nuMoM2b$_{p}reprocessing Documentation$

Release 0.0.1

Alexander L. Hayes

Jun 24, 2019

Getting Started:

A Python package for pre-processing `nuMoM2b` data with configuration files.

```
{
  "comments": [
    "Screening questions asked during visit 1.",
    "These are primarily yes/no questions.",
  ],
  "log_file": "nuMoM2b_screening_questions.log",
  "csv_path": "~/Desktop/PrecisionHealth/Data/numom_data/",
  "target": {
    "name": "Ancillary/Pregnancy_outcomes.csv",
    "variables": ["PublicID", "oDM"]
  },
  "files": [
    {
      "name": "Screening_Admin_Visits/Visit1.csv",
      "variables": [
        "PublicID", "V1AD03", "V1AD05", "V1AD17", "V1AD18",
        "V1AF01", "V1AF03", "V1AF03a1", "V1AF03a2",
        "V1AF03a3", "V1AF03a4", "V1AF03b", "V1AF03c",
        "V1AF04", "V1AF14", "V1AG01", "V1AG02",
        "V1AG03", "V1AI01"
      ]
    }
  ],
  "groupings": []
}
```

This is one component within a wider research project for predicting adverse events over the course of a woman's pregnancy. This package serves the dual role of assisting with pre-processing tasks and for producing reproducible partitions of the data based on configuration files.

CHAPTER 1

Getting Started

This section describes how to get organized and get the code running. This will try to keep concepts as accessible as possible, but some familiarity with Python projects, UNIX systems, and Git would certainly be helpful.

1.1 Getting Organized

The running assumption will be that the nuMoM2b data is contained in a directory (likely unzipped from a `numom2b.zip`), and this `nuMoM2b_preprocessing/` repository is available on your local machine.

For example, we find it helpful to organize work as follows (though these may be tweaked with config files):

```
Precision-Health-Initiative/
├── Data/
│   ├── pregnancy_outcomes.csv
│   ├── Screening.csv
│   └── Visit1.csv
└── nuMoM2b_preprocessing/
    ├── README.md
    └── numom2b_preprocessing/
```

`nuMoM2b_preprocessing` may be cloned from GitHub:

```
git clone https://github.com/hayesall/nuMoM2b_preprocessing.git
```

If you're using Anaconda, this would be a good time to create an environment:

```
conda create -n PHI python=3.7
conda activate PHI
```

... and install dependencies.

```
pip install -r numom2b_preprocessing/requirements.txt
```

1.2 Documentation

Documentation is currently hosted at https://doc.nuMoM2b.org, but local copies may also be built using Sphinx.

A separate requirements file is in the `documentation/` directory.

```
pip install -r documentation/requirements.txt
```

A `make.bat` and `Makefile` are included:

```
cd documentation
make html
```

If the build is successful, a copy will reside in a new `build/html/` directory.

```
open build/html/index.html           # (macOS)
xdg-open build/html/index.html       # (Linux)
```

CHAPTER 2

Architecture

The architecture of `nuMoM2b_preprocessing` is intended to be fairly simple, having one component for **parsing configuration options** and another for performing the **aggregations and joins** on the contents of the data.

