

---

# **CICADA**

***Release 1.0.0***

**Oct 18, 2019**



<b>1</b>	<b>Calcium imaging pipeline</b>	<b>1</b>
<b>2</b>	<b>Contents</b>	<b>3</b>
<b>3</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Python Module Index</b>	<b>31</b>
	<b>Index</b>	<b>33</b>



# CHAPTER 1

---

## Calcium imaging pipeline

---



## 2.1 Dependencies

PyCICADA has the following minimum requirements, which must be installed before you can get started using PyNWB.

1. Python 3.6, or 3.7
2. pip

PyCICADA has been tested on Ubuntu 18.04.1 LTS, Windows 10 and macOS Mojave, using Python 3.6 & 3.7

## 2.2 Installation

### 2.2.1 Install release from PyPI

The Python Package Index (PyPI) is a repository of software for the Python programming language.

To install or update PyCICADA distribution from PyPI simply run:

```
$ pip install -U pycicada
```

This will automatically install the following required dependencies:

1. h5py
2. numpy
3. sortedcontainers
4. shapely
5. qtpy
6. Pillow

7. PyYAML
8. scanimage-tiff-reader
9. pyabf
10. hdf5storage
11. pathlib
12. scipy
13. matplotlib
14. PyQt5

## 2.3 Introduction

CICADA stands for Calcium Imaging Complete Automated Data Analysis.

It's a Python pipeline aimed for analyzing calcium imaging data.

### 2.3.1 Motivation

We notice a lack of toolboxes or analysis pipelines for calcium imaging data, using open source language. Our motivation was to build an easy-to-use pipeline, which doesn't need programming skills. In that purpose, we offer a Graphical User Interface as well as a command line usage.

In order to tackle the broadness of scientists' needs, we built the pipeline to be easily extendable with a plugin-like interface, for data format and analyses.

### 2.3.2 Supported data format

The default data format chosen was [Neurodata Without Borders: Neurophysiology \(NWB:N\)](#). that is a data standard for neurophysiology, providing neuroscientists with a common standard to share, archive, use, and build analysis tools for neurophysiology data. NWB:N is designed to store a variety of neurophysiology data, including data from intracellular and extracellular electrophysiology experiments, data from optical physiology experiments, and tracking and stimulus data.

We provide some tools to convert their data into NWB, with a plugin-like interface. (Curently in Preprocessing)

### 2.3.3 Functionalities

- We offer a Graphical User Interface (GUI) as well as a command line usage.
- We have already implemented or aim to implement various kind of analyses such cell assemblies detection, network analyses, display functions (rasters, cells map) etc. . .
- The GUI offers specifics functionalities: - Subjects and recorded sessions can be filtered or grouped in order to easily select the data to be analysed, and saved for future use. - Subjects and and recorded sessions' informations can be displayed and modified depending on the data format. - Analyses are grouped by family. Each analysis provide a way to check the data see if they are compatible. - Analyses arguments can be saved in a yaml file allowing to easily load them later. - A multi-thread implementation allows multiple analysis at the same time. - A progression bar allows to follow the analysis current status.



## 2.4 Software architecture

### 2.4.1 Analysis

Each analysis is represented by a class that inherits CicadaAnalysis (in the module `cicada.analysis`).

Each CicadaAnalysis instance instantiate an ArgumentAnalysisHandler that communicates with the GUI and handle the arguments that will be passed to the analysis.

Each CicadaAnalysis instance allows to check the compatibility of the data to analyze though the method `check_data()`.

The method `run_analysis()` will be called to start the analysis.

The abstract class CicadaAnalysisFormatWrapper allows,through inheritance, to write a wrapper for a given data format, such as nwb. It allows to get the content from specific data format, and adding a new format consists in creating a new instance of CicadaAnalysisFormatWrapper, without any change in the CicadaAnalysis instances.

### 2.4.2 Graphical User Interface

The GUI uses Qt through QtPy which wrapps PyQt5 and PySide.

The GUI is composed of 3 main modules.

1. A list that displays the loaded recorded sessions to analyse (`SessionsWidget`).
2. A tree that displays the available analyses, updated depending on the data to analyse. (`AnalysisTreeApp`)
3. A pop-up panel that allows to set arguments for the given analysis and to follow the status of the analysis. (`AnalysisParametersApp`). A corresponding overview of the opened panels is available on the main window (`AnalysisOverview`)

## 2.5 How to use

To be added

## 2.6 Pre-processing

Convert CI data to NWB file format

### 2.6.1 Data to NWB

```
cicada.preprocessing.cicada_data_to_nwb.convert_data_to_nwb(data_to_convert_dir,
                                                             de-
                                                             fault_convert_to_nwb_yaml_file,
                                                             nwb_files_dir)
```

Convert all default\_config\_data\_for\_conversion located in dir\_path and put it in NWB format then create the file. Use the yaml file contains in dir\_path to convert the default\_config\_data\_for\_conversion. A yaml file with in its name session\_data and one with subject\_data must be in directory. Otherwise nothing will happend. A yaml file with abf in its name will need to be present to convert the abf default\_config\_data\_for\_conversion.

#### Parameters

- **data\_to\_convert\_dir** (*str*) – Absolute path to the directory containing all data
- **default\_convert\_to\_nwb\_yaml\_file** (*str*) – Absolute path to the default YAML file to convert an NWB file
- **nwb\_files\_dir** (*str*) – Absolute path to the directory where to save the nwb file created

```
cicada.preprocessing.cicada_data_to_nwb.create_convert_class(class_name,  
                                                             config_dict,  
                                                             converter_dict,  
                                                             nwb_file,  
                                                             yaml_path,    files,  
                                                             dir_path)
```

#### Parameters

- **class\_name** –
- **config\_dict** –
- **converter\_dict** –
- **nwb\_file** –
- **yaml\_path** –
- **files** –
- **dir\_path** –

Returns:

```
cicada.preprocessing.cicada_data_to_nwb.create_nwb_file(subject_data_yaml_file,  
                                                         session_data_yaml_file)
```

Create an NWB file object using all metadata containing in YAML file

#### Parameters

- **subject\_data\_yaml\_file** (*str*) – Absolute path to YAML file containing the subject metadata
- **session\_data\_yaml\_file** (*str*) – Absolute path to YAML file containing the session metadata

```
cicada.preprocessing.cicada_data_to_nwb.filter_list_according_to_keywords(list_to_filter,  
                                                                           key-  
                                                                           words,  
                                                                           key-  
                                                                           words_to_exclude)
```

Conditional loop to remove all files or directories not containing the keywords # or containing excluded key-words. Inplace list modification

#### Parameters

- **list\_to\_filter** (*list*) – List containing all files/directories to be filtered
- **keywords** (*str*) – If the list doesn't contain the keyword, remove it from list
- **keywords\_to\_exclude** (*str*) – If the list contains the keyword, remove it from list

Examples:

```
>>> print(filter_list_of_files(["file1.py", "file2.c", "file2.h"], "2", "h"))  
["file2.c"]
```

```
cicada.preprocessing.cicada_data_to_nwb.filter_list_of_files(dir_path, files,
                                                           extensions, directory=None)
```

Take a list of file names and either no extensions (empty list or None) and remove the directory that starts by “.” or a list of extension and remove the files that are not with this extension. It returns a new list

#### Parameters

- **dir\_path** (*str*) – path in which the files are
- **files** (*list*) – List of files to be filtered
- **extensions** (*str*) – File extension to use as a filter
- **directory** (*str*) – directory in which looking for files with a given extensions or return files in this directory

#### Exemples:

```
>>> print(filter_list_of_files(["file1.py", "file2.c", "file3.h"], "py"))
["file1.py"]
```

## 2.6.2 Run preprocessing

### 2.6.3 NWB file class

```
class cicada.preprocessing.convert_to_nwb.ConvertToNWB(nwb_file)
    NWB file object class
```

```
convert (**kwargs)
    Convert the data and add to the nwb_file
```

**Parameters** *\*\*kwargs* – arbitrary arguments

### 2.6.4 Suite 2P ROIs

```
class cicada.preprocessing.convert_suite_2p_rois_to_nwb.ConvertSuite2pRoisToNWB(nwb_file)
    Class to convert ROIs data from Suite2P to NWB if raw_traces from Suite2p are available they are loaded.
    Otherwise if the movie is available, build the raw_traces. create_roi_response_series
```

```
convert (**kwargs)
    Convert the data and add to the nwb_file
```

**Parameters** *\*\*kwargs* – arbitrary arguments

### 2.6.5 2D series

```
class cicada.preprocessing.convert_processed_2d_series_to_nwb.ConvertProcessed2dSeriesToNWB(nwb_file)
    Class to convert 2D series to NWB
```

```
convert (**kwargs)
    Convert the data and add to the nwb_file
```

**Parameters** *\*\*kwargs* – arbitrary arguments

```
static load_rasterplot_in_memory (rasterplot_file_name,          matlab_string,  
                                frames_to_add=None)
```

Get 2D series data from file

**Parameters**

- **rasterplot\_file\_name** (*str*) – Absolute path to file containing 2D series
- **matlab\_string** (*str*) – Key to the data from matlab and npz files
- **frames\_to\_add** (*dict*) – Key is the frame where you add blank frames and value is the number of frames to add
- **is** *None*. (*Default*) –

**Returns** Raster data as a 2d array

**Return type** raster (np.array)

## 2.6.6 Calcium imaging movie

```
class cicada.preprocessing.convert_ci_movie_to_nwb.ConvertCiMovieToNWB (nwb_file)  
    Class to convert Calcium Imaging movies to NWB
```

```
convert (**kwargs)
```

