bw2preagg

Release 0.1

Apr 22, 2020

Contents:

bw2preagg is used to generate dependently sampled LCI and LCIA arrays for whole databases.

It uses Brightway2 framework and presamples. The arrays are stored as `numpy.ndarray` files, and can be integrated in Brightway2 models using other modules (brightway2-aggregated, presamples).

The resulting result arrays allow the use of aggregated LCI or LCIA results (also known as cradle-to-gate results) in LCA while also correctly integrating the uncertainty of LCI data, see Lesage et al. 2018.

Generating samples of cradle-to-gate results for all activities in a database can be a lengthy process, especially when dealing with large LCI databases like ecoinvent. The functions are therefore geared towards large, bulk calculations, by e.g. facilitating multiprocessing and breaking down the task in multiple "batches" that can run on different systems.

The samples are calculated in six steps:

Step 1: Setup Ready your system for sample generation: setup a brightway2 *project*, import an ecoinvent database from ecoSpold files, and generate a number of files, stored in *common files directory*, that will be used by future steps.

Step 2: Generating base presamples Generate a *presamples package* for all **A** and **B** matrix elements. The presample package will have as many rows as there are matrix elements, and as many columns as the number of requested iterations. These will be injected in **A** and **B** matrices during LCI calculations in Step 4.

Step 3: Generating balancing presamples (Optional) Generate a **balanced** *presamples package* for water and another for land transformation exchanges. These are also injected in **A** and **B** matrices during LCI calculations in Step 4, overriding the base presamples from Step 2.

Step 4: Generating LCI samples Generate LCI arrays for all activities in the database. These use the presamples packages from previous steps.

Step 5: Generating LCIA samples (Optional) Transform the LCI arrays from Step 4 to arrays of LCIA scores.

Step 6: Concatenating samples (Optional) Concatenate LCI or LCIA arrays from multiple *samples_batch* folders.

If you just want to run and calculate the samples, you can jump to the *Run-through: LCI and LCIA array production* section.

The most important concepts, input parameters and the structure of the output directory are summarized in the *Glossary*.

CHAPTER 1

Table of contents

1.1 Installing

Note: It is good practice to install packages like this in dedicated virtual environments. An example is given in the *run through*.

The package is hosted on Pypi and can be installed using pip:

```
pip install bw2preagg
```

This will also install all dependencies, including Brightway2 framework and presamples.

1.2 Step 1: Setup

1.2.1 Introduction

Setting things up to use `bw2preagg` is done through a single function, `setup_project`, which is imported in the namespace with `from bw2preagg import *`. This function does multiple things to ready your system for sample generation:

- sets up a brightway2 *project*. To use an existing project, simply pass `overwrite_project=False`
- imports an ecoinvent database from ecoSpold files. To use an existing imported LCI database, simply pass `overwrite_database=False`. If the database does not exist in the project, or if `overwrite_database=True`, you must pass a valid path `database_dir` pointing to a directory with ecoSpold2 files.
- generates a number of files that are used later and store these in the *common_files* subdirectory of the *result_dir directory*. To skip generation of these files if they exist already, simply pass `force_write_common_files=False`.

- generates deterministic LCI results for all activities, stored in the *deterministic* subdirectory of the *result_dir directory*.

1.2.2 Technical reference

bw2preagg.setup_project.**setup_project**(*project_name*, *database_name*, *result_dir*, *database_dir=None*, *overwrite_project=False*, *overwrite_database=False*, *save_det_lci=True*, *force_write_common_files=False*, *default_bw2setup=True*)

Create project, import databases and generate common files as required

Parameters

- **project_name** (*str*) – Name of the brightway2 project in which to import data. If project does not exist, it will be created.
- **database_name** (*str*) – Name of the existing LCI database or to give to the LCI database being imported.
- **result_dir** (*str*) – Path to the directory where data used or generated by bw2preagg is saved.
- **database_dir** (*str, default=None*) – Path to directory with ecoSpold2 data to be imported, None if LCI database is not to be imported (i.e. if it exists already).
- **overwrite_project** (*bool, default=False*) – If True, then the existing project with the name project_name is deleted first, and all data is reimported for a clean slate import.
- **overwrite_database** (*bool, default=False*) – If True, then the existing LCI database with name database_name in the brightway2 project is deleted first, and LCI data is reimported
- **force_write_common_files** (*bool, default=True*) – If True, then the common files are generated even if they already exist at given location
- **save_det_lci** (*bool, default=True*) – If True, deterministic LCI arrays are saved in the deterministic subfolder of the result_dir
- **default_bw2setup** (*bool, default=True*) – If True, run bw2setup to include default elementary flows and LCIA methods

Returns

Return type None

1.3 Step 2: Generating base presamples

1.3.1 Introduction

This step generates a *presamples package* for all technosphere (**A**) and biosphere (**B**) elements for an LCI database. The only function that the user needs for this step is `generate_base_presamples`, which is imported in the namespace with `from bw2preagg import *`.