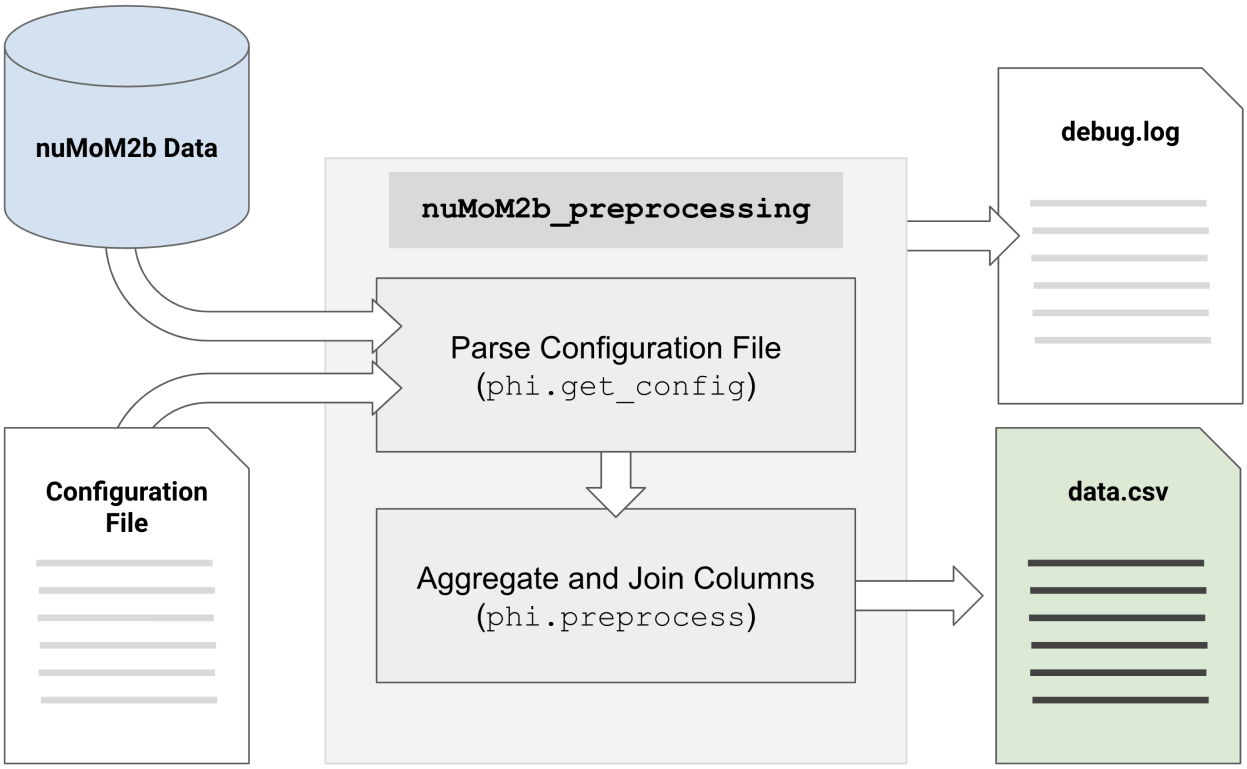
The main source of complexity is therefore in writing the configuration files. However, these can be fairly consistent, and can be shared.

Input:

1. Configuration File
2. nuMoM2b data base

Output:

1. `data.csv` contains a single table
2. `debug.log` (Optional) contains a log of all operations performed


nuMoM2b Data
nuMoM2b_preprocessing
debug.log
Parse Configuration File
(phi.get_config)
Configuration File
Aggregate and Join Columns
(phi.preprocess)
data.csv

CHAPTER 3

Configuration Files

Perhaps you want to know whether a woman's *weight during the first visit* might be informative for determining whether or not she develops gestational diabetes later in her pregnancy.

Weight was recorded in pounds or kilograms, and gestational diabetes is what we want to predict. We need to:

1. Convert weight to a common unit (we will use pounds here)
2. Combine the two measurements into a single measurement of weight
3. Compare this relation with gestational diabetes

Each of these may be defined as options in a configuration file. The configuration files here are written in JSON (JavaScript Object Notation). JSON itself is not fully covered here (there are many tutorials elsewhere online), but the main ideas should be fairly straightforward after seeing a few examples.

We'll begin with a skeleton and build toward a file with everything we need. This file does not work yet, but shows us all the keys that we will need to work with.

```
{
    "comments": ["comments are ignored."],
    "log_file": "debug.log",
    "csv_path": "../FullData/numom_data/",
    "target": {},
    "files": [],
    "groupings": []
}
```

Let's start with the `"target"`. The target is the file that contains information about what we want to predict. It is described separately from the other files to make a few things more convenient behind-the-scenes, and to leave room in the future for possibly defining behavior depending on what is being predicted.

The `"target"` key needs two values: `"name"` and `"variables"`. These allow us to specify where the file is (relative to the `"csv_path"`) and what variables we want to include.

`oDM` indicates whether the woman developed gestational diabetes, and `"PublicID"` is a primary key which identifies her across records.

```
{
    "comments": ["comments are ignored."],
    "log_file": "debug.log",
    "csv_path": "../FullData/numom_data/",
    "target": {
        "name": "Ancillary/Pregnancy_outcomes.csv",
        "variables": ["PublicID", "oDM"]
    },
    "files": [],
    "groupings": []
}
```

Adding this to `example_config.json` is enough to produce a `data.csv` when we run the script as a command-line module.

```
$ python -m numom2b_preprocessing -c example_config.json
```

```
oDM
3.0
1.0
3.0
2.0
```