Convert the data and add to the nwb\_file

**Parameters** **\*\*kwargs** – arbitrary arguments

## 2.6.7 ABF

```
class cicada.preprocessing.convert_abf_to_nwb.ConvertAbfToNWB (nwb_file)  
    Class to convert ABF data to NWB
```

```
convert (**kwargs)
```

The goal of this function is to extract from an Axon Binary Format (ABF) file its content and make it accessible through the NWB file. The content can be: LFP signal, piezzo signal, speed of the animal on the treadmill. All, None or a few of these informations could be available. One information always present is the timestamps, at the abf sampling\_rate, of the frames acquired by the microscope to create the calcium imaging movie. Such movie could be the concatenation of a few movies, such is the case if the movie need to be saved every x frames for memory issue for ex. If the movie is the concatenation of many, then there is an option to choose to extract the information as if 2 frames concatenate are contiguous in times (such as then LFP signal or piezzo would be match movie), or to add interval\_times indicating at which time the recording is on pause and at which time it's starting again. The time interval containing this information is named "ci\_recording\_on\_pause" and you can get it doing: if 'ci\_recording\_on\_pause' in nwb\_file.intervals: pause\_intervals = nwb\_file.intervals['ci\_recording\_on\_pause']

**Parameters**

- **\*\*kwargs** (*dict*) – kwargs is a dictionary, potentials keys and values types are:
- **abf\_yaml\_file\_name** – mandatory parameter. The value is a string representing the path
- **file\_name of the yaml file associated to this abf file. In the abf** (*and*) –
- **frames\_channel** – mandatory parameter. The value is an int representing the channel
- **the abf in which is the frames timestamps data.** (*of*) –

- **abf\_file\_name** – mandatory parameter. The value is a string representing the path
- **file\_name of the abf file.** (*and*) –

**detect\_run\_periods** (*run\_data*, *min\_speed*)

Using the data from the abf regarding the speed of the animal on the treadmill, return the speed in cm/s at each timestamps as well as period when the animal is moving (using *min\_speed* threshold)

**Parameters**

- **run\_data** (*list*) – Data from the subject run
- **min\_speed** (*int*) – Minimum speed

**Returns** List of movements periods *speed\_during\_movement\_periods* (*list*) : List of subject speed during movements *speed\_by\_time* (*list*) : List of subject speed by time

**Return type** *mvt\_periods* (*list*)

**determine\_ci\_frames\_indices** ()

Using the frames data channel, estimate the timestamps of each frame of the calcium imaging movie. If there are breaks between each recording (the movie being a concatenation of different movies), then there is an option to either skip those non registered frames that will be skipped in all other data (lfp, piezzo, ...) or to determine how many frames to add in the movie and where so it matches the other data recording in the abf file

**process\_run\_data** (*run\_channel*)

Using the information in *run\_channel*, will add to the *nwb\_file* the speed of the subject at each acquisition frame of the movie in cm/s

**Parameters** **run\_channel** (*int*) – Run channel

## 2.6.8 Utils

**class** *cicada.preprocessing.utils.ComparableItem* (*value*)

Make it possible to sort a list of items of different types, such as int and string

*cicada.preprocessing.utils.class\_name\_to\_file\_name* (*class\_name*)

Transform the string representing a *class\_name*, by removing the upper case letters, and inserting before them an underscore if 2 upper case letters don't follow. Underscore are also inserted before numbers ex: ConvertAbfToNWB -> convert\_abf\_to\_nwb :param *class\_name*: string :return:

*cicada.preprocessing.utils.flatten* (*list*)

Flatten a nested list no matter the nesting level

**Parameters** **list** (*list*) – List to flatten

**Returns** List without nest

### Examples

```
>>> flatten([1,2,[[3,4],5],[7]])
[1,2,3,4,5,7]
```

*cicada.preprocessing.utils.get\_continuous\_time\_periods* (*binary\_array*)

take a binary array and return a list of tuples representing the first and last position(included) of continuous positive period :param *binary\_array*: :return:

```
cicada.preprocessing.utils.load_tiff_movie_in_memory(tif_movie_file_name,  
                                                    frames_to_add=None)
```

Load tiff movie from filename using Scan Image Tiff

**Parameters** `tif_movie_file_name` (*str*) – Absolute path to tiff movie

**Returns** Tiff movie as 3D-array

**Return type** tiff\_movie (array)

```
cicada.preprocessing.utils.load_tiff_movie_in_memory_using_pil(tif_movie_file_name,  
                                                             frames_to_add=None)
```

Load tiff movie from filename using PIL library

**Parameters**

- `tif_movie_file_name` (*str*) – Absolute path to tiff movie
- `frames_to_add` – dict with key an int representing the frame index after which add frames. the value is the number of frames to add (integer)

**Returns** Tiff movie as 3D-array

**Return type** tiff\_movie (array)

```
cicada.preprocessing.utils.merging_time_periods(time_periods,  
                                                min_time_between_periods)
```

Take a list of pair of values representing intervals (periods) and a merging threshold represented by `min_time_between_periods`. If the time between 2 periods are under this threshold, then we merge those periods. It returns a new list of periods. :param `time_periods`: list of list of 2 integers or floats. The second value represent the end of the period, the value being included in the period. :param `min_time_between_periods`: a float or integer value :return: a list of pair of list.

```
cicada.preprocessing.utils.sort_by_param(nwb_path_list, param_list)
```

Sort NWB files depending on a list of parameters

**Parameters**

- `nwb_path_list` (*list*) – List of absolute path to NWB files
- `param_list` (*list*) – List of parameters to sort by

**Returns** List of NWB files sorted

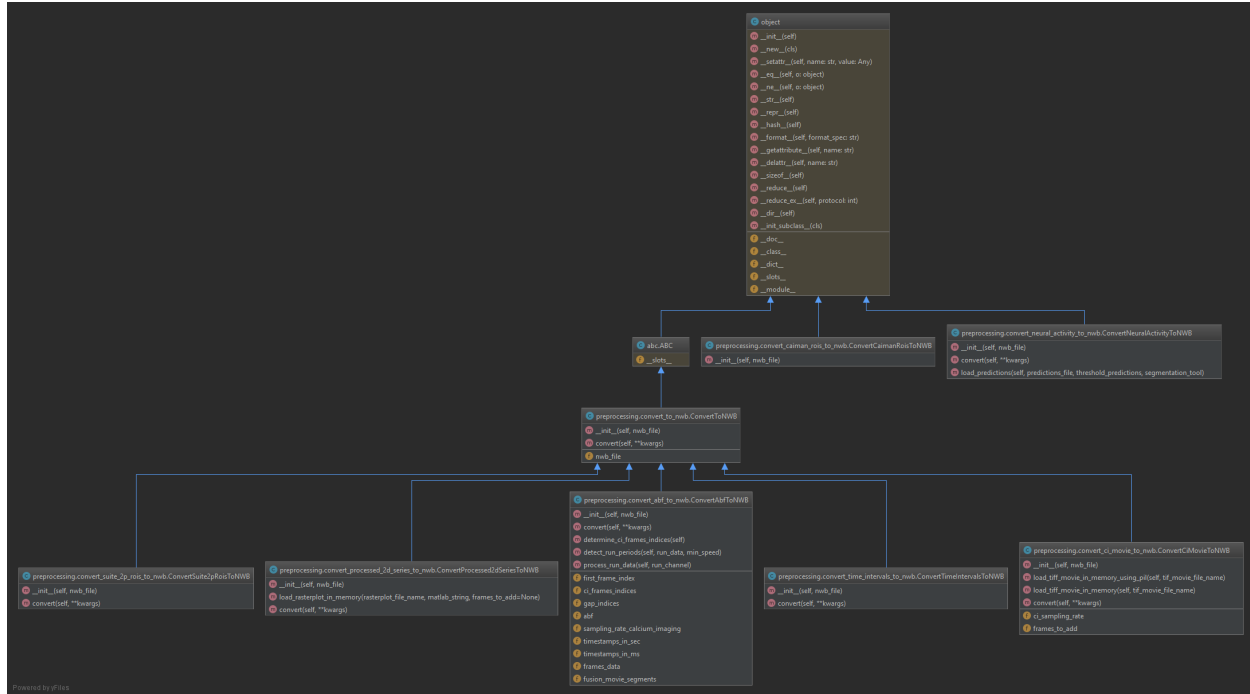
**Return type** nwb\_sorted\_list (*list*)

```
cicada.preprocessing.utils.update_frames_to_add(frames_to_add, nwb_file,  
                                                ci_sampling_rate)
```

Update `frames_to_add` (dict), based on `pause_intervals` and `ci_frames_time_series` :param `frames_to_add`: dict, with key an int representing the frame index after which add frames. :param the value is the number of frames to add: :type the value is the number of frames to add: integer :param `nwb_file`: nwb file, will get `nwb_file.intervals['ci_recording_on_pause']` and :param `nwb_file.get_acquisition`: :type `nwb_file.get_acquisition`: “ci\_frames”

Returns:

## 2.6.9 Appendix



## 2.7 Analysis

Analysises

### 2.7.1 Main analysis class

```
class cicada.analysis.cicada_analysis.CicadaAnalysis (name, short_description,
family_id=None,
long_description=None,
data_to_analyse=None,
data_format=None, config_handler=None, gui=True)
```

An abstract class that should be inherit in order to create a specific analyse

**add\_argument\_for\_gui** (with\_incremental\_order=True, \*\*kwargs)

**Parameters**

- **\*\*kwargs** –
- **with\_incremental\_order** – boolean, if True means the order of the argument will be the same as when added

Returns:

**add\_ci\_movie\_arg\_for\_gui** (long\_description=None)

Will add an argument for gui, named ci\_movie that will list all calcium imaging available for each session

Returns:

**add\_segmentation\_arg\_for\_gui()**

Will add an argument for gui, named segmentation that will list all segmentations available for each session  
Returns:

**check\_data()**

Check the data given one initiating the class and return True if the data given allows the analysis implemented, False otherwise. :return: a boolean

**create\_results\_directory(dir\_path)**

Will create a directory in dir\_path with the name of analysis and time at which the directory is created so it can be unique. The attribute \_results\_path will be updated with the path of this new directory :param dir\_path: path of the dir in which create the results dir

Returns: this new directory

**get\_data\_identifiers()**

Return a list of string representing each data to analyse Returns:

**get\_data\_to\_analyse()**

**Returns** a list of the data to analyse

**get\_results\_path()**

Return the path when the results from the analysis will be saved or None if it doesn't exist yet Returns:

**run\_analysis(\*\*kwargs)**

Run the analysis :param kwargs: :return:

**set\_arguments\_for\_gui()**

Need to be implemented in order to be used through the graphical interface. super().set\_arguments\_for\_gui() should be call first to instantiate an AnalysisArgumentsHandler and create the attribution for results\_path :return: None

**set\_data(data\_to\_analyse, data\_format='nwb')**

A list of :param data\_to\_analyse: list of data\_structure :param data\_format: indicate the type of data structure. for NWB, NIX

**update\_original\_data()**

To be called if the data to analyse should be updated after the analysis has been run. :return: boolean: return True if the data has been modified

**update\_progressbar(time\_started, increment\_value=0, new\_set\_value=0)**

**Parameters**

- **time\_started** (*float*) – Start time of the analysis
- **increment\_value** (*float*) – Value that should be added to the current value of the progress bar
- **new\_set\_value** (*float*) – Value that should be set as the current value of the progress bar

## 2.7.2 Argument handler

**class** cicada.analysis.cicada\_analysis\_arguments\_handler.**AnalysisArgumentsHandler**(cicada\_analys

Handle the AnalysisArgument instances for a given CicadaAnalysis instance. Allows to create the widgets and get the values to pass to run\_analysis() of the CicadaAnalysis instance.

**add\_argument(\*\*kwargs)**

**Parameters** **\*\*kwargs** –



Returns:

**check\_arguments\_validity** ()

Check if all mandatory arguments have been filled Returns: True if we can run the analysis

**get\_analysis\_argument** (*arg\_name*)

Parameters *arg\_name* –

Returns:

**get\_analysis\_arguments** (*sorted=False*)

Parameters *sorted* –

Returns:

**get\_gui\_widgets** (*group\_by\_family=False*)

Get the list of widgets necessary to fill the arguments for the cicada analysis associated :param group\_by\_family: if True, group the widgets in one widget to be grouped together if they belong to the same :param family. AnalysisArgument will have an attribute named family\_widget whose value is a string.:

Returns:

**load\_analysis\_argument\_from\_yaml\_file** (*file\_name*)

Set the analysis argument value based on the value in the yaml file. The :param file\_name:

Returns:

**save\_analysis\_arguments\_to\_yaml\_file** (*path\_dir, yaml\_file\_name*)

Save the arguments value to a yaml file. The first key will represent the argument name then the value will be a dict with the argument details such as the type etc... :param path\_dir: directory in which save the yaml file :param yaml\_file\_name: yaml file name, with the extension or without (will be added in that case)

Returns:

**set\_argument\_value** (*arg\_name, \*\*kwargs*)

Set an argument values, will be use to run analysis :param arg\_name: :param \*\*kwargs:

Returns:

**set\_widgets\_to\_default\_value** ()

Set the widgets to the default value of their AnalysisArgument Returns:

## 2.7.3 Format wrapper

**class** cicada.analysis.cicada\_analysis\_format\_wrapper.**CicadaAnalysisFormatWrapper** (*data\_ref,*  
*data\_format,*  
*load\_data=Tr*

An abstract class that should be inherit in order to create a specific format wrapper

**age**

Age of the subject :return: None if age unknown

**contains\_ci\_movie** (*consider\_only\_2\_photons*)

Indicate if the data object contains at least one calcium imaging movie :param consider\_only\_2\_photons: boolean, it True means we consider only 2 photons calcium imaging movies, :param if other exists but not 2 photons, then False will be return:

Returns: True if it's the case, False otherwise

**genotype**

Genotype of the subject :return: None if age unknown

**get\_ci\_movies** (*only\_2\_photons*)

Return a dict with as key a string identifying the movie, and as value a dict of CI movies a string as file\_name if external, or a 3d array :param only\_2\_photons: return only the 2 photon movies

Returns:

**get\_identifier** (*session\_data*)

Get the identifier of one of the data to analyse :param session\_data: Data we want to know the identifier

Returns: A hashable object identifying the data

**get\_intervals\_names** ()

Return a list representing the intervals contains in this data Returns:

**get\_pixel\_mask** (*segmentation\_info*)

Return pixel\_mask which is a list of list of pair of integers representing the pixels coordinate (x, y) for each cell. the list length is the same as the number of cells. :param segmentation\_info: object (could be list, dict etc...) given information about how to reach the pixel\_mask :param data:

Returns:

**get\_roi\_response\_series** ()

Returns: a list or dict of objects representing all roi response series (rrs) names rrs could represents raw traces, or binary raster, and its link to a given segmentation. The results returned should allow to identify the segmentation associated. Object could be strings, or a list of strings, that identify a rrs and give information how to get there.

**get\_segmentations** ()

Returns: a list or dict of objects representing all segmentation names up the segmentation planes (like in nwb) Object could be strings, or a list of strings, that identify a segmentation and give information how to get there.

**identifier**

Identifier of the session :return:

**load\_data** ()

Load data in memory Returns:

**sex**

Sex (gender) of the subject :return: None if sex unknown

**species**

Species of the subject :return: None if age unknown

**subject\_id**

Id of the subject :return: None if age unknown

**weight**

Id of the subject :return: None if age unknown

## 2.7.4 NWB wrapper

```
class cicada.analysis.cicada_analysis_nwb_wrapper.CicadaAnalysisNwbWrapper(data_ref,  
                                                                    load_data=True)
```

Allows to communicate with the nwb format

**age**

Age of the subject :return: None if age unknown

**contains\_ci\_movie** (*consider\_only\_2\_photons*)

Indicate if the data object contains at least one calcium imaging movie represented by an instance of `pynwb.image.ImageSeries` :param *consider\_only\_2\_photons*: boolean, if True means we consider only 2 photons calcium imaging movies, :param if other exists but not 2 photons, then False will be return:

Returns: True if it's the case, False otherwise

**genotype**

Genotype of the subject :return: None if age unknown

**get\_ci\_movies** (*only\_2\_photons*)

Return a dict with as key a string identifying the movie, and as value a dict of CI movies a string as *file\_name* if external, or a 3d array :param *only\_2\_photons*: return only the 2 photon movies

Returns:

**get\_identifier** (*session\_data*)

Get the identifier of one of the data to analyse :param *session\_data*: Data we want to know the identifier

Returns: A hashable object identifying the data

**get\_intervals\_names** ()

Return a list representing the intervals contains in this data Returns:

**get\_pixel\_mask** (*segmentation\_info*)

Return *pixel\_mask* which is a list of list of pair of integers representing the pixels coordinate (x, y) for each cell. the list length is the same as the number of cells. :param *segmentation\_info*: a list of 3 elements: first one being the name of the module, then the name :param of *image\_segmentation* and then the name of the segmentation plane.:

Returns:

**get\_roi\_response\_series** ()

Returns: a list or dict of objects representing all roi response series (rrs) names rrs could represents raw traces, or binary raster, and its link to a given segmentation. The results returned should allow to identify the segmentation associated. Object could be strings, or a list of strings, that identify a rrs and give information how to get there.

**get\_segmentations** ()

Returns: a dict that for each step till *plane\_segmentation* represents the different option. First dict will have as keys the name of the modules, then for each modules the value will be a new dict with keys the *ImageSegmentation* names and then the value will be a list representing the segmentation plane

**identifier**

Identifier of the session :return:

**load\_data** ()

Load data in memory Returns:

**sex**

Sex (gender) of the subject :return: None if sex unknown

**species**

Species of the subject :return: None if age unknown

**subject\_id**

Id of the subject :return: None if *subject\_id* unknown

**weight**

Id of the subject :return: None if weight unknown

## 2.7.5 Cells count

## 2.7.6 Connectivity graph

```
class cicada.analysis.cicada_connectivity_graph.CicadaConnectivityGraph
```

```
    check_data()
```

Check the data given one initiating the class and return True if the data given allows the analysis implemented, False otherwise. :return: a boolean

```
    run_analysis(**kwargs)
```

test :param kwargs: :return:

```
    update_original_data()
```

To be called if the data to analyse should be updated after the analysis has been run. :return: boolean: return True if the data has been modified

## 2.7.7 Frames count

## 2.7.8 Hubs analysis

```
class cicada.analysis.cicada_hubs_analysis.CicadaHubsAnalysis
```

```
    check_data()
```

Check the data given one initiating the class and return True if the data given allows the analysis implemented, False otherwise. :return: a boolean

```
    run_analysis(**kwargs)
```

test :param kwargs: :return:

```
    update_original_data()
```

To be called if the data to analyse should be updated after the analysis has been run. :return: boolean: return True if the data has been modified

## 2.7.9 PSTH analysis

```
class cicada.analysis.cicada_psth_analysis.CicadaPsthAnalysis
```

```
    check_data()
```

Check the data given one initiating the class and return True if the data given allows the analysis implemented, False otherwise. :return: a boolean

```
    run_analysis(**kwargs)
```

test :param kwargs: :return:

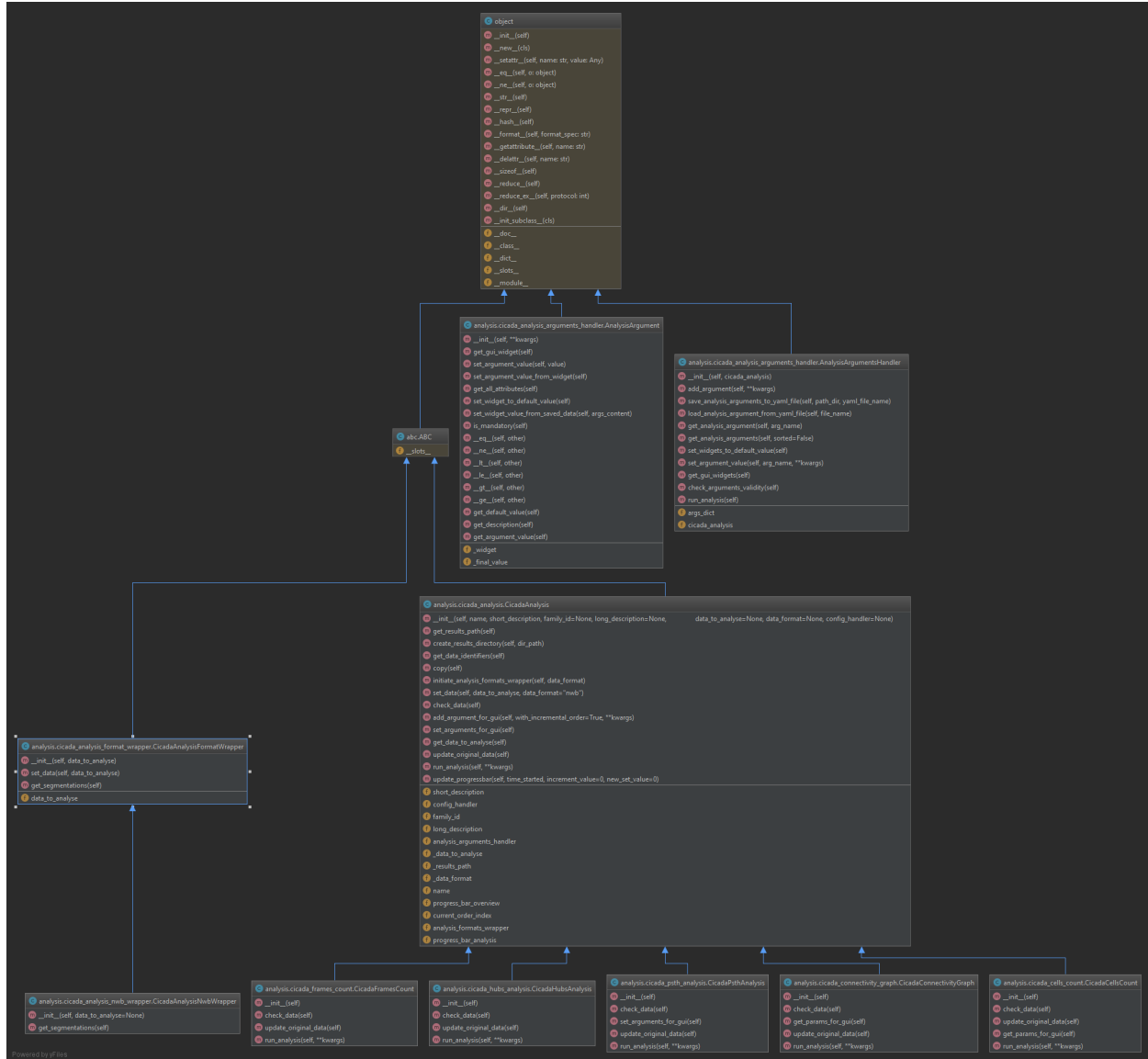
```
    set_arguments_for_gui()
```

Returns:

```
    update_original_data()
```

To be called if the data to analyse should be updated after the analysis has been run. :return: boolean: return True if the data has been modified

## 2.7.10 Annexe



## 2.8 GUI

Graphic interface to launch analyses

### 2.8.1 Main Window

**class** cicada.gui.cicada\_main\_window.CicadaMainWindow(*config\_handler*)

Main window of the GUI

**about** ()

Small about QMessageBox for the project

**add\_data()**

Open a directory

**add\_group\_data(group\_name)**

Add a group of saved sessions to the current list of session

**Parameters** **group\_name** (*str*) – Name of the group saved in YAML

**closeEvent(event)**

Close all analyses windows on main window close

**createActions()**

Create some QAction

**createMenus()**

Create menu bar and put some menu in it

**create\_group\_menu()**

Create group menu

**create\_sort\_menu()**

Create sort menu

**load\_data\_from\_config()**

Check if the last dir opened is saved in config and load it automatically

**load\_data\_from\_dir(dir\_name, method)**

Load data (currently only NWB) from selected directory

**Parameters**

- **dir\_name** (*str*) – Path to data
- **method** (*str*) – String to choose whether to add data to the existing dataset or open a new one, takes two values : 'add' or 'clear'

**load\_group(group\_name)**

Load a group of saved sessions, it will clear the current session list

**Parameters** **group\_name** (*str*) – Name of the group saved in YAML

**load\_group\_from\_config()**

Load groups from a YAML file in the config folder

**on\_group(param, state)**

Give group list and parameters value to populate QListWidget

**Parameters**

- **param** (*str*) – Parameter to group by
- **state** (*int*) – State of the checkbox

**on\_sort(param, state)**

Give sorted list to populate QListWidget :param param: Parameter to sort by :type param: str :param state: State of the checkbox :type state: int

**openWindow()**

Open all widgets in a CentralWidget and call some menus that needed those widgets

**open\_new\_dataset()**

Open a directory

**populate\_menu()**

Populate the menu to load groups

**save\_last\_data\_location** (*dir\_name*)

Keep path to last data directory selected in a YAML in config

**Parameters** *dir\_name* (*str*) – Path to data to be saved

**see\_all\_groups** ()

Display a widget with all existing groups

**select\_group** (*group\_name*)

Select all sessions of a group

**Parameters** *group\_name* (*str*) – Name of the group saved in YAML

**uncheck\_all\_sort** ()

Uncheck all checkboxes in sort menu

**uncheck\_group** (*param*=")

Uncheck group menu parameter

**Parameters** *param* (*str*) – Parameter name to uncheck

**class** `cicada.gui.cicada_main_window.MusketeersWidget` (*config\_handler*, *ci-*  
*cada\_main\_window=None*)

Gather in a layout the 3 main sub-windows composing the gui: displaying the subject sessions, the analysis tree and an overview of the running analysis

## 2.8.2 Filter/group sessions

## 2.8.3 Display metadata

## 2.8.4 Analyses list

**class** `cicada.gui.cicada_analysis_tree_gui.AnalysisTreeApp` (*parent*, *config\_handler*,  
*to\_parameters\_button=None*)

**create\_tree\_model** ()

Create the tree model Returns: the tree model, an instance of QAnalysisTreeModel

**doubleClickedItem** (*idx*)

Method called when the user double click in the tree :param idx: Index of the branch clicked

Returns:

**init\_ui** ()

Set some of the elements used to build the tree Returns:

**invalidate\_all\_items** ()

invalidate all items if the tree Returns:

**load\_arguments\_parameters\_section** ()

Used to load the parameters section with widgets, based on the current selection in the tree. If the selection is not on any valid tree item, nothing will happen Returns: None

**set\_data** (*data\_to\_analyse*, *data\_format*)

Give to the tree the data that the analysis classes will be given to analyze. Allows the tree to inactivate the analyses for which the data don't fulfill the requirements :param data\_to\_analyse: a list of data in a given format (could be nwb or other) :param data\_format: format of the data, must be a string. So far only "nwb" is supported

Returns: None

```
class cicada.gui.cicada_analysis_tree_gui.QAnalysisTreeModel (tree_item,      par-  
                                                         ent=None)
```

```
    columnCount (self, parent: QModelIndex = QModelIndex()) → int
```

```
    data (index, role)
```

**Parameters**

- **index** –
- **role** –

**Returns**

```
    flags (self, QModelIndex) → Qt.ItemFlags
```

```
    headerData (self, int, Qt.Orientation, role: int = Qt.DisplayRole) → Any
```

```
    index (self, int, int, parent: QModelIndex = QModelIndex()) → QModelIndex
```

```
    parent (self, QModelIndex) → QModelIndex  
    parent(self) -> QObject
```

```
    rowCount (self, parent: QModelIndex = QModelIndex()) → int
```

```
class cicada.gui.cicada_analysis_tree_gui.QAnalysisTreeView (tree_item,      con-  
                                                         fig_handler, par-  
                                                         ent=None)
```

```
    drawBranches (self, QPainter, QRect, QModelIndex)
```

```
    isIndexHidden (q_model_index)
```

Avoid to select the line that display the family name :param q\_model\_index: :return:

```
    keyPressEvent (event)
```

**Parameters** **event** – P -> set background picture

Returns:

```
cicada.gui.cicada_analysis_tree_gui.fill_tree_item_with_dict (root_tree,      in-  
                                                         stances_dict)
```

Recursive function that fills the root\_tree according to data in instances\_dict. :param root\_tree: instance of TreeItem :param instances\_dict: Contains instance of cicada\_analysis, in a hierarchy similar to the one we want the tree to be :return:

## 2.8.5 Overview of analyses

```
class cicada.gui.cicada_analysis_overview.AnalysisOverview (config_handler, par-  
                                                         ent=None)
```

Class containing the overview linked to an analysis

```
    add_analysis_overview (cicada_analysis, analysis_id, obj)
```

