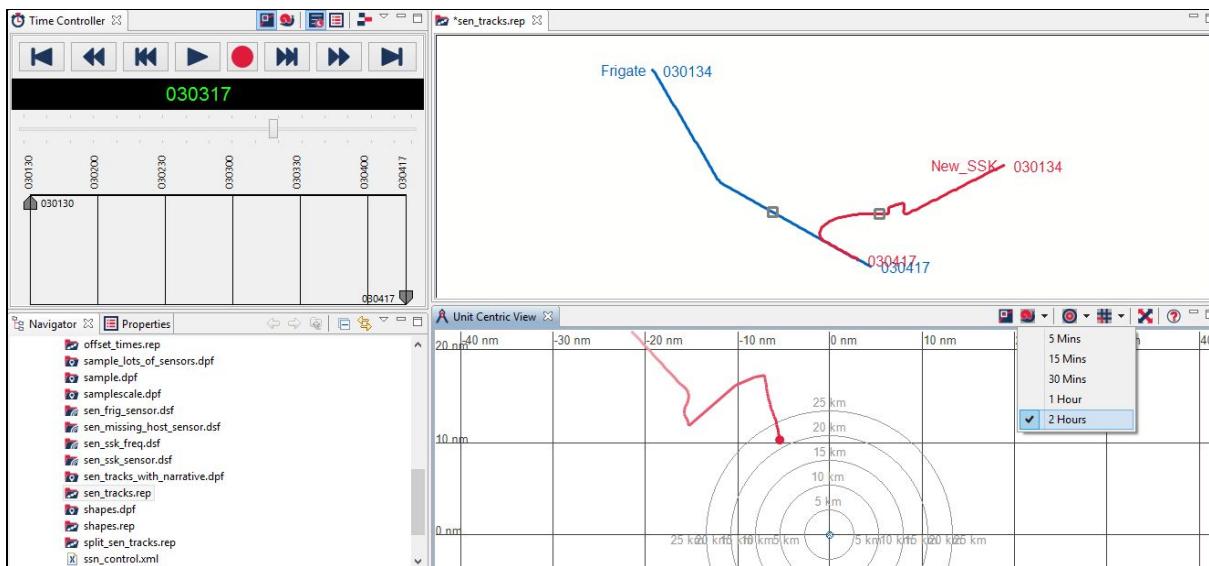
5. Now let us observe relative plotting. Move the slider, in the **Time Controller** view, towards the end of the engagement. You will notice that on the main plot that the red vessel is directly behind the blue one. But, since it is travelling slower, its range increases. This is also shown in the **Unit Centric View**.



6. Click on the **Snail Painter** button  to switch to the **Snail Paint** mode. Now experiment with the snail length options by clicking on the down arrow on the **Snail Painter** button to see the available options.



The snail length is the length of the secondary track, **Track: NEW\_SSK**, which will be displayed when you select a particular option. For example, if you select **2 Hours**, then the length of **Track: NEW\_SSK** will represent two hours from its current position. The current position is where the slider is in the **Time Controller** view. An example is shown in the below screenshot.

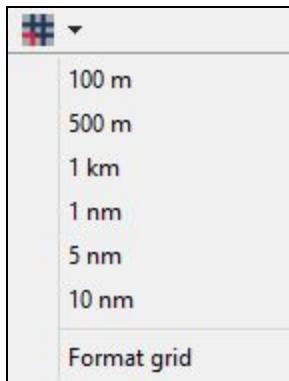


**Note:** Zoom in on the track to notice the changes.

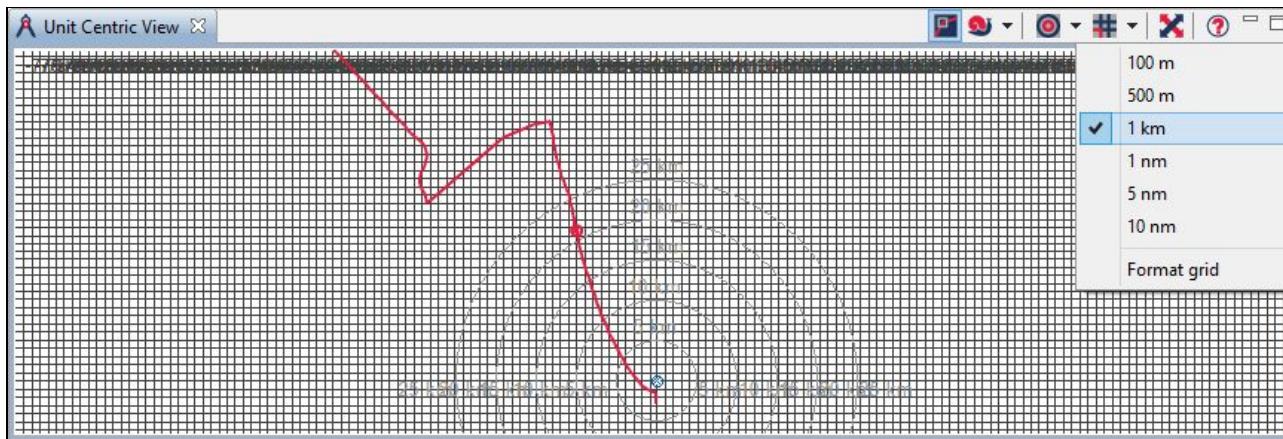
#### 4.2.3 Experimenting with Local Grid

- Now experiment with the size options in the **Show local grid** button . Click on the down arrow on the **Show local grid** button to see the available options.





You can use the size options in the **Local Grid** feature to vary the distance between the grid lines. For example, if you select **1 km** the grid lines will be displayed as shown in the below screenshot.



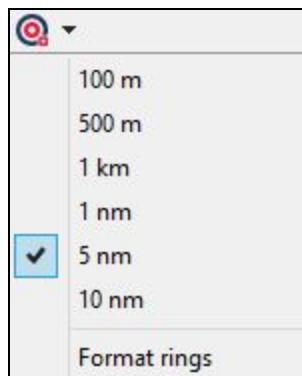
#### 4.2.4 Experimenting with Range Rings

**Range Rings** are graphical representation to depict the distance between two points. In the **Unit Centric View**, range rings are useful as to trace the position of non-primary tracks relative to the primary track. Now we will look at the **Show range rings** and the range options.

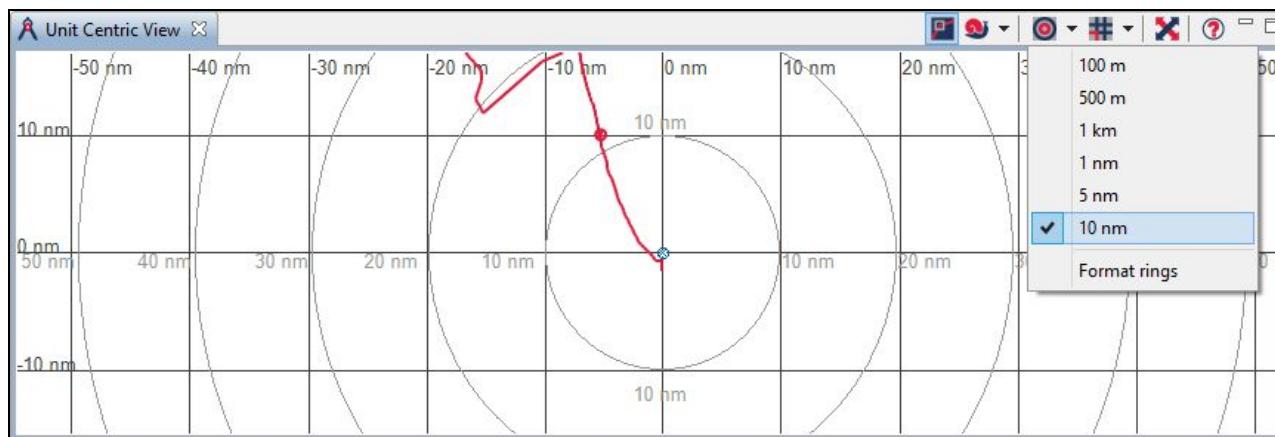
1. Toggle the **Show range rings** button  to switch on / off the range rings. The **Unit Centric view** when the range rings are off is shown in the below screenshot.



2. Now experiment with the range options in the **Show range rings** button . Click on the down arrow on the **Show range rings** button to see the available options.



You can use the range options in the **Show range rings** button to vary the frequency of the range rings that are displayed, in the **Unit Centric View**. For example, if you select **10 nm** the range rings will be displayed as shown in the below screenshot.



## 4.3 Boost Performance Via Lightweight Tracks

As an analyst your main object of interest in **Debrief** are the tracks. Traditional Debrief tracks are heavyweight entities that contain complex internal features, including the ability to compose a track from multiple TMA solutions, storing measured sensor data, and constructing a track using Dead Reckoning. These capabilities of **Debrief** tracks cause the performance to slow down. To overcome this issue you can use **Lightweight Tracks** as an alternative.

**Lightweight Tracks** are simpler, faster version of a conventional track; they have higher performance and can be plotted easily in bulk. You can organize **Lightweight Tracks** into layers which allows them to be switched on or off together. Lightweight Tracks can be converted to a normal track enabling deeper analysis when necessary, and conversely any normal track can be converted into a lightweight track for faster performance.

In this section, you will learn about lightweight tracks, converting between normal track and lightweight tracks, organizing lightweight tracks into a new layer. For more information, refer to the following sections:

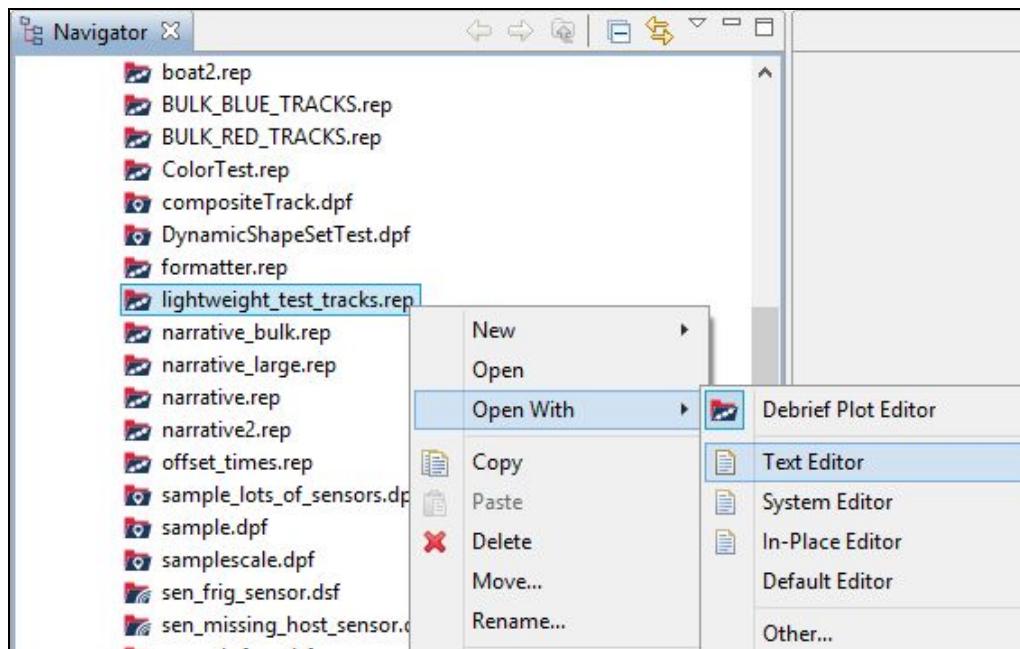
- [Inspecting Lightweight Track Data](#)
- [Opening Lightweight Track](#)
- [Converting Lightweight Track to Normal Track](#)
- [Assigning Lightweight Tracks as Primary Track and Secondary Track](#)
- [Creating Range Plot of Multiple Lightweight Tracks against Primary Track](#)
- [Converting Normal Track to Lightweight Track in an Existing Layer](#)
- [Converting Normal Track to Lightweight Track in a New Layer](#)

### 4.3.1 Inspecting Lightweight Track Data

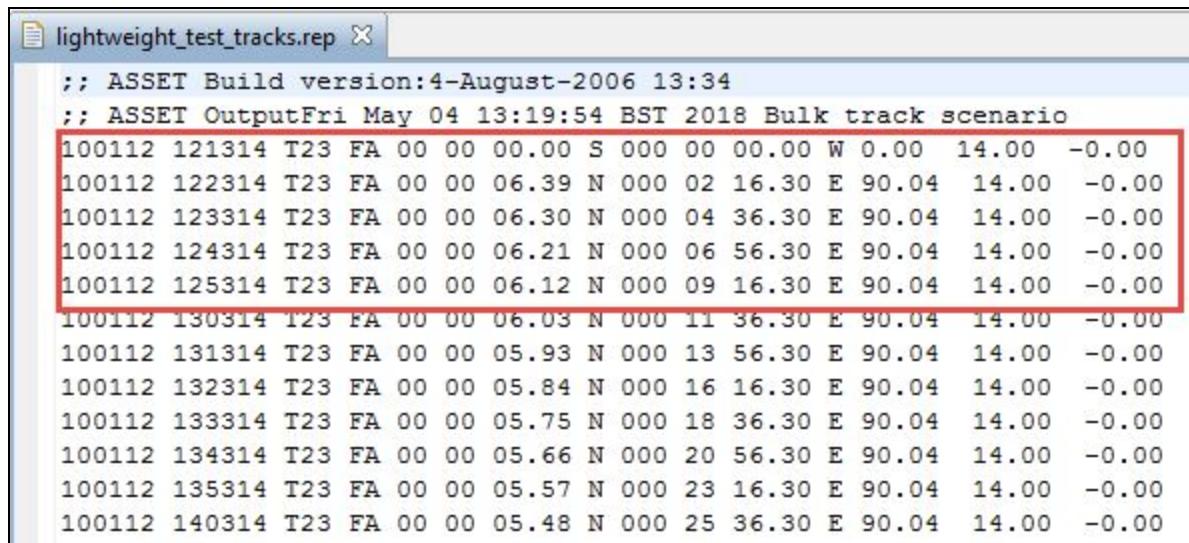
Let us first see how lightweight tracks are represented in **.REP** files, using Debrief's built-n **Text Editor**. For this we have to open the sample file **lightweight\_test\_tracks.rep**.

#### Loading Sample File

1. In the **Navigator** view, open the **sample\_data** folder and then locate the file **lightweight\_test\_tracks.rep** 
2. Right click on the file **lightweight\_test\_tracks.rep** and then select **Open with** and then click **Text Editor**. 



3. The file **lightweight\_test\_tracks.rep** will display in the **Text Editor**.



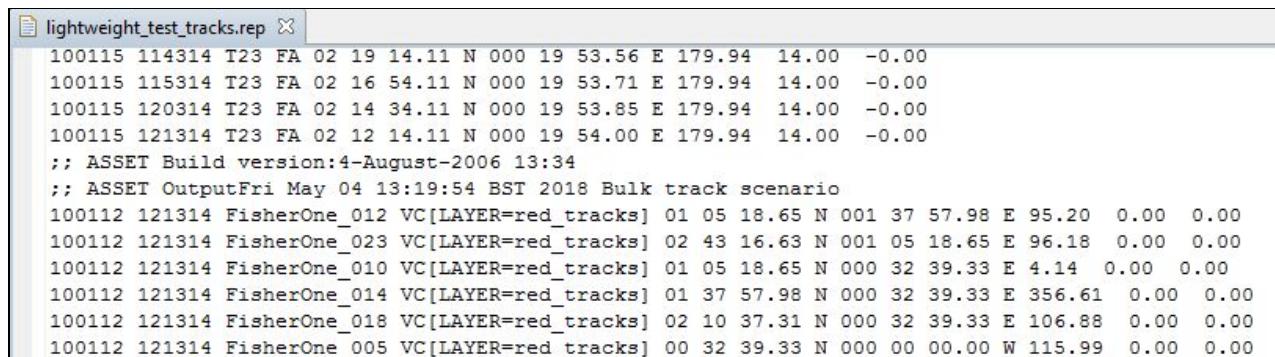
```

;; ASSET Build version:4-August-2006 13:34
;; ASSET OutputFri May 04 13:19:54 BST 2018 Bulk track scenario
100112 121314 T23 FA 00 00 00.00 S 000 00 00.00 W 0.00 14.00 -0.00
100112 122314 T23 FA 00 00 06.39 N 000 02 16.30 E 90.04 14.00 -0.00
100112 123314 T23 FA 00 00 06.30 N 000 04 36.30 E 90.04 14.00 -0.00
100112 124314 T23 FA 00 00 06.21 N 000 06 56.30 E 90.04 14.00 -0.00
100112 125314 T23 FA 00 00 06.12 N 000 09 16.30 E 90.04 14.00 -0.00
100112 130314 T23 FA 00 00 06.03 N 000 11 36.30 E 90.04 14.00 -0.00
100112 131314 T23 FA 00 00 05.93 N 000 13 56.30 E 90.04 14.00 -0.00
100112 132314 T23 FA 00 00 05.84 N 000 16 16.30 E 90.04 14.00 -0.00
100112 133314 T23 FA 00 00 05.75 N 000 18 36.30 E 90.04 14.00 -0.00
100112 134314 T23 FA 00 00 05.66 N 000 20 56.30 E 90.04 14.00 -0.00
100112 135314 T23 FA 00 00 05.57 N 000 23 16.30 E 90.04 14.00 -0.00
100112 140314 T23 FA 00 00 05.48 N 000 25 36.30 E 90.04 14.00 -0.00

```

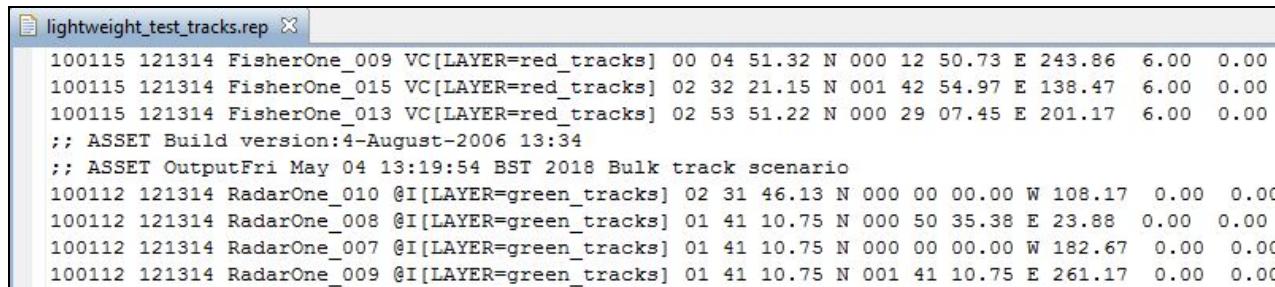
4. In the **Text Editor**, note that **T23** is a normal track, expressed in traditional REP format..

Now scroll down and you will notice that multiple further tracks indicate that they should be stored in either in the **red\_tracks** layer or in the **green\_tracks** layer. By specifying the target layer, **Debrief** will load them as lightweight tracks.



```
lightweight_test_tracks.rep
100115 114314 T23 FA 02 19 14.11 N 000 19 53.56 E 179.94 14.00 -0.00
100115 115314 T23 FA 02 16 54.11 N 000 19 53.71 E 179.94 14.00 -0.00
100115 120314 T23 FA 02 14 34.11 N 000 19 53.85 E 179.94 14.00 -0.00
100115 121314 T23 FA 02 12 14.11 N 000 19 54.00 E 179.94 14.00 -0.00
;; ASSET Build version:4-August-2006 13:34
;; ASSET OutputFri May 04 13:19:54 BST 2018 Bulk track scenario
100112 121314 FisherOne_012 VC[LAYER=red_tracks] 01 05 18.65 N 001 37 57.98 E 95.20 0.00 0.00
100112 121314 FisherOne_023 VC[LAYER=red_tracks] 02 43 16.63 N 001 05 18.65 E 96.18 0.00 0.00
100112 121314 FisherOne_010 VC[LAYER=red_tracks] 01 05 18.65 N 000 32 39.33 E 4.14 0.00 0.00
100112 121314 FisherOne_014 VC[LAYER=red_tracks] 01 37 57.98 N 000 32 39.33 E 356.61 0.00 0.00
100112 121314 FisherOne_018 VC[LAYER=red_tracks] 02 10 37.31 N 000 32 39.33 E 106.88 0.00 0.00
100112 121314 FisherOne_005 VC[LAYER=red_tracks] 00 32 39.33 N 000 00 00.00 W 115.99 0.00 0.00
```

5. Scroll down further to view the tracks in the **green\_tracks** layer.



```
lightweight_test_tracks.rep
100115 121314 FisherOne_009 VC[LAYER=red_tracks] 00 04 51.32 N 000 12 50.73 E 243.86 6.00 0.00
100115 121314 FisherOne_015 VC[LAYER=red_tracks] 02 32 21.15 N 001 42 54.97 E 138.47 6.00 0.00
100115 121314 FisherOne_013 VC[LAYER=red_tracks] 02 53 51.22 N 000 29 07.45 E 201.17 6.00 0.00
;; ASSET Build version:4-August-2006 13:34
;; ASSET OutputFri May 04 13:19:54 BST 2018 Bulk track scenario
100112 121314 RadarOne_010 @I[LAYER=green_tracks] 02 31 46.13 N 000 00 00.00 W 108.17 0.00 0.00
100112 121314 RadarOne_008 @I[LAYER=green_tracks] 01 41 10.75 N 000 50 35.38 E 23.88 0.00 0.00
100112 121314 RadarOne_007 @I[LAYER=green_tracks] 01 41 10.75 N 000 00 00.00 W 182.67 0.00 0.00
100112 121314 RadarOne_009 @I[LAYER=green_tracks] 01 41 10.75 N 001 41 10.75 E 261.17 0.00 0.00
```

6. Close the **lightweight\_test\_tracks.rep** file (that you have opened in the **Text Editor**)

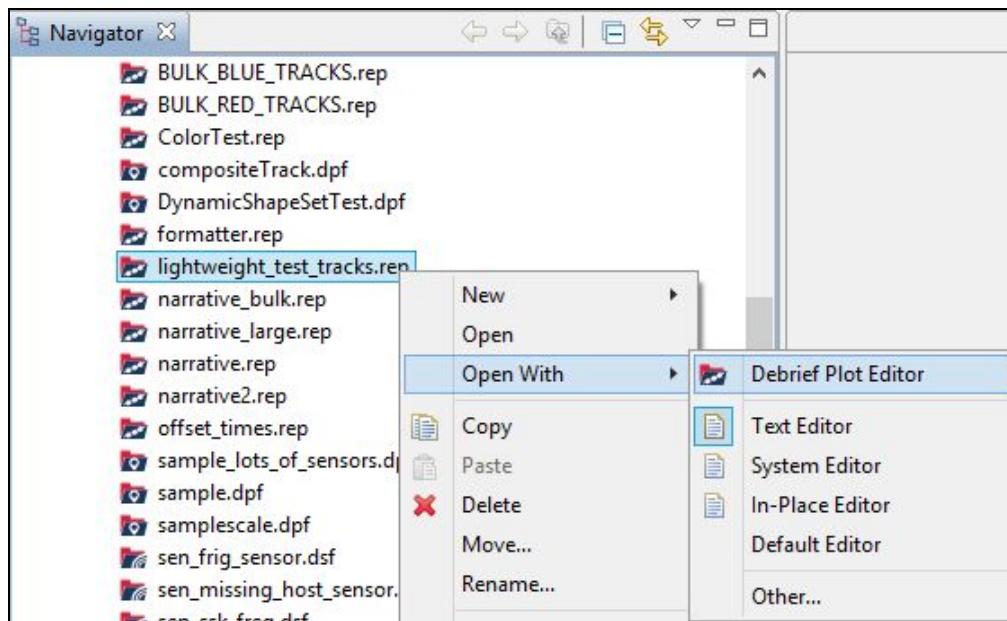


#### 4.3.2 Opening Lightweight Track

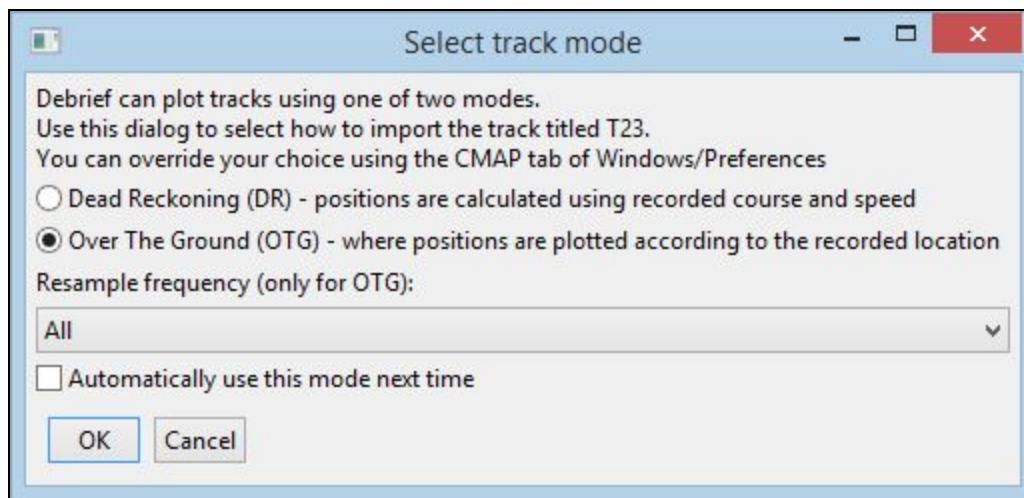
Now that you are familiar with the normal track and the lightweight tracks in the **lightweight\_test\_tracks.rep**, let us move on to experimenting with the lightweight tracks.

1. In the **Navigator** view, right click on the file **lightweight\_test\_tracks.rep** and click **Open with** and then click **Debrief Plot Editor**.





The **Select track mode** dialog will display.



Select OTG mode, and the **lightweight\_test\_tracks.rep** file will open in the **Plot Editor**.



2. In the **Outline** view, you can see the **red\_tracks** (25 items) and **green\_tracks** (10 items).



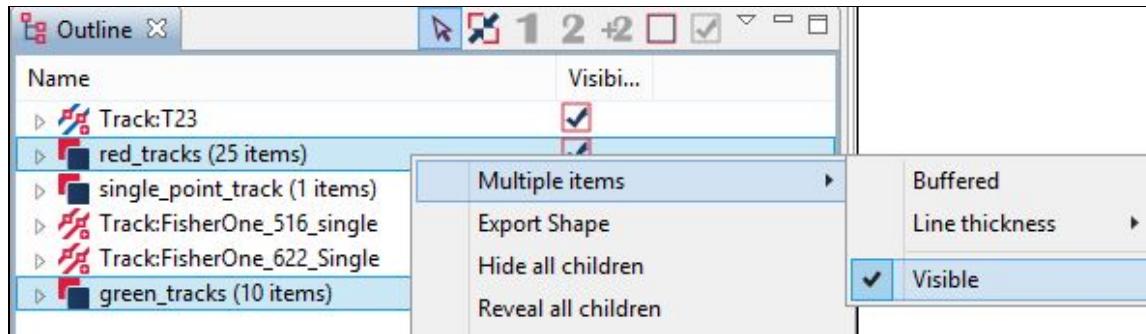
Name	Visibi...
Track:T23	<input checked="" type="checkbox"/>
red_tracks (25 items)	<input checked="" type="checkbox"/>
single_point_track (1 items)	<input checked="" type="checkbox"/>
Track:FisherOne_516_single	<input checked="" type="checkbox"/>
Track:FisherOne_622_Single	<input checked="" type="checkbox"/>
green_tracks (10 items)	<input checked="" type="checkbox"/>

3. Let's experiment with switching on / off the visibility of **red\_tracks** and **green\_tracks**.

Click on **red\_tracks (25 items)** and press **Ctrl** and then click on **green\_tracks (10 items)**.



