PyMAP Documentation

Release

PyMap Authors

Mar 09, 2017

Contents

PyMAP is a python module that allows large-scale methylation analysis using Illumina 450k platform. The following documentation are provided for this module.

Contents:

CHAPTER 1

Introduction

Why pyMAP?

This is a very interesting question that deserves to be answered properly. All 450K platform analysis software that have been developed so far are made using R scripts. While we appreciate R as a powerful statistical platform, we also believe in diversity of different analysis platforms. Python, which is a simple yet powerful language, can also be used for these analysis. This language is increasingly being used by scientists to address computational aspect of their works. pyMAP, implemented in native python, offers a more powerful alternative which utilizes python scripts that can be deployed easily and can also be modified to suit the study.

The purpose of this package is to make analysis and visualization of methylation data easier for ordinary scientists.

Documentation

Here, we have provided detailed documentations of Classes and Functions of PyMAP module. Documentations are mostly based on a single dataset that are used throughout the examples for consistency in data structures. The dataset is publicly available and can be downloaded from http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE42308 .

Examples

Here goes examples of what PyMAP can do!

In the first example, we will create an annotation object that you can use to extract methylation values form samples. This object will hold Illumina probe information.

```
# Import Annotation submodule to parse and prepare probe information.
import pymap.Annotation

# Create Annotation object.
# This object well parse through all probes annotation information Illumina has␣
↪provided for probes used in 450K platform.
```

```
annotation = pymap.Annotation.Annotator()

# Get number of probes in the annotation object.
probe_number = annotation.get_number()
print(probe_number)

# Remove known SNPs easily with a simple method.
# This filteration step might be useful for most studies in human subjects.
annotation.remove_snp_probes()

# Get number of probes after SNP removal from the annotation object.
probe_number = annotation.get_number()
print(probe_number)
```

In the second example, we parse samples:

```
# Import Core submodule to parse data.
import pymap.Core

# Parse a single data file.
parsed_samples = Core.ParseFile("data.txt")

# Parse multiple data file.
parsed_samples = Core.ParseBatch("/Data")

# Get Sample data from either one or multiple files. samples contains a list of␣
↪samples.
# Please see sample object documentations.
samples = parsed_samples.get_samples()
```

In the third example, we integrate Core and Annotation objects to extract methylation data from p53 gene probes:

```
# Get all probes associated with p53 gene:
probe_list = annotations.get_probes_from_gene("TP53")

# Export data associated with selected probes and samples into a data frame:
pymap.Core.write_data("data.txt", samples, probe_list)

# Export probe and methylation data into BED file format.
# In this example we export data for the first sample (sample[0]).
pymap.Core.probes_to_bed("Export/test2.bed", probe_list, samples[0])

# Export all samples into separate BED file.
pymap.Core.samples_to_bed("Export/me", probe_list, samples)
```

In the fourth example, we extract features from features.

```
# Get probes from get_all_probe_id function of annotation object.
probes = annotations.get_probes(annotations.get_all_probe_ids())

# Get probes that are positioned in the island.
probe_list = Annotation.get_probes_from_feature(probes,
Annotation.Feature(Annotation.CpG_location.ISLAND))

# Get probes that are within 200 bp of TSS.
probe_list = Annotation.get_probes_from_feature(probes,
Annotation.Feature(Annotation.Location.TSS200)

```

```
# Get probes that are positioned close to BRCA1.
probe_list = Annotation.get_probes_from_feature(probes, "BRCA1"))
```

In the following example we generate a heatmap of probes that are associated with BRCA1 gene.

