
anesthPlot

Release beta

Christophe Desbois

Feb 25, 2022

MAIN_{SCRIPT} :

1	Welcome to anesthPlot's documentation!	1
2	main script	3
3	modules	5
4	Indices and tables	23
	Python Module Index	25
	Index	27

WELCOME TO ANESTHPLOT'S DOCUMENTATION!

anesthPlot is a python package developped to extract, manipulate and plots anesthesia data recorded from the Monitor Software to be used mostly in a teaching environment.

Warning: This project is:

- a work in progres
- the processes are mainly focused on horses anesthesia
- in our environment the data recorded came from either
 - an as3 or as5 anesthesia monitor (ekg, invasive pressure, etCO2, halogenate, spirometry)
 - a Taphonius equine ventilator
 - (some ekg data extracted using a Televet holter system)

1.1 Features

- you can **load** recordings from a trend or a wave file
 - **from command line:**

```
python anesthPlot/anesplot/__main__.py
-> will open an GUI choose menu to choose the recording
(MonitorTrend, TaphoniusTrend, MonitorWave, TelevetWave(export))
```

- * will build a **standard debriefing** (trends) **plot series** (script usage)
 - global histograms (cardiovascular and anesthesia summary)
 - cardiovascular trends time based plots
 - respiratory trends time based plots
 - anesthesia trends time based plots
- * or will build a **plot for wave** recording
 - one or two waves on the same plot (script usage)
- you can also use this code as a **python package**
 - **usage :**

```
import anesplot.record_main as rec
trendname = 'a_full_path_to_csv_file'
# nb if no filename is provided, a chooseFile Gui will be called to choose_
↪ the file
trends = rec.MonitorTrend(trendname)
#(you can also use trends = rec.taphTrend())
wavename = rec.trendname_to_wavename(trendname)
waves = rec.MonitorWave(wavename)

trends.show_graphs() # -> set of plots for 'clinical' debriefing purposes

waves.plot_waves() # -> one or two traces
# ... adjust manually the scales of the display
waves.define_a_roi() # -> to register the plotting scales
waves.animate_fig() #-> to build an animation using these parameters
```

- additional functions are available to extract instaneous heart rate
 - * see anesplot/treatrec/ekg_to_hr.py

MAIN SCRIPT

2.1 anesplot.record_main module

main script/module to load and display an anesthesia record

can be runned as a script:: python record_main.py

or imported as a package:

```
import anesplot.record_main as rec
%gui qt5 (required only to use the dialogs if using spyder)

# objects:
mtrends = rec.MonitorTrend()
waves = rec.MonitorWave(rec.trendname_to_wavename(mtrends.filename))
ttrends = rec.TaphTtrend()

# use methods and or attributes:
mtrends.show_graphs() -> clinical debrief selection
waves.plot_wave() -> select one or two waves to plot

...
```

`anesplot.record_main.choosefile_gui`(*dirname: Optional[str] = None*) → str
Select a file via a dialog and return the (full) filename.

Parameters `dirname` (*str, optional*) –

DESCRIPTION. location to place the gui ('generally paths['data']) else home The default is None.

Returns `fname[0]` – DESCRIPTION. : full name of the selected file

Return type str

`anesplot.record_main.trendname_to_wavename`(*name: str*) → str
just compute the supposed (full)name

`anesplot.record_main.select_type`(*question: Optional[str] = None, items: Optional[list] = None, num: int = 0*) → str
display a pulldown menu to choose the kind of recording

Parameters

- **question** (*str, optional*) – The question that appears in the dialog (default is None).
- **items** (*list, optional*) – the list of all items in the pulldown menu. (default is None).

- **num** (*int*, *optional*) – number in the list the pointer will be one. The default is 0.

Returns kind of recording in [monitorTrend, monitorWave, taphTrend, telvet].

Return type str

`anesplot.record_main.select_wave_to_plot(waves: list, num=1) → str`
select the wave trace to plot

Parameters

- **waves** (*list*) – list of available waves traces
- **num** (*TYPE*, *optional*) – index of the waves in the plot (1 or 2)

Returns wave name

Return type str

`anesplot.record_main.plot_trenddata(datadf: pandas.core.frame.DataFrame, header: dict, param_dico: dict) → dict`
generate a series of plots for anesthesia debriefing purposes

Parameters

- **datadf** (*pd.DataFrame*) – recorded data (MonitorTrend.data or TaphTrend.data).
- **header** (*dict*) – recording parameters (MonitorTrend.header or TaphTrend.header).
- **param_dico** (*dict*) – plotting parameters (MonitorTrend.param or TaphTrend.param).

Returns afig_dico : {names:fig_obj} of displayed figures

Return type dict

class `anesplot.record_main.MonitorTrend(filename: Optional[str] = None, load: bool = True)`
Bases: `anesplot.record_main._SlowWave`

monitor trends recordings:

input = filename : path to file load = boolean to load data (default is True)

file [str] short name

filename [str] long name

header [dict] record parameters

param [dict] parameters

clean_trend [external] clean the data

show_graphs [external] plot clinical main plots

class `anesplot.record_main.TaphTrend(filename: Optional[str] = None, monitorname: Optional[str] = None)`

Bases: `anesplot.record_main._SlowWave`

taphonius trends recordings input : filename : path to file

data : `pd.DataFrame` = recorded data header : dictionary = recorded info (patient, ...) param : dictionary = usage information (file, scales, ...) actions : `pd.DataFrame`

clean_trend : 'to be developped' show_graphs : plot the clinical debrief 'suite'


```

extract_events()
    decode the taph messages, build events, actions and ventil_drive

plot_ventil_drive(all_traces: bool = False)
    plot the ventilation commands that have been used

plot_events(todrop: Optional[list] = None, dtime: bool = False)
    plot the events as a time display, dtime allow dtime use

export_taph_events(save_to_file=False)
    export in a txt files all the events (paths:~/temp/events.txt)

class anesplot.record_main.TelevetWave(filename=None)
    Bases: anesplot.record_main._FastWave

    class to organise teleVet recordings transformed to csv files. input:

        filename : str (fullpath, default:None)

class anesplot.record_main.MonitorWave(filename: Optional[str] = None, load: bool = True)
    Bases: anesplot.record_main._FastWave

    class to organise monitorWave recordings. input : filename = path to file load = boolean to load data (default
    is True)

    attributes ... FILLMEq
    methods ... FILLME

anesplot.record_main.main(file_name: Optional[str] = None)
    main script called from command line call : "python record_main.py" call a GUI, load recording and display a
    series of plt.figure NB filename will be placed in the clipboard

    Parameters file_name (str, optional) – recordfile fullname (default is None).

    Return type None.

