Table of Contents

Required Downloads

Installing Anaconda

Mac Installation

1. Navigate to the Anaconda homepage, anaconda.com
2. Click the “Get Started”
3. Click the “Download Anaconda installers”
4. Locate the installers for Mac and click on the “64-Bit Graphical Installer” to download
5. Double click the downloaded file
6. Click the “Continue” on the pop up with the message “This package will run a program to determine if the software can be installed”
7. Click “Continue” on the first page of the installer menu
8. Click "Continue" on the Read Me page
9. Click "I Agree" to the terms of the software license agreement
10. Click "Continue" on the license page
11. On the Destination Select page choose one of the following options
12. “Install for me only”
13. Downloads anaconda for the current desktop profile
14. “Install on a specific disk” (administrator privileges required)
15. Downloads anaconda for all desktop profiles
16. Choose the disk that contains the target desktop profiles
17. Click “Continue” on the Destination page
18. Click “Continue” on the PyCharm IDE page
19. Click “Close” on the Summary page

Windows Installation

1. Navigate to the Anaconda homepage, anaconda.com
2. Click the “Get Started” button
3. Click the “Download Anaconda installers” button
4. Locate the installers for your Windows and click on the “64-Bit Graphical Installer” to download
5. Double click the downloaded file to launch the installer
6. Do not launch installer from Favorites folder
7. Click “Next” on the first page of the installer menu
8. Click “I Agree” to the terms of the software license agreement
9. On the Destination Select page choose one of the following options
10. “Just Me” *Less prone to errors*
11. Downloads anaconda for the current desktop profile

12. “Install on a specific disk” (administrator privileges required)
13. Downloads anaconda for all desktop profiles
14. Choose the disk that contains the target desktop profiles
15. On the “Choose install Location page”, use the default destination folder to install anaconda in
16. Make sure the directory path doesn’t contain spaces or unicode characters
17. Regular alphabetical characters are allowed
18. See resources for unicode character table
19. Click “Next
20. On the Advanced Installation Options Page
21. Do not check the box to add Anaconda3 your PATH environment variable
22. Check the box to register Anaconda3 as your default python
23. Click “install”

Windows Troubleshooting

1. If antivirus software is active, temporarily disable it. Re-enable after installation concludes
2. If you installed for all users, OR under Administrative Privileges: uninstall anaconda (Uninstall directions in Resources) and reinstall for “Just me” without Administrative Privileges
3. Double check that the Installation directory path doesn’t contain spaces or unicode characters. If it does then uninstall and reinstall in a new directory

Anaconda Resources

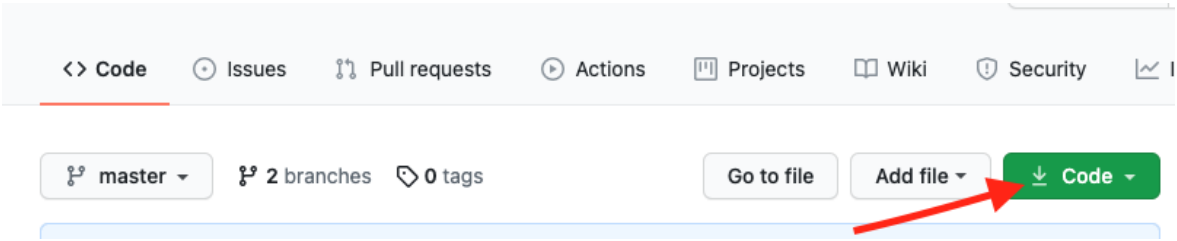
- Installation Documentation: https://docs.anaconda.com/anaconda/install/
- Uninstall Documentation: https://docs.anaconda.com/anaconda/install/uninstall/
- User Guide: https://docs.anaconda.com/anaconda/user-guide/
- Unicode Character Chart: https://en.wikipedia.org/wiki/List_of_Unicode_characters

Downloading Pycharm

Choose the community download for your OS and follow download instructions found on https://www.jetbrains.com/pycharm/download/