Let's modify the `"files"` key to include the variables for weight. There are two variables that encode this measure, "V1BA01_KG" when weight was recorded in kilograms and "V1BA01_LB" when weight was recorded in pounds. Once again we include "PublicID" to keep track of which record corresponds to which person.

```
{
    "comments": ["comments are ignored."],
    "log_file": "debug.log",
    "csv_path": "../FullData/numom_data/",
    "target": {
        "name": "Ancillary/Pregnancy_outcomes.csv",
        "variables": ["PublicID", "oDM"]
    },
    "files": [
        {
            "name": "Screening_Admin_Visits/Visit1.csv",
            "variables": ["PublicID", "V1BA01_KG", "V1BA01_LB"]
        }
    ],
    "groupings": []
}
```

```
$ python -m numom2b_preprocessing -c example_config.json
```

```
oDM,V1BA01_KG,V1BA01_LB
3.0,NaN,180
1.0,NaN,130
3.0,NaN,144
2.0,76,NaN
```

Now that we have the variables we want, we can use the `"groupings"` section to convert them to common units. Operations defined in the `"groupings"` section are executed from top to bottom.

First, we multiply the `"V1BA01_KG"` variable by 2.20462, which converts the measurements to pounds. Then, we take the last measurement between `"V1BA01_LB"` and `"V1BA01_KG"`, then place the result (`"rename"`) into a

new "V1BA01" variable.

This can be written as follows:

```
{
  "comments": ["comments are ignored."],
  "log_file": "debug.log",
  "csv_path": "../FullData/numom_data/",
  "target": {
      "name": "Ancillary/Pregnancy_outcomes.csv",
      "variables": ["PublicID", "oDM"]
  },
  "files": [
    {
      "name": "Screening_Admin_Visits/Visit1.csv",
      "variables": ["PublicID", "V1BA01_KG", "V1BA01_LB"]
    }
  ],
  "groupings": [
    {
      "operator": "multiply_constant",
      "columns": ["V1BA01_KG"],
      "constant": 2.20462
    },
    {
      "operator": "last",
      "columns": ["V1BA01_LB", "V1BA01_KG"],
      "rename": "V1BA01"
    }
  ]
}
```

```
$ python -m numom2b_preprocessing -c example_config.json
```

```
oDM,V1BA01
3.0,180
1.0,130
3.0,144
2.0,167.551
```

Generalizing from this example, configuration files allow us to specify:

1. The variables of interest
2. Where those variables are located
3. How to transform and aggregate the variables

3.1 What is Next?

The outcome from nuMoM2b_preprocessing is a data.csv file. The exact types of machine learning or statistical modeling you perform next is up to you.

CHAPTER 4

Categorical Screening Questions

This config files extracts results of screening questions asked during the screening interview. Most of these are the sort of questions which would be asked in a clinical setting, but also might be filled in by a patient.

`"csv_path"` and `"target.name"` may need to be adjusted depending on file locations on your specific machine.

```
{
  "comments": [
    "Screening questions asked during visit 1.",
    "These are primarily yes/no questions.",
  ],
  "log_file": "nuMoM2b_screening_questions.log",
  "csv_path": "~/Desktop/PrecisionHealth/Data/numom_data/",
  "target": {
    "name": "Ancillary/Pregnancy_outcomes.csv",
    "variables": ["PublicID", "oDM"]
  },
  "files": [
    {
      "name": "Screening_Admin_Visits/Visit1.csv",
      "variables": [
        "PublicID", "V1AD03", "V1AD05", "V1AD17", "V1AD18", "V1AF01",
        "V1AF03", "V1AF03a1", "V1AF03a2", "V1AF03a3", "V1AF03a4",
        "V1AF03b", "V1AF03c", "V1AF04", "V1AF14", "V1AG01", "V1AG02",
        "V1AG03", "V1AI01"
      ]
    }
  ],
  "groupings": []
}
```

CHAPTER 5

Continuous Visit-1 Measurements

This normalizes measurements taken as part of the screening questions or the visit-1 measurements.