The resulting presample package is stored in the *presamples* subdirectory of the *result_dir directory*. Information about the presamples packages are stored in a *presamples resource*, which are in turn included in a *campaign*.

The *samples_batch* argument (*int*) allows generating presamples in batches: each batch has its own presample package, its own seed, and is associated with its own *presamples resource* and *campaign*. Since the seed is determined from the samples_batch value, presample packages produced on different computers are guaranteed to have the same values, increasing reproducibility of preaggregated LCI arrays and of LCA results that are based on these arrays.

1.3.2 Technical reference

bw2preagg.base_presamples.**generate_base_presamples**(*project_name*, *database_name*, *result_dir*, *iterations*, *samples_batch*, *overwrite_ps=True*, *ps_base_name='base'*)

Generate presamples for all elements of A and B matrices of given database

The presamples are stored in a presamples resource in result_dir and added to a presamples campaign for easy reuse. It is recommended that presamples are generated in batches (e.g. of 1000 iterations each). This is implemented via the samples_batch argument.

Parameters

- **project_name** (*str*) – Name of the brightway2 project where the database is imported
- **database_name** (*str*) – Name of the LCI database
- **result_dir** (*str*) – Path to directory where results are stored
- **iterations** (*int*) – Number of iterations to include in sample
- **samples_batch** (*int, default=0*) – Integer id for sample batch. Used for campaigns names and for generating a seed for the RNG. The maximum value is 14.
- **overwrite_ps** (*bool, default=True*) – Overwrite presamples package if it exists
- **ps_name_base** (*str, default="base"*) – Base name for presamples resource.

Returns

Return type None

1.4 Step 3: Generating balancing presamples

1.4.1 Introduction

Randomly sampling exchange values will usually break the intended balance across balances in a given activity (e.g. the ratio of water in to water out for a given activity will be very different across Monte Carlo iterations). This optional step generates balanced water and land transformation exchanges for a given LCI database, and stores each in a dedicated *presamples package*. The only function that the user needs for this step is `generate_balancing_presamples`, which is imported in the namespace with `from bw2preagg import *`.

Warning: The calculation of these balanced exchanges takes a long time (on the order of a few hours). However, it is only done once - this is the advantage of `bw2preagg`.

The presamples packages are also stored in the *presamples* subdirectory of the *result_dir directory*, and a corresponding *presamples resource* is added to a *campaign* with the same *samples_batch* id.

Balancing presamples are typically used to replace values in *base presamples* with the corresponding *samples_batch* id during LCI calculations.

For more information on the balancing itself, consult the dedicated water exchange and land transformation exchange balancing packages.

1.4.2 Technical reference

bw2preagg.balancing_presamples.**generate_balancing_presamples**(*project_name, database_name, result_dir, iterations, samples_batch, overwrite_ps=True, balance_water=True, balance_land=True, ecoinvent_version='3.6', land_from_patterns=['Transformation, from'], land_to_patterns=['Transformation, to'], expect_base_presamples=True*)

Generate balancing presamples for a given db

These can be sampled for water exchanges and/or land transformation exchanges. The presamples are stored in a presamples resource in result_dir and added to a presamples campaign for easy reuse. It is recommended that presamples are generated in batches (e.g. of 1000 iterations each). This is implemented via the samples_batch argument.

Parameters

- **project_name** (*str*) – Name of the brightway2 project where the database is imported
- **database_name** (*str*) – Name of the LCI database
- **result_dir** (*str*) – Path to directory where results are stored
- **iterations** (*int*) – Number of iterations to include in sample
- **samples_batch** (*int, default=0*) – Integer id for sample batch. Used for campaigns names and for generating a seed for the RNG. The maximum value is 14.
- **overwrite_ps** (*bool, default=True*) – Overwrite presamples package if it exists
- **balance_water** (*bool, default=True*) – Balance water exchanges
- **balance_land** (*bool, default=True*) – Balance land transformation exchanges
- **ecoinvent_version** (*str, default="3.6"*) – Release number of ecoinvent database
- **land_from_patterns** (*list of string, default=['Transformation, from']*) – Patterns used to identify land transformation inputs
- **land_to_patterns** (*list of string, default=['Transformation, to']*) – Patterns used to identify land transformation outputs

- **expect_base_presamples** (*bool, default=True*) – If True, ValueError is raised if base presamples package for corresponding samples_batch is missing

Returns

Return type None

1.5 Step 4: Generating LCI samples

1.5.1 Introduction

This step is where the actual LCI arrays are calculated. It works by successively instantiating as many brightway2 `MonteCarloLCA` as there are activities in the LCI database, and calculating the resulting LCI results for the same number of iterations as those contained in the corresponding presamples package.

