phosphomatics-api-wrapper

Release 0.0.1

Michael Leeming

Jan 07, 2022

CONTENTS:

CHAPTER

ONE

INTRODUCTION

Phosphomatics is a public resource for statistical and downstream analysis of phosphoproteomics data. The graphical web resource can be accessed here. A limited API is available that allows users to automate the process of uploading and initial data processing. Here, we provide a python wrapper to this API.

This is very much a work in progress so use at your own risk.

1.1 Installation

Create a virtual environment:

```
virtualenv venv
source venv/bin/activate
```

Either install from PyPi:

```
pip install phosphomatics-api-wrapper
```

Or clone the GitHub repo and install manually:

```
git clone https://github.com/mgleeming/phosphomatics-api-wrapper.git
cd phosphomatics-api-wrapper
pip install -r requirements.txt
python setup.py install
```

1.2 API Keys

A unique key is required to use the phosphomatics API. To obtain an API key, contact the devlopers.

CHAPTER

TWO

PARAMETER FILES

Phosphomatics uses a yaml-style parameter file to configure initial data processing and track the creation of phosphorylation site data groups. The parameter file can be opened and edited in any plain text editor, or, alternatively, in python using the yaml library. An example parameter data file can be downloaded here and the text is copied at the end of this page.

2.1 Structure

The minimal parameter file that can be used for initial processing of data consists of five blocks; `sampleAlias`, `columnAssignments`, `sampleMaps`, `comparisons` and `processing`. The order of these blocks is not important.

2.1.1 Sample Alias Block

The `sampleAlias` block allows you to specify an alias for your sample names. This can be useful since many proteomics search software label these columns with the raw mass spectrometry data file name which is frequently long and verbose. Here, we alias columns in the original input files ('QUANT_CTRL', 'QUANT_CTRL_1'…) to 'CTRL', 'CTRL_1' which removes the unnecessary '**QUANT_**' prefix:

```
sampleAlias:
  QUANT_CTRL: CTRL
  QUANT_CTRL_1: CTRL_1
  QUANT_CTRL_2: CTRL_2
  QUANT_THZ1: THZ1
  QUANT_THZ1_1: THZ1_1
  QUANT_THZ1_2: THZ1_2
  QUANT_U0126: U0126
  QUANT_U0126_1: U0126_1
  QUANT_U0126_2: U0126_2
```

Note: After a sample alias has been set, the alias must be used at each subsequent point in the parameter file.

2.1.2 Column Assignments Block

The column assignments block contains information about how individual columns from a larger data file are to be utilised for analysis. This section consists of 7 sub sections.

We must specify which columns are to be used for protein identifiers, phosphorylation site and residue specification as well as quantitative data. An example of these sections is below:

```
upidColumn: <COLUMN_FOR_PROTEIN_UNIPROT_ID>
residueColumn: <COLUMN_FOR_PHOSPHORYLATED_RESIDUE_(S/T/Y)>
siteColumn: <COLUMN_FOR_PHOSPHORYLATION_POSITION_IN_PROTEIN>
quantColumns:
- <COLUMN_FOR_SAMPLE_1_QUANT_DATA>
- <COLUMN_FOR_SAMPLE_2_QUANT_DATA>
  ...
- <COLUMN_FOR_SAMPLE_X_QUANT_DATA>
```

For example, using the phosphomatics example data, this `columnAssignments` block would be:

```
columnAssignments
  upidColumn: ID
  residueColumn: Residue
  siteColumn: Position
  quantColumns:
  - CTRL
  - CTRL_1
  - CTRL_2
  - THZ1
  - THZ1_1
  - THZ1_2
  - U0126
  - U0126_1
  - U0126_2
```

2.1.3 Sample Map Block

The next section allows us to specify which samples correspond to which treatment groups. Here, keys are sample aliases and values are treatment groups. We've created three treatment groups called `CTRL`, `THZ1` and `U0126`:

```
sampleGroupMap:
  CTRL: CTRL
  CTRL_1: CTRL
  CTRL_2: CTRL
  THZ1: THZ1
  THZ1_1: THZ1
  THZ1_2: THZ1
  U0126: U0126
  U0126_1: U0126
  U0126_2: U0126
```

