pydnameth Documentation

Release 0.2.4

Aaron Blare

May 01, 2020

CONTENTS:

1 pydnameth **3**
- 1.1 Documentation . . . 3
- 1.2 Features . . . 3
- 1.3 Copyrights . . . 3

2 Installation **5**
- 2.1 Stable release . . . 5
- 2.2 From sources . . . 5

3 Usage **7**
- 3.1 Import . . . 7
- 3.2 Info . . . 7
- 3.3 Config . . . 7
- 3.4 Released Experiments . . . 12
- 3.5 Usage Examples . . . 16

4 Contributing **25**
- 4.1 Types of Contributions . . . 25
- 4.2 Get Started! . . . 26
- 4.3 Deploying . . . 28

5 Credits **29**
- 5.1 Development Lead . . . 29
- 5.2 Contributors . . . 29

6 History **31**
- 6.1 0.2.4 (2019-11-06) . . . 31
- 6.2 0.1.0 (2019-01-24) . . . 31

7 Indices and tables **33**

Python Module Index **35**

Index **37**

CHAPTER

ONE

PYDNAMETH

DNA Methylation Analysis Package

This package provides some pipelines for analysis mythylation data. The main goal is to find correlations between methylation on the one hand, and age, sex, disease, and other characteristics of subjects on the other.

Examples of free-access methylation datasets:

- GSE40279
- GSE87571

1.1 Documentation

Available at https://pydnameth.readthedocs.io.

1.2 Features

- Defining best age-predictors CpGs for different subjects subsets.
- Defining best observable-specic CpGs (sex-specific, disease-specific) which are differently methylated with age.
- Building Epigenetic Clock.
- Plotting subjects distribution depending on the observable (sex, disease).
- Plotting methylation profiles for CpGs.

1.3 Copyrights

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install `pydnameth`, run this command in your terminal:

```
$ pip install -U pydnameth
```

This is the preferred method to install pydnameth, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

If you want to save figures locally as `.pdf` and `.png` run the command:

```
$ conda install -c plotly plotly-orca psutil
```

2.2 From sources

The sources for pydnameth can be downloaded from the GitHub repo.

You can either clone the public repository:

```
$ git clone git://github.com/AaronBlare/pydnameth
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

3.1 Import

To use pydnameth in a project:

```
import pydnameth as pdm
```

3.2 Info

As a rule, 4 files are provided in each methylation dataset:

- `betas.txt` - contains methylation data itself. Rows correspond to individual CpGs, and columns correspond to subjects.
- `annotations.txt` - contains information about CpGs. Rows correspond to individual CpGs, and columns correspond to CpG's characteristics (gene, bop, coordinate, etc.).
- `observables.txt` - contains information about subjects. Rows correspond to subjects, and columns correspond to subject's observables (age, gender, disease, etc.).
- `cells.txt` - contains information about cell types population. For example, if DNA methylation profiles taken from human whole blood, then for each patient a different proportion of blood cells types is possible. Rows in file correspond to subjects, and columns correspond different cell types proportions.

The first line in each file is usually a header. File names and file extensions may differ, but content is the same. Currently supported only `.txt` extension.

These files must be located in the same directory. After running experiments, new directories with results and cached data files with `.pkl` and `.npz` extensions will appear in this directory.

For all experiments provided by `pydnameth` you need to specify config information.

3.3 Config

For each experiment you need to create instances:

- `pdm.Data`
- `pdm.Annotations`
- `pdm.Attributes`

3.3.1 Data

`pdm.Data` contains information about dataset. For creating instance of `pdm.Data` you need to specify next fields:

name

Name of the file without extension (currently supported only `.txt` extension), which contains methylation data.

Example:

```
name = 'betas'
```

path

Path to directory, which contains `base` directory.

Example:

```
path = 'C:/Data'
```

base

Name of the directory, where the necessary files are located and where the files with the results will be saved.

Example:

```
base = 'GSE87571'
```

Example

Example of creating `pdm.Data` instance:

```
data = pdm.Data(
    name='betas',
    path='C:/Data',
    base='GSE40279'
)
```

3.3.2 Annotations

`pdm.Annotations` allows you to define a subset of CpGs that will be considered in the experiment. For creating instance of `pdm.Annotations` you need to specify next fields:

name

Name of the file without extension (currently supported only `.txt` extension), which contains information about CpGs.

Example:

```
name = 'annotations'
```

exclude

Name of the file without extension (currently supported only `.txt` extension), which contains CpGs to exclude. If equals to `'none'`, then no excluded CpGs.

Example:

```
exclude = 'none'
```

cross_reactive

Should cross-reactive CpGs be considered in the experiment? Currently supported options (`string`):

- `'ex'` - excluded all cross-reactive CpGs.
- `'any'` - all CpGs are considered.

Example:

```
cross_reactive = 'ex'
```

snp

Should SNP CpGs be considered in the experiment? Currently supported options (`string`):

- `'ex'` - excluded all SNP CpGs.
- `'any'` - all CpGs are considered.

Example:

```
snp = 'ex'
```

chr

What chromosomes are considered in the experiment? Currently supported options (`string`):

- `'NS'` - CpGs only on non-sex chromosomes are considered.
- `'X'` - CpGs only on X chromosome are considered.
- `'Y'` - CpGs only on Y chromosome are considered.
- `'any'` - all CpGs are considered.

Example:

```
chr = 'NS'
```

gene_region

Should we consider CpGs which are mapped on genes? Currently supported options (`string`):

- `'yes'` - only CpGs which are mapped on genes are considered.
- `'no'` - only CpGs which are not mapped on genes are considered.
- `'any'` - all CpGs are considered.

Example:

```
gene_region = 'yes'
```

geo

CpGs on what geo-types should be considered? Currently supported options (`string`):

- `'shores'` - only CpGs on shores are considered.
- `'shores_s'` - only CpGs on southern shores are considered.
- `'shores_n'` - only CpGs on northern shores are considered.
- `'islands'` - only CpGs on islands are considered.
- `'islands_shores'` - only CpGs on islands or shores are considered.
- `'any'` - all CpGs are considered.

Example:

```
gene_region = 'any'
```

probe_class

What CpGs probe class should be considered? Currently supported options (`string`):

- `'A'` - class A CpGs are considered.
- `'B'` - class B CpGs are considered.
- `'C'` - class C CpGs are considered.
- `'D'` - class D CpGs are considered.
- `'A_B'` - class A and B CpGs are considered.
- `'any'` - all CpGs are considered.

Example:

```
probe_class = 'any'
```

Example

Example of creating `pdm.Annotations` instance:

```
annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)
```

3.3.3 Attributes

`pdm.Attributes` describes information about subjects. For creating instance of `pdm.Attributes` you need to specify next fields:

target

Name of target observable column (`string`)

Example:

```
target = 'age'
```

observables

Specifies observables of subjects under consideration. Should be `pdm.Observables` instance. For creating `pdm.Observables` instance you need to specify:

- `name` - name of the file without extension (currently supported only `.txt` extension), which contains information about subjects.

Example:

```
name = 'observables'
```

- `types` - python `dict` with `key` - header of target observable and `value` - value of target observable. Also values can be `'any'` if you want to consider all existing values.

Example:

```
{'gender': 'F'}
```

cells

Specifies cell types population. Should be `pdm.Cells` instance. For creating `pdm.Cells` instance you need to specify:

- `name` - name of the file without extension (currently supported only `.txt` extension), contains information about cell types population.

Example:

```
name = 'cells'
```

- `types` - python `list` of cell types which should be considered in the experiment (`string` headers in `file_name`) or string `'any'` if you want to consider all cells types.

Example:

```
types = ['Monocytes', 'B', 'CD4T', 'NK', 'CD8T', 'Gran']
```

Example

Example of creating `pdm.Attributes` instance:

```
observables = pdm.Observables(
    name='observables',
    types={'gender': 'F'}
)

cells = pdm.Cells(
    name='cells',
    types='any'
)

attributes = pdm.Attributes(
    target='age',
    observables=observables,
    cells=cells
)
```

3.4 Released Experiments

The name of the functions provided by the `pydnameth` package are follow the next logic:

- First part is data type for the experiment. For example, `betas`, `residuals` or `attributes`.
- Second part answers the question: `WHAT WE WANT TO DO?`. For example, `table` - table with data and characteristics processing, `plot` - data plotting.
- Third part answers the question: `HOW WE WANT TO DO?`. Specifies the method for the experiment. For example, `linreg` - linear regression method.

Currently released functions:

`pydnameth.scripts.develop.betas.clock.`**`betas_clock_special`**(*data*, *annotations*, *attributes*, *file*, *method_params=None*)

Producing epigentic clock, using best CpGs which are provided in input file.

Epigentic clock represents as table: Each row corresponds to clocks, which are built on all CpGs from the previous rows including the current row. Columns:

- item: CpG id.
- aux: gene, on which CpG is mapped.
- R2: determination coefficient of linear regression between real and predicted target observable. A statistical measure of how well the regression line approximates the data points.

- r: correlation coefficient of linear regression between real and predicted target observable.
- evs: explained variance regression score.
- mae: mean absolute error regression loss.
- rmse: root mean square error

Possible parameters of experiment:

- `'type'`: type of clocks.

  Possible options:

  `'all'`: iterative building of clocks starting from one element in the model, ending with `'size'` elements in the model.

  `'single '`: building of clocks only with `'size'` elements in the model.

  `'deep'`: iterative building of clocks starting from one element in the model, ending with `'size'` elements in the model, but choosing all possible combinations from `'size'` elements.
- `'part'`: the proportion of considered number of subject in the test set. From `0.0` to `1.0`.
- `'size'`: maximum number of exogenous variables in a model.
- `'runs'` number of bootstrap runs in model

Parameters

- **`data`** – pdm.Data instance, which specifies information about dataset.
- **`annotations`** – pdm.Annotations instance, which specifies subset of CpGs.
- **`attributes`** – pdm.Attributes instance, which specifies information about subjects.
- **`method_params`** – parameters of experiment.

`pydnameth.scripts.develop.betas.plot.`**`betas_plot_scatter`**(*data, annotations, attributes, observables_list, child_method=<Method.linreg: 'linreg'>, data_params=None, method_params=None*)

Plotting methylation level from observables as scatter for provided subjects subsets and provided CpG list.

Possible parameters of experiment:

- `'x_range'`: can be `'auto'` or list with two elements, which are borders of target axis.
- ...

Parameters

- **`data`** – pdm.Data instance, which specifies information about dataset.
- **`annotations`** – pdm.Annotations instance, which specifies subset of CpGs.
- **`attributes`** – pdm.Attributes instance, which specifies information about subjects.
- **`observables_list`** – list of subjects subsets. Each element in list is dict, where `key` is observable name and `value` is possible values for this observable.
- **`method_params`** – parameters of experiment.

`pydnameth.scripts.develop.betas.table.`**`betas_table_aggregator_linreg`**(*data, annotations, attributes, observables_list, data_params=None, method_params=None*)

Producing table with information about observable-specificity of target data type and target observable for each CpG.

Columns:

- item: CpG id.
- aux: gene, on which CpG is mapped.
- area_intersection_rel: relative intersection area of polygons which is equals area of polygon(s) intersection to area of polygons union ratio.
- slope_intersection_rel: relative intersection area of allowed regions for slopes of linear regression.
