Welcome to USEPA_CTI's documentation!

Introduction

Python package for the evaluation of various NOx emissions binning algorithms for the US EPA Cleaner Trucks Initiative (CTI)

US EPA Clean Truck Initiative Time Based Window Code

Example Command Line Usage

Default options, using data from sample_data folder:

```
python cti_process_TBW.py --source_path sample_data --hdiut
```

Overlapping 300-second windows with power bins based on rate power CO2 normalization with a 25% window average power bin cutoff and a < 1 MPH true idle bin:

```
python cti_process_TBW.py --source_path sample_data --hdiut --window_step_secs 1 --window_length_s
```

Overlapping 180-second windows with power bins based on rate power CO2 normalization with 8% and 25% window average power bin cutoffs:

```
python cti_process_TBW.py --source_path sample_data --hdiut --window_step_secs 1 --window_length_s
```

For non-overlapping windows, set the window_step_secs equal to the window_length_secs

usage and command-line options

```
cti_process_TBW.py [-h] [--source_path SOURCE_PATH]
                   [--output_path OUTPUT_PATH] [--profile PROFILE]
                   [--verbose] [--include INCLUDE] [--exclude EXCLUDE]
                   [--window_length_secs WINDOW_LENGTH_SECS]
                   [--window_step_secs WINDOW_STEP_SECS]
                   [--window_min_secs WINDOW_MIN_SECS] [--hdiut]
                   [--idle_speed_thresh_mph IDLE_SPEED_THRESH_MPH]
                   [--ftp_co2_gphphr FTP_CO2_GPHPHR]
                   [--co2_normalization] [--true_idle_bin]
                   [--hp_cutpoints_pct HP_CUTPOINTS_PCT]
                   [--reuse_output_folder]
```

Time-Based Window Processor, generates window plots for cutpoint analysis

optional arguments:

`-h, --help`
: show this help message and exit

`--source_path` *SOURCE_PATH*

Path to folder containing files to process [default: .]]

`--output_path` *OUTPUT_PATH*

Path to folder for output results [default: .output]

`--profile` *PROFILE*

Path and filename to a cti_data_source_profile spreadsheet or “prompt” to launch file browser [default: cti_data_source_profile.xlsx]

`--verbose`

Enable verbose messages and file outputs

`--include` *INCLUDE*

File filter, files to include/accept [default: *.csv]

`--exclude` *EXCLUDE*

File filter, files to exclude/reject [default: *calcs.csv]

`--window_length_secs` *WINDOW_LENGTH_SECS*

time-based window length (seconds) [default: 300]

`--window_step_secs` *WINDOW_STEP_SECS*

time-based window step (seconds) [default: 300]

`--window_min_secs` *WINDOW_MIN_SECS*

time-based window minimum size (seconds) [default: 30]

`--hdiut`

Data comes from EPA Heavy-Duty In-Use Testing

`--idle_speed_thresh_mph` *IDLE_SPEED_THRESH_MPH*

Speed threshhold for idle bin below this speed [default: 1]

`--ftp_co2_gphphr` *FTP_CO2_GPHPHR*

FTP CO2 g/hp-hr for this engine

`--co2_normalization`

NOx g/hp-hr = NOx_g/CO2_g * CO2_g/FTP_hp-hr

`--true_idle_bin`

Add extra bin for true idle (vehicle speed < idle_speed_thresh_mph for entire window)

`--hp_cutpoints_pct` *HP_CUTPOINTS_PCT*

Horsepower cutpoints for bin definitions [default: 25]

`--reuse_output_folder`

Reuse output folder, do not delete prior results

This is development code written by EPA staff and is intended only for evaluation purposes-it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

usepa_cti package

Submodules

usepa_cti.cti_common module

cti_common.py

Support and shared routines for window data analysis

Note: This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

@author: US EPA

`usepa_cti.cti_common.`**`dataframe_to_numeric`**(*df*, *verbose=False*)

Convert all possible columns of a pandas dataframe to numeric values

Parameters:
- **df** – Pandas dataframe to convert
- **verbose** – If True then column names are printed to the console during processing

Returns: dataframe **df** with data converted to numeric values

`usepa_cti.cti_common.`**`handle_command_line_options`**(*app_description='Generic CTI App'*, *additional_args=[]*, *additional_options=[]*)

Handle command line options shared across CTI processor scripts

Parameters:
- **app_description** – 'Generic CTI App'
- **additional_args** – None
- **additional_options** – Command-line options

Returns: `runtime_options` object

`usepa_cti.cti_common.`**`prep_calcs_dataframe`**(*data_filename*, *data_source_profile*, *verbose=False*, *start_time=''*)

Pull in data file header, process time vector and process engine speeds and powers

Parameters:
- **data_filename** – Name of file to process
- **data_source_profile** – an object of class DataSourceProfile
- **verbose** – if True then optional outputs are printed to the console