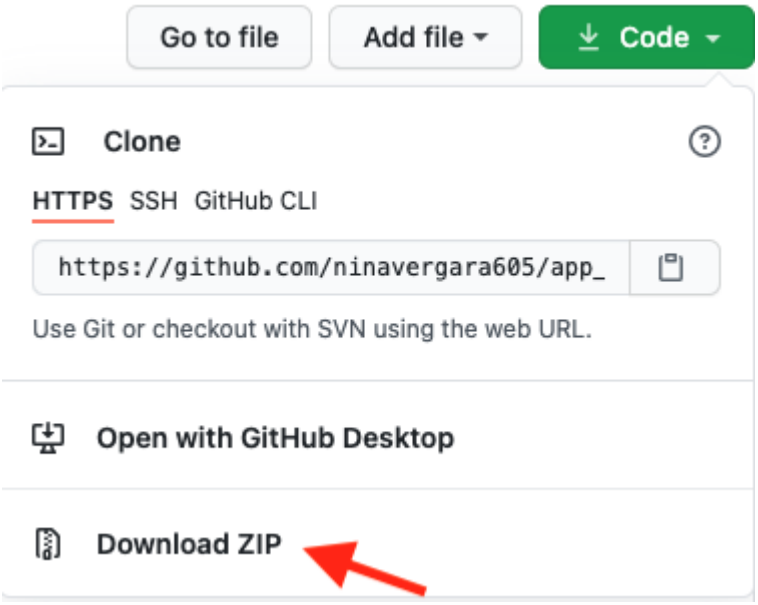
Downloading the Pipeline

1. Navigate to https://github.com/ninavergara605/app_name-Null



2. Click the green “Code” button

3. Click the “Download Zip” button



4. Unzip the downloaded file

GitHub Resources

- Downloading zip file: https://www.itprotoday.com/development-techniques-and-management/how-do-i-download-files-github?reg_form=adv
- Cloning a Repository (advanced): https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository-from-github/cloning-a-repository

Preparing the Data

Creating the ROI Template

Introduction

Fixations can be categorized and binned by Regions of Interest (ROI’s). ROI events are defined by presentation and metadata attributes. Presentation attributes contain spatial and temporal display information while metadata attributes contain identifying information. The spatial coordinates of each ROI can either be in cartesian or raster format. For more information on coordinate systems, please see section Coordinate System Conversion.

This template must be created in excel.

ROI Event Attributes

Possible Attributes

The ROI templates allow for the following attributes to be tracked:

- roi_label: The name of an ROI. Used for identification purposes.
- roi_id: A numerical ID of an ROI. Used for identification purposes
- top_left_xy: The top left coordinate of an ROI box.
- bottom_right_xy: The bottom left coordinate of an ROI box
- start: The presentation timing window start in Milliseconds
- stop: The presentation timing window end in Milliseconds

All coordinates must be comma separated, i.e. “x,y”

Required Attributes

The metadata attributes, “roi_label” and “roi_id”, are required for all static and dynamic ROI. These values must be unique.

Optional Attributes

The coordinate (ending in “_xy”) and temporal (“start” and “stop”) attributes are optional. However, at least one pair needs to be filled in for every ROI in order to be correctly paired.

Note: Partially filling out spatial and temporal attributes isn’t currently supported. This has not been tested and may lead to incorrect data pairing.

Static vs Dynamic ROI

A static ROI event appears in the same location and within the same time window for all trials and participants. A dynamic ROI event appears in a different location and/or time window across trials, blocks, or subjects.

Defining Static ROI

For static Roi, metadata and event attributes are listed under the header “static”.

static					
roi_label	roi_id	top_left_xy	bottom_right_xy	start	stop
static_1	1	x0,y0	x1,y1	t0	t1
static_2	2	x2,y2	x3,y3	t2	t3

Defining Dynamic ROI

Dynamic ROI Metadata

In addition the ‘roi_label’ and ‘roi_id’ metadata attributes, the dynamic ROI also contains an “event_key_map_column” section. These values indicate which column of the Dynamic Event Key Map should be associated with their respective dynamic ROI attributes.

dynamic_roi_metadata		
event_key_map_column	roi_label	roi_id
dynamic_1_key	dynamic_1	3
dynamic_2_key	dynamic_2	4

Dynamic Event Keys

Dynamic event keys are associated with unique positions and timing attributes of dynamic ROI's. These keys are used in the dynamic event key map to indicate an ROI's location and timing information.