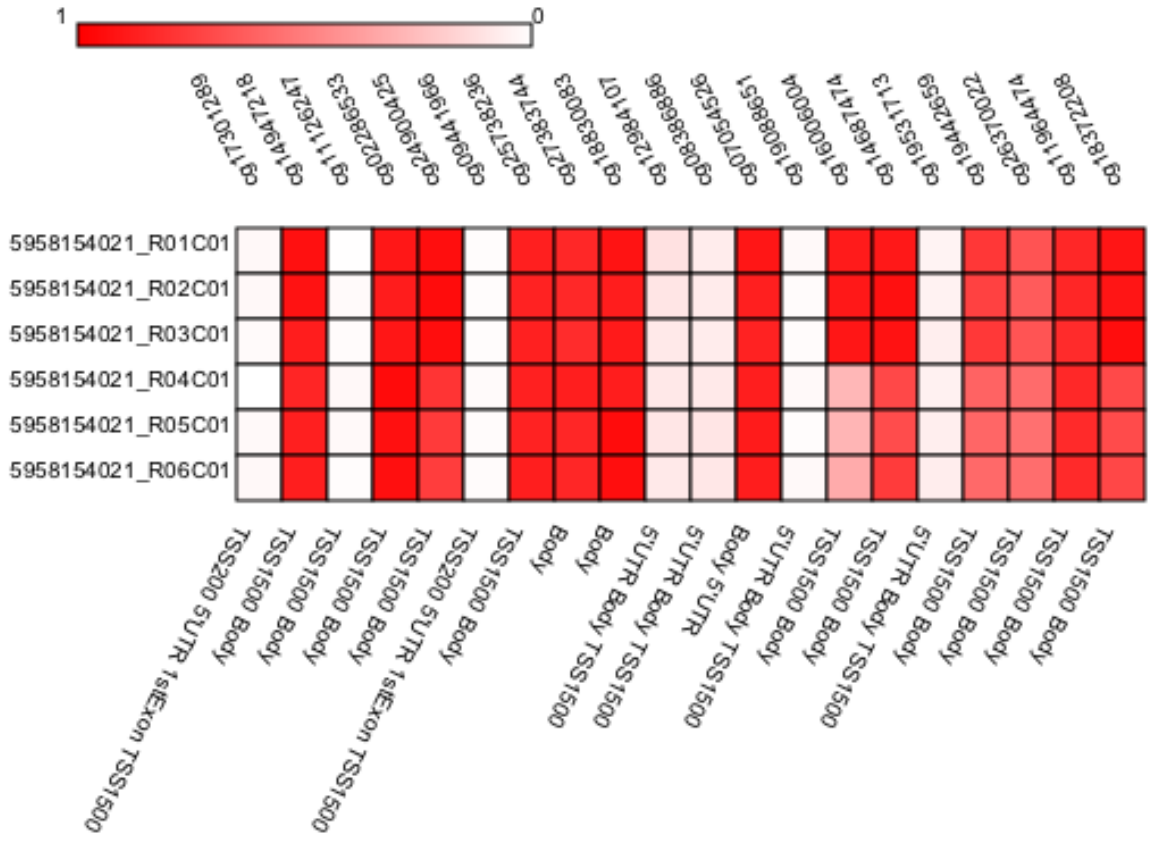
```
# Get probes that are positioned close to BRCA1.
probe_list = Annotation.get_probes_from_feature(probes, "BRCA1"))

# Sort probes based on genomic coordinate.
probe_list = annotations.sort_coord_probe(probe_list)

# Import plot package.
import pymap.Plot

# Generate heatmap
Plot.Heatmap(samples, probe_list, "BRCA1_probes.png")
```

Output heatmap image



The following example will generate BoxPlot figure of probes that are associated with BRCA1.