- **start_time** – Optional start time for processing data based on time signal

Returns: (source_dataframe, calcs_dataframe) tuple

usepa_cti.cti_common.**prep_vehicle_speed**(*source_dataframe*, *calcs_dataframe*, *data_profile*)

Attempt to convert vehicle speed column from source dataframe to numeric values, then populate MPH and m/s speeds in the calcs dataframe

Parameters:
- **source_dataframe** – Pandas dataframe containing source data vehicle speed
- **calcs_dataframe** – Calculated values dataframe with vehicle speed in mph and m/s
- **data_profile** – an object of class DataSourceProfile

Returns: (source_dataframe, calcs_dataframe) tuple

class usepa_cti.cti_common.**runtime_options**

Bases: **object**

Container class for runtime options

usepa_cti.cti_common.**scale_signal**(*source_dataframe*, *source_signal*, *source_signal_scale*)

Scale signal (i.e. column) from source dataframe using source signal scale factor

Parameters:
- **source_dataframe** – Pandas dataframe containing signal to scale
- **source_signal** – Name (column heading) of signal to scale
- **source_signal_scale** – Numeric scale factor or 'degF->degC' or 'degC->degF'

Returns: scaled signal

usepa_cti.cti_data_source_profile module

cti_data_source_profile.py

Class to define and interpret a data source profile (engine specs, signal source, destination and scaling) spreadsheet

Note: This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

@author: US EPA

class usepa_cti.cti_data_source_profile.**DataSourceProfile**(*profile_filename*)

Bases: **object**

Class to define and interpret a data source profile (engine specs, signal source, destination and scaling) spreadsheet

get_power_rating(*filename*)

Parse engine power rating from HDIUT data filename

Parameters: **filename** – name of file to parse
Returns: self

`load_data_source_profile`(*profile_filename*)

Attempt to read a CTI data source profile spreadsheet and populate the `DataSourceProfile` properties

Parameters: **profile_filename** – path and filename of the data source profile to read

`read_parameter`(*index_str*, *allrows=False*)

Read parameter (row) from data source profile dataframe

Parameters:
- **index_str** – index (row name) into `self.dataframe`
- **allrows** – if True then all rows at the given index are read at once

Returns: single cell value or value of the desired row

`validate_predefined_input`(*input_str*, *valid_inputs*)

Check to see if a predefined parameter is one of the the predefined (allowed) choices

Warning: Exception raised on validation failure

Parameters:
- **input_str** – string to validate
- **valid_inputs** – python `dict` or `set` of acceptable string values

Returns: validated input string or raise exception if not valid

usepa_cti.cti_file_io module

cti_file_io.py

File system routines for general use, collects functionality from sys, os and shutil

Note: This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

@author: US EPA

`usepa_cti.cti_file_io.`**`delete_folder`**(*dstfolder*)

Delete the file system folder and all its contents

Parameters: **dstfolder** – pathname of folder to delete

`usepa_cti.cti_file_io.`**`file_exists`**(*filename*)

Verify the existence of filename

Parameters: **filename** – File pathname of the file to validate

:returns True if file is accessible, else False

usepa_cti.cti_file_io.**get_filename**(*filename*)

Returns file name without extension, e.g. /somepath/somefile.txt -> somefile

Parameters: **filename** – file name, including path to file as required
Returns: file name without extension

usepa_cti.cti_file_io.**get_filenameext**(*filename*)

Returns file name including extension, e.g. /somepath/somefile.txt -> somefile.txt

Parameters: **filename** – file name, including extension, including path to file as required
Returns: file name including extension

usepa_cti.cti_file_io.**get_filepath**(*filename*)

Returns path to file, e.g. /somepath/somefile.txt -> /somepath

Parameters: **filename** – file name, including path to file as required
Returns: file path, not including the file name

usepa_cti.cti_file_io.**get_filepathname**(*filename*)

Returns file name without extension, including path, e.g. /somepath/somefile.txt -> /somepath/somefile

Parameters: **filename** – file name, including path to file as required
Returns: file name without extension, including path

usepa_cti.cti_file_io.**get_parent_foldername**(*filepathnameext*)

Returns the parent folder of the given file e.g. /apath/somepath/somefile.txt -> somepath

Parameters: **filepathnameext** – file name, including extension and path to file
Returns: parent folder of the given file

usepa_cti.cti_file_io.**network_copyfile**(*remote_path*, *srcfile*)

Copy file to remote path

Parameters:
- **remote_path** – Path to file destination
- **srcfile** – source file name, including extension and path to file

usepa_cti.cti_file_io.**relocate_file**(*remote_path*, *local_filename*)

Move local file out to remote path and return the filename in that remote context

Parameters:
- **remote_path** – Path to file destination
- **local_filename** – local source file name, including extension and path to file as required

usepa_cti.cti_file_io.**sysprint**(*str*)

Performs ECHO command of str in CMD window

Parameters: **str** – string to echo

usepa_cti.cti_file_io.**validate_file**(*filename*)