dynamic_event_options				
key	top_left_xy	bottom_right_xy	start	stop
x	x4,y4	x5,y5	t0	t3
z	x1,y1	x3,y3	t2	t3
y	x2,y2	x4,y4	t1	t2

Dynamic Event Map

In a separate file, a column is created for each dynamic ROI. The column names are paired with their respective ROI labels under the 'dynamic_roi_metadata' section in the ROI template. The values of each column are populated with event keys that are associated with timing and location information. These keys are defined in the 'dynamic_event_options' section of the ROI template.

subject	trial	dynamic_1_key	dynamic_2_key
1201	1	y	x
1201	2	z	y

Building Dynamic ROI

For each column, the program pulls the 'roi_label' and 'roi_id' associated with the column name under the 'dynamic_roi_metadata' section of the ROI template. The program then pulls the spatial coordinates and timing information associated with the column values from the 'dynamic_event_options' section of the ROI template.

Using the tables above, the dynamic ROI for the first trial would be created as:

subject	trail	roi_label	roi_id	top_left_xy	bottom_right_xy	start	stop
1201	1	dynamic_1	3	x2,y2	x4y4	t1	t2
	1	dynamic_2	4	x4,y4	x5,y5	t0	t3

Putting It All Together

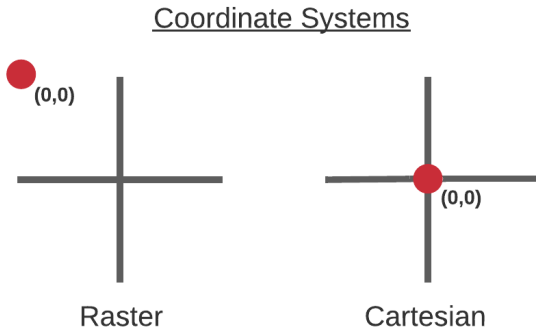
An ROI template with static and dynamic ROI that includes all possible event attributes:

static						dynamic_roi_metadata			dynamic_event_options				
roi_label	roi_id	top_left_xy	bottom_right_xy	start	stop	event_key_map_column	roi_label	roi_id	key	top_left_xy	bottom_right_xy	start	stop
static_1	1	x0,y0	x1,y1	t0	t1	dynamic_1_key	dynamic_1	3	x	x4,y4	x5,y5	t0	t3
static_2	2	x2,y2	x3,y3	t2	t3	dynamic_2_key	dynamic_2	4	z	x1,y1	x3,y3	t2	t3
									Y	x2,y2	x4,y4	t1	t2

Coordinate System Conversion

Overview

A Cartesian coordinate system has the origin (0,0) in the middle of the screen, while a raster system places the origin at the top left of the screen. In the Raster system, all coordinates are positive and the y- axis is inverted. Raster systems are commonly seen in image processing applications. E-prime outputs all fixations in a Raster Coordinate System.



Application Dependencies

The application utilizes coordinates that are in Raster format to determine if fixations fall within an ROI. The user has a choice to record ROI coordinates in the ROI Template either cartesian or raster format. If the coordinates in the ROI Template are in Cartesian format, the user input option of calc_roi_raster_coords needs to be set to True.

Cartesian to Raster Conversion

If we have the cartesian coordinate of x0,y0 and the screen aspect ratio indicating screen width and screen height, then the raster coordinates can be calculated as follows:

raster_x = x0 + (screen width/2) #shifts x above the negative threshold

raster_y = 1024-(y0 + (screen height/2)) #shifts y above negative threshold and inverts axis

Resources:

- Cartesian coordinate system description : https://mathinsight.org/cartesian_coordinates
- Raster coordinate system description: https://en.wikipedia.org/wiki/Raster_graphics

File Naming

When File Naming is Important

File names are used to label imported If more than one matching file is found in the input path directory. A file is considered matching if the extension matches the default type specified for the input variable. See User Input for default values.

Splitting File Names

filenames are split on datatype transition and '-,_' characters.

