cartons_inventory Reference

Cartons

class `cartons_inventory.cartons.`**`CartonInfo`**(*carton*, *plan*, *category_label*, *stage='N/A'*, *active='N/A'*)

Bases: `object`

Saves targetdb info for cartons.

This class takes basic information from a carton (`name`, `plan`, and `category_label` at minimum) and at instantiation sets the carton dependent (as opposed to target dependent) information of the carton. `stage` and `active` parameters can also be provided but currently nothing is done with those. Carton dependent information is either taken from input parameters of __init__ or by the assign_carton_info function that also set the boolean in_targetdb to check the existence of the carton.

Then, function assign_target_info assigns target dependent information which can be the magnitude placeholders used for the different photometric system in the carton (calculate_mag_placeholders=True), and/or python sets with the unique values found per cadence, lambda, and instrument in the carton, along with `priority` and `value` ranges.

Finally, function process_cartons wraps all the functions of this class. Based on the value of the `origin` parameter, takes as input a file from rsconfig or custom, or takes a selection criteria to search cartons in targetdb. With this function we can evaluate the existence of a list of

cartons, check their content, save a selection criteria as an input file ready to be used by process_cartons, runs assign_target_info to get target parameter set, ranges, and/or magnitude_placeholders, saves an output .csv file with the information of each carton, or return a list of all the CartonInfo objects.

Parameters

- **carton** (*str*) – Carton name in table targetdb.carton
- **plan** (*str*) – Plan in table targetdb.version
- **category_label** (*str*) – Label in targetdb.category table (e.g. science, standard_boss, guide)
- **stage** (*str*) – Robostrategy stage, could be srd, open, none, filler. Default is 'N/A'
- **active** (*str*) – `y` or `n` to check if it is active in robostrategy. Default is 'N/A'
- **mapper_label** (*str*) – Label in targetdb.mapper (MWM or BHM)
- **program** (*str*) – Program in targetdb.program table

- **version_pk** (*int*) – ID in targetdb.version table
- **tag** (*str*) – tag in targetdb.version table
- **mapper_pk** (*int*) – Mapper_pk in targetdb.carton table. 0 for MWM and 1 for BHM
- **category_pk** (*int*) – category_pk in targetdb.carton table (e.g. 0 for science)
- **in_targetdb** (*bool*) – True is carton/plan/category_label combination is found in targetdb, false if not.
- **sets_calculated** (*bool*) – True when in_targetdb is True and target dependent parameters value_min, value_max, priority_min, priority_max, cadence_pk, cadence_label, lambda_eff, instrument_pk, and instrument_label have been calculated for the carton using assign_target_info(calculate_sets=True)
- **mag_placeholders_calculated** (*bool*) – True when magnitude placeholdes used for SDSS, TMASS, and GAIA photometric systems have been calculated. These are calculated using check_magnitude_outliers function

`assign_carton_info`()

Assigns carton dependent information for cartons in targetdb.

If the carton/plan/category_label combination in the CartonInfo object is found in targetdb this function assigns attributes for carton dependent parameters (parameters shared for all targets in the carton). These paraemters are mapper_label, program, version_pk, tag, mapper_pk, and category_pk. Finally it set in_targetdb attribute as True when found in the database.

`assign_target_info`(*calculate_sets=True*, *calculate_mag_placeholders=False*)

Assignt target dependent information for cartons in targetdb.

This function calls return_target_dataframe to get a Pandas DataFrame with target dependent information for a carton. Then it sets different attributes to the CartonInfo object depending on the values of calculate_sets and calculate_mag_placeholders

Parameters

- **calculate_sets** (*bool*) – If true this function assigns the attributes value_min, value_max, priority_min, priority_max, cadence_pk, cadence_label, lambda_eff, instrument_pk, and instrument_label, based on information from targetdb. It also sets the attribute sets_calculated as True to keep record.

- **calculate_mag_placeholders** (*bool*) – If true this function assigns the attribute magnitude_placeholders using check_mag_outliers function, and sets mag_placeholers_calculated=True to keep record. magnitude_placeholres is a set with all the combination of photometric system (SDSS, TMASS, GAIA) and mag placeholder used for that photometric system in that carton (None, Invalid, 0.0, -9999.0, 999, 99.9).

