

TimeStudio process manual – Dynamic AOI analysis

Author: Pär Nyström

Last revision: 2013-05-13

Innehåll

| | |
|--|---|
| TimeStudio process manual – Dynamic AOI analysis..... | 1 |
| 1. Purpose..... | 2 |
| 2. Definitions..... | 2 |
| 3. Data set..... | 2 |
| 4. Overview..... | 3 |
| 5. Step-by-step-instructions..... | 5 |
| 5.1 Launch MATLAB and TimeStudio..... | 5 |
| 5.2 Add data and create subjects..... | 5 |
| 5.3 Read data files into memory..... | 7 |
| 5.4 Changing the names of events in the dataset..... | 11 |
| 5.5 Compensating for bad calibration of gaze position..... | 13 |
| 5.6 Plotting the gaze trajectories to see raw data..... | 14 |
| 5.7 Cleaning up before analysis..... | 17 |
| 5.8 Defining Areas of Interest for the event “learn”..... | 17 |
| 5.9 Getting the latency results..... | 22 |
| 6. Final comments and remarks..... | 25 |
| 7. References..... | Fel! Bokmärket är inte definierat. |
| 8. Acknowledgements..... | 25 |

1. Purpose

This manual contains step-by-step instructions of how to analyze looking time data from dynamic areas of interest (AOIs) in the TimeStudio framework. The manual is written for novel users of TimeStudio.

This manual assumes that you have downloaded the latest TimeStudio core and the data files mentioned below. If you do not know how to start TimeStudio, please double check that you have followed the steps described in <http://timestudioproject.com/getting-started>

2. Definitions

Buttons that need to be activated in order to execute a command or fields within the workspace are indicated with quotation marks in the text below. Plugin names are marked with bold. Letters in red circles in figures and text indicate where in TimeStudio a particular action takes place.

3. Data set

This manual covers how to set up a workflow to analyze static aois in eye tracking studies. In the manual some demo datasets are used to illustrate how and why each step is necessary.

The files needed to follow this manual are:

dynamic_aoi_subj1.txt
dynamic_aoi_subj2.txt
dynamic_aoi_subj3.txt
dynamic_aoi_subj4.txt
dynamic_aoi_subj5.txt
dynamic_aoi_subj6.txt

Each file is a recording of one test subject and the files can be downloaded from

http://www.timestudioproject.com/manuals/dynamic_aoi_manual_data_files.zip

The data is taken from a large eye tracking study in which several different stimuli was presented to young infants. In this tutorial we will describe how to analyze gaze latencies when viewing a movie clips where an object several times disappears behind an occluder and reappears in a predictable location. The subjects were 10 month old infants, and over time the subjects learn to anticipate the

reappearance of the object. An example of the stimuli is also found in the zip file with the data files (dynamic_aoi_example.mpg). This video is also found online at:

http://timestudioproject.com/manuals/dynamic_aoi_example_stimuli.mpg

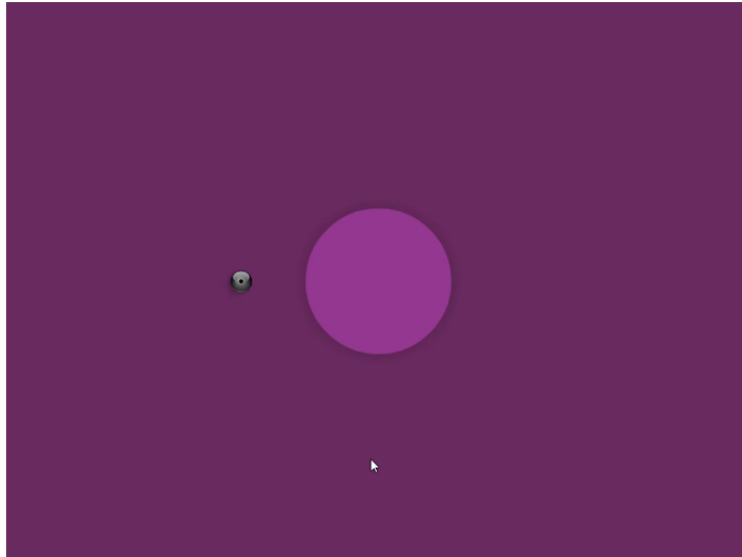


Figure 1. Screenshot of stimuli, with the moving object being left of the occluder in the center.

4. Overview

This tutorial demonstrates how TimeStudio can be used to analyze gaze latencies to AOIs using the example specified in section 3. In order to complete the analysis a series of operations need to be performed in TimeStudio.

- Launch MATLAB and TimeStudio
- Add data and define participants
- Read added data files
- Remove a time lag in gaze data in relation to event timestamps
- Adjust gaze position due to poor calibrations
- Plot a gaze plot with all subjects to get feedback on calibration correction
- Clear existing AOI definitions from memory
- Add new AOI definitions to memory
- Extraction and visualization of latencies

When this tutorial is complete the workflow and results windows should look identical to Figure 2.

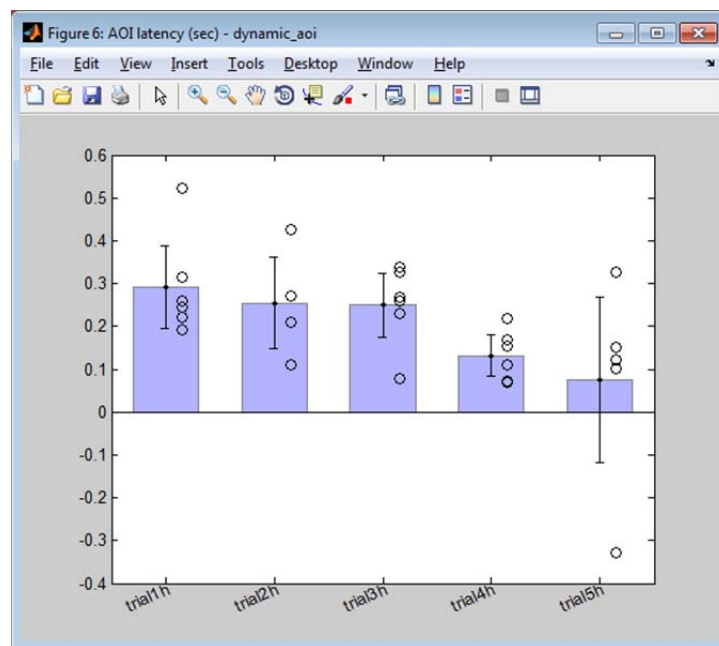
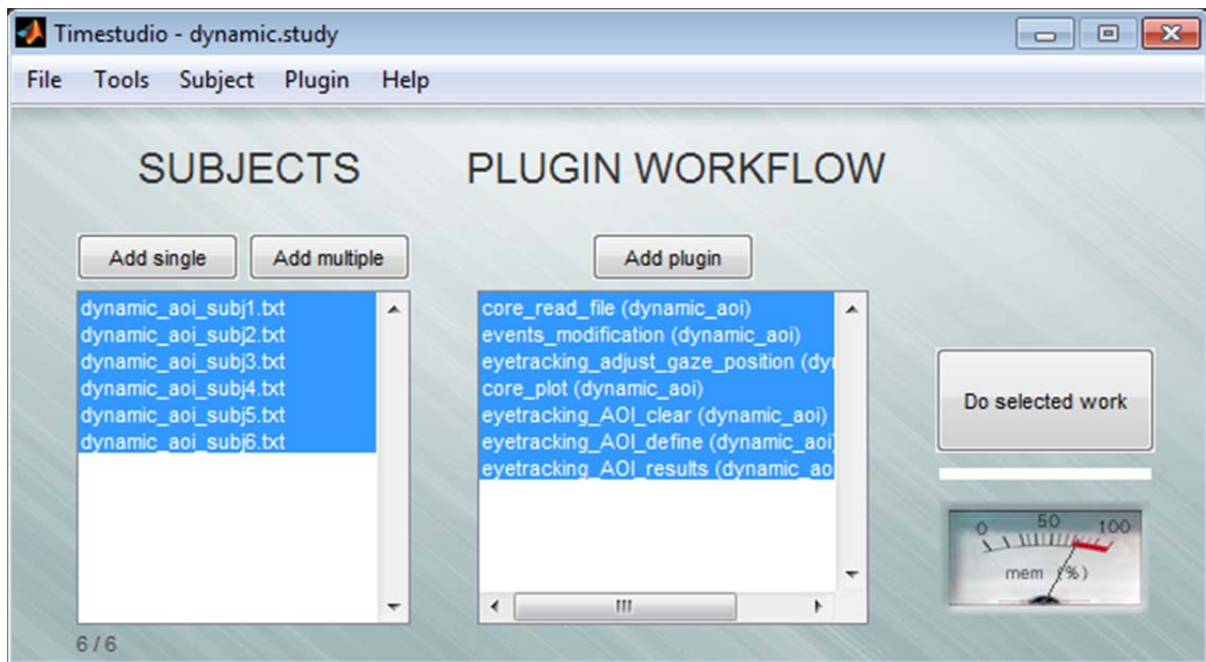


Figure 2. Final state of workflow and a graph demonstrating latencies to aois

5. Step-by-step-instructions

5.1 Launch MATLAB and TimeStudio

1. Start by launching MATLAB.
2. Either add the TimeStudio folder to the MATLAB path, or move into the TimeStudio folder.
3. Start TimeStudio by typing 'TimeStudio' in the MATLAB command window and press enter.

When TimeStudio starts the main window is displayed as seen in **Fel! Hittar inte referenskälla..**

The empty field to the left (**Fel! Hittar inte referenskälla.A**) is called "subjects" or "list of subjects" and the empty field to the right (**Fel! Hittar inte referenskälla.B**) is called "plugin workflow".