4. Right click on the selected items and click **Multiple items** and then untick **Visible**.

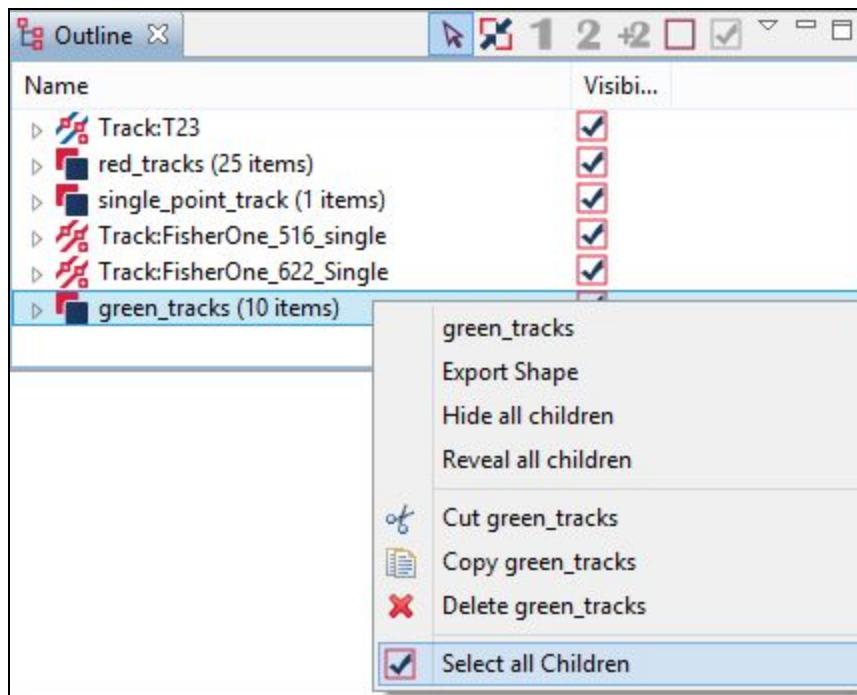


5. You will notice that the **red\_tracks** and the **green\_tracks** will not be displayed in the plot. Now make the **red\_tracks** and the **green\_tracks** visible again.

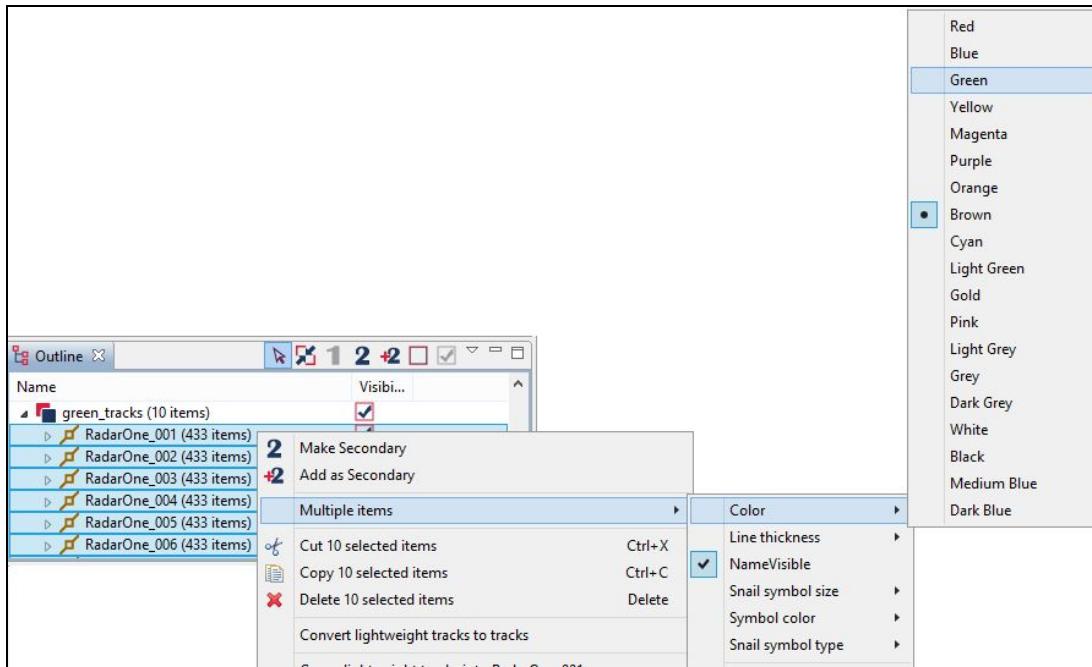


You can also change the attributes of the tracks, like color, line thickness, line style and so on. Now let us change the color of the **green\_tracks**.

6. Right click on **green\_tracks** and then click **Select all Children**.

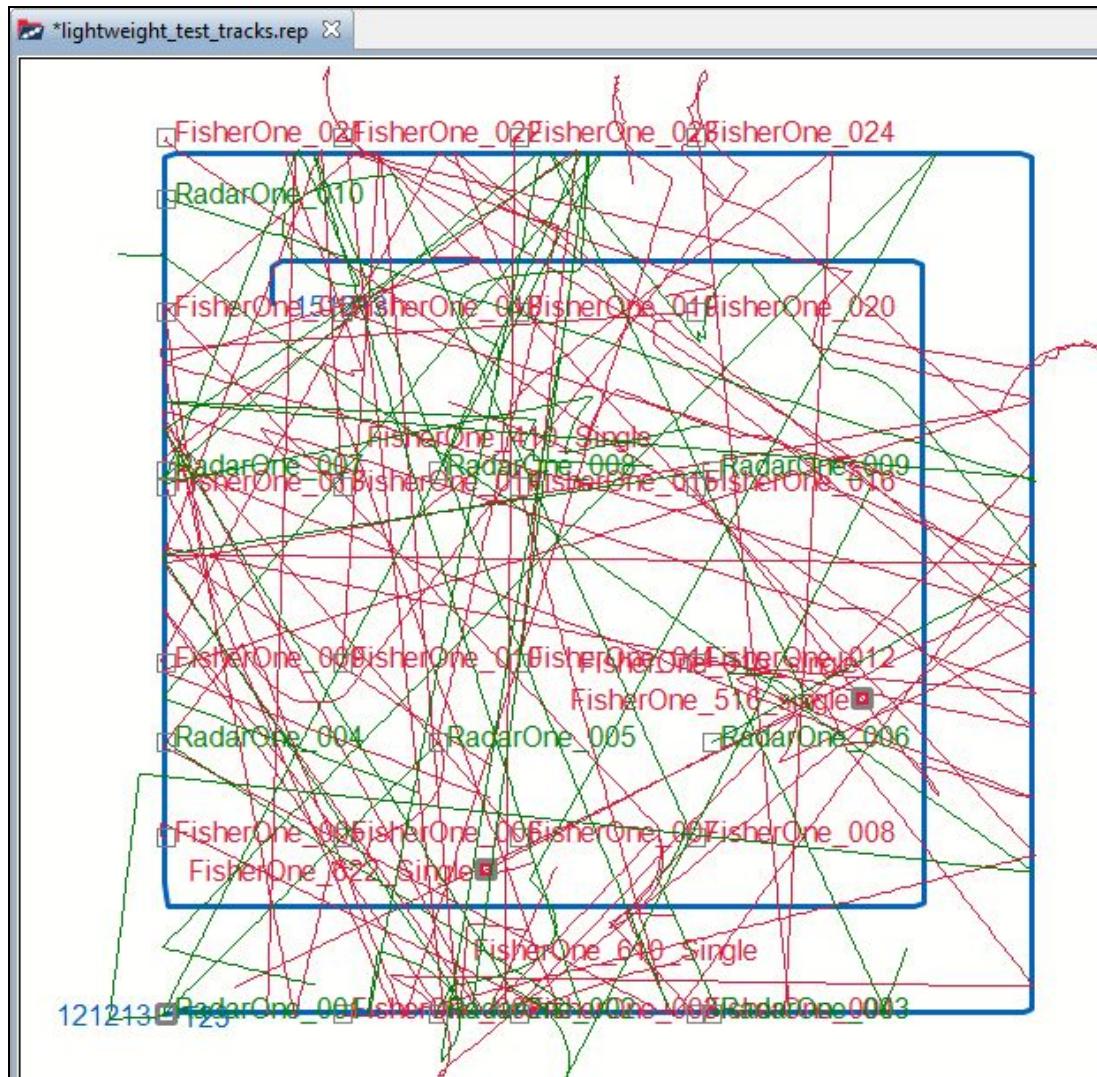


7. Right click on the selected children and the click **Multiple Items > Color > Green**.

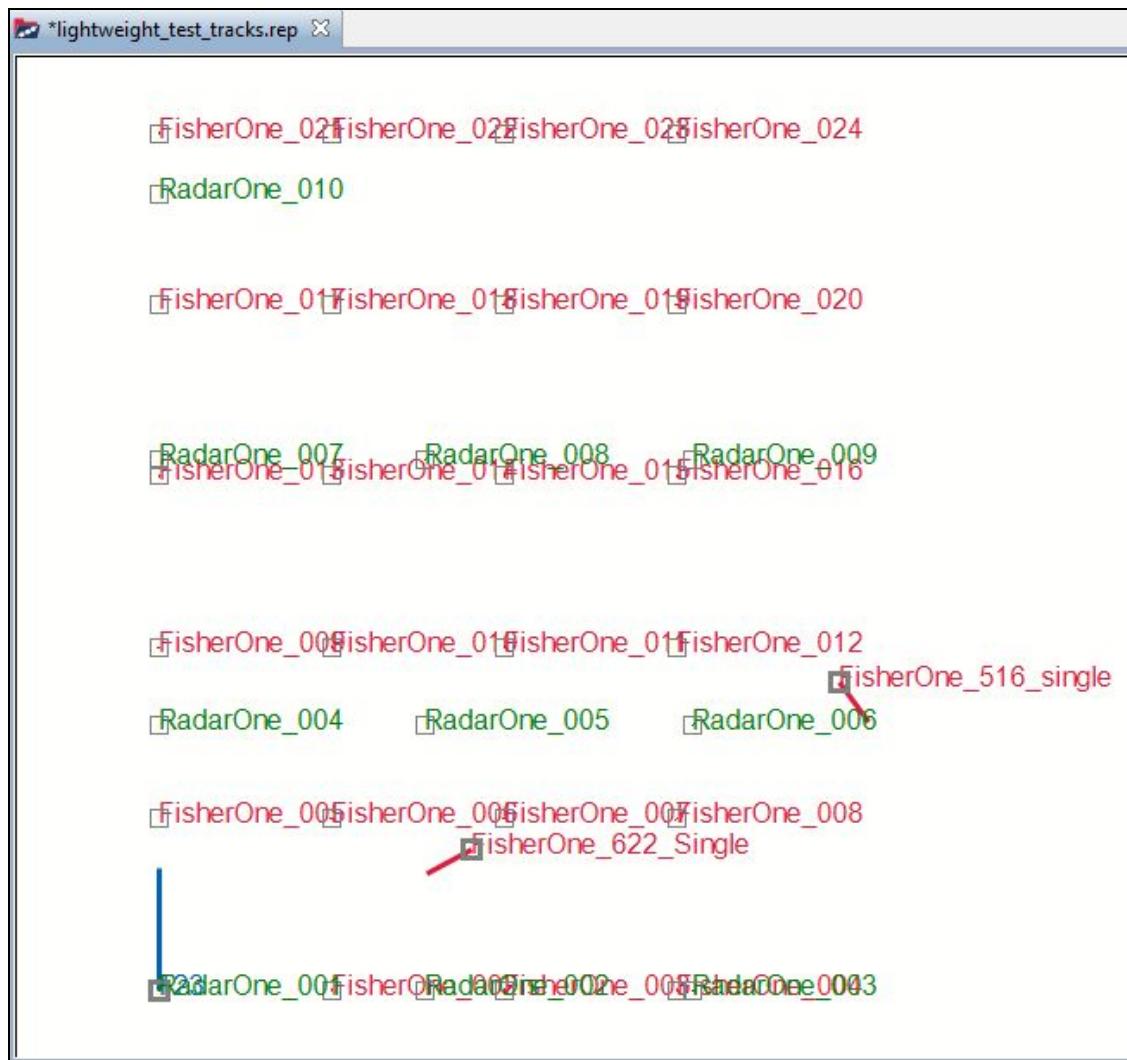


The color of the selected child tracks will change to green in the **Plot Editor**.





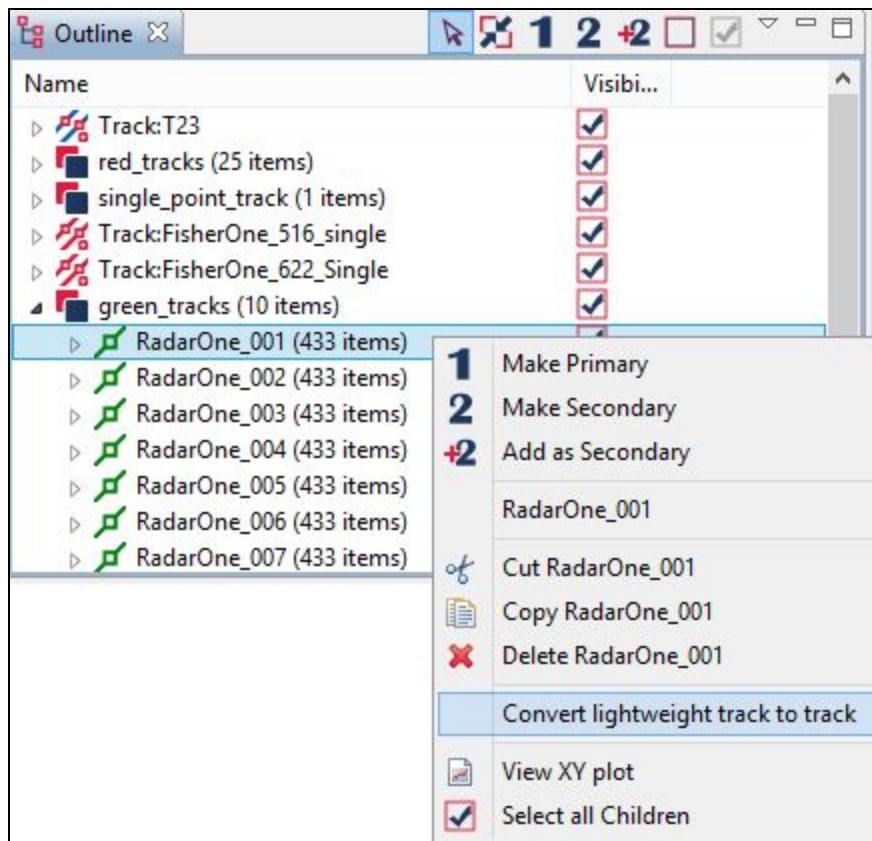
8. Now change the time mode to **Snail**. In the **Time Controller** view, click on the **Snail** button to view the changes.



#### 4.3.3 Converting Lightweight Track to Normal Track

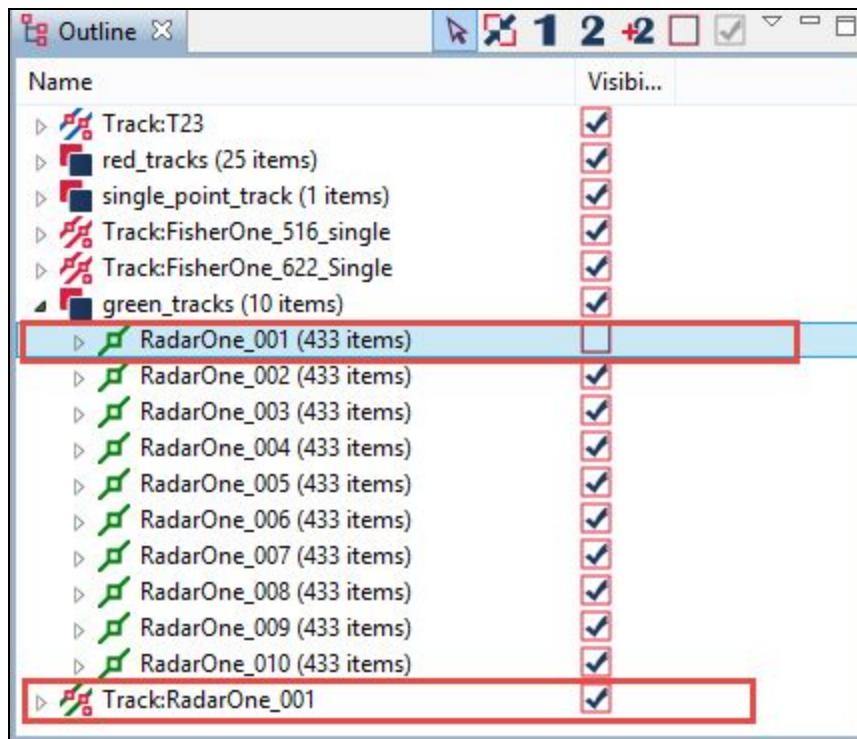
Now let us convert a lightweight track to a conventional track and observe the changes in the track and in the **Outline** view. When you convert a lightweight track to a conventional one, the original track will be hidden in the **Plot Editor** and a new top level track will be created in the **Outline** view.

1. In the **Outline** view, right click on **RadarOne\_001 (433 items)** and select **Convert lightweight track to track**.



2. The original track **RadarOne\_001 (433 items)** will be hidden and a new normal track **Track:RadarOne\_001** will be created and visible in the **Outline** view.



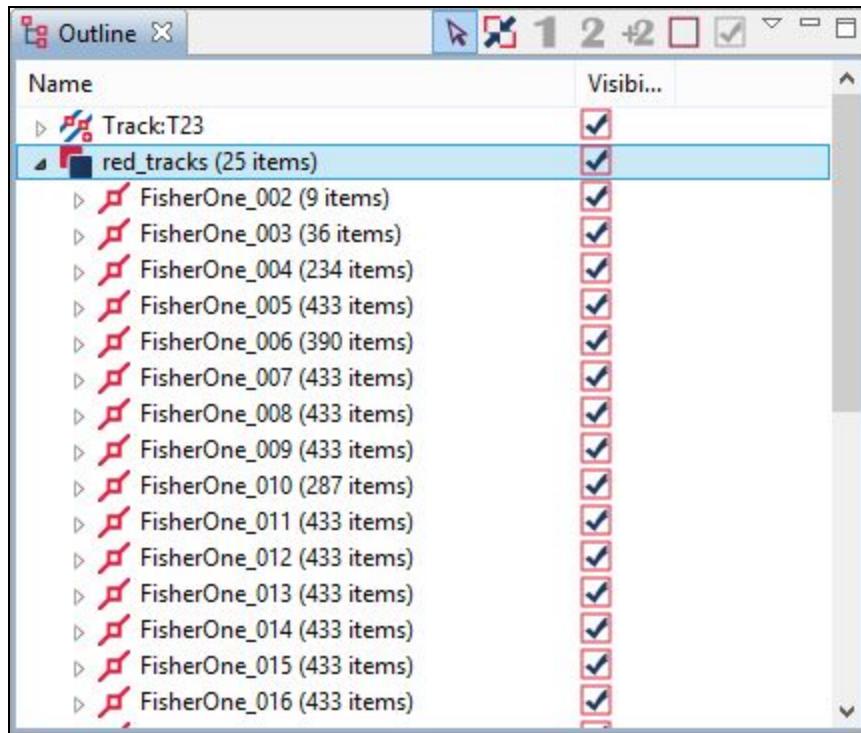


#### 4.3.4 Assigning Lightweight Tracks as Primary Track and Secondary Track

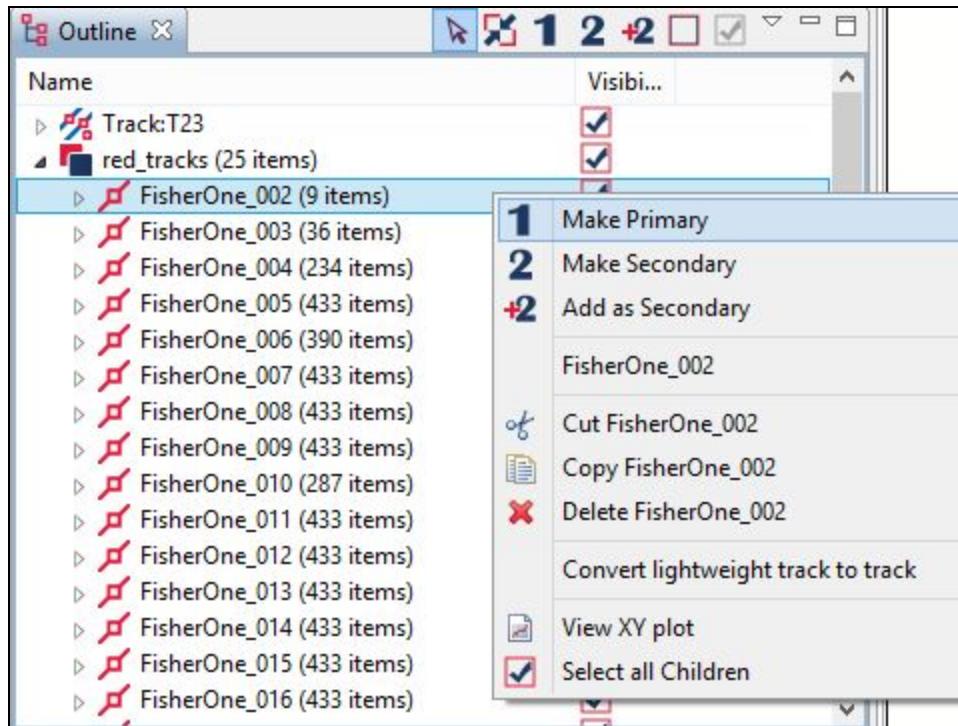
Now let us assign lightweight tracks as primary and secondary tracks.

1. Click on the arrow next to **red\_tracks (25 items)** to expand it.





2. Right click on **FisherOne\_002 (9 items)** and select **Make Primary**. The lightweight track **FisherOne\_002 (9 items)** will be assigned as the primary track.



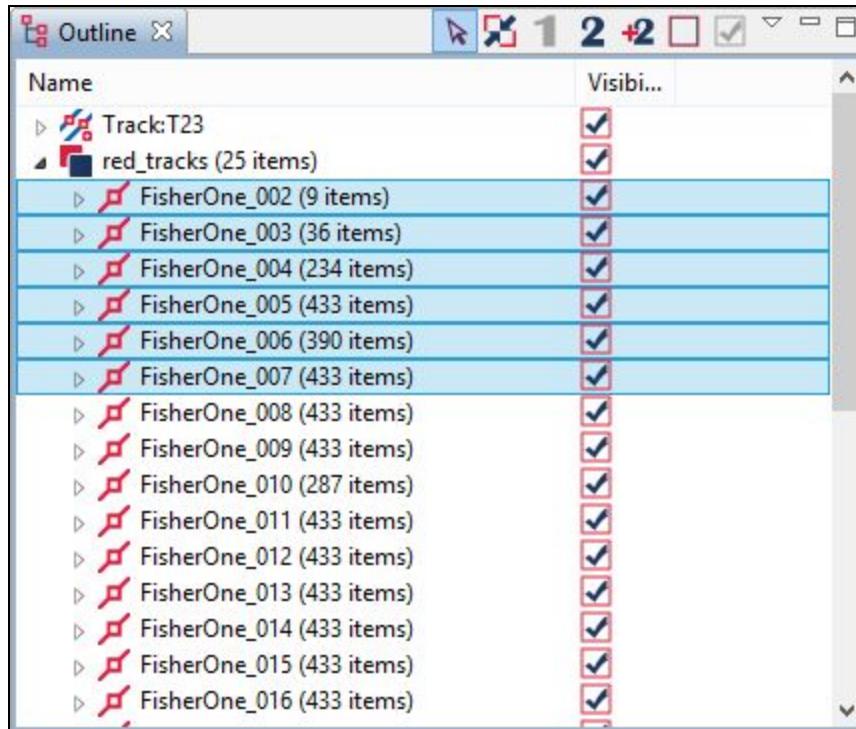
3. Then right click on **FisherOne\_003 (36 items)** and select **Make Secondary**. The lightweight track **FisherOne\_003 (36 items)** will be assigned as the secondary track.



#### 4.3.5 Creating Range Plot of Multiple Lightweight Tracks against Primary Track

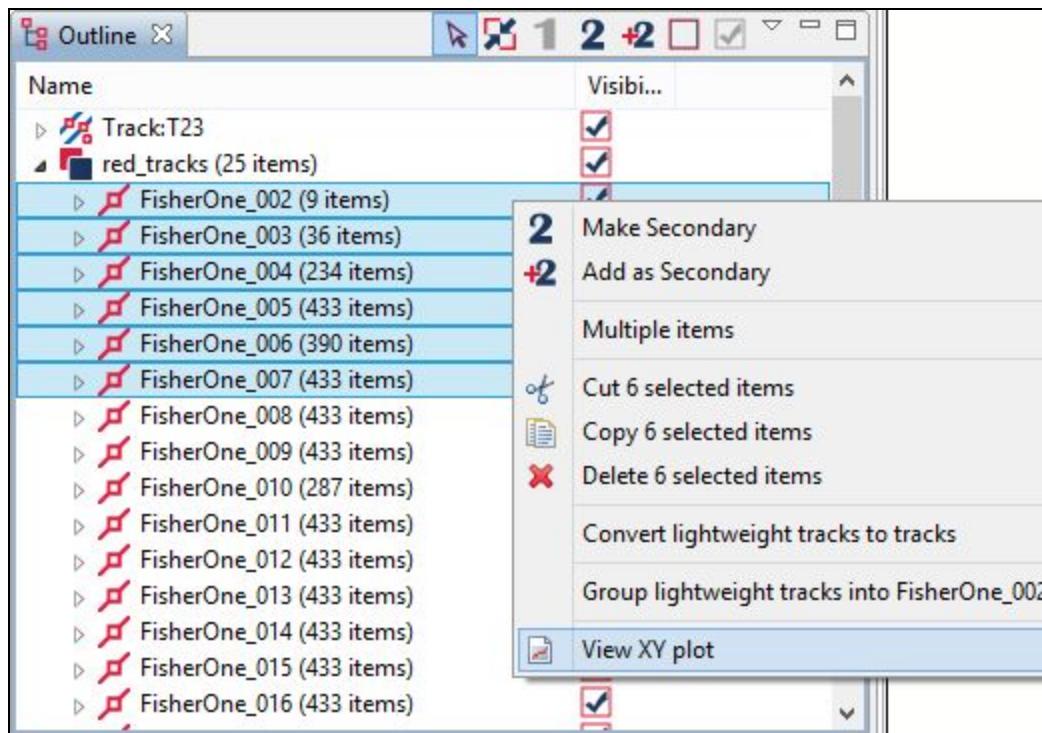
Let us now create a range plot of multiple lightweight tracks against the primary track **FisherOne\_002 (9 items)**.

1. To select the primary track and other tracks, click on **FisherOne\_002 (9 items)** and press shift and then click on **FisherOne\_007 (433 items)**. All the six tracks are selected.

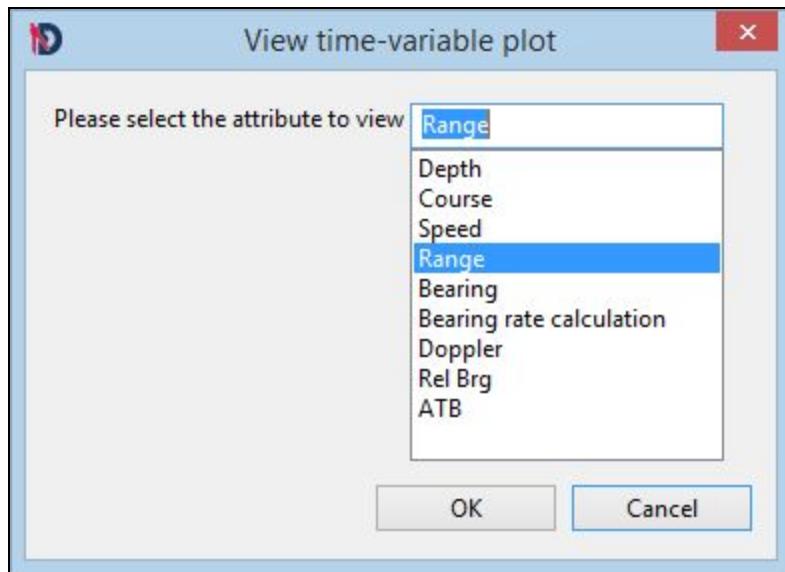


2. Right click on the selected tracks and the select **View XY plot**.



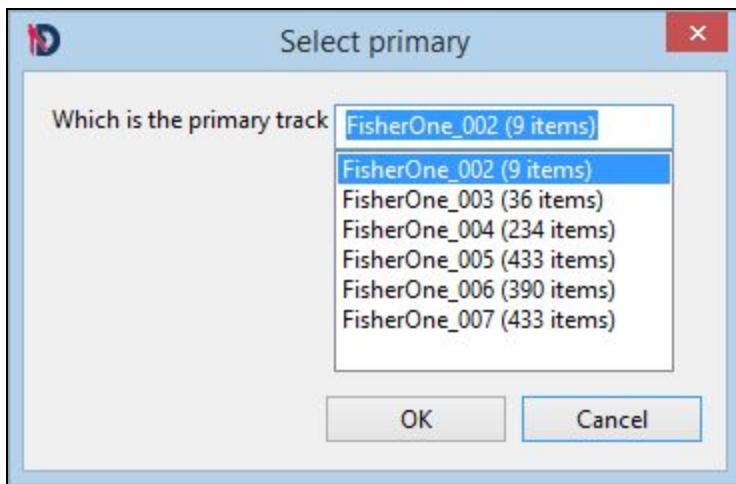


The **View time-variable plot** dialog will display.