- max_abs_slope: maximal absolute slope between all provided subjects subsets
- ...
- z_value: number of standard deviations by which data point is above the mean value.
- The considered data point is the difference between two linear regressions slopes.
- abs_z_value: absolute z_value
- p_value: probability of rejecting the null hypothesis that the difference in slopes is zero.
- ...

For each subjects subset the next columns are added to the resulting table:

- R2_***: determination coefficient. A statistical measure of how well the regression line approximates the data points.
- intercept_***: estimated value of the intercept of linear regression.
- slope_***: estimated value of the slope of linear regression.
- intercept_std_***: standard error of the estimate of the intercept of linear regression.
- slope_std_***: standard error of the estimate of the slope of linear regression.
- intercept_p_value_***: p-value for the intercept of linear regression.
- slope_p_pvalue_***: p-value for the slope of linear regression.
- ...

Where *** is the name of subjects subset.

Possible parameters of experiment:

- None

Parameters

- **`data`** – pdm.Data instance, which specifies information about dataset.
- **`annotations`** – pdm.Annotations instance, which specifies subset of CpGs.

- **attributes** – pdm.Attributes instance, which specifies information about subjects.
- **observables_list** – list of subjects subsets. Each element in list is dict, where `key` is observable name and `value` is possible values for this observable.
- **method_params** – parameters of experiment.

`pydnameth.scripts.develop.betas.table.`**`betas_table_linreg`**(*data, annotations, attributes, method_params=None*)

Producing table with information for linear regression between beta values and methylation level for each CpG.

Each row corresponds to specific CpG.

Columns:

- item: CpG id.
- aux: gene, on which CpG is mapped.
- R2: determination coefficient. A statistical measure of how well the regression line approximates the data points.
- intercept: estimated value of the intercept of linear regression.
- slope: estimated value of the slope of linear regression.
- intercept_std: standard error of the estimate of the intercept of linear regression.
- slope_std: standard error of the estimate of the slope of linear regression.
- intercept_p_value: p-value for the intercept of linear regression.
- slope_p_pvalue: p-value for the slope of linear regression.
- ...

Possible parameters of experiment:

- None

Parameters

- **data** – pdm.Data instance, which specifies information about dataset.
- **annotations** – pdm.Annotations instance, which specifies subset of CpGs.
- **attributes** – pdm.Attributes instance, which specifies information about subjects.
- **method_params** – parameters of experiment.

`pydnameth.scripts.develop.observables.plot.`**`observables_plot_histogram`**(*data, annotations, attributes, observables_list, method_params=None*)

Plotting histogram for target observable distribution for provided subjects subsets and provided CpG list.

Possible parameters of experiment:

- `'bin_size'`: bin size for numeric target.

  For categorical target is not considered.
- `'opacity'`: opacity level. From `0.0` to `1.0`.
- `'barmode'`: type of barmode.

  Possible options:

  `'overlay'` for overlaid histograms.

  `'stack'` for stacked histograms.
- `'x_range'`: can be `'auto'` or list with two elements, which are borders of target axis.

Parameters

- **data** – pdm.Data instance, which specifies information about dataset.
- **annotations** – pdm.Annotations instance, which specifies subset of CpGs.
- **attributes** – pdm.Attributes instance, which specifies information about subjects.
- **cpg_list** – List of CpGs for plotting
- **observables_list** – list of subjects subsets. Each element in list is dict, where `key` is observable name and `value` is possible values for this observable.
- **method_params** – parameters of experiment.

3.5 Usage Examples

3.5.1 attributes_plot_observables_histogram