```


MODULES

3.1 anesplot package

3.1.1 Subpackages

anesplot.config package

Submodules

anesplot.config.build_recordrc module

build a 'recordRc.yaml' configuration file to adapt to a specific computer location at the root of anesplot

- input <-> 'data' : to load the records
- output <-> 'save' : to save the plots

```
anesplot.config.build_recordrc.filedialog(kind="", direc-  
                                         tory='/Users/cdesbois/pg/chrisPg/anesthPlot/anesplot/config',  
                                         for_open=True, fmt="", is_folder=False)  
    general dialog function.
```

```
anesplot.config.build_recordrc.read_config()  
    locate & load the yaml file.
```

```
anesplot.config.build_recordrc.write_configfile(path)  
    record the yaml file.
```

```
anesplot.config.build_recordrc.main()  
    main function for script execution.
```

anesplot.config.load_recordrc module

load an already generated 'recordRc.yaml' configuration file

- input <-> 'data' : to load the records
- output <-> 'save' : to save the plots

`anesplot.config.load_recordrc.build_paths()`

read the yaml configuration file.

`anesplot.config.load_recordrc.adapt_with_syspath(path_dico)`

add the folder location to the system path.

Module contents

anesplot.loadrec package

Submodules

anesplot.loadrec.explore module

Created on Thu Mar 12 16:52:13 2020

@author: cdesbois

`anesplot.loadrec.explore.gui_choosefile(paths=None)`

select a file via a dialog and return the file name.

anesplot.loadrec.loadmonitor_trendrecord module

Created on Wed Jul 24 13:43:26 2019 @author: cdesbois

load a monitor trend recording:

- choose a file
- load the header to a dictionary
- load the data into a pandas dataframe

`anesplot.loadrec.loadmonitor_trendrecord.choosefile_gui(dirname: Optional[str] = None) → str`
Select a file via a dialog and return the (full) filename.

Parameters `dirname` (*str, optional*) –

location to place the gui ('generally paths['data']) else home (default is None).

Returns the choosed file fullname.

Return type str

`anesplot.loadrec.loadmonitor_trendrecord.loadmonitor_trendheader(filename: str) → dict`
load the file header.

Parameters `filename` (*str*) – full name of the file.

Returns the content of the header.

Return type dict

`anesplot.loadrec.loadmonitor_trendrecord.loadmonitor_trenddata(filename: str, headerdico: dict) → pandas.core.frame.DataFrame`
load the monitor trend data

Parameters

- **filename** (*str*) – full name of the datafile.
- **headerdico** (*dict*) – fileheader content.

Returns the recorded data.

Return type pd.DataFrame

anesplot.loadrec.loadmonitor_waverecord module

Created on Wed Jul 24 14:56:58 2019 @author: cdesbois

load a monitor wave recording:

- choose a file
- load the header to a pandas dataframe
- load the date into a pandas dataframe

anesplot.loadrec.loadmonitor_waverecord.**choosefile_gui**(*dirname: Optional[str] = None*) → str
Select a file via a dialog and return the (full) filename.

Parameters **dirname** (*str, optional*) – DESCRIPTION. The default is None.

Returns str

Return type the choosed file full name

anesplot.loadrec.loadmonitor_waverecord.**loadmonitor_waveheader**(*filename: Optional[str] = None*) → dict
load the wave file header.

Parameters **filename** (*str, optional*) – full name of the file (default is None).

Returns content of the header.

Return type dict

anesplot.loadrec.loadmonitor_waverecord.**loadmonitor_wavedata**(*filename: Optional[str] = None*) → pandas.core.frame.DataFrame
load the monitor wave csvDataFile.

Parameters **filename** (*str, optional*) – full name of the file (default is None).

Returns the recorded wave data

Return type pandas.DataFrame

anesplot.loadrec.loadtaph_trendrecord module

Created on Wed Jul 24 15:30:07 2019 @author: cdesbois

load a taphonius data recording:

- choose a file
- load the patient datafile to a dictionary
- load the physiological date into a pandas dataframe

nb = 4 files per recording :

- .pdf -> anesthesia record ‘manual style’
 - .xml -> taphonius technical record -> to be extracted
 - Patient.csv -> patient id and specifications
 - SD...csv -> anesthesia record
-

`anesplot.loadrec.loadtaph_trendrecord.build_taph_decodedate_dico`(*pathdict: Optional[dict] = None*) → dict

list all the taph recordings and the paths to the record

Parameters `pathdict` (*dict, optional*) – dictionary containing {‘taph’: pathToTheData}, (default is None).

Returns get all the recorded files expressed as {date : filename}.

Return type dict

`anesplot.loadrec.loadtaph_trendrecord.extract_record_day`(*monitor_file_name: str*) → str
extract the date as ‘YYYY_MM_DD’ from a monitor_filename

Parameters `monitor_file_name` (*str*) – monitor file name (shortname).

Returns same date expressed as YYYY_MM_DD.

Return type str

`anesplot.loadrec.loadtaph_trendrecord.choose_taph_record`(*monitorname: Optional[str] = None*) → str

explore the recording folders and proposes to select one

Parameters `monitorname` (*str, optional*) – a monitor file (short) name to place the pointer in the pull down menu.

Returns selected file (full) name.

Return type str

`anesplot.loadrec.loadtaph_trendrecord.loadtaph_trenddata`(*filename: str*) → pandas.core.frame.DataFrame

load the taphoniusData trends data.

Parameters `filename` (*str*) – selected file (full) name.

Returns the recorded data.

Return type pandas.DataFrame

`anesplot.loadrec.loadtaph_trendrecord.loadtaph_patientfile`(*filename: str*) → dict
load the taphonius patient.csv file (‘header’ in monitor files, description)

Parameters `filename` (*str*) – the taph recording file (full) name (‘SDYYYYMMDD...’). (the headername will be reconstructed inside the function)

Returns the patient description data.

Return type dict

anesplot.loadrec.loadtelevet module

Created on Wed Jul 31 16:22:06 2019 @author: cdesbois

load televet exported (csv) data: to be developped

`anesplot.loadrec.loadtelevet.choosefile_gui(dirpath: Optional[str] = None) → str`
select a file using a dialog

Parameters `dirpath` (*str, optional*) – location of the data, ex : paths['data']. (The default is None -> '~').

Returns full name of the selected file.

Return type str

`anesplot.loadrec.loadtelevet.loadtelevet(fname: Optional[str] = None, all_traces: bool = False) → pandas.core.frame.DataFrame`

load the televetCsvExportedFile

Parameters

- **fname** (*str, optional*) – (full) name of the file (default is None).
- **all_traces** (*bool, optional*) – load all the derivations (default is False).

Returns the recorded traces.