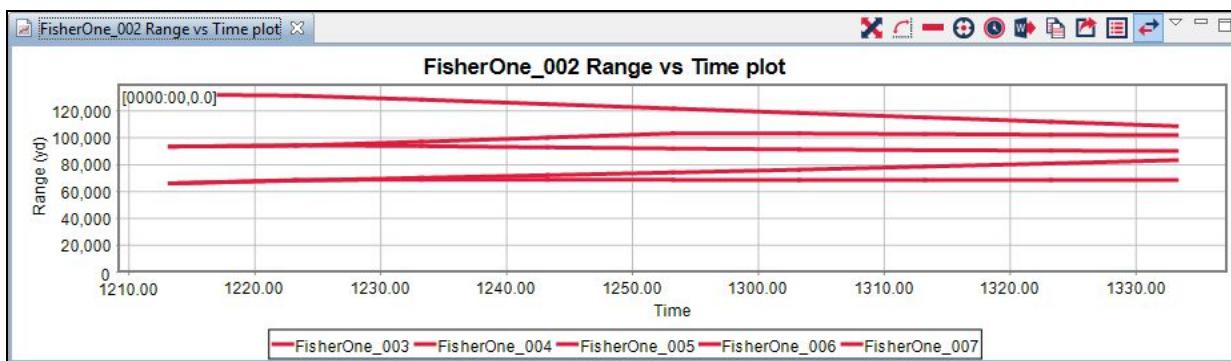


3. Select **Range** and click **OK**. The **Select primary** dialog will display.





4. Select **FisherOne\_002 (9 items)** as the primary track and click **OK**. The **FisherOne\_002 Range vs Time plot** will display.

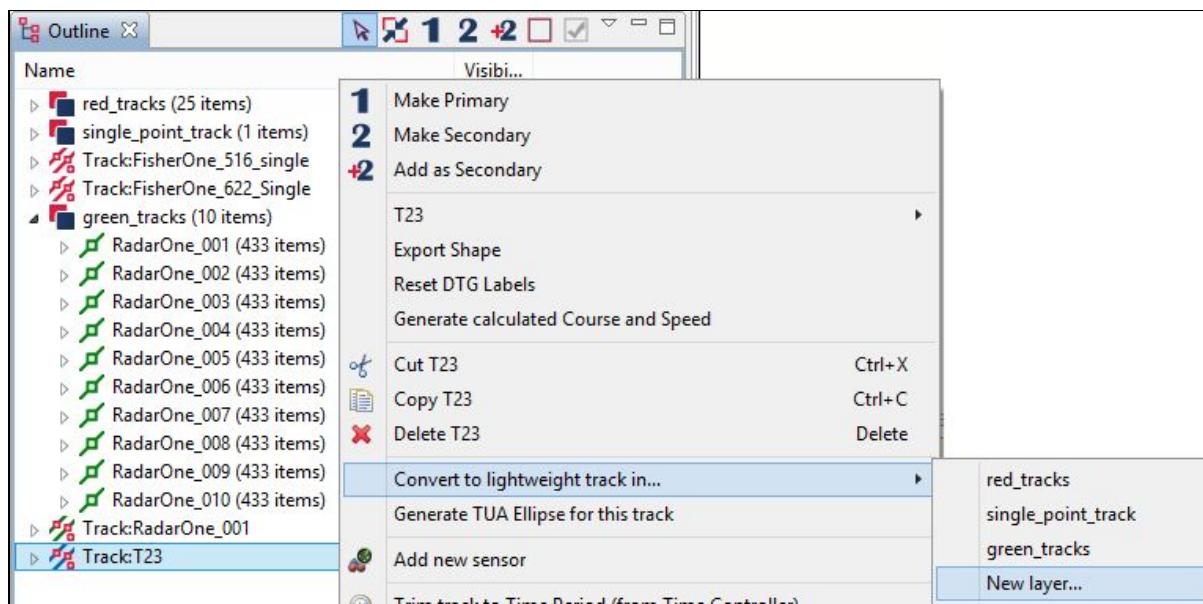


#### 4.3.6 Converting Normal Track to Lightweight Track

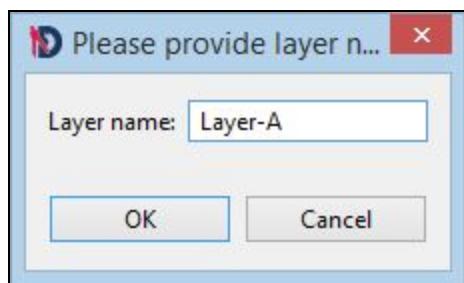
You can convert a normal track into a lightweight track in an existing layer or in a new layer. This will enable you to organize the tracks in layers and manage each layer individually. Now let us first learn how to convert a normal track into a lightweight track in a new layer.

To do so, we will convert the normal track **T23** into a lightweight track in an existing layer, **green\_tracks**.

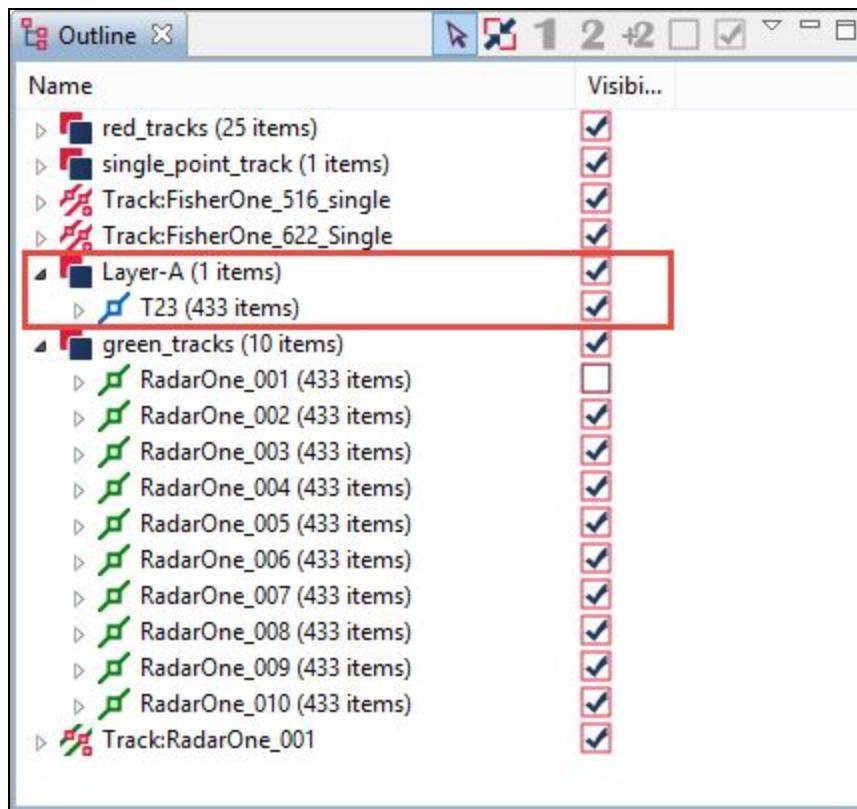
1. In the **Outline** view, select **Track:T23**, right click on it and select **Convert to lightweight track in..** and click **New layer...**



The create layer dialog will display.



2. Enter **Layer name** as **Layer-A** and click **OK**. □
  
3. In the **Outline** view, the track **T23** is converted to a lightweight track and is under the newly created layer **Layer-A**. □



**Note:** To convert a normal track into a lightweight track, select the track to convert, right-click on it and then select an existing layer. In the **Outline** view, the normal track will not be shown instead it will be shown as a lightweight track under the selected layer.

## 4.4 Paste REP from Clipboard

**Note:** REP (short form for Replay Data) is a legacy text file format that Debrief adopted in the mid-90s. The format has been extended to support new data types, such as annotations..

As a scientist, there may be instances where you wish to experiment with copying and pasting experimental data into **Debrief**. You may be getting yourself familiarised about a new REP format or loading data that you have manually entered in a text file. **Debrief** enables you to do this by using the **Paste REP from Clipboard** option. We will learn about this option in this tutorial. By the end of this tutorial you would have learnt to copy text data from the Text Editor and paste it into a plot, copy data from Excel file and paste it into a plot, and copy annotations data and finally paste it into a plot. For detailed information, refer to the following sections:

- [Pasting REP Data from Text Editor](#)
- [Pasting REP Data from Excel File](#)
- [Working with Annotations](#)

## Populating Data in Debrief

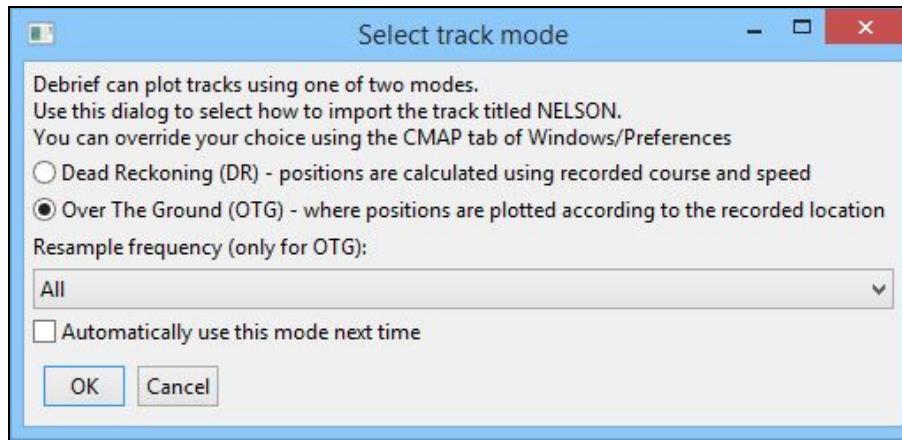
Before we begin with the actual tutorial of copying and pasting data, let us load data into **Debrief**.

1. Let us begin by loading a plot data. In the **Navigator** view, locate the file **boat1.rep**.

Double click on **boat1.rep** to open the file.



2. The **Select track mode** dialog will display.

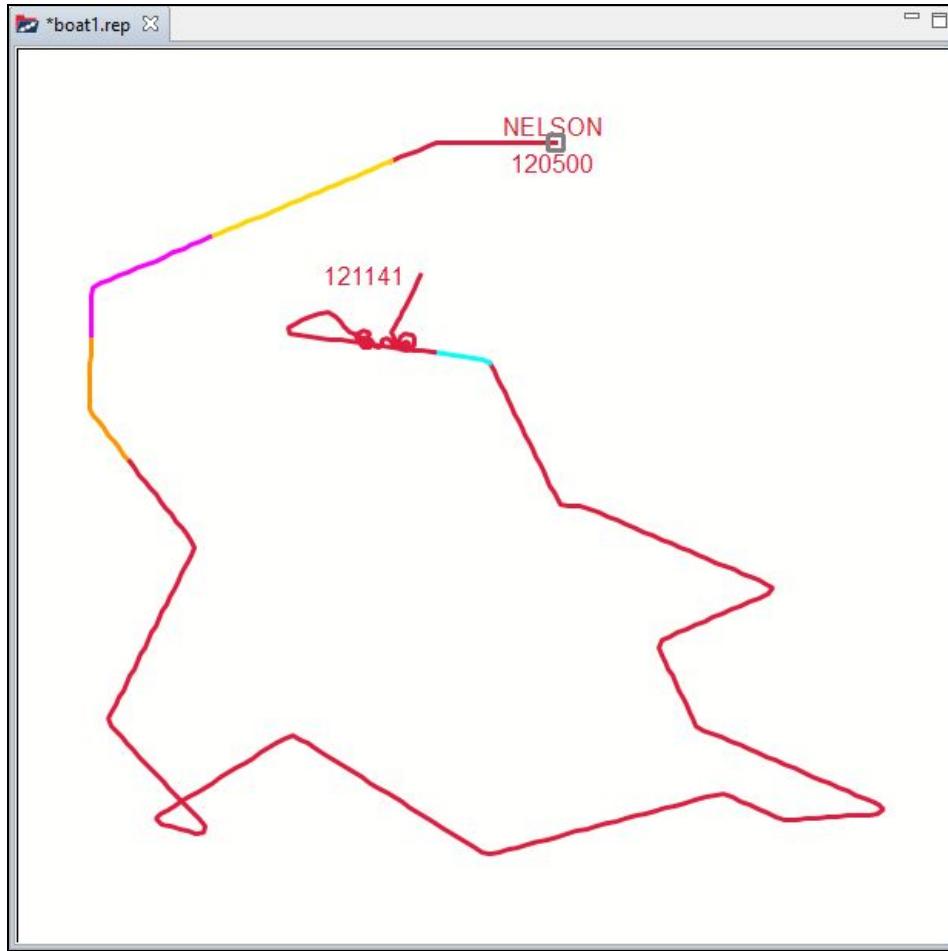


3. Select **Over The Ground** and click **OK**.



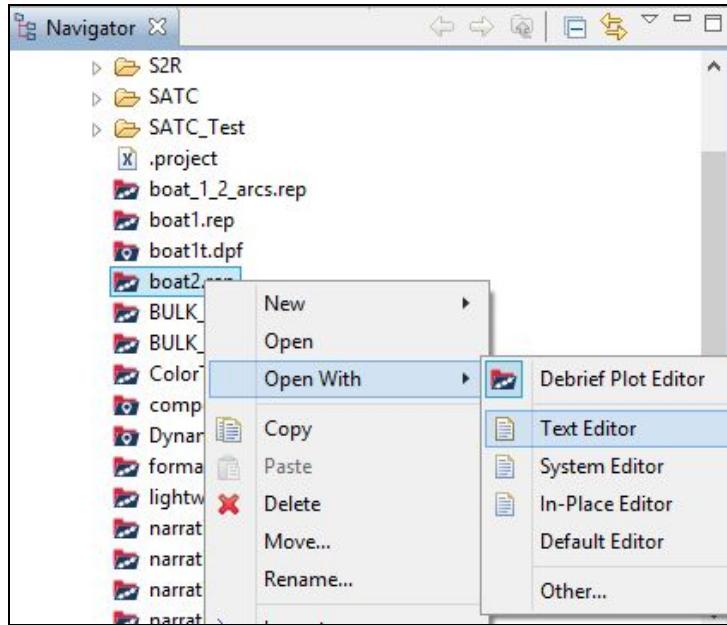
4. The file **boat1.rep** will display in the **Plot Editor**. Click on **Fit to window** button  to view the entire plot.





5. We need some text data. So let us load the text data of **boat2.rep**. In the **Navigator** view, locate the file **boat2.rep**. Right click on **boat2.rep**, select **Open with** and then click **Text Editor**.

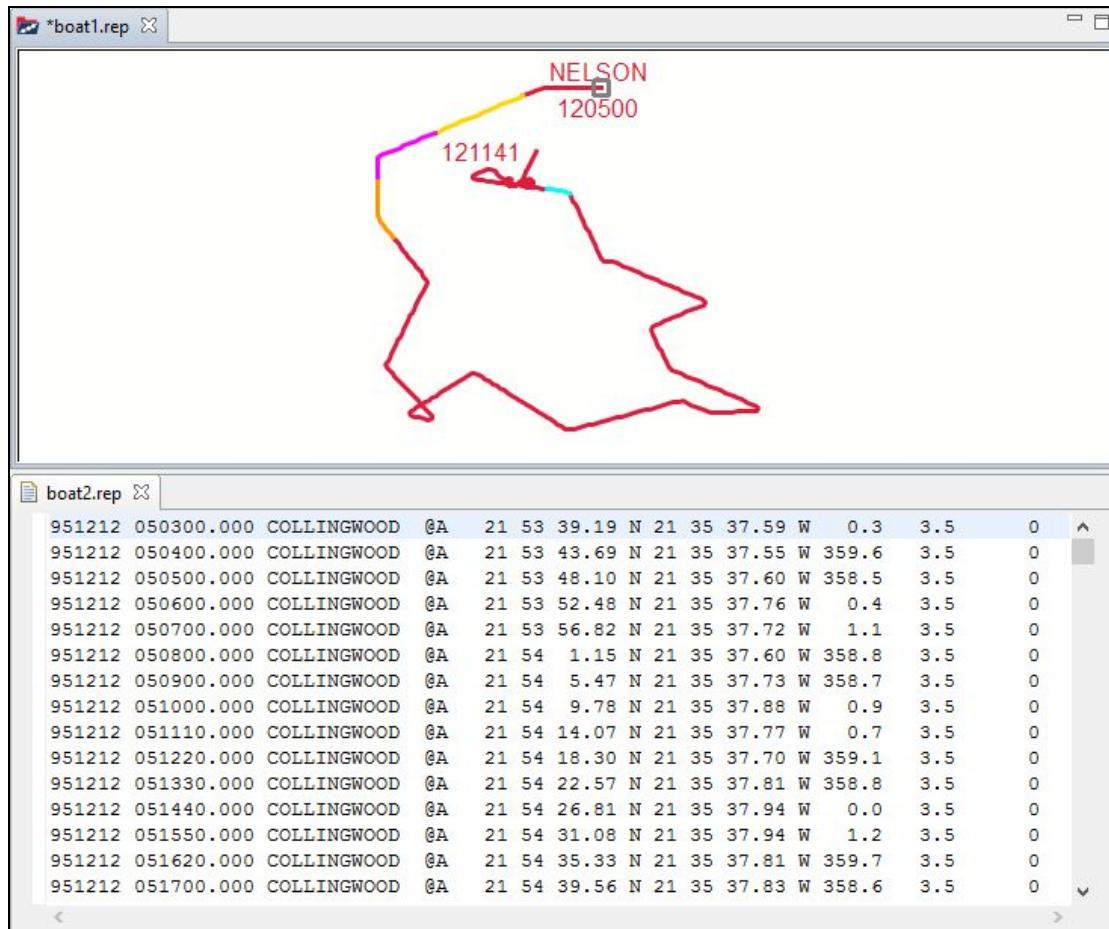




6. **boat2.rep** will display in the text editor.

	*boat1.rep	boat2.rep	
951212	050300.000	COLLINGWOOD	@A
951212	050400.000	COLLINGWOOD	@A
951212	050500.000	COLLINGWOOD	@A
951212	050600.000	COLLINGWOOD	@A
951212	050700.000	COLLINGWOOD	@A
951212	050800.000	COLLINGWOOD	@A
951212	050900.000	COLLINGWOOD	@A
951212	051000.000	COLLINGWOOD	@A
951212	051110.000	COLLINGWOOD	@A
951212	051220.000	COLLINGWOOD	@A
951212	051330.000	COLLINGWOOD	@A
951212	051440.000	COLLINGWOOD	@A
951212	051550.000	COLLINGWOOD	@A
951212	051620.000	COLLINGWOOD	@A
951212	051700.000	COLLINGWOOD	@A

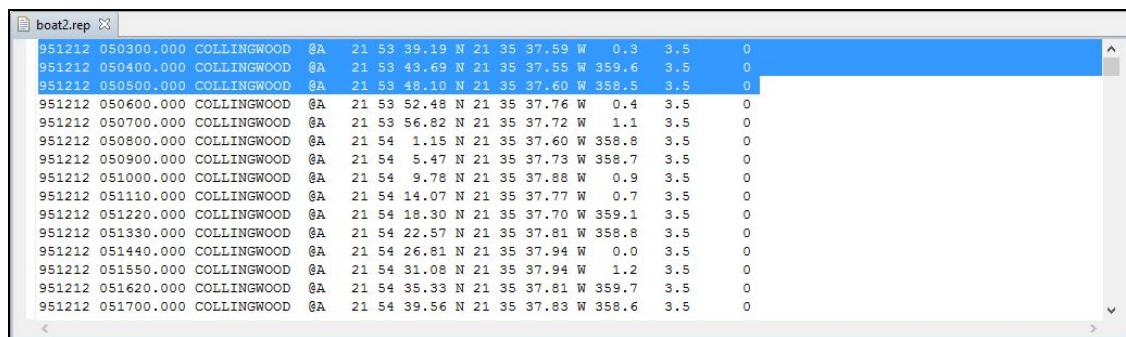
7. For convenience, let us rearrange the **Plot Editor** and the **Text Editor**. Drag the **Text Editor** and drop it below the **Plot Editor**, as shown in the screenshot below.



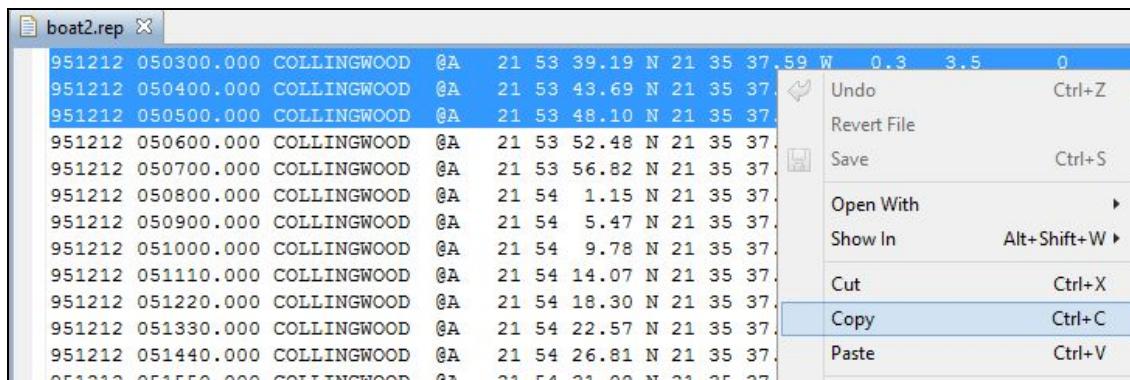
#### 4.4.1 Pasting REP Data from Text Editor

We will begin with learning how to copy data from the **Text Editor** and then pasting it into the **Plot Editor**, using the **Paste REP from clipboard** option.

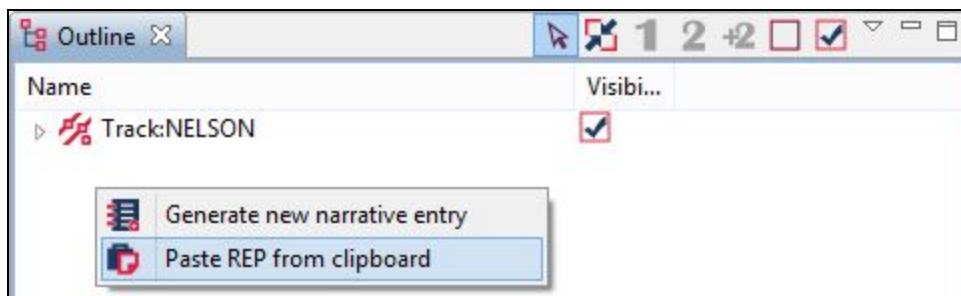
1. We need to copy text data to be pasted. Select the first three lines in the **boat2.rep** file.



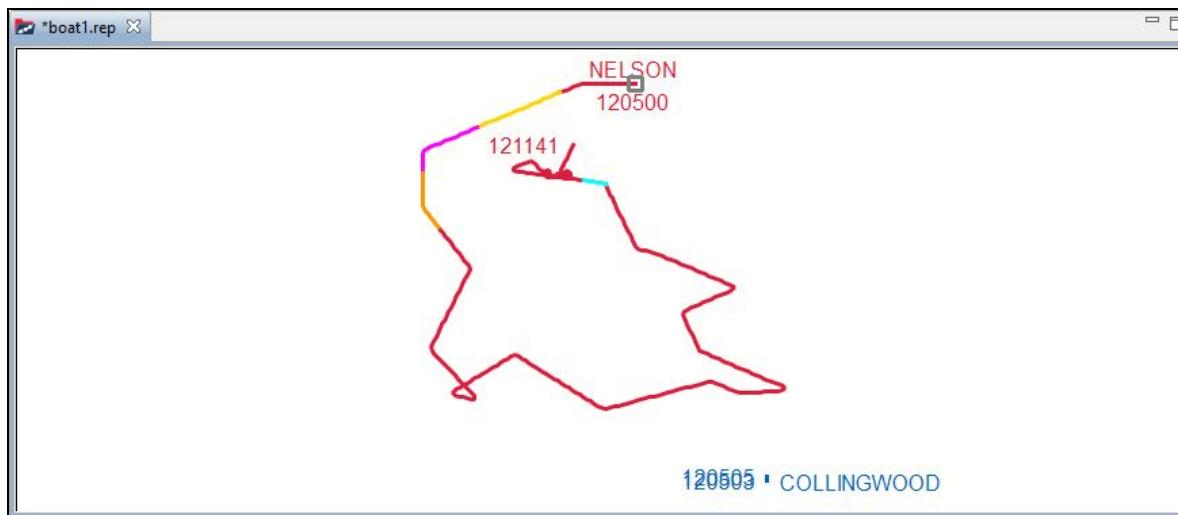
2. Right click on the selected data and select **Copy**.



3. Let us now paste the copied data into the **Outline** view. Note that there is no **Outline** view available for **boat2.rep** because it is text data. We have to paste the data in the **Outline** view of **boat1.rep**. Click on the tab of **boat1.rep** (in the **Plot Editor**). The **Outline** view will display. In the **Outline** view of **boat1.rep**, right click in an empty space and click **Paste REP from clipboard**.

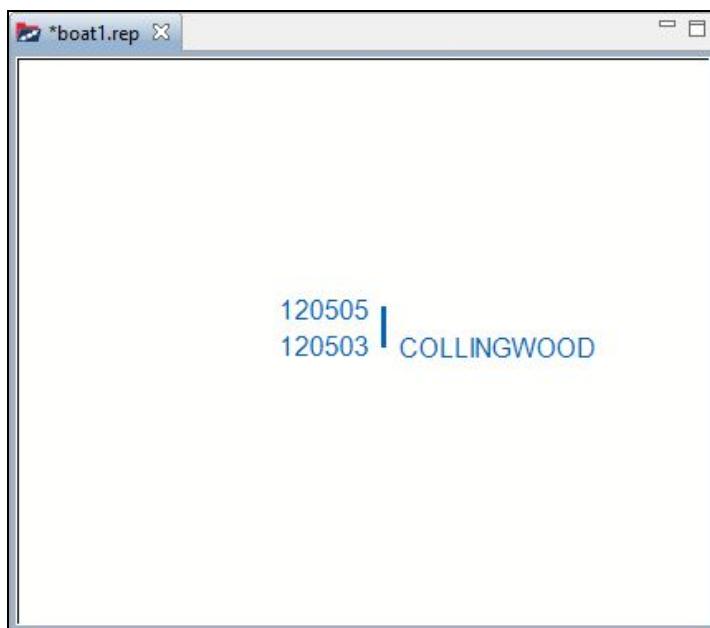


4. The pasted data will display in the **Plot Editor**. If it is not visible, click on **Fit to window** button.



5. Zoom in on the new track that has appeared in the **Plot Editor**. Remember to zoom in,

click on the **Zoom in** button  and click the mouse in the top-right corner, in the **Plot Editor**, and then drag to the bottom-left corner.



6. Let us manually generate some more data. Select the third line of data in the **boat2.rep** file, copy it (to copy you can either right click on it and select **Copy** or press **Ctrl+C**) and paste (to paste you can either right click and select **Paste** or press **Ctrl+V**) it thrice just below it in the **Text Editor**.



7. Now let us change the data values. Refer to the screenshot below and change the values manually. The changes are highlighted in red rectangles in the screenshot.