```
import pydnameth as pdm

data = pdm.Data(
    name='cpg_beta',
    path='C:/Data',
    base='GSE87571'
)

annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)

observables = pdm.Observables(
    name='observables',
    types={}
)
```

(continues on next page)

(continued from previous page)

```
cells = pdm.Cells(
    name='cells',
    types='any'
)

attributes = pdm.Attributes(
    target='age',
    observables=observables,
    cells=cells
)

observables_list = [
    {'gender': 'F'},
    {'gender': 'M'}
]

pdm.attributes_plot_observables_histogram(
    data=data,
    annotations=annotations,
    attributes=attributes,
    observables_list=observables_list,
    params={
        'bin_size': 1.0,
        'opacity': 0.75,
        'barmode': 'overlay'
    }
)
```

3.5.2 cpg_plot_methylation_scatter

```
import pydnameth as pdm

cpg_list = [
    'cg13982318',
    'cg11868595',
    'cg08900404'
]

data = pdm.Data(
    name='cpg_beta',
    path='C:/Data',
    base='GSE87571'
)

annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)
```

(continues on next page)

(continued from previous page)

```
observables = pdm.Observables(
    name='observables',
    types={}
)

cells = pdm.Cells(
    name='cells',
    types='any'
)

attributes = pdm.Attributes(
    target='age',
    observables=observables,
    cells=cells
)

observables_list = [
    {'gender': 'F'},
    {'gender': 'M'}
]

pdm.cpg_plot_methylation_scatter(
    data=data,
    annotations=annotations,
    attributes=attributes,
    observables_list=observables_list,
    cpg_list=cpg_list,
    params={
        'x_range': [10, 110]
    }
)
```

3.5.3 cpg_proc_clock_linreg

```
import pydnameth as pdm

data = pdm.Data(
    name='cpg_beta',
    path='C:/Data',
    base='GSE87571'
)

annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)

cells = pdm.Cells(
    name='cells',
```

(continues on next page)

(continued from previous page)

```
    types='any'
)

obs_list = [
    {'gender': 'F'},
    {'gender': 'M'},
    {'gender': 'any'}
]

for obs in obs_list:

    observables = pdm.Observables(
        name='observables',
        types=obs
    )

    attributes = pdm.Attributes(
        target='age',
        observables=observables,
        cells=cells
    )

    pdm.cpg_proc_clock_linreg(
        data=data,
        annotations=annotations,
        attributes=attributes,
        params={
            'type': 'all',
            'part': 0.25,
            'size': 100,
            'runs': 100,
        }
    )
```

3.5.4 cpg_proc_table_linreg

```
import pydnameth as pdm

data = pdm.Data(
    name='cpg_beta',
    path='C:/Data',
    base='GSE87571'
)

annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)
```

(continues on next page)

(continued from previous page)

```
cells = pdm.Cells(
    name='cells',
    types='any'
)

obs_list = [
    {'gender': 'F'},
    {'gender': 'M'},
    {'gender': 'any'}
]

for obs in obs_list:

    observables = pdm.Observables(
        name='observables',
        types=obs
    )

    attributes = pdm.Attributes(
        target='age',
        observables=observables,
        cells=cells
    )

    pdm.cpg_proc_table_linreg(
        data=data,
        annotations=annotations,
        attributes=attributes
    )
```

3.5.5 cpg_proc_table_variance_linreg

```
import pydnameth as pdm

data = pdm.Data(
    name='cpg_beta',
    path='C:/Data',
    base='GSE87571'
)

annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)

cells = pdm.Cells(
    name='cells',
    types='any'
)
```

(continues on next page)

(continued from previous page)

```
obs_list = [
    {'gender': 'F'},
    {'gender': 'M'},
    {'gender': 'any'}
]

for obs in obs_list:

    observables = pdm.Observables(
        name='observables',
        types=obs
    )

    attributes = pdm.Attributes(
        target='age',
        observables=observables,
        cells=cells
    )

    pdm.cpg_proc_table_variance_linreg(
        data=data,
        annotations=annotations,
        attributes=attributes
    )
```

3.5.6 cpg_proc_table_polygon

```
import pydnameth as pdm

data = pdm.Data(
    name='cpg_beta',
    path='C:/Data',
    base='GSE87571'
)

annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)

observables = pdm.Observables(
    name='observables',
    types={}
)

cells = pdm.Cells(
    name='cells',
    types='any'
```

(continues on next page)

(continued from previous page)

```
)

attributes = pdm.Attributes(
    target='age',
    observables=observables,
    cells=cells
)

observables_list = [
    {'gender': 'F'},
    {'gender': 'M'}
]

pdm.cpg_proc_table_polygon(
    data=data,
    annotations=annotations,
    attributes=attributes,
    observables_list=observables_list
)
```

3.5.7 cpg_proc_table_z_test_linreg

```
import pydnameth as pdm

data = pdm.Data(
    name='cpg_beta',
    path='C:/Data',
    base='EPIC'
)

annotations = pdm.Annotations(
    name='annotations',
    exclude='none',
    cross_reactive='ex',
    snp='ex',
    chr='NS',
    gene_region='yes',
    geo='any',
    probe_class='any'
)

observables = pdm.Observables(
    name='observables',
    types={}
)

cells = pdm.Cells(
    name='cells',
    types='any'
)

attributes = pdm.Attributes(
    target='age',
    observables=observables,
    cells=cells
```

(continues on next page)

(continued from previous page)