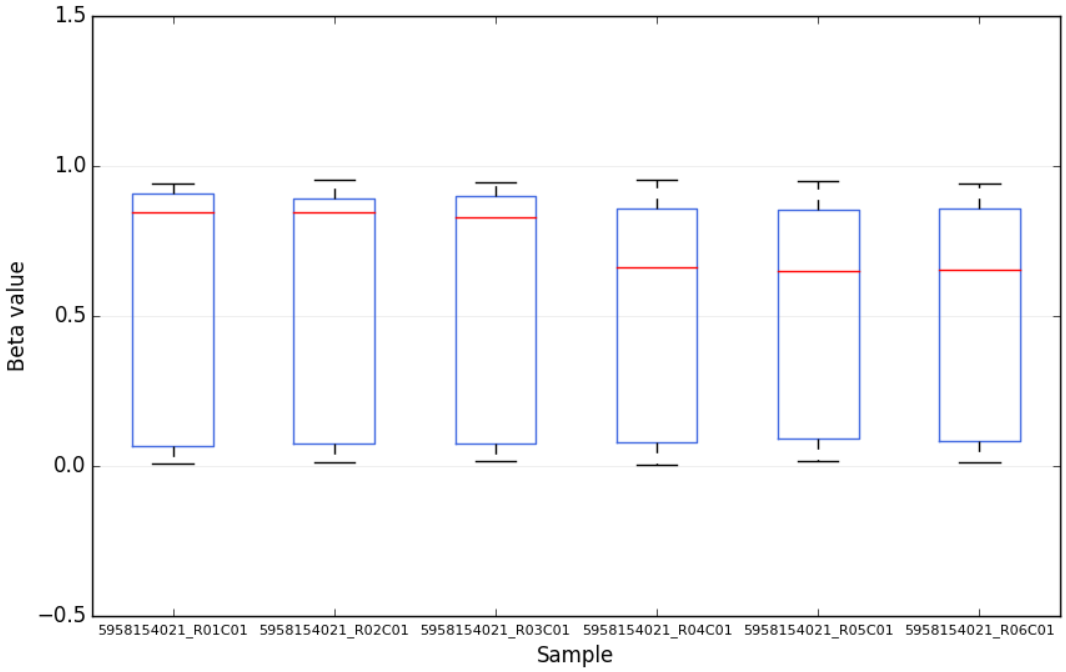
```
# import the plotting module.
import pymap.Plot

# Get probes that are positioned close to BRCA1.
```

```
probe_list = Annotation.get_probes_from_feature(probes, "BRCA1"))

# Plot the probe methylation values.
Plot.BoxPlot(probe_list, samples)
```

Output BoxPlot image



Citation and Contact

Please cite this package with the Github URL until it is published. Please contact authors using Gitub platform for any inquiry.

CHAPTER 2

Annotation module

class Annotation.**Annotator**

This class parse all information about Illumina probes.

USAGE:

```
annotations = Annotation.Annotator()
```

get_all_probe_ids()
Get all probe ids.

Returns a list of probe ids.

get_all_probes()
Get a list of all probes.

Returns A list of all probes in the Annotation object.

get_coord(*probe*)
Get genomic coordinate of a probe.

Parameters **probe** – A probe object

Returns An integer indicator of probe numbers

get_keys(*dic_keys*)
Get Probe id from probe dictionaries

Parameters **dic_keys** – Probe dict.

Returns returns a list of probe id.

get_number()
Get numbers of probes

Returns An integer representing the number of probes.

get_probe(*probe_id*)
This function returns the info associated with an id.

Parameters **probe_id** – ILLUMINA ID

Returns all info

get_probes(*list_of_ids*)
This function returns a list of probe object from a list of ids.

Parameters **list_of_ids** –

Returns A list of probe objects.

get_probes_from_chr_loc(*chr_loc*)
Get a list of probes that are within a genomic region

Parameters **chr_loc** – Genomic location interval

Returns A list of probes.

get_probes_from_cpg(*cpg_loc*)
Get a list probe objects from cpg location.

Parameters **cpg_loc** – from CpG object

Returns probes.

get_probes_from_gene(*gene_name*)
Get a list probe objects from an associated gene name.

Parameters **gene_name** – Gene name in string format

Returns A probe.

get_probes_from_loc(*loc*)
Get a list probe objects from genomic location.

Parameters **loc** – from Location object.

Returns probes.

get_probes_id_from_chr_loc(*chr_loc*)
Get a list of probe ids that are witihn a genomic region

Parameters **chr_loc** – Genomic location interval

Returns A list of probe ids.

get_probes_id_from_cpg(*cpg_loc*)
Get all probes ids associated with CpG sites.

Parameters **cpg_loc** –

Returns a list of probe ids.

get_probes_id_from_gene(*gene_name*)
Get all probes ids associated with a gene.

Parameters **gene_name** –

Returns a lst of probe ids.

get_probes_id_from_loc(*probe_loc*)
Get all probes ids associated with genomic locations.

Parameters **probe_loc** –

Returns a list of probe ids.

get_probes_id_from_probe(*probe_list*)
Get all probes ids from a list of probe objects.

