# anesthPlot

***Release beta***

**Christophe Desbois**

**Oct 07, 2021**

CHAPTER

ONE

# WELCOME TO ANESTHPLOTS 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 an as3 or as5 anesthesia machine monitoring ekg, invasive pressure, etCO2, halogenate, spirometry.

## 1.1 Features

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

```
python anesthPlot/anesplot/__main__.py
```

    - 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
    - build a **plot for wave** recording
      - one or two waves on the same plot (script usage)
- can also be used as a **python package**
  - **usage :**

```
import anesplot.record_main as rec
trendname = 'a_full_path_to_csv_file'
trends = rec.MonitorTrend(trendname)
wavename = rec.trendname_to_wavename(trendname)
waves = rec.MonitorWave(trends)
```

(continues on next page)

(continued from previous page)

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

waves.plot_waves() # -> one or two traces

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

CHAPTER

TWO

# 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 anestplot.record_main as rec

---

anesplot.record_main.**choosefile_gui**(*dir_path=None*)
Select a file via a dialog and return the (full) filename.

**Parameters** **dir_path** (*str*) – location to place the gui (generally paths[data]) else home

**Returns** **fname[0]** – filename

**Return type** str

anesplot.record_main.**trendname_to_wavename**(*name*)
just compute the supposed name

anesplot.record_main.**select_type**(*question=None*, *items=None*, *num=0*)
select the recording type:

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

**Return type** str

anesplot.record_main.**select_wave**(*waves*, *num=1*)
select the recording type:

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

**Return type** str

anesplot.record_main.**build_param_dico**(*file=None*, *asource=None*, *pathdico={'cwd': '/Users/cdesbois/pg/chrisPg/anesthPlot', 'data': '/Users/cdesbois/enva/clinique/recordings/anesthRecords/onPanelPcRecorded', 'recordMain': '/Users/cdesbois/pg/chrisPg/anesthPlot/anesplot', 'root': '/Users/cdesbois', 'sBg': '/Users/cdesbois/ownCloud', 'sFig': '/Users/cdesbois/ownCloud', 'save': '/Users/cdesbois/ownCloud', 'utils': '/Users/cdesbois/pg'}*)
initialise a dict save parameters -> TODO see min vs sec

**Parameters**

- **file** (*str*) – the recording filename

- **source** (*str*) – the origin of the recording

**Returns**

**dico** –

**a dictionary describing the situation** [item, xmin, xmax, ymin, ymax, path, unit, save, memo, file, source]

**Return type** dict

anesplot.record_main.**plot_trenddata**(*df*, *header*, *param_dico*)

clinical main plots of a trend recordings

parameters df : pdDataframe

recorded data (MonitorTrend.data)

**header** [dict] recording parameters (MonitorTrend.header)

**param_dico** [dict] plotting parameters (MonitorTrend.param)

**Returns** **afig_dico**

**Return type** dict of name:fig

**class** anesplot.record_main.**Waves**(*filename=None*)

Bases: object

the base object to store the records.

**class** anesplot.record_main.**SlowWave**(*filename=None*)

Bases: *anesplot.record_main.Waves*

class for slowWaves = trends

**file** [str] short name

**filename** [str] long name

**clean_trend : external**

clean the data

**show_graphs : external**

plot clinical main plots

**clean_trend**()

clean the data, remove irrelevant, input = self.data, output = pandas dataFrame nb doesnt change the obj.data in place

**show_graphs**()

basic clinical plots

**class** anesplot.record_main.**MonitorTrend**(*filename=None*, *load=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

**source** [str] recording apparatus (default = monitor)

**fs** [float] sampling rate

**param** [dict] display parameters

**clean_trend** [external] clean the data

**show_graphs** [external] plot clinical main plots

class anesplot.record_main.**TaphTrend**(*filename=None*)
Bases: *anesplot.record_main.SlowWave*

taphonius trends recordings

input FILLME

attributes FILLME

**load_header**()
load the header -> pandas.dataframe

**extract_taph_events**()
extract Taph events

**Parameters** **data** (*pandas dataframe*) – record df form taphonius recording)

**Returns** events dataframe

**Return type** eventdf pandas dataframe

class anesplot.record_main.**FastWave**(*filename=None*)
Bases: *anesplot.record_main.Waves*

class for Fastwaves = continuous recordings.

**plot_wave**(*traces_list=None*, *dtime=None*)
simple choose and plot for a wave input = none -> GUI, or list of waves to plot (max=2)

**define_a_roi**(*erase=False*)
define a Region Of Interest (roi).

input : erase (boolean) default=False takes the figure attribute return a dictionary containing:

dt : xscale datetime location pt: xscale point location sec: xscale seconde location ylims: ylimits traces: waves used to draw the figure fig : the related figure

**animate_fig**(*speed=1*, *save=False*, *savedir='~'*)
build a video the previous builded figure

use .fig attribute (builded through .plot_wave()) and .roi attribute (builded thourhg .define_a_roi())

**Parameters**

- **speed** (*int, optional*) – speed of the video, defaults to 1
- **save** (*boolean, optional*) – save or just display, defaults to False
- **savedir** (*str, optional*) – directory to save the animation, defaults to ~

**Returns** video file

**Return type** mp4

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

class to organise teleVet recordings transformed to csv files.

class anesplot.record_main.**MonitorWave**(*filename=None*, *load=True*)
Bases: *anesplot.record_main.FastWave*

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

attibutes FILLME

methods FILLME

anesplot.record_main.**main**()
main script called from command line call : python anesthPlot/anesplot/__main__.py args : optional filename (fullname)

return: set of plots for either monitorTrend, monitorWave oe televet recording

CHAPTER

THREE

# 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=''*, *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.

#### 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 date into a pandas dataframe

---

anesplot.loadrec.loadmonitor_trendrecord.**choosefile_gui**(*dir_path=None*)
select a file using a dialog.

**Parameters** **dir_path** (*str*) – optional location of the data (paths[data])

**Returns** filename (full path)

**Return type** str

anesplot.loadrec.loadmonitor_trendrecord.**loadmonitor_trendheader**(*filename*)
load the file header.

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

**Returns** header

**Return type** dict

anesplot.loadrec.loadmonitor_trendrecord.**loadmonitor_trenddata**(*filename*, *headerdico*)
load the monitor trend data

**Parameters**

- **filename** (*str*) – fullname

- **headerdico** (*dict*) – fileheader

**Returns** df = trends data

**Return type** pandas.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**(*dir_path=None*)
select a file using a dialog.

**Parameters** **dir_path** (*str*) – optional location of the data (paths[data])

**Returns** filename (full path)

**Return type** str

anesplot.loadrec.loadmonitor_waverecord.**loadmonitor_waveheader**(*filename=None*)
load the wave file header.

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

**Returns** header

**Return type** pandas.Dataframe

anesplot.loadrec.loadmonitor_waverecord.**loadmonitor_wavedata**(*filename=None*)
load the monitor wave csvDataFile.

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

**Returns** df = trends 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

---

anesplot.loadrec.loadtaph_trendrecord.**choosefile_gui**(*dir_path=None*)
select a file using a dialog.

**Parameters** **dir_path** (*str*) – optional location of the data (paths[data])

**Returns** filename (full path)

**Return type** str

anesplot.loadrec.loadtaph_trendrecord.**loadtaph_trenddata**(*filename*)
load the taphoniusData trends data.

**Parameters** **filename** (*str*) – fullname

**Returns** df = trends data

**Return type** pandas.Dataframe

anesplot.loadrec.loadtaph_trendrecord.**loadtaph_patientfile**(*headername*)
load the taphonius patient.csv file

**Parameters** **headername** (*str*) – fullname

**Returns** descr = patient_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**(*dir_path=None*)
select a file using a dialog.

**Parameters** **dir_path** (*str*) – optional location of the data (paths[data])

**Returns** filename (full path)

**Return type** str

anesplot.loadrec.loadtelevet.**loadtelevet**(*fname=None*, *all_traces=False*)
load the televetCsvExportedFile.

**Parameters**

- **file** (*str*) – name of the file
- **all_traces** (*bool*) – load all the derivations

**Returns** df = 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.**color_axis**(*ax*, *spine='bottom'*, *color='r'*)
change the color of the label & tick & spine.

**Parameters**

- **ax** (*matplotlib.pyplot.axis*) – the axis
- **spine** (*str*) – optional location in [bottom, left, top, right]
- **colors** (*str*) – optional color

anesplot.plot.trend_plot.**append_loc_to_fig**(*ax*, *dt_list*, *label='g'*)
append vertical lines to indicate a location eg: arterial blood gas

**Parameters**

- **ax** (*matplotlib.pyplot.axis*) – the axis
- **dt_list** (*[datetime]*) – list of datetime values
- **label** (*str*) – a key to add to the label (default is g)

**Returns res** a dictionary containing the locations

**Return type** dict

anesplot.plot.trend_plot.**save_graph**(*path*, *ext='png'*, *close=True*, *verbose=True*)
Save a figure from pyplot. :param path: The path (and filename, without the extension) to save the

figure to.

**Parameters**

- **ext** (*string (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** (*boolean (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** (*boolean (default=True)*) – Whether to print information about when and where the image has been saved.

anesplot.plot.trend_plot.**plot_header**(*descr*, *param=None*)
plot the header of the file.

**Parameters**

- **descr** (*dict*) – header of the recording
- **param** (*dict*) – dictionary of parameters

**Returns fig** plot of the header

**Return type** pyplot.figure

anesplot.plot.trend_plot.**hist_cardio**(*data*, *param=None*)
mean arterial pressure histogramme using matplotlib.

**Parameters**

- **data** (*pandas.DataFrame*) – the recorded trends data (keys used : ip1m and hr),
- **param** (*dict*) – parameters (save=bolean, path: path to directory)

**Returns fig** matplotlib.pyplot.figure

anesplot.plot.trend_plot.**plot_one_over_time**(*x*, *y*, *colour*)
plot y over x using colour

anesplot.plot.trend_plot.**hist_co2_iso**(*data*, *param=None*)
CO2 and iso histogramme (NB CO2 should have been converted from % to mmHg)

**Parameters**

- **data** (*pandas.Dataframe*) – the trends recorded data
- **param** (*dict*) – dictionary of parameters

**Returns** fig pyplot.figure

anesplot.plot.trend_plot.**cardiovasc**(*data*, *param=None*)
cardiovascular plot

**Parameters**

- **data** (*pandas.Dataframe*) – the recorded trends data keys 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= pyplot.figure

anesplot.plot.trend_plot.**cardiovasc_p1p2**(*data*, *param=None*)
cardiovascular plot with central venous pressure (p2)

**Parameters**

- **data** (*pandas.Dataframe*) – the trends recorded data keys used :[ip1s, ip1m, ip1d, hr, ip2s, ip2m, ip2d]
- **param** (*dict*) – dict(save: boolean, path[save], xmin, xmax, unit, dtime = boolean for time display in HH:MM format)

**Returns** fig= pyplot.figure

anesplot.plot.trend_plot.**co2iso**(*data*, *param=None*)
anesth plot (CO2/iso)

**Parameters**

- **data** (*pandas.Dataframe*) – the recorded data keys used :[ip1s, ip1m, ip1d, hr]
- **param** (*dictionary*) – dict(save: boolean, path[save], xmin, xmax, unit, dtime = boolean for time display in HH:MM format)

:returns fig= pyplot.figure

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

anesplot.plot.trend_plot.**co2o2**(*data*, *param*)
respiratory plot (CO2 and Iso)

**Parameters**

- **data** (*pandas.DataFrame*) – recorded trends data keys 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= pyplot.figure

anesplot.plot.trend_plot.**ventil**(*data*, *param*)
plot ventilation parameters (.tvInsp, .pPeak, .pPlat, .peep, .minVexp, .co2RR, .co2exp )

**Parameters**

- **data** (*pandas.DataFrame*) – recorded data, keys 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= pyplot.figure

anesplot.plot.trend_plot.**recrut**(*data*, *param*)
display a recrut manoeuver (.pPeak, .pPlat, .peep, .tvInsp)

**Parameters**

- **data** (*pandas.DataFrame*) – recorded data keys 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= pyplot.figure

anesplot.plot.trend_plot.**ventil_cardio**(*data*, *param*)
build ventilation and cardiovascular plot

**Parameters**

- **data** (*pandas.DataFrame*) – teh recorded trends data keys 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= pyplot.figure

anesplot.plot.trend_plot.**save_distri**(*data*, *path*)
save as O_.. the 4 distributions graphs for cardiovasc annd respi

anesplot.plot.trend_plot.**fig_memo**(*path*, *fig_name*)
append latex citation commands in a txt file inside the fig folder create the file iif it doesnt exist

### anesplot.plot.wave_plot module

Created on Tue Apr 19 09:08:56 2016

@author: cdesbois

anesplot.plot.wave_plot.**color_axis**(*ax*, *spine='bottom'*, *color='r'*)
change the color of the label & tick & spine.

**Parameters**

- **ax** (*matplotlib.pyplot.axis*) – the axis
- **spine** (*str*) – optional location in [bottom, left, top, right]
- **colors** (*str*) – optional color

anesplot.plot.wave_plot.**plot_wave**(*data*, *keys*, *param*)
plot the waves recorded (from as5)

**Parameters**

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

**Returns fig** plt.figure the plot

**Returns lines** plt.line2D the line to animate

(Nb plot data/index, but the xscale is indicated as sec)

anesplot.plot.wave_plot.**get_a_roi**(*waves*)
use the drawn figure to extract the relevant data in order to build an animation

**Parameters** **waves** (*MonitorWave object*) – a wave recording

**Returns** a dictionary containing ylims, xlims(point, dtime and sec),

traces used to build the plot, the fig object :rtype: dictionary

anesplot.plot.wave_plot.**create_video**(*waves*, *speed=1*, *save=False*, *savedir='~'*)
create a video from a figure

### 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['fs'])
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*, *fs=300*, *species='horse'*, *mult=1*)
detect the peak locations