Ex:

- 1201study_3 -----> 1201, study, 3
- 598-2033carrot ----> 598, 2033, carrot

Metadata

Defining Metadata Keys

Metadata Keys Indicate relevant subject and experiment information that will be pulled from file names and/or behavioral data. These keys are used to label output and aid in pairing Eye-tracking data with Behavioral and ROI data.

Metadata Keys must be separately defined for behavioral, ASC filenames, and ROI Event data if they are to be used for analysis

Extracting Metadata from File names

Metadata is pulled from file names that have been split according to the in The File Naming section.

Dropping Unnecessary Information

If filenames have irrelevant information, the 'drop' keyword can be used to discard groups after splitting.

Ex:

if the metadata keys were defined as: ['Subject', 'drop', 'block']

...and a split file name is: ['1201', 'unnecessary', 'Test']

...the pairing would be: 'Subject': 1201, 'block': 'Test'

Standardizing split file name data

File names are standardized by the rules found in the Automatic Data Cleaning: Strings Section.

Extracting Metadata From DataFrames

If subject Behavioral or ROI Event data is stored in a combined file, the metadata keys must be present as column names.

Automatic Data Cleaning

Paths

Standardizing Paths

String paths are turned into Pathlib objects. These objects ensure that the path is in the correct format for the operating system. In addition, any directories that do not yet exist are created. For more Pathlib attributes, see the resources below.

File Names

If a filename does not split into the required number of groups to be paired with the metadata keys, then the full path will be printed out with the tag: “invalid path” and discarded.

Strings

Raw strings or strings found in dictionaries or lists are converted into a numeric datatype, if possible. If the string is found to contain uppercase alphabetical characters, then they will be lowered.

Tables

Tables are created from CSV and Excel file formats. The first row should contain column names.

String column names and column values are cleaned by the rules above. Any rows containing null values in a metadata columns will be dropped to prevent pairing errors in later analysis

ASC Files

ASC files will be filtered for ‘EFIX’, ‘EBLINK’,’SSAC’, and ‘ESACC’ values. A column labeled ‘trial_id’ will be generated from a cumulative count of each ‘START’ values found in the files

Resources

- Pathlib Library: https://docs.python.org/3/library/pathlib.html

User Input

ROI Options

calc_roi_raster_coords: *boolean*

Boolean: default is False. Set to ‘True’ if roi template coordinates are in the cartesian coordinate system. If set to true, the program will need either the screen aspect_ratio input (below) or an asc_directory_path to pull the screen dimensions from.

aspect_ratio: *tuple*

Should be a tuple of (screen_width, screen_height). Used if calc_raster_coords = True.

If no aspect ratio is provided and ASC files are present, aspect ratios will be extracted from the ASC files. This is especially helpful if participants were run with different aspect ratios

Behavioral Options

behavior_test_path: *string*

String path leading to the behavioral data. This data can include test scores, roi event key maps, and other information that needs to be attached to output. This needs to be a '.xlsx' file.

behavior_test_trial_col: *string*

The name of the trial id column located in the behavior data. This is needed to pair the ASC data with the behavior test data, since a trial id column is automatically created when ASC data is imported.

behavior_metadata_keys: *list of strings*

The column names of the metadata columns found in the behavior data. Should be in list format.

attach_behavior_cols: *list of strings*

Column names that should be attached to all output.

Eye-Tracking Options

asc_directory_path: *string*

String path leading to the asc file directory. Asc files can be nested within the directory and grouped with other files. Any file containing an '.asc' extension flagged for import.

asc_metadata_keys must be provided if an asc directory path is defined

attach_movement_cols: list of strings

Column names that will be attached to all output. Columns are generated from eprime output.

Possible column names are:

- 'file_position'
- 'type'
- 'eye'
- 'block_relative_start'
- 'block_relative_stop'
- 'start'
- 'stop'
- 'duration'
- 'type_count'

asc_metadata_keys: *list of strings*

The labels of the metadata extracted from the ASC filenames. See Metadata section for more information.

Binning Options

time_bin_size: *integer*

The size of the time bins for Response-locked and Stimulus-locked binning analysis. The default bin size is 250 ms.

summary_filter_out: *dictionary*

A dictionary containing column names and values to be excluded when calculating proportion of view binning summary.

Ex: Wanting to exclude rows based on the 'roi_label' column. We don't want to include rows associated with 'scene' and 'whole_screen' roi

Summary_filter_out = {'roi_label': ['scene', 'whole_screen']}

summary_filter_for: *dictionary*

A dictionary containing column names and values to be included when calculating proportion of view binning summary.

ex: Wanting to include rows based on the 'roi_label' column. We want to calculate the proportion of view for the two people roi.

summary_filter_out={'roi_label': ['person_1', 'person_2']}

Entropy Options

calculate_entropy: *boolean*

Indicates whether entropy should be calculated. Will not execute if ROI or ASC inputs are not provided.

target_roi_entropy: *list of strings or integers*

If provided, entropy will only be calculated for the ROI listed. If the list contains strings, the strings should be ROI_labels, matching those found in the ROI template. If the list contains integers, the values should match those found in the 'roi_id' columns of the ROI template.

Exclude_diagonals: *boolean*

Indcates whether within ROI transitions should be considered when calculating transition entropy. Default is False.

Roi Event Map Options

roi_event_map_path: *string*

Takes a directory containing ROI event Key map files. Each file should be in CSV format and labeled with subject block metadata.

roi_event_map_metadata_keys must be provided if an roi event map directory is defined

At this time, all key maps must contain a 'trial_id' column for successful pairing with the Eye-tracking data

roi_event_map_filename_contains: *string*

Used to identify roi event map files. Files will be chosen based on partial string matches

Ex: filenames have 'event_map' in their names like '1201_1_event_map.csv'

Roi_event_map_filename_contains = 'event_map'

roi_event_map_metadata_keys: *list of strings*

Labels for the metadata values that will be extracted from the event map filenames.

ex: ['subject_id' , 'block_id']

roi_event_map_import_skip_rows: *integer or list of integers*

Specifies row(s) to skip while importing the event maps. Integers should be zero indexed, so the first row of a file is 0, the second is 1...

Ex: skip first row because it's blank:

Roi_event_map_import_skip_rows = 0

attach_event_cols: *list of strings*

Columns present in the event maps that need to be attached to all export files.

ex: [‘col_1’ , ‘col_2’]

Export Options

output_directory_path: *string*

A path leading to the desired storage directory for results. Any folders that are specified in the path and don’t exist will automatically be created.

output_folder_name: *string*

The desired name for the output data folder. Default is ‘processed_data’.

Plotting Options

plot_fixations: b*oolean*

A value of True indicates that the fixations should be plotted. If roi locations will be included in the plot if the data is provided.

group_by: *list of strings*

Specifies the column or index names for how the fixations should be grouped together. If an empty list is provided, the fixation metadata keys + the trial_id column will be used to divide fixations.

figure_shape: *tuple of integers*

Indicates the layout of plots for each page with a tuple of (number_rows, number_columns). The number of plots per page is equal to number_rows * number_columns. The default value is (3,2).

Running the Application

Opening the Application

1. Open the pycharm application
2. Click the ‘open’ folder icon
3. Click on the PeyeTracker folder and select ‘OK’
4. Navigate to ‘main.py’ located in ‘./PeyeTracker/PeyeTracker/main.py’

Activating the Virtual Environment

1. Open the main menu of PyCharm
2. Select the ‘File’ dropdown located at the top left corner of the screen
3. Select ‘New Project Setup’ --> ‘Preferences for New Projects...’
4. On the left side of the preferences window, select ‘Python Interpreter’
5. Click on the gear icon next to the ‘python interpreter’ dropdown menu and select ‘Add..’

6. In the left-hand panel, select ‘Conda Environment’
7. Select the ‘Existing environment’ bullet point
8. Inside the ‘Interpreter’ dropdown menu, navigate to anaconda’s python 3.8. interpreter
    a. Mine was located in ‘opt/anaconda3/bin/python3.8’
9. Check the ‘Make available to all projects’ box
10. Select ‘OK’

Specifying Runtime Configuration

1. In the top right corner select ‘add configuration’
2. Double click on ‘add new run configuration’
3. Select ‘python’
4. Specify the configuration name
5. For script path navigate to ‘main.py’
6. Under the ‘execution’ dropdown check ‘run with python console’
7. On the bottom of the window check ‘activate tool window’

Run

1. press the green play button in the upper right-hand corner

Test Data

Example Configuration 1

```
# Export Options
'output_directory_path': None
,'output_folder_name': None
```

Since the output_directory_path and output_folder_name isn’t specified, the default values will be used. The results will be stored in a folder called ‘processed_data’ and this folder will be stored within the app directory.

```
# Eye Movement Options
,'asc_directory_path': 'test_data/asc_files'
,'attach_movement_cols': ['type']
,'asc_metadata_keys': ['subject_id', 'block_id']
,'asc_trial_sets': None
```

The ASC data is located inside the test_data --> asc_files folder. In this example, the *'type'* column of the eye_movements.csv folder (created by the application as an intermediate file) will be attached to all results.

Looking at the individual ASC files, it's apparent that the filenames will be split into two groups. These groups will be used to label the data inside of the files, and we'll use the 'asc_metadata_keys' input to names for the extracted groups.

#Behavior Test Options

,'behavior_test_path': None

,'behavior_test_trial_col': None

,'behavior_metadata_keys': None

,'attach_behavior_cols': None

ROI Event map options are used to specify ROI parameters for each subject, so the behavior test options will be left blank.

Roi Template Options

,'roi_template_path': 'test_data/roi_templates/roi_template_version_1.xlsx'

,'calc_roi_raster_coords': False

,'aspect_ratio': None

The roi template path used in this example is stored in the 'test_data/roi_templates' directory under the file name 'roi_template_version_1.xlsx'. Since the coordinates in the ROI template are already in raster format, we can leave the 'calc_roi_raster_coords' boolean value set to *False* and the 'aspect_ratio' input as *None*.

Roi Event Map Options

,'roi_event_map_path' : 'test_data/roi_event_maps'

,'roi_event_map_metadata_keys' : ['subject_id', 'block_id']

,'roi_event_map_trial_column': 'trial_id'

,'attach_event_cols': ['phase']

,'roi_event_map_filename_contains' : None

,'roi_event_map_import_skip_rows': None

,'roi_event_map_columns' : None

,'add_roi_event_map_trial_id': False

The ROI event map directory is *'test_data/roi_event_maps',* so this path is used for the 'roi_event_map_path' argument.

Since we're reading in multiple files, the filenames will be split and the groups are used to label the file contents. We're interested in pairing the ASC and ROI data, so it's important that some or all the 'roi_event_map_metadata_keys' and 'asc_metadata_keys' match. In this example, the ROI event map filenames are structured in the same manner as the ASC files, so the same metadata keys are defined for both.

For pairing both sets of data, it's important to also define a column name for trial metadata, which should be located inside the ROI event map files. In this case, the trial column is labeled *'trial_id'*, this string value is used as input for the 'roi_event_map_trial_column'.

The 'roi_event_map_filename_contains' argument is used to filter the selected files from the directory, which is useful if the event maps are stored in a folder which contains miscellaneous CSV files. Since the directory path specified contains all files stored in the event map directory, this argument is set to the default parameter of *None.*

The ROI event map files are already cleaned, so there's no need to skip any rows when importing. This means that the 'roi_event_map_import_skip_rows' and 'roi_event_map_columns' arguments are set to *None*.

Since the ROI event maps already contained a trial column, *'trial_id'*, there's no need the program to generate one, so the 'add_roi_event_map_trial_id' column is set to *False.*

Binning Options

,'time_bin_size':250

,'summary_filter_out': None

,'summary_filter_for': None

The 'time_bin_size' is used to specify the frequency at which the data is binned at in milliseconds. In this example, the value is set to *250*. For calculating the proportion of viewing

summary, this example wants to include all ROI specified in the ROI Template, so the ‘summary_filter_out’ and ‘summary_filter_for’ arguments are set to *None*.

```
# Entropy Options
,'calculate_entropy': True
,'target_roi_entropy': None
,'exclude_diagonals': False
```

To generate entropy analysis, the boolean for ‘calculate_entropy’ is set to True. All the ROI should be used for constructing the transition matrices, so the ‘target_roi_entropy’ is set to *None*. The ‘exclude_diagonals’ argument is set to *False* which indicates that inner ROI transitions or self transitions should be included in the entropy calculations as well.