

---

# **anesthPlot**

***Release beta***

**Christophe Desbois**

**Sep 09, 2021**



**MAIN**<sub>*SCRIPT*</sub> :



## 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 process
- the processes are mainly focused on horses anesthesia
- in our environment the data recorded came from an as3 or as5 anesthesia machine

### 1.1 Features

- **load** recordings from a trend or a wave recordings
- 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 (script usage))
- can be used as a **python package**
  - **usage :**

```
import anesthplot.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)
trends.show_graphs()
```



## 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

---

`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.check()`

print the loaded recordings

`anesplot.record_main.plot_trenddata(file, df, header, param_dico)`

clinical main plots of a trend recordings

**Parameters**

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

`anesplot.record_main.plot_monitorwave_data(headdf, wavedf)`

not implemented for the moment

**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**(tracesList=None)

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

**define\_a\_roi**()

define a ROI.

**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**()



## 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(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, header)`

load the monitor trend data

**Parameters**

- `filename` (*str*) – filename

- **header** (*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*)  
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*)  
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(file=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.plot_header(descr, param={'save': False})`  
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={})`  
mean arterial pressure histogramme using matplotlib.

#### Parameters

- **data** (`pandas.DataFrame`) – the recorded trends data (keys used : 'ip1m' and 'hr'),
- **param** (`dict`) – parameters (save=boolean, '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={})`

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={})`

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={})`

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={})`

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 doesn't 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** (*int*) – limits in point value (index)

- **maxi** (*int*) – 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)

## 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.build\_anim module

#### 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 anesplot.record_main as rec
from 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)
- **dataframe ekgdf** (*pandas*) – contains the wave recording ((wekg\_lowpass)
- **tochange\_df** (*pandas.DataFrame*) – 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="", savepath=None)`  
save the beats locations as csv and hd5 file

#### Parameters

- **beatde** (*pd.dataframes*) –
- **savepath** (*path to save in*) –

`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="", savepath=None)`  
save the trends data to a csv and hd5 file, including an ihr column

**Parameters**

- **trenddf** (*pd.dataframes*) –
- **savename** (*str*) –
- **savepath** (*str*) – path to save in (default= current working directory)

`anesplot.treatrec.ekg_to_hr.save_waves_data(wavedf, savename="", savepath=None)`  
save the trends data to a csv and hd5 file, including an ihr column

**Parameters**

- **trenddf** (*pd.dataframes*) –
- **savename** (*str*) – savepath : path to save in (default= current working directory)

## anesplot.treatrec.extract\_hypotension module

Spyder Editor

This is a temporary script file.

`anesplot.treatrec.extract_hypotension.extract_hypotension(trends, pamin=70)`  
return a dataframe with the beginning and ending phses of hypotension

### Parameters

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

**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(trends, durdf, durmin=15, pamin=70)`  
plot the hupotentions phases

### Parameters

- **trends** (*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(trends, 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 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

### Returns

**Return type** pandas.DataFrame

## Module contents

### 3.1.2 Submodules

### 3.1.3 Module contents



## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### a

- anesplot, ??
- anesplot.config, ??
- anesplot.config.build\_recordRc, ??
- anesplot.config.load\_recordRc, ??
- anesplot.loadrec, ??
- anesplot.loadrec.explore, ??
- anesplot.loadrec.loadmonitor\_trendrecord, ??
- anesplot.loadrec.loadmonitor\_waverecord, ??
- anesplot.loadrec.loadtaph\_trendrecord, ??
- anesplot.loadrec.loadtelevet, ??
- anesplot.plot, ??
- anesplot.plot.trend\_plot, ??
- anesplot.plot.wave\_plot, ??
- anesplot.record\_main, ??
- anesplot.treatrec, ??
- anesplot.treatrec.clean\_data, ??
- anesplot.treatrec.ekg\_to\_hr, ??
- anesplot.treatrec.extract\_hypotension, ??
- anesplot.treatrec.hr\_to\_hrv, ??
- anesplot.treatrec.wave\_func, ??