Package Profile

Ania Macioszek

1 Introduction

Package Profile was created to deal with genome profiles, generated by for example ChIP-seq experiments. For now, it deals with .wig and .sgr formats. Package is independent from other BioPython libraries, so it can be used without the rest of BioPython project.

2 Usage

2.1 Reading from file

To import Profile library, type

```
from Bio import Profile
```

Now we will be able to create Profile object by reading some profile from file. In this tutorial we will use small sample files, that you will find in [biopython_directory]/Bio/Profile/. Currently supported profiles are .wig and .sgr (see section 3 for details about these formats). To load a file to Profile object, type:

```
my_profile = Profile.Profile("biopython/Bio/Profile/sample_profile.sgr")
```

or

```
my_profile = Profile.Profile("biopython/Bio/Profile/sample_profile.wig",
"wig")
```

Note that when you load .sgr file you don't need to provide information about used format, because .sgr is the default one. Wig files can have some additional data in the first line of the file, that reading function can load into Profile object. To suppress this, set readMetadata argument to False. To provide additional metadata use argument addMetadata. Metadata should be provided as dictionary. Example command might look like this:

```
my_profile = Profile.Profile("biopython/Bio/Profile/sample_profile.wig,
"wig", readMetadata = False,
```

```
addMetadata = {"description"="my awesome profile", date="27.07.1410"})
```

.sgr format doesn't contain any additional data, so when format is set to "sgr" argument readMetadata doesn't change behaviour of reading function. Of course you can provide additional data anyway. If readMetadata is True and metadata is provided with addMetadata argument they are both read; if some field appears in both of them, the one from addMetadata overwrites the one from file.

2.2 Writing to file

To write Profile object to file, type

```
my_profile.writeProfile("some_profile.sgr")
```

or

```
my_profile.writeProfile("some_profile.wig", "wig")
```

This will create file some_profile.sgr (or some_profile.wig) in the current directory in the format specified by the second argument. Note that .sgr format is default.

2.3 Profile object

Object Profile has following attributes:

- metadata - it's a dictionary storing all metadata about the Profile, read from the file or provided later with addMetadata function.
- resolution - in case of .wig files, resolution is read from the second line (it should look like this: "variableStep chrom=chr1 span=50, where span is resolution of the profile); in case of .sgr files, it's estimated from first coordinates.
- coordinates - dictionary storing list of coordinates for each chromosome, for which values are defined
- values - dictionary storing list of profile values, corresponding to coordinates written in coordinates attribute

addMetadata(dict) function takes as an argument dictionary with keys describing names of the fields in metadata and values corresponding to their values. Note that if key already exists in Profile's metadata it will be overwritten.

Sometimes you have couple of profiles you want to analyse, all from the same organism, yet with different names of chromosomes, what can make analysis bit more difficult. So, Profile provides function to change chromosome names. To see what are the names of the chromosomes in Profile, use function getChromosomes(). To change their names, use function setChromosomeNames(list_of_chromosomes).

Sample pipeline with Profile object might look like that:

```
>>> from Bio import Profile
>>> my_profile = Profile.Profile("sample_profile.wig","wig")
>>> my_profile.metadata
{'name': 'sample profile', 'description': 'some random sample profile'}
>>>my_profile.addMetadata({"replicate" : 1,
"name"="chip-seq dreb2a binding sites"})
>>> my_profile.metadata
{'name': 'chip-seq dreb2a binding sites',
'description': 'some random sample profile',
'replicate': 1}
>>> my_profile.getChromosomes()
['Chr1', 'Chr2', 'Chr3', 'Chr4', 'Chr5', 'Chrc', 'Chrm']
>>> new_chromosomes = ["chrI", "chrII", "chrIII", "chrIV", "chrV", "chrC", "chrM"]
>>> my_profile.setChromosomeNames(new_chromosomes)
>>> my_profile.writeProfile("changed_chr_names_and_description.sgr")
```

3 Supported formats

Currently supported formats are .sgr and .wig. .sgr format is assumed to look like this:

```
chr1 1 1.12
chr1 11 3.432
chr1 21 5.09
...
chr2 2 0
chr2 12 0.91
```

that is, it should have three column: in the first one we have chromosome name, in the second position on the chromosome and value in the third. .wig file should look like this:

```
track name="foo" description="some profile"
variableStep chrom=chr1 span=50
3 12.45
53 32.454
103 41.11
...
track
variableStep chrom=chr2 span=50
1 4.19
51 3.78
```

The first line should start with "track", followed by optional metadata. Metadata could be also written like this:

```
track name=foo description=some_profile
```

But if you want to have spaces in your metadata you should use quotation marks (important note: don't mix those two methods). The second line should start with "variableStep" (fixedStep is not yet supported), then "chrom=" followed by chromosome name (no spaces), then optionally span (currently not read). Part describing profile itself should consist from two columns: first is position, second - value. After each chromosome two first lines should be written again (however, the first line could be only "track", the metadata isn't read again). Separator between columns or fields of metadata could be either tab or space.