Return type pandas.DataFrame

Module contents

anesplot.plot package

Submodules

anesplot.plot.trend_plot module

Created on Tue Apr 19 09:08:56 2016 @author: cdesbois

collection of functions to plot the trend data

`anesplot.plot.trend_plot.remove_outliers(df: pandas.core.frame.DataFrame, key: str, limits: Optional[dict] = None) → pandas.core.series.Series`

remove outliers

Parameters

- **df** (*pd.DataFrame*) – the data.
- **key** (*str*) – a column label to extract the trace.
- **limits** (*dict, optional*) – {limLow: val, limHigh:val} (default is None).

Returns `ser` – data without the outliers.

Return type pandas.Series

`anesplot.plot.trend_plot.color_axis(ax: matplotlib.axes._axes.Axes, spine: str = 'bottom', color: str = 'r')`
change the color of the label & tick & spine.

Parameters

- **ax** (*plt.Axes*) – the axis to work on.
- **spine** (*str, optional (default is "bottom")*) – optional location in ['bottom', 'left', 'top', 'right'] .
- **color** (*str, optional (default is "r")*) – the color to use.

Return type None.

`anesplot.plot.trend_plot.append_loc_to_fig(ax: matplotlib.axes._axes.Axes, dt_list: list, label: str = 'g')`
→ dict

append vertical lines to indicate a time location 'for eg: arterial blood gas'

Parameters

- **ax** (*plt.Axes*) – the axis to add on.
- **dt_list** (*list*) – list of datetime values.
- **label** (*str, optional (default is 'g')*) – a key to add to the label.

Returns a dictionary containing the locations.

Return type dict

`anesplot.plot.trend_plot.save_graph(path: str, ext: str = 'png', close: bool = True, verbose: bool = True)`
Save a figure from pyplot

Parameters

- **path** (*str*) – The path (and filename, without the extension) to save the figure to.
- **ext** (*str, optional (default='png')*) – The file extension. This must be supported by the active matplotlib backend (see matplotlib.backends module). Most backends support 'png', 'pdf', 'ps', 'eps', and 'svg'.
- **close** (*bool, optional (default=True)*) – Whether to close the figure after saving. If you want to save the figure multiple times (e.g., to multiple formats), you should NOT close it in between saves or you will have to re-plot it.
- **verbose** (*bool, optional (default=True)*) – Whether to print information about when and where the image has been saved.

Return type None.

`anesplot.plot.trend_plot.plot_header(descr: dict, param: Optional[dict] = None)` →
matplotlib.figure.Figure

plot the header of the file.

Parameters

- **descr** (*dict*) – header of the recording.
- **param** (*dict, optional (default is None)*) – dictionary of parameters. .

Returns **fig** – plot of the header.

Return type pyplot.Figure

`anesplot.plot.trend_plot.hist_cardio(data: pandas.core.frame.DataFrame, param: Optional[dict] = None)` → matplotlib.figure.Figure

mean arterial pressure histogramme using matplotlib.

Parameters

- **data** (*pd.DataFrame*) – the recorded trends data(keys used : ‘ip1m’ and ‘hr’),.
- **param** (*dict, optional (default is None).*) – parameters (save=boolean, ‘path’: path to directory).

Returns matplotlib.pyplot.figure.

Return type TYPE

anesplot.plot.trend_plot.**plot_one_over_time**(*x, y, colour: str*) → matplotlib.figure.Figure
plot y over x using colour

Parameters

- **x** (*float*) –
- **y** (*float*) –
- **colour** (*str*) – matplotlib.pyplot color

Returns fig

Return type pyplot.figure()

anesplot.plot.trend_plot.**hist_co2_iso**(*data: pandas.core.frame.DataFrame, param: Optional[dict] = None*) → matplotlib.figure.Figure
plot CO2 and iso histogramme (NB CO2 should have been converted from % to mmHg)

Parameters

- **data** (*pd.DataFrame*) – the recorded data.
- **param** (*dict, optional (default is None).*) – parameters.

Returns matplotlib.pyplot.Figure.

Return type TYPE

anesplot.plot.trend_plot.**cardiovasc**(*data: pandas.core.frame.DataFrame, param: Optional[dict] = None*) → matplotlib.figure.Figure
cardiovascular plot

Parameters

- **data** (*pd.DataFrame*) – the recorded trends data, columns used :[‘ip1s’, ‘ip1m’, ‘ip1d’, ‘hr’].
- **param** (*dict, optional (default is None)*) –
dict(**save: boolean, path[‘save’], xmin, xmax, unit, dtm** = boolean for time display in HH:MM format)

Return type matplotlib.pyplot.Figure

anesplot.plot.trend_plot.**cardiovasc_p1p2**(*data: pandas.core.frame.DataFrame, param: Optional[dict] = None*) → pandas.core.frame.DataFrame
cardiovascular plot with central venous pressure (p2)

Parameters

- **data** (*pd.DataFrame*) –
the trends recorded data columns used :[‘ip1s’, ‘ip1m’, ‘ip1d’, ‘hr’, ‘ip2s’, ‘ip2m’, ‘ip2d’].
- **param** (*dict, optional (default is None)*) –

dict(save: boolean, path['save'], xmin, xmax, unit, dttime = boolean for time display in HH:MM format).

Returns matplotlib.pyplot.Figure.

Return type TYPE

`anesplot.plot.trend_plot.co2iso(data: pandas.core.frame.DataFrame, param: Optional[dict] = None) → matplotlib.figure.Figure`

plot CO2/iso over time

Parameters

- **data** (*pd.DataFrame*) – the recorded data. Columns used :['ip1s', 'ip1m', 'ip1d', 'hr'].
- **param** (*dict, optional (default is None).*) –

dict(save: boolean, path['save'], xmin, xmax, unit, dttime = boolean for time display in HH:MM format)

Return type matplotlib.pyplot.Figure

`anesplot.plot.trend_plot.func(ax, x, y1, y2, color='tab:blue', x0=38)`

`anesplot.plot.trend_plot.co2o2(data: pandas.core.frame.DataFrame, param: dict) → matplotlib.figure.Figure`

respiratory plot : CO2 and Iso

Parameters

- **data** (*pd.DataFrame*) – recorded trends data columns used :['co2insp', "co2exp", "o2insp", "o2exp"].
- **param** (*dict*) –

dict(save: boolean, path['save'], xmin, xmax, unit, dttime = boolean for time display in HH:MM format).

