**NEIVA User Guide**

|  |
| --- |
| Text  Description automatically generated |

Version 1.1

Last updated-

Table of Contents

1. Home directories -----------------------------------------------------------------------------------2
2. Data (NEIVA\_v\_1.1/Data/) -----------------------------------------------------------------------3
   1. Base database ----------------------------------------------------------------------------------3
   2. Raw database ----------------------------------------------------------------------------------4
   3. Primary database ------------------------------------------------------------------------------4
3. Source Code (NEIVA\_v\_1.1/SRC/) -------------------------------------------------------------6
   1. Database design code flow chart -------------------------------------------------------------7
   2. Module 1: DataPrep ---------------------------------------------------------------------------8
   3. Module 2: DataInt ----------------------------------------------------------------------------11
   4. Module 3: DataCalc --------------------------------------------------------------------------13
4. Adding a new data
   1. Step 1: Insert in pdb
   2. Step 2: Modify the backend datasets
5. Tools (NEIVA\_v\_1.1/SRC/) --------------------------------------------------------------------15

6.1 How to use the tools -------------------------------------------------------------------------17

NEIVA is a biomass burning emissions inventory. It consists of data files, database design codes and useful tools. This document describes the overall database structure and codes.

**1. Home directories (pathway/NEIVA \_v\_1.1/)**

**Data**: All Emissions and property data is stored in this folder.

**Docs**: Contains documents and user guide.

**SRC**: Contains the database design codes and the backend database.

**Tools**: Contains useful function codes.

**2. Data (NEIVA\_v\_1.1/Data/)**

**base\_db:** The Akagi et al., 2011 supplement data sheets are converted to individual dataset and stored in this database. Referred as ‘bdb’ in the codes and this document

**raw\_db:** Data from selected publications are stored here. Referred as ‘rdb’ in the codes and this document.

**primary\_db:** Data from base and raw database are reformatted and stored here. This is the main input in the module 1 (explained in section 4). Referred as ‘pdb’ in the codes and this document.

**Integrated\_dataset:** The integrated dataset of pdb datasets. The main output of module 2.

**Reommended\_EF:** The main output of module 3.

**2.1 Base database**

The bdb comprises of the following data files:

|  |  |
| --- | --- |
| **Dataset Name** | **Source** |
| boreal\_forest | Akagi et al., 2011 supplementary table; S2: Boreal Forest |
| chaparral | Akagi et al., 2011 supplementary table; S6: Chaparral |
| charcoal\_burning | Akagi et al., 2011 supplementary table; S10: Charcoal burning |
| charcoal\_making | Akagi et al., 2011 supplementary table; S9: Charcoal making |
| cookstove | Akagi et al., 2011 supplementary table; S8: Cooking stoves |
| crop\_residue | Akagi et al., 2011 supplementary table; S13: Crop Residue |
| dung\_burning | Akagi et al., 2011 supplementary table; S11: Dung burning |
| garbage\_burning | Akagi et al., 2011 supplementary table; S14: Garbage burning |
| open\_cooking | Akagi et al., 2011 supplementary table; S7: Open cooking |
| pasture\_maintenance | Akagi et al., 2011 supplementary table; S12: pasture maintenance |
| peatland | Akagi et al., 2011 supplementary table; S5: Peatland |
| savanna | Akagi et al., 2011 supplementary table; S1: Savanna |
| temperate\_forest | Akagi et al., 2011 supplementary table; S4: Temperate Forest |
| tropical\_forest | Akagi et al., 2011 supplementary table; S3: Tropical Forest |

**2.2 Raw database**

The rdb comprises of the following datafiles.