What ensures that the LCI are *dependently* sampled is the use of presample packages, which inject the same values for all technosphere (**A**) and biosphere (**B**) elements.

The resulting LCI arrays are stored in the *LCI* subdirectory of the *result_dir directory*. Each *samples_batch* has its own *subdirectory*

Warning: Because of the large number of LCI datasets in LCI databases, this step is extremely long (on the order of days). `bw2preagg` implements three strategies to help reduce this time:

1- *samples_batch*: Allows calculating multiple sets of dependently sampled arrays with smaller number of iterations (one batch). These "batches" can then be concatenated into arrays with the required number of iterations.

2- *parallel_jobs*: Allows the parallel calculation of LCI arrays on multiple CPU of a given computer (default=1).

3- *slices*: Useful on computer clusters, allows further splitting up of the activity list into smaller slices and sending these to different jobs.

Several functions can be of interest, all pf which are imported in the namespace with `from bw2preagg import *`. The `calculate_lci_array` calculates the actual LCI arrays. It is rarely directly invoked by a user, but rather called from `set_up_lci_calculations`, that gathers the necessary information to run `calculate_lci_array` for a specified set of activities. `set_up_lci_calculations` itself is rarely invoked direclty by a user, but rather from a top-level `dispatch_lci_calculators` function that splits the LCI calculation across machines (*slices*) and across CPUs (*parallel_jobs*).

1.5.2 Technical reference

dispatch_lci_calculators

The top-level function is `dispatch_lci_calculators`. It is typically the only one a user will interact with.

`dispatch_lci_calculators` verifies that all required data (project, presamples, common files, etc.) are actually available, splits the work first across slices (to run on multiple computers in a cluster) and then across CPUs (to use MultiProcessing) and invokes `set_up_lci_calculations`.

bw2preagg.lci.**dispatch_lci_calculators**(*project_name*, *database_name*, *result_dir*, *samples_batch=0*, *parallel_jobs=1*, *slice_id=None*, *number_of_slices=None*)

Dispatches LCI array calculations to distinct processes (multiprocessing)

If number_of_slices/slice_id are not None, then only a subset of database activities are processed.

The number of iterations is determined by the number of columns in the presample packages referenced in the corresponding campaign.

If slice_id and number_of_slices are not None, will only treat a subset of activities.

Parameters

- **project_name** (*str*) – Name of the brightway2 project where the database is imported
- **database_name** (*str*) – Name of the LCI database
- **result_dir** (*str*) – Path to directory where results are stored
- **samples_batch** (*int, default=0*) – Integer id for sample batch. Used for campaigns names and for generating a seed for the RNG. The maximum value is 14.
- **parallel_jobs** (*int, default=1*) – Number of parallel jobs to run using multiprocessing
- **slice_id** (*int, default=None*) – ID of slice. Useful when calculations are split across many computers or jobs on a computer cluster. If None, LCI arrays are generated for all activities in the database.
- **number_of_slices** (*int, default=None*) – Number of slices over which the calculations are split. Useful when calculations are split across many computers or jobs on a computer cluster. If None, LCI arrays are generated for all activities in the database.

set_up_lci_calculations

The set_up_lci_calculations function then gathers the necessary information for the specified subset of activities and invokes the subsequent calculate_lci_array for each activity for which LCI arrays need to be calculated.

bw2preagg.lci.**set_up_lci_calculations**(*activity_list*, *result_dir*, *worker_id*, *database_name*, *samples_batch*, *project_name*)

Dispatch LCI calculation for a list of activities

Parameters

- **activity_list** (*list*) – List of codes to activities for which LCI arrays should be calculated
- **result_dir** (*str*) – Path to directory where results are stored
- **worker_id** (*int*) – Identification of the worker if using MultiProcessing, used only for error messages.
- **database_name** (*str*) – Name of the LCI database
- **samples_batch** (*int*) – Integer id for sample batch. Used for campaigns names and for generating a seed for the RNG. The maximum value is 14.
- **project_name** (*str*) – Name of the brightway2 project where the database is imported

Returns

Return type None

calculate_lci_array

`calculate_lci_array` is the function where the actual LCI calculation occurs, using the brightway2 `MonteCarloLCA` class.