The last section allows us to control the order in which data is presented. For example, with time series data, we usually want to plot/tabulate data in order of increasing time post-treatment. In the block below, indices can be entered beside individual files (1,2,3...) and the data will then be displayed with the specified order. The sample indexed 1 will be presented left most and the highest index will be presented right-most.:

```
sampleIndexMap:
  CTRL: 1
  CTRL_1: 2
  CTRL_2: 3
  THZ1: 4
  THZ1_1: 5
  THZ1_2: 6
  U0126: 7
  U0126_1: 8
  U0126_2: 9
```

The final sampleMaps block would be:

```
sampleMaps:
  sampleGroupMap:
    CTRL: CTRL
    CTRL_1: CTRL
    CTRL_2: CTRL
    THZ1: THZ1
    THZ1_1: THZ1
    THZ1_2: THZ1
    U0126: U0126
    U0126_1: U0126
    U0126_2: U0126
  sampleIndexMap:
    CTRL: 1
    CTRL_1: 2
    CTRL_2: 3
    THZ1: 4
    THZ1_1: 5
    THZ1_2: 6
    U0126: 7
    U0126_1: 8
    U0126_2: 9
```

2.1.4 Comparisons Block

The comparison block allows you to pre-define group comparisons. For each comparison, phosphomatics will conduct t-tests and phosphorylation sites that meet specified fold-change and p-value thresholds will be placed into new data groups.

To define a group comparison:

```
comparisons:
- foldChangeThreshold: '1'
  group1: CTRL
  group2: THZ1
  name: CTRL_THZ1
  pvalThreshold: '2'
- foldChangeThreshold: '1'
  group1: CTRL
  group2: U0126
```

(continues on next page)

(continued from previous page)

```
  name: CTRL_U0126
  pvalThreshold: '2'
```

Here, we've defined two separate group comparisons: THZ1 is compared to CTRL and U0126 is compared to CTRL. The `name` parameter is used to set the name of the data group into which differentially abundant phosphorylation sites will be placed. The foldChangeThreshold and pvalThreshold are used to set the fold change and p value cutoffs, respectively.

2.1.5 Processing Block

The processing block defines the quantitative data filtering and pre-processing steps that will be conducted prior to statistical analysis.

Filtering

The filtering block allows you to remove phosphorylation sites with too great a proportion of missing values and those phosphorylation sites that are assigned to undesired proteins such as decoys or contaminants.

An example of a filtering block is given below:

```
filtering:
  doFiltering: 'true'
  filterTerms:
  - REV_
  - CON_
  minValues: '2'
  minValuesIn: group
```

Here, we activate the filtering process by setting `doFiltering` to `'true'`. Setting this parameter to any other value will bypass filtering.

Values added to the `filterTerms` list will be used to remove phosphorylation sites mapping to proteins containing these terms anywhere in the test of their `upidColumn`. For example, here we remove phosphorylation sites that contain `REV_` or `CON_` in their protein identifier.

The `minValues` parameter allows you to specify the minimum number of non-zero values that must be present for a phosphorylation site to be included in subsequent analysis. The `minValuesIn` parameter restricts how these missing values can be distributed. Valid options are:

- **total** : The `minValues` setting must be reached by any combination of samples.
- **group** : The `minValues` setting must be reached within samples of a treatment group.

Imputation

The imputation block allows you to specify a strategy by which missing values that remain after filtering are replaced.

An example of an imputation block is given below:

```
imputation:
  doImputation: 'true'
  imputeCategory: group
  imputeType: median
```

Here, we activate imputation by setting `doImputation` to `'true'`. Setting this parameter to any other value will bypass imputation.

The `imputeCategory` values sets which valid (non-zero) values are used to calculate the new replacement values.

- **group** : The replacement value is calculated using the valid values for a phosphorylation site within the same treatment group as the missing value.
- **site** : The replacement value is calculated using the valid values for a phosphorylation site the phosphorylation site, i.e. regardless of group.

The `imputeCategory` values sets the mathematical function that will be used to calculate the replacement value.

- **min** : The minimum of valid values will be used.
- **median** : The median of the valid values will be used.
- **mean** : The mean of the valid values will be used

normalisation

Normalisation of quantification values can be used to correct for global differences in phosphorylation site abundances caused by small random errors in protein loading.