Parameters **probe_list** – A list of probe ids.

Returns a list of probe ids.

remove_snp_probes()
This function will removes all SNPs associated with probes.

Returns returns a new probe listing.

sort_coord_probe(*probes*)
This function sorts probes based on the probe genomic location. Best used in combination with plotting module.

Parameters **probes** – Input probe list.

Returns A sorted probe list.

class Annotation.**ChrLoc**(*chromosome*, *start*, *end*)

defines a chromosomal interval.

USAGE:

```
my_probe = Annotation.ChrLoc("X", 122333232, 123334444)
```

class Annotation.**CpG_location**
CpG location is defined here.

ISLAND = 'Island'

NSHELF = 'N_Shelf'

NSHORE = 'N_Shore'

SSHELF = 'S_Shelf'

SSHORE = 'S_Shore'

class Annotation.**Feature**(*feature*)
This class parse features associated with probes.

class Annotation.**Location**
Probe location is defined here.

BODY = 'Body'

EXON = 'Exon'

TSS1500 = 'TSS1500'

TSS200 = 'TSS200'

UTR3 = "3'UTR"

UTR5 = "5'UTR"

class Annotation.**Probe**

This class holds probe info.

USAGE:

```
my_probe = Annotation.Probe()
```

class Annotation.**SNP**
This class defines the SNPs in probes. Can be used to filter probes.

Annotation.**get_probes**(*annotations*, *probes_ids*)
Get a list of probes from probe ids

Parameters

- **annotations** – Annotation object that has been initiated properly.
- **probes_ids** – A list of probe ids.

Returns A list of probes.

Annotation.**get_probes_from_feature**(*probes_ids*, *filter_val*)
This function returns filters probes based on a feature.

Parameters

- **probes_ids** – The probe ids that you would like to filter.
- **filter_val** – Feature to be filtered.

Returns Returns a list of probes.

CHAPTER 3

Core Module

Core.Stats module

class Core.Stats.**Dist**(*beta_list*, *sample_id*, *lab*)

class Core.Stats.**TopList**(*samples1*, *samples2*, *all_probes*)

get()

ttest(*samples1*, *samples2*, *probe*)

class Core.Stats.**Transform**

class Core.Stats.**Ttest**(*sample1*, *sample2*, *probe*, *diff_id_list*)
Bases: threading.Thread

run()

Module contents

class Core.**Group**(*list_of_samples*)
Create a new Group groups.

Parameters **list_of_samples** – A list of sample objects.

Returns A Group object.

class Core.**ParseBatch**(*folder*, *delim='t'*, *avg_beta_header='.AVG_Beta'*)
Parse a series of data in a folder.

Parameters

- **folder** – A string that represent a folder.
- **delim** – delimitation character used in the data file [default = tab].

- **avg_beta_header** – A string that represents average beta values [default = .AVG_Beta].

Returns A ParseBatch object. Use get_samples() function to retrieve sample information.

get_samples()
Return all sample objects created from all files

class Core.**ParseConfig**

class Core.**ParseFile**(*filename*, *delim='t'*, *avg_beta_header='.AVG_Beta'*)
Parse a single file. The file could still have multiple groups. This module automatically finds and parses them.

Parameters

- **filename** – A string that represent a data file name.
- **delim** – delimitation character used in the data file [default = tab].
- **avg_beta_header** – A string that represents average beta values [default = .AVG_Beta].

Returns A ParseFile object. Use get_samples() function to retrieve sample information.

check_file(*filename*)
Check input filename

Parameters **filename** – A string that represents a data file.

Returns A boolean value.

get_samples()
Returns all groups in this file.

class Core.**Sample**(*name=None*, *probes=None*)
Sample data object. Each *sample* has a name which is a string type and Probe methylation data which is a dictionary type.

Parameters

- **name** – Name of the sample.
- **probes** – methylation data of the sample - in dict type.

Returns A Sample object.

Core.**get_all_beta**(*sample*)
Get all beta values.

Returns A list of beta

Core.**get_all_sample_name**(*samples*)
Get all sample name.

Returns A list that contain sample names.

Core.**get_genes_from_probes**(*probe_list*)
Get gene names and number of probes associated with each gene.