`"csv_path"` and `"target.name"` may need to be adjusted depending on file locations on your specific machine.

```
{
  "comments": [
    "Numeric attributes from the screening questions and first visit."
  ],
  "log_file": "debug.log",
  "csv_path": "~/Desktop/PrecisionHealth/Data/numom_data/",
  "files": [
    {
      "name": "Screening_Admin_Visits/Visit1.csv",
      "variables": [
        "PublicID", "V1BA02a", "V1BA02b", "V1BA02c",
        "V1BA03a", "V1BA03b", "V1BA03c",
        "V1BA04a", "V1BA04b", "V1BA04c",
        "V1BA05a", "V1BA05b", "V1BA05c",
        "V1BA06a1", "V1BA06a2",
        "V1BA06b1", "V1BA06b2",
        "V1BA07a", "V1BA07b", "V1BA07c"
      ]
    }
  ],
  "target": {
    "name": "Ancillary/Pregnancy_outcomes.csv",
    "variables": ["PublicID", "oDM"]
  },
  "groupings": [
    {
      "operator": "last",
      "columns": ["V1BA02a", "V1BA02b", "V1BA02c"],
      "rename": "V1BA02_last"
    },
    {
```

(continues on next page)

(continued from previous page)

```
      "operator": "last",
      "columns": ["V1BA04a", "V1BA04b", "V1BA04c"],
      "rename": "V1BA04_last"
    },
    {
      "operator": "last",
      "columns": ["V1BA07a", "V1BA07b", "V1BA07c"],
      "rename": "V1BA07_last"
    },
    {
      "operator": "last",
      "columns": ["V1BA03a", "V1BA03b", "V1BA03c"],
      "rename": "V1BA03_last"
    },
    {
      "operator": "last",
      "columns": ["V1BA05a", "V1BA05b", "V1BA05c"],
      "rename": "V1BA05_last"
    },
    {
      "operator": "last",
      "columns": ["V1BA06a1", "V1BA06a2"],
      "rename": "V1BA06a_last"
    },
    {
      "operator": "last",
      "columns": ["V1BA06b1", "V1BA06b2"],
      "rename": "V1BA06b_last"
    }
  ]
}
```

CHAPTER 6

Overview `numom2b_preprocessing`

Warning: Features are fairly experimental and may change between versions.

6.1 numom2b_preprocessing.get_config.parameters()

Read a configuration file and parse the parameters.

```
>>> import numom2b_preprocessing
>>> PARAMETERS = numom2b_preprocessing.parameters("phi_config.json")
```

6.2 numom2b_preprocessing.preprocess.run()

Use the configuration parameters to manipulate the data.

This should generally be used in tandem with the `get_config` module.

```
>>> import numom2b_preprocessing
>>> PARAMETERS = numom2b_preprocessing.parameters("phi_config.json")
>>> df = numom2b_preprocessing.run(PARAMETERS)
```

CHAPTER 7

get_config

Read parameters from a `config.json` file and make these available through a `get_config.parameters()` function.

`numom2b_preprocessing.get_config.`**`parameters`**(*config='phi_config.json'*)

Read the parameters from a configuration file.

Parameters **`config`** (*str*) – JSON configuration path/file_name.json

Returns dict

7.1 Available Options

- `"csv_path"`: Path to directory where all `.csv` files are located
- `"files"`: List of entries naming individual files
  - `"name"`: `.csv` file name (`csv_path` will be appended to the beginning)
  - `"variables"`: Names of columns to include from an individual file
- `"target"`: Target file (what you want to predict)
  - `"name"`: `.csv` file name (`csv_path` will be appended to the beginning)
  - `"variables"`: Names of columns to include from the target file
- `"groupings"`: List of objects describing how to aggregate columns
  - `"operator"`: "mean", "last", or "count"
  - `"columns"`: List of column names to apply aggregation operator to. *These columns are dropped after aggregating*
  - `"rename"`: [**optional**] Name to apply to the new column of aggregated values

  If none is specified, the column is named according to which columns were aggregated and what operator was used

7.2 Example Usage

```
>>> from numom2b_preprocessing import get_config
>>> _params = get_config.parameters(config="phi_config.json")
```

7.3 Example File

```
{
  "comments": [
    "Screening questions asked during visit 1.",
    "These are primarily yes/no questions.",
  ],
  "log_file": "nuMoM2b_screening_questions.log",
  "csv_path": "~/Desktop/PrecisionHealth/Data/numom_data/",
  "target": {
    "name": "Ancillary/Pregnancy_outcomes.csv",
    "variables": ["PublicID", "oDM"]
  },
  "files": [
    {
      "name": "Screening_Admin_Visits/Visit1.csv",
      "variables": [
        "PublicID", "V1AD03", "V1AD05", "V1AD17", "V1AD18", "V1AF01",
        "V1AF03", "V1AF03a1", "V1AF03a2", "V1AF03a3", "V1AF03a4",
        "V1AF03b", "V1AF03c", "V1AF04", "V1AF14", "V1AG01", "V1AG02",
        "V1AG03", "V1AI01"
      ]
    }
  ],
  "groupings": []
}
```

CHAPTER 8

preprocess

Combine and aggregate columns across tables.

numom2b_preprocessing.preprocess.**run**(*config_parameters*)

Parameters **config_parameters** –

Returns pandas.core.frame.DataFrame

Preprocess the data according to the configuration parameters.

conf_parameters should be passed from *numom2b_preprocessing.get_config.parameters()*

CHAPTER 9

VariableCleaner

Clean individual variables.

class numom2b_preprocessing.clean_variables.**VariableCleaner**(*data_frame*)

Clean individual variables in-place.

clean(*operations_list*)

Parameters **operations** – List of dictionaries with 'operator', 'columns', and 'value' keys.

CHAPTER 10

ColumnAggregator

Aggregate columns.

class numom2b_preprocessing.aggregate_columns.**ColumnAggregator**(*data_frame*)

Mutate a data frame according to configuration parameters. Drop the modified columns.

aggregate(*operations_list*)

Mutate a data frame according to configuration parameters.

Parameters **operations_list** – List of dictionaries with 'operator' and 'columns' keys.

Examples:

```
>>> data_frame = pd.DataFrame({"ID": [0, 1, 2], "a": [3.3, 4.5, 1.2], "b": [3,
↪ 2, 4]})
>>> ca = ColumnAggregator()
>>> ca.aggregate(
...     [
...         {
...             "operator": "mean",
...             "columns": ["a, "b"],
...             "rename": "mean_a_b",
...         },
...     ],
... )
```

CHAPTER 11

RowFilter

Conditionally filter rows.

class numom2b_preprocessing.row_filter.**RowFilter**(*data_frame*)
Filter rows from a DataFrame.

filter(*operations_list*)

Parameters **operations_list** – List of dictionaries with 'operator' and 'columns' keys.

Maintained by Alexander L. Hayes, a Ph.D. student with the Indiana University ProHealth Group working on the Precision Health Initiative (ϕ). Alexander can be reached at hayesall@iu.edu.

CHAPTER 12

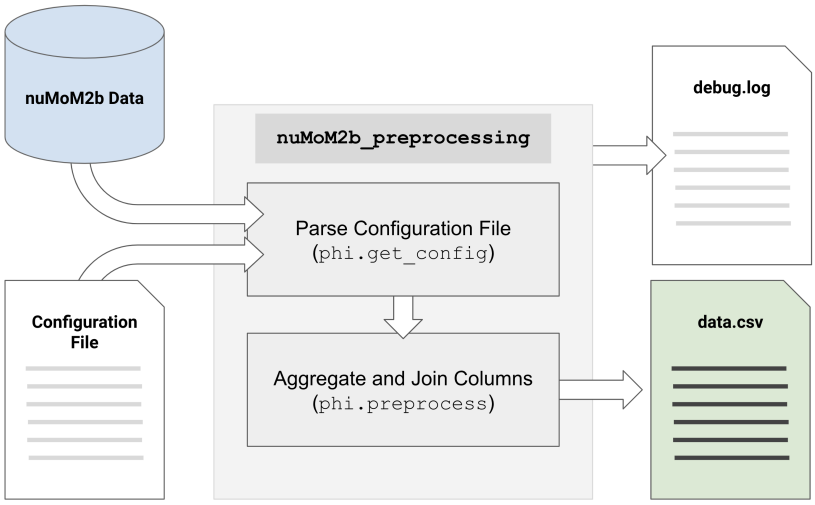
Getting Started

Pointers for getting nuMoM2b_preprocessing working on your local machine.

CHAPTER 13

Architecture

High-level overview of how this project is organized.



CHAPTER 14

Writing Configuration Files

Configuration files define which variables should be used and how they should be aggregated. This is a worked example for writing a configuration file.

```
{
  "csv_path": "../FullData/numom_data/",
  "target": {
      "name": "Ancillary/Pregnancy_outcomes.csv",
      "variables": ["PublicID", "oDM"]
  },
  "files": [
    {
      "name": "Screening_Admin_Visits/Visit1.csv",
      "variables": ["PublicID", "V1BA01_KG", "V1BA01_LB"]
    }
  ]
}
```

14.1 Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index