Add widgets to track the corresponding analysis :param cicada\_analysis: CicadaAnalysis instance :type cicada\_analysis: CicadaAnalysis :param analysis\_id: Randomly generated ID linked to the analysis :type analysis\_id: str :param obj: The analysis window's object itself :type obj: object

```
    keyPressEvent (self, QKeyEvent)
```



```
class cicada.gui.cicada_analysis_overview.AnalysisState (analysis_id,          ci-  
                                                    cada_analysis,      par-  
                                                    ent=None,           with-  
                                                    out_bringing_to_front=False)
```

Class containing the name of the analysis and the subjects analysed

```
bring_to_front (window_id, event)
```

Bring corresponding analysis window to the front (re-routed from the double click method)

#### Parameters

- **window\_id** (*QWidget*) – Analysis Widget object
- **event** (*QEvent*) – Double click event

```
deleteLater ()
```

Re-implementation of the deleteLater method to properly delete the whole widget

```
class cicada.gui.cicada_analysis_overview.ResultsButton (cicada_analysis)
```

Class containing the button to open the result folder

```
deleteLater ()
```

Re-implementation of the deleteLater method to properly delete the widget

```
open_explorer ()
```

Open the file explorer depending on the OS

## 2.8.6 Analysis parameters

```
class cicada.gui.cicada_analysis_parameters_gui.AnalysisData (cicada_analysis,  
                                                                argu-  
                                                                ments_section_widget,  
                                                                config_handler,  
                                                                parent=None)
```

```
keyPressEvent (event)
```

Parameters **event** – P -> set background picture

Returns:

```
populate_session_list (session_list)
```

Add all session to the QListWidget :param session\_list: List of all sessions' identifier :type session\_list: list

```
class cicada.gui.cicada_analysis_parameters_gui.AnalysisPackage (cicada_analysis,  
                                                                analy-  
                                                                sis_name,  
                                                                name,  
                                                                main_window,  
                                                                con-  
                                                                fig_handler,  
                                                                parent=None)
```

Widget containing the whole analysis window

```
bring_to_front (window_id, event)
```

Bring corresponding window to the front (re-routed from the double click method)

#### Parameters

- **window\_id** (*QWidget*) – Analysis Widget object

- **event** (*QEvent*) – Double click event

**errOutputWritten** (*text, path*)

Append std.err text to the QLabel and create an err file.

**Parameters**

- **text** (*str*) – Output of the standard output in python interpreter
- **path** (*str*) – path where we will output the err file

**normalOutputWritten** (*text, path*)

Append std.out text to the QLabel and create a log file.

**Parameters**

- **text** (*str*) – Output of the standard output in python interpreter
- **path** (*str*) – path where we will output the log file

**on\_close** (*event*)

Check if an analysis is still on going and prompt the user to let him know then ask whether he still wants to close. If yes, delete the associated overview and stop the thread

**Parameters event** (*QEvent*) – Qt Event triggered when attempting to close the window

```
class cicada.gui.cicada_analysis_parameters_gui.AnalysisParametersApp (thread_name,  
                                                                    progress_bar,  
                                                                    anal-  
                                                                    y-  
                                                                    sis_name,  
                                                                    con-  
                                                                    fig_handler,  
                                                                    par-  
                                                                    ent=None)
```

Class containing the parameters widgets

**create\_widgets** (*cicada\_analysis*)

**Parameters cicada\_analysis** (*CicadaAnalysis*) – Chosen analysis

**keyPressEvent** (*event*)

**Parameters event** – P -> set background picture

Returns:

**load\_arguments** ()

Will open a FileDialog to select a yaml file used to load arguments used for a previous analysis

**reset\_arguments** ()

Reset all arguments to default value

**run\_analysis** ()

Check if the parameters are valid and then create a thread which will run the analysis

**save\_yaml\_with\_name** ()

Save parameters as a YAML file under the name given by the user. The path is retrieved from the config file and if it doesn't exist a QFileDialog will be displayed to select the path

**tabula\_rasa** ()

Erase the widgets and make an empty section

---

```

class cicada.gui.cicada_analysis_parameters_gui.CheckBoxWidget (analysis_arg,
                                                                parent=None)

    Used to set a boolean value

    get_value ()
        Return the value of the widget Returns:

    set_value (value)
        Set the widget value to the value passed Returns:

    to_stretch ()
        Indicate if the widget should take all the space of a horizontal layout how might share the space with
        another widget Returns: Boolean

class cicada.gui.cicada_analysis_parameters_gui.ColorDialogWidget (analysis_arg,
                                                                show_alpha_channel,
                                                                parent=None)

    Widget used to select a color

    get_value ()
        Returns: a tuple of 4 floats representing RGBA with values from 0.0 to 1.0

    open_dialog ()
        Open the color dialog Returns:

    set_value (value)

        Parameters value – a list or tuple of 3 or 4 float between 0.0 to 1.0, RGB or RGBA values,
        the A representing the alpha

        Returns:

    to_stretch ()
        Indicate if the widget should take all the space of a horizontal layout how might share the space with
        another widget Returns: Boolean

    update_button_color ()
        Returns:

class cicada.gui.cicada_analysis_parameters_gui.ComboBoxWidget (analysis_arg,
                                                                parent=None)

    add_multiple_combo_boxes (session_id, choices_dict, legends, index)
        Allows to add multiple combo boxes, each changing the content of the next one for on given session_id
        :param session_id: :param choices_dict: each key represent a content to put in the list and the value could
        be either None, either :param another dict which keys will be the content of the next ComboBox etc... or
        instead of a dict as value it: :param could be a list that will define the content.: :param legends: :param
        index:

        Returns:

    get_value ()
        Returns:

    set_value (value)
        Set a new value. Either value is None and nothing will happen If value is a list instance, :param value:

        Returns:

    to_stretch ()
        Indicate if the widget should take all the space of a horizontal layout how might share the space with
        another widget Returns: Boolean

```

**class** cicada.gui.cicada\_analysis\_parameters\_gui.**EmittingErrStream** (*parent=None*)

Class managing the std.err redirection

**write** (*text*)

Override of the write function used to display output :param text: Python output from stdout :type text: str

**class** cicada.gui.cicada\_analysis\_parameters\_gui.**EmittingStream** (*parent=None*)

Class managing the std.out redirection

**write** (*text*)

Override of the write function used to display output :param text: Python output from stdout :type text: str

**class** cicada.gui.cicada\_analysis\_parameters\_gui.**FileDialogWidget** (*analysis\_arg,*  
*direc-*  
*tory\_only,*  
*exten-*  
*sions=None,*  
*par-*  
*ent=None*)

Create a widget that will contain a button to open a FileDialog and a label to display the file or directory choosen  
A label will also explain what this parameter do

**get\_value** ()

Return the argument

**Returns** Dictionary with the set value

**Return type** result\_dict (dict)

**one\_for\_all** ()

Allows to select the same dir of files for all session Returns:

**set\_value** (*value*)

Set the value :param value: either None, either a string or either a dictionary

Returns:

**to\_stretch** ()

Indicate if the widget should take all the space of a horizontal layout how might share the space with  
another widget Returns: Boolean

**class** cicada.gui.cicada\_analysis\_parameters\_gui.**FinalMeta**

**class** cicada.gui.cicada\_analysis\_parameters\_gui.**LineEditWidget** (*analysis\_arg,*  
*parent=None*)

**get\_value** ()

Return the value of the widget Returns:

**set\_value** (*value*)

Set the widget value to the value passed Returns:

**to\_stretch** ()

Indicate if the widget should take all the space of a horizontal layout how might share the space with  
another widget Returns: Boolean

**class** cicada.gui.cicada\_analysis\_parameters\_gui.**ListCheckboxWidget** (*analysis\_arg,*  
*choices\_attr\_name,*  
*par-*  
*ent=None*)

Allows multiple choices

```

get_value()
    Returns:

set_value(value)
    Set the value. :param value: value is either a string or integer or float, or a list. If a list, then item whose
    value matches :param one of the elements in the list will be checked:

    Returns: None

to_stretch()
    Indicate if the widget should take all the space of a horizontal layout how might share the space with
    another widget Returns: Boolean

class cicada.gui.cicada_analysis_parameters_gui.MyFileDialogQPushButton(key_name,
                                                                    file_dialog,
                                                                    file_dialogs_dict,
                                                                    parent=None)

    Special button for opening file dialog

open_dialog()
    Open the QFileDialog :param key_name:

    Returns:

class cicada.gui.cicada_analysis_parameters_gui.MyQComboBox
    Special instance of ComboBox allowing to handle change so that it is connected to other combo_boxes

selection_change(index)
    Called if the selection is changed either by the user or by the code :param index:

    Returns:

class cicada.gui.cicada_analysis_parameters_gui.MyQFrame(analysis_arg=None,
                                                         parent=None,
                                                         with_description=True)

change_mandatory_property(value)
    Changing the property allowing to change the style sheet depending on the mandatory aspect of the argument :param value:

    Returns:

set_property_to_missing()
    Allows the change the stylesheet and indicate the user that a Returns:

class cicada.gui.cicada_analysis_parameters_gui.ParameterWidgetModel

get_value()
    Return the value of the widget Returns:

set_value(value)
    Set the widget value to the value passed Returns:

to_stretch()
    Indicate if the widget should take all the space of a horizontal layout how might share the space with
    another widget Returns: Boolean

class cicada.gui.cicada_analysis_parameters_gui.ProgressBar(remaining_time_label,
                                                         parent=None)

    Class containing the progress bar of the current analysis

```

**update\_progress\_bar** (*time\_elapsed*, *increment\_value=0*, *new\_set\_value=0*)

Update the progress bar in the analysis widget and the corresponding remaining time :param *time\_elapsed*: Time elapsed since beginning of analysis, in seconds :type *time\_elapsed*: float :param *increment\_value*: Value that should be added to the current value of the progress bar :type *increment\_value*: float :param *new\_set\_value*: Value that should be set as the current value of the progress bar :type *new\_set\_value*: float