`build_query_target`()

Creates the query with the target dependet information of the carton.

`check_existence`(*log*, *verbose=True*)

Checks if the carton/plan/category_label from object is found in targetdb.

This function checks whether a carton exists or not in targetdb, to be used when a list of cartons is used in process_cartons (i.e. `origin` rsconfig or custom) or to check the existence of a single carton.

Parameters

- **log** (*SDSSLogger*) – Log used to store information of cartons_inventory

- **verbose** (*bool*) – If true and if the carton is not found in the database the function will print and save on log information to try to correct the input file from which the carton/plan/category_label was taken (and stored in the object). If no carton with that name is found in targetdb

it will print the associated warning, and if cartons with the same name but different plan or category_label are found a line with input file format will be printed for each of those cartons so the user can replace the line in the input file with one of the options proposed.

Returns

cartons_aleternatives (*Pandas DataFrame*) – A Pandas DataFrame that for each carton/plan/category_label combination not found in targetdb has an entry for it and for all the alternative cartons found in targetdb that have the same carton name but different plan or category. For each entry the dataframe contains the columns carton, plan, category_label, stage, active, tag, version_pk, and in_targetdb.

`print_param`(*par*, *width*, *log*)

logs a message with width=width containing a parameter from carton object.

`print_range`(*par*, *width*, *log*)

logs a message with width=width containing the range of a parameter from the carton.

`return_target_dataframe`()

Executes query from build_query_target and returns it in a Pandas DataFrame.

`visualize_content`(*log*, *width=140*)

Logs and prints information from targetdb for a given carton.

cfg = *{'bands': {'GAIA': ['bp', 'rp', 'gaia_g'], 'SDSS': ['g', 'r', 'i', 'z'], 'TMASS': ['j', 'h', 'k']}, 'db_fields': {'carton_dependent': ['program', 'version_pk', 'tag', 'mapper_pk', 'mapper_label', 'category_pk'], 'input_dependent': ['plan', 'category_label', 'stage', 'active'], 'set_ranges': ['value', 'priority'], 'sets': ['value', 'priority', 'cadence_pk', 'cadence_label', 'lambda_eff', 'instrument_pk', 'instrument_label']}}*

cartons_inventory.cartons.check_mag_outliers(*datafr*, *bands*, *systems*)

Returns a list with all the types of outliers found for each photometric system.

Parameters

- **datafr** (*Pandas DataFrame*) – Containing the magnitudes from different photometric systems for the stars in a given carton.

- **bands** (*strings list*) – Containing the bands to search each belonging to a given photometric system.

- **system** (*strings list*) – Photometric system to which each band listed belongs to. The options are ‘SDSS’, ‘TMASS’, and ‘GAIA’. The system to which a band belongs is defined by the index of the band in the list (i.e. band[ind] belongs to systems[ind])

Returns

placeholders (*set*) – A set of strings where each string starts with the photometric system, then an underscore and finally the type of magnitude outlier that at least one magnitude of the corresponding system has. The type of outliers are: None (For empty entries), Invalid (For Nan's and infinite values), and <<Number>> (For values brighter than -9, dimmer than 50, or equal to zero), in the latter cases the number itself is returned as the outlier type.
For example if a carton contains stars with h=999.9, k=999.9, j=None, and bp=Inf. This function will return {'TMASS_999.9', 'TMASS_None', 'GAIA_Invalid}.

`cartons_inventory.cartons.`**`gets_carton_info`**(***carton_list_filename, header_length=1, delimiter='\|'***)
Get the necessary information from the input carton list file.

`cartons_inventory.cartons.`**`print_centered_msg`**(***st, width, log***)
Logs and prints string st with width=width in the log

`cartons_inventory.cartons.`**`process_cartons`**(***origin='rsconfig', files_folder='./files/', inputname=None, delim='\|', check_exists=True, verb=True, return_objects=False, write_input=False, write_output=True, assign_sets=True, assign_placeholders=False, visualize=False, overwrite=False, all_cartons=True, cartons_name_pattern=None, versions='latest', forced_versions=None, unique_version=None***)
Get targetdb information for list of cartons or selection criteria and outputs .csv file.