5.2 Add data and create subjects

In order to work with the data that is stored in the data files you need to load them into the Matlab memory. However, before you can load them you must tell TimeStudio where to find them.

TimeStudio keeps track of data files by assigning them to subjects in the subject list.

1. Press the "Add multiple subjects" button (Figure 3C) in the main TimeStudio window. This will open a file browser window.
2. Locate the six data files that belong to this demonstration, they should be named "dynamic_aoi_subjx.txt" with x being 1 to 6.
3. In the pop-up file browser window, mark all five files and press the "Open"-button to add all six files and create an individual subject for each file.

This should add six subjects to the subject list (Figure 4A).

Tip1: If each subject have multiple data files, press the "Add single subject" and select the data files for one subject. A new subject is then created which contain references to many data files.

Tip2: Double click a subject in the subject list! You will then be able to change the name, group, and other properties of the subject. You can also see which data files are assigned to this subject!

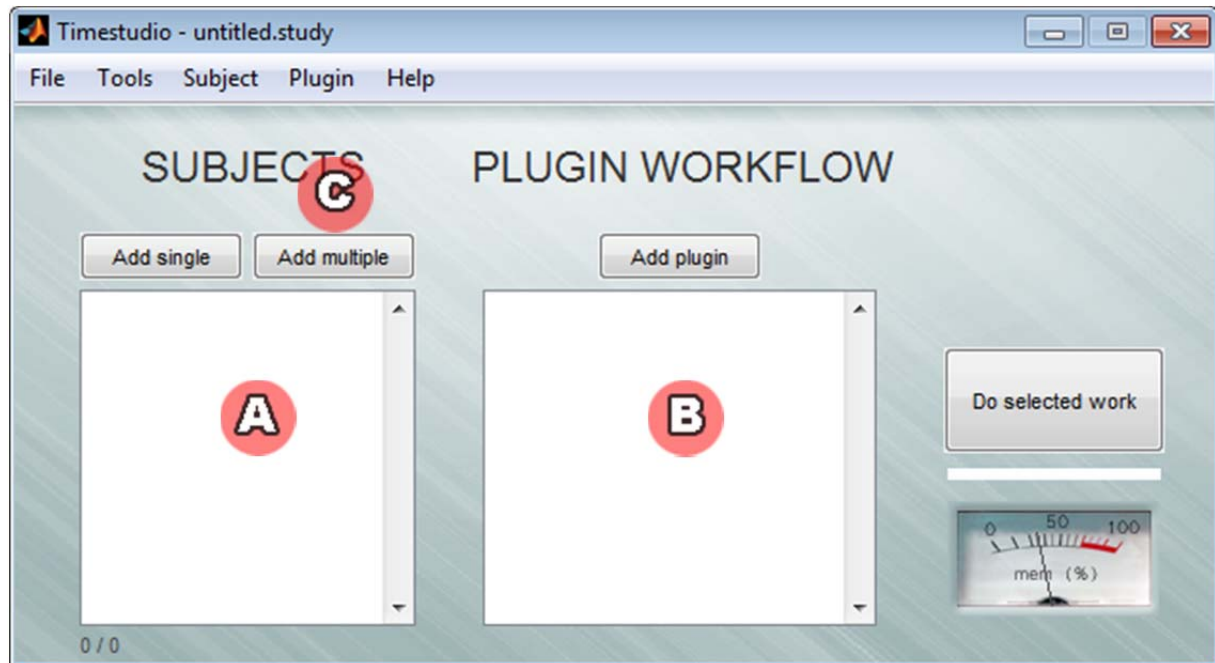


Figure 3. The main window at start up

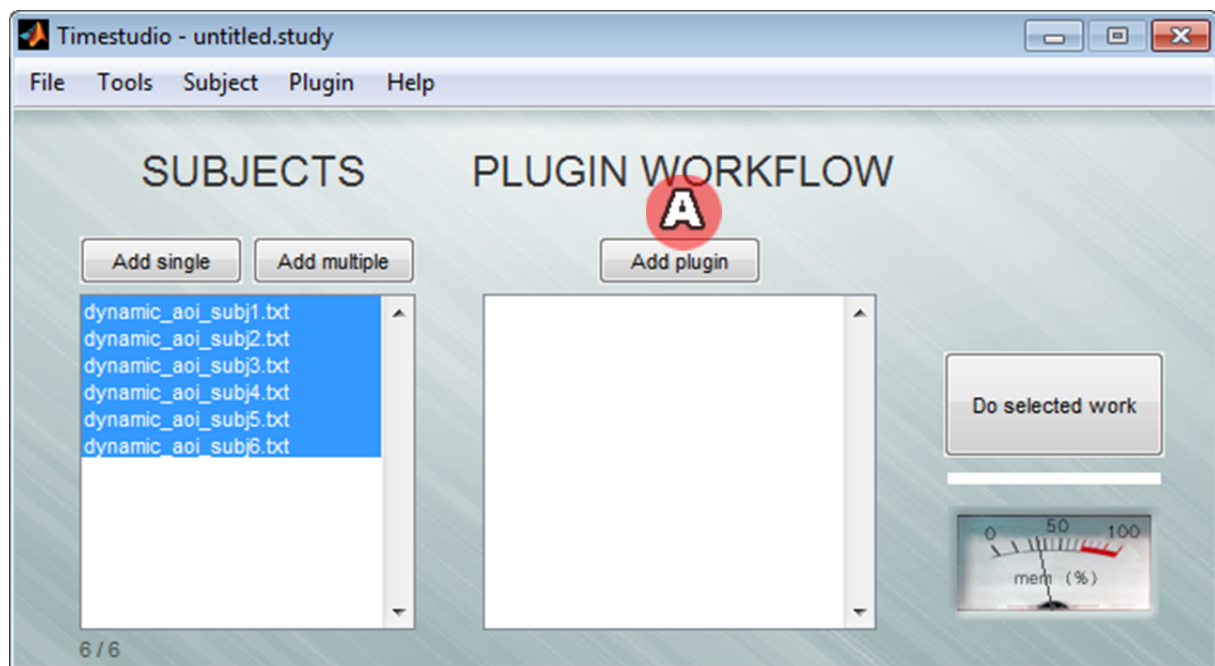


Figure 4. The main window after subjects have been created.

5.3 Read data files into memory

To extract relevant data we need to read the actual gaze data from the data files. To load the data files into memory you need to parse the data files and extract relevant data (such as x and y position of gaze, pupil size, stimuli events etc.). This is the purpose of the first plugin in our workflow. Later we will add other plugins as a chain to perform the whole analysis, but the first plugin is usually one that reads a data file. In our case it will be the plugin called **core_read_file**.

1. Press the “Add plugin” button (**Fel! Hittar inte referenskölla.A**) in the TimeStudio main window to bring up the pop up menu for selecting and adding a plugin to the plugin workflow. Hover to “core” and select **core_read_file** from the pop-out menu. This should bring up the window seen in Figure 5.
2. Start with adding a data field. Above A, in Figure 5, enter ‘eyetracking’.
3. The next step is to add variables. They are entered by typing the variable name in the “Add variable”-box that is below the A, and pressing enter. In this way enter the variables ‘x’, ‘y’, and ‘pupil’, one by one.

As can be seen in the list box B, there are two predefined variables – ‘events’ and ‘time’. These are used and required by almost all plugins and are therefore added by default.

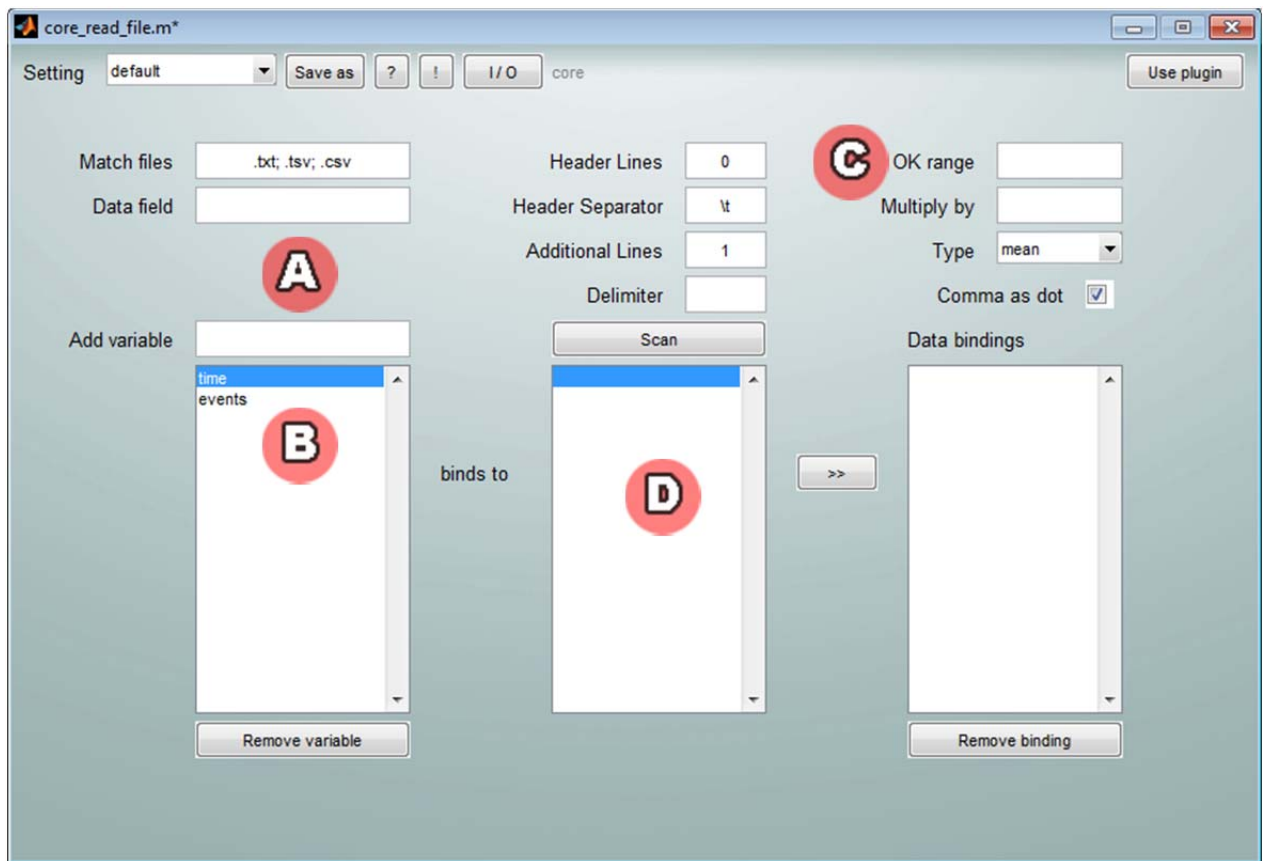


Figure 5. The *core_read_file* window.

