WasteAndMaterialFootprint Documentation

Release 0.1.2

Stewart Charles McDowall

Dec 28, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

The WasteAndMaterialFootprint tool is a python package that allows one to calculate the waste and material footprint of any product or service inside of life cycle assessment (LCA) databases.

*** THESE DOCUMENTS ARE STILL UNDER CONSTRUCTION ***

The full api reference is available on this site, however

1.1 Motivation

1.2 Limitations

CHAPTER

TWO

INSTALLATION

2.1 Dependencies

The program is written in Python and the required packages are listed in the *requirements.txt* file. These should be installed automatically when installing the program.

The main dependencies are:

- brightway2
- premise
- wurst

2.2 Installation Instructions

It is recommended to use a fresh virtual environment to install the program.

You can simply clone the repo and run:

```
python src/WasteAndMaterialFootprint/main.py
```

This will not install any of the dependencies, so you will need to install them manually if you don't already have them.

A better option: the program can be installed using pip:

```
pip install WasteAndMaterialFootprint
```

Or, if you want to install the latest version from GitHub:

```
pip install git+https://github.com/Stew-McD/WasteAndMaterialFootprint.git
```

Or for an editable install (good for development and testing):

```
git clone https://github.com/Stew-McD/WasteAndMaterialFootprint.git
cd WasteAndMaterialFootprint
pip install -e .
```

CHAPTER

THREE

USAGE

The program can be used directly from the command line, or imported as a Python module. This will run the program using the default settings. See the configuration section for more information on how to change the settings.

3.1 Command Line

You should clone the repo, navigate to the `WasteAndMaterialFootprint` folder, and then run the program using:

```
python src/WasteAndMaterialFootprint/main.py
```

3.2 Python Module

The program can be imported as a Python module:

```
import WasteAndMaterialFootprint as wmf
wmf.run()
```

CHAPTER

FOUR

CONFIGURATION

By default, the program will create a folder `config` in the current working directory containing the default configuration files:

4.1 General Settings: `user_settings.py`

This is the main configuration file, the one that you might want to edit to match your project structure and your needs. By default, the program will take a brightway2 project named `default` and copy that to a new project named `SSP-cutoff`, which is then copied to a new project named `WMFootprint-SSP-cutoff`.

Doing it this way isolates the components and allows you to keep your original brightway2 project as it was. If space is an issue, you can set all of the project names to be the same.

If you are happy with the default settings, you can just run the program and it will create the databases for you. If you want to change the settings, you can edit the `user_settings.py` file that you can find in the `config` directory of your working directory.

These are some extracts from `user_settings.py` with the most important settings (the ones you might want to change) and their default values:

```
# Choose whether to use premise to create future scenario databases
use_premise = True
# Choose whether to use WasteAndMaterialFootprint to edit the databases (you could also␣
↪turn this off and just use the package as an easy way to make a set of future scenario␣
↪databases)
use_wmf = True

# Choose the names of the projects to use
project_premise_base = "default"
project_premise = "SSP-cutoff"
project_base = project_premise
project_wmf = f"WMFootprint-{project_base}"

# Choose the name of the database to use (needed for premise only, the WMF tool will run␣
↪all databases except the biospheres)
database_name = "ecoinvent-3.9.1-cutoff"

# if you want to use a fresh project
delete_existing_premise_project = False
delete_existing_wmf_project = False
```

(continues on next page)

(continued from previous page)

```
# Choose the premise scenarios to generate (see FutureScenarios.py for more details)
# Not all combinations are available, the code in FutureScenarios.py will filter out the␣
↪scenarios that are not possible
# the default is to have an optimistic and a pessimistic scenario with SSP2 for 2030,␣
↪2065 and 2100

models = ["remind"]
ssps = ["SSP2"]
rcps = ["Base","PkBudg500"]
years = [2030,2065,2100,]
```

4.2 Waste Search Settings: `queries_waste.py`

This file sets up search parameters for different waste and material flow categories, crucial for the `SearchWaste.py` script. It leverages a `.pickle` file created by `ExplodeDatabase.py`.

4.2.1 Categories

Handles various categories like digestion, composting, incineration, recycling, landfill, etc.

4.2.2 Query Types

Two sets of queries are created:

1. `queries_kg` for waste flows in kilograms.
2. `queries_m3` for waste flows in cubic meters.

4.2.3 Adjusting Search Terms

- **Search Keywords**: Tweak the `AND`, `OR`, `NOT` lists to refine your search.

4.2.4 Category-Specific Changes

- **Adding Categories**: You can add new categories to the `names` list.
- **Modifying Queries**: Update the query parameters for each category based on your requirements.