`bw2preagg.lci.`**`calculate_lci_array`**(*database_name*, *act_code*, *presamples_paths*, *g_dimensions*, *total_iterations*, *g_samples_dir*)

Return LCI array using specified presamples for given activity

Typically invoked from generate_LCI_samples.set_up_lci_calculations

Parameters

- **database_name** (*str*) – Name of the LCI database
- **act_code** (*str*) – code of the activity
- **presamples_paths** (*list*) – list of paths to presamples packages
- **g_dimensions** (*int*) – Number of rows in biosphere matrix
- **total_iterations** (*int*) – Number of iterations (i.e. number of columns in presample arrays)
- **g_samples_dir** (*str*) – Path to directory where LCI arrays will be saved

Returns

Return type None

1.6 Step 5: Generating LCIA samples

1.6.1 Introduction

This optional step converts the stored LCI arrays into method-specific arrays of LCIA scores.

LCIA arrays can be distinguished by:

- number of columns:
  - **one column**: a deterministic results (`result_type=deterministic`)
  - **n columns**: one column per MonteCarlo iteration (use function `result_type=probabilistic`)
- number of rows:
  - **one row**: representing the sum of characterized elementary flows (`return_total=True`)
  - **m rows**: *m* characterized elementary flows (`return_per_exchange=True`).

While functions such as `calculate_lcia_array_from_activity_code` can calculate LCIA arrays for specific activities, use `save_all_lcia_score_arrays` to generate and save LCIA arrays for all activities.

The resulting LCIA arrays are stored in a method-specific subdirectory of the *result_dir directory*. The name of the subdirectory is a string *abbreviation* of the method name (see here for more detail on method abbreviations). *Totals* and *per reference flow* are further separated in subdirectories, and finally each *samples_batch* has its own subdirectory.

Note: It is absolutely necessary to have access to the biosphere dictionary of the LCA object that was used to generate the samples, the *ref_bio_dict*. It is stored in the *common_files* directory.

1.6.2 Technical reference

Top level functions

save_all_lcia_score_arrays

bw2preagg.lcia.**save_all_lcia_score_arrays**(*result_dir*, *method*, *result_type='probabilistic'*, *samples_batch=0*, *dtype=<class 'numpy.float32'>*, *return_total=True*, *return_per_exchange=True*, *ignore_missing=True*, *project_name=None*)

Calculate and save LCIA score arrays for given set of LCI arrays and method

Parameters

- **result_dir** (*str*) – Path to directory where results are stored
- **method** (*tuple*) – LCIA method identification in brightway2 (tuple)
- **result_type** (*str, default='probabilistic'*) – Specify whether probabilistic or deterministic results should be generated
- **samples_batch** (*int, default=0*) – Integer id for sample batch. Used for campaigns names and for generating a seed for the RNG. The maximum value is 14.
- **dtype** (*str or dtype*) – dtype to which the resulting LCIA array must be converted
- **return_total** (*bool, default=True*) – If True, LCIA array with total scores across elementary flows are saved
- **return_per_exchange** (*bool, default=None*) – If True, LCIA array with scores per elementary flows are saved
- **ignore_missing** (*bool, default=True*) – If True, will calculate LCIA score arrays for all files in the corresponding LCI folder. If False, the calculation of LCIA score arrays will only proceed if all expected LCI arrays are found in the LCI array.
- **project_name** (*str*) – Name of the brightway2 project where the database is imported

Returns

Return type None

calculate_lcia_array_from_activity_code

bw2preagg.lcia.**calculate_lcia_array_from_activity_code**(*result_dir*, *act_code*, *method*, *samples_batch=0*, *dtype=<class 'numpy.float32'>*, *return_total=True*, *project_name=False*)

Retrieves LCI array and calculates LCIA array for a given method

Parameters

- **result_dir** (*str*) – Path to directory where results are stored
- **act_code** (*str*) – Code of the activity
- **method** (*tuple*) – LCIA method identification in brightway2 (tuple)

- **samples_batch** (*int, default=0*) – Integer id for sample batch. Used for campaigns names and for generating a seed for the RNG. The maximum value is 14.
- **dtype** (*dtype or string, default=np.float32*) – dtype the result should be converted to.
- **return_total** (*bool, default=True*) – If True, the function returns an array of total scores If False, the function returns an array of characterized elementary flows
- **project_name** (*str*) – Name of the brightway2 project where the database is imported

Helper functions

get_cf_with_indices

bw2preagg.lcia.**get_cf_with_indices**(*method*, *ref_bio_dict*, *project_name*)

Extract row indices of biosphere matrix and associated cfs for given method

A reference biosphere dictionary, associated with a given LCA object, must be supplied to ensure that the biosphere row indices are correct.