4. After this, the number of header lines in the files to be read should be entered in the “Header Lines”-box (C in Figure 5). In our case, enter ‘18’ in this box, and the properties (column names, that is) from our input files will be listed in the list box at D in Figure 5.
5. This enables us to bind our variables to different data columns in the input files. Select ‘x’ in the variable list box (B in Figure 5), then select ‘GazepointX (R)’ and ‘GazepointX (L)’ in the properties list box (D) and finally press the “>>”-button to the left of D the figure. This will create a binding that will be displayed in the “Data bindings” list box. ‘x’ will be bound to the mean value of ‘GazepointX (R)’ and ‘GazepointX (L)’.

The results so far can be seen in Figure 6.

6. Continue with binding the variable ‘y’ to ‘GazepointY (R)’ and ‘GazepointY (L)’ in the same way.
7. Next up is to bind ‘time’ to ‘Timestamp’. This is done in the same way as for the other variables except that ‘0.001’ should be entered into the “Multiply by”-box (next to A in Figure

6). This means that the values of Timestamp are multiplied by 0.001, which gives is the time in seconds in this example.

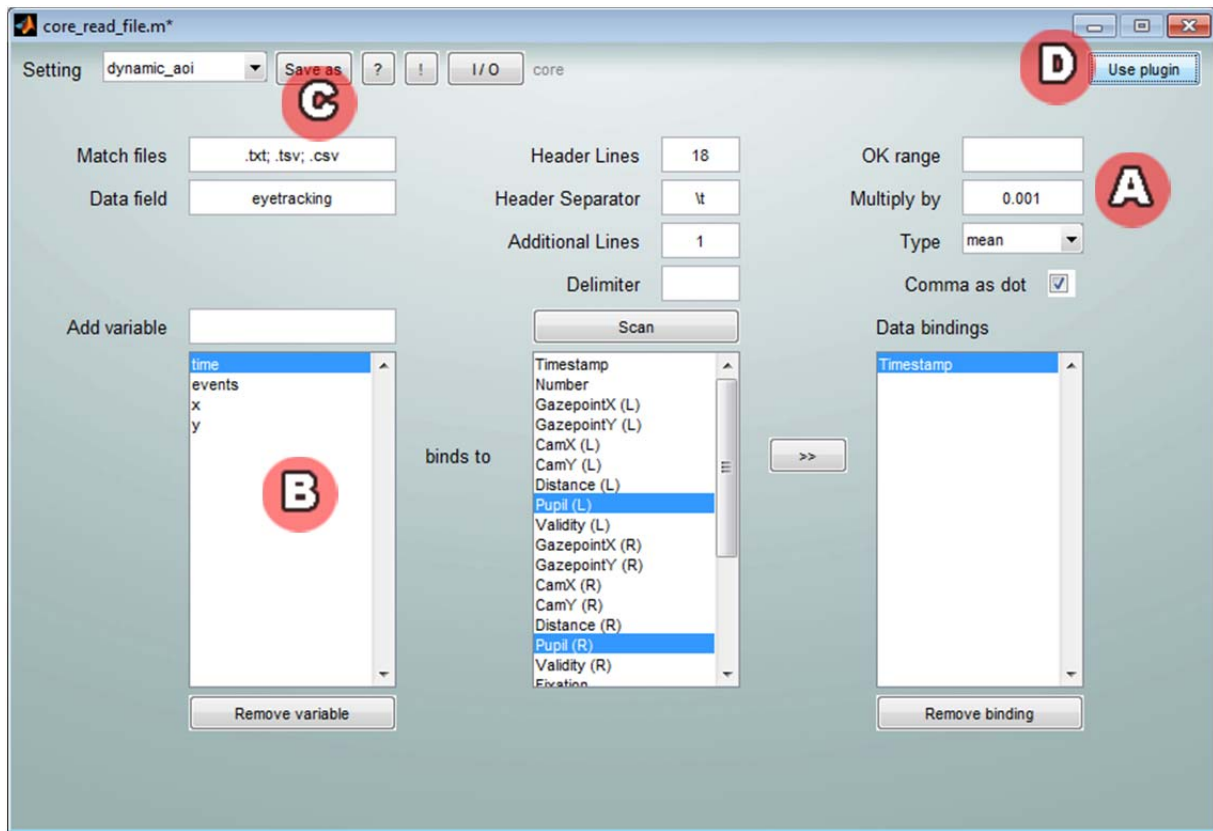


Figure 6. Adding bindings in `core_read_file`.

8. As the final binding we should bind 'events' with 'Description'. Now the "Type" should be set to 'event'.
9. The final steps are to set the acceptable range for some of the variables. Select 'x' in the variable-list box (at B in Figure 6), and then enter '0 1024' in the "OK range"-box (close to A in Figure 6).
10. Next, select 'y' and set the ok range to '0 768', and finish by selecting 'pupil' and setting the range to '2 10'.

Now that we have made all the settings required to read our data files we need to save them before using them.

11. Press “Save as” (above C in Figure 6). In the pop-up window enter some sensible name (e.g. ‘dynamic_aoi’), then press “Save”.
12. Press “Use plugin” (at D in Figure 6) to close the settings window and add this plugin with these settings to the plugin workflow.

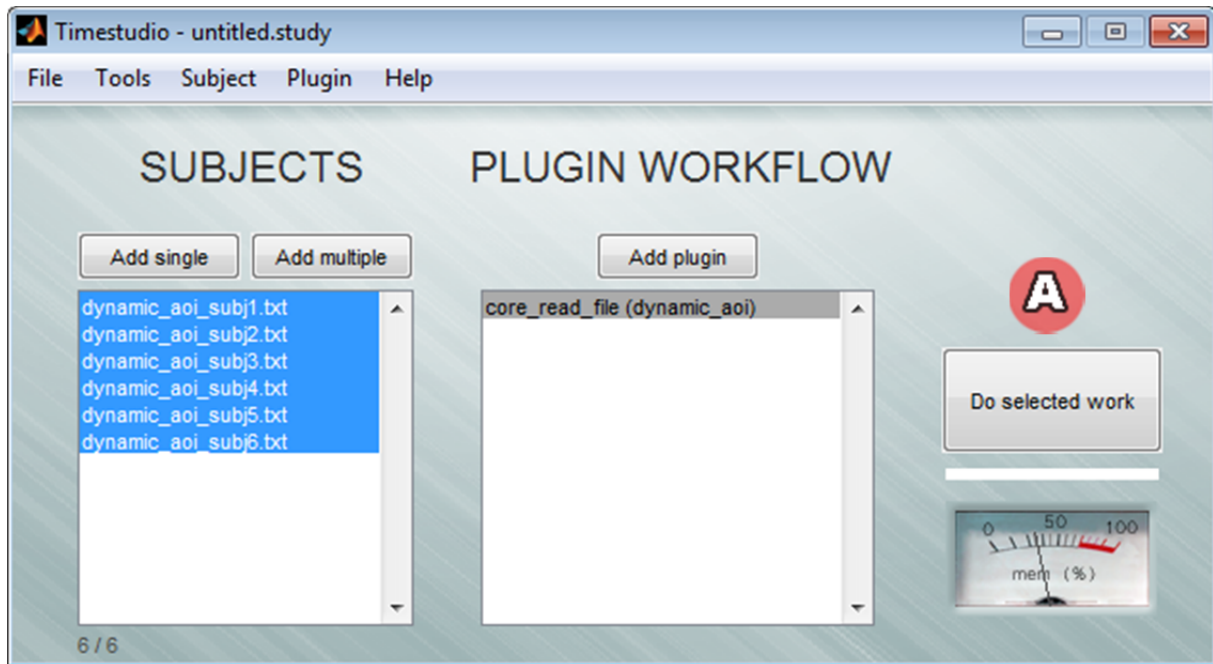


Figure 7. The main window after the first plugin was added.

Back in the main window:

13. Select all five subjects in the subject field, and select **core_read_file** in the plugin workflow (as can be seen in Figure 7).
14. Press the “Do selected work” button (Figure 7A) to read the data from the added files into the subjects.

While TimeStudio is processing a wait bar appears, and additional information is printed in the MATLAB command window.

5.4 Changing the names of events in the dataset

After loading the data into memory it is possible to find events in the datastream that represent when the stimuli was presented to the subjects. Only a few of these events are relevant for our analysis. We will now add a plugin that can rename these events: **events_modification**. In our example we have presented one video (called 'occlusionLearn_round.mov@:Oculomotortracking:' during recording) with several occlusion events. We will rename this event to "learn".

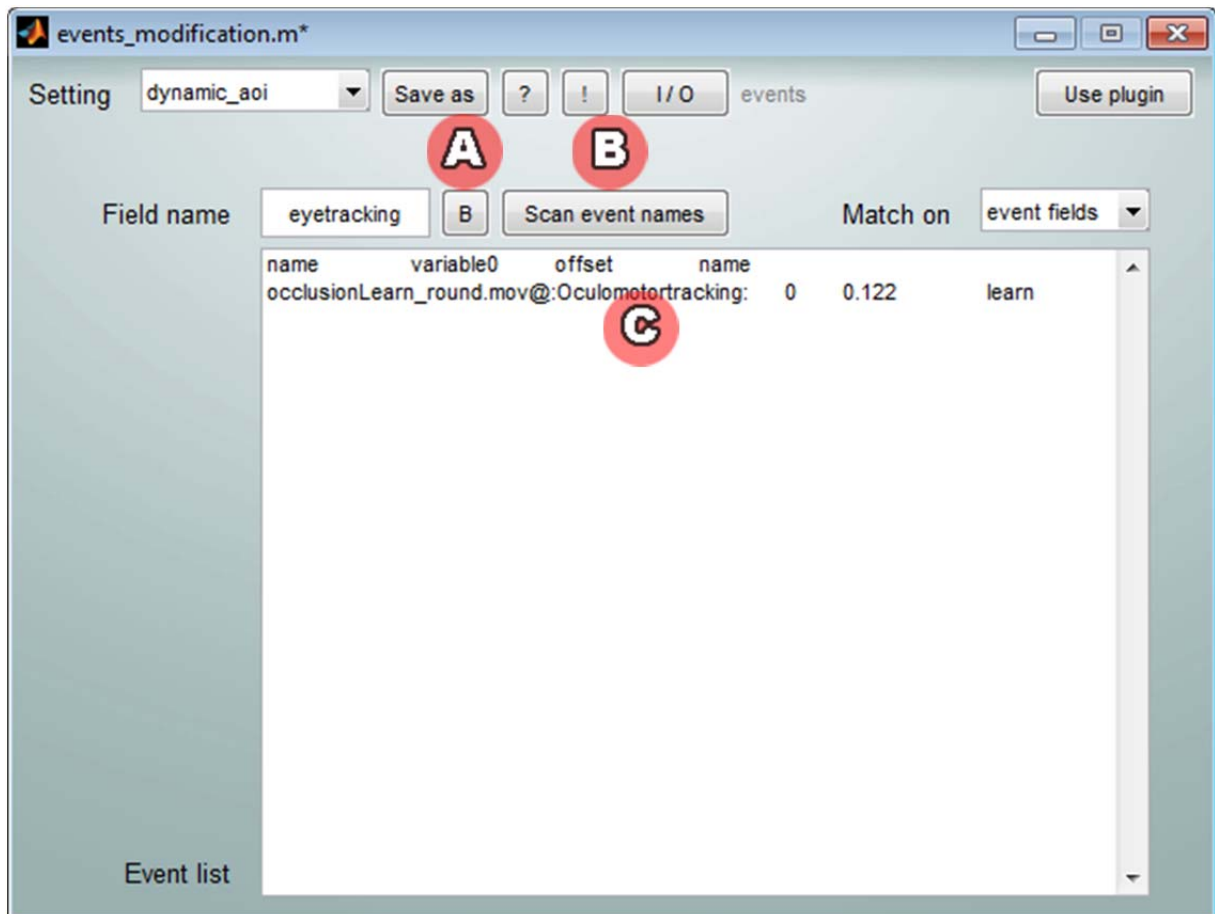


Figure 8. The *core_events_modification* plugin which can rename events and change time offsets.

We can also change the timing of the events to account for a time lag in the hardware setup where the data was recorded that was discovered after a timing test. The timing test showed that there was a 0.122 second time lag in the data

1. As before, press "Add plugin" from the main menu to bring up the pop up for selecting plugins, press "events" and select **events_modification**. This will open up the settings window for the plugin, see Figure 8.
2. Browse (Figure 8A) for the name of the data field to use. The only field visible should be 'eyetracking'. Select 'eyetracking' and press "done".

3. Then press the “Scan event names” button (Figure 8B), and the previously large empty field named “Event List” (Figure 8C) should now be filled with different names for events that are present in the data set.
4. There are many more events than we need to change, so delete the lines with events that are not of interest and make sure that the list looks like in Figure 8. The text should be tab separated. Please note that in Figure 8 the text is not tab separated, because all text should be visible in the manual. When the user enter the text, the rightmost part (‘learn’) will not be visible.
5. When your list looks OK, press the “Save as” button and save the setting as ‘dynamic_aoi’.
6. Press “Use plugin” to add the plugin to the “Plugin workflow” in the TimeStudio main window.

Back in the main window (Figure 9)

7. Select all subjects and the **events_modification** plugin, then press “Do selected work” button to apply the modifications to the events.

Your subjects will now have some events that are called “learn” that keeps information on the stimuli was presented and that has corrected for a time lag of 0.122 seconds.

Tip 4: Since the text is tab separated you can copy the list and paste it into Excel or most other spreadsheet programs. The values will then be aligned in rows and columns and can be easily changed. When you are done you can select all in your spreadsheet and copy it back into the TimeStudio plugin window!

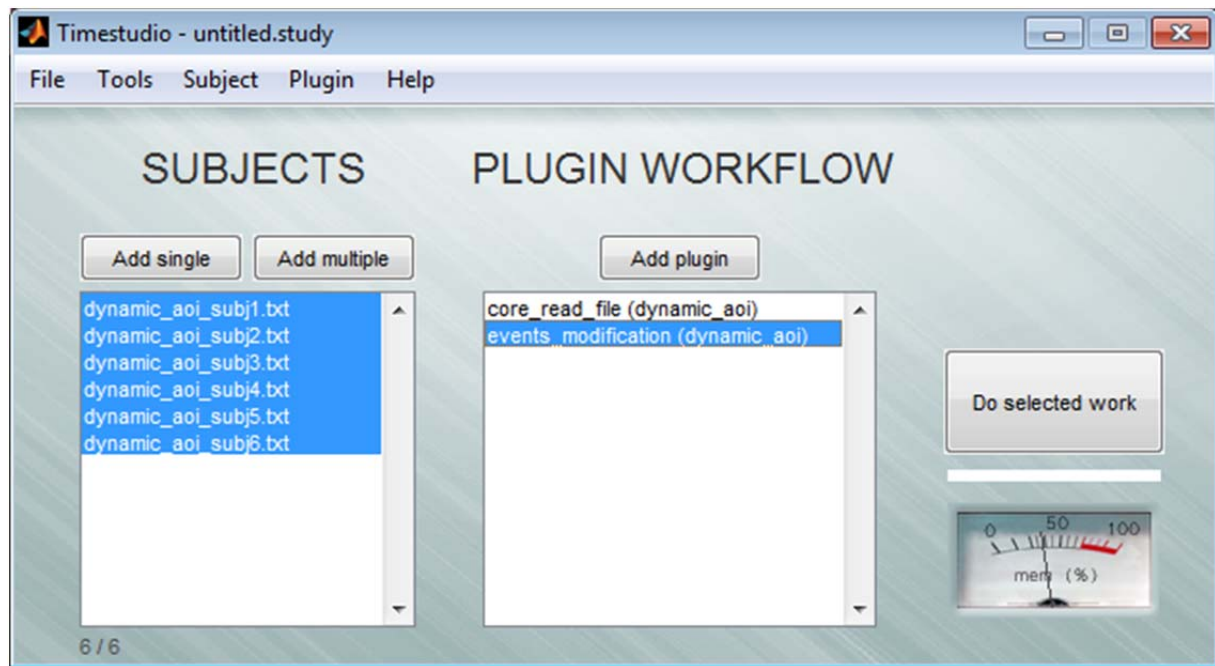


Figure 9. The main window after adding events_modification.

5.5 Compensating for bad calibration of gaze position

The data files in this example are recorded with 10 months old infants viewing the stimuli. At this age the participants cannot be instructed to sit still, and it is sometimes hard to get a good calibration. Also, during the session the infants tend to move their heads which leads to displacement of the gaze recording. It is possible to transpose the recorded gaze position to be better aligned with the stimuli presented. This is done by calculating histograms on the x and y position of the gaze and center the peak at a known position.

1. Press "Add plugin" and open the **eyetracking_adjust_gaze_position** plugin. The plugin is shown in Figure 10
2. Press the "B"-button at Figure 10A to bring up a window where you can select which event you want to use to align your data. Select the event "learn" and press "Done".
3. Input the values "0 30" in the time range textfield (Figure 10B). By doing this the plugin will only use gaze values starting from 0 seconds from event start to 30 seconds after event start (which is the whole length of the stimuli presented).
4. Next, enter the values, shown in the text boxes, controlling the x alignment (Figure 10C) and y alignment (Figure 10D). To get visual feedback on how the histogram is shifted it is possible to check the Figure 10E checkbox. One plot per subject will then be plotted, showing the

histogram for the whole screen, the interval to find peaks in, the peak position and the correct position.

5. To finish, press “Save as” and save the setting with the name ‘dynamic_aoi’.
6. Press “Use plugin” to add the the plugin to the workflow.

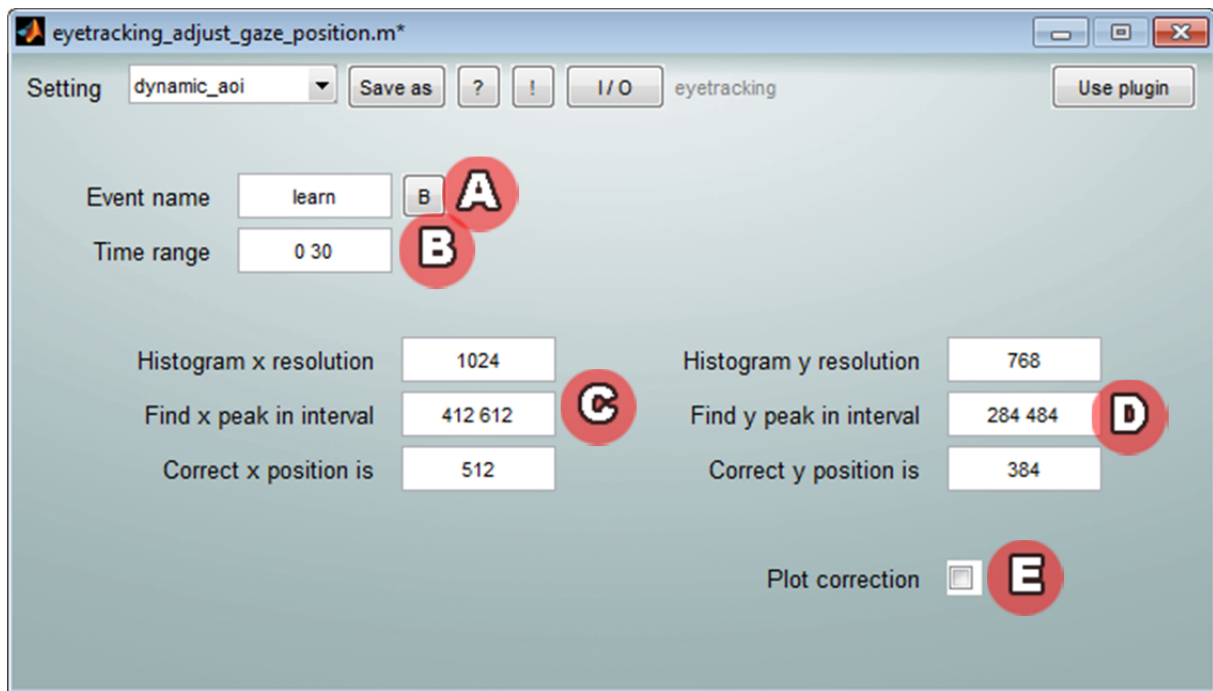


Figure 10. The *eyetracking_adjust_gaze_position* plugin setting window.

5.6 Plotting the gaze trajectories to see raw data

It is very important to check the quality of the data, and an easy way of making sure that the data is plausible is to create a plot for visual inspection (you may also try to run the workflow without adjusting the gaze position to see how that plugin affects the data).

1. Press “Add plugin” and open the **core_plot** plugin. The plugin is shown in Figure 11
2. Press the “B”-button at Figure 11A to bring up a window where you can select which data field to use. The only field visible should be ‘eyetracking’. Select ‘eyetracking’ and press “Done”.
3. Press the “B”-button at Figure 11B to bring up a window where you can select which event you want to use to align your data. Select the event “learn” and press “Done”.

4. Input the values “0 30” in the time range textfield Figure 11C). By doing this the plugin will only use gaze values starting from 0 seconds from event start to 30 seconds after event start (which is the whole length of the stimuli presented).
5. The final step is to select which data to plot. This is done for x- and y-data on the left and right side of D in Figure 11. Browse with the “B”-button and select x and y, respectively.
6. Save as ‘dynamic_aoi’ and press “Use plugin”.
7. The main window should now look as Figure 12.
8. Select **eyetracking_adjust_gaze_position** and **core_plot** and press “Do selected work”.

TimeStudio will then process the data and create a figure identical to Figure 13.

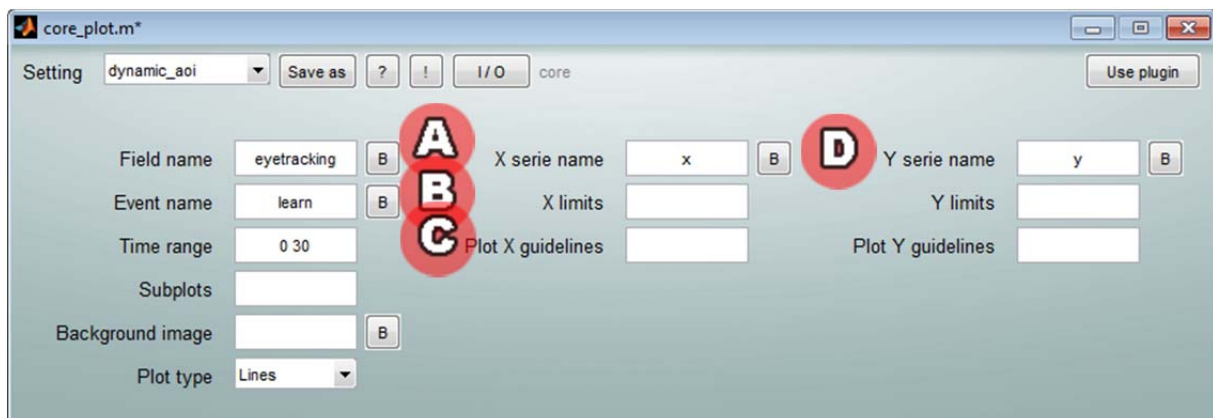


Figure 11. The core_plot plugin.

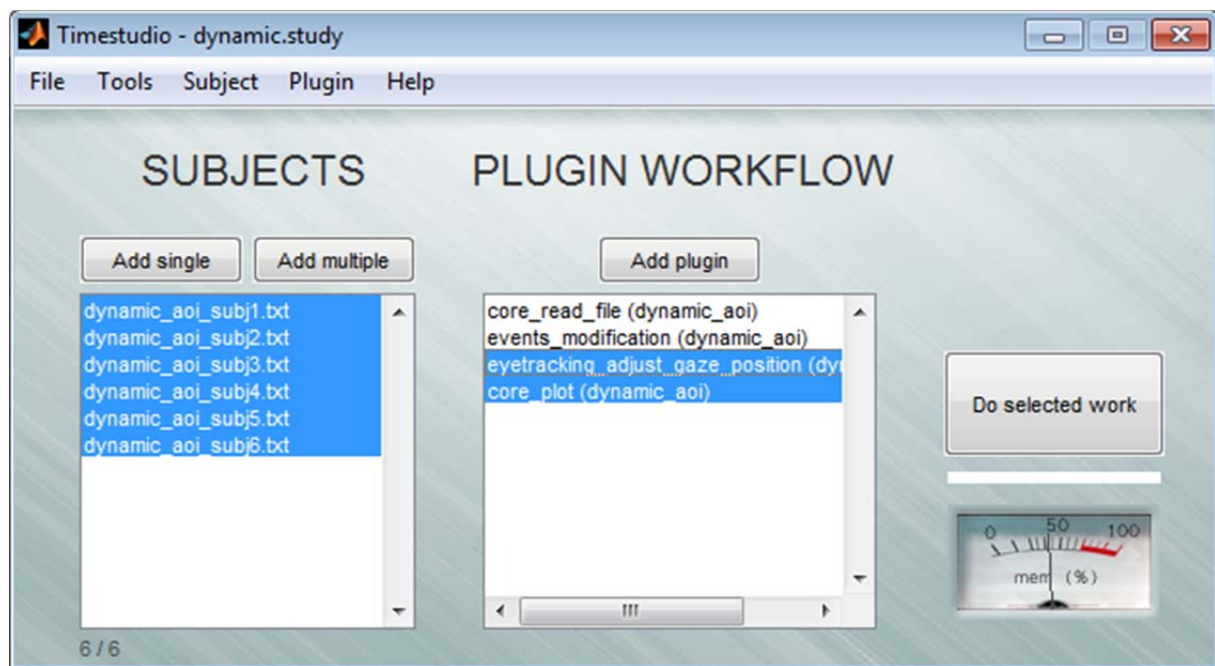


Figure 12. The main window with eyetracking_adjust_gaze_position and core_plot added.

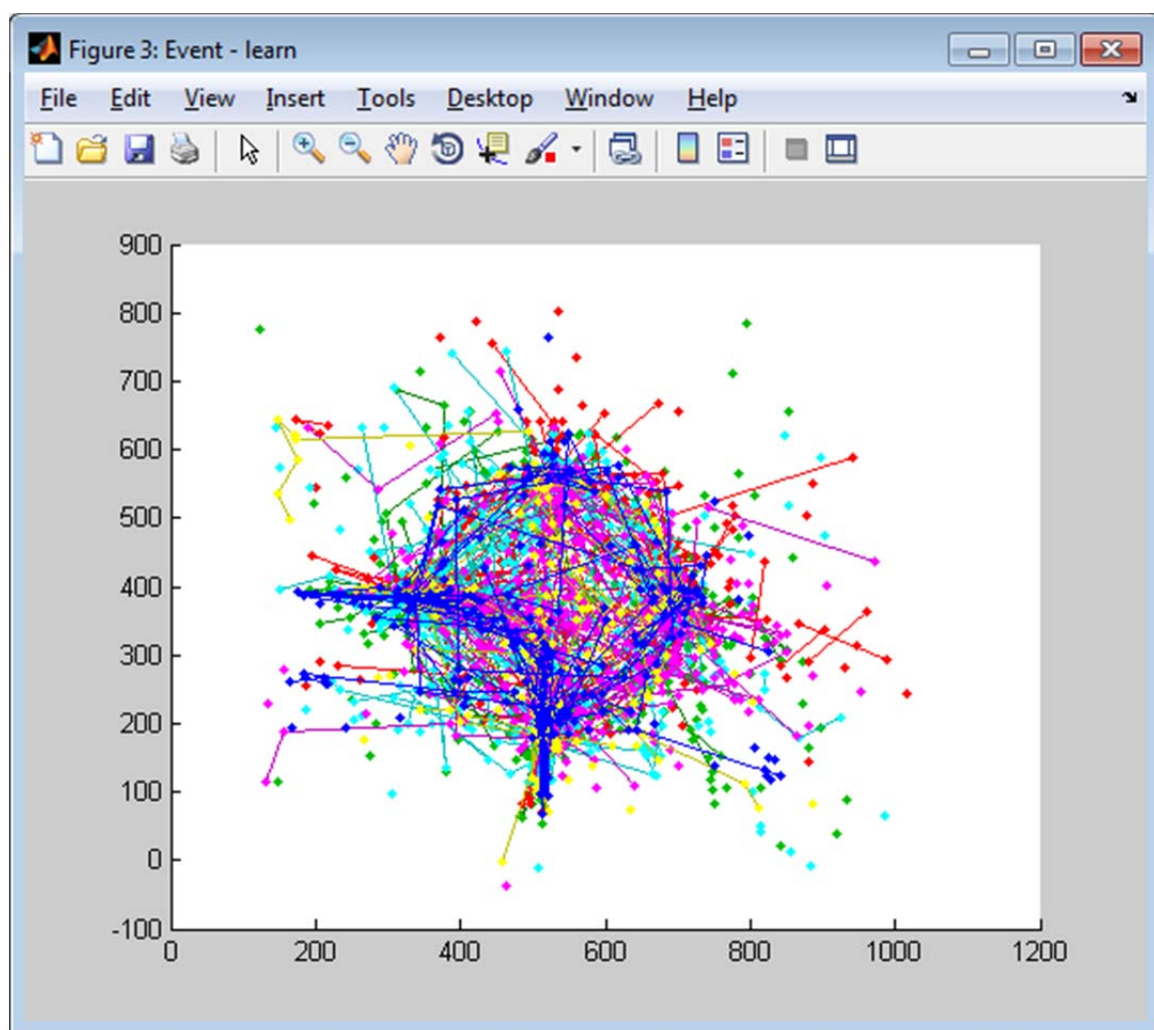


Figure 13. A plot of the subjects gaze behavior during stimuli presentation.

Until now we have only made sure that we have data to work with in memory and adjusted so that we have events that represent our different videos. We will now continue with working with areas of interest.

5.7 Cleaning up before analysis

In eye tracking studies it is common to define interesting areas on a monitor. These areas are called areas of interest (AOIs), and it is possible to define multiple AOIs in TimeStudio with a variety of behaviors. When areas are defined they will stay in memory until they are explicitly cleared by using the plugin **eyetracking_AOI_clear**. If you only run the whole workflow once you do not need to clear the AOIs, but it is good practice to always include the plugin **eyetracking_AOI_clear** before your AOI definitions. That way you do not risk that any old AOI definitions remain in memory.

1. Press “Add plugin” from the main menu to bring up the pop up for selecting plugins and select **eyetracking_AOI_clear** (Figure 14).
2. This plugin has no parameters to set. Save it as ‘dynamic_aoi’ and click “Use plugin” to add the plugin to the workflow in the main window.

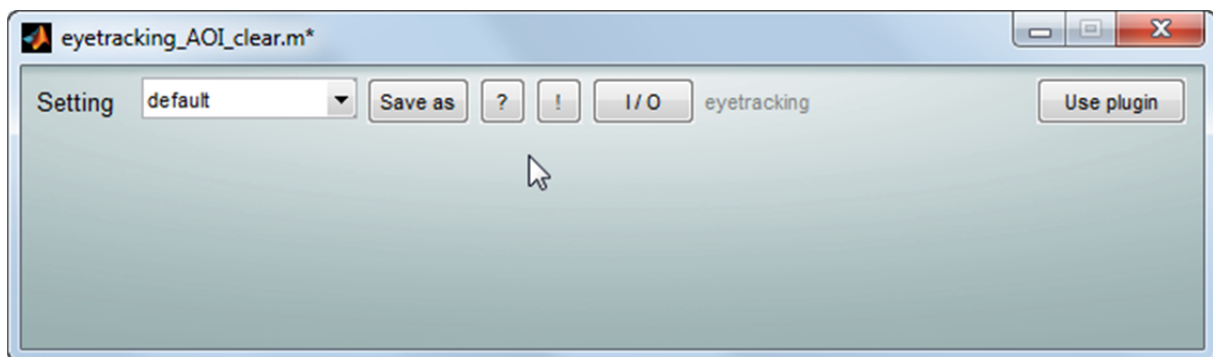


Figure 14. The *eyetracking_AOI_clear* plugin which will clear existing AOI definitions.

5.8 Defining Areas of Interest for the event “learn”

To define AOIs the plugin **eyetracking_AOI_define** should be used.

1. In the usual way, press “Add plugin” from the main menu to bring up the pop up for selecting plugins and select **eyetracking_AOI_define**.

This plugin will open two windows: one window with the plugin settings (Figure 15) and a preview window (not shown).

The **eyetracking_AOI_define** is one of the most complex plugins in TimeStudio and it can be used in many different contexts. We will describe a common way to define dynamic AOIs here (please see the manual about static AOIs for defining less complex behavior).

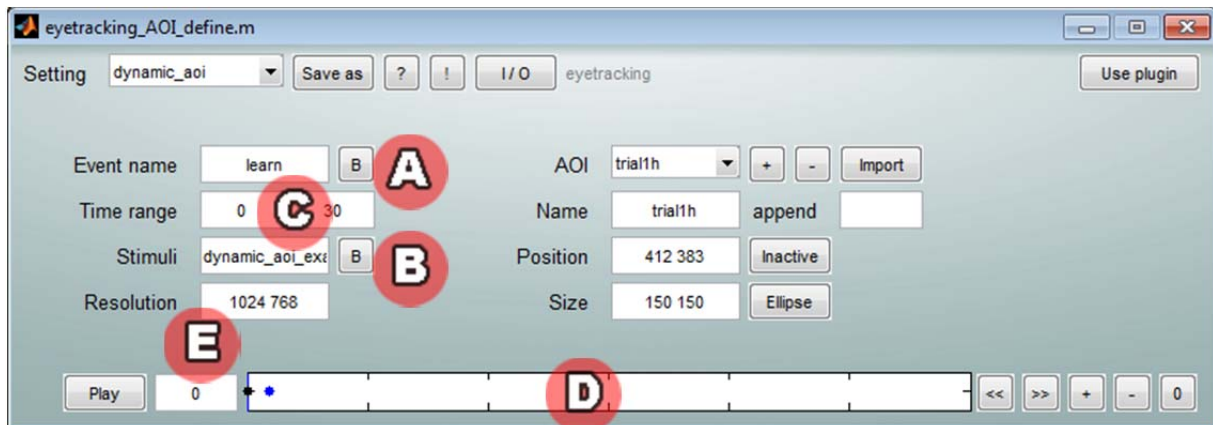


Figure 15. The *eyetracking_AOI_define* setting window.

2. Press the “B”-button at Figure 15A to bring up a window where you can select which event you want to apply the AOIs to. Select the event ‘learn’ and press “Done”.
3. In this example it is possible to show the stimuli in the preview window. Do this by clicking the “B”-button at Figure 15B and open the file *dynamic_aoi_example_stimuli.mpg*. The first frame of the stimuli movie should now appear in the preview window.
4. As in previous plugins the time interval to process should be given. Insert the values 0 and 30 in the time range textfields (**Fel! Hittar inte referenskölla.C**). When you hit enter (or leave the text fields) the timeline (Figure 15D) at the bottom of the setting window will update. You can use the mouse to click or drag in the timeline to change the current time (Figure 15E) that is shown in the AOI preview window, and the movie will update simultaneously. Before you continue, make sure that the current time is set to 0.

When the **eyetracking_AOI_define** plugin is opened the first aoi is already added to the timeline.

5. Change the name of the aoi by changing the name text field (Figure 16A) from ‘aoi1’ to ‘trial1h’.
6. Reposition the center of the aoi to x-coordinate 412 and y-coordinate 383 by entering ‘412 383’ in the position text field (Figure 16B).

7. Change the size of the aoi (Figure 16C) to '150 150', meaning 150 pixels wide and 150 pixels high.
8. Also, press the button at Figure 16C to toggle the shape of the AOI from "Rectangle" to "Ellipse". The first aoi has now got the correct name, location, size and shape.

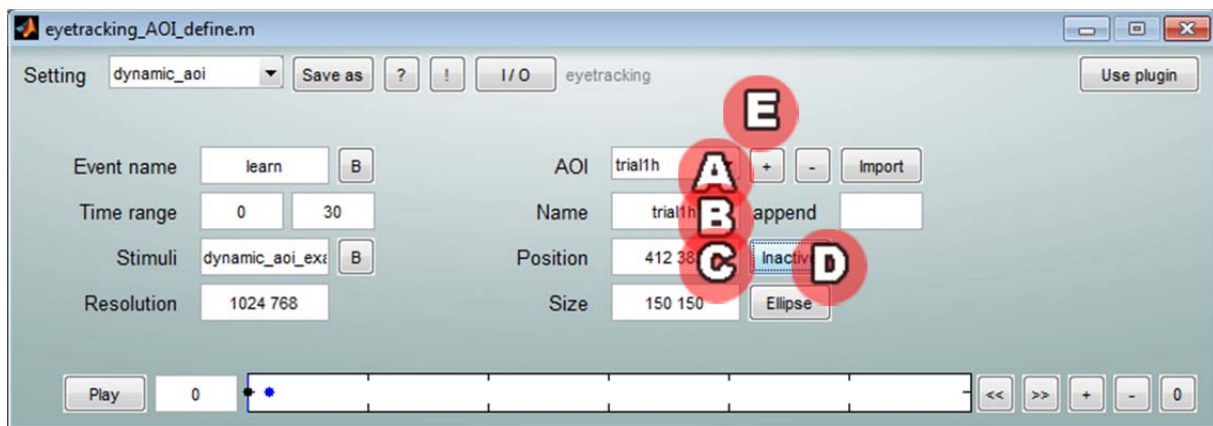


Figure 16. The eyetracking_AOI_define setting window continued.

9. Add more aois by pressing the "+"-sign at Figure 16E so that you can create similar aois with the names "trial2h", "trial3h", "trial4h" and "trial5h".

All of the aois should be inactive at time 0, so they should be red in the preview window (see figure 16). We will now add dynamic behavior by turning the aois on and off at different time points.

10. Start by selecting the aoi "trial1h" in the dropdown menu next to the E in Figure 16.

Tip: right click in the preview window to bring up preview alternatives. If you hide all inactive aois, only the currently selected aoi will be shown even if it is inactive.

Tip: in this example we do not change size or position of the aois. However, this is easy to achieve by changing the size or position at the different time points instead of changing the active/inactive states.



Figure 17. The eyetracking_AOI_define preview window with 5 inactive aois.



Figure 18. The eyetracking_AOI_define setting window - activating aois.

The first aoi should become active before the object reappears on the left side of the occluder for the first time, 4.011 seconds after the movie starts.

11. Set the current time to 4.011 by changing the value in Figure 18A and hitting enter.

12. Then click the "Inactive" button (Figure 18B) so that the aoi becomes "Active".

When you make a change to an aoi the plugin will create a key frame at that time point, which is indicated by a blue dot at the corresponding time in the timeline (you can use the mouse to drag the timeline over the key frame to see how the state of the aoi changes at different times). For more information on key frames please see http://en.wikipedia.org/wiki/Key_frame.

13. Then change the time to 5.011 (Figure 18A) which corresponds to the first frame when the object is visible behind the occluder.

14. Press the "0" button (Figure 18C) and another key frame will be added, but this time nothing about the aoi appearance will change.

The "0" key sets the zero time for this particular aoi, which means that the resulting latencies from the analysis is relative to this time point. An aoi can only have one zero time, and this key frame is marked by a black color on the key frame dot.

15. Change time again, to 6.011 (Figure 18A) and deactivate the aoi by pressing the "Active" button at Figure 18B. The settings window should now look identical to Figure 18.

Repeat the key frame procedure for the other aois, but use the following key frames instead:

| | |
|---------|---|
| trial2h | At time 10.067, activate aoi At time 11.067, set zero time At time 12.067, deactivate aoi |
| trial3h | At time 16.160, activate aoi At time 17.160, set zero time At time 18.160, deactivate aoi |
| trial4h | At time 22.249, activate aoi At time 23.249, set zero time At time 24.249, deactivate aoi |
| trial5h | At time 28.319, activate aoi At time 29.319, set zero time At time 30.319, deactivate aoi |

All aois are now defined.

16. Save the settings as 'dynamic_aoi' and click "Use plugin" to add the plugin to the workflow in the main window.

17. Run the **eyetracking_AOI_define** plugin by pressing "Do selected work".

5.9 Getting the latency results

When the aois has been defined it is possible to extract a number of measures such as latencies to first gaze visit in the aoi, looking time in aoi, number of visits in aoi and pupil size. In this example we are interested in getting the gaze latencies to the aois when they are active. This can be extracted by adding the plugin **eyetracking_AOI_results**.

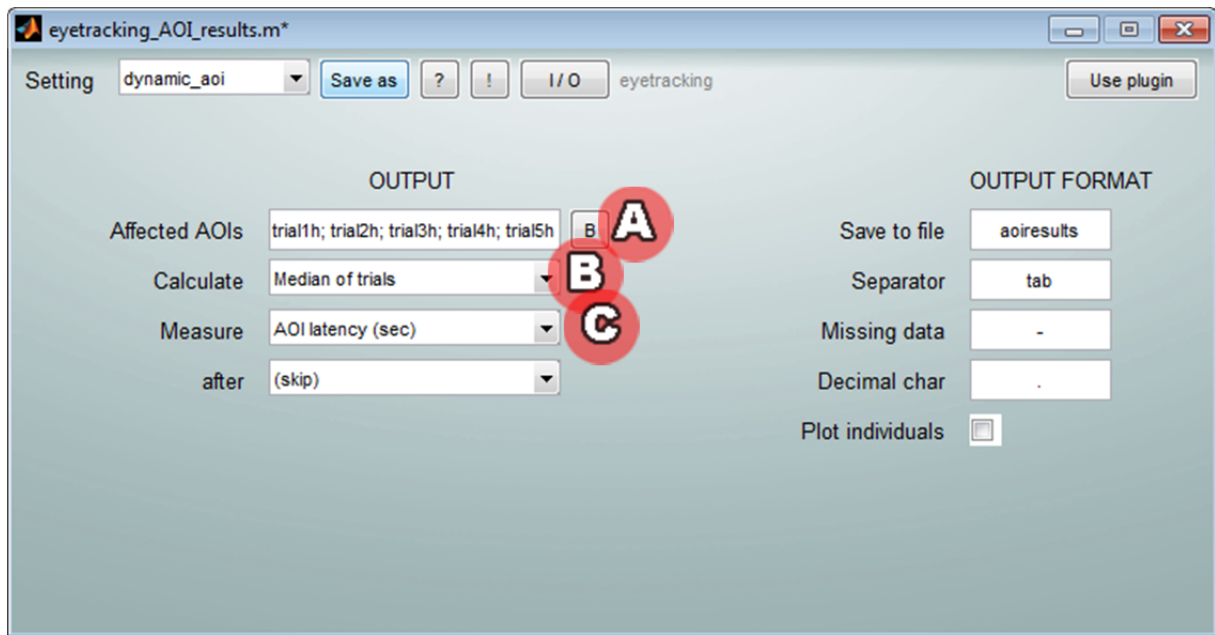


Figure 19. The *eyetracking_AOI_results* plugin.

1. Press “Add plugin” from the main menu to bring up the pop up for selecting plugins and select **eyetracking_AOI_results**.
2. Start by pressing the “B”-button to browse for which aois you want to get results from (Figure 19A). Select all aois and press “Done”.

It is possible to make a calculation over all trials within a subject (mean/median/min/max/range). This value is used for plotting the results, and will appear in a results text field later.

3. Select ‘Median of trials’ in dropdown Figure 19B.
4. Select what kind of measure we want to use for every trial (looking time/latency/visits etc.) in dropdown Figure 19C. In this example we use ‘AOI latency (sec)’ which tells the plugin to extract the latency to each aoi within each trial relative to the aoi zero time (all aois will have possible latency values ranging from -1 to 1).
5. Save the plugin as ‘dynamic_aoi’ and press “Use plugin”.

Back in the main window, run the results plugin by having all subjects in the subject list selected and only the **eyetracking_AOI_results** plugin selected in the workflow list. Then press “Do selected work” to calculate the results.

The results plugin may take some time to finish, but when it is done it will create three new windows (Figure 20, Figure 21, and Figure 22). The top window (Figure 20) is a bar plot showing the median latency time for each AOI. Error bars are standard error. Beside the error bars there are also circles representing the median value of each individual subject. By looking at this plot you can get a feeling for differences and distributions of the AOI latencies. In our example we can see that the subjects have shorter latencies at the end of the movie (trial4h and trial5h) than in the beginning of the movie (trial1h and trial2h).