Verify the existence of file, exit app on failure

Parameters: **filename** – File pathname of the file to validate

Warning: Exits app on failure

usepa_cti.cti_file_io.**validate_folder**(*dstfolder*)

Verify the existence of a folder and try to create it if doesn't exist

```
validate_folder('C:\Users\Temp')
```

Parameters: **dstfolder** – Path the folder to validate/create

Attention: Exits app on failure

usepa_cti.cti_plot module

cti_plot.py

Plotting functions based on matplotlib pyplot

Note: This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

@author: US EPA

usepa_cti.cti_plot.**fplothg**(*x*, *y*, **args*, ***kwargs*)

Create a new figure window and plot Y v. X, activate plot grid

Parameters:
- **x** – x data points
- **y** – y data points
- **args** – matplotlib pyplot arguments
- **kwargs** – matplotlib pyplot keyword arguments

Returns: (figure, axis) tuple

usepa_cti.cti_plot.**fplotyyhg**(*x*, *y*, *ylinespec*, *y2*, *y2linespec*)

Create a new figure window and plot Y v. X and Y2 v. X, with independent vertical axes, activate plot grid

Parameters:
- **x** – x data points
- **y** – first set of y data points
- **ylinespec** – matplotlib line spec for first set of y data
- **y2** – second set of y data points

- **y2linespec** – matplotlib line spec for second set of y data

Returns: (figure, axis1, axis2) tuple

usepa_cti.cti_plot.**label_xy**(*ax*, *x_label_str*, *y_label_str*)

Label x-axis, y-axis and set axis title

Parameters:
- **ax** – plot (axis) to label
- **x_label_str** – x axis label
- **y_label_str** – y axis label

usepa_cti.cti_plot.**label_xyt**(*ax*, *x_label_str*, *y_label_str*, *title_str*)

Label x-axis, y-axis and set axis title

Parameters:
- **ax** – plot (axis) to label
- **x_label_str** – x axis label
- **y_label_str** – y axis label
- **title_str** – axis title

usepa_cti.cti_plot.**lineat**(*ax*, *y*, **args*, ***kwargs*)

Plot a horizontal line

Parameters:
- **ax** – plot (axis) to draw on
- **y** – y value of line
- **args** – matplotlib pyplot arguments
- **kwargs** – matplotlib pyplot keyword arguments

usepa_cti.cti_plot.**vlineat**(*ax*, *x*, **args*, ***kwargs*)

Draw a vertical line at:

Parameters:
- **ax** – plot (axis) to draw on
- **x** – x value of line
- **args** – matplotlib pyplot arguments
- **kwargs** – matplotlib pyplot keyword arguments

usepa_cti.cti_process_TBW module

cti_process_TBW.py

Process time-based windows for NOx emissions

> **Note:** This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

@author: US EPA

usepa_cti.cti_process_TBW.**tbw_processor**(*data_filename*, *output_folder*, *__options*)

Process file for NOx emissions using time-based windows

Parameters:
- **data_filename** – Name of file to process
- **output_folder** – Name of output file folder
- **__options** – Data structure of command line / runtime options settings

Returns: Generates plots in ::output_folder

usepa_cti.cti_unit_conversions module

cti_unit_conversions.py

common engineering unit conversions

> **Note:** This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

```
# example usage:
import cti_unit_conversions as convert
engine_power_hp = engine_power_kW * convert.kw2hp
temp_degF = convert.degC2degF(temp_degC)
```

@author: US EPA

usepa_cti.cti_unit_conversions.**degC2degF**(*C*)

usepa_cti.cti_unit_conversions.**degF2degC**(*F*)

usepa_cti.cti_window_processor module

cti_window_processor.py

> **Note:** This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

@author: US EPA

usepa_cti.cti_window_processor.**find_windows**(*data*, *time_chan*, *window_chan*, *window_size*, *integrate_chans*, *data_chans=[]*, *scaling_dict={}*, *window_step=1*, *max_dt=1*, *verbose=False*)

Calculate windows of size (integrated quantity) window_size from the data dataframe using the window_chan column

Parameters:
- **data** – pandas dataframe of time-based emissions data
- **time_chan** – name (i.e. column heading) of time channel
- **window_chan** – name of channel to integrate (non-negative values only) to define window span
- **window_size** – desired window size (::window_chan integrated quantity)

- **integrate_chans** – other channel names to integrate over the window duration, string or list of strings
- **scaling_dict** – dictionary of multipliers for scaling signals (i.e. unit conversion)
- **window_step** – time interval between the start of consecutive windows, in seconds
- **max_dt** – maximum time step allowed (larger time steps are truncated to max_dt) - allows removal of time gaps
- **verbose** – if True then window contents are printed to the console

Returns: a pandas dataframe containing results by window

Module contents

__init__.py

Shared definitions for window processing code

> **Note:** This is development code written by EPA staff and is intended only for evaluation purposes—it does not represent how we may or may not use the resulting output in the development or promulgation of future rules

@author: US EPA

Indices and tables

- Index
- Module Index
- Search Page