Parameters

- **method** (*tuple*) – Identification of the LCIA method, using Brightway2 tuple identifiers
- **ref_bio_dict** (*dict*) – Dictionary mapping elementary flow keys to matrix rows
- **project_name** (*str*) – Name of the brightway2 project where the database is imported

Returns

- **B_row_indices** (*list of int*) – List of biosphere matrix row indices
- **cfs** (*list of float*) – List of associated characterization factors

calculate_lcia_array_from_arrays

bw2preagg.lcia.**calculate_lcia_array_from_arrays**(*LCI_array*, *B_row_indices*, *cfs*, *dtype=<class 'numpy.float32'>*, *return_total=True*)

Calculation of LCIA from array LCI array, B row indices and cfs

Used by other functions to carry out actual calculations.

Parameters

- **LCI_array** (*numpy array*) – Life cycle inventory array, with rows equal to elementary flows and columns equal to separate LCI, e.g. from different MonteCarlo iterations
- **B_row_indices** (*list of int*) – List of biosphere matrix row indices
- **cfs** (*list of float*) – List of associated characterization factors
- **dtype** (*str or dtype*) – dtype to which the resulting LCIA array must be converted
- **return_total** (*bool, default=True*) – If True, sum LCIA array rows, returning total scores If False, returns an LCIA array with the same dimension as the LCI array

Returns **lcia_array** – Array of LCIA scores

Return type numpy.ndarray

1.7 Step 6: Concatenating samples

1.7.1 Introduction

This optional step allows the concatenation of samples across multiple *samples_batch* folders.

It can be used for LCI or LCIA arrays, and results can be saved in a new directory.

1.7.2 Technical reference

concat_samples_arrays_in_result_type_dir

To concatenate LCI arrays or LCIA arrays for which the name of the folder is known, use `concat_samples_arrays_in_result_type_dir`:

bw2preagg.concat_batches.**concat_samples_arrays_in_result_type_dir**(*result_dir*, *sb_id_list=None*, *result_type_dirname='LCI'*, *totals_or_per_exchanges=None*, *sim_name=None*, *dest=None*, *fail_if_samples_batches_different=F*, *ignore_missing=True*)

Concatenate and save LCI sample arrays in result dir

Parameters

- **result_type** (*str*) – Path to the directory where data used or generated by bw2preagg is saved.
- **sb_id_list** (*list, default=None*) – List of samples_batch ids. If None, all samples_batch are processed.
- **result_type_dirname** (*str, default="LCI"*) – Name of the result type subdirectory where samples batches are found
- **sim_name** (*str, default=None*) – Name to give the directory in which results will be saved
- **dest** (*str*) – Path to location where concatenated array directory will be saved
- **fail_if_samples_batches_different** (*bool, default=False*) – If False, will raise ValueError if arrays available in samples_batch folders are not the same If True, will concatenate only those arrays that are available in all samples_batch folders
- **ignore_missing** (*bool, default=True*) – If False, will concatenate arrays in LCI folder only if all expected LCI arrays are present

concat_lcia_samples_arrays_from_method_tuple

To concatenate LCIA arrays from a method tuple, use `concat_lcia_samples_arrays_from_method_tuple`:

bw2preagg.concat_batches.**concat_lcia_samples_arrays_from_method_tuple**(*result_dir*, *method*, *sb_id_list=None*, *totals_or_per_exchanges='totals'*, *project_name=None*, *sim_name=None*, *dest=None*, *fail_if_samples_batches_differ=False*, *ignore_missing=True*)

Concatenate and save LCIA sample arrays in result dir

Parameters

- **result_type** (*str*) – Path to the directory where data used or generated by bw2preagg is saved.
- **method** (*tuple*) – LCIA method identification in brightway2 (tuple)
- **sb_id_list** (*list*) – List of samples_batch ids. If None, all samples_batch are processed.
- **totals_or_per_exchanges** (*str, default="totals"*) – Deal with totals or results per elementary flows
- **project_name** (*str*) – Name of the brightway2 project where the database is imported
- **sim_name** (*str, default=None*) – Name to give the directory in which results will be saved. If None, a default name is generated from sb_id_list
- **dest** (*str*) – Path to location where concatenated array directory will be saved
- **fail_if_samples_batches_different** (*bool, default=False*) – If False, will raise ValueError if arrays available in samples_batch folders are not the same If True, will concatenate only those arrays that are available in all samples_batch folders
- **ignore_missing** (*bool, default=True*) – If False, will concatenate arrays in LCI folder only if all expected LCI arrays are present

1.8 Run-through: LCI and LCIA array production

1.8.1 Scripts

The easiest way to produce the required arrays is to use scripts found in a dedicated github repository here. Simply download or clone the repository content on your computer.

