# Module `process`

## Classes

`class DataProcessor`

### Instance variables

`var base_csv_file`

The current file to be studied -> This file corresponds to the day of the week we want to study.

`var selected_directory`

The currrent directory to be studied -> A directory corresponds to a site

### Methods

`def calculate_weekday_mean(self)`

The process that calculates the model file for a sepecific day of the week

`def data_for_day(self)`

Read the data of a specific date

#### Returns

`selected_date`
: The date selected from the list of available dates

`data_selected_date`
: the data for that specific date

`def data_model_from_file(self, weekday)`

read the data of a csv file that corresponds to the model file of the day of the week

#### Args

**weekday** : `string`
: the name in english of a day of the week

#### Returns

`data`
: dataframe that corresponds to the model of that data

`def data_normalization(self, date, data)`

Normalises the data so i can be used to calculate

#### Args

**date** : `string`
: he date of the data to be used

**data** : `pandas date`

The data of a site

### Returns

`results`
: Dataframe of the normalized data

`def day_modelfile_selection(self)`

Select the model file to be used

### Returns

`selected_csv`
: the name of the file that has bee selected to act has the model file

`def dayfile_selection(self)`

Selected the data from a specific day of the week [data from monday to sunday]

### Returns

`selected_csv`
: the name of the day of the week file that has bee selected

`def directory_selection(self)`

Select the directory that corresponds to the site that is to be used since each main directroy is seperated by the tagid

`def file_selection(self)`

Select the base file to be used to construct the model files

`def generate_model_weekday(self, weekday)`

Generate the model for a weekday

### Args

**`weekday`** : `string`
: the name in english of a day of the week

`def mean_weekday(self, csv_file=None)`

Generate the model file for a specific day of the week

### Args

**`csv_file`** : `filename`, optional
: File which contains data from the same day. Defaults to None.

### Returns

`_type_`
: *description*

`def proccessus(self)`

Process that seperates the base file from pubstack into the correct format

`def saving_preprocess(self)`

Process to create the skeleton of all the data files witch corresponds to weekdays

`def simple_verification(self, date, normalized_data)`

Using simple statistic indicators, calculates scores for each data entry

## Args

**`date`** : `panda date`
: the date of the specific data

**`normalized_data`** : `dataframe`
: data that has been normalized with data_normalization()

## Returns

`_type_`
: *description*

`def statistic_model_data(self)`

Calculate basic scores for each data entry of a model data and update the results with the z score

## Returns

`lower_bound`
: Using the interpercentile range, create a lower boundry that if passed mean that the data is abnormal

`def tag_division(self)`

# Index

## Classes