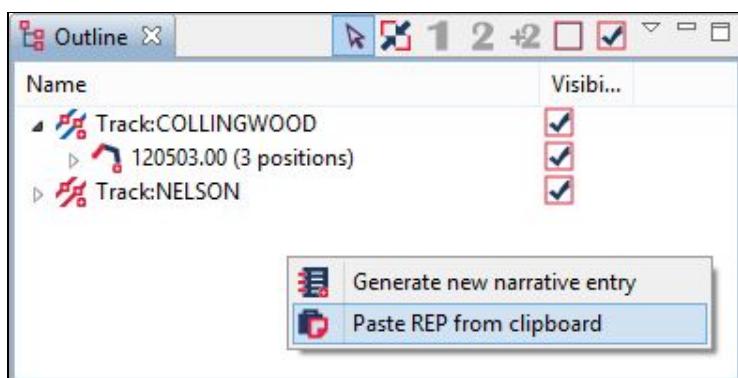


boat2.rep												
951212	050300.000	COLLINGWOOD	GA	21	53	39.19	N	21	35	37.59	W	0.3
951212	050400.000	COLLINGWOOD	GA	21	53	43.69	N	21	35	37.55	W	359.6
951212	050500.000	COLLINGWOOD	GA	21	53	48.10	N	21	35	37.60	W	358.5
951212	050600.000	COLLINGWOOD	GA	21	53	48.10	N	21	35	36.60	W	358.5
951212	050700.000	COLLINGWOOD	GA	21	53	49.10	N	21	35	35.60	W	358.5
951212	050800.000	COLLINGWOOD	GA	21	53	50.10	N	21	35	35.60	W	358.5
951212	050600.000	COLLINGWOOD	GA	21	53	52.48	N	21	35	37.76	W	0.4
951212	050700.000	COLLINGWOOD	GA	21	53	56.82	N	21	35	37.72	W	1.1
951212	050800.000	COLLINGWOOD	GA	21	54	1.15	N	21	35	37.60	W	358.8
951212	050900.000	COLLINGWOOD	GA	21	54	5.47	N	21	35	37.73	W	358.7
951212	051000.000	COLLINGWOOD	GA	21	54	8.70	N	21	35	37.73	W	358.7

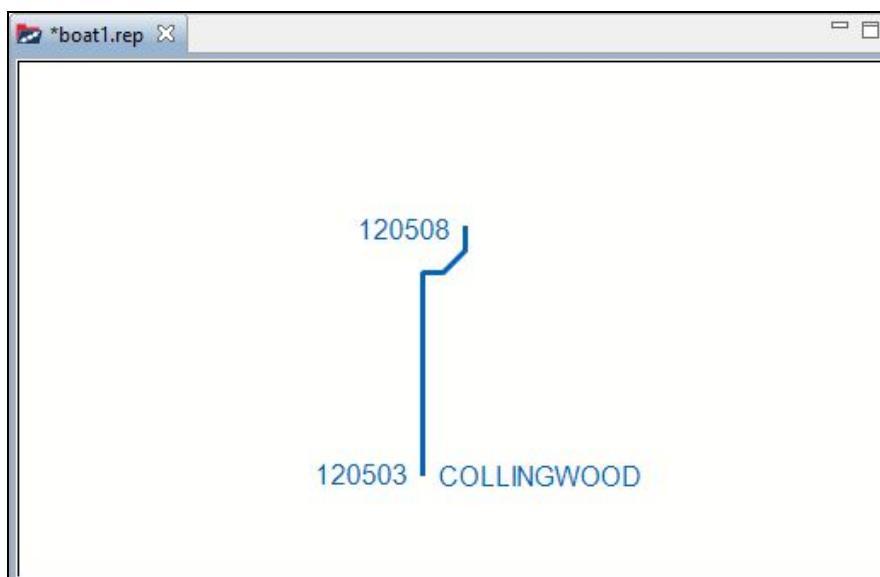
8. Now select and copy the newly entered data.



9. In the **Outline** view of **boat1.rep**, click on the arrow next to **Track:COLLINGWOOD** to expand it. And then right-click in an empty space and click **Paste REP from clipboard**.

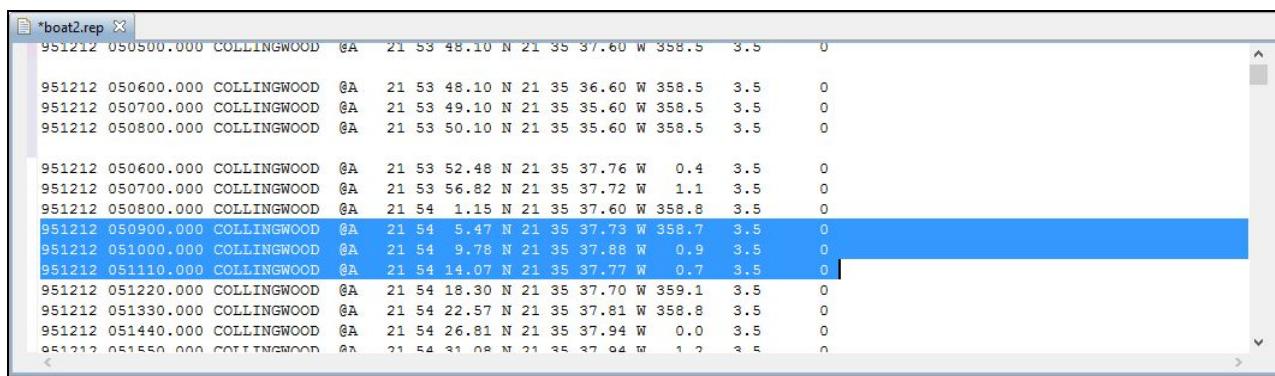
10. The pasted data will display in the **Plot Editor**.



## 4.4.2 Pasting REP Data from MS Excel File

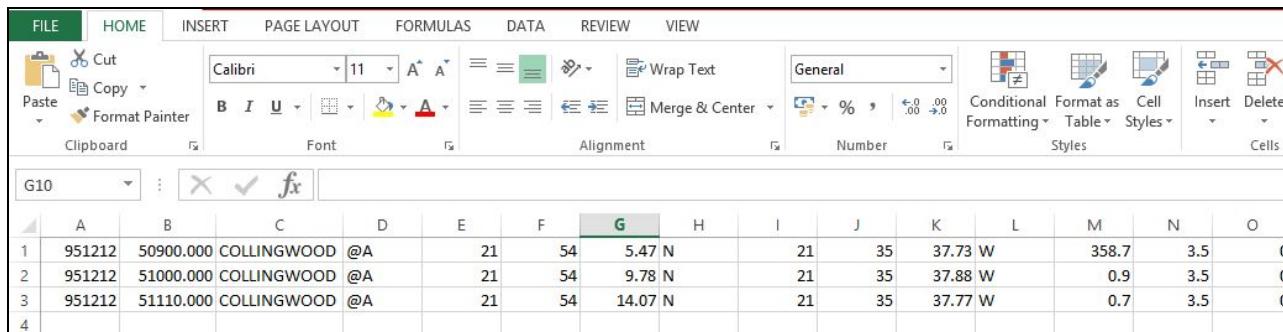
Now that we have learnt how to copy text data from the **Text Editor** and paste it in the **Outline** view, let us now learn how to copy data from MS Excel file and paste it in **Debrief**. **Debrief** lets you to copy data from an Excel file and paste it into the **Outline** view of a plot, using the same **Paste REP from clipboard** option.

1. Let us first populate an Excel file with data. Scroll down in boat2.rep, in the **Text Editor**, and select the data (shown in the screenshot below) and copy it.



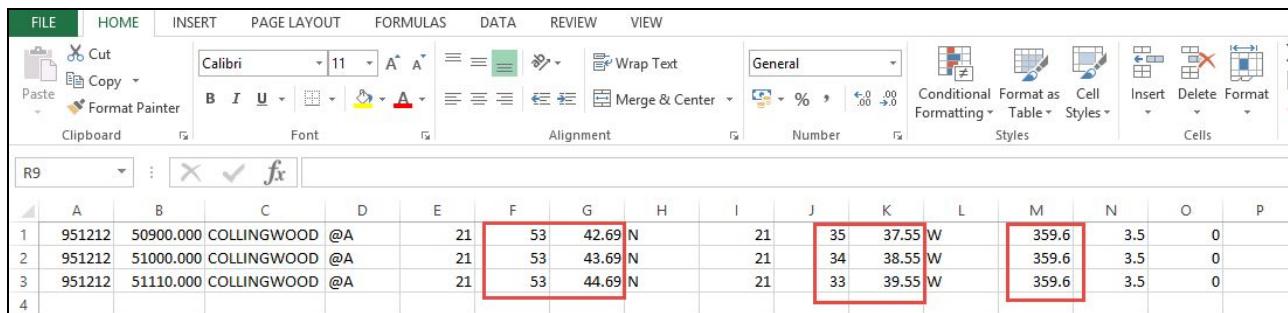
951212 050500.000 COLLINGWOOD @A	21	53	48.10	N	21	35	37.60	W	358.5	3.5	0
951212 050600.000 COLLINGWOOD @A	21	53	48.10	N	21	35	36.60	W	358.5	3.5	0
951212 050700.000 COLLINGWOOD @A	21	53	49.10	N	21	35	35.60	W	358.5	3.5	0
951212 050800.000 COLLINGWOOD @A	21	53	50.10	N	21	35	35.60	W	358.5	3.5	0
951212 050600.000 COLLINGWOOD @A	21	53	52.48	N	21	35	37.76	W	0.4	3.5	0
951212 050700.000 COLLINGWOOD @A	21	53	56.82	N	21	35	37.72	W	1.1	3.5	0
951212 050800.000 COLLINGWOOD @A	21	54	1.15	N	21	35	37.60	W	358.8	3.5	0
951212 050900.000 COLLINGWOOD @A	21	54	5.47	N	21	35	37.73	W	358.7	3.5	0
951212 051000.000 COLLINGWOOD @A	21	54	9.78	N	21	35	37.88	W	0.9	3.5	0
951212 051100.000 COLLINGWOOD @A	21	54	14.07	N	21	35	37.77	W	0.7	3.5	0
951212 051200.000 COLLINGWOOD @A	21	54	18.30	N	21	35	37.70	W	359.1	3.5	0
951212 051330.000 COLLINGWOOD @A	21	54	22.57	N	21	35	37.81	W	358.8	3.5	0
951212 051440.000 COLLINGWOOD @A	21	54	26.81	N	21	35	37.94	W	0.0	3.5	0
951212 051550.000 COLLINGWOOD @A	21	54	31.08	N	21	35	37.94	W	1.2	3.5	0

2. Now open an Excel file and paste the copied data into it.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	951212	50900.000	COLLINGWOOD	@A	21	54	5.47	N	21	35	37.73	W	358.7	3.5	0
2	951212	51000.000	COLLINGWOOD	@A	21	54	9.78	N	21	35	37.88	W	0.9	3.5	0
3	951212	51110.000	COLLINGWOOD	@A	21	54	14.07	N	21	35	37.77	W	0.7	3.5	0
4															

3. Refer to the screenshot below and manually modify the data. The changed data is highlighted in red rectangles.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	951212	50900.000	COLLINGWOOD	@A	21	53	42.69	N		21	35	37.55	W	359.6	3.5	0
2	951212	51000.000	COLLINGWOOD	@A	21	53	43.69	N		21	34	38.55	W	359.6	3.5	0
3	951212	51110.000	COLLINGWOOD	@A	21	53	44.69	N		21	33	39.55	W	359.6	3.5	0
4																

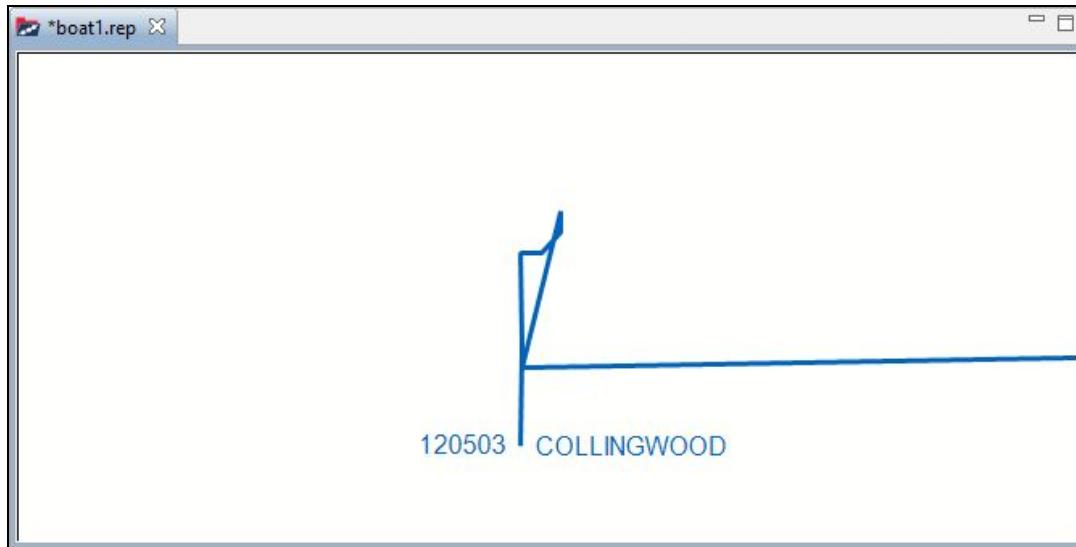
4. Select and copy the data from the Excel file.



5. Switch to **Debrief**. In the **Outline** view of **boat1.rep**, right click in an empty space and select **Paste REP from clipboard**.

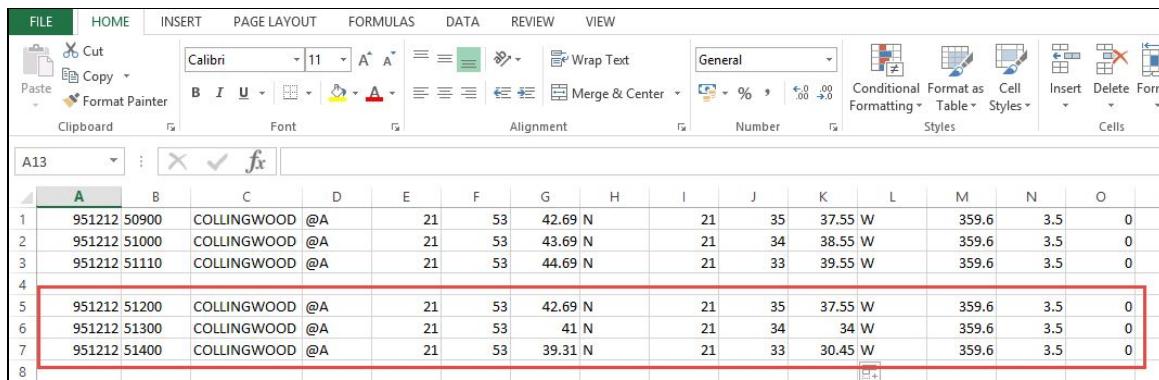


6. The new data will display in the **Plot Editor**.



5. Let us create some more data in Excel. Use Excel to generate more data, refer to the screenshot below. Enter the data manually.





A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	951212 50900	COLLINGWOOD	@A	21	53	42.69 N		21	35	37.55 W		359.6	3.5	0
2	951212 51000	COLLINGWOOD	@A	21	53	43.69 N		21	34	38.55 W		359.6	3.5	0
3	951212 51110	COLLINGWOOD	@A	21	53	44.69 N		21	33	39.55 W		359.6	3.5	0
4														
5	951212 51200	COLLINGWOOD	@A	21	53	42.69 N		21	35	37.55 W		359.6	3.5	0
6	951212 51300	COLLINGWOOD	@A	21	53	41 N		21	34	34 W		359.6	3.5	0
7	951212 51400	COLLINGWOOD	@A	21	53	39.31 N		21	33	30.45 W		359.6	3.5	0
8														

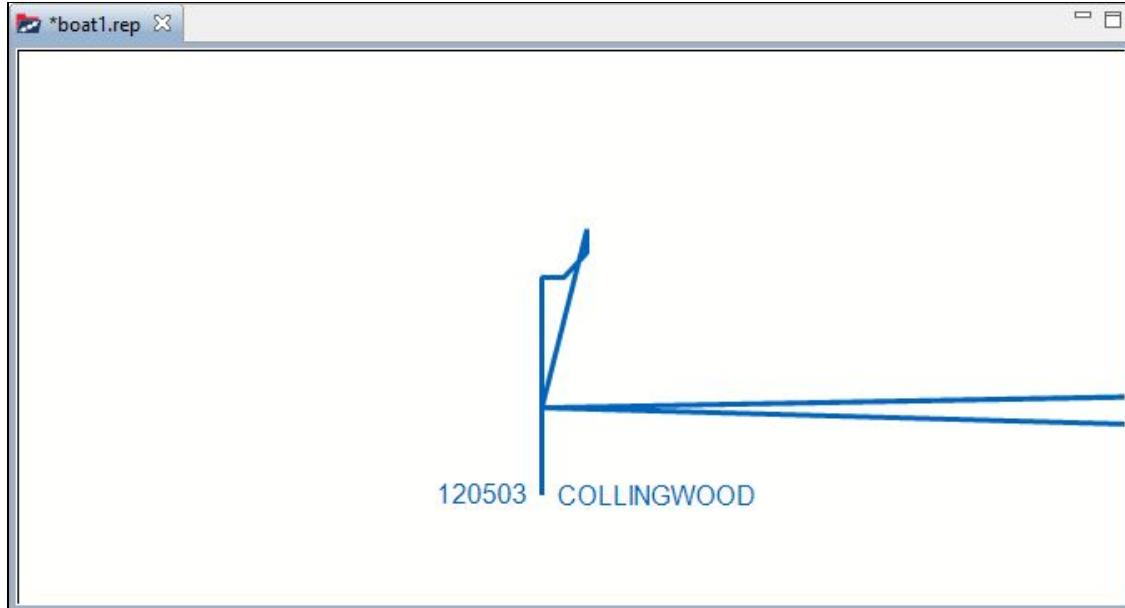
6. Select and copy the newly generated data.



7. Switch to **Debrief**. In the **Outline** view of **boat1.rep**, right click in an empty space and select **Paste REP from clipboard**.



8. The new data will display in the **Plot Editor**.



9. We are done with the text data part. Close **boat2.rep**. Don't save any changes as we do not need them.



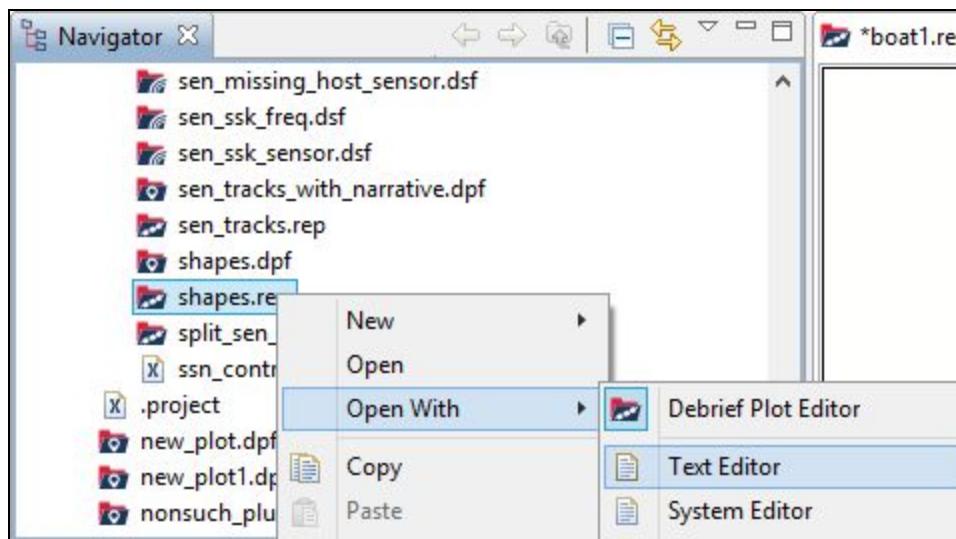
### 4.4.3 Working with Annotations

In addition to copying and pasting track data, **Debrief** allows you to copy and paste annotations using the **Paste REP from clipboard** option. Annotations are the non track-related graphic elements added to a plot. Examples of annotations are rectangles, ellipses and lines. Let us now look at copying annotations and pasting it onto a plot.

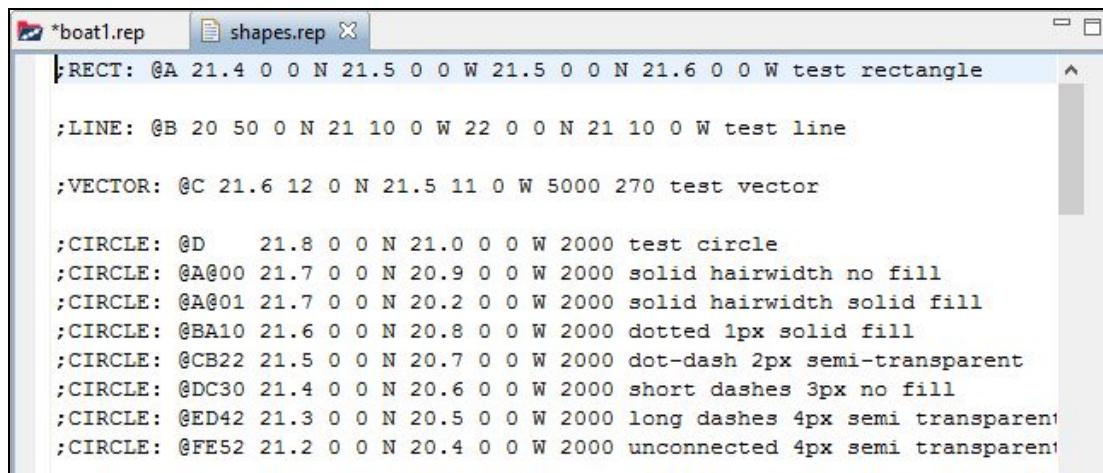
1. Let us open the **shapes.rep** file. Locate the file in the **Navigator** view.



2. Right click on **shapes.rep**, select **Open with** and then click **Text Editor**.

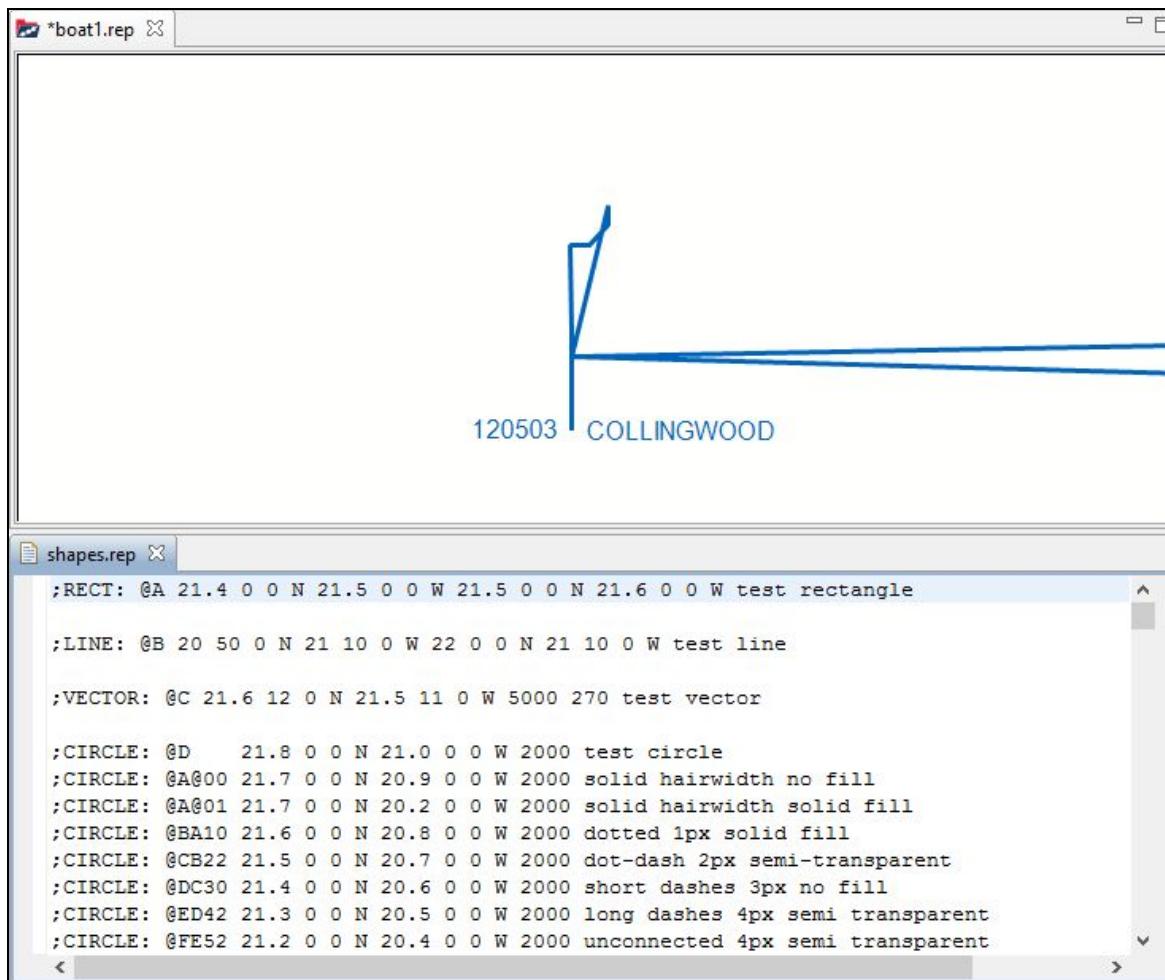


3. The file **shapes.rep** will display in the **Text Editor**.



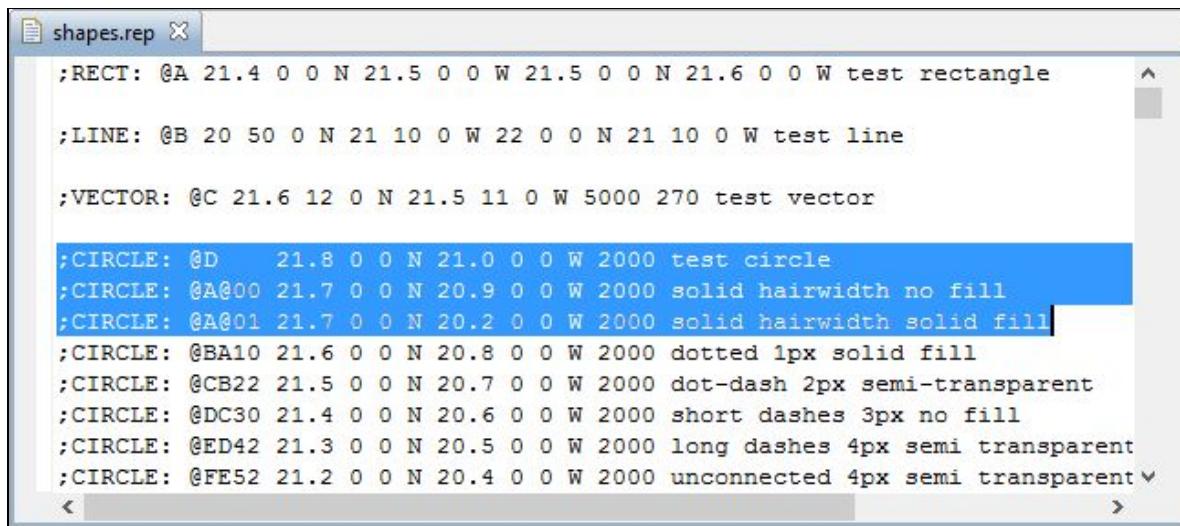
```
;RECT: @A 21.4 0 0 N 21.5 0 0 W 21.5 0 0 N 21.6 0 0 W test rectangle
;LINE: @B 20 50 0 N 21 10 0 W 22 0 0 N 21 10 0 W test line
;VECTOR: @C 21.6 12 0 N 21.5 11 0 W 5000 270 test vector
;CIRCLE: @D 21.8 0 0 N 21.0 0 0 W 2000 test circle
;CIRCLE: @A@00 21.7 0 0 N 20.9 0 0 W 2000 solid hairwidth no fill
;CIRCLE: @A@01 21.7 0 0 N 20.2 0 0 W 2000 solid hairwidth solid fill
;CIRCLE: @BA10 21.6 0 0 N 20.8 0 0 W 2000 dotted 1px solid fill
;CIRCLE: @CB22 21.5 0 0 N 20.7 0 0 W 2000 dot-dash 2px semi-transparent
;CIRCLE: @DC30 21.4 0 0 N 20.6 0 0 W 2000 short dashes 3px no fill
;CIRCLE: @ED42 21.3 0 0 N 20.5 0 0 W 2000 long dashes 4px semi transparent
;CIRCLE: @FE52 21.2 0 0 N 20.4 0 0 W 2000 unconnected 4px semi transparent
```

4. For convenience, let us rearrange the **Plot Editor** and the **Text Editor**. Drag the **Text Editor** and drop it below the **Plot Editor**, as shown in the screenshot below.



