anesthPlot

Release beta

Christophe Desbois

Apr 12, 2022

MAIN$_{S}CRIPT$:

anesthPlot is a python package developped to extract, manipulate and plots **anesthesia data recorded during equine anesthesia**. Is is using recording generated by the Monitor Software or the Taphonius equine anesthesia machine.

The main purpose is an usage in a **teaching environment** (briefing-debriefing approach), but this package can also facilite recording manipulation for other purposes.

Warning: This is a work in progres,

- the processes are mainly focused on horses anesthesia (default values)
- 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 .csv data extracted using a Televet holter system)

CHAPTER

ONE

FEATURES

- you can **load** recordings from a trend or a wave file
  - **from command line ('script mode'):**

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

    - this script approach will build a **standard plot series** for debriefing purposes:
      - global histograms (cardiovascular and anesthesia summary)
      - cardiovascular time based trends plots
      - respiratory time based trends plots
      - anesthesia time based trends plots
    - or will build a user selected **plot for wave** recording
      - one or two waves on the same plot (script usage, pop_up menu to choose)
  - **or by using the code as a python package ('import mode'):**

```
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
```

Hint: after **'import anesplot.record_main as rec'**

'rec.get_basic_debrief_commands()' will prefill the clipboard with this standard code

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

Hint: 'rec.get_guide()**'** allow the filling of the clipboard with standard approaches

CHAPTER

TWO

MAIN SCRIPT : ANESPLOT/__MAIN__.PY

Note:

- it is the entry point to the program (‘record_main.py’)
- it can be called directly from a terminal :

```
"python -m anesplot" or "python anesplot/__main__.py"
```

CHAPTER

THREE

MODULES

3.1 anesplot.loadrec package

3.1.1 Submodules

3.1.2 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.

3.1.3 anesplot.loadrec.export_reload module

Created on Tue Mar 8 11:09:29 2022

@author: cdesbois

export to hdf the records objects, and load them back

anesplot.loadrec.export_reload.**export_data_to_hdf**(*savename*, *mtrend=None*, *ttrend=None*, *mwave=None*)
export the recordings in an hdf file keys are

mtrends_data, mtrends_header, mtrends_param ttrends_data, ttrends_header, ttrends_param mwaves_data, mwaves_header, mwaves_param

Parameters

- **savename** (*str*) – the savename to use.
- **mtrend** (*MonitorTrend object, optional (default is None)*) – The Monitor recording
- **ttrend** (*TaphoniusTrend, optional (default is None)*) – the taphonius recording
- **mwave** (*MonitorWave, optional (default is None)*) – the wave recording

Return type None.

anesplot.loadrec.export_reload.**build_obj_from_hdf**(*savename: str*)
build MonitorTrend, TaphTrenbd and MonitorWave objects fill them from hdf file

Parameters **savename** (*str*) – the path to the saved hdf file.

Returns

- *MonitorTrend, TaphTrend and MonitorWave*
- *(empty objects if the corresponding keys are not present in the file)*

3.1.4 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 date 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

3.1.5 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

3.1.6 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 selct 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 ('SDYYYMMDD...'). (the headername will be reconstructed inside the function)

Returns the patient description data.

Return type dict

anesplot.loadrec.loadtaph_trendrecord.**shift_datetime**(*datadf: pandas.core.frame.DataFrame*, *minutes_to_add: Optional[int] = None*) → pandas.core.frame.DataFrame

add a datetime shift to the dataframe to compensate computer time shift (usually one hour)

Parameters

- **datadf** (*pd.DataFrame*) – a recording (that have to contain 'datetime' and 'time' column.
- **minutes_to_add** (*int, optional*) – DESCRIPTION. The default is None.

Returns **datadf** – the recording with shifted datetime and time columns.

Return type pd.DataFrame

anesplot.loadrec.loadtaph_trendrecord.**shift_elapsed_time**(*datadf: pandas.core.frame.DataFrame*, *minutes_to_add: Optional[int] = None*) → pandas.core.frame.DataFrame

add a elapsedtime shift to the dataframe to compensate recording start

Parameters

- **datadf** (*pd.DataFrame*) – a recording (that have to contain 'datetime' and 'time' column.
- **minutes_to_add** (*int, optional (default is None)*) –

Returns **datadf** – the recording with shifted eTime and eTimeMin columns.

Return type pd.DataFrame

anesplot.loadrec.loadtaph_trendrecord.**sync_elapsed_time**(*datetime_0: datetime.datetime*, *taphdatadf: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

use the first point of monitor recording to sync the taph elapsed time (s and min) !!! beware, datetime should be the same one the two devices ... or corrected !!!