Returns matplotlib.pyplot.Figure

Return type TYPE

`anesplot.plot.trend_plot.ventil(data: pandas.core.frame.DataFrame, param=<class 'dict'>) → matplotlib.figure.Figure`

plot ventilation

Parameters

- **data** (*pd.DataFrame*) – recorded trend data columns used : (tvInsp, pPeak, pPlat, peep, minVexp, co2RR, co2exp)
 - **param** (*dict, optional*) –
- param: dict(save: boolean, path['save'], xmin, xmax, unit, dttime = boolean for time display in HH:MM format)**

Returns fig

Return type matplotlib.pyplot.Figure

`anesplot.plot.trend_plot.recruit(data: pandas.core.frame.DataFrame, param: dict) → matplotlib.figure.Figure`

display a recruit manoeuver

Parameters

- **data** (*pd.DataFrame*) – recorded trend data. Columns used : (pPeak, pPlat, peep, tvInsp)

- **param** (*dict*) –

dict(**save**: **boolean**, **path**['save'], **xmin**, **xmax**, **unit**, **dtime** = **boolean** for time display in HH:MM format)

Returns **fig**

Return type matplotlib.pyplot.Figure

`anesplot.plot.trend_plot.ventil_cardio`(*data: pandas.core.frame.DataFrame, param: dict*) →
matplotlib.figure.Figure

build ventilation and cardiovascular plot

Parameters

- **data** (*pd.DataFrame*) – trend data. Columns used : ['ip1s', 'ip1m', 'ip1d', 'hr']

- **param** (*dict*) –

dict(**save**: **boolean**, **path**['save'], **xmin**, **xmax**, **unit**, **dtime** = **boolean** for time display in HH:MM format).

Returns **fig**

Return type matplotlib.pyplot.Figure

`anesplot.plot.trend_plot.sat_hr`(*data: pandas.core.frame.DataFrame, param: dict*) →
matplotlib.figure.Figure

plot a sat and sat_hr over time

Parameters

- **taphdata** (*pd.DataFrame*) – the taph recording
- **dtime** (*boolean, optional (default is True)*) – plot over datetime (or elapsed time)

Returns **fig** – DESCRIPTION.

Return type TYPE

`anesplot.plot.trend_plot.save_distri`(*data: pandas.core.frame.DataFrame, path: dict*)
save as 'O_.' the 4 distributions graphs for cardiovasc and respi

Parameters

- **data** (*pd.DataFrame*) – trends data.

- **path** (*dict*) –

dict(**save**: **boolean**, **path**['save'], **xmin**, **xmax**, **unit**, **dtime** = **boolean** for time display in HH:MM format)..

Return type None.

`anesplot.plot.trend_plot.fig_memo`(*apath: str, fig_name: str*)
append latex frame command in a txt file inside the fig folder create the file if it doesn't exist

Parameters

- **apath** (*str*) – dirname to export to.
- **fig_name** (*str*) – figure short name.

Return type None.

anesplot.plot.wave_plot module

Created on Tue Apr 19 09:08:56 2016

@author: cdesbois

anesplot.plot.wave_plot.color_axis(*ax: matplotlib.axes._axes.Axes, spine: str = 'bottom', color: str = 'r'*)
change the color of the label & tick & spine.

Parameters

- **ax** (*plt.Axes*) – the axis to work on.
- **spine** (*str, optional (default is "bottom")*) – location in ['bottom', 'left', 'top', 'right']
- **color** (*str, optional (default is "r")*) – color to use

Return type None.

anesplot.plot.wave_plot.plot_wave(*data: pandas.core.frame.DataFrame, keys: list, param: dict*) → *matplotlib.figure.Figure*
plot the waves recorded (from as5) (Nb plot data/index, but the xscale is indicated as sec)

Parameters

- **data** (*pd.DataFrame*) – recorded waves data.
- **keys** (*list*) – one or two in ['wekg', 'ECG', 'wco2', 'wawp', 'wflow', 'wap'].
- **param** (*dict*) – {mini: limits in point value (index), maxi: limits in point value (index)}.

Return type *matplotlib.pyplot.Figure*

anesplot.plot.wave_plot.get_roi(*fig: matplotlib.figure.Figure, df: pandas.core.frame.DataFrame, params: dict*) → *dict*

use the drawn figure to extract the relevant data in order to build an animation

Parameters

- **fig** (*plt.Figure*) – the figure to get data from.
- **df** (*pd.DataFrame*) – waves recording.
- **params** (*dict of parameters*) –

Returns containing ylims, xlims(point, dtime and sec)

Return type *dict*

anesplot.plot.wave_plot.create_video(*data: pandas.core.frame.DataFrame, param: dict, roi: dict, speed: int = 1, save: bool = False, savename: str = 'example', savedir: str = '~'*)

create a video from a figure

Parameters

- **data** (*pd.DataFrame*) – waves data.
- **param** (*dict*) – recording parameters.
- **roi** (*dict*) – containing ylims, xlims(point, dtime and sec).
- **speed** (*int, optional (default is 1)*) – speed of the video.
- **save** (*bool, optional (default is False)*) – to save or not to save.
- **savename** (*str, optional (default is "example")*) – save (short) name.

- **savedir**(*str*, *optional* (default is "~").) – save dirname (full).

Returns

- *.mp4 file*
- *.png file*

Module contents

Created on Tue Apr 19 09:08:56 2016

functions to plot the trend data

@author: cdesbois

anesplot.treatrec package

Submodules

anesplot.treatrec.clean_data module

Created on Wed Jul 31 16:05:29 2019

@author: cdesbois

`anesplot.treatrec.clean_data.clean_trenddata(df)`
remove artifacts in the recorded trends

anesplot.treatrec.ekg_to_hr module

Created on Wed Feb 12 16:52:00 2020 @author: cdesbois

function used to treat an EKG signal and extract the heart rate typically (copy, paste and execute line by line)

0. after

```
:: import pandas as pd
    import anesplot.record_main as rec from anesplot.treatrec import ekg_to_hr as tohr
```

1. load the data in a pandas dataframe:

(through classes `rec.MonitorTrend` & `rec.MonitorWave`)

```
trendname = '' # fullname
or
trendname = rec.choosefile_gui()
```

```
wavename = rec.trendname_to_wavename(trendname)
-
# load the data
trends = rec.MonitorTrend(trendname)
waves = rec.MonitorWave(wavename)
-
# format the name
name = trends.header['Patient Name'].title().replace(' ', '')
name = name[0].lower() + name[1:]
```

2. treat the ekg wave:

- get parameters
- build a dataframe to work with (waves)
- low pass filtering
- build the beat locations (beat based dataframe):

```
params = waves.param
ekg_df = pd.DataFrame(waves.data.wekg)
ekg_df['wekg_lowpass'] = rec.wf.fix_baseline_wander(ekg_df.wekg,
                                                    waves.param['sampling_freq'])
beat_df = tohr.detect_beats(ekg_df.wekg_lowpass, mult=1)
```

3. perform the manual adjustments required:

- based on a graphical display of beat locations, an rr values
- build a container for the manual corrections:

```
figure = tohr.plot_beats(ekg_df.wekg_lowpass, beat_df)
to_change_df = pd.DataFrame(columns=beat_df.columns.insert(0, 'action'))
```

- remove or add peaks : zoom on the figure to observe only one peak, then:

```
to_change_df = tohr.remove_beat(beat_df, ekg_df, to_change_df, figure)
or
to_change_df = tohr.append_beat(beat_df, ekg_df, to_change_df, figure,
                                yscale=1)
```

- combine to update the beat_df with the manual changes:

```
beat_df = tohr.update_beat_df(beat_df, to_change_df,
                              path_to_file="", from_file=False)
```

- save the peaks locations:

```
tohr.save_beats(beat_df, to_change_df, savename='', savepath=None)
(# or reload
beat_df = pd.read_hdf('beatDf.hdf', key='beatDf') )
```

4. go from points values to continuous time:

```
beat_df = tohr.compute_rr(beat_df)
ahr_df = tohr.interpolate_rr(beat_df)
tohr.plot_rr(ahr_df, params)
```

5. append intantaneous heart rate to the initial data:

```
ekg_df = tohr.append_rr_and_ihr_to_wave(ekg_df, ahr_df)
waves.data = tohr.append_rr_and_ihr_to_wave(waves.data, ahr_df)
trends.data = tohr.append_ihr_to_trend(trends.data, waves.data, ekg_df)
```

6. save:

```
tohr.save_trends_data(trends.data, savename=name, savepath='data')
tohr.save_waves_data(waves.data, savename=name, savepath='data')
```

`anesplot.treatrec.ekg_to_hr.detect_beats`(*ser: pandas.core.series.Series, fs: int = 300, species: str = 'horse', mult: float = 1*) → `pandas.core.frame.DataFrame`

detect the peak locations of the beats

Parameters

- **ser** (*pd.Series*) – the EKG time series.
- **fs** (*int, optional (default is 300)*) – sampling frequency.
- **species** (*str, optional (default is "horse")*) – the species.
- **mult** (*float, optional (default is 1)*) – correction / 1 for qRs amplitude.

Returns df

Return type `pandas.DataFrame`

`anesplot.treatrec.ekg_to_hr.plot_beats`(*ekgdf: pandas.core.frame.DataFrame, beatdf: pandas.core.frame.DataFrame*) → `matplotlib.figure.Figure`

plot beat location on ekg display and rr values over time

Parameters

- **ekgdf** (*pd.DataFrame.*) – waves data (wekg & wekg_lowpass)
- **beatdf** (*pd.DataFrame*) – the location of the beats (columns used are [p_loc and y_loc]).

Returns fig

Return type `matplotlib.pyplot.Figure`

`anesplot.treatrec.ekg_to_hr.append_beat`(*beatdf: pandas.core.frame.DataFrame, ekgdf: pandas.core.frame.DataFrame, tochangedf: pandas.core.frame.DataFrame, fig: matplotlib.figure.Figure, lim: Optional[Tuple] = None, yscale: float = 1*) → `pandas.core.frame.DataFrame`

append a beat coordonate from the figure to the tochangedf[‘toAppend’]

Parameters

- **beatdf** (*pd.DataFrame*) – beat position (point based location : p_locs)
- **ekgdf** (*pd.DataFrame*) – waves data (wekg_lowpass).
- **tochangedf** (*pd.DataFrame*) – the beat to add or remove (point based toAppend & toRemove)
- **fig** (*plt.Figure*) – the figure to get the location.
- **lim** (*TYPE, optional (default is None)*) – ptBasedLim optional to give it manually
- **yscale** (*TYPE, optional (default is 1)*) – amplitude mutliplication factor for detection.

Returns **tochangedf** – incremented changedf (pt location).

Return type *pd.DataFrame*

methods :

locate the beat in the figure, append to a dataframe['toAppend'] 0.: if not present : build a dataframe:

```
>>> to_change_df = pd.DataFrame(columns=['toAppend', 'toRemove'])
```

1.: locate the extra beat in the figure (cf plot_beats()) and zoom to observe only a negative peak

2.: call the function:

```
>>> to_change_df = remove_beat(beatdf, ekgdf, tochangedf, fig)
-> the beat parameters will be added the dataframe
```

.in the end of the manual check, update the beat_df

- first : save beat_df and to_change_df
- **second** [run:]

```
>>> beat_df = update_beat_df()
```

`anesplot.treatrec.ekg_to_hr.remove_beat`(*beatdf: pandas.core.frame.DataFrame, ekgdf: pandas.core.frame.DataFrame, tochangedf: pandas.core.frame.DataFrame, fig: matplotlib.pyplot.figure, lim: Optional[Tuple] = None*) → *pandas.core.frame.DataFrame*

remove a beat coordinate from the figure to the tochangedf['toRemove']

Parameters

- **beatdf** (*pd.DataFrame*) – beat position (point based location : p_locs)
- **ekgdf** (*pd.DataFrame*) – waves data (wekg_lowpass).
- **tochangedf** (*pd.DataFrame*) – the beat to add or remove (point based toAppend & toRemove)
- **fig** (*plt.Figure*) – the figure to get the location.
- **lim** (*TYPE, optional (default is None)*) – ptBasedLim optional to give it manually
- **yscale** (*TYPE, optional (default is 1)*) – amplitude mutliplication factor for detection.

Returns

- **tochangedf** (*pd.DataFrame*)
- *incremented changedf (pt location).*
- *locate the beat in the figure, append to a dataframe['toRemove']*
- **0.** (*if not present build a dataframe:*) – `>>> to_change_df = pd.DataFrame(columns=['toAppend', 'toRemove'])`
- **1.** (*locate the extra beat in the figure (cf plot_beats())*) – and zoom to observe only a negative peak
- **2.** (*call the function:::*) – `>>> to_change_df = remove_beat(beatdf, ekgdf, tochangedf, fig)`
–> the beat parameters will be added the DataFrame
- *.(in the end of the manual check, update the beat_df –*
– first : save beat_df and to_change_df
– **second** [run]

```
>>> beat_df = update_beat_df())
```

`anesplot.treatrec.ekg_to_hr.save_beats(beatdf: pandas.core.frame.DataFrame, tochangedf: pandas.core.frame.DataFrame, savename: str = "", dirpath: Optional[str] = None)`

save the beats locations as csv and hd5 file

Parameters

- **beatde** (*pd.dataframes*) –
- **tochangedf** (*pandas.dataframe*) –
- **savename** (*filename*) –
- **dirpath** (*path to save in*) –
- **output** –
- **-----** –
- **file** (*hdf*) –
- **key='beatDf'** –