**Parameters**

- **ser** (*pandas.series*) – the data
- **fs** (*integer*) – sampling frequency
- **species** (*string*) – in [horse]
- **mult** (*float*) – correction / 1 for qRs amplitude

**Returns** df=pandas.Dataframe

anesplot.treatrec.ekg_to_hr.**plot_beats**(*ecg*, *beats*)
plot ecg waveform + beat location

anesplot.treatrec.ekg_to_hr.**append_beat**(*beatdf*, *ekgdf*, *tochange_df*, *fig*, *lim=None*, *yscale=1*)
locate the beat in the figure, append to a dataframe[toAppend]

**Parameters**

- **beatdf** (*pandas.Dataframe*) – contains the point based location (pLocs)
- **ekgdf** (*pandas dataframe*) – contains the wave recording ((wekg_lowpass)
- **tochange_df** (*pandas.Dataframa*) – to store the beats toAppend or toRemove
- **fig** (*pyplot.Figure*) – figure to find time limits
- **lim** (*integer*) – ptBasedLim optional to give it manually
- **yscale** (*float*) – amplitude mutliplication factor for detection (default=1)

**Returns** tochange_df: incremented changedf (pt location)

**Return type** pandasDataframe

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, tochange_df, 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*, *ekgdf*, *tochange_df*, *fig*, *lim=None*)
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, tochange_df, 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*, *tochangedf*, *savename=''*, *dirpath=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*, *tochangedf*, *path_to_file=''*, *from_file=False*)
implement in the beat location the manual corrections fromFile = True force the disk loading of the dataframes

anesplot.treatrec.ekg_to_hr.**compute_rr**(*beatdf*, *fs=None*)
compute rr intervals (from pt to time)

**Parameters**

- **beatdf** (*pd.DataFrame*) – with pLoc

- **fs** (*integer*) – sampling frequency

**Returns** with: rr = rr duration rrDiff = rrVariation rrSqDiff = rrVariation^2

**Return type** pd.DataFrame

anesplot.treatrec.ekg_to_hr.**interpolate_rr**(*beatdf*, *kind=None*)
interpolate the beat_df (pt -> time values)

**Parameters**

- **beatDf** (*pd.Dataframe*) –
- **kind** (*str*) – linear or cubic(default)

**Returns** espts = evenly spaced points rrInterpol = interpolated rr

**Return type** pdDatatrame with evenly spaced data

anesplot.treatrec.ekg_to_hr.**plot_rr**(*ahr_df*, *param*, *HR=False*)
plot RR vs pt values + rrSqDiff

**Parameters**

- **pdDataFrame** (*hr_df =*) –
- **params** – dict containing fs as key

anesplot.treatrec.ekg_to_hr.**append_rr_and_ihr_to_wave**(*wave*, *ahrdf*)
append rr and ihr to the waves based on pt value (ie index)

anesplot.treatrec.ekg_to_hr.**plot_agreement**(*trenddf*)
plot ip1HR & ihr to check agreement

anesplot.treatrec.ekg_to_hr.**append_ihr_to_trend**(*trenddf*, *wavedf*, *ekgdf*)
append ihr (instataneous heart rate) to the trends

anesplot.treatrec.ekg_to_hr.**save_trends_data**(*trenddf*, *savename=''*, *dirpath='data'*)

save the trends data to a csv and hd5 file, including an ihr column

trenddf : pd.dataframes savename : str dirpath : str

path to save in (default= current working directory)

hdf file, key=trends_data

anesplot.treatrec.ekg_to_hr.**save_waves_data**(*wavedf*, *savename=''*, *dirpath='data'*)
save the trends data to a hd5 file, including an ihr column

**Parameters**

- **trenddf** (*pd.dataframes*) –
- **savename** (*str*) – dirpath : path to save in (default=data)
- **output** –
- **------** –
- **hdf_file** –
- **key='waves_data'** –

### 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** – 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*, *durmin=15*, *pamin=70*)
plot the hupotentions 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 throught 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.Serie** (*ser=*) –
- **win_lengh** (*integer*) – window lenght for averaging (in sec),
- **fs** (*int*) – sampling frequency

anesplot.treatrec.wave_func.**return_points**(*df*, *fig*)
return a tupple 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

**Returns**

**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 anesplot from a terminal** -> PYTHONPATH=<pathToAnesthPlot> python -m anesplot -> generate a quick plotting of most interestings parts

2. **from an ipython terminal** -> import anesthPlot.anesplot.recordmain as rec -> trends = rec.MonitorTrend() -> waves = rec.MonitorWave() -> 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 anesplot.record_main as rec sys.path.append(os.path.expanduser(~/pg/utils)) from utils 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

CHAPTER

FOUR

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## a

# INDEX

## S



## T



## U



## V



## W