- **none** : No normalisation applied
- **median** : Samples normalised by median intensity
- **tic** : Samples normalised by total intensity of all quantified phosphorylation sites
- **quantile** : Samples normalised such that distributions of quantification values of quantification values are the same.

transform

The `transformation` parameter allows you to apply log2 transformation to quantitative data so that an approximate normal distribution is obtained. Valid options are:

- **none** : Transformation is bypassed
- **log2** : Data are log2 transformed.

An example of a completed and valid processing block is below:

```
processing:
  filtering:
    doFiltering: 'true'
    filterTerms:
    - REV_
    - CON_
    minValues: '2'
    minValuesIn: group
  imputation:
    doImputation: 'false'
    imputeCategory: group
    imputeType: median
  normalisation: median
  transform: log2
```

2.2 Example Parameter File

An example parameter data file can be downloaded here and the text is copied below:

```
columnAssignments:
  quantColumns:
  - CTRL
  - CTRL_1
  - CTRL_2
  - THZ1
  - THZ1_1
  - THZ1_2
  - U0126
  - U0126_1
  - U0126_2
  residueColumn: Residue
  siteColumn: Position
  upidColumn: ID
comparisons:
- foldChangeThreshold: '1'
  group1: THZ1
  group2: CTRL
  name: THZ1_CTRL
  pvalThreshold: '2'
- foldChangeThreshold: '1'
  group1: U0126
  group2: CTRL
  name: U0126_CTRL
  pvalThreshold: '2'
processing:
  filtering:
    doFiltering: 'true'
    filterTerms:
    - REV_
    - CON_
    minValues: '2'
    minValuesIn: group
  imputation:
    doImputation: 'true'
    imputeCategory: group
    imputeType: median
  normalisation: median
  transform: log2
sampleAlias:
  QUANT_CTRL: CTRL
  QUANT_CTRL_1: CTRL_1
  QUANT_CTRL_2: CTRL_2
  QUANT_THZ1: THZ1
  QUANT_THZ1_1: THZ1_1
  QUANT_THZ1_2: THZ1_2
  QUANT_U0126: U0126
  QUANT_U0126_1: U0126_1
  QUANT_U0126_2: U0126_2
```

(continues on next page)

(continued from previous page)

```
sampleMaps:
  sampleGroupMap:
    CTRL: CTRL
    CTRL_1: CTRL
    CTRL_2: CTRL
    THZ1: THZ1
    THZ1_1: THZ1
    THZ1_2: THZ1
    U0126: U0126
    U0126_1: U0126
    U0126_2: U0126
  sampleIndexMap:
    CTRL: 1
    CTRL_1: 2
    CTRL_2: 3
    THZ1: 4
    THZ1_1: 5
    THZ1_2: 6
    U0126: 7
    U0126_1: 8
    U0126_2: 9
```

CHAPTER

THREE

USAGE EXAMPLES

3.1 Creating and running a new analysis

The example below shows how to create a new phosphomatics experiment, upload data and parameter files and run the initial data processing routine.

```
import phosphomaitcs.phosphomaitcs as pa

# Instantiate phosphomatics with your API key
exp = pa.Phosphomatics( key = 'YOUR_API_KEY')

# start a new experiment
# - this retrieves a new datasetToken unique to this experiment
exp.startNewExperiment()

# upload your experimental data
exp.uploadExperimentalData('path_to_phospho_data.tsv')

# upload parameter file for your analysis
exp.uploadParameterSet('path_to_parameter_file.yaml')

# call initial data processing routine
exp.process()

# print your datasetToken
print(exp.getDataSetToken())
```

The dataset token can then be used to visualise the results in the phosphomatics web server [here](https://www.phosphomatics.com/)

3.2 Accessing a previous analysis

```
import phosphomaitcs.phosphomaitcs as pa

# Instantiate phosphomatics with your API key
exp = pa.Phosphomatics( key = 'YOUR_API_KEY')

# Set the dataset token for the previous analysis
exp.setDataSetToken('DATASET_TOKEN_FOR_PREV_ANALYSIS')
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search