`anesplot.treatrec.ekg_to_hr.update_beat_df(beatdf: pandas.core.frame.DataFrame, tochangedf: pandas.core.frame.DataFrame, path_to_file: str = "", from_file: bool = False)`

implement in the beat location the manual corrections

Parameters

- **beatdf** (*pd.DataFrame*) – beat position (point based location : p_locs)
- **tochangedf** (*pd.DataFrame*) – the beat to add or remove (point based toAppend & toRemove)
- **path_to_file** (*str, optional (default is "")*) – dirpath to the saved file.
- **from_file** (*bool, optional (default is False)*) – fromFile = True force the disk loading of the dataframes

Returns **beatdf** – updated beat position

Return type *pd.DataFrame*

`anesplot.treatrec.ekg_to_hr.compute_rr(beatdf: pandas.core.frame.DataFrame, fs: Optional[int] = None)`
 \rightarrow pandas.core.frame.DataFrame

compute rr intervals (from pt to time)

Parameters

- **beatdf** (*pd.DataFrame*) – beat position (point based location : p_locs)
- **fs** (*int, optional (default is None -> 300)*) – the sampling frequency

Returns **beatdf** – beat position updated with rrvalues: ‘rr’ = rr duration ‘rrDiff’ = rrVariation ‘rrSqDiff’ = rrVariation^2

Return type *pd.DataFrame*

`anesplot.treatrec.ekg_to_hr.interpolate_rr(beatdf: pandas.core.frame.DataFrame, kind: Optional[str] = None)`
 \rightarrow pandas.core.frame.DataFrame

interpolate the beat_df (pt -> time values)

Parameters

- **beatdf** (*pd.DataFrame*) – beat position (point based location : p_locs).
- **kind** (*str, optional (default is None -> "cubic")*) – interpolation (in ['linear', 'cubic'])

Returns **ahr_df** – evenly spaced data with ‘espts’ = evenly spaced points & ‘rrInterpol’ = interpolated rr

Return type *pd.DataFrame*

`anesplot.treatrec.ekg_to_hr.plot_rr(ahr_df: pandas.core.frame.DataFrame, param: dict, HR: bool = False)`
 \rightarrow matplotlib.figure.Figure

plot RR vs pt values + rrSqDiff

Parameters

- **ahr_df** (*pd.DataFrame*) – DESCRIPTION.
- **param** (*dict*) – containing ‘sampling_freq’ as key.
- **HR** (*bool, optional (default is False)*) – to display HR instead of rr

Returns **fig**

Return type *plt.Figure*

`anesplot.treatrec.ekg_to_hr.append_rr_and_ihr_to_wave(ekgdf: pandas.core.frame.DataFrame, ahrdf: pandas.core.frame.DataFrame)`
 \rightarrow pandas.core.frame.DataFrame

append rr and ihr to the waves based on pt value (ie index)

Parameters

- **ekgdf** (*pd.DataFrame*) – waves data
- **ahrdf** (*pd.DataFrame*) – evenly spaced interpolated data.

Returns **df** – added iHR to ekgdf.

Return type *pd.DataFrame*

`anesplot.treatrec.ekg_to_hr.plot_agreement(trenddf: pandas.core.frame.DataFrame)`
 plot ip1HR & ihr to check agreement

`anesplot.treatrec.ekg_to_hr.append_ihr_to_trend(trenddf: pandas.core.frame.DataFrame, wavedf: pandas.core.frame.DataFrame, ekgdf: pandas.core.frame.DataFrame) → pandas.core.frame.DataFrame`

append 'ihr' (instataneous heart rate) to the trends

Parameters

- **trenddf** (*pd.DataFrame*) – DESCRIPTION.
- **wavedf** (*pd.DataFrame*) – DESCRIPTION.
- **ekgdf** (*pd.DataFrame*) – DESCRIPTION.

Returns **trenddf** – DESCRIPTION.

Return type TYPE

`anesplot.treatrec.ekg_to_hr.save_trends_data(trenddf: pandas.core.frame.DataFrame, savename: str = "", dirpath: str = 'data')`

save the trends data to a csv and hd5 file, including an ihr column (key='trends_data')

Parameters

- **trenddf** (*pd.DataFrame*) – the (updated) trend recording.
- **savename** (*str, optional (default is "" -> _trendData)*) – (short) file name to use
- **dirpath** (*str, optional*) – DESCRIPTION. The default is cwd.

Return type None.

`anesplot.treatrec.ekg_to_hr.save_waves_data(wavedf: pandas.core.frame.DataFrame, savename: str = "", dirpath: str = 'data')`

save the waves data to a csv and hd5 file, including an ihr column (key='waves_data')

Parameters

- **wavedf** (*pd.DataFrame*) – the (updated) trend recording.
- **savename** (*str, optional (default is "" -> _trendData)*) – (short) file name to use
- **dirpath** (*str, optional*) – DESCRIPTION. The default is cwd.

Return type None.

anesplot.treatrec.extract_hypotension module

Spyder Editor

This is a temporary script file.

`anesplot.treatrec.extract_hypotension.extract_hypotension(atrend, pamin=70)`

return a dataframe with the beginning and ending phses of hypotension

Parameters

- **atrend** (*MonitorTrend object*) –
- **pamin** (*float= threshold de define hypotension on mean arterial pressure*) –
- **70** (*((default is)*) –

Returns **durdf** – transitions (up and down, in seconds from beginning) and duration in the hypotension state (in seconds)

Return type pandas DataFrame containing

`anesplot.treatrec.extract_hypotension.plot_hypotension(atrend, durdf, durmin=15, pamin=70)`
plot the hypotensions phases

Parameters

- **atrend** (*TYPE*) – DESCRIPTION.
- **durdf** (*TYPE*) – DESCRIPTION.
- **durmin** (*TYPE, optional*) – DESCRIPTION. The default is 15.

Returns **fig** – DESCRIPTION.

Return type TYPE

`anesplot.treatrec.extract_hypotension.scatter_length_meanhypo(atrend, durdf)`
draw a scatter plot (hypotensive arterial value vs duration of hypotension) :param trends: :type trends: MonitorTrend :param durdf: :type durdf: pandas dataframe containing the value and duration

Returns **fig**

Return type matplotlib.pyplot figure

`anesplot.treatrec.extract_hypotension.plot_all_dir_hypo(dirname=None, scatter=False)`
walk through the folder and plot the values

anesplot.treatrec.hr_to_hrv module

`anesplot.treatrec.hr_to_hrv.build_hrv_limits(spec='horse')`
return a dico containing HRV limits (VLF, LF, HF) input : spec in ['horse', 'man']

anesplot.treatrec.wave_func module

Created on Fri Dec 8 12:46:41 2017

@author: cdesbois

`anesplot.treatrec.wave_func.fix_baseline_wander(data, fs=500)`
BaselineWanderRemovalMedian.m from ecg-kit. Given a list of amplitude values (data) and sample rate (sr), it applies two median filters to data to compute the baseline. The returned result is the original data minus this computed baseline.