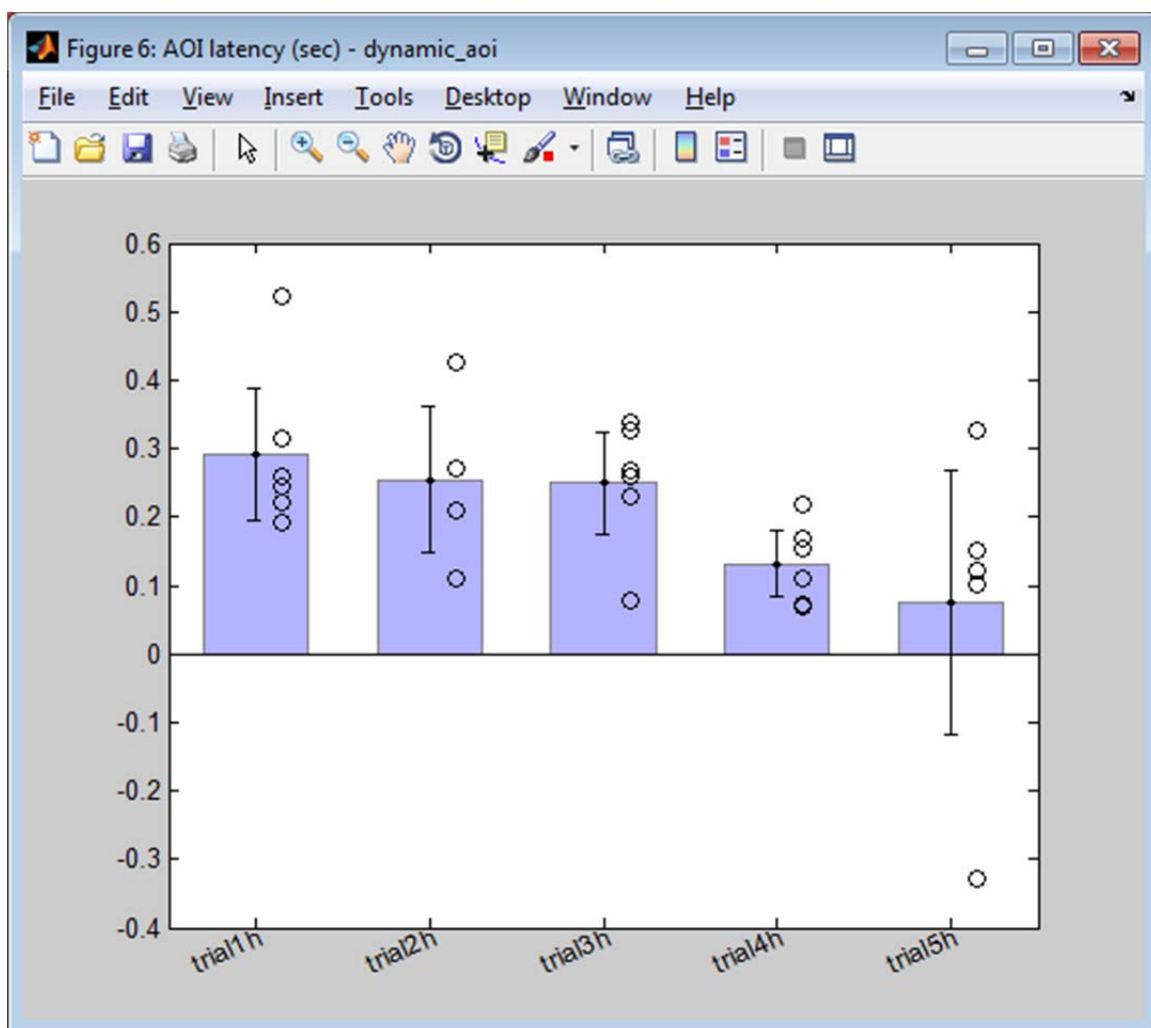


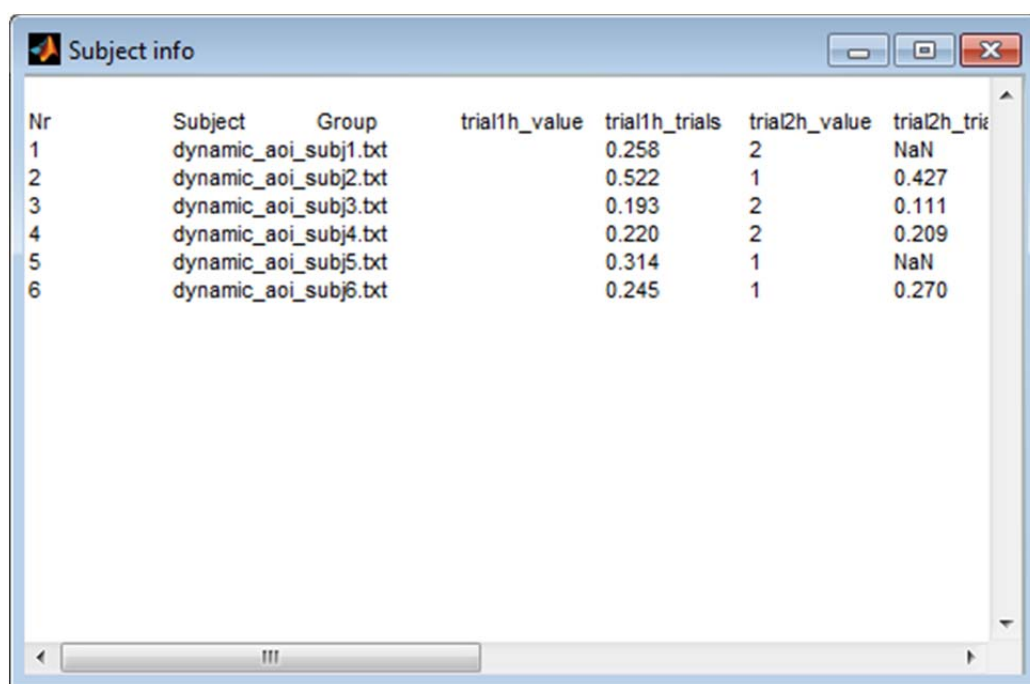
Figure 20. The resulting plot from eyetracking_AOI_results.

If you close the plot window or move it to the side you will see a window with a text field with the data that was represented in the plot (together with the number of trials for each subject and AOI), see Figure 21. You can click in this text field and copy the content by pressing Ctrl+A followed by

Ctrl+C. You can now paste the data into a spreadsheet or statistical program such as Excel, Statistica, SPSS, etc., for further analysis of this data. However, to get the value for each individual trial you need to move or close this window to reveal the last results window (Figure 22).

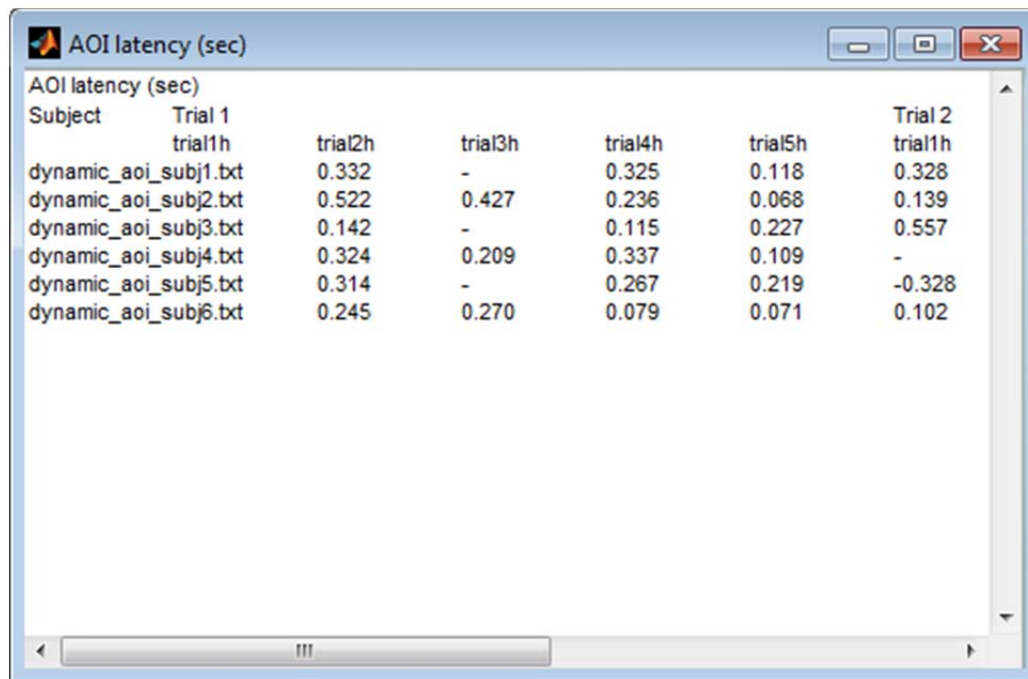
The last result window contains the values for each AOI for every individual trial for every subject. Just like the previous window you can copy and paste it into a spreadsheet or statistical program by clicking in the text field and pressing Ctrl+A followed by Ctrl+C. You can then do further analysis of your data by removing outliers and doing the statistical test of your choice.

In most cases it is sufficient to stop here and calculate the statistics in third party software.



| Nr | Subject | Group | trial1h_value | trial1h_trials | trial2h_value | trial2h_trials |
|----|-----------------------|-------|---------------|----------------|---------------|----------------|
| 1 | dynamic_aoi_subj1.txt | | 0.258 | 2 | NaN | |
| 2 | dynamic_aoi_subj2.txt | | 0.522 | 1 | 0.427 | |
| 3 | dynamic_aoi_subj3.txt | | 0.193 | 2 | 0.111 | |
| 4 | dynamic_aoi_subj4.txt | | 0.220 | 2 | 0.209 | |
| 5 | dynamic_aoi_subj5.txt | | 0.314 | 1 | NaN | |
| 6 | dynamic_aoi_subj6.txt | | 0.245 | 1 | 0.270 | |

Figure 21. Subject summary from eyetracking_AOI_results.



| Subject | Trial 1 | | | | | Trial 2 |
|-----------------------|---------|---------|---------|---------|---------|---------|
| | trial1h | trial2h | trial3h | trial4h | trial5h | trial1h |
| dynamic_aoi_subj1.txt | 0.332 | - | | 0.325 | 0.118 | 0.328 |
| dynamic_aoi_subj2.txt | 0.522 | 0.427 | | 0.236 | 0.068 | 0.139 |
| dynamic_aoi_subj3.txt | 0.142 | - | | 0.115 | 0.227 | 0.557 |
| dynamic_aoi_subj4.txt | 0.324 | 0.209 | | 0.337 | 0.109 | - |
| dynamic_aoi_subj5.txt | 0.314 | - | | 0.267 | 0.219 | -0.328 |
| dynamic_aoi_subj6.txt | 0.245 | 0.270 | | 0.079 | 0.071 | 0.102 |

Figure 22. Trial information from eyetracking_AOI_results.

6. Final comments and remarks

Worth mentioning is that during this demonstration all plugins has been applied one by one as they were added. It is also possible to first add all plugins (still in this particular order) to the study and then applying them all at once by selecting all subjects and all plugins and then pressing the “Do selected work” button.

7. Acknowledgements

The authors want to thank the following people for comments and suggestions for this manual and for testing the workflow: Gustaf Gredebäck, Terje Falck-Ytter, Joshua Jovrud , Tommie Forslund, Christine Fawsett and Ida Hensler. The manual was funded by grants ERC-StG- 312292-CACTUS and the Swedish Research Council in partnership with FAS, FORMAS and VINNOVA [Cross-disciplinary research programme concerning children's and young people's mental health; grant number 259 - 2012-24].