Parameters

- **datetime_0** (*datetime.datetime*) – the 0 of the time (usually monitor-datadf.datetime.iloc[0]
- **taphdatadf** (*pd.DataFrame*) – the taph recording.

Returns **taphdatadf** – the corrected taph recording.

Return type pd.DataFrame

3.1.7 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

3.1.8 Module contents

3.2 anesplot.plot package

3.2.1 Submodules

3.2.2 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**(*datadf: pandas.core.frame.DataFrame*, *key: str*, *limits: Optional[dict] = None*) → pandas.core.series.Series

remove outliers

Parameters

- **datadf** (*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**(*ax0: 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**(*ax0: 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

- **ax0** (*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=bolean, '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**(*datadf: 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, dtime = boolean for time display in HH:MM format)

Return type matplotlib.pyplot.Figure

anesplot.plot.trend_plot.**cardiovasc_p1p2**(*datadf: 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, dtime = 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, optiona (default is None).*) –

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

Returns maplotlib.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, dtime = boolean for time display in HH:MM format)

Returns **fig**

Return type matplotlib.pyplot.Figure

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

display a recrut 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**(*datadf: pandas.core.frame.DataFrame*, *param: dict*) → matplotlib.figure.Figure

build ventilation and cardiovascular plot

Parameters

- **datadf** (*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**(*datadf: 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 annd 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.

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

Parameters

- **datadf** (*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*, *datadf: pandas.core.frame.DataFrame*, *params: dict*) → dict

use the drawn figure to extract the x and x limits

Parameters

- **fig** (*plt.Figure*) – the figure to get data from.
- **datadf** (*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*

3.2.4 Module contents

Created on Tue Apr 19 09:08:56 2016

functions to plot the trend data

@author: cdesbois

3.3 anesplot.treatrec package

3.3.1 Submodules

3.3.2 anesplot.treatrec.arterial_func module

Created on Tue Mar 29 13:00:08 2022

@author: cdesbois

anesplot.treatrec.arterial_func.**get_peaks**(*ser: pandas.core.series.Series*, *up: bool = True*, *annotations: bool = False*) → pandas.core.frame.DataFrame

extract a peak location from an arterial time series

Parameters

- **ser** (*pd.Series*) – arterial time series (val = arterial wave, index = sec).
- **up** (*bool, optional (default is True)*) – extraction of the 'up' peaks. (false -> 'down peaks')
- **annotations** (*bool, optional (default is False)*) – plot annotations peak and indications.

Returns **peaksdf** – 'ploc' & 'sloc' : point and second based beat location 'wap' & 'peak_heights' : the arterials values 'local_max' & 'local_min' : boolean for local maxima and minima

Return type pd.DataFrame that contains

anesplot.treatrec.arterial_func.**compute_systolic_variation**(*ser: pandas.core.series.Series*) → float

return the systolic variation : (maxi - mini) / mean

anesplot.treatrec.arterial_func.**plot_sample_systolic_pressure_variation**(*mwave*, *lims: Optional[Tuple] = None*, *teach: bool = False*, *annotations: bool = False*)

extract and plot the systolic pressure variation"

Parameters

- **mwave** (*monitor trend object*) – the monitor recording
- **lims** (*tuple, (default is None)*) – the limits to use (in sec) If none the mwave.roi will be used
- **teach** (*boolean (default is False)*) – if true added markers on the most relevant differences
- **annotations** (*boolean(default False)*) – if true plot all detected pulse

Returns **fig** – the matplotlib figure.

Return type plt.Figure

anesplot.treatrec.arterial_func.**median_filter**(*num_std=3*)

anesplot.treatrec.arterial_func.**plot_record_systolic_variation**(*mwave*, *annotations=False*)

plot systolic variation over the whole record

Parameters

- **mwave** (*rec.MonitorWave object*) – a monitor recording containing an arterial ('wap') recording.
- **annotations** (*bool (default=False)*) – add indications of peak detection in the graph

Returns

- **fig** (*pyplot.Figure*) – pressure, sys_var and hr plot
- **df** (*pandas.DataFrame*) – peaks locations and description.

anesplot.treatrec.arterial_func.**get_xlims**()

3.3.3 anesplot.treatrec.clean_data module

Created on Wed Jul 31 16:05:29 2019

@author: cdesbois

anesplot.treatrec.clean_data.**clean_trenddata**(*datadf*)
remove artifacts in the recorded trends

3.3.4 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)

(NB templates are available in the anesplot/guide folder

you can execute line by line in a file the following process: >> import anesplot.record_main as rec paths = rec.paths rec.get_guide(paths) -> fill the choice in the ipython terminal -> paste the template in the file <<)