Parameters **probe_list** – A list of probes.

Returns A dictionary of genes names and probes numbers.

Core.**get_id_beta**(*sample*)
Get all beta values.

Returns return beta values of a sample.

Core.**get_probe_avg**(*probe_id*, *samples*, *verbose=False*)
Get Probe AVG values.

Parameters **probe_id** – A list of probe ids.

Returns A list of avg beta values.

Core.**get_probes_avg**(*probe_id_list*, *sample*)
Get probe AVG beta values from a list of probes for all groups

Parameters **probe_id_list** – A list of probe ids.

Returns A list of beta values.

Core.**get_sample_by_name**(*samples*, *sample_name*)
Returns a sample by name.

Parameters **sample_name** – Sample name, a string.

Returns Return a sample object.

Core.**get_sample_by_no**(*samples*, *sample_no*)
Returns a sample by number [zero based].

Parameters **sample_no** – Sample number, a zero based integer.

Returns Return a sample object.

Core.**probes_to_bed**(*filename*, *probes*, *sample*)
Writes a BED file containing the probe beta info.

Parameters

- **filename** – A filename to be stored.
- **probes** – A list of Probe info.
- **sample_no** – The sample number to include in the BED file.

Returns Static function - stores a file.

Core.**samples_to_bed**(*base_filename*, *probes*, *samples*)
Return a BED file representative of all groups for the provided probes.

Parameters

- **base_filename** – A base name for output file
- **probes** – A list of probes objects.
- **samples** – A list of groups to extract data.

Returns Static function - stores a file.

Core.**write_data**(*file_name*, *samples*, *probes*)
Export data to data table

Parameters

- **samples** – A list of groups.
- **probes** – A list of probes.

Returns Writes a data file.

CHAPTER 4

Plot module

Plot contents

class Plot.**BoxPlot**(*probe_list*, *samples*, *filename='boxplot'*, *imgtype='png'*)
Box plot class creates a BoxPlot figure from probes and groups.

Parameters

- **probe_list** – A list of probes.
- **samples** – A list of groups.
- **filename** – output filename.
- **imgtype** – output image format.

Returns writes an image onto disk.

class Plot.**BoxPlotGroups**(*probe_list*, *group1*, *group2*, *filename='boxplot'*, *imgtype='png'*)
Box plot class creates a BoxPlot figure from Grouped groups.

Parameters

- **probe_list** – A list of probes.
- **groups** – A list of groups.
- **filename** – output filename.
- **imgtype** – output image format.

Returns writes an image onto disk.

class Plot.**Heatmap**(*samples*, *probes*, *file_name*, *properties=<Plot.Properties instance>*)
This class creates a heatmap object

Parameters

- **probes** – A list of probes.
- **samples** – A list of groups.

- **file_name** – output filename.
- **properties** – An instance of Properties class *[optional]*.

Returns writes a PNG image onto disk.

static block(*ctx*, *x*, *y*, *size*, *intensity*, *nan=False*)
Create a single block. Used by Heatmap class.

Parameters

- **ctx** – cairo context
- **x** – x-coordinate
- **y** – y-coordinate
- **size** – block size
- **intensity** – color intensity
- **nan** – null value

Returns Draws a block

class Plot.**Properties**(*size=20*, *xoff=20*, *yoff=100*)
Defines the style of Heatmap plots

size defines the block size. *xoff* defines x offset. *yoff* defines y offset.

CHAPTER 5

PlotProbes Executable

About

PlotProbe.py generates heatmap plot of probes that are associated with a gene.

Usage

./PlotProbes.py -gene TP53 -out out.png -data Data/methyl_data.txt

CHAPTER 6

convertbed Executable

About

convertbed.py creates a BED file that contains all information about probes that are associated with a specified gene.

Usage

./convertbed.py -file Data/METHYL_DATA.txt -out TP53.bed -gene TP53

CHAPTER 7

getidfromgene Executable

About

getidfromgene.py generates a text file that contains probe ids associated with a gene

Usage

./getidfromgene.py -gene TP53 -out y

CHAPTER 8

samplebed module

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index

P

R

S

T

U

W