`anesplot.treatrec.wave_func.rol_mean(ser, win_lengh=1, fs=500)`
returns a rolling mean of a RR serie

Parameters

- **pd.Series** (*ser=*) –
- **win_lengh** (*integer*) – window lenght for averaging (in sec),
- **fs** (*int*) – sampling frequency

`anesplot.treatrec.wave_func.return_points(df, fig)`
return a tuple containing the point values of ROI

Parameters

- **df** (*anesthesia record dataframe*) –
- **fig** (*pyplot.figure*) –

Returns ROI

Return type dict

`anesplot.treatrec.wave_func.restrict_time_area(df1, mini=None, maxi=None)`
return a new dataframe with reindexation

Parameters

- **df1** (*pandas.DataFrame*) –
- **mini** (*integer*) – miniPointValue
- **maxi** (*integer*) – maxiPointValue

Return type pandas.DataFrame

Module contents

3.1.2 Submodules

3.1.3 Module contents

anesthPlot is a package to plot/use clinical anesthesia records for teaching

three way to use it:

1. **run directly anesthplot from a terminal** -> PYTHONPATH=<pathToAnesthPlot> python -m anesthplot -> generate a quick plotting of most interesting parts (e.g. to use during an anesthesia debriefing session)
2. **from an ipython terminal** -> import anesthPlot.anesplot.recordmain as rec -> mtrends = rec.MonitorTrend() -> waves = rec.MonitorWave() -> ttrends = rec.TaphTrend() -> ... and use the objects trends and waves
3. import the module in a python environment (see below)

(the presets are actually designed

- for use with equine anesthesia
- to load data from a Monitor generated datex AS3/5 monitoring machine)

typical use when importing the module to build a clinical case

```
import os
import sys
```

```
import numpy as np
import pandas as pd
```

```
import anesthplot.record_main as rec
sys.path.append(os.path.expanduser('~/.pg/utills'))
from utills import saveGraph
import bloodGases.bgmain_manual as bgman
```

```
paths = rec.paths
paths['save'] = os.path.expanduser('~/.toPlay/temp/')
os.chdir(paths['save'])
```

```
## globals
def save_plot(name):
```

```
    filename = os.path.join(paths['save'], 'fig', name)
    saveGraph(filename, ext='png', close=False, verbose=True)
```

```
def explore_hdf(filename):
```

```
    try:
        hdf = pd.HDFStore(filename)
        keys= [key.replace('/', '') for key in hdf.keys()]
        print(' found h5_file { }
```

```
        that contains {}'.format(filename, keys))
    hdf.close()
    except: print('{} is not an h5 file'.format(filename))
saveName = os.path.join(paths['save'], 'data', 'aname.h5')
explore_hdf(saveName)
## load and work trendName = rec.choosefile_gui(paths['data']) WaveName = rec.choosefile_gui(paths['data'])
# build objects with headers trends = rec.MonitorTrend(trendName, load=True) waves = rec.MonitorWave(waveName,
load=True
# or append data (pretreated ones) #trends.data = pd.read_hdf(saveName, 'trend_df') #waves.data =
pd.read_hdf(saveName, 'wave_df')
#remove filenames del waveName, trendName
# now you are ready to work with loaded trends and waves
anesplot.get_basic_debrief_commands()
```

INDICES AND TABLES

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

PYTHON MODULE INDEX

a

- `anesplot`, 21
- `anesplot.config`, 6
 - `anesplot.config.build_recordrc`, 5
 - `anesplot.config.load_recordrc`, 5
- `anesplot.loadrec`, 9
 - `anesplot.loadrec.explore`, 6
 - `anesplot.loadrec.loadmonitor_trendrecord`, 6
 - `anesplot.loadrec.loadmonitor_waverecord`, 7
 - `anesplot.loadrec.loadtaph_trendrecord`, 7
 - `anesplot.loadrec.loadtelevet`, 9
- `anesplot.plot`, 13
 - `anesplot.plot.trend_plot`, 9
 - `anesplot.plot.wave_plot`, 12
- `anesplot.record_main`, 3
- `anesplot.treatrec`, 21
 - `anesplot.treatrec.clean_data`, 14
 - `anesplot.treatrec.ekg_to_hr`, 14
 - `anesplot.treatrec.extract_hypotension`, 19
 - `anesplot.treatrec.hr_to_hrv`, 20
 - `anesplot.treatrec.wave_func`, 20

A

[adapt_with_syspath\(\)](#) (in module *anesplot.config.load_recordrc*), 6
[anesplot](#)
 module, 21
[anesplot.config](#)
 module, 6
[anesplot.config.build_recordrc](#)
 module, 5
[anesplot.config.load_recordrc](#)
 module, 5
[anesplot.loadrec](#)
 module, 9
[anesplot.loadrec.explore](#)
 module, 6
[anesplot.loadrec.loadmonitor_trendrecord](#)
 module, 6
[anesplot.loadrec.loadmonitor_waverecord](#)
 module, 7
[anesplot.loadrec.loadtaph_trendrecord](#)
 module, 7
[anesplot.loadrec.loadtelevet](#)
 module, 9
[anesplot.plot](#)
 module, 13
[anesplot.plot.trend_plot](#)
 module, 9
[anesplot.plot.wave_plot](#)
 module, 12
[anesplot.record_main](#)
 module, 3
[anesplot.treatrec](#)
 module, 21
[anesplot.treatrec.clean_data](#)
 module, 14
[anesplot.treatrec.ekg_to_hr](#)
 module, 14
[anesplot.treatrec.extract_hypotension](#)
 module, 19
[anesplot.treatrec.hr_to_hrv](#)
 module, 20
[anesplot.treatrec.wave_func](#)

 module, 20

[append_beat\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 16
[append_ihr_to_trend\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 18
[append_loc_to_fig\(\)](#) (in module *anesplot.plot.trend_plot*), 10
[append_rr_and_ihr_to_wave\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 18

B

[build_hrv_limits\(\)](#) (in module *anesplot.treatrec.hr_to_hrv*), 20
[build_paths\(\)](#) (in module *anesplot.config.load_recordrc*), 5
[build_taph_decodedate_dico\(\)](#) (in module *anesplot.loadrec.loadtaph_trendrecord*), 8