4.2.5 Optimizing Search Efficiency

You can choose to include or exclude whatever you want. For instance, "non-hazardous" is not included as it's derivable from other categories and slows down the process.

4.2.6 Validating Search Terms

Isolate the function of `SearchWaste.py` to validate your search terms. That means, turning off the other functions in `user_settings.py`, or running the module directly. You can achieve this by setting the following in `user_settings.py`:

```
use_premise = False
do_search = True
do_methods = False
do_edit = False
```

4.3 Material Search Settings: `queries_materials.py`

The `queries_materials` module creates demand methods in the WasteAndMaterialFootprint tool. It aligns with the EU CRM list 2023 and the ecoinvent database, incorporating additional strategic materials for comprehensive analysis. More can be easily added, as wished by the user.

This function uses the string tests `startswith` in `SearchMaterial.py` to identify activities beginning with the specified material name. This allows one to be more specific with the search terms (the `,` can be critical sometimes).

4.3.1 Structure and Customisation

Tuple Structure

- **First Part (Activity Name)**: Specifies the exact activity in the database (e.g., `market for chromium`).
- **Second Part (Material Category)**: Aggregates related activities under a common category (e.g., `chromium`), enhancing data processing efficiency.

Customisation Options

- **Add or Remove Materials**: Adapt the tuple list by including new materials or removing irrelevant ones.
- **Refine Search Terms**: Update material categories for a better fit with your database, ensuring precision in naming, especially with the use of commas.

4.3.2 Usage Considerations

- **Material Quantity**: The current list comprises over 40 materials. Modify this count to suit your project's scope.
- **Database Alignment**: Check that the material names correspond with your specific database version, like ecoinvent v3.9.1.

Example Tuples

- ("market for chromium", "chromium")
- ("market for coal", "coal")
- ("market for cobalt", "cobalt")
- ("market for coke", "coke")
- ("market for copper", "copper")
- ("market for tap water", "water")
- ("market for water,", "water")

CHAPTER

FIVE

EXAMPLES

See the *examples* directory for an examples of how to use the WasteAndMaterialFootprint package. The folder *batteries* contains a small case study using the package to calculate the waste and material footprints of several battery technologies in ecoinvent 3.10.

CHAPTER

SIX

API REFERENCE

6.1 WasteAndMaterialFootprint package

6.1.1 WasteAndMaterialFootprint.main module

main Module

Main module of the *WasteAndMaterialFootprint* tool.

This script serves as the entry point for the *WasteAndMaterialFootprint* tool. It orchestrates the overall process, including the setup and execution of various subprocesses like database explosion, material and waste searches, and the editing of exchanges.

The script supports both single and multiple project/database modes, as well as the option to use multiprocessing. It also facilitates the use of the premise module to generate future scenario databases.

Customisation:

- Project and database names, and other settings can be edited in *config/user_settings.py*.
- Waste search query terms can be customised in *config/queries_waste.py*.
- The list of materials can be modified in *config/queries_materials.py*.

Usage:

To use the default settings, run the script with *python main.py*. Arguments can be provided to change project/database names or to delete the project before running.

`WasteAndMaterialFootprint.main.`**`EditExchanges`**`(`*`args`*`)`

Edit exchanges in the database.

This function adds waste and material flows to the activities and verifies the database.

Parameters
`args` – Dictionary containing database and project settings.

Returns
None

WasteAndMaterialFootprint.main.**ExplodeAndSearch**(*args*)

Exploding the database into separate exchanges, searching for waste and material flows, and processing these results.

This includes:

- ExplodeDatabase.py
- SearchWaste.py
- SearchMaterial.py

Parameters
args – Dictionary containing database and project settings.

Returns
None

WasteAndMaterialFootprint.main.**run**()

Main function serving as the wrapper for the WasteAndMaterialFootprint tool.

This function coordinates the various components of the tool, including:
creating future scenario databases, setting up and processing each database for waste and material footprinting, and combining results into a custom database. adding LCIA methods to the project for each of the waste/material flows.

The function supports various modes of operation based on the settings in *config/user_settings.py*. Specifications for material and waste searches can be customised in *queries_materials*.

6.1.2 Submodules

6.1.3 WasteAndMaterialFootprint.ExchangeEditor module

ExchangeEditor Module

This module is responsible for editing exchanges with wurst and Brightway2. It appends relevant exchanges from the *db_wmf* (database containing waste and material exchange details) to activities identified by *WasteAndMaterialSearch()* in the specified project's database (*db_name*). Each appended exchange replicates the same amount and unit as the original technosphere waste and material exchange.