5. Select the data (highlighted in the following screenshot) and copy it.





```

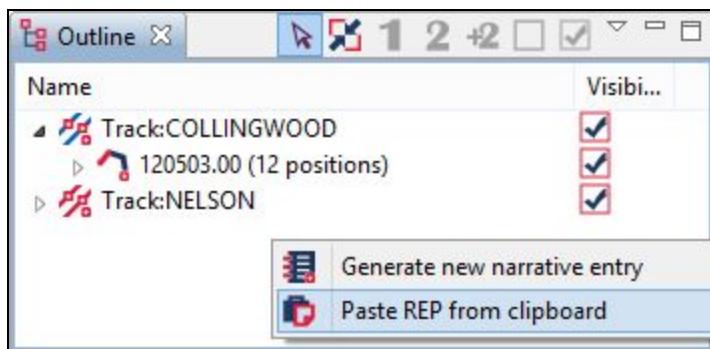
;RECT: @A 21.4 0 0 N 21.5 0 0 W 21.5 0 0 N 21.6 0 0 W test rectangle

;LINE: @B 20 50 0 N 21 10 0 W 22 0 0 N 21 10 0 W test line

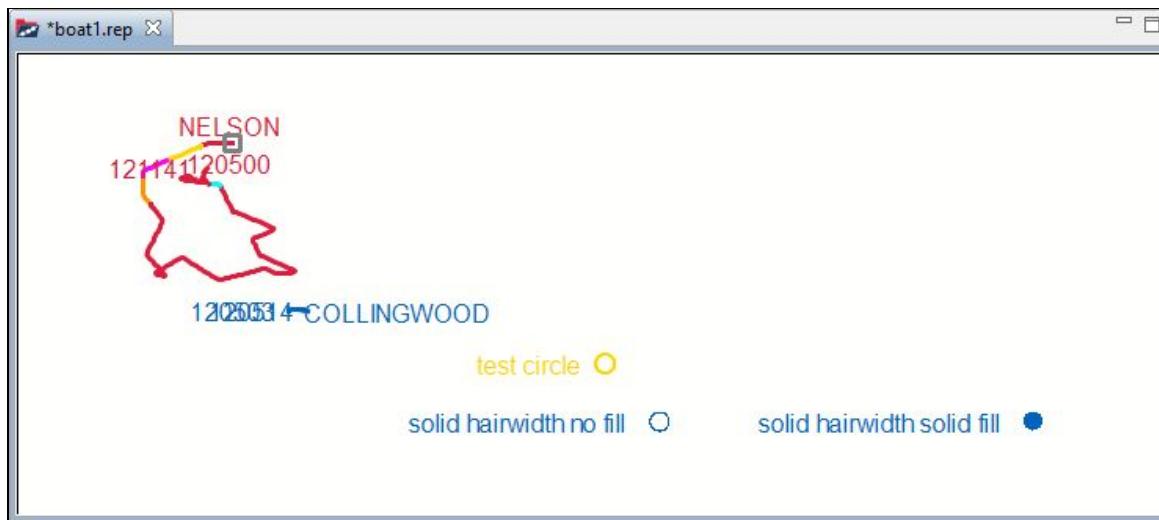
;VECTOR: @C 21.6 12 0 N 21.5 11 0 W 5000 270 test vector

;CIRCLE: @D 21.8 0 0 N 21.0 0 0 W 2000 test circle
;CIRCLE: @A@00 21.7 0 0 N 20.9 0 0 W 2000 solid hairwidth no fill
;CIRCLE: @A@01 21.7 0 0 N 20.2 0 0 W 2000 solid hairwidth solid fill
;CIRCLE: @BA10 21.6 0 0 N 20.8 0 0 W 2000 dotted 1px solid fill
;CIRCLE: @CB22 21.5 0 0 N 20.7 0 0 W 2000 dot-dash 2px semi-transparent
;CIRCLE: @DC30 21.4 0 0 N 20.6 0 0 W 2000 short dashes 3px no fill
;CIRCLE: @ED42 21.3 0 0 N 20.5 0 0 W 2000 long dashes 4px semi transparent
;CIRCLE: @FE52 21.2 0 0 N 20.4 0 0 W 2000 unconnected 4px semi transparent
    
```

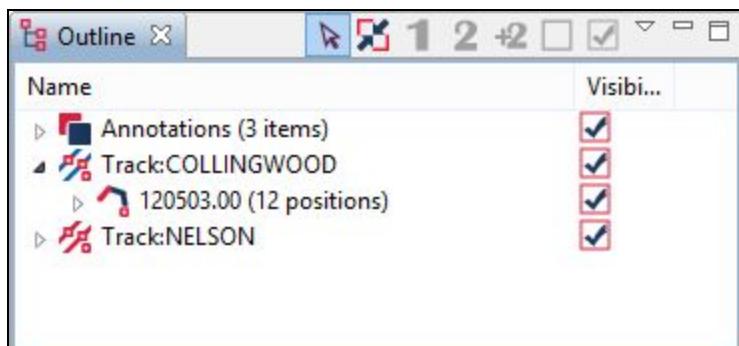
6. In the **Outline** view of **boat1.rep**, right click in an empty space. Click **Paste REP from clipboard**.



7. The annotations will display in the **Plot Editor**. Click on **Fit to window** button  to view the entire plot.



8. Also the **Outline** view is updated accordingly.



9. Now let us experiment with copying some text data from **shapes.rep** and pasting it to **boat1.rep**. Scroll down in the **Text Editor** and copy the text highlighted in the screenshot below.

```

:IRCLE: @BA10 21.6 0 0 N 20.8 0 0 W 2000 dotted 1px solid fill
:IRCLE: @CB22 21.5 0 0 N 20.7 0 0 W 2000 dot-dash 2px semi-transparent
:IRCLE: @DC30 21.4 0 0 N 20.6 0 0 W 2000 short dashes 3px no fill
:IRCLE: @ED42 21.3 0 0 N 20.5 0 0 W 2000 long dashes 4px semi transparent
:IRCLE: @FE52 21.2 0 0 N 20.4 0 0 W 2000 unconnected 4px semi transparent

:TEXT: @E 21.7 0 0 N 21.5 0 0 W test text
:TEXT: WB 21.72 0 0 N 21.52 0 0 W wreck symbol
:TEXT: EB 21.66 0 0 N 21.52 0 0 W

:TEXT: aB[LAYER=SVG_Annotations] 22.16 0 0 N 19.52 0 0 W merchant a
:TEXT: bB[LAYER=SVG_Annotations] 22.06 0 0 N 19.52 0 0 W fishing b
:TEXT: cB[LAYER=SVG_Annotations] 21.96 0 0 N 19.52 0 0 W pleasure_craft c
:TEXT: dB[LAYER=SVG Annotations] 21.86 0 0 N 19.52 0 0 W coastguard law enforcement d
    
```

10. Paste the copied text it in the **Outline** view of **boat1.rep**.



11. The annotations will display in the **Plot Editor**. Click on the **Fit to window** button  to view the entire plot.



## 4.5 Conclusion

That concludes the advanced analysis part of **Debrief**. With the end of this tutorial, you have learnt the basic skills of **Debrief**. In the next tutorial, you will learn all about Scripting in **Debrief**.

Signed: \_\_\_\_\_ Date: \_\_\_\_\_

## 5 Scripting in Debrief

Welcome to the **Debrief** Scripting Tutorial. As an Analyst, you may want to add functionality directly to **Debrief** in order to support specific data manipulation tasks, including adding data or importing data from unknown file types. The ability to control **Debrief** from a macro language will allow users to introduce extra functionalities to **Debrief**, without actually modifying the original application. Scripting is introduced in **Debrief** to enable power users to add more features / capabilities to **Debrief**, by writing their own scripts. The objective of scripting is to expose the object model of **Debrief** and let the advanced users make changes to the data. To facilitate this, the **Scripting Perspective** is introduced. In **Debrief**, scripting can also be used for:

- Applying bulk changes to Debrief objects
- Importing unexpected data-types from files
- Conducting ad-hoc calculations on Debrief data

In this tutorial, you will learn the **Scripting Perspective** feature of **Debrief**. On completion you will know how to record a script, access **Debrief** objects, create a script in a file, configure a script to create toolbar buttons, and create keyboard shortcuts to run scripts. You will also learn how to debug scripts using the Debug Perspective.

We will cover the following sections in this tutorial:

- [Scripting Perspective Overview](#)
- [Understanding Scripting Perspective UI](#)
- [Setting Path for Scripts Folder](#)
- [Opening Scripting Perspective](#)
- [Recording Scripts](#)
- [Accessing Debrief Objects](#)
- [Creating Scripts in File](#)
- [Debugging in the Debug Perspective](#)

### 5.1 Scripting Perspective Overview

The **Scripting Perspective** feature of **Debrief** allows you add extra features to either streamline existing **Debrief** capabilities, or to add new ones. It allows you to interact with **Debrief** objects such as Track & sensor data, Annotations, Chart features, integrate scripts into **Debrief** user interface, associate scripts with elements in Outline View, integrate macros as toolbar buttons, in the **Debrief** interface.

If and when any of these new features / capabilities are found to have lasting usage and wide appeal, then the feature(s) may get integrated into core **Debrief**, for all users.

Javascript is the scripting language used to support in **Debrief** Scripting for the reason that the Java library that **Debrief** is developed with includes a Javascript interpreter, which significantly eases the deployment of **Debrief** with an internal scripting capability..

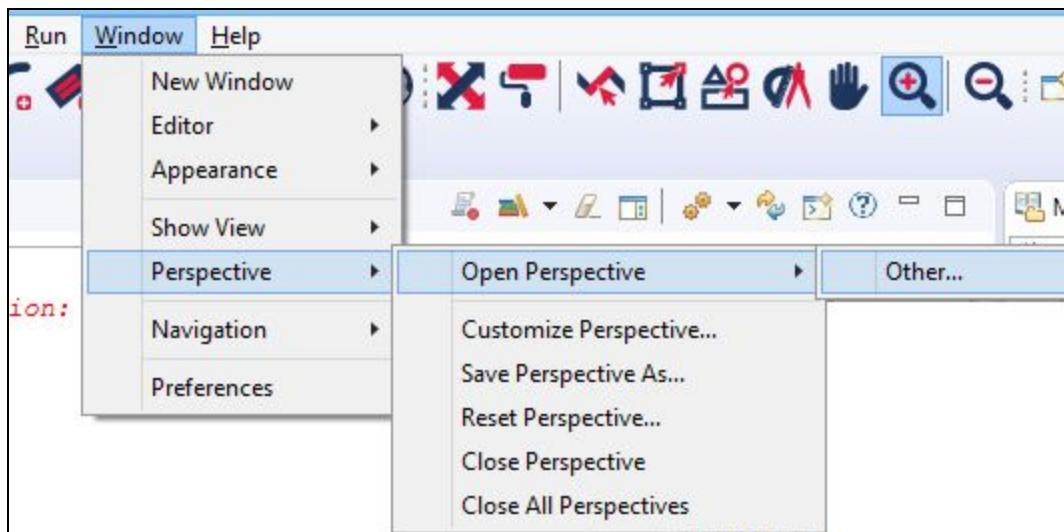
There are two main capabilities in **Debrief** Scripting and they are:

- **Modules** - These are packages of existing **Debrief** functionality that are made available to users. The modules, in addition to allowing control of a **Debrief** plot, also act as a channel to the deeper levels of the **Debrief** object model.
- **Scripts** - These are individual Javascript files that are under the control of the user. They contain the control logic necessary for the functionality required by the user, along with calls to **Debrief** modules. You can store your script files in your **Debrief** workspace, and edit/run them when required. When suitably configured, **Debrief** can integrate these scripts into its UI.

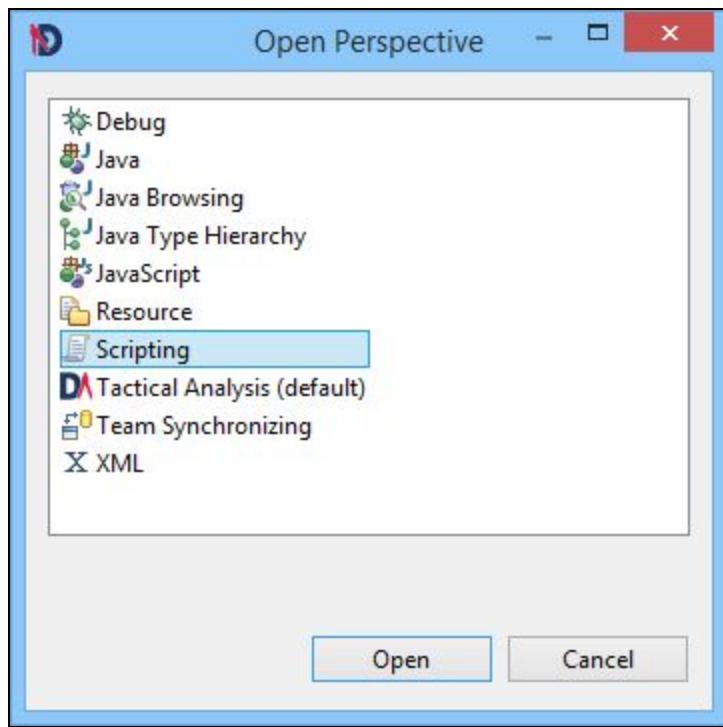
## 5.2 Opening Scripting Perspective

Let's now open the **Scripting Perspective** and learn how to use it.

1. To open the Scripting Perspective, navigate to **Window > Perspective > Open Perspective > Other...**



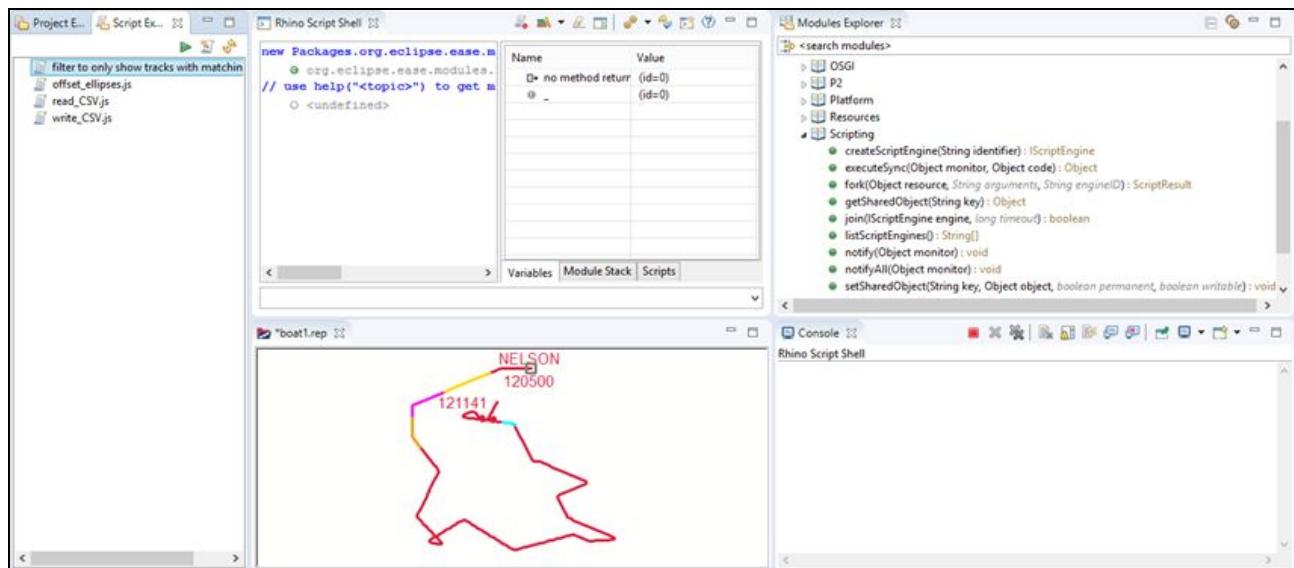
The **Open Perspective** dialog will display.

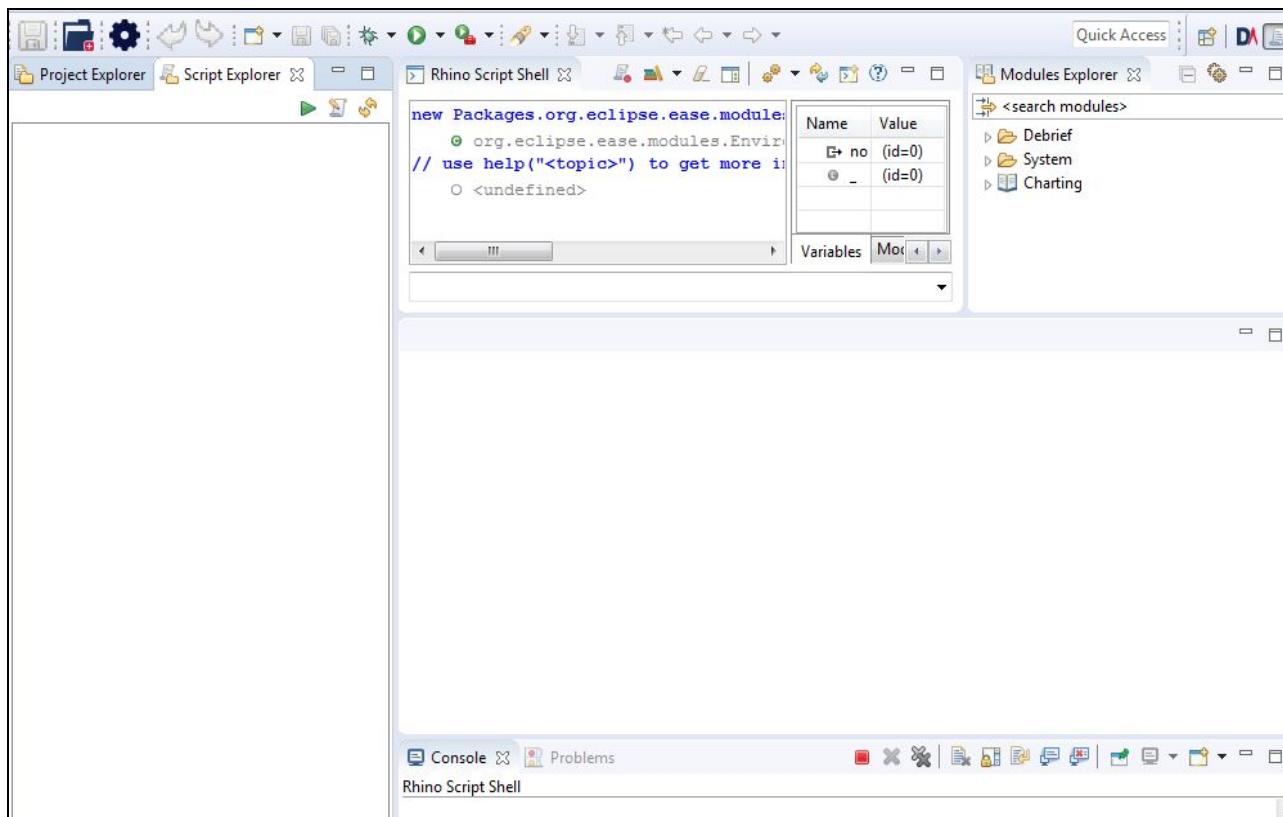


2. Select **Scripting** option and then click **Open**.

**Note:** When you open Scripting Perspective, by default, the Modules Explorer view and Script Shell will open. In case these views do not open, you can open them from **Window > Show View > Script Shell**.

The **Scripting Perspective** will display.





## 5.3 Setting Path for Scripts Folder

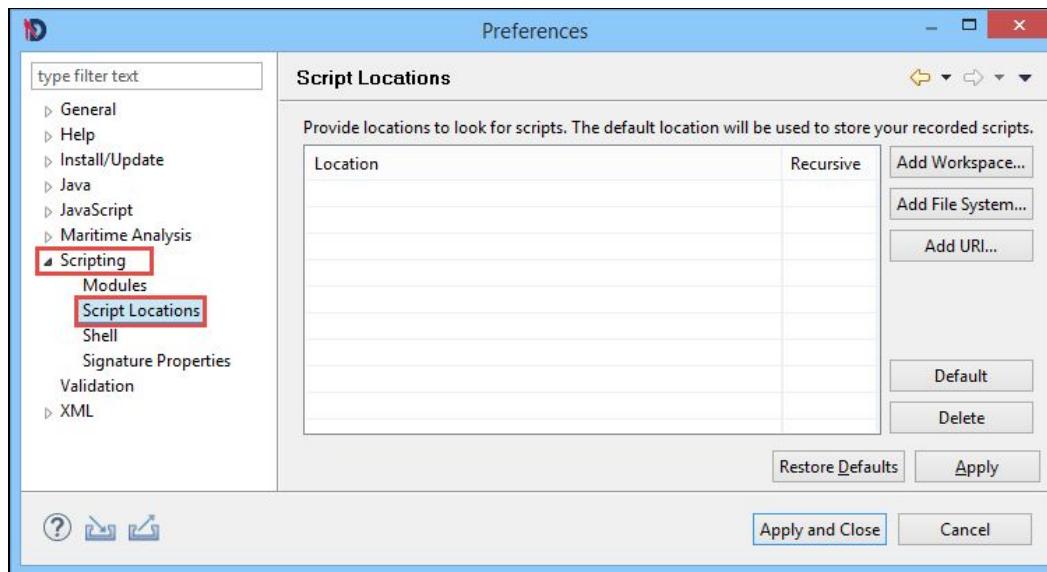
We have to set the path for the scripts folder to let **Debrief** know where your scripts are located. Let's begin with setting the path.

1. Navigate to **Windows > Preferences**.



The **Preferences** dialog will display.





2. Click **Scripting** on the left side and then click **Script Locations**.



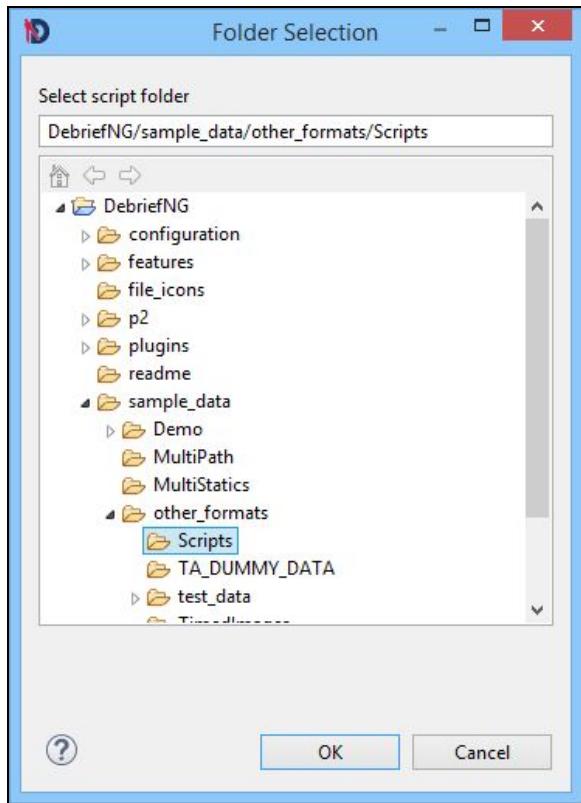
3. Click **Add Workspace...**



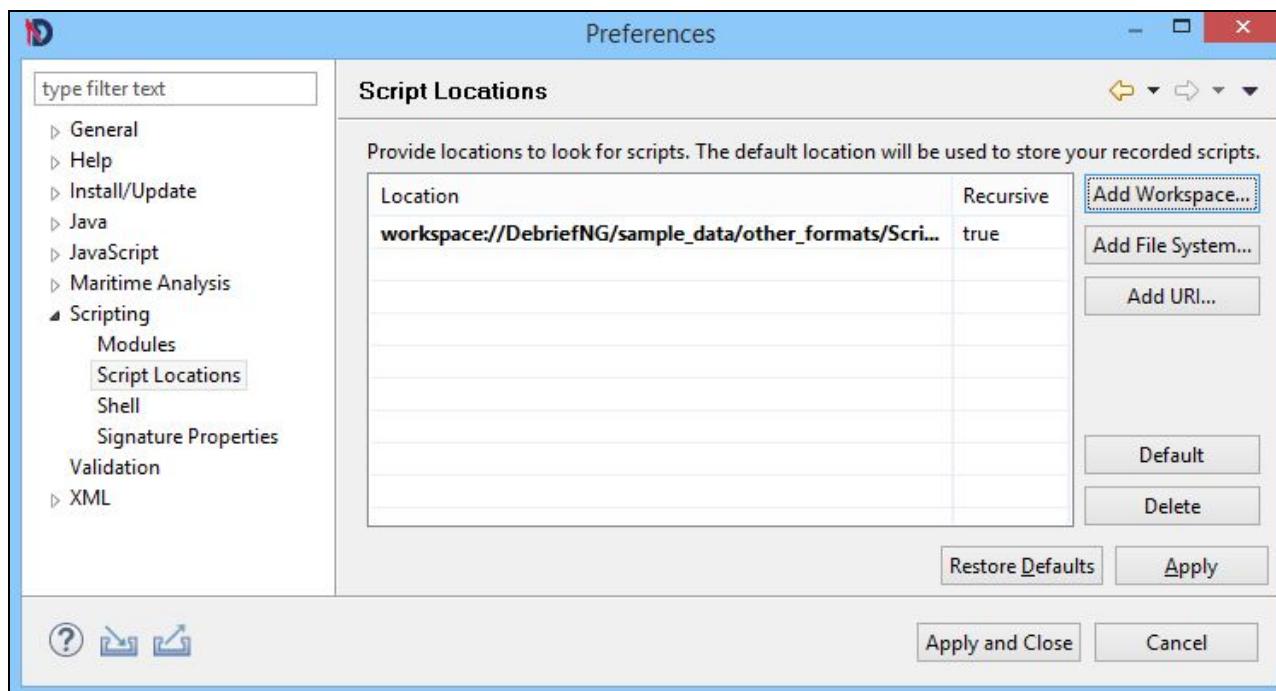
**Note:** Make sure you have created a project in your workspace. In case you have not created a project, refer to section **1.8.1 Generating a Project for Your Data**, in **Tutorial 1**.



The **Folder Selection** dialog will display.



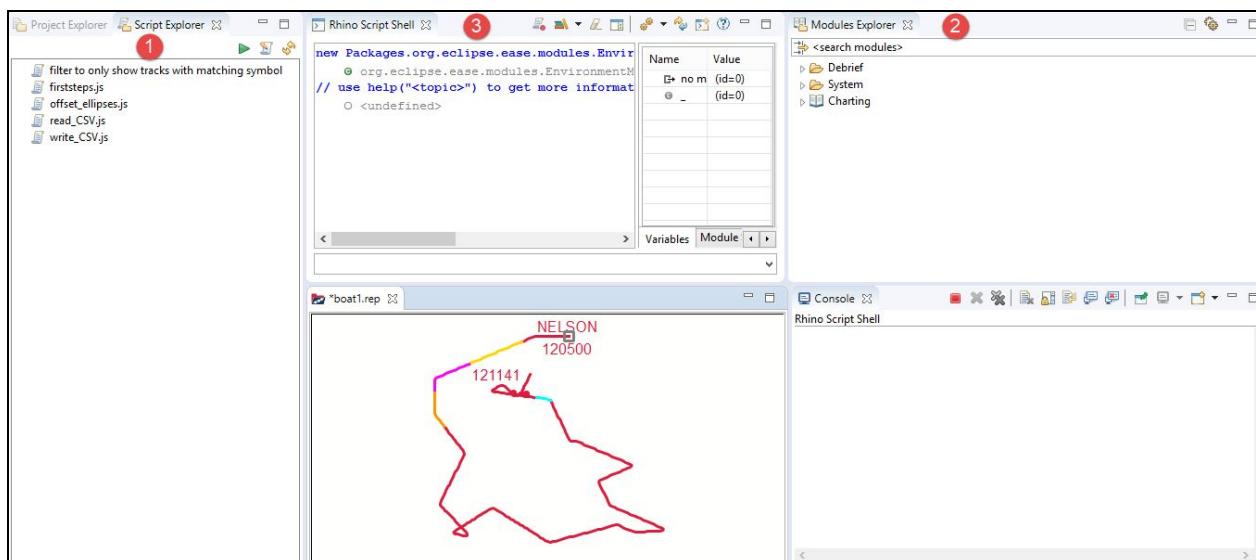
4. Select a folder, on your system, where you wish to save your scripts. If you have not already created a folder to save your scripts, you can select the **sample\_data/other\_formats/Scripts** folder of your **Debrief** installation.
5. The selected location will display in the **Preferences** dialog. Now click **Apply and Close**.



The scripts folder path will be set.

## 5.4 Understanding Scripting Perspective UI

The Scripting Perspective interface of **Debrief** provides effective support for scripting. The user interface for the **Debrief** Scripting Perspective is shown in the below screenshot.

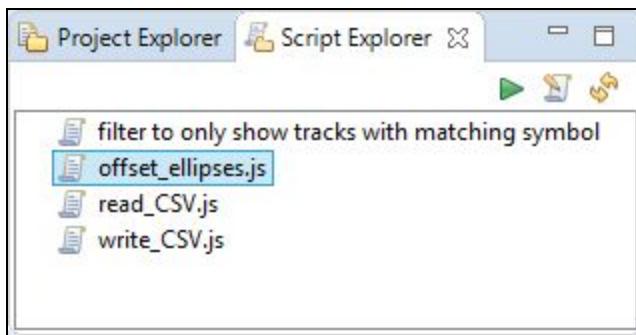


The Scripting Perspective interface has three main windows:

1. **Script Explorer**
2. **Modules Explorer**
3. **Script Shell**

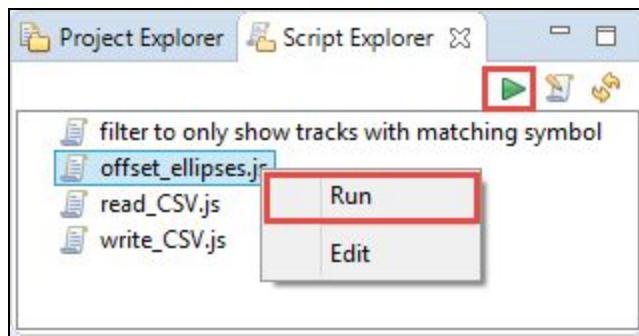
Now, let's look at each window in detail:

1. **Script Explorer:** Similar to how the **Navigator** view displays projects and other resources that you can work with in normal **Debrief** analysis, the **Script Explorer** provides an easy-access to scripts that you have created. To let **Debrief** know where your scripts are located you should first set the path of the folder. Refer to [Setting Path for Scripts Folder](#) to set the path. Below is a sample screenshot of the **Script Explorer**.



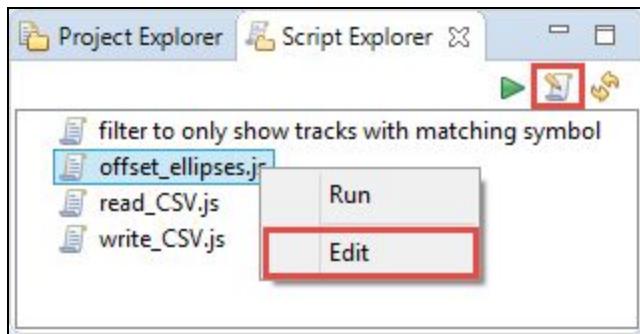
To run a script you can either:

- Click on the script that you wish to run and then click the **Run Script** button  on the **Script Explorer** toolbar or
- Right-click on the script you wish to run and select **Run** from the pop-up menu.

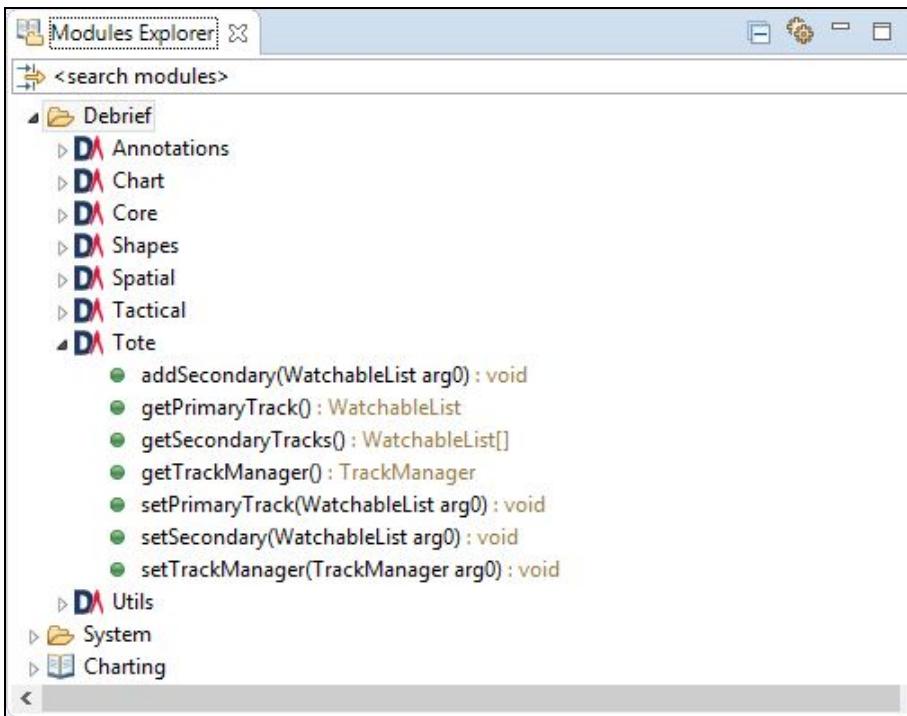


Similarly, to edit a script you can either:

- Click on the script that you wish to edit and then click the **Edit Script** button  on the **Script Explorer** toolbar or
- Right-click on the script that you wish to edit and select **Edit** from the pop-up menu.

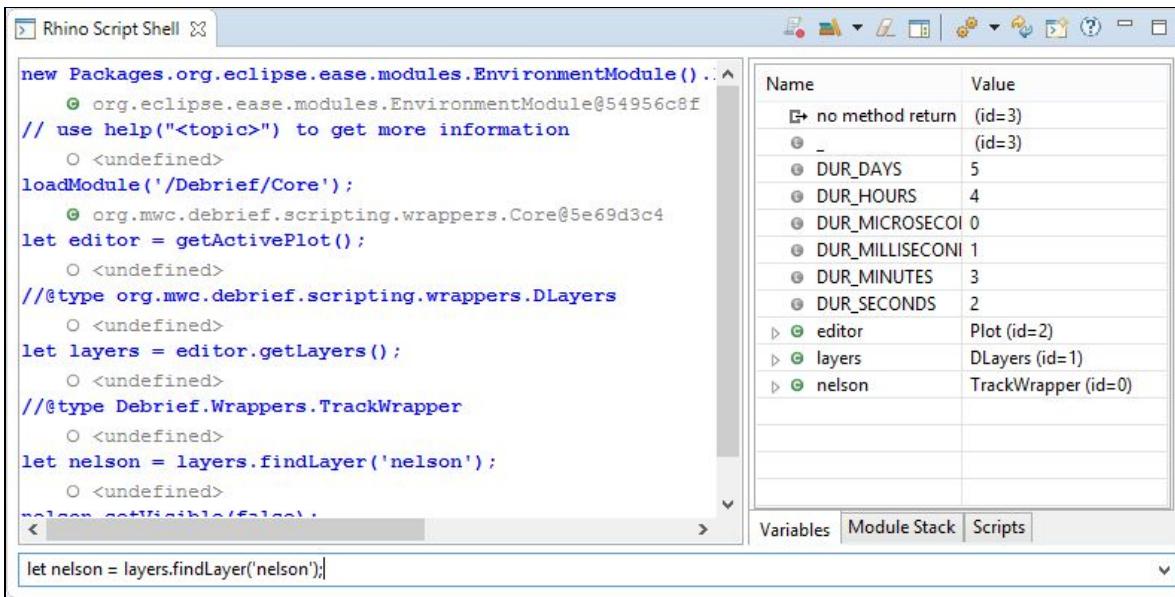


2. **Modules Explorer:** This view is your route into the **Debrief** object model. It displays the list of callable methods that are organised into various categories. To use them, you can drag and drop these methods from the **Modules Explorer** into the **Script Shell**. We will look at this function in detail later in this tutorial. The modules in this view combine both ways of interacting with the **Debrief** object model and a range of other interactions - such as showing popup dialogs, opening files, and drawing XY Charts. Below is a sample screenshot of the **Modules Explorer**.



3. **Script Shell:** This view allows you to interact live with objects of **Debrief**. You can make changes to **Debrief** objects directly through this view, without creating a new script. This view allows you to experiment with calling the **Debrief** modules as well. To reuse a previous line, just double-click on it, and it will be pulled back into the edit control. It also includes a live **Variables** view on the variables that are currently in scope, with the

ability to drill down into their contents. Below is an sample screenshot of the **Script Shell**.



The screenshot shows the "Rhino Script Shell" window. On the left, there is a code editor containing a script. The script starts with "new Packages.org.eclipse.ease.modules.EnvironmentModule()", followed by comments and several "let" statements. On the right, there is a table titled "Variables" with columns "Name" and "Value". The table lists various variables and their values, such as "no method return" (id=3), "DUR\_DAYS" (5), "DUR\_HOURS" (4), and "editor" (Plot (id=2)). At the bottom of the window, there are tabs for "Variables", "Module Stack", and "Scripts".

```

new Packages.org.eclipse.ease.modules.EnvironmentModule()...
    org.eclipse.ease.modules.EnvironmentModule@54956c8f
// use help("<topic>") to get more information
    <undefined>
loadModule('/Debrief/Core');
    org.mwc.debrief.scripting.wrappers.Core@5e69d3c4
let editor = getActivePlot();
    <undefined>
//@type org.mwc.debrief.scripting.wrappers.DLayers
    <undefined>
let layers = editor.getLayers();
    <undefined>
//@type Debrief.Wrappers.TrackWrapper
    <undefined>
let nelson = layers.findLayer('nelson');
    <undefined>
nelson.setVisible(true);
    <undefined>
let nelson = layers.findLayer('nelson');
    <undefined>

```

Now that we have sufficient knowledge of the Scripting Perspective and its UI components, it is now time to get some hands on experience.

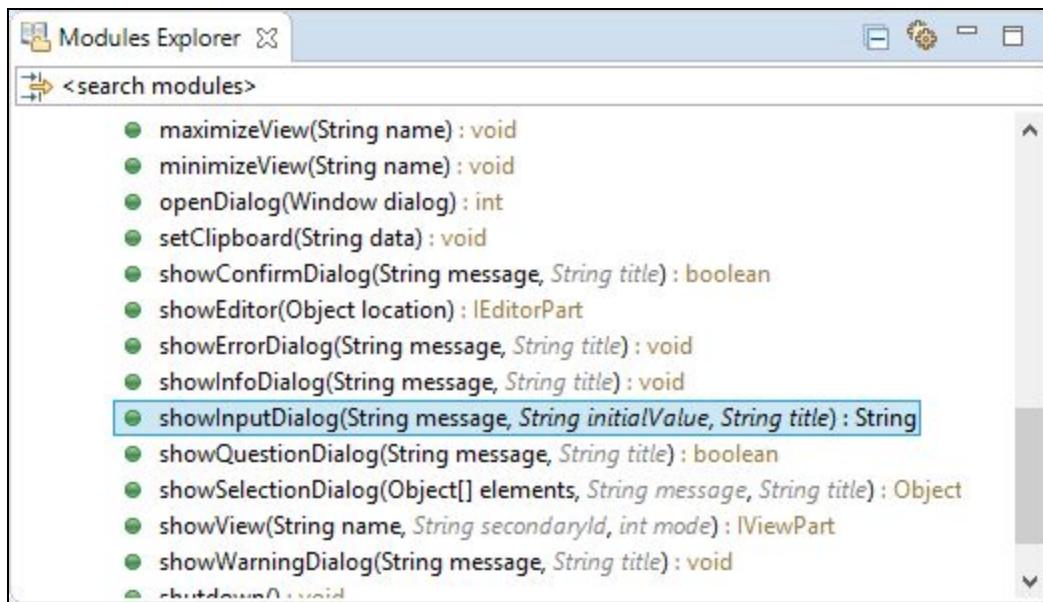
## 5.5 Recording Scripts

**Debrief** allows you to record scripts in a file, via the Scripting Perspective feature. Now let us learn how to do this.

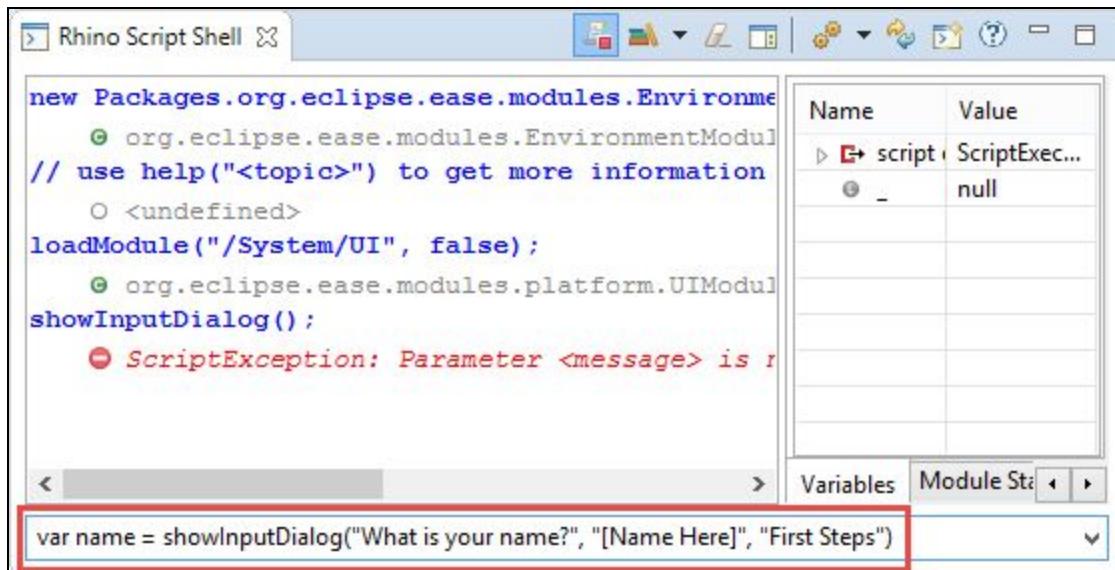
1. In the **Script Shell**, click the **Toggle Script Recording** button  .
2. Now from the Modules Explorer, drag **System > UI > showInputDialog()** and drop it into the Script Shell.

**Note:**

- When you drag the **ShowInputDialog()** and drop it in the script shell, you will get a Script Exception message "**Parameter <message> is not optional**". Ignore the message. However the UI Module will load successfully.
- As an alternate method, you can drag **System > UI** and drop it into the **Script Shell**. And then run the command `var name = showInputDialog("What is your name?", "[Name Here]", "First Steps")` in the Script Shell.



3. Now double-click the **showInputDialog()** in the Script Shell to pull it back into the edit control.
4. Edit the command `var name = showInputDialog("What is your name?", "[Name Here]", "First Steps")` in the Script Shell.

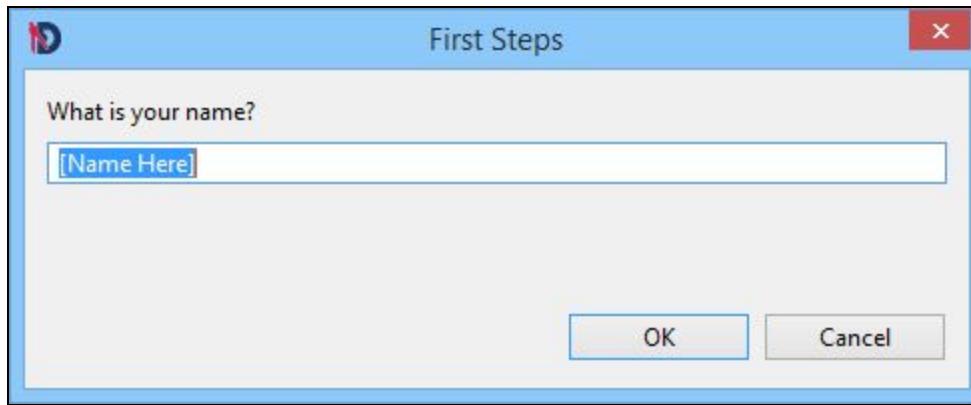


**Note:** To auto-complete a command you can use the keyboard shortcut **Ctrl+space**.

5. Press **Enter**.



The dialog named **First Steps** will display.

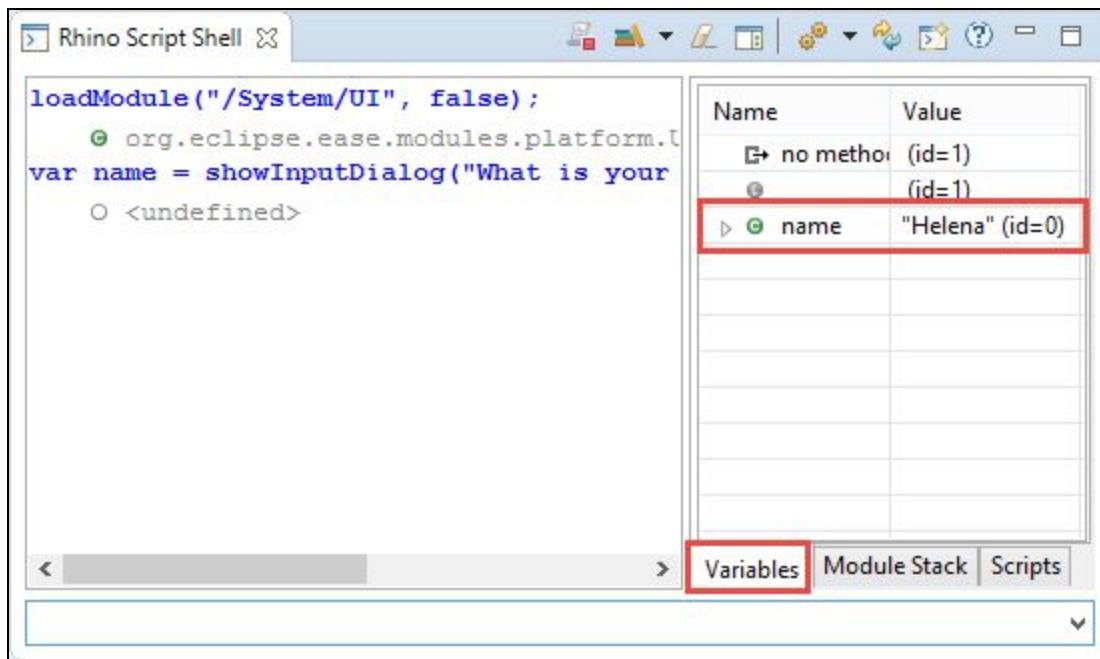


6. Enter your name in the **What is your name?** field and click **OK**.

**Note:** If you do not enter a name in the dialog, the default value **[Name Here]** will be taken as the value of the variable **name**.



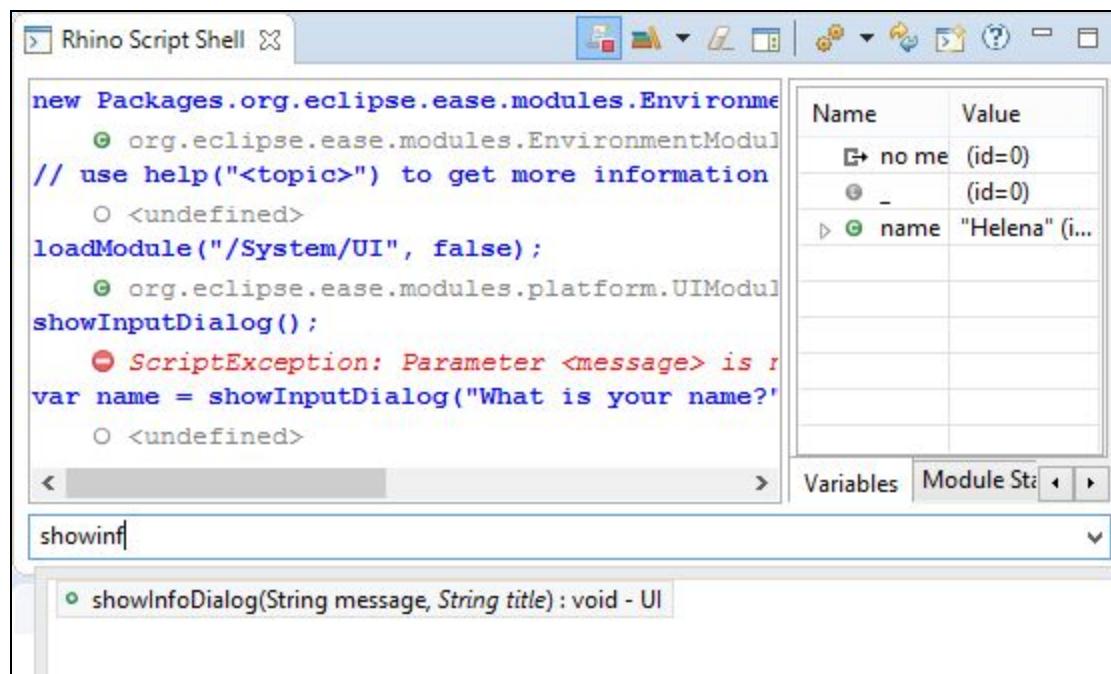
7. The value of the variable **Name** will now be displayed with the new value, in the **Variables** pane. I entered **Helena** as the value for the variable **name**.



8. From the Modules Explorer, drag **System > UI > showInfoDialog()** and drop it into the Script Shell. Again, ignore the script exception message.



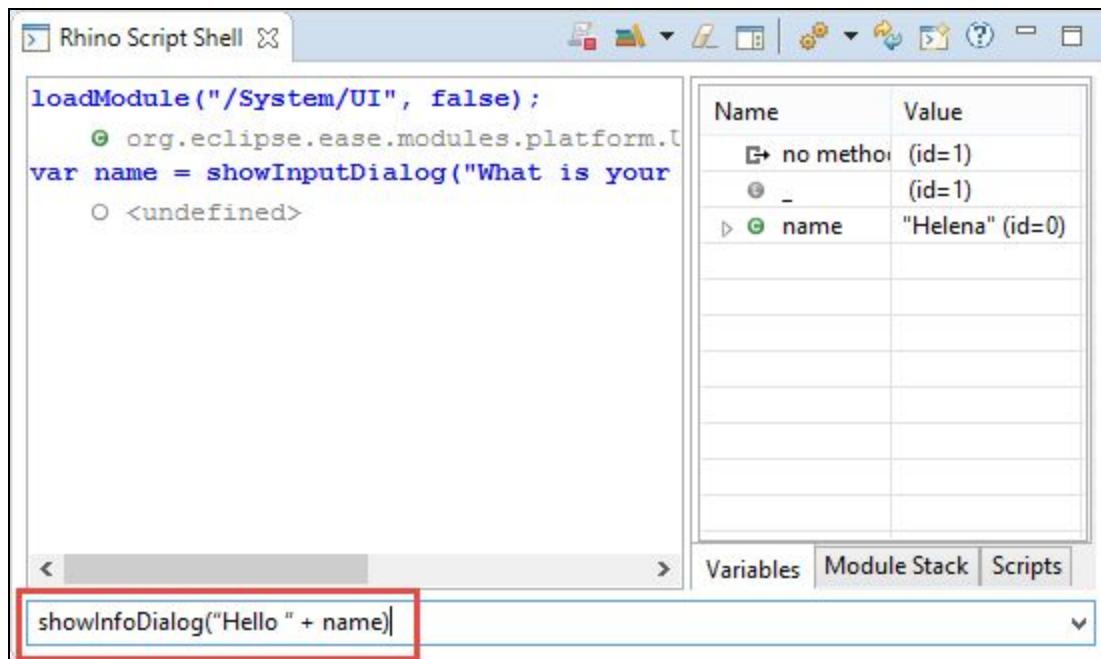
9. Now let us run another command. Type **showInf**, and press **Ctrl+space** and then double-click **showInfoDialog** to select it.



The screenshot shows the Rhino Script Shell interface. The left pane contains a script with several errors highlighted in red. The right pane shows a variable table with three entries: 'no me' (id=0), '\_' (id=0), and 'name' set to "Helena". Below the table are tabs for 'Variables' and 'Module St'. At the bottom of the shell window, there is a search bar containing 'showinf' and a dropdown menu showing the selected 'showInfoDialog(String message, String title) : void - UI' command.

10. Now populate the command with the values **"Hello " + name**





11. Press **Enter**.



The information dialog will display.



12. Click **OK** to close the dialog. Now let us look at the Script Shell. You will notice that you have written the following steps :

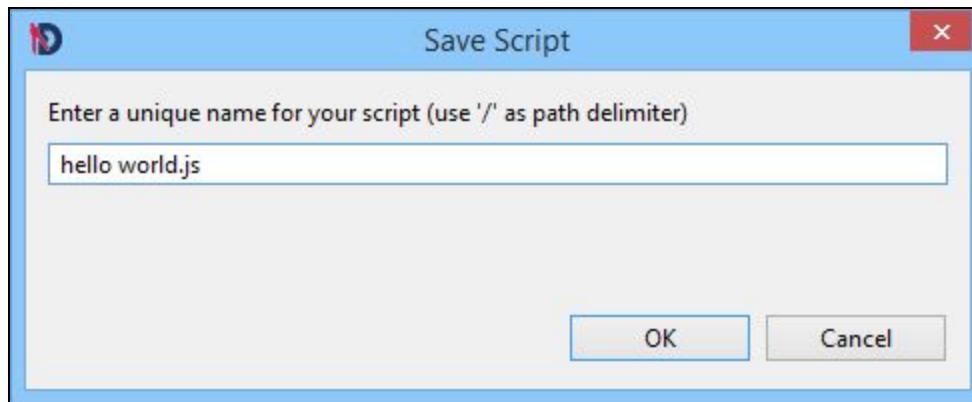


```
loadModule("/System/UI", false);
var name = showInputDialog("What is your name","name here",
"First steps");
showInfoDialog("Hello " + name);
```

12. Now click the **Toggle Script Recording** button  again to stop recording the script.



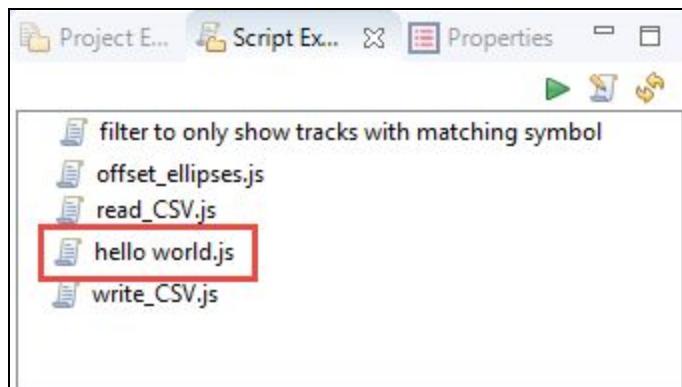
The **Save Script** dialog will display and prompt you to enter a name for the script.



13. Enter a name for the script and click **OK**. I have given the name **hello world.js**



Your script will be saved and displayed in the **Script Explorer**.



## 5.6 Using Debrief Objects in Scripting

Debrief via its **Scripting Perspective** feature allows users to apply bulk changes to Debrief objects. The Modules Explorer allows you to access deeper levels of the **Debrief** object model and the Script Shell allows you to interact live with the objects.

The **Scripting Perspective** allows analysts to work on the objects in Debrief's Outline view. From the **Outline** view, analysts will be able to retrieve objects and then manipulate them. The API also allows

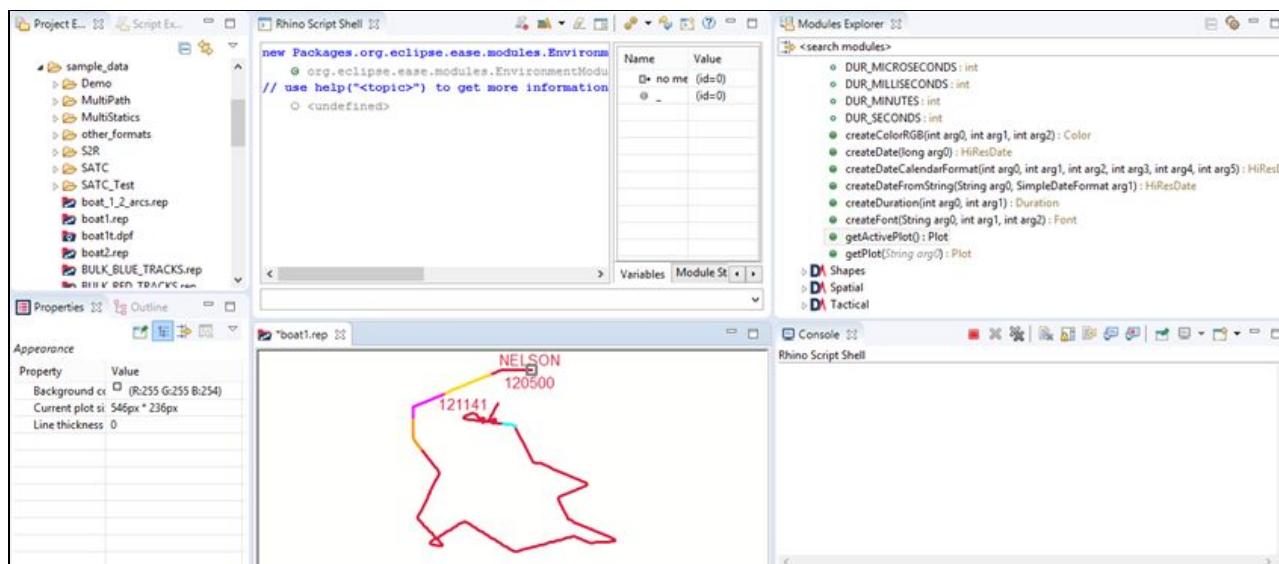
them to change the map view or the current time. To start with, you must have a **Debrief** plot. From the plot object, you can then drill down into the assorted child elements. Next in this tutorial we will learn how to access the **Debrief** objects.

In this example, we will access the **Debrief** object plot of the file **boat1.rep**, find the layer Nelson in the plot, define a new color and change the color of the track Nelson in the plot. We need to load data into **Debrief** first. Let's start by doing it.

1. Keeping the **Scripting Perspective** open, double-click on the file **boat1.rep** in the **Navigator** view.

2. Select **Over the Ground** option in the **Select track mode** dialog.

3. **boat1.rep** will display along with the **Scripting Perspective**.



4. To begin with, we have to load the **Core** module into the Script Shell and for this, enter the below command in the **Script Shell**:

```
loadModule('/Debrief/Core');
```

**Note:** Alternatively, you can drag **Debrief > Core**, from the **Modules Explorer**, and drop it into the **Script Shell**.

5. Now we have to get the current editor, so enter the below command in the Script Shell:

```
let editor = getActivePlot();
```

6. To get code completion, we need to declare the type of the data object. We do this by providing a specially formatted comment. Put the below comment, so as to declare the

type of the layers object that we're about to create:

```
// @type org.mwc.debrief.scripting.wrappers.DLayers
```

7. Next task is to get the layers object. To do so, enter the below line to the Script Shell:

```
let layers = editor.getLayers();
```

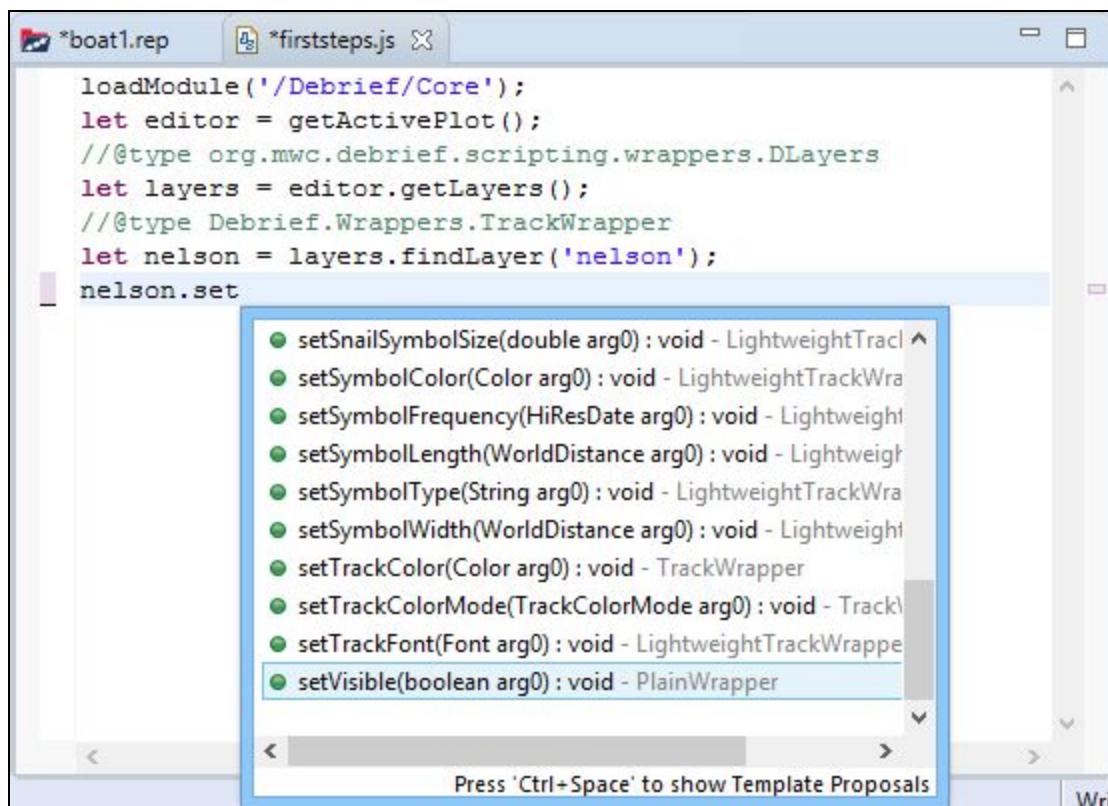
8. Add a type comment to get the proposals under Nelson track like this:

```
//@type Debrief.Wrappers.TrackWrapper
```

9. Next we have to get the track:

```
let nelson = layers.findLayer('Nelson');
```

11. Next step is to hide the track Nelson. Enter `nelson.set` and press **Ctrl+space** to get the auto-complete options.



12. Complete the code by adding `nelson.setVisible(false);` to the script.

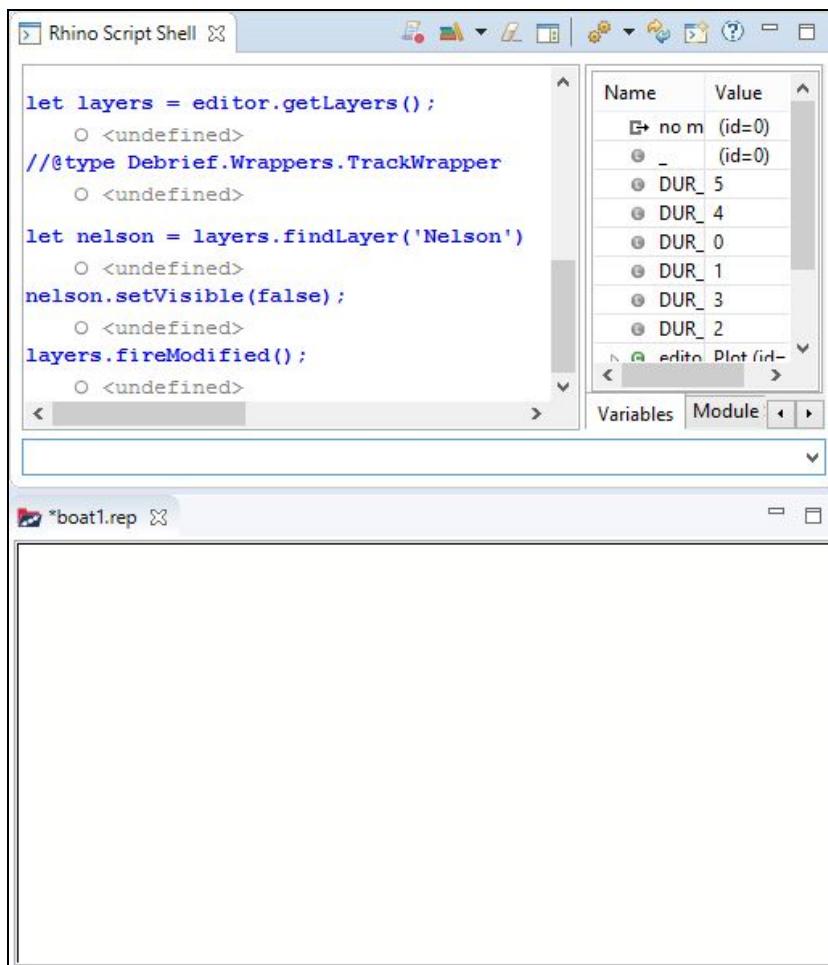
13. We will now update the plot by entering the below command in the Script Shell:

```
layers.fireModified();
```

**Note:** We have opted to run the commands in the **Script Shell**, instead of running the script on the selected file, so that you can see the changes in the **Plot Editor** as you run the commands. If you

create a Javascript file, add the code and then run the script, you will be able to view only the final changes in the **Plot Editor**.

14. You will notice that the **Nelson** track, of **boat1.rep**, will disappear in the **Plot Editor**.



15. We have to make the track Nelson visible again. Enter the below line in the Script Shell:

```
nelson.setVisible(true);
```

16. Update the plot again to reflect the change by entering the below command in the Script Shell:

```
layers.fireModified();
```

17. The track Nelson will become visible in the **Plot Editor**.

18. We will now create a new color for the track by entering the command: `let purple = createColorRGB(100, 10, 215);` in the Script Shell:

19. Now we will update the color of the track Nelson to newly created color. Enter the command `nelson.setColor(purple);` in the Script Shell.

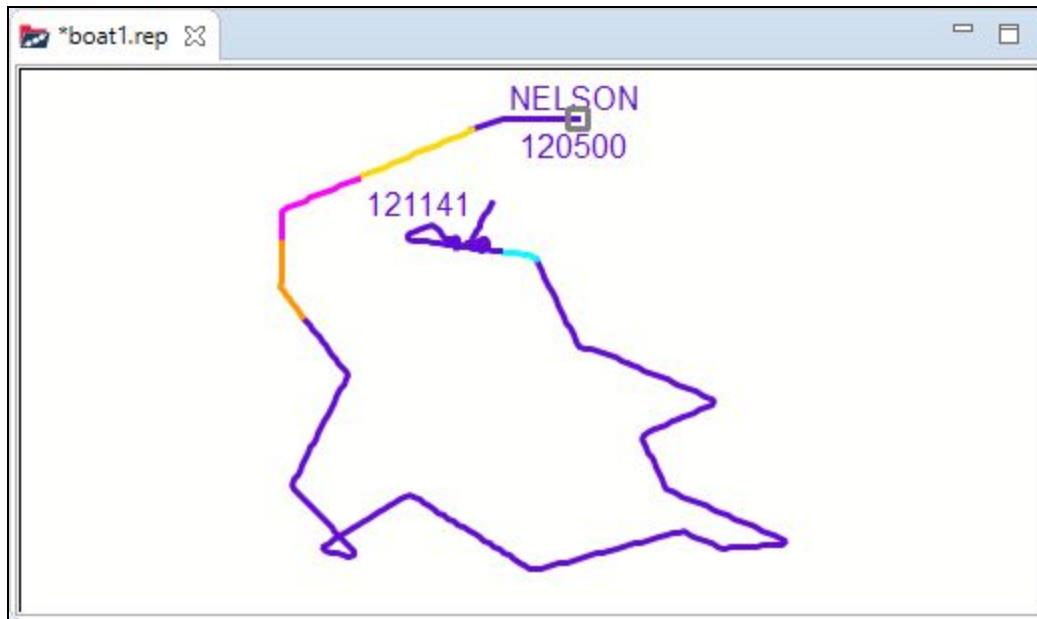


20. Update the plot again to reflect the color change by entering the below command in the script shell

```
layers.fireModified();
```



21. The color of the track Nelson in **boat1.rep** will be updated to the new color.



## 5.7 Creating Scripts

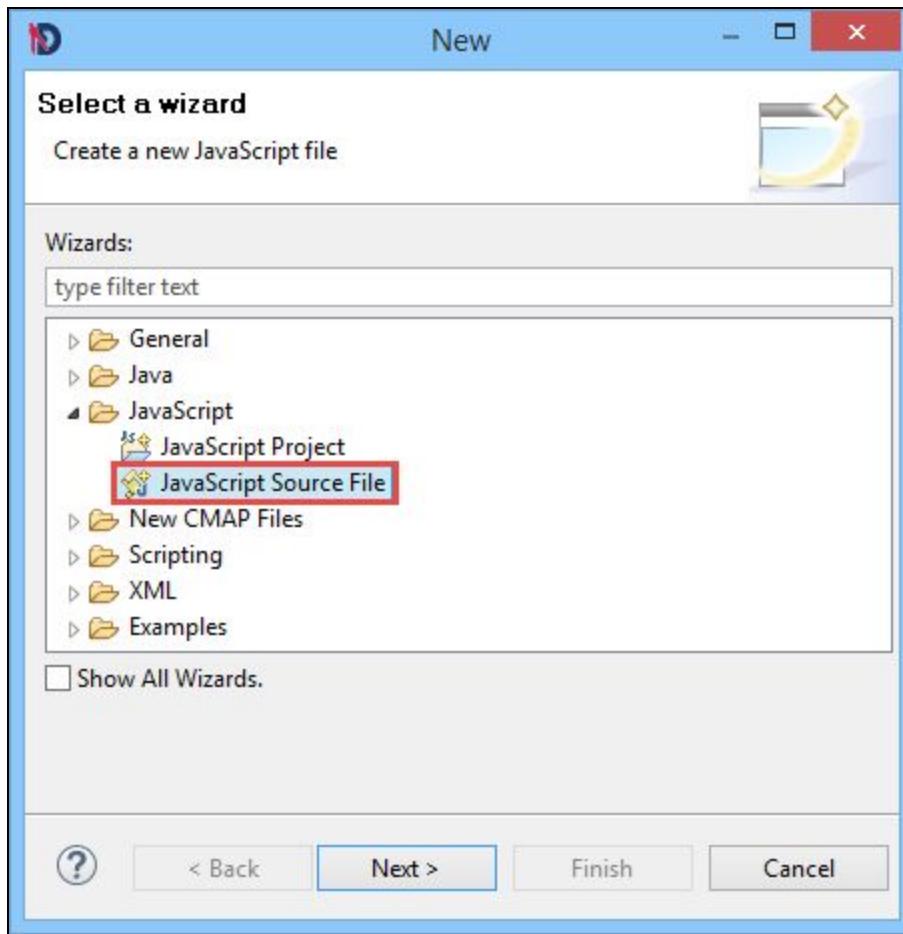
Debrief's **Scripting Perspective** feature allows you to create scripts in a file and run them whenever required. You can run a Javascript on .REP files and .DPF files. In this section, we will learn how to create a script in a file and how to run the script on a .REP file. We will also learn how to configure the script to create a command button and how to create a keyboard shortcut to run the script.

- [Creating Script in File](#)
- [Configuring Script to Create Command Button](#)
- [Creating Keyboard Shortcut for Script](#)

### 5.7.1 Creating Script in File

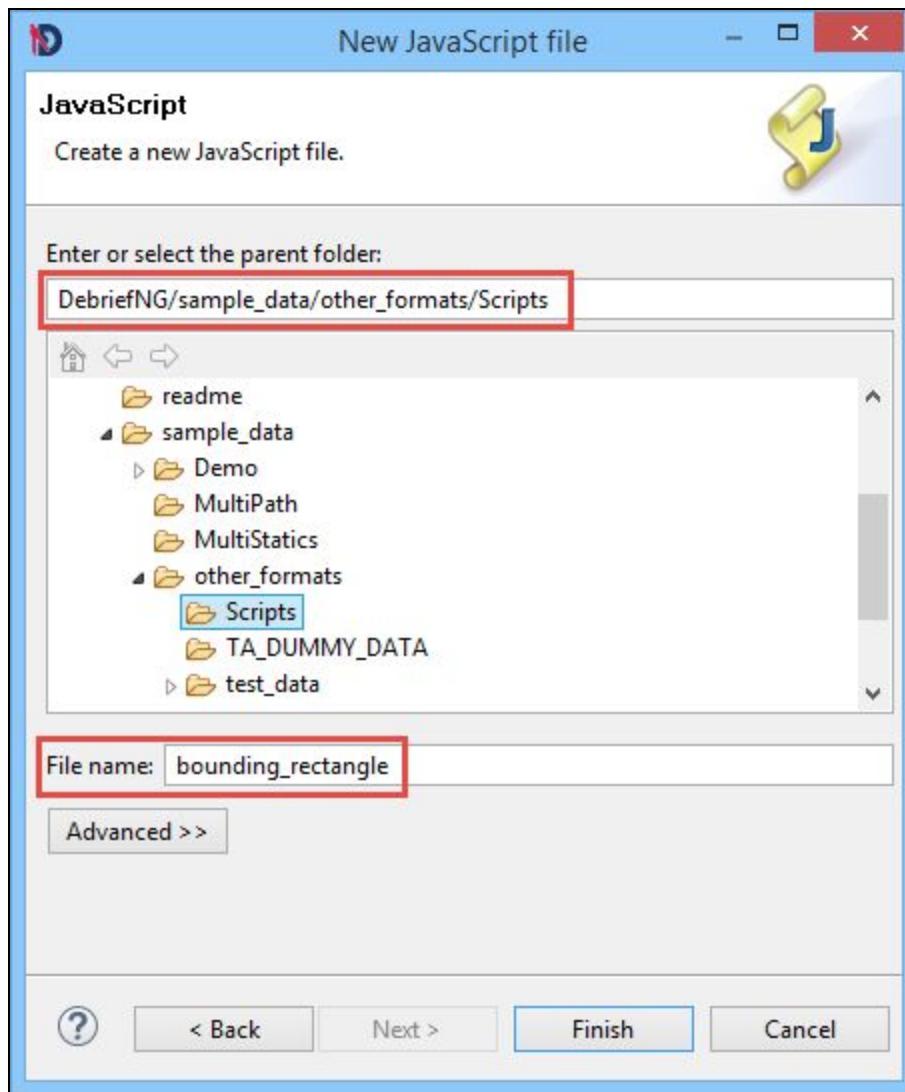
We shall now learn to create a script in a file. This script will draw a bounding rectangle within its own layer, sized to display around a plot track. We will run this script on the boat1.rep file.

1. We have to set the location where you want to store your scripts. If you have not done this yet, refer to [Setting Path for Scripts Folder](#).
2. After setting the path to the scripts folder, we have to create a new javascript file called **bounding\_rectangle.js**. To create a new file, navigate to **File > New > Other**.
3. The **New** dialog will display.

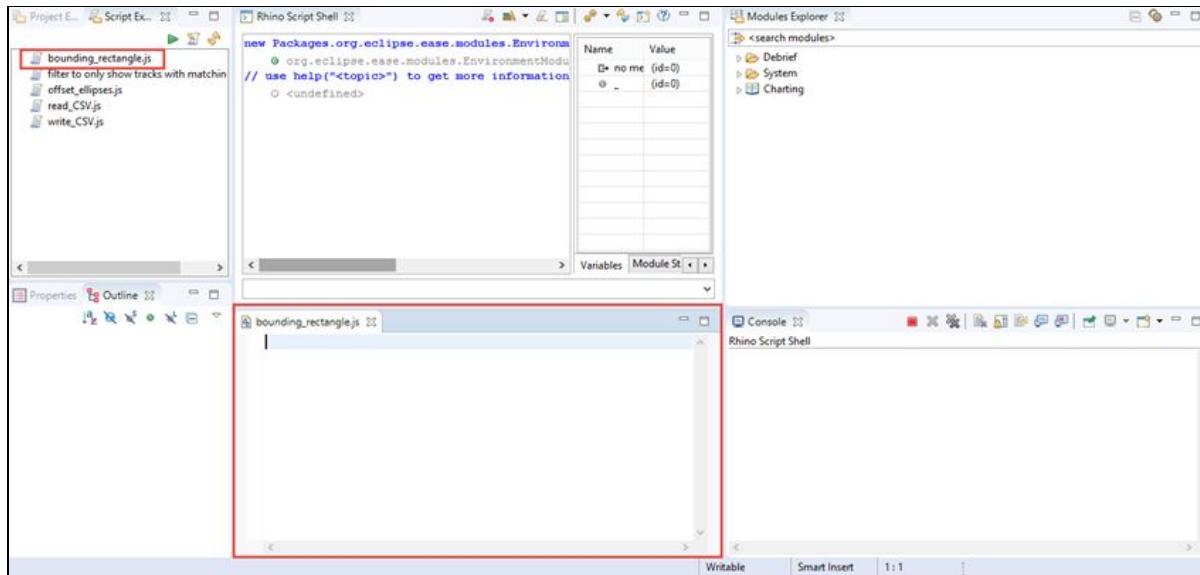


4. Click **JavaScript** and select **JavaScript Source File**. Now click **Next**.

The **New File** dialog will display.



5. Enter or select the path of the folder where you want to store your script in the **Enter or select the parent folder** field. I have selected the **sample\_data/other\_formats/Scripts** folder of my **Debrief** installation.
6. Enter the file name **bounding\_rectangle** in the **File name** field and click **Finish**.
7. The javascript file **bouding\_rectangle.js** will be created and will be displayed in the **Script Explorer**. The blank file **bouding\_rectangle.js** will open in the script editor.



- Let us start writing lines of codes in the script file that we just created. We have to load some modules so that we can access methods within those modules. We will need the modules **Annotations** (to get `createRectangleShape` method), **Core** (to get the methods `getActivePlot`, to `createColorRGB`) and **Shapes** (to get label location constants) for this tutorial. To load the modules, include the below code in the script:

```
loadModule('/Debrief/Shapes'); //to get createRectangleShape  
method  
  
loadModule('/Debrief/Core'); // to get active plot, to create  
color  
  
loadModule('/Debrief/Annotations');//to get label location  
constants
```

9. We have to next get the editor. Add the below line to the script:

```
let editor = getActivePlot();
```

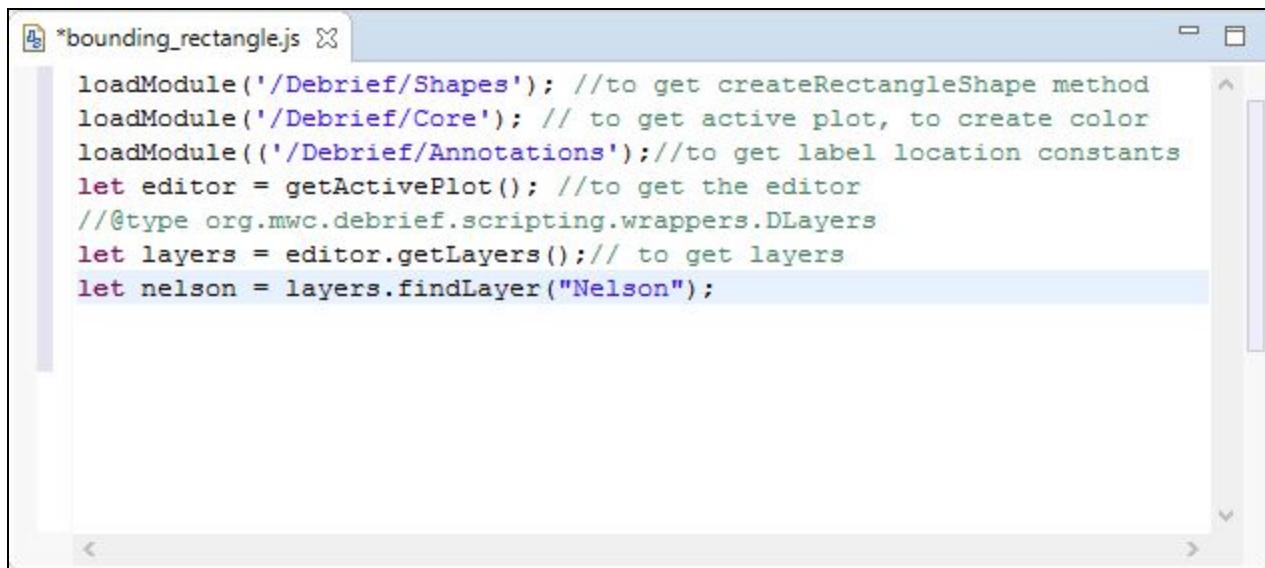
10. Next to get layers, add the following lines to the script :

```
//@type org.mwc.debrief.scripting.wrappers.DLayers  
let layers = editor.getLayers();
```

11. Next thing is to get the track, to run the script on. For this add the below line to the script:

```
let nelson = layers.findLayer("Nelson");
```

12. After this step, your script editor should look like the below screenshot:



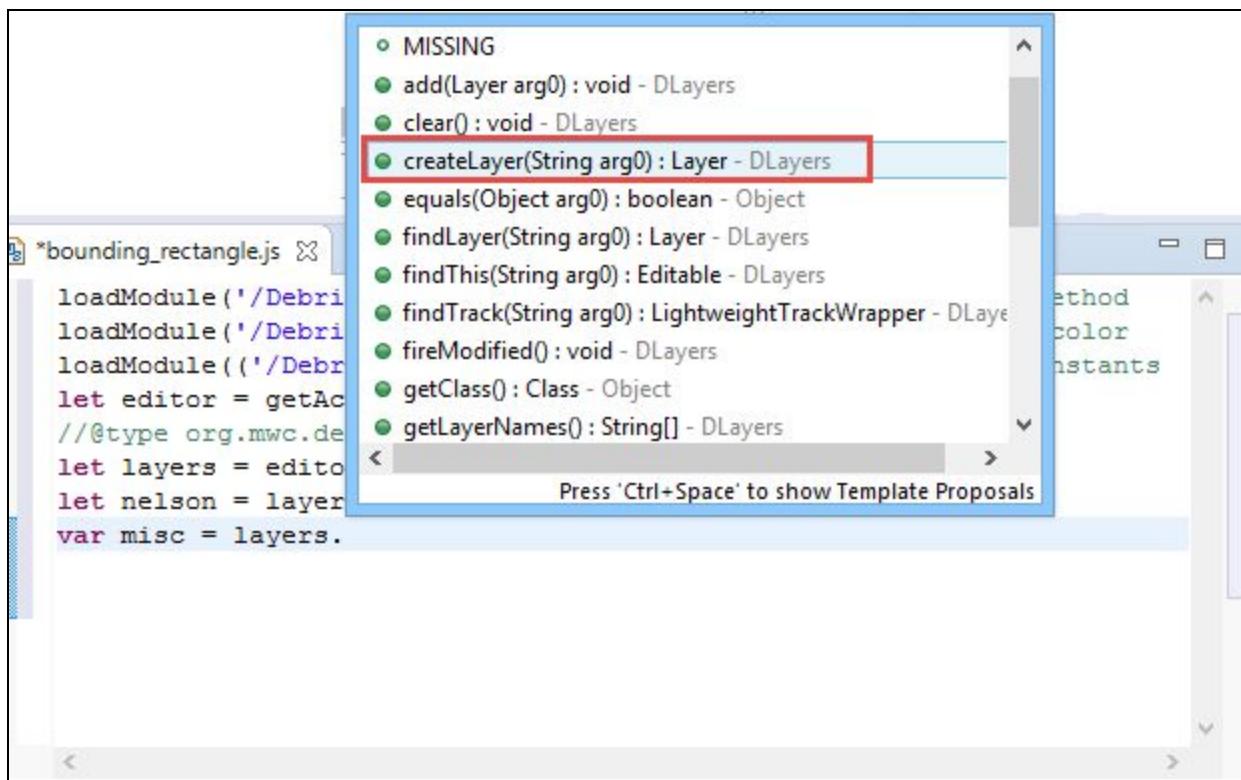
```

*bounding_rectangle.js ✎

loadModule('/Debrief/Shapes'); //to get createRectangleShape method
loadModule('/Debrief/Core'); // to get active plot, to create color
loadModule('/Debrief/Annotations');//to get label location constants
let editor = getActivePlot(); //to get the editor
//@type org.mwc.debrief.scripting.wrappers.DLayers
let layers = editor.getLayers();// to get layers
let nelson = layers.findLayer("Nelson");

```

14. Let's continue with populating the script file. Now to create a new layer add the line **var misc = layers.** and press **Ctrl + space** to get code-completion. From the pop-up menu select the **createLayer** method from the proposals list and give a meaningful name to your new layer **var misc = layers.createLayer("Misc");**



The screenshot shows a code editor window with a script file named "bounding\_rectangle.js". The code has been partially typed as follows:

```

*bounding_rectangle.js ✎

loadModule('/Debrief/Shapes');
loadModule('/Debrief/Core');
loadModule('/Debrief/Annotations');
let editor = getActivePlot();
//@type org.mwc.debrief.scripting.wrappers.DLayers
let layers = editor.getLayers();
let nelson = layers.findLayer("Nelson");
var misc = layers.

```

A code completion dropdown menu is displayed over the line where "misc" is defined. The menu lists several methods available for the "layers" object, including "MISSING", "add", "clear", and "createLayer". The "createLayer" method is highlighted with a red border. Below the list, a tooltip says "Press 'Ctrl+Space' to show Template Proposals".

15. The next step is to retrieve the bounds of the nelson track. For this add the below line to the script:

```
let bounds = nelson.getBounds();
```

16. Let's create a new color for the rectangle. To do this add the below line to the script:

```
let rectColor = createColorRGB(100, 100, 215);
```

17. We have to next create a rectangle that will bound the track. To create a rectangle shape, we will need 2 world Locations - Top-Left and Bottom-Right. Firstly, let's create the top-left and bottom-right locations by adding the following:

```
let topLeft = bounds.getTopLeft();  
let botRight = bounds.getBottomRight();
```

18. We have to now create the rectangle by using createRectangleShape method. Add the below line to the script:

```
var rect = createRectangleShape( topLeft, botRight, "First  
Rectangle", rectColor);
```

19. Next step is to convert extract the bounding coordinates into a string of text. Let's add the below line to the script:

```
let boundString = bounds.toString();
```

20. Let's set the rectangle title to bounds string by adding the following command to the script:

```
rect.setLabel(boundString);
```

21. To set the title of the rectangle to be displayed at bottom of the track.

```
rect.setLabelLocation(LABEL_LOCATION_BOTTOM);
```

22. Finally to add the rectangle to the above created layer, include the below line to the script:

```
misc.add(rect);
```

23. Update the map by triggering the method fireModified on the layer by adding the below line to the script:

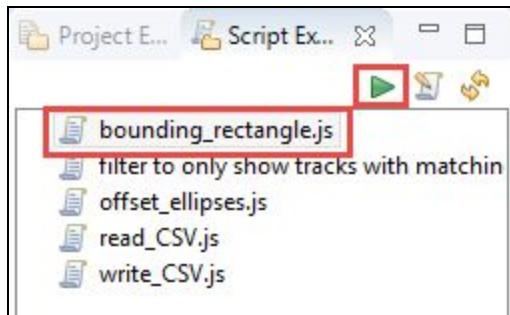
```
layers.fireModified();
```

24. Click the save icon  on the main toolbar to save the script.

25. We have now completed the script. Let's run the script on a file to view the output. From the **Navigator** view / **Project Explorer** open the file **boat1.rep**, in **Over the Ground** mode.

26. We will run the script on the file **boat1.rep**. To run the script you can either

- Select the script **bounding\_rectangle.js** in the Script Explorer and click run or

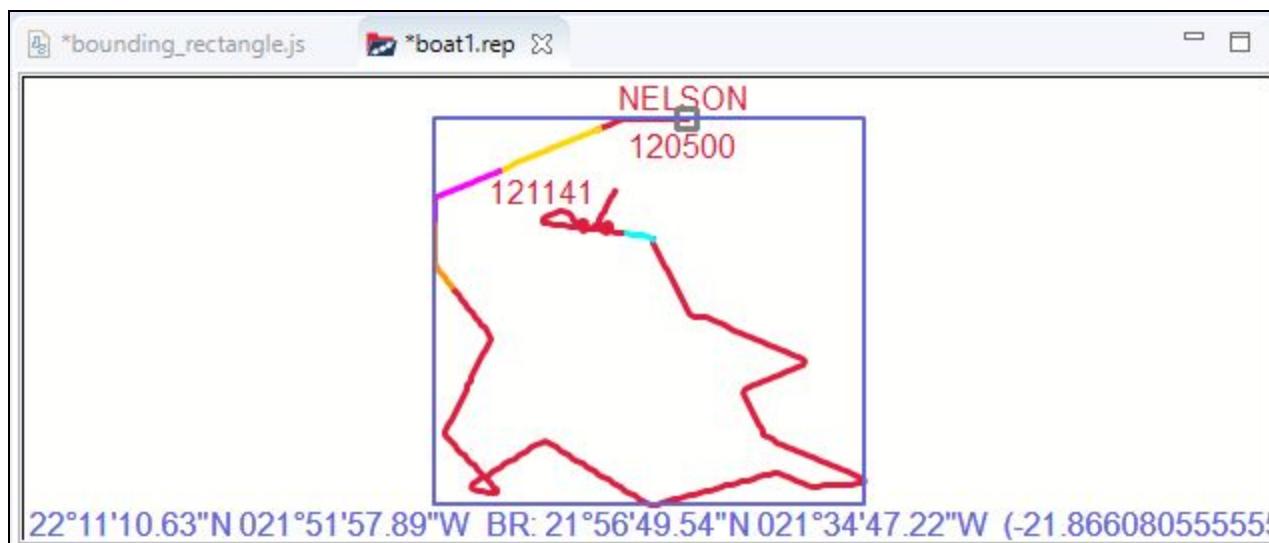


- Drag the script from Script Explorer and drop it to the Script Shell.

**Note:** When you run the script, the **Plot Editor** should always be the active window.



33. A bounding rectangle will display, around the track, in the **boat1.rep** file.



### 5.7.2 Configuring Script to Create Command Button

We have now created the script in a file and run it successfully to obtain the desired result. Debrief's Scripting Perspective enables users to integrate scripts into the Debrief user interface, for example to create a command button that will appear on the Script Explorer toolbar. Let's learn how to do that:

1. Add the below comment lines at the beginning of script `bounding_rectangle.js`:

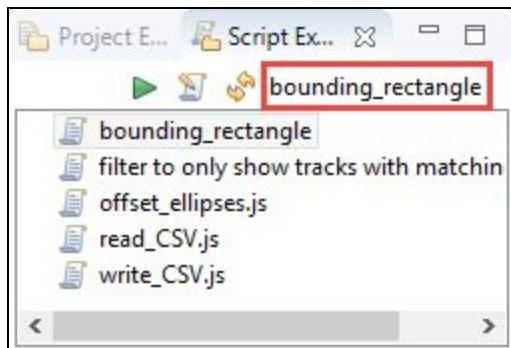


```
////////////////////////////////////////////////////////////////////////  
// name : bounding_rectangle
```

```
// toolbar : Script Explorer
// script-type : JavaScript
// description : Create rectangle and add label.
// ****
```

**Note:** The parameter **name** controls how the script is displayed in the **Script Explorer**. Without this block of script also the filename that you have specified (while creating the file) will be displayed. However when the **name** parameter is provided it will switch from the filename to the value of the **name** parameter.

2. Now save the script by clicking save icon  on the Main Toolbar.
3. You will notice, on the toolbar for the **Script Explorer** view, a new button **bounding\_rectangle** will display.



**Note:** If you cannot see the button after step 2, close **Script Explorer** view and open it again.

### 5.7.3 Creating Keyboard Shortcut for Script

**Debrief Scripting Perspective** allows you to create keyboard shortcut to run scripts. Let's create a keyboard shortcut **Ctrl+5** to run the script **bounding\_recatngle.js**.

1. Add the below comment lines at the beginning of script **bounding\_rectangle.js**:

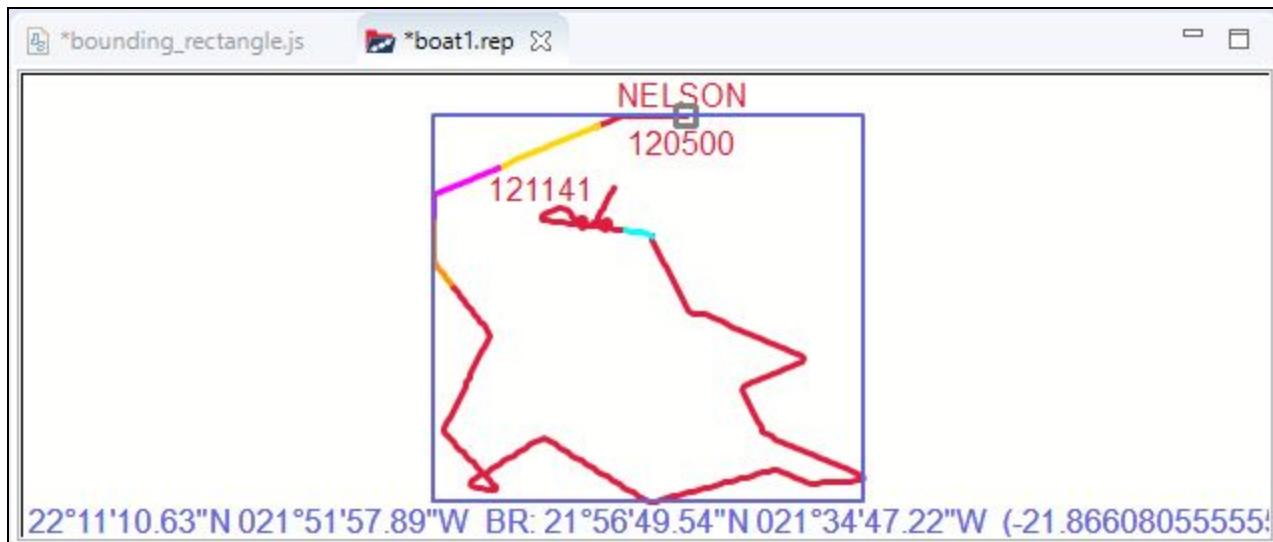
```
*****  
// name : bounding_rectangle  
// keyboard : Ctrl+5  
// script-type : JavaScript  
// description : Create Rectangle shape under new layer  
//  
*****
```

2. Now save the script by clicking save icon  on the **Main Toolbar**.

3. Close the file **boat1.rep** without saving the changes.

4. Open the file **boat1.rep** again and press **Ctrl+5** to run the script.

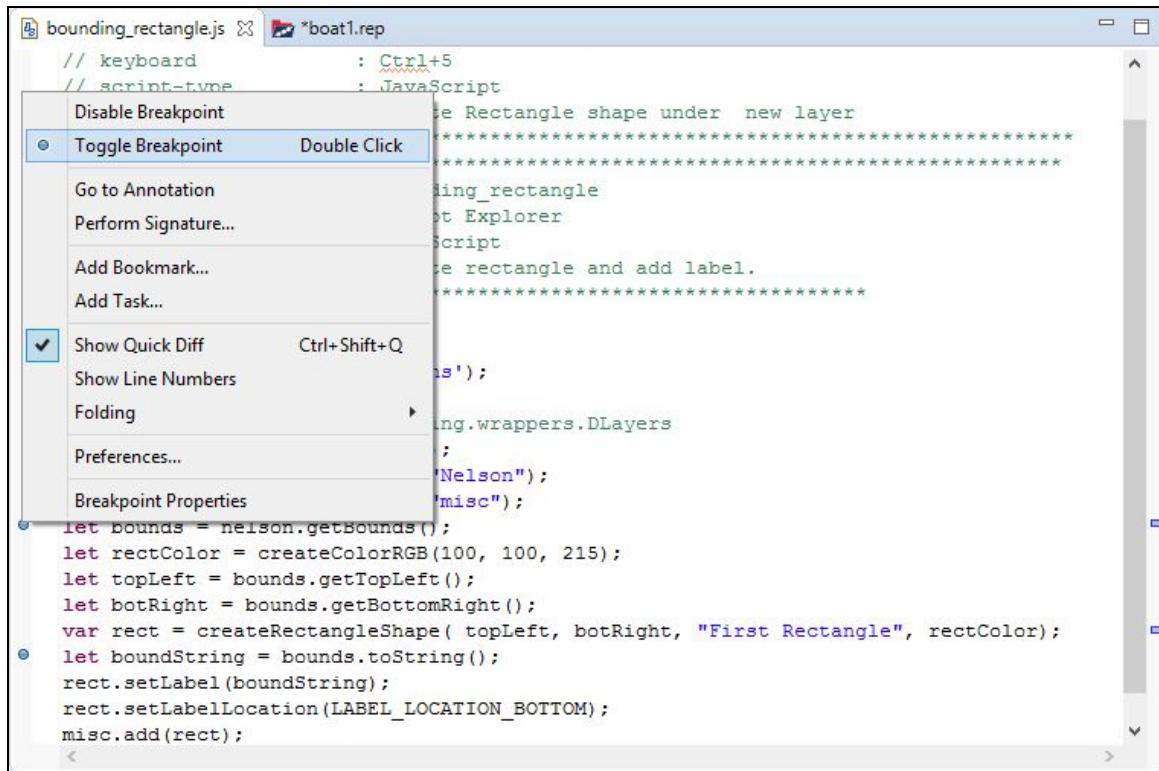
5. A bounding rectangle will display, around the track, in the **boat1.rep** file.



## 5.8 Debugging in the Debug Perspective

We have reached the final section of this tutorial. In this section of the tutorial you will learn debugging in the Scripting Perspective.

1. In the script editor, right-click on the left pane next to line 8 of the script. The pop-up menu will display. Select **Toggle Breakpoint**.



```

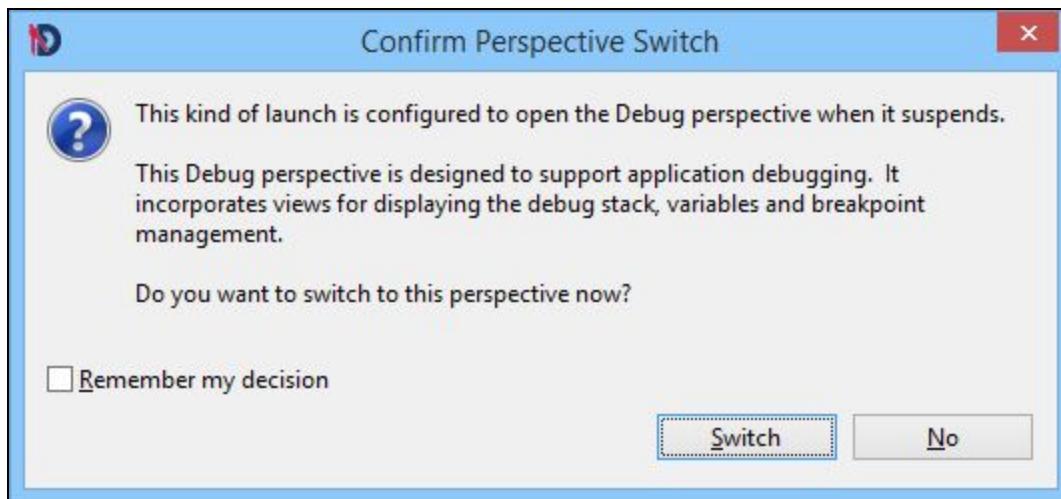
// keyboard : Ctrl+5
// script-type : JavaScript
Disable Breakpoint
Toggle Breakpoint Double Click
***** Rectangle shape under new layer *****
Go to Annotation
Perform Signature...
Add Bookmark...
Add Task...
Show Quick Diff Ctrl+Shift+Q
Show Line Numbers
Folding
Preferences...
Breakpoint Properties
let bounds = nelson.getBounds();
let rectColor = createColorRGB(100, 100, 215);
let topLeft = bounds.getTopLeft();
let botRight = bounds.getBottomRight();
var rect = createRectangleShape(topLeft, botRight, "First Rectangle", rectColor);
let boundString = bounds.toString();
rect.setLabel(boundString);
rect.setLabelLocation(LABEL_LOCATION_BOTTOM);
misc.add(rect);
<

```

2. Next right-click anywhere in the script editor and select **Debug As** and then click **1EASE Script**.



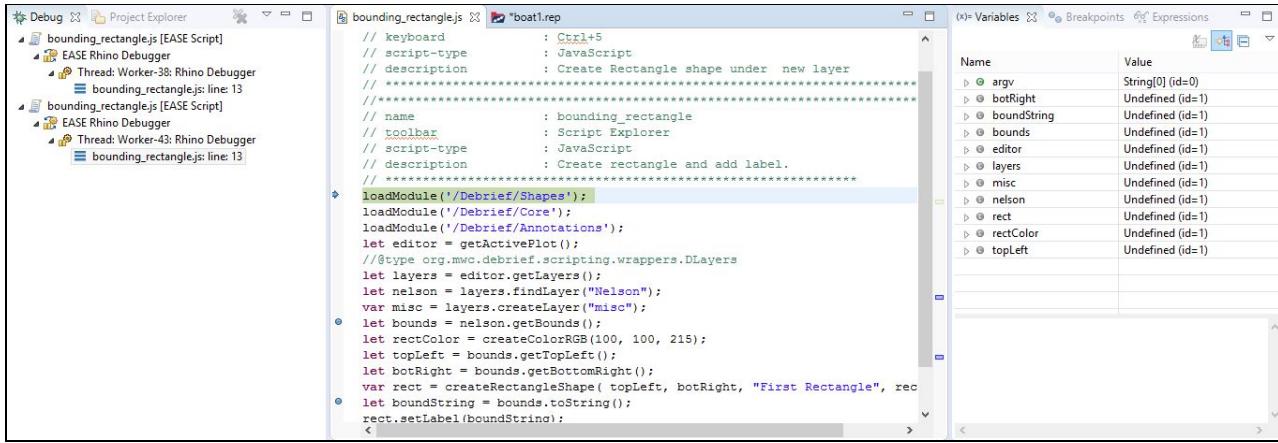
3. The **Confirm Perspective Switch** dialog will display.



4. The Debug perspective supports debugging along with displaying the debug stack, variables and breakpoint management. Click **Switch**.

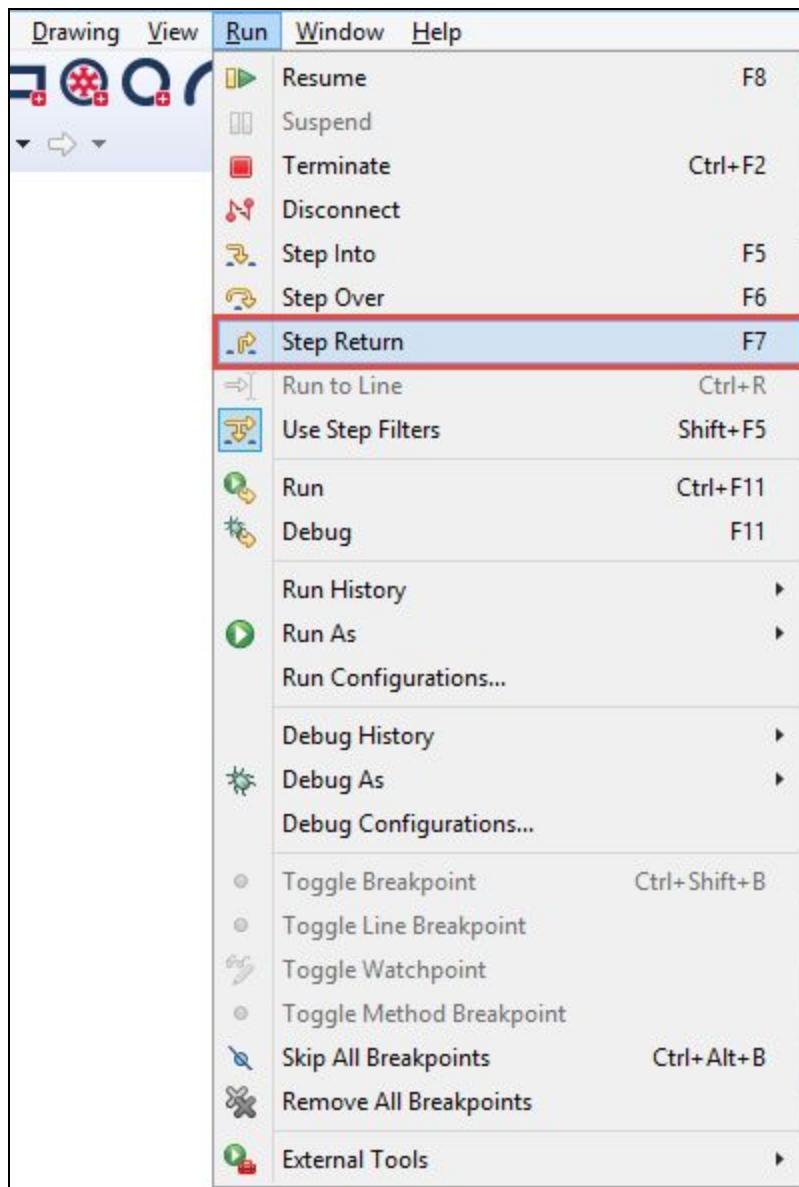


5. The **Debug Perspective** will display.



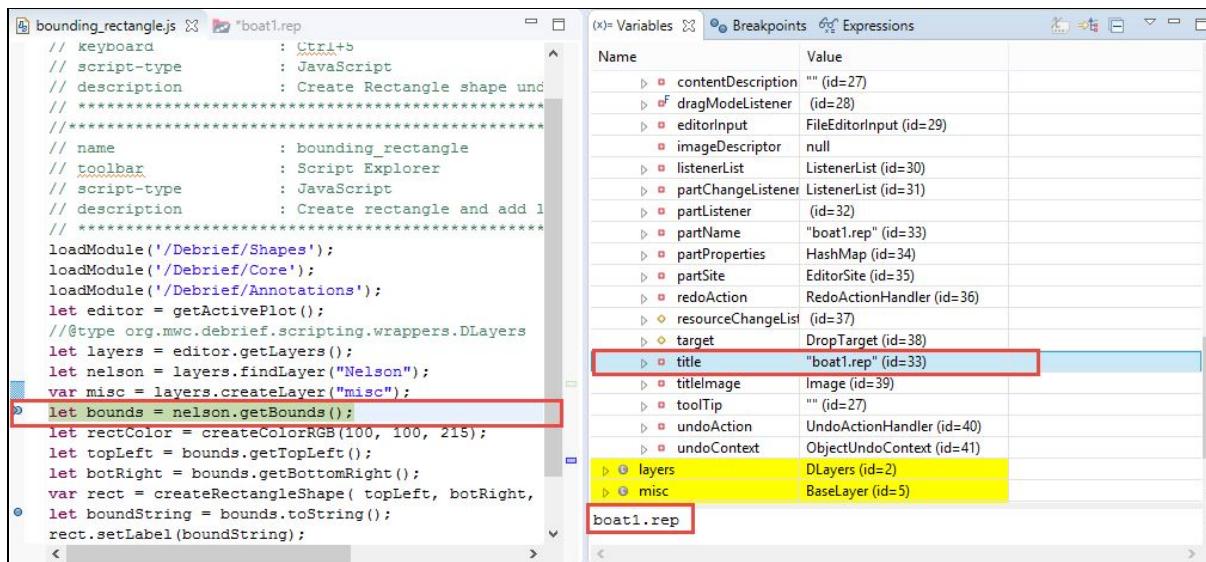
The screenshot shows the Debrief interface in the Debug perspective. The Project Explorer pane on the left lists two EASE Script files: 'bounding\_rectangle.js' and 'boat1.rep'. The Script Editor pane in the center displays the code for 'bounding\_rectangle.js'. The code creates a rectangle shape under a new layer, loads modules for Shapes, Core, and Annotations, gets an active plot, and creates a rectangle with specific properties. The Variables pane on the right shows various variables and their values, such as argv, botRight, boundString, bounds, editor, layers, misc, nelson, rect, rectColor, and topLeft. The Breakpoints and Expressions panes are also visible at the top of the interface.

6. Debrief lets you to debug a script line by line by using the **Step Over (F6)** option and also debug the script at the designated break points by using the **Step Return (F7)** option. For this tutorial we will use the **Step Return** option. Now click **Run** on the **Main Menu** and then select **Step Return** or press **F7** which is the keyboard shortcut for **Step Return** option.



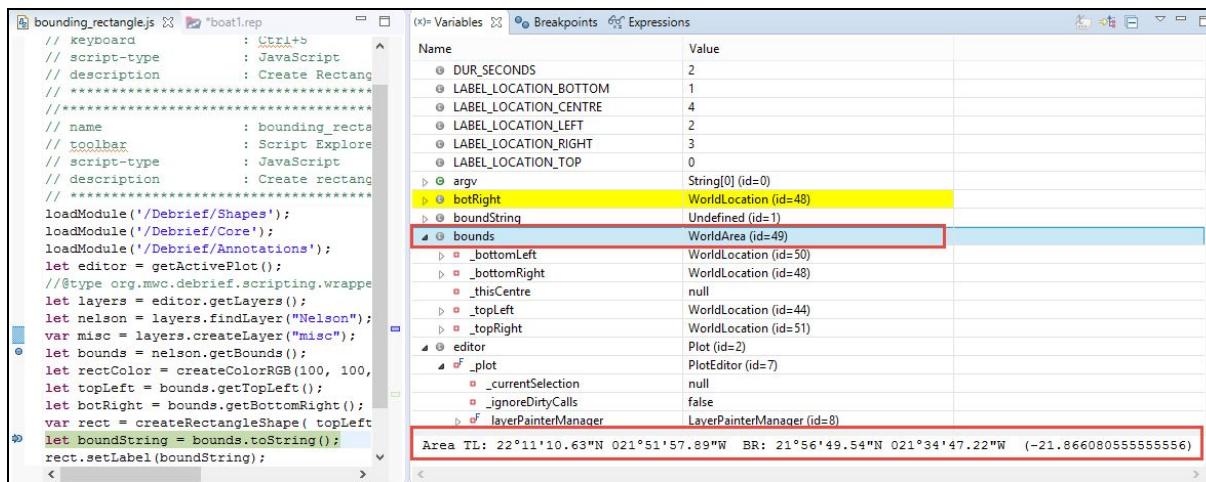
7. The debugger jumps to line 8 In the **Variables** view, click on the arrow next to **editor** variable and then click on **title** to get the value of the variable **title**. The value of title is **boat1.rep** and it will display at the bottom of the **Variables** view.





Name	Value
contentDescription	"" (id=27)
dragModeListener	(id=28)
editorInput	FileEditorInput (id=29)
imageDescriptor	null
listenerList	ListenerList (id=30)
partChangeListener	ListenerList (id=31)
partListener	(id=32)
partName	"boat1.rep" (id=33)
partProperties	HashMap (id=34)
partSite	EditorSite (id=35)
redoAction	RedoActionHandler (id=36)
resourceChangeList	(id=37)
target	DropTarget (id=38)
<b>title</b>	<b>"boat1.rep" (id=33)</b>
titleImage	Image (id=39)
toolTip	"" (id=27)
undoAction	UndoActionHandler (id=40)
undoContext	ObjectUndoContext (id=41)
layers	DLayers (id=2)
misc	BaseLayer (id=5)

8. Press F7 again, now the debugger jumps to line 21. Now let's see the value of the variable **bounds**. In the **Variables** view, double-click on the variable bounds. It's value will display at the bottom of the **Variable** view.

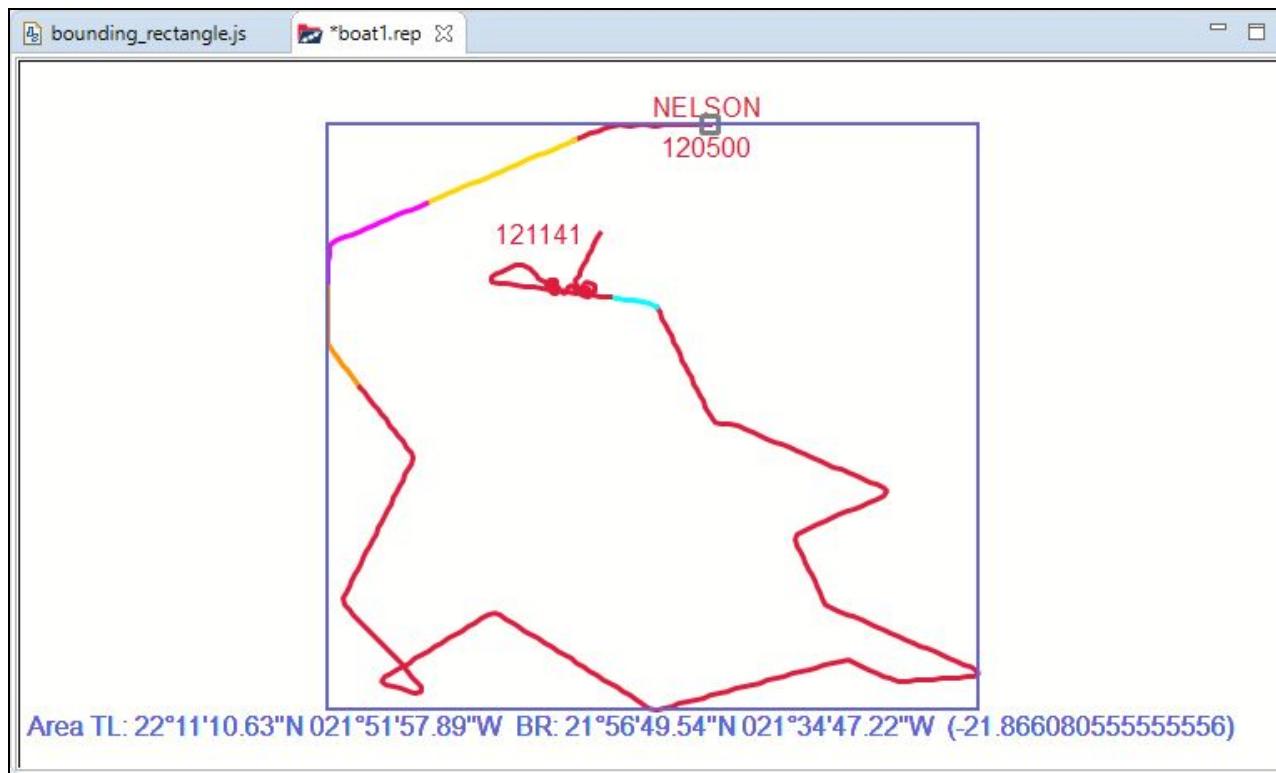


Name	Value
DUR_SECONDS	2
LABEL_LOCATION_BOTTOM	1
LABEL_LOCATION CENTRE	4
LABEL LOCATION LEFT	2
LABEL LOCATION RIGHT	3
LABEL LOCATION TOP	0
argv	String[0] (id=0)
<b>botRight</b>	WorldLocation (id=48)
<b>boundString</b>	Undefined (id=1)
<b>bounds</b>	WorldArea (id=49)
_bottomLeft	WorldLocation (id=50)
_bottomRight	WorldLocation (id=48)
_thisCentre	null
_topLeft	WorldLocation (id=44)
_topRight	WorldLocation (id=51)
editor	Plot (id=2)
_plot	PlotEditor (id=7)
_currentSelection	null
_ignoreDirtyCalls	false
layerPainterManager	LayerPainterManager (id=8)

Area TL: 22°11'10.63"N 021°51'57.89"W BR: 21°56'49.54"N 021°34'47.22"W (-21.866080555555556)

9. Now press F7 to complete the execution of the script. Now switch to the Plot editor to view the output on the boat1.rep.

10. The bounding rectangle will display and the value of the variable **bounds** is its label.



## 5.9 Conclusion

This concludes the tutorial on **Debrief Scripting Perspective**. By now you should be comfortable with the Scripting Perspective, record a script, access **Debrief** objects, create script in a file, configure scripts to create toolbar buttons, create keyboard shortcuts to run scripts, and debug scripts through the debug perspective. If your role includes Single Sided Reconstruction, we suggest you move onto next tutorial.

Signed: \_\_\_\_\_ Date: \_\_\_\_\_