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, threshold=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='', dirpath=None)
(# or reload
beat_df = pd.read_hdf('beatloc_df.hdf', key='beatlocdf') )
```

4. go from points values to continuous time:

```
beat_df = tohr.point_to_time_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, dirpath='data')
tohr.save_waves_data(waves.data, savename=name, dirpath='data')
```

---

anesplot.treatrec.ekg_to_hr.**detect_beats**(*ser: pandas.core.series.Series*, *fs: int = 300*, *species: str = 'horse'*, *threshold: 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.
- **threshold** (*float, optional (default is -1)*) – correction for qRs amplitude. (positive means higher than, negative means lower than)

Returns **df**

Return type pandas.DataFrame

anesplot.treatrec.ekg_to_hr.**plot_beats**(*ekgdf: pandas.core.frame.DataFrame*, *beatlocdf: 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)
- **beatlocdf** (*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**(*beatlocdf: 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

- **beatlocdf** (*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(beatlocdf, 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**(*beatlocdf: 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

- **beatlocdf** (*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(beatlocdf, 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**(*beatlocdf: pandas.core.frame.DataFrame*, *tochangedf: pandas.core.frame.DataFrame*, *savename: str = ''*, *dirpath: Optional[str] = None*, *csv: bool = False*)

save the beats locations as csv and hd5 file

Parameters

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

anesplot.treatrec.ekg_to_hr.**update_beatloc_df**(*beatlocdf: 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

- **beatlocdf** (*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 **beatlocdf** – updated beat position

Return type pd.DataFrame

anesplot.treatrec.ekg_to_hr.**point_to_time_rr**(*beatlocdf: pandas.core.frame.DataFrame*, *fs: Optional[int] = None*) → pandas.core.frame.DataFrame

compute rr intervals (from pt to time)

Parameters

- **beatlocdf** (*pd.DataFrame*) – beat position (point based location : p_locs)

- **fs** (*int, optional (default is None -> 300)*) – the sampling frequency

Returns **beatlocdf** – beat position updated with rrvalues: 'rr' = rr duration 'rrDiff' = rrVariation 'rrSqDiff' = rrVariation^2

Return type pd.DataFrame

anesplot.treatrec.ekg_to_hr.**interpolate_rr**(*beatlocdf: pandas.core.frame.DataFrame*, *kind: Optional[str] = None*) → pandas.core.frame.DataFrame

interpolate the beat_df (pt -> time values)

Parameters

- **beatlocdf** (*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*) → 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*) → 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 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.

3.3.5 anesplot.treatrec.extract_hypotension module

Created on Wed Jul 31 16:05:29 2019

@author: cdesbois

scan folders and check for hypotension

anesplot.treatrec.extract_hypotension.**extract_hypotension**(*atrend*, *pamin: int = 70*) → pandas.core.frame.DataFrame

return a dataframe with the beginning and ending phases of hypotension

Parameters

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

Returns **durdf** – transitionts (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: pandas.core.frame.DataFrame*, *durmin: int = 15*, *pamin: int = 70*) → matplotlib.figure.Figure

plot the hypotentions phases

Parameters

- **atrend** (*MonitorTrend*) – trend data.
- **durdf** (*pd.DataFrame*) – hypotension duration data.
- **durmin** (*int, optional (default is 15)*) – The minimal duration of an hypotension period

Returns **fig**

Return type plt.Figure

anesplot.treatrec.extract_hypotension.**scatter_length_meanhypo**(*atrend*, *durdf: pandas.core.frame.DataFrame*) → matplotlib.figure.Figure

draw a scatter plot (hypotensive arterial value vs duration of hypotension)

Parameters

- **atrend** (*MonitorTrend*) – the recorded trend data.
- **durdf** (*pd.DataFrame*) – value and duration of the hypotension periods.

Returns scatter plot.

Return type plt.Figure

anesplot.treatrec.extract_hypotension.**plot_all_dir_hypo**(*dirname: Optional[str] = None*, *scatter: bool = False*) → str

walk throught the folder and plot the values

Parameters

- **dirname** (*str, optional (default is None)*) – The name of the directory to scan
- **scatter** (*bool, optional (default is False)*) – generate a scatter plot or not

Returns **filename**

Return type str

3.3.6 anesplot.treatrec.hr_to_hrv module

Created on Wed Jul 31 16:05:29 2019

@author: cdesbois

rr to hrv … to be continued (see Yann work)