C

[cardiovasc\(\)](#) (in module *anesplot.plot.trend_plot*), 11
[cardiovasc_p1p2\(\)](#) (in module *anesplot.plot.trend_plot*), 11
[choose_taph_record\(\)](#) (in module *anesplot.loadrec.loadtaph_trendrecord*), 8
[choosefile_gui\(\)](#) (in module *anesplot.loadrec.loadmonitor_trendrecord*), 6
[choosefile_gui\(\)](#) (in module *anesplot.loadrec.loadmonitor_waverecord*), 7
[choosefile_gui\(\)](#) (in module *anesplot.loadrec.loadtelevet*), 9
[choosefile_gui\(\)](#) (in module *anesplot.record_main*), 3
[clean_trenddata\(\)](#) (in module *anesplot.treatrec.clean_data*), 14
[co2iso\(\)](#) (in module *anesplot.plot.trend_plot*), 11
[co2o2\(\)](#) (in module *anesplot.plot.trend_plot*), 11
[color_axis\(\)](#) (in module *anesplot.plot.trend_plot*), 9
[color_axis\(\)](#) (in module *anesplot.plot.wave_plot*), 12
[compute_rr\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 17
[create_video\(\)](#) (in module *anesplot.plot.wave_plot*), 13

D

`detect_beats()` (in module *anes-plot.treatrec.ekg_to_hr*), 16

E

`export_taph_events()` (*anes-plot.record_main.TaphTrend* method), 4

`extract_events()` (*anesplot.record_main.TaphTrend* method), 4

`extract_hypotension()` (in module *anes-plot.treatrec.extract_hypotension*), 19

`extract_record_day()` (in module *anes-plot.loadrec.loadtaph_trendrecord*), 8

F

`fig_memo()` (in module *anesplot.plot.trend_plot*), 12

`filedialog()` (in module *anes-plot.config.build_recordrc*), 5

`fix_baseline_wander()` (in module *anes-plot.treatrec.wave_func*), 20

`func()` (in module *anesplot.plot.trend_plot*), 11

G

`get_roi()` (in module *anesplot.plot.wave_plot*), 13

`gui_choosefile()` (in module *anes-plot.loadrec.explore*), 6

H

`hist_cardio()` (in module *anesplot.plot.trend_plot*), 10

`hist_co2_iso()` (in module *anesplot.plot.trend_plot*), 10

I

`interpolate_rr()` (in module *anes-plot.treatrec.ekg_to_hr*), 18

L

`loadmonitor_trenddata()` (in module *anes-plot.loadrec.loadmonitor_trendrecord*), 6

`loadmonitor_trendheader()` (in module *anes-plot.loadrec.loadmonitor_trendrecord*), 6

`loadmonitor_wavedata()` (in module *anes-plot.loadrec.loadmonitor_waverecord*), 7

`loadmonitor_waveheader()` (in module *anes-plot.loadrec.loadmonitor_waverecord*), 7

`loadtaph_patientfile()` (in module *anes-plot.loadrec.loadtaph_trendrecord*), 8

`loadtaph_trenddata()` (in module *anes-plot.loadrec.loadtaph_trendrecord*), 8

`loadtelevet()` (in module *anes-plot.loadrec.loadtelevet*), 9

M

`main()` (in module *anesplot.config.build_recordrc*), 5

`main()` (in module *anesplot.record_main*), 4

module

anesplot, 21

anesplot.config, 6

anesplot.config.build_recordrc, 5

anesplot.config.load_recordrc, 5

anesplot.loadrec, 9

anesplot.loadrec.explore, 6

anesplot.loadrec.loadmonitor_trendrecord, 6

anesplot.loadrec.loadmonitor_waverecord, 7

anesplot.loadrec.loadtaph_trendrecord, 7

anesplot.loadrec.loadtelevet, 9

anesplot.plot, 13

anesplot.plot.trend_plot, 9

anesplot.plot.wave_plot, 12

anesplot.record_main, 3

anesplot.treatrec, 21

anesplot.treatrec.clean_data, 14

anesplot.treatrec.ekg_to_hr, 14

anesplot.treatrec.extract_hypotension, 19

anesplot.treatrec.hr_to_hrv, 20

anesplot.treatrec.wave_func, 20

MonitorTrend (class in *anesplot.record_main*), 4

MonitorWave (class in *anesplot.record_main*), 4

P

`plot_agreement()` (in module *anes-plot.treatrec.ekg_to_hr*), 18

`plot_all_dir_hypo()` (in module *anes-plot.treatrec.extract_hypotension*), 19

`plot_beats()` (in module *anesplot.treatrec.ekg_to_hr*), 16

`plot_events()` (*anesplot.record_main.TaphTrend* method), 4

`plot_header()` (in module *anesplot.plot.trend_plot*), 10

`plot_hypotension()` (in module *anes-plot.treatrec.extract_hypotension*), 19

`plot_one_over_time()` (in module *anes-plot.plot.trend_plot*), 10

`plot_rr()` (in module *anesplot.treatrec.ekg_to_hr*), 18

`plot_trenddata()` (in module *anesplot.record_main*), 3

`plot_ventil_drive()` (*anes-plot.record_main.TaphTrend* method), 4

`plot_wave()` (in module *anesplot.plot.wave_plot*), 12

R

`read_config()` (in module *anes-plot.config.build_recordrc*), 5

`recrut()` (in module *anesplot.plot.trend_plot*), 12

[remove_beat\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 17
[remove_outliers\(\)](#) (in module *anesplot.plot.trend_plot*), 9
[restrict_time_area\(\)](#) (in module *anesplot.treatrec.wave_func*), 20
[return_points\(\)](#) (in module *anesplot.treatrec.wave_func*), 20
[rol_mean\(\)](#) (in module *anesplot.treatrec.wave_func*), 20

S

[save_beats\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 17
[save_distri\(\)](#) (in module *anesplot.plot.trend_plot*), 12
[save_graph\(\)](#) (in module *anesplot.plot.trend_plot*), 10
[save_trends_data\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 18
[save_waves_data\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 18
[scatter_length_meanhypo\(\)](#) (in module *anesplot.treatrec.extract_hypotension*), 19
[select_type\(\)](#) (in module *anesplot.record_main*), 3
[select_wave_to_plot\(\)](#) (in module *anesplot.record_main*), 3

T

[TaphTrend](#) (class in *anesplot.record_main*), 4
[TelevetWave](#) (class in *anesplot.record_main*), 4
[trendname_to_wavename\(\)](#) (in module *anesplot.record_main*), 3

U

[update_beat_df\(\)](#) (in module *anesplot.treatrec.ekg_to_hr*), 17

V

[ventil\(\)](#) (in module *anesplot.plot.trend_plot*), 11
[ventil_cardio\(\)](#) (in module *anesplot.plot.trend_plot*), 12

W

[write_configfile\(\)](#) (in module *anesplot.config.build_recordrc*), 5