1.8.2 Description of demo simulation for run-through

The general objective of this demo simulation is to:

- generate and save two sets of 100 iteration LCI arrays for the ecoinvent v3.6 cutoff database. These samples should be balanced for water and land transformation exchanges
- split the work as follows:
  - For the simulation on the Windows machine, all calculations are done within a *slice*, but across two sets of CPU

  - For the simulation on the computer cluster, calculations are done in 2 *slices*, and across four sets of CPU

- generate and save corresponding LCIA arrays for three LCIA methods:
  - ('IPCC 2013', 'climate change', 'GWP 100a')
  - ('ReCiPe Midpoint (H) V1.13', 'water depletion', 'WDP')
  - ('ReCiPe Midpoint (H) V1.13', 'natural land transformation', 'NLTP')
- Concatenate the LCIA scores across both samples batches and store these in a folder called "concat_preagg_demo".

1.8.3 Sample simulation on a single Windows computer

Install bw2preagg

Before being able to run the scripts, you need to install `bw2preagg`. You should carry out the following operations in a virtual environment. In this run-through, I use a Conda environment, but you can substitute it with something else:

```
conda create --name preagg_env python=3.7
activate preagg_env
```

You can then install the package and all dependencies via pip:

```
pip install bw2preagg
```

Modify parameter values

The parameters used by the various functions are centralized in the *params.txt* file, found in the single_windows_machine folder of the scripts folder you downloaded from github.

The parameters are currently those used in the sample simulation. You should review and modify as needed the parameter values. The parameters are grouped in three sections:

- Parameters that **must** be modified (e.g. filepaths)
- Parameters that can safely be modified (e.g. number of iterations)
- Parameters you should modify only if you know what you are doing

To modify the parameters, open the params.txt file in a text editor, make changes and save.

Running scripts

All the scripts to use are in the single_windows_machine folder of the scripts directory. To use, simply activate the environment where bw2preagg was installed, navigate to the *single_windows_machine* directory where the scripts can be found, and launch the appropriate batch file:

```
activate preagg_env
cd path/to/scripts/single_windows_machine
some_file.bat
```

You will be presented with the parameter values and asked to confirm that all is ok before proceeding. Click `Y` to proceed if everything looks good.

Set-up project

This creates the project, imports databases and generates common files as required.

You should set the following parameters in the `params.txt` file:

- result_dir
- database_dir
- database_name
- ecoinvent_version (limited to 3.4 and 3.6 for now)
- project_name

Then, in the conda command prompt:

```
activate preagg_env
cd path/to/scripts/single_windows_machine
setup_windows.bat
```

After running, all the files in the *common_files directory* of the new `result_dir` should should have been added.

Base presamples generation

To create a presamples package for all **A** and **B** matrix elements.

You should set the iterations and *samples_batch* parameters in the *params.txt* file.

Then, in the conda command prompt:

```
activate preagg_env
cd path/to/scripts/single_windows_machine
base_presamples_windows.bat
```

After running, there should be a new *presamples package* in the `presamples` subdirectory.

In the demo simulation, you would:

- set `iterations=100` and `samples_batch=0` in *params.txt*, save and then run `base_presamples_windows.bat`.
- leave `iterations=100` and set `samples_batch=1` in *params.txt*, save and then rerun `base_presamples_windows.bat`

Balancing presamples generation

This will create a presamples package for water and land transformation exchanges.

> **Warning:** This step takes a few hours. See the documentation on *balancing presamples*

In the conda command prompt:

```
activate preagg_env
cd path/to/scripts/single_windows_machine
balancing_presamples_windows.bat
```

After running, there should be a new *presamples package* in the `presamples` subdirectory for land transformation and for water exchanges.

In the demo simulation, you would:

- leave `iterations=100` and set `samples_batch=0` in *params.txt*, save and then run `balancing_presamples_windows.bat`.
- leave `iterations=100` and set `samples_batch=1` in *params.txt*, save and then rerun `balancing_presamples_windows.bat`

Generate LCI arrays

This will create LCI samples arrays.

Warning: This step takes a several days!!! See the documentation on *generating LCI arrays* for some strategies to keep time down. Therse calculations should be done on a dedicated computer or, better, on a *computer cluster* .

You should set the `parallel_jobs` parameter in the `params.txt` file.

Then, in the conda command prompt:

```
activate preagg_env
cd path/to/scripts/single_windows_machine
lci_windows.bat
```

After running this, there will be as many `numpy.ndarray` files stored in the subdirectory result_dir/probabilistic/LCI/0 as there are activities in the database. Each one has as many rows as elementary flows in the database, and as many columns as there are iterations.