|  |  |
| --- | --- |
| **Dataset Name** | **Source** |
| rdb\_bf\_hatch17 | doi.10.5195/acp-17-1471-2017; Sheet name- Black Spruce |
| rdb\_chrb\_stockwell16 | doi.10.5194/acp-16-11043-2016; Table 6 |
| rdb\_cookingfire\_stockwell16 | doi.10.5194/acp-16-11043-2016; Table S8 |
| rdb\_crr\_hatch17 | doi.10.5194/acp-17-1471-2017; Sheet name- Chinese rice straw |
| rdb\_crr\_holder17 | doi.10.1016/j.atmosenv.2016.06.043; Table S2, S3, S5 and Table 3. |
| rdb\_crr\_lasko18 | doi.10.1016/j.envpol.2018.01.08; Table 2 |
| rdb\_crr\_liu16 | doi.10.1002/2016JD025040; Table 3 |
| rdb\_crr\_stockwell16 | doi.10.5194/acp-16-11043-2016; Table S9 |
| rdb\_cs\_coffey17 | doi.10.102/acs.est.7b02436; Table S2 |
| rdb\_cs\_fleming18 | doi.10.5194/acp-18-15169-2018; Table 1 |
| rdb\_gb\_stockwll16 | doi.10.519/acp-16-11043-2016; Table S7 |
| rdb\_gb\_yokelson13 | doi.10.5194/acp-13-89-2013; Table S1 |
| rdb\_goetz18 | doi.10.5194/acp-18-14653-2018; Supplement section-3,4 |
| rdb\_hatch15 | doi.10.5194/acp-15-1865-2015; Table S1 |
| rdb\_jayarathne18 | doi.10.5194/acp-18-2585-2018; Table 1,2,3 |
| rdb\_koss18 | doi.10.5194/acp-18-3299-2018; Table S3 |
| rdb\_p\_hatch17 | doi.10.5195/acp-17-1471-2017; Sheet name- Peat |
| rdb\_p\_jayarathne18 | doi.10.5194/acp-18-2585-2018; Table 2 section 3.2 |
| rdb\_p\_roulster18 | doi.10.1029/2017/JD027827; Section- ‘Discussion and Conclusion’ |
| rdb\_p\_smith17 | doi.10.1002/2017GB005709; Table 3 |
| rdb\_p\_stockwel16 | doi:10.5194/acp-16-11711-2016; Table S2 |
| rdb\_p\_watson19 | doi.10.5194/acp-19-14173-2019; Table 2,3,4 |
| rdb\_pokhrel16 | doi.10.5194/acp-16-9549-2016; Table S2 |
| rdb\_selimovic18 | doi.10.5194/acp-18-2929-2018; Table S2 |
| rdb\_stockwell15 | doi.10.5194/acp-15-845-2015; Table S2 |
| rdb\_sv\_desservettaz17 | doi.10.1002/2016JD025925; Table 4; Column- ‘This Study’ |
| rdb\_tmf\_hatch17 | doi.10.5195/acp-17-1471-2017; Sheet name- Ponder pine |
| rdb\_tmf\_liu17 | doi.10.1002/2013GL058392; Table 3 |
| rdb\_tmf\_muller16 | doi.10.5194/acp-16-3813-2016; Table-2,3 |
| rdb\_trf\_hodgson18 | doi.10.5194/acp-18-5619-2018; Table 3; row- ‘This Study’ |

**2.3 Primary database**

The bdb and rdb datasets are reformatted and stored in pdb. The datasets from rdb that are stored in pdb have identical name but the ‘rdb\_’ is replaced with ‘pdb\_’. The datasets from bdb that are stored in pdb have ‘pdb\_akagi11\_’ string attached in the dataset name.

**3. Source Code (NEIVA\_v\_1.1/SRC/)**

The database design process has three modules- DataPrep, DataInt and DataCalc. Each module has an execution/main script. The execution/main scripts import function from subscripts, datasets and finally export output dataset.

**Usage information:** All scripts import default\_GenPathways.py script. This script generates required file pathways based on where a user saves the NEIVA\_v\_1.1 folder. A user should not modify this script unless he/she wish to add a new pathway in the script.

**3.1 Database design code flow diagram**

This flow diagram displays the execution script and subscripts of each module. The main input and output dataset of the modules are shown in blue. The datasets that can be modified by users are in dashed orange box. Datasets in black dashed box shows they are produced in module 2 then they are used in module 3 as backend input dataset.

|  |
| --- |
| **Diagram  Description automatically generated** |
| Figure 1: Database design code flow diagram. |

**3.2 Module 1: DataPrep**

The pdb datasets are verified in this module. It shows flags if a dataset doesn’t pass the verifications. If the datasets, passes all verifications it shows that the datasets are prepared for next steps. The execution script **Run\_DataPrep** imports pdb datasets and functions from **ScanClean** subscript. A user may visit ScanClean to see the verification codes.

**3.3 Module 2: DataInt**

The pdb datasets are integrated into a single dataset. This dataset is then arranged and sorted. Finally, it is exported as ‘Integrated\_Dataset’ in NEIVA\_v\_1.1/Data/ directory. The execution script **Run\_DataInt** uses various other subscripts to complete this task. The task of each subscript is discussed below-

The **DataInt\_mainFunc** has codes for data integration process. The ‘pm\_order\_seq’ is called from backend database in this script. This dataset has different classes of particulate matter ordered in sequence; it is used to sort particulate matter.

**Merge\_MultLumCom** further merges the multiple lumped compounds in a specific formula. In some study emission factor is reported for a group of compounds instead a single compound, these are referred as lumped compound and ‘LumCom’ in code. For instance, 1,3-Butadiene + 1,2-Butadiene, assorted hcs, unknown, C6 carbonyls are referred as lumped compounds. There are multiple lumped compounds in some specific formula. Most of the cases they refer to similar compounds. These rows are merged. Because we do not want to account the same compound twice in our analysis.

**Get\_lumCom\_Spec** splits a lumped compound into individual compounds then searches the individual compounds within the data frame and arranges them together.

**Gen\_lntData\_efcol\_InfoTable** creates an information dataset of ‘Integrated\_dataset’. This dataset has measurement type and fire type of each EF column of ‘Integrated\_dataset’.

**3.4 Module 3: DataCalc**

The calculation steps are carried out in this module and the Recommended\_EF dataset is produced. The execution script **Run\_DataCalc** imports the following subscripts-

**﻿** **ER\_ADJ\_calc** carries out the lab EF emission ratio adjusted to field EF calculation steps.

**﻿AVG\_n\_FC\_calc** carries out the fractional contribution calculation of lumped compounds. Finally, calculates the average EF across studies.

**verify\_calc\_steps** verify the calculation steps. The results and figures of verification are imported to SRC/backend\_db/verify\_calc/ directory.

**4. Adding a new data**

As shown in figure 1 A new data is added in pdb. The orange box highlights the datasets where a user should modify while a new data is added. The codes can incorporate new data in the process, no modification is required.

**4.1 Step 1: Insert in pdb**

The new dataset structure should be compatible with pdb datasets. There are a few requirements for a new dataset before inserting in to pdb. A user is recommended to visit the pdb datasets to better follow the requirements-

* The dataset should have mm (molecular weight), chemical formula (formula), compound, pollutant category, emission factor and id columns.

**Usage information:** There are tools in NEIVA\_v\_1.1/Tools/adding\_new\_data.py.

* The EF column names have specific pattern. An EF column name starts with ‘EF’ string. The combined string of first author last name and publication year is referred as ‘study’. The study is attached at the end of column name. In between ‘EF’ and ‘study’ fuel type is mentioned. Visit SRC/backend\_db/pdb\_fuelType2fireType\_info dataset. This has fuel list of different fuels. Find the fuel of the new data and insert in the column name. For example, ‘EF ponder\_pine\_hatch15’. Please notice, there is only one space after ‘EF’, all other space is replaced by ‘\_’. If the fuel of the new data is not in the list. Add the fuel in this list.
* The dataset names have a specific pattern depending on which database it is stored. If a dataset is stored in rdb the dataset name starts with ‘rdb\_’ string. Similarly, if a dataset is stored in pdb the dataset name starts with ‘pdb\_’. Datasets that have e EF for single fire type a brief annotation for that fire type is attached in the name. For instance, ‘rdb\_bf’ means the dataset is stored in rdb and it has EF data for only boreal forest fire type. If the dataset has EF data for multiple fire type, then brief annotation is not attached. At the end, the last name of the first author is attached.

|  |  |
| --- | --- |
| **Fire type** | **Brief annotation** |
| Savanna | sv |
| Boreal forest | bf |
| Tropical forest | trf |
| Temperate forest | tmf |
| Peatland | p |
| Chaparral | chp |
| Open cooking | ocook |
| Cookstove | cs |
| Charcoal making | chrm |
| Charcoal burning | chrb |
| Pasture maintenance | pstm |
| Garbage burning | gb |
| Dung burning | db |

**4.1 Step 2: Modify the backend datasets**

A user is required to modify a few datasets in SRC/backend\_db/ to incorporate the new dataset in the codes. A list of backend datasets and instructions on modification is given below-

|  |  |
| --- | --- |
| **Dataset name** | **Instructions on modification** |
| pdb\_all\_dataset\_info | Add a new row in this dataset. Specify the new dataset name, fire type and measurement type. If the dataset has EF for multiple fire type, then specify ‘multiple fuel’ in ‘fire type’ column. |
| pdb\_fuelType2FireType\_info | If the fuel of the new data is not in this dataset. Add the fuel and fire type of the fuel in this dataset. |
| pdb\_cookingfire\_info | If the new data has cookstove that is not in this dataset. Add the cookstove in this dataset. |
| nmog\_multLumCom\_slc\_no\_merge | Inspect ‘nmog\_MultLumCom’ dataset. If a user decides not to merge the lumped compounds specify the formula in this dataset. |
| nmog\_slc\_id\_altName | In case a user wants to alter compound name, specify the altered name in ‘alter name’ column. |
| nmog\_LumCom\_altName | In case a user wants to alter compound name, specify the altered name in ‘alter name’ column. |

**5. Tools (NEIVA\_v\_1.1/Tools/)**

Useful functions are given in different script files. Users may call these functions in their own script and use them.

**DataScan** has various data checking functions-

﻿**Scan\_UnwantedString:** Scans and report if \ufeff is spotted in any cell.

**ScanCell:** Scans and report if input string is found/not found in any cell.

**ScanCol:** Scans and report if input string is spotted in the input column.

**Scan\_dupId:** Reports files if duplicated id is spotted.

**ScanMolecule:** Scan a molecule in formula column and reports found/not found.

**ScanCompound:** Scan a compound in ‘compound’ column and reports found/not found.

**ScanID:** Scan an id in ‘compound’ column and reports found/not found.

**check\_pollutant\_category\_col:** Checks the pollutant category column. It checks the categories.

Reports if any unmatched category is spotted.

**check\_inorganic\_gas:** Checks the pollutant category column. It matches the compounds. Report if any unmatched compound is spotted.

**DataAlt** has various data replacing, altering, and dropping functions-

**ScanDel\_UnwantedString:** Reports and deletes \ufeff from cells.

**ScanDel\_Cell:** Reports and deletes input\_string from cells.

**CleanSpace\_ColName:** Replaces unwanted space from column name.

**CleanSpace\_cell:** Replaces unwanted space from cell.

**RepVal\_by\_Com:** Replaces a value from a column with an input value. It asks to input a compound from users.

**RepVal\_by\_id:** Replaces a value from a column with an input value.

**altid:** Replaces old id with new id.

**RepVal\_from\_col:** Replaces a value from a column with another value.

**DropCol:** Drops column from selected data frames.

**5.1 How to use the tools**

Here, is an example on how to use the tools-

|  |
| --- |
| >> Import the script and all functions. |
|  |
| >> Type any function name from the script. |
|  |
| >> It asks for input. Type the input. |
|  |
| >> Type the database for this task. I have selected bdb. |
| Table  Description automatically generated |
| >> Shows the output. |
| Table  Description automatically generated |