```
)

observables_list = [
    {'gender': 'F'},
    {'gender': 'M'}
]

pdm.cpg_proc_table_z_test_linreg(
    data=data,
    annotations=annotations,
    attributes=attributes,
    observables_list=observables_list
)
```

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/AaronBlare/pydnameth/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

pydnameth could always use more documentation, whether as part of the official pydnameth docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/AaronBlare/pydnameth/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up `pydnameth` for local development.

4.2.1 Environment

We use Anaconda Python distribution and PyCharm IDE

You can use any other Python distributions and IDEs.

4.2.2 GitHub Init

1. Fork the `pydnameth` repo on GitHub.
2. Travis CI and AppVeyor are continuous integration tools. Login using your Github credentials. It may take a few minutes for Travis CI and AppVeyor to load up a list of all your GitHub repos. Turn on testing your origin repository on Travis CI and AppVeyor. To do this, log into your personal account, synchronize the repositories with GitHub and add `pydnameth` project.

4.2.3 Local

1. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pydnameth.git
```

2. Create a virtual environment for your project and activate it:

```
$ conda create --name your-env-name python=x.x
$ activate your-env-name
```

   where `your-env-name` is the name you want to call your environment, and replace `x.x` with the Python version you wish to use (for example 3.7)

3. Go to the root of `pydnameth` project and install project in 'editable' or 'develop' mode while you are working on it:

```
$ pip install --editable .
```

   `.` refers to the current working directory. This allows the project to be both installed and editable in project form.

4. Install all libs in `requirements_dev.txt`:

```
$ pip install -r requirements_dev.txt
```

   If with some package `err-pkg` error occurs, try:

```
$ conda install err-pkg --channel=conda-forge
```

If you want to save figures locally as `.pdf` and `.png` run the command:

```
$ conda install -c plotly plotly-orca psutil
```

And repeat command:

```
$ pip install -r requirements_dev.txt
```

5. If `requirements_dev.txt` file was updated, you should repeat the command:

```
$ pip install -r requirements_dev.txt
```

6. If you update `requirements_dev.txt` file, you should recreate environment for `tox` (only locally):

```
$ tox --recreate -e env
```

Where `env` is name for `tox` environment.

4.2.4 Git Pipeline

1. `master` branch is always in production, tested and complete.
2. `development` is the branch closest to `master` but has changes that should be merged to `master`. Anyone who starts working on a new feature or bug fixing should always branch out from `development`.
3. Branch out from `development` with new branch for bug or feature:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass flake8 and the tests:

```
$ tox
```

5. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

4.2.5 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.7, 3.6 and 3.5. Check https://travis-ci.org/AaronBlare/pydnameth/pull_requests and make sure that the tests pass for all supported Python versions.

4.3 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in `HISTORY.rst`). Then run:

```
$ git add HISTORY.rst
$ git commit -m "Changelog for upcoming release x.x.x."
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Kalyakulina Alena <kalyakulina.alena@gmail.com>
- Yusipov Igor <yusipov.igor@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 0.2.4 (2019-11-06)

- Project encapsulating.

6.2 0.1.0 (2019-01-24)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

pydnameth.scripts.develop.betas.clock, 12
pydnameth.scripts.develop.betas.plot, 13
pydnameth.scripts.develop.betas.table, 13
pydnameth.scripts.develop.observables.plot, 15

INDEX

B

`betas_clock_special()` (*in module pydnameth.scripts.develop.betas.clock*), 12
`betas_plot_scatter()` (*in module pydnameth.scripts.develop.betas.plot*), 13
`betas_table_aggregator_linreg()` (*in module pydnameth.scripts.develop.betas.table*), 13
`betas_table_linreg()` (*in module pydnameth.scripts.develop.betas.table*), 15

O

`observables_plot_histogram()` (*in module pydnameth.scripts.develop.observables.plot*), 15

P

`pydnameth.scripts.develop.betas.clock` (*module*), 12
`pydnameth.scripts.develop.betas.plot` (*module*), 13
`pydnameth.scripts.develop.betas.table` (*module*), 13
`pydnameth.scripts.develop.observables.plot` (*module*), 15