Takes as input a file with a list of cartons from rsconfig (origin=``rsconfig``) or custom (origin=``custom``) or a selection criteria to be applied on targetdb (origin=``targetdb) in which case an input list file can also be created (with write_input=True) for future use.

This function can be used to check the existence of the cartons (check_exist=True) in which case it returns a dataframe with the alternative cartons information, or it can be used to call assign_target_info to get the targetdb information of all the cartons (check_exists=False) and store it in a .csv file and/or return the CartonInfo objects.

The function also has provides the option of logging and printing the targetdb information from all the cartons in a human readable way by using visualize=True.

Parameters

- **origin** (*str*) – `rsconfig` to use input list file from rsconfig, `custom` to use custom input list of carton or `targetdb` to look for cartons in targetdb based on
  the `all_cartons`, `cartons_name_pattern`, `versions`, `forced_versions`, and `unique_versions` parameters.

- **files_folder** (*str*) – Main folder where input and output files would be stored. In this folder subfolders rsconfig, custom, and targetdb are expected.

- **inputname** (*str or None*) – Name of input file to be searched in <<files_folder>>/<<origin>> folder

- **delim** (*str*) – Delimiter character to use when creating output .csv file

- **check_exists** (*bool*) – If true and origin is rsconfig or custom the function looks for alternatives to cartons that exist in targetdb but have different values of plan or category_label than carton object. In this case the function returns a dataframe with the original carton versions not found and the alternatives and exits the function

- **verb** (*bool*) – If True function logs and prints alternatives to replace the input lines corresponding to carton/plan/category_label combinations not found in targetdb with lines corresponding to the same carton but with existing plan/category_label combinations.

- **return_objects** (*bool*) – If True the function returns the CartonInfo objects.

- **write_input** (*bool*) – If True the function writes a file to be used then as input by process_cartons with the cartons retrieved by the targetdb query.

- **write_output** (*bool*) – If True the function creates an output .csv file with the information of each CartonInfo object.

- **assign_sets** (*bool*) – If True assign_target_info assigns the attributes for target dependent parameters for the carton. For each parameter returns a python set with all the values present in the carton targets or the range spanned by them.

- **assign_placeholders** (*bool*) – If True assign_target_info assigns magnitude placeholders found in targetdb for each photometric system (SDSS, TMASS, GAIA) for each carton using check_mag_outliers.

- **visualize** (*bool*) – If True we log and print all the information found in targetdb for each carton in a human readable way

- **overwrite** (*bool*) – If True enables that inputfile like and output file could be overwritten.

- **all_cartons** (*bool*) – If True and origin=targetdb cartons with any name are taken from targetdb. from targetdb

- **cartons_name_pattern** (*str or None*) – If True and origin=targetdb only cartons with pattern name cartons_name_pattern are are taken

from targetdb. The string uses * character as wildcard

- **versions** (*str*) – If True and origin=targetdb sets the versions that would be taken for each carton name If `single` only versions matching `unique_version` will be taken, if `latest` only the latest version of each carton would be taken, if `all` all versions from each carton is taken.

- **forced_versions** (*dict or None*) – If present, and origin=targetdb all cartons in this dictionary are forced to consider only the version in the dictionary corresponding value, independent on the `versions` value.

- **unique_version** (*Int or None*) – If present, origin=targetdb, and versions=single then only this version_pk will be considered for each carton

Returns

- If check_exists=True returns **cartons_aleternatives** (*Pandas DataFrame*) – A Pandas DataFrame that for each carton/plan/category_label combination not found in targetdb has an entry for it and for all the alternative cartons found in targetdb that have the same carton name but different plan or category. For each entry the dataframe contains the columns carton, plan, category_label, stage, active, tag, version_pk, and in_targetdb.

- If check_exists=False and return_objects=True returns the **cartons** (CartonInfo objects) - Objects with the carton dependent attributes assigned at instantiation plus the target dependent attributes that were set by assign_target_info.