In the demo simulation, you would:

- set `samples_batch=0` and `parallel_jobs=2` in *params.txt*, save and then run `lci_windows.bat`.
- set `samples_batch=1` and leave `parallel_jobs=2` in *params.txt*, save and then rerun `lci_windows.bat`.

Generate LCIA arrays

The method for which you can calculate LCIA arrays are found in the `/data/methods.json` file of the scripts directory. You select the method by setting the `method_idx` to the correct index value. For example, for ("IPCC 2013", "climate change", "GWP 100a"), we have `method_idx=714`.

Note that you can also *modify* the `/data/methods.json` file, but added methods should exist in the project in which you are working and the integrity of the json file should not be corrupted.

By default, probabilistic LCIA arrays with total impacts are generated. To generate LCIA arrays with results per elementary flow, set `return_per_exchange=True``in the single_windows_machine/params.txt file. To generate deterministic LCIA arrays, set ``result_type=deterministic`.

Then, in the conda command prompt:

```
activate preagg_env
cd path/to/scripts/single_windows_machine
lcia_windows.bat
```

In the demo simulation, you would:

- set `samples_batch=0` and `method_idx=714` in *params.txt*, save and then run `lcia_windows.bat`.
- change `samples_batch=1` and leave `method_idx=714` in *params.txt*, save and then rerun `lcia_windows.bat`.
- redo the first two steps with `method_idx=762` (for water scarcity).
- redo the first two steps with `method_idx=756` (for land transformation)

Concatenate LCIA arrays

This will concatenate the LCIA scores generated above.

Note: When using these scripts, the concatenated arrays will always contain all samples_batches in the *result_dir*. To use only a subset of these, you will need to interact with the *function* directly.

You should set the following parameters in the *params.txt* file:

- concat_result_type (LCI or LCIA)
- method_idx
- sim_name (name of folder in which to save arrays)
- dest (destination of arrays)
- fail_if_samples_batches_different (default=False)
- ignore_missing_concat (default=False)

Then, in the conda command prompt:

```
activate preagg_env
cd path/to/scripts/single_windows_machine
concat_windows.bat
```

In the demo simulation, you would:

- set `concat_result_type=LCIA`, `method_idx=714`, `sim_name=concat_preagg_demo` and `dest=some_valid_path` in *params.txt*, save and then run `concat_windows.bat`.
- change `method_idx=762` (for water scarcity) and rerun `concat_windows.bat`.
- change `method_idx=756` (for water scarcity) and rerun `concat_windows.bat`.

1.8.4 On a cluster

Given the time required to generate the LCI samples, you should really use a computer cluster if you have access to one.

More detail to follow.

1.9 Glossary

`campaign` Collection of presample resources (ordered).

`campaigns.db` Database in Brightway2 project used to manage presample ressources. Used by `bw2preagg` to store paths to base and balancing presample resources associated with a given *samples_batch*.

parallel_jobs Argument used in LCI generation function to determine how many parallel jobs to run on different CPUs of one computer.

presamples package Core data type of the presamples package. Folder containing data to inject in LCA matrices as well as matrix indices to identify where these data should be injected.

presamples resource Data on a presamples package, savec in `campaigns.db`.

project Brightway2 project, which is a self-contained, top-level container for LCI data, LCIA methods, etc. used in Brightway2.

Note: The argument *project_name* refers to the Brightway2 project used by `bw2preagg`.

project_name Name of the Brightway2 project where the LCI database and LCIA methods were imported and that contains the *campaigns.db*.

ref_bio_dict Dictionary containing biosphere matrix row indices for elementary flows.

samples_batch A set of presamples, LCI arrays and LCIA arrays that are all generated from the same base data. Used to split out the work iteration-wise (e.g. calculate 5 batches of 1000 iterations rather than one batch of 5000 iterations). The total calculation time is not decreased, but it allows one to generate batches on different computers and makes results available, albeit perhaps with less iterations than required, more quickly.

slices Subset of activity codes to treat as a set. Used when using computer clusters to generate LCI arrays.