WasteAndMaterialFootprint.ExchangeEditor.**ExchangeEditor**(*project_wmf*, *db_name*, *db_wmf_name*)

Append relevant exchanges from *db_wmf* to each activity in *db_name* identified by *WasteAndMaterialSearch()*.

This function modifies the specified project's database by appending exchanges from the *db_wmf* to activities identified by *WasteAndMaterialSearch()*. The appended exchanges mirror the quantity and unit of the original technosphere waste and material exchange.

Parameters

- **project_wmf** (*str*) – Name of the Brightway2 project to be modified.
- **db_name** (*str*) – Name of the database within the project where activities and exchanges are stored.
- **db_wmf_name** (*str*) – Name of the database containing waste and material exchange details.

Returns
None. Modifies the given Brightway2 project by appending exchanges and logs statistics about the added exchanges.

Return type
None

Raises
Exception – If any specified process or exchange is not found in the database.

6.1.4 WasteAndMaterialFootprint.ExplodeDatabase module

ExplodeDatabase Module

This module is responsible for exploding a Brightway2 database into a single-level list of all exchanges. It utilizes the wurst package to unpack the database, explode it to a list of all exchanges, and save this data in a DataFrame as a .pickle binary file.

WasteAndMaterialFootprint.ExplodeDatabase.**ExplodeDatabase**(*db_name*)

Explode a Brightway2 database into a single-level list of all exchanges using wurst.

Parameters
db_name (*str*) – Name of the Brightway2 database to be exploded.

Returns
None The function saves the output to a file and logs the operation, but does not return any value.

Return type
None

6.1.5 WasteAndMaterialFootprint.FutureScenarios module

FutureScenarios Module

This module is responsible for creating future databases with premise.

WasteAndMaterialFootprint.FutureScenarios.**FutureScenarios**(*scenario_list*)

Create future databases with premise.

This function processes scenarios and creates new databases based on the premise module. It configures and uses user-defined settings and parameters for database creation and scenario processing.

Returns
None The function does not return any value but performs operations to create and configure databases in Brightway2 based on specified scenarios.

Raises
Exception – If an error occurs during the processing of scenarios or database creation.

WasteAndMaterialFootprint.FutureScenarios.**MakeFutureScenarios**()

Main function to run the FutureScenarios module. Only activated if *use_premise* is set to True in *user_settings.py*.

Calls the *FutureScenarios* function to create new databases based on the list of scenarios and settings specified in *user_settings.py*.

WasteAndMaterialFootprint.FutureScenarios.**check_existing**(*desired_scenarios*)

Check the project to see if the desired scenarios already exist, and if so, remove them from the list of scenarios to be created. Quite useful when running many scenarios, as it can take a long time to create them all, sometimes crashes, etc.

args: desired_scenarios (list): list of dictionaries with scenario details

returns: new_scenarios (list): list of dictionaries with scenario details that do not already exist in the project

WasteAndMaterialFootprint.FutureScenarios.**make_possible_scenario_list**(*filenames*, *desired_scenarios*, *years*)

Make a list of dictionaries with scenario details based on the available scenarios and the desired scenarios.

args: filenames (list): list of filenames of available scenarios desired_scenarios (list): list of dictionaries with scenario details years (list): list of years to be used

returns: scenarios (list): list of dictionaries with scenario details that are available and desired

6.1.6 WasteAndMaterialFootprint.MakeCustomDatabase module

MakeCustomDatabase Module

This module contains functions for creating an xlsx representation of a Brightway2 database and importing it into Brightway2.

Main functions: - dbWriteExcel: Creates an xlsx file representing a custom Brightway2 database. - dbExcel2BW: Imports the custom database (created by dbWriteExcel) into Brightway2.

WasteAndMaterialFootprint.MakeCustomDatabase.**dbExcel2BW**()

Import the custom database (created by dbWriteExcel) into Brightway2.

This function imports a custom Brightway2 database from an Excel file into the Brightway2 software, making it available for further environmental impact analysis.

Returns
None

WasteAndMaterialFootprint.MakeCustomDatabase.**dbWriteExcel**()

Create an xlsx file representing a custom Brightway2 database.

This function generates an Excel file which represents a custom database for Brightway2, using predefined directory and database settings.

Returns
Path to the generated xlsx file.

WasteAndMaterialFootprint.MakeCustomDatabase.**determine_unit_from_name**(*name*)

Determine the unit based on the name.

Parameters
name – The name from which to infer the unit.

Returns
The inferred unit as a string.