Returns:

**update\_progress\_bar\_overview** (*name*, *increment\_value=0*, *new\_set\_value=0*)

Update the overview progress bar

#### Parameters

- **name** (*str*) – Analysis ID
- **time\_started** (*float*) – Start time of the analysis
- **increment\_value** (*float*) – Value that should be added to the current value of the progress bar
- **new\_set\_value** (*float*) – Value that should be set as the current value of the progress bar

**class** `cicada.gui.cicada_analysis_parameters_gui.RemainingTime` (*parent=None*)

Class containing the remaining time of the analysis

**static correct\_time\_converter** (*time\_to\_convert*)

Convert a float in a correct duration value :param *time\_to\_convert*: Float value representing seconds to be converted in a correct duration with MM.SS :type *time\_to\_convert*: float

**Returns** String of the correct duration

**Return type** *time\_text* (*str*)

**update\_remaining\_time** (*progress\_value*, *time\_elapsed*, *done=False*)

Update the remaining time :param *progress\_value*: Current progress bar value :type *progress\_value*: float :param *time\_elapsed*: Time elapsed since the beginning of the analysis (in sec) :type *time\_elapsed*: float :param *done*: True if the analysis is done and false if still running :type *done*: bool

**class** `cicada.gui.cicada_analysis_parameters_gui.SameFamilyWidgetsContainer` (*widgets*,  
*parent=None*)

A QFrame used to group widgets that belongs to a same group. Just useful for visual purposes

**to\_stretch** ()

Indicate if the widget should take all the space of a horizontal layout how might share the space with another widget Returns: Boolean

**class** `cicada.gui.cicada_analysis_parameters_gui.SliderWidget` (*analysis\_arg*, *parent=None*)

Used to set a numerical value

**get\_value** ()

Return the value of the widget Returns:

**set\_value** (*value*)

Set the widget value to the value passed Returns:

**to\_stretch** ()

Indicate if the widget should take all the space of a horizontal layout how might share the space with another widget Returns: Boolean

---

```

class cicada.gui.cicada_analysis_parameters_gui.Worker (name, cicada_analysis, analysis_arguments_handler, parent)

    Thread to manage multiple analyses at the same time

    run ()
        Run the analysis

    setProgress (name, time_elapsed=0, increment_value=0, new_set_value=0)
        Emit the new value of the progress bar and time remaining

        Parameters
        • name (str) – Analysis ID
        • time_elapsed (float) – Start elapsed (in sec)
        • increment_value (float) – Value that should be added to the current value of the progress bar
        • new_set_value (float) – Value that should be set as the current value of the progress bar

    set_results_path (results_path)
        Set the selected path to the results in the “Open result folder” button in the corresponding overview

        Parameters results_path (str) – Path to the results

cicada.gui.cicada_analysis_parameters_gui.except_hook (cls, exception, traceback)
    Redirect exception to std.err so we can display stack trace on exceptions

```





## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### C

`cicada.analysis.cicada_analysis`, [11](#)  
`cicada.analysis.cicada_analysis_arguments_handler`,  
    [12](#)  
`cicada.analysis.cicada_analysis_format_wrapper`,  
    [13](#)  
`cicada.analysis.cicada_analysis_nwb_wrapper`,  
    [14](#)  
`cicada.analysis.cicada_connectivity_graph`,  
    [16](#)  
`cicada.analysis.cicada_hubs_analysis`,  
    [16](#)  
`cicada.analysis.cicada_psth_analysis`,  
    [16](#)  
`cicada.gui.cicada_analysis_overview`, [20](#)  
`cicada.gui.cicada_analysis_parameters_gui`,  
    [21](#)  
`cicada.gui.cicada_analysis_tree_gui`, [19](#)  
`cicada.gui.cicada_main_window`, [17](#)  
`cicada.preprocessing.cicada_data_to_nwb`,  
    [5](#)  
`cicada.preprocessing.cicada_preprocessing_run`,  
    [7](#)  
`cicada.preprocessing.convert_abf_to_nwb`,  
    [8](#)  
`cicada.preprocessing.convert_ci_movie_to_nwb`,  
    [8](#)  
`cicada.preprocessing.convert_processed_2d_series_to_nwb`,  
    [7](#)  
`cicada.preprocessing.convert_suite_2p_rois_to_nwb`,  
    [7](#)  
`cicada.preprocessing.convert_to_nwb`, [7](#)  
`cicada.preprocessing.utils`, [9](#)



## A

`about()` (*cicada.gui.cicada\_main\_window.CicadaMainWindow* method), 17  
`add_analysis_overview()` (*cicada.gui.cicada\_analysis\_overview.AnalysisOverview* method), 20  
`add_argument()` (*cicada.analysis.cicada\_analysis\_arguments\_handler.AnalysisArgumentsHandler* method), 12  
`add_argument_for_gui()` (*cicada.analysis.cicada\_analysis.CicadaAnalysis* method), 11  
`add_ci_movie_arg_for_gui()` (*cicada.analysis.cicada\_analysis.CicadaAnalysis* method), 11  
`add_data()` (*cicada.gui.cicada\_main\_window.CicadaMainWindow* method), 17  
`add_group_data()` (*cicada.gui.cicada\_main\_window.CicadaMainWindow* method), 18  
`add_multiple_combo_boxes()` (*cicada.gui.cicada\_analysis\_parameters\_gui.ComboBoxWidget* method), 23  
`add_segmentation_arg_for_gui()` (*cicada.analysis.cicada\_analysis.CicadaAnalysis* method), 11  
`age` (*cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper* attribute), 13  
`age` (*cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper* attribute), 14  
`AnalysisArgumentsHandler` (class in *cicada.analysis.cicada\_analysis\_arguments\_handler*), 12  
`AnalysisData` (class in *cicada.gui.cicada\_analysis\_parameters\_gui*), 21  
`AnalysisOverview` (class in *cicada.gui.cicada\_analysis\_overview*), 20  
`AnalysisPackage` (class in *cicada.gui.cicada\_analysis\_parameters\_gui*), 21  
`AnalysisParametersApp` (class in *cicada.gui.cicada\_analysis\_parameters\_gui*), 22  
`AnalysisState` (class in *cicada.gui.cicada\_analysis\_overview*), 20  
`AnalysisTreeApp` (class in *cicada.gui.cicada\_analysis\_tree\_gui*), 19

## B

`bring_to_front()` (*cicada.gui.cicada\_analysis\_overview.AnalysisState* method), 21  
`bring_to_front()` (*cicada.gui.cicada\_analysis\_parameters\_gui.AnalysisPackage* method), 21

## C

`change_mandatory_property()` (*cicada.gui.cicada\_analysis\_parameters\_gui.MyQFrame* method), 25  
`check_arguments_validity()` (*cicada.analysis.cicada\_analysis\_arguments\_handler.AnalysisArgumentsHandler* method), 13  
`check_data()` (*cicada.analysis.cicada\_analysis.CicadaAnalysis* method), 16  
`check_data()` (*cicada.analysis.cicada\_connectivity\_graph.CicadaConnectivityGraph* method), 16  
`check_data()` (*cicada.analysis.cicada\_hubs\_analysis.CicadaHubsAnalysis* method), 16  
`check_data()` (*cicada.analysis.cicada\_psth\_analysis.CicadaPsthAnalysis* method), 16  
`CheckBoxWidget` (class in *cicada.gui.cicada\_analysis\_parameters\_gui*), 22  
`cicada.analysis.cicada_analysis` (module), 11  
`cicada.analysis.cicada_analysis_arguments_handler` (module), 12

cicada.analysis.cicada\_analysis\_format\_wrapper (module), 13  
 cicada.analysis.cicada\_analysis\_nwb\_wrapper (module), 14  
 cicada.analysis.cicada\_connectivity\_graph (module), 16  
 cicada.analysis.cicada\_hubs\_analysis (module), 16  
 cicada.analysis.cicada\_psth\_analysis (module), 16  
 cicada.gui.cicada\_analysis\_overview (module), 20  
 cicada.gui.cicada\_analysis\_parameters\_gui (module), 21  
 cicada.gui.cicada\_analysis\_tree\_gui (module), 19  
 cicada.gui.cicada\_main\_window (module), 17  
 cicada.preprocessing.cicada\_data\_to\_nwb (module), 5  
 cicada.preprocessing.cicada\_preprocessing\_run (module), 7  
 cicada.preprocessing.convert\_abf\_to\_nwb (module), 8  
 cicada.preprocessing.convert\_ci\_movie\_to\_nwb (module), 8  
 cicada.preprocessing.convert\_processed\_2d\_series\_to\_nwb (module), 7  
 cicada.preprocessing.convert\_suite\_2p\_rois\_to\_nwb (module), 7  
 cicada.preprocessing.convert\_to\_nwb (module), 7  
 cicada.preprocessing.utils (module), 9  
 CicadaAnalysis (class in cicada.analysis.cicada\_analysis), 11  
 CicadaAnalysisFormatWrapper (class in cicada.analysis.cicada\_analysis\_format\_wrapper), 13  
 CicadaAnalysisNwbWrapper (class in cicada.analysis.cicada\_analysis\_nwb\_wrapper), 14  
 CicadaConnectivityGraph (class in cicada.analysis.cicada\_connectivity\_graph), 16  
 CicadaHubsAnalysis (class in cicada.analysis.cicada\_hubs\_analysis), 16  
 CicadaMainWindow (class in cicada.gui.cicada\_main\_window), 17  
 CicadaPsthAnalysis (class in cicada.analysis.cicada\_psth\_analysis), 16  
 class\_name\_to\_file\_name() (in module cicada.preprocessing.utils), 9  
 closeEvent() (cicada.gui.cicada\_main\_window.CicadaMainWindow method), 18  
 ColorDialogWidget (class in cicada.gui.cicada\_analysis\_parameters\_gui), 23  
 columnCount() (cicada.gui.cicada\_analysis\_tree\_gui.QAnalysisTreeModel method), 20  
 ComboBoxWidget (class in cicada.gui.cicada\_analysis\_parameters\_gui), 23  
 ComparableItem (class in cicada.preprocessing.utils), 9  
 contains\_ci\_movie() (cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper method), 13  
 contains\_ci\_movie() (cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper method), 14  
 convert() (cicada.preprocessing.convert\_abf\_to\_nwb.ConvertAbfToNwb method), 8  
 convert() (cicada.preprocessing.convert\_ci\_movie\_to\_nwb.ConvertCiMovieToNwb method), 8  
 convert() (cicada.preprocessing.convert\_processed\_2d\_series\_to\_nwb.ConvertProcessed2dSeriesToNwb method), 7  
 convert() (cicada.preprocessing.convert\_suite\_2p\_rois\_to\_nwb.ConvertSuite2pRoisToNwb method), 7  
 convert() (cicada.preprocessing.convert\_to\_nwb.ConvertToNWB method), 7  
 convert\_data\_to\_nwb() (in module cicada.preprocessing.cicada\_data\_to\_nwb), 5  
 ConvertAbfToNWB (class in cicada.preprocessing.convert\_abf\_to\_nwb), 8  
 ConvertCiMovieToNWB (class in cicada.preprocessing.convert\_ci\_movie\_to\_nwb), 8  
 ConvertProcessed2dSeriesToNWB (class in cicada.preprocessing.convert\_processed\_2d\_series\_to\_nwb), 7  
 ConvertSuite2pRoisToNWB (class in cicada.preprocessing.convert\_suite\_2p\_rois\_to\_nwb), 7  
 ConvertToNWB (class in cicada.preprocessing.convert\_to\_nwb), 7  
 correct\_time\_converter() (cicada.gui.cicada\_analysis\_parameters\_gui.RemainingTime static method), 26  
 create\_convert\_class() (in module cicada.preprocessing.cicada\_data\_to\_nwb), 6  
 create\_group\_menu() (cicada.gui.cicada\_main\_window.CicadaMainWindow method), 18  
 create\_nwb\_file() (in module cicada.preprocessing.cicada\_data\_to\_nwb), 5

6  
 create\_results\_directory() (ci-  
   cada.analysis.cicada\_analysis.CicadaAnalysis  
   method), 12  
 create\_sort\_menu() (ci-  
   cada.gui.cicada\_main\_window.CicadaMainWindow  
   method), 18  
 create\_tree\_model() (ci-  
   cada.gui.cicada\_analysis\_tree\_gui.AnalysisTreeApp  
   method), 19  
 create\_widgets() (ci-  
   cada.gui.cicada\_analysis\_parameters\_gui.AnalysisParametersApp  
   method), 22  
 createAction() (ci-  
   cada.gui.cicada\_main\_window.CicadaMainWindow  
   method), 18  
 createMenus() (ci-  
   cada.gui.cicada\_main\_window.CicadaMainWindow  
   method), 18

**D**  
 data() (cicada.gui.cicada\_analysis\_tree\_gui.QAnalysisTreeModel  
   method), 20  
 deleteLater() (ci-  
   cada.gui.cicada\_analysis\_overview.AnalysisState  
   method), 21  
 deleteLater() (ci-  
   cada.gui.cicada\_analysis\_overview.ResultsButton  
   method), 21  
 detect\_run\_periods() (ci-  
   cada.preprocessing.convert\_abf\_to\_nwb.ConvertAbfToNwb  
   method), 9  
 determine\_ci\_frames\_indices() (ci-  
   cada.preprocessing.convert\_abf\_to\_nwb.ConvertAbfToNwb  
   method), 9  
 doubleClickedItem() (ci-  
   cada.gui.cicada\_analysis\_tree\_gui.AnalysisTreeApp  
   method), 19  
 drawBranches() (ci-  
   cada.gui.cicada\_analysis\_tree\_gui.QAnalysisTreeView  
   method), 20

**E**  
 EmittingErrStream (class in ci-  
   cada.gui.cicada\_analysis\_parameters\_gui),  
   23  
 EmittingStream (class in ci-  
   cada.gui.cicada\_analysis\_parameters\_gui),  
   24  
 errOutputWritten() (ci-  
   cada.gui.cicada\_analysis\_parameters\_gui.AnalysisParametersApp  
   method), 22  
 except\_hook() (in module ci-  
   cada.gui.cicada\_analysis\_parameters\_gui),

27  
**F**  
 FileDialogWidget (class in ci-  
   cada.gui.cicada\_analysis\_parameters\_gui),  
   24  
 fill\_tree\_item\_with\_dict() (in module ci-  
   cada.gui.cicada\_analysis\_tree\_gui), 20  
 filter\_list\_according\_to\_keywords() (in  
   module ci-  
   cada.preprocessing.cicada\_data\_to\_nwb),  
   6  
 filter\_list\_of\_files() (in module ci-  
   cada.preprocessing.cicada\_data\_to\_nwb),  
   6  
 FinalMeta (class in ci-  
   cada.gui.cicada\_analysis\_parameters\_gui),  
   24  
 flags() (cicada.gui.cicada\_analysis\_tree\_gui.QAnalysisTreeModel  
   method), 20  
 flatten() (in module cicada.preprocessing.utils), 9

**G**  
 genotype (cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnal  
   attribute), 13  
 genotype (cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysis  
   attribute), 15  
 get\_analysis\_argument() (ci-  
   cada.analysis.cicada\_analysis\_arguments\_handler.AnalysisArgum  
   method), 13  
 get\_analysis\_arguments() (ci-  
   cada.analysis.cicada\_analysis\_arguments\_handler.AnalysisArgum  
   method), 13  
 get\_nwb\_movies() (ci-  
   cada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisF  
   method), 14  
 get\_ci\_movies() (ci-  
   cada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwb  
   method), 15  
 get\_continuous\_time\_periods() (in module ci-  
   cada.preprocessing.utils), 9  
 get\_data\_identifiers() (ci-  
   cada.analysis.cicada\_analysis.CicadaAnalysis  
   method), 12  
 get\_data\_to\_analyse() (ci-  
   cada.analysis.cicada\_analysis.CicadaAnalysis  
   method), 12  
 get\_gui\_widgets() (ci-  
   cada.analysis.cicada\_analysis\_arguments\_handler.AnalysisArgum  
   method), 13  
 get\_package\_identifier() (ci-  
   cada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisF  
   method), 14

get\_identifier() (cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper  
 method), 15  
 get\_intervals\_names() (cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper  
 method), 14  
 get\_intervals\_names() (cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper  
 method), 15  
 get\_pixel\_mask() (cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper  
 method), 14  
 get\_pixel\_mask() (cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper  
 method), 15  
 get\_results\_path() (cicada.analysis.cicada\_analysis.CicadaAnalysis  
 method), 12  
 get\_roi\_response\_series() (cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper  
 method), 14  
 get\_roi\_response\_series() (cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper  
 method), 15  
 get\_segmentations() (cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper  
 method), 14  
 get\_segmentations() (cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper  
 method), 15  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.CheckBoxWidget  
 method), 23  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.ColorDialogWidget  
 method), 23  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.ComboBoxWidget  
 method), 23  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.FileDialogWidget  
 method), 24  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.LineEditWidget  
 method), 24  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.ListCheckBoxWidget  
 method), 24  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.ParameterWidgetModel  
 method), 25  
 get\_value() (cicada.gui.cicada\_analysis\_parameters\_gui.SliderWidget  
 method), 26  
 H  
 headerData() (cicada.gui.cicada\_analysis\_tree\_gui.QAnalysisTreeModel  
 method), 20  
 I  
 identifier(cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper  
 attribute), 14



`load_group()` (`cicada.gui.cicada_main_window.CicadaMainWindow` dataset () (ci-  
*method*), 18 `cada.gui.cicada_main_window.CicadaMainWindow`  
`load_group_from_config()` (ci- *method*), 18  
`cada.gui.cicada_main_window.CicadaMainWindow`  
*method*), 18  
`load_rasterplot_in_memory()` (ci-  
`cada.preprocessing.convert_processed_2d_series_to_nwb.ConvertProcessed2dSeriesToNWB`  
*static method*), 7  
`load_tiff_movie_in_memory()` (in module ci-  
`cada.preprocessing.utils`), 9  
`load_tiff_movie_in_memory_using_pil()` (in module `cicada.preprocessing.utils`), 10

## M

`merging_time_periods()` (in module ci-  
`cada.preprocessing.utils`), 10  
`MusketeersWidget` (class in ci-  
`cada.gui.cicada_main_window`), 19  
`MyFileDialogQButton` (class in ci-  
`cada.gui.cicada_analysis_parameters_gui`),  
25  
`MyQComboBox` (class in ci-  
`cada.gui.cicada_analysis_parameters_gui`),  
25  
`MyQFrame` (class in ci-  
`cada.gui.cicada_analysis_parameters_gui`),  
25

## N

`normalOutputWritten()` (ci-  
`cada.gui.cicada_analysis_parameters_gui.AnalysisPackage`  
*method*), 22

## O

`on_close()` (`cicada.gui.cicada_analysis_parameters_gui.AnalysisPackage`  
*method*), 22  
`on_group()` (`cicada.gui.cicada_main_window.CicadaMainWindow`  
*method*), 18  
`on_sort()` (`cicada.gui.cicada_main_window.CicadaMainWindow`  
*method*), 18  
`one_for_all()` (ci-  
`cada.gui.cicada_analysis_parameters_gui.FileDialogWidget`  
*method*), 24  
`open_dialog()` (ci-  
`cada.gui.cicada_analysis_parameters_gui.ColorDialogWidget`  
*method*), 23  
`open_dialog()` (ci-  
`cada.gui.cicada_analysis_parameters_gui.MyFileDialogQButton`  
*method*), 25  
`open_explorer()` (ci-  
`cada.gui.cicada_analysis_overview.ResultsButton`  
*method*), 21

## P

`ParameterWidgetModel` (class in ci-  
`cada.gui.cicada_analysis_parameters_gui`),  
25  
`parent()` (`cicada.gui.cicada_analysis_tree_gui.QAnalysisTreeModel`  
*method*), 20  
`populate_menu()` (ci-  
`cada.gui.cicada_main_window.CicadaMainWindow`  
*method*), 18  
`populate_session_list()` (ci-  
`cada.gui.cicada_analysis_parameters_gui.AnalysisData`  
*method*), 21  
`process_run_data()` (ci-  
`cada.preprocessing.convert_abf_to_nwb.ConvertAbfToNWB`  
*method*), 9  
`ProgressBar` (class in ci-  
`cada.gui.cicada_analysis_parameters_gui`),  
25

## Q

`QAnalysisTreeModel` (class in ci-  
`cada.gui.cicada_analysis_tree_gui`), 19  
`QAnalysisTreeView` (class in ci-  
`cada.gui.cicada_analysis_tree_gui`), 20

## R

`RemainingTime` (class in ci-  
`cada.gui.cicada_analysis_parameters_gui`),  
26  
`reset_arguments()` (ci-  
`cada.gui.cicada_analysis_parameters_gui.AnalysisParametersAp`  
*method*), 22  
`ResultsButton` (class in ci-  
`cada.gui.cicada_analysis_overview`), 21  
`rowCount()` (`cicada.gui.cicada_analysis_tree_gui.QAnalysisTreeModel`  
*method*), 20  
`run()` (`cicada.gui.cicada_analysis_parameters_gui.Worker`  
*method*), 27  
`run_analysis()` (ci-  
`cada.analysis.cicada_analysis.CicadaAnalysis`  
*method*), 12  
`run_analysis()` (ci-  
`cada.analysis.cicada_connectivity_graph.CicadaConnectivityGra`  
*method*), 16  
`run_analysis()` (ci-  
`cada.analysis.cicada_hubs_analysis.CicadaHubsAnalysis`  
*method*), 16

```

run_analysis()
    cada.analysis.cicada_psth_analysis.CicadaPsthAnalysis
    method), 16
run_analysis()
    (ci-
    method), 24
    cada.gui.cicada_analysis_parameters_gui.AnalysisParametersApp(cicada.gui.cicada_analysis_parameters_gui.ListCheckboxesApp), 22
    method), 25
    set_value() (cicada.gui.cicada_analysis_parameters_gui.ParameterWidget), 25
    method), 25
SameFamilyWidgetsContainer (class in ci-
    cada.gui.cicada_analysis_parameters_gui),
    26
    set_value() (cicada.gui.cicada_analysis_parameters_gui.SliderWidget), 26
    set_widgets_to_default_value() (ci-
    cada.analysis.cicada_analysis_arguments_handler.AnalysisArgumentsHandler), 13
save_analysis_arguments_to_yaml_file()
    (cicada.analysis.cicada_analysis_arguments_handler.AnalysisArgumentsHandler), 13
    setProgress() (ci-
    cada.gui.cicada_analysis_parameters_gui.Worker), 27
save_last_data_location()
    (ci-
    cada.gui.cicada_main_window.CicadaMainWindow), 18
    method), 18
    sex(cicada.analysis.cicada_analysis_format_wrapper.CicadaAnalysisFormatWrapper), 14
save_yaml_with_name()
    (ci-
    cada.gui.cicada_analysis_parameters_gui.AnalysisParametersApp), 22
    attribute), 15
    species(cicada.analysis.cicada_analysis_nwb_wrapper.CicadaAnalysisNwbWrapper), 15
see_all_groups()
    (ci- SliderWidget (class in ci-
    cada.gui.cicada_main_window.CicadaMainWindow), 19
    cada.gui.cicada_analysis_parameters_gui),
    26
select_group()
    (ci- sort_by_param() (in module ci-
    cada.gui.cicada_main_window.CicadaMainWindow), 19
    cada.preprocessing.utils), 10
    species(cicada.analysis.cicada_analysis_format_wrapper.CicadaAnalysisFormatWrapper), 14
selection_change()
    (ci-
    cada.gui.cicada_analysis_parameters_gui.MyQComboBox), 25
    species(cicada.analysis.cicada_analysis_nwb_wrapper.CicadaAnalysisNwbWrapper), 15
    attribute), 15
set_argument_value()
    (ci- subject_id(cicada.analysis.cicada_analysis_format_wrapper.CicadaAnalysisFormatWrapper), 14
    cada.analysis.cicada_analysis_arguments_handler.AnalysisArgumentsHandler), 13
    subject_id(cicada.analysis.cicada_analysis_nwb_wrapper.CicadaAnalysisNwbWrapper), 15
    attribute), 15
set_arguments_for_gui()
    (ci-
    cada.analysis.cicada_analysis.CicadaAnalysis), 12
    T
set_arguments_for_gui()
    (ci- tabula_rasa() (ci-
    cada.analysis.cicada_psth_analysis.CicadaPsthAnalysis), 16
    cada.gui.cicada_analysis_parameters_gui.AnalysisParametersApp), 22
    method), 22
set_data() (cicada.analysis.cicada_analysis.CicadaAnalysis), 12
    stretch() (cicada.gui.cicada_analysis_parameters_gui.CheckBoxWidget), 23
    method), 23
set_data() (cicada.gui.cicada_analysis_tree_gui.AnalysisTreeApp), 19
    stretch() (cicada.gui.cicada_analysis_parameters_gui.ColorDialog), 23
    method), 23
set_property_to_missing()
    (ci- to_stretch() (cicada.gui.cicada_analysis_parameters_gui.ComboBoxWidget), 25
    cada.gui.cicada_analysis_parameters_gui.MyQFrame), 23
    method), 23
    to_stretch() (cicada.gui.cicada_analysis_parameters_gui.FileDialog), 25
    method), 25
set_results_path()
    (ci-
    cada.gui.cicada_analysis_parameters_gui.Worker), 27
    to_stretch() (cicada.gui.cicada_analysis_parameters_gui.LineEditWidget), 24
    method), 24
set_value() (cicada.gui.cicada_analysis_parameters_gui.CheckBoxWidget), 23
    stretch() (cicada.gui.cicada_analysis_parameters_gui.ListCheckboxesApp), 25
    method), 25
set_value() (cicada.gui.cicada_analysis_parameters_gui.ColorDialog), 23
    stretch() (cicada.gui.cicada_analysis_parameters_gui.ParameterWidget), 25
    method), 25
set_value() (cicada.gui.cicada_analysis_parameters_gui.ComboBoxWidget), 23
    stretch() (cicada.gui.cicada_analysis_parameters_gui.SameFamilyWidgetsContainer), 26
    method), 26

```

`to_stretch()` (*cicada.gui.cicada\_analysis\_parameters\_gui.SliderWidget*  
method), 26

## U

`uncheck_all_sort()` (*ci-*  
*cada.gui.cicada\_main\_window.CicadaMainWindow*  
method), 19

`uncheck_group()` (*ci-*  
*cada.gui.cicada\_main\_window.CicadaMainWindow*  
method), 19

`update_button_color()` (*ci-*  
*cada.gui.cicada\_analysis\_parameters\_gui.ColorDialogWidget*  
method), 23

`update_frames_to_add()` (*in module ci-*  
*cada.preprocessing.utils*), 10

`update_original_data()` (*ci-*  
*cada.analysis.cicada\_analysis.CicadaAnalysis*  
method), 12

`update_original_data()` (*ci-*  
*cada.analysis.cicada\_connectivity\_graph.CicadaConnectivityGraph*  
method), 16

`update_original_data()` (*ci-*  
*cada.analysis.cicada\_hubs\_analysis.CicadaHubsAnalysis*  
method), 16

`update_original_data()` (*ci-*  
*cada.analysis.cicada\_psth\_analysis.CicadaPsthAnalysis*  
method), 16

`update_progress_bar()` (*ci-*  
*cada.gui.cicada\_analysis\_parameters\_gui.ProgressBar*  
method), 25

`update_progress_bar_overview()` (*ci-*  
*cada.gui.cicada\_analysis\_parameters\_gui.ProgressBar*  
method), 26

`update_progressbar()` (*ci-*  
*cada.analysis.cicada\_analysis.CicadaAnalysis*  
method), 12

`update_remaining_time()` (*ci-*  
*cada.gui.cicada\_analysis\_parameters\_gui.RemainingTime*  
method), 26

## W

`weight` (*cicada.analysis.cicada\_analysis\_format\_wrapper.CicadaAnalysisFormatWrapper*  
attribute), 14

`weight` (*cicada.analysis.cicada\_analysis\_nwb\_wrapper.CicadaAnalysisNwbWrapper*  
attribute), 15

`Worker` (*class in ci-*  
*cada.gui.cicada\_analysis\_parameters\_gui*),  
26

`write()` (*cicada.gui.cicada\_analysis\_parameters\_gui.EmittingErrStream*  
method), 24

`write()` (*cicada.gui.cicada\_analysis\_parameters\_gui.EmittingStream*  
method), 24