1.9.1 Structure of the result_dir

```
├── common_files
│   ├── A_as_coo.xlsx (Row index, Col index, value for deterministic A matrix)
│   ├── A_as_coo_scipy.pickle (deterministic A matrix as SciPy sparse COO matrix)
│   ├── activity_dict.pickle
│   ├── B_as_coo.xlsx (Row index, Col index, value for deterministic B matrix)
│   ├── B_as_coo_scipy.pickle (deterministic B matrix as SciPy sparse COO matrix)
│   ├── bio_dict.pickle
│   ├── biosphere_description.xlsx (description of elementary flows in B matrix, per␣
↪row index)
│   ├── cfs.npy (array with characterization factors, methods as columns and␣
↪elementary flows as rows)
│   ├── cfs.xlsx (same as cfs.npy, but in Excel, with method names in columns)
│   ├── IO_Mapping.pickle
│   ├── ordered_activity_codes.json
│   ├── product_dict.pickle
│   └── technosphere_description.xlsx (description of products/activities in A matrix,
↪ per index)
├── presamples
│   ├── base_0 (base presamples package, samples_batch id=0)
│   ├── water_0 (water exchange balancing presamples package, samples_batch id=0)
│   ├── land_0 (land transformation exchange balancing presamples package, samples_
↪batch id=0)
│   ├── base_1
│   ├── water_1
│   ├── land_1
│   └── ...
├── deterministic
│   ├── LCI
|   |   ├── 0 (sample_batch id = 0)
|   |   |   ├── activity_code_0.npy (rows=elementary flows, columns=iterations)
|   |   |   ├── activity_code_1.npy
|   |   |   ├── activity_code_2.npy
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── ...
│   │   │   └── activity_code_n.npy
│   │   ├── 1 (sample_batch id)
│   │   │   ├── activity_code_0.npy
│   │   │   ├── activity_code_1.npy
│   │   │   ├── activity_code_2.npy
│   │   │   ├── ...
│   │   │   └── activity_code_n.npy
│   ├── LCIA
│   │   ├── method_abbrev_0
│   │   │   ├── totals
│   │   │   │   ├── 0 (sample_batch id)
│   │   │   │   │   ├── code_0.npy
│   │   │   │   │   ├── code_1.npy
│   │   │   │   │   ├── code_2.npy
│   │   │   │   │   ├── ...
│   │   │   │   │   └── code_n.npy
│   │   │   │   ├── 1 (sample_batch id)
│   │   │   │   │   ├── code_0.npy
│   │   │   │   │   ├── code_1.npy
│   │   │   │   │   ├── code_2.npy
│   │   │   │   │   ├── ...
│   │   │   │   │   └── code_n.npy
│   │   │   ├── per_exchange
│   │   │   │   ├── 0 (sample_batch id)
│   │   │   │   │   ├── code_0.npy
│   │   │   │   │   ├── code_1.npy
│   │   │   │   │   ├── code_2.npy
│   │   │   │   │   ├── ...
│   │   │   │   │   └── code_n.npy
│   │   │   │   ├── 1 (sample_batch id)
│   │   │   │   │   ├── code_0.npy
│   │   │   │   │   ├── code_1.npy
│   │   │   │   │   ├── code_2.npy
│   │   │   │   │   ├── ...
│   │   │   │   │   └── code_n.npy
├── probabilistic
│   ├── LCI
│   │   ├── 0 (sample_batch id = 0)
│   │   │   ├── activity_code_0.npy (rows=elementary flows, columns=iterations)
│   │   │   ├── activity_code_1.npy
│   │   │   ├── activity_code_2.npy
│   │   │   ├── ...
│   │   │   └── activity_code_n.npy
│   │   ├── 1 (sample_batch id)
│   │   │   ├── activity_code_0.npy
│   │   │   ├── activity_code_1.npy
│   │   │   ├── activity_code_2.npy
│   │   │   ├── ...
│   │   │   └── activity_code_n.npy
│   ├── LCIA
│   │   ├── method_abbrev_0
│   │   │   ├── totals
│   │   │   │   ├── 0 (sample_batch id)
│   │   │   │   │   ├── code_0.npy
│   │   │   │   │   ├── code_1.npy
│   │   │   │   │   ├── code_2.npy
```

(continues on next page)

(continued from previous page)

```
│   │   │   │   │   ├── ...
│   │   │   │   │   └── code_n.npy
│   │   │   │   ├── 1 (sample_batch id)
│   │   │   │   │   ├── code_0.npy
│   │   │   │   │   ├── code_1.npy
│   │   │   │   │   ├── code_2.npy
│   │   │   │   │   ├── ...
│   │   │   │   │   └── code_n.npy
│   │   │   │   ├── per_exchange
│   │   │   │   │   ├── 0 (sample_batch id)
│   │   │   │   │   │   ├── code_0.npy
│   │   │   │   │   │   ├── code_1.npy
│   │   │   │   │   │   ├── code_2.npy
│   │   │   │   │   │   ├── ...
│   │   │   │   │   │   └── code_n.npy
│   │   │   │   │   ├── 1 (sample_batch id)
│   │   │   │   │   │   ├── code_0.npy
│   │   │   │   │   │   ├── code_1.npy
│   │   │   │   │   │   ├── code_2.npy
│   │   │   │   │   │   ├── ...
│   │   │   │   │   │   └── code_n.npy
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index