anesplot.treatrec.hr_to_hrv.**build_hrv_limits**(*spec='horse'*)

return a dico containing HRV limits (VLF, LF, HF) input : spec in [‘horse’, ‘man’]

3.3.7 anesplot.treatrec.manage_events module

Created on Sat Dec 18 10:30:54 2021

@author: cdesbois

to extract the events from the taphonius files

anesplot.treatrec.manage_events.**convert_day**(*txt: str*) → str
get a day YYYYmonthD an convert it to YYY-month-D

anesplot.treatrec.manage_events.**extract_taphmessages**(*df: pandas.core.frame.DataFrame*, *display: bool = False*) → Tuple[Any, Any]
extract the messages

Parameters

- **df** (*pd.DataFrame*) – dt_event_df.
- **display** (*bool, optional (default is False)*) – print the messages in the terminal

Returns

- *dict* – acts : dict of actions.
- *dict* – content: dict of taph messages

anesplot.treatrec.manage_events.**build_event_dataframe**(*datadf: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame
build a pandas datafame with a countinuous datetime:event pairs

Parameters **datadf** (*pd.DataFrame*) – the taphonius recording.

Returns **dteventsdf** – dataframe with index=datetime.

Return type pd.DataFrame

anesplot.treatrec.manage_events.**extract_ventilation_drive**(*dteventsdf: pandas.core.frame.DataFrame*, *acts: Optional[set] = None*) → pandas.core.frame.DataFrame
extract a dataframe containing the ventilatory management

Parameters

- **dteventsdf** (*pd.DataFrame*) – a container for taph generated events (dtime as index, event as column).
- **acts** (*set, optional (default is None)*) – container for action messages.

Return type pd.DataFrame with datetime index and one column per action (ex 'rr changed')

anesplot.treatrec.manage_events.**plot_ventilation_drive**(*df: pandas.core.frame.DataFrame*, *param: dict*, *all_traces: bool = False*) → matplotlib.figure.Figure
plot the ventilatory drive ie the data that were changed

Parameters

- **df** (*pd.DataFrame*) – ventildrive_df
- **param** (*dict*) – the recording parameters
- **all** (*bool (default is False)*) – to include {'buffer', 'ip', 'mwpl'}

Returns **fig**

Return type plt.Figure

anesplot.treatrec.manage_events.**plot_events**(*dteventsdf: pandas.core.frame.DataFrame*, *param: dict*, *todrop: Optional[list] = None*, *dtime: bool = False*) → matplotlib.pyplot.figure

plot all events

Parameters

- **dteventsdf** (*pd.DataFrame*) – the data with a datetime index, and an event column
- **param** (*dict*) – data recording parameters (just to get the filename)
- **todrop** (*list, optional (default is None)*) – str in the columns to drop the column
- **dtime** (*boolean (default is False)*) – to use dtime as the xscale.

Returns fig

Return type plt.Figure

anesplot.treatrec.manage_events.**extract_event**(*df: pandas.core.frame.DataFrame*) → dict

extract timestamp of the messages

Parameters **df** (*pd.DataFrame*) – pandasDataFrame containing the taphonius events.

Returns {message : [timestamp]}.

Return type dict

anesplot.treatrec.manage_events.**build_dataframe**(*acts*) → pandas.core.frame.DataFrame

build a dataframe containing all the actions, one per column

3.3.8 anesplot.treatrec.wave_func module

Created on Fri Dec 8 12:46:41 2017

@author: cdesbois

anesplot.treatrec.wave_func.**fix_baseline_wander**(*data: pandas.core.series.Series*, *fs: int = 500*) → pandas.core.series.Series

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.

Parameters

- **data** (*pd.DataFrame*) – the wave recording.
- **fs** (*int, optional (default is 500)*) – The sampling frequency.

Returns DESCRIPTION.

Return type list

anesplot.treatrec.wave_func.**rol_mean**(*ser: pandas.core.series.Series*, *win_lengh: int = 1*, *fs: int = 500*) → list

returns a rolling mean of a RR serie

Parameters

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

anesplot.treatrec.wave_func.**return_points**(*wavedf: pandas.core.frame.DataFrame*, *fig: matplotlib.figure.Figure*) → dict

return a tupple containing the point values of ROI

Parameters

- **wavedf** (*pd.DataFrame*) – teh wave recording.
- **fig** (*plt.Figure*) – the plot to extract the xscale from.

Returns the Region Of Interest.

Return type dict

anesplot.treatrec.wave_func.**restrict_time_area**(*df1: pandas.core.frame.DataFrame*, *mini: Optional[int] = None*, *maxi: Optional[int] = None*) → pandas.core.frame.DataFrame

return a new dataframe with reindexation

Parameters

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

Return type pandas.DataFrame

3.3.9 Module contents

3.4 anesplot.config package

3.4.1 Submodules

3.4.2 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=''*, *directory='/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.

3.4.3 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.

3.4.4 Module contents

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a