WasteAndMaterialFootprint.MakeCustomDatabase.**get_files_from_tree**(*dir_searchmaterial_results*, *dir_searchwaste_results*)

Collects filenames from the SearchMaterial and SearchWasteResults directories.

Parameters

- **dir_searchmaterial_results** – Directory path for SearchMaterial results.
- **dir_searchwaste_results** – Directory path for SearchWasteResults.

Returns
Sorted list of filenames.

6.1.7 WasteAndMaterialFootprint.MethodEditor module

MethodEditor Module

This module provides functions for adding, deleting, and checking methods related to waste and material footprints in a project.

Function Summary:

- *AddMethods*: Adds new methods to a project based on a custom biosphere database.
- *DeleteMethods*: Removes specific methods from a project, particularly those related to waste and material footprints.
- *CheckMethods*: Lists and checks the methods in a project, focusing on those associated with waste and material footprints.

WasteAndMaterialFootprint.MethodEditor.**AddMethods**()

Add methods to the specified project based on entries in the custom biosphere database.

Parameters

- **project_wmf** – Name of the project.
- **db_wmf_name** – Name of the database.

WasteAndMaterialFootprint.MethodEditor.**CheckMethods**()

Check methods associated with the "WasteAndMaterial Footprint" in the specified project.

Parameters
project_wmf – Name of the project.

WasteAndMaterialFootprint.MethodEditor.**DeleteMethods**()

Delete methods associated with the "WasteAndMaterial Footprint" in the specified project.

Parameters
project_wmf – Name of the project.

6.1.8 WasteAndMaterialFootprint.SearchMaterial module

SearchMaterial Module

This script loads data from '<db name>_exploded.pickle', runs search queries, and produces a CSV to store the results and a log entry. The search queries are formatted as dictionaries with fields NAME, CODE, and search terms keywords_AND, keywords_OR, and keywords_NOT. These queries are defined in *config/queries_waste.py*.

WasteAndMaterialFootprint.SearchMaterial.**SearchMaterial**(*db_name*, *project_wmf*)

Search for materials in a specified database and extract related information.

This function takes a database name as input, sets the project to the respective database, and looks for activities involving a predefined list of materials. It extracts relevant details of these activities, such as ISIC and CPC classifications, and saves the details to a CSV file. It also extracts related material exchanges and saves them to another CSV file.

Parameters

- **db_name** – The name of the database to search in.

- **project_wmf** – The Brightway2 project to set as current for the search.

Returns
None

Raises
Exception – If there is any error in reading the materials list from the file.

6.1.9 WasteAndMaterialFootprint.SearchWaste module

SearchWaste Module

This script loads data from '<db name>_exploded.pickle', runs search queries, and produces CSV files to store the results and a log entry. The search queries are formatted as dictionaries with fields NAME, CODE, and search terms keywords_AND, keywords_OR, and keywords_NOT. These queries are defined in *config/queries_waste.py*.

Functionality

Provides a function, *SearchWaste()*, that loads data from '<db name>_exploded.pickle', runs search queries, and produces result CSVs and log entries.

WasteAndMaterialFootprint.SearchWaste.**SearchWaste**(*db_name*)

Load data from '<db name>_exploded.pickle', run search queries, and produce result CSVs and log entries.

This function processes waste-related data from a given database and runs predefined queries to identify relevant waste exchanges. The results are saved in CSV files and log entries are created for each search operation.

Parameters
db_name (*str*) – The database name to be used in the search operation.

Note: The queries are defined in *config/queries_waste.py*.

6.1.10 WasteAndMaterialFootprint.VerifyDatabase module

VerifyDatabase Module

This module contains a function to verify a (WasteAndMaterialFootprint) database within a given project in Brightway2. It performs a verification by calculating LCA scores for random activities within the specified database using selected methods.

WasteAndMaterialFootprint.VerifyDatabase.**VerifyDatabase**(*project_name*, *database_name*, *check_material=True*, *check_waste=True*, *log=True*)

Verifies a database within a given project in Brightway2 by calculating LCA scores for random activities using selected methods.

This function assesses the integrity and validity of a specified database within a Brightway2 project. It performs LCA calculations on random activities using Waste Footprint and Material Demand Footprint methods, and logs the results.

Parameters

- **project_name** (*str*) – The name of the Brightway2 project.
- **database_name** (*str*) – The name of the database to be verified.

- **check_material** (*bool*) – If True, checks for Material Demand Footprint methods.
- **check_waste** (*bool*) – If True, checks for Waste Footprint methods.
- **log** (*bool*) – If True, logs the results.

Returns

Exit code (0 for success, 1 for failure).

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX