methylize Documentation

Release 1.1.1

FOXO Technologies, inc.

Jul 08, 2022

Contents:

CHAPTER 1

Getting Started

`methylize` is a python package for analyzing output from Illumina methylation arrays. It complements `methylprep` and `methylcheck` and provides methods for computing differentially methylated probes and regions, and annotating these regions with the UCSC Genome Browser. View on ReadTheDocs.

1.1 methylize is part of the methylsuite

`methylize` is part of the methylsuite of python packages that provide functions to analyze DNA methylation data from Illumina's Infinium arrays (27k, 450k, and EPIC, as well as mouse arrays). This package is focused on analysis of processed methylation data, such as EWAS using Manhattan and Volcano plots. `methylize` functions are designed to work with a minimum of knowledge and specification required. But you can always override the "smart" defaults with custom settings if the default settings don't work for your data. The entire `methylsuite` is designed in this format: to offer ease of use while still maintaining flexibility for customization as needed.

1.2 Methylsuite package components

You should install all three components, as they work together. The parts include:

- `methylprep`: for processing `idat` files or downloading GEO datasets from NIH. Processing steps include
  - infer type-I channel switch
  - NOOB (normal-exponential convolution on out-of-band probe data)
  - poobah (p-value with out-of-band array hybridization, for filtering low signal-to-noise probes)
  - qualityMask (to exclude historically less reliable probes)
  - nonlinear dye bias correction (AKA signal quantile normalization between red/green channels across a sample)
  - calculate beta-value, m-value, or copy-number matrix
  - large batch memory management, by splitting it up into smaller batches during processing
- `methylcheck`: (this package) for quality control (QC) and analysis, including
  - functions for filtering out unreliable probes, based on the published literature
    - Note that `methylprep process` will exclude a set of unreliable probes by default. You can disable that using the –no_quality_mask option from CLI.
  - sample outlier detection
  - array level QC plots, based on Genome Studio functions
  - a python clone of Illumina's Bead Array Controls Reporter software (QC)
  - data visualization functions based on `seaborn` and `matplotlib` graphic libraries.
  - predict sex of human samples from probes
  - interactive method for assigning samples to groups, based on array data, in a Jupyter notebook
- `methylize` provides more analysis and interpretation functions
  - differentially methylated probe statistics (between treatment and control samples)
  - volcano plots (which probes are the most different?)
  - manhattan plots (where in genome are the differences?)

1.3 Table of Contents

1.4 Installation

```
pip3 install methylize
```

Installation will also install the other parts of the `methylsuite` (methylprep and methylcheck) if they are not already installed.

If progress bar is missing: If you don't see a progress bar in your jupyterlab notebook, try this:

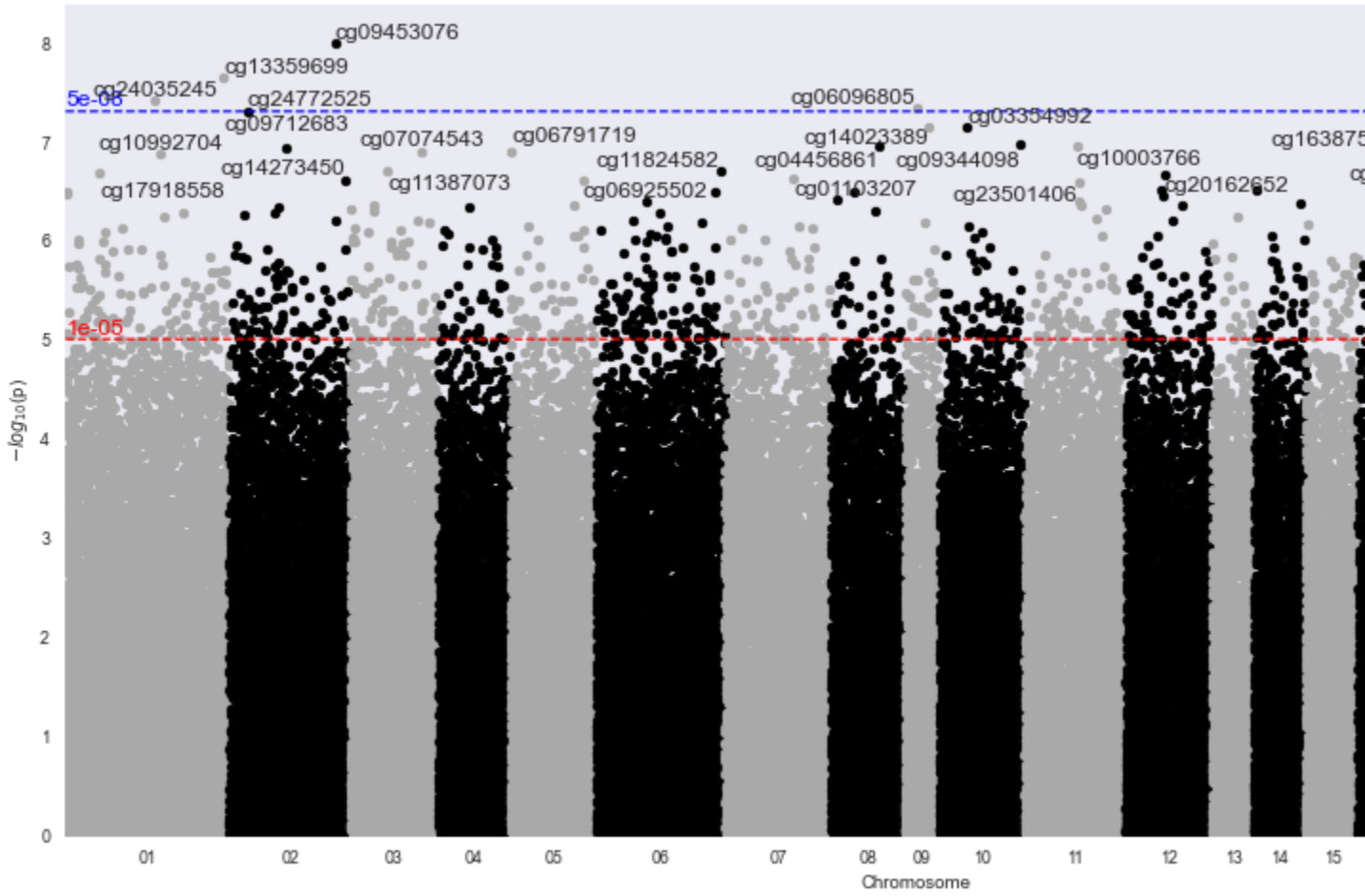
```
- conda install -c conda-forge nodejs
- jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

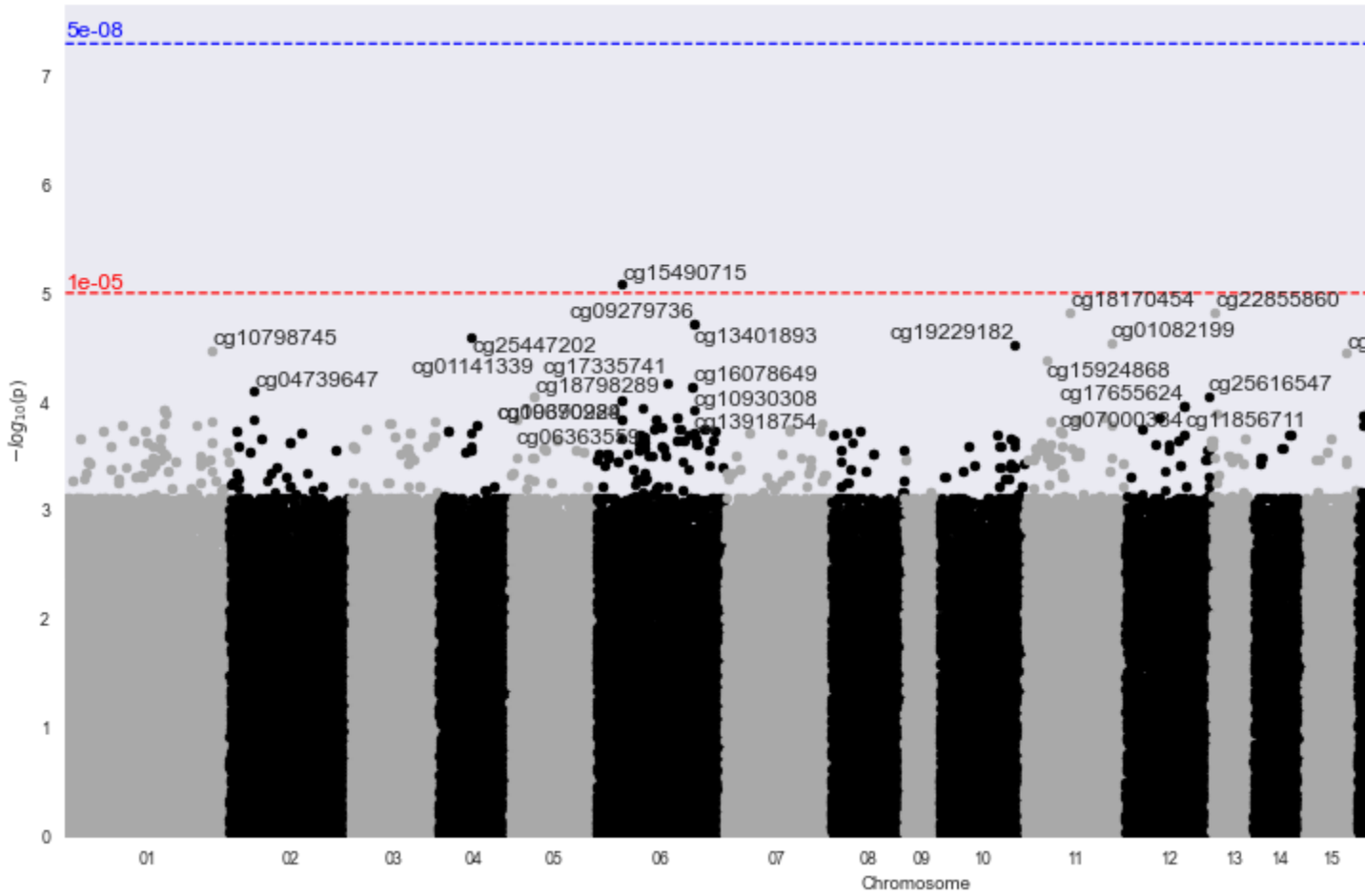
1.5 Methylize Package

The `methylize` package contains both high-level APIs for processing data from local files and low-level functionality allowing you to analyze your data AFTER running `methylprep` and `methylcheck`. For greatest usability, import `methylize` into a Jupyer Notebook along with your processed sample data (a DataFrame of beta values or m-values and a separate DataFrame containing meta data about the samples).

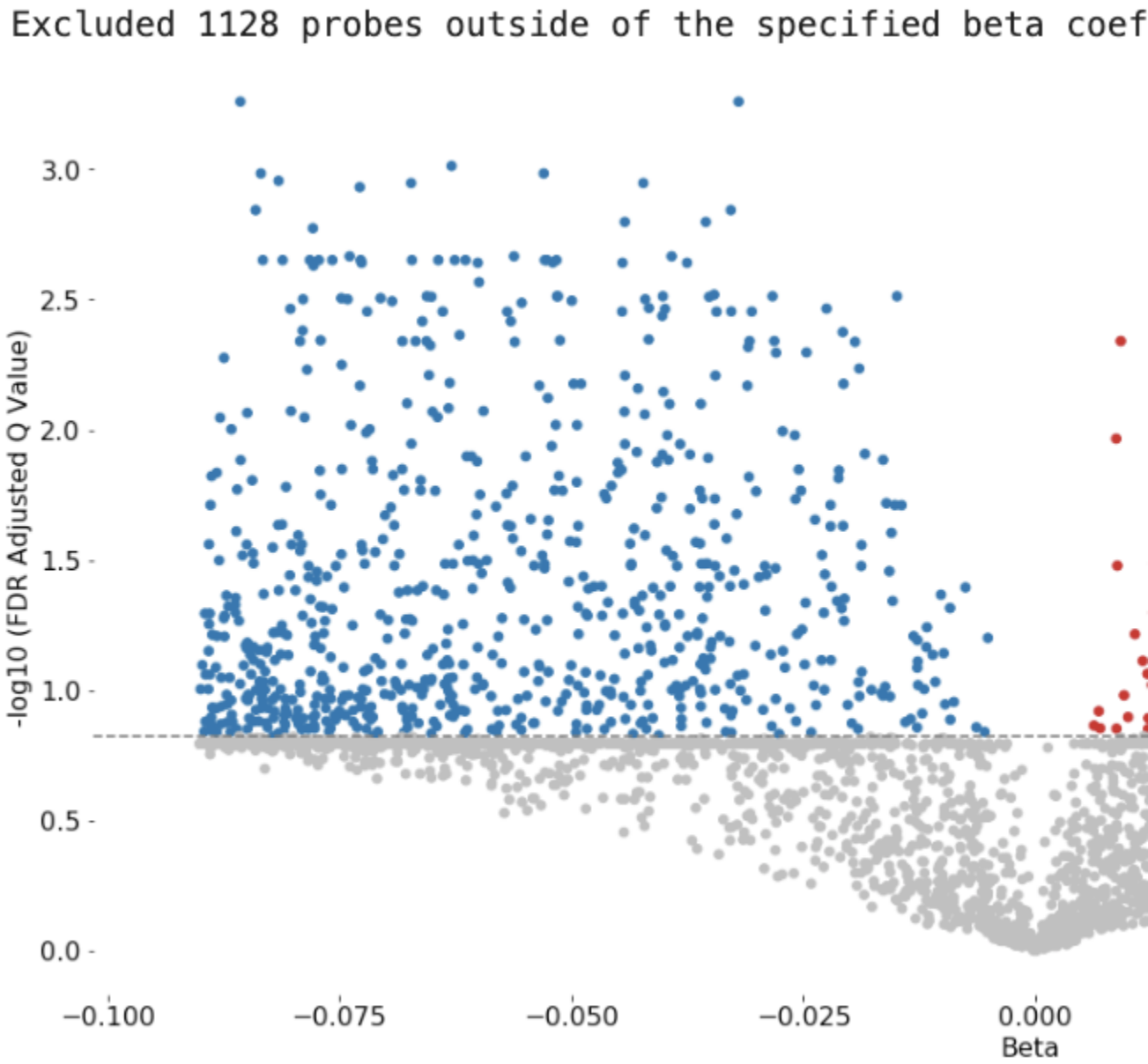
`Methylize` allows you to run linear or logistic regression on all probes and identify points of interest in the methylome where DNA is differentially modified. Then you can use these regression results to create *volcano plots* and *manhattan plots*.

1.5.1 Sample Manhattan Plot




5e-08
1e-05
cg15490715
cg09279736
cg13401893
cg10798745
cg25447202
cg01141339
cg17335741
cg04739647
cg18798289
cg16078649
cg10930308
cg13918754
cg06363559
cg19229182
cg18170454
cg22855860
cg01082199
cg15924868
cg17655624
cg25616547
cg11856711
−log10(p)
Chromosome

1.5.2 Sample Volcano Plot



Customizable: Plot size, color palette, and cutoff p-value lines can be set by the user. Exporting: You can export all probe statistics, or just the significant probes as CSV or python pickled DataFrame.

1.6 Differentially methylated position/probe (DMP) detection

The `diff_meth_pos(meth_data, phenotypes)` function searches for individual differentially methylated positions/probes (DMPs) by regressing methylation `beta values` or `M-values` for each sample at a given genomic location against the phenotype data for those samples.

1.6.1 Phenotypes

Can be provided as

```
- a list of strings,
- integer binary data,
- numeric continuous data
- pandas Series, DataFrame or numpy array
```

Only 2 phenotypes are allowed with logistic regression. Use Linear regression with numeric (measurement) phenotypes like age or time. You can pass in the GEO meta DataFrame associated with a dataset along with `column=<columnname>` kwarg. The order of the items in the phenotype should match the order of samples in the beta values or M-values.

Covariates are also supported for logistic (but not linear) regression. Pass in `covariates=True` to treat all other columns in a phenotype DataFrame as covariates, or pass in a list of column names to specify specific parts of the DataFrame. Note that supplying too many covariates for small sample sizes will lead to most probes failing with Linear Algebra or Perfect Separation errors.

For details on all of the other adjustable input parameters, refer to the API for diff_meth_pos()

1.6.2 Returns

A pandas dataframe of regression `statistics` with one row for each probe and these columns:

```
- `Coefficient`: regression coefficient
- `StandardError`: standard error
- `95%CI_lower`: lower limit of the coefficient's 95% confidence interval
- `95%CI_upper`: upper limit of the coefficient's 95% confidence interval
- `PValue`: p-value: phenotype group A vs B - likelihood that the difference is␣
↪significant for this probe/location
- `Rsquared`: proportion (0 to 1) of probe variance explained by your phenotype.␣
↪Linear Regression Only.
- `FDR_QValue`: p-values corrected for multiple comparisons using the Benjamini-
↪Hochberg FDR method. The False Discovery Rate (FDR) corrected p-values will remain␣
↪comparable, regardless of the number of additional comparisons (probes) you include.

If a `q_cutoff` is specified, only probes with significant q-values less than the␣
↪cutoff will be returned in the DataFrame.
```

1.7 Differentially methylated regions (DMR)

Pass in your `diff_meth_pos` (DMP) stats results DataFrame as input, and it will calculate and annotate differentially methylated regions (DMR) using the `combined-pvalues` pipeline. This function returns list of output files.

```
- calculates auto-correlation
- combines adjacent p-values
- performs false discovery rate (FDR) adjustment
- finds regions of enrichment (i.e. series of adjacent low p-values)
- assigns significance to those regions
- annotates significant regions with possibly relevant nearby Genes,
  using the UCSC Genome Browser Database
- annotates candidate genes with expression levels for the sample tissue type,
```

(continues on next page)

(continued from previous page)

```
  if user specifies the sample tissue type.
- returns everything in a CSV that can be imported into other Genomic analysis␣
↪packages.
```

For more details on customizing the inputs and outputs, see API for the diff_meth_regions(stats, array_type) function.

1.8 Loading processed data

Assuming you previously used `methylprep` to process a data set like this:

```
python -m methylprep -v process -d GSE130030 --betas
```

This creates two files, `beta_values.pkl` and `sample_sheet_meta_data.pkl`. You can load both in `methylize` like this:

Navigate to the folder where `methylrep` saved its processed files, and start a python interpreter:

```
>>>import methylcheck
>>>data, meta = methylcheck.load_both()
INFO:methylize.helpers:loaded data (485512, 14) from 1 pickled files (0.159s)
INFO:methylize.helpers:meta.Sample_IDs match data.index (OK)
```

Or if you are running in a notebook, specify the path:

```
import methylcheck
data, meta = methylcheck.load_both('<path_to...>/GSE105018')
```

This also validates both files, and ensures that the `Sample_ID` column in meta DataFrame aligns with the column names in the `data DataFrame`.

1.8.1 Differentially methylated regions (DMR) in methylize

Adopted from the ''combined-pvalues'' package by Brend Pedersen et al, 2013: Comb-p: software for combining, analyzing, grouping and correcting spatially correlated P-values doi: 10.1093/bioinformatics/bts545

What are Differentially methylated regions (DMRs)?

Genomic regions where DNA methylation levels differ between two groups of samples. DNA methylation is associated with cell differentiation, regulation, and proliferation, so these regions indicate that nearby genes may be involved in transcription regulation. There are several different types of DMRs. These include:

- tissue-specific DMR (tDMR),
- cancer-specific DMR (cDMR),
- development stages (dDMRs),
- reprogramming-specific DMR (rDMR),
- allele-specific DMR (AMR),
- aging-specific DMR (aDMR).

How to run DMR

First, assuming you have processed data using `methylprep`, use `methylize` to convert a dataframe of beta or M-values into differentially-methylated-probe (DMP) statistics, using `methylize.diff_meth_pos`. You will need to provide the data along with a list of sample labels for how to separate the data into two (treatment / control) groups or more groups (levels of a phenotypic characteristic, such as age or BMI):

```
meth_data = pd.read_pickle('beta_values.pkl')
phenotypes = ["fetal","fetal","fetal","adult","adult","adult"]
test_results = methylize.diff_meth_pos(meth_data, phenotypes)
```

`phenotypes` can be a list, numpy array, or pandas Series (Anything list-like). The results will be a dataframe with p-values, a measure of the the likelihood that each probe is significantly different between the phenotypic groups. The lower the p-value, the more likely it is that groups differ:

```
IlmnID                        Coefficient  StandardError        PValue  95%CI_lower  95
↪%CI_upper  Rsquared  FDR_QValue
cg04680738_II_R_C_rep1_EPIC     -0.03175       0.001627  1.171409e-06    -0.992267  ␣
↪-0.992168  0.984496    0.009706
cg18340948_II_R_C_rep1_EPIC      0.10475       0.005297  1.084911e-06     0.992256  ␣
↪ 0.992570  0.984887    0.009706
cg03681905_II_F_C_rep1_EPIC     -0.04850       0.002141  4.843320e-07    -0.994254  ␣
↪-0.994157  0.988444    0.009706
cg01815889_II_F_C_rep1_EPIC     -0.05075       0.002735  1.579625e-06    -0.991492  ␣
↪-0.991308  0.982875    0.009816
cg05995891_II_R_C_rep1_EPIC     -0.04050       0.002407  2.812417e-06    -0.989670  ␣
↪-0.989474  0.979254    0.013981
...                                  ...            ...           ...          ...  ␣
↪       ...       ...          ...
cg05855048_II_R_C_rep1_EPIC      0.00025       0.016362  9.883052e-01    -0.025827  ␣
↪ 0.038289  0.000039    0.999077
cg23130711_II_R_C_rep1_EPIC      0.00025       0.016530  9.884235e-01    -0.026217  ␣
↪ 0.038553  0.000038    0.999156
cg10163088_II_F_C_rep1_EPIC     -0.00025       0.017168  9.888530e-01    -0.039573  ␣
↪ 0.027696  0.000035    0.999550
cg04079257_II_F_C_rep1_EPIC     -0.00025       0.017430  9.890214e-01    -0.039997  ␣
↪ 0.028300  0.000034    0.999679
cg24902557_II_F_C_rep1_EPIC      0.00025       0.017533  9.890857e-01    -0.028535  ␣
↪ 0.040163  0.000034    0.999704
```

The FDR_QValue is the key result. FDR_Q is the "False Discovery Rate Q-value": The adjustment corrects individual probe p-values for the number of repeated tests (once for each probe on the array).

Next, you take this whole dataframe output from `diff_meth_pos` and feed it into the DMR function, `diff_meth_regions`, along with the type of methylation array you are using:

```
manifest_or_array_type = '450k'
files_created = methylize.diff_meth_regions(stats_results, manifest_or_array_type,␣
↪prefix='docs/example_data/450k_test/g69')
```

This will run a while. It compares all of the probes and clusters CpG probes that show a difference together if they are close to each other in the genome sequence. There are a lot of adjustable parameters you can play with in this function. Refer to the docs for more details.

When it completes, it returns a list of files that have been saved to disk:

```
docs/research_notebooks/dmr.acf.txt
docs/research_notebooks/dmr.args.txt
```

(continues on next page)

(continued from previous page)

```
docs/research_notebooks/dmr.fdr.bed.gz
docs/research_notebooks/dmr.manhattan.png
docs/research_notebooks/dmr.regions-p.bed.gz
docs/research_notebooks/dmr.slk.bed.gz
docs/research_notebooks/dmr_regions.csv
docs/research_notebooks/dmr_regions_genes.csv
docs/research_notebooks/dmr_stats.csv
docs/research_notebooks/stats.bed
```

Most of these are intermediate processing files that might be useful when imported into other tools, but the main summary file is the one that ends in `regions_genes.csv`:

```
chrom  chromStart chromEnd         min_p  ...  z_sidak_p          name      genes␣
↪distances   descriptions

21     2535657    2535711  6.662000e-12  ...  3.874000e-08  cg06415891  VLDLR-AS1
↪    3   Homo sapiens VLDLR antisense RNA 1 (VLDLR-AS1)...
22     2535842    2535892  2.816000e-04  ...  8.913000e-01  cg12443001      HCG22
↪   16   Homo sapiens HLA complex group 22 (HCG22), ...
25     2577833    2577883  2.048000e-11  ...  1.347000e-07  cg22948959      HLA-C
↪   33   Homo sapiens major histocompatibility compl...
1       876868     876918       0.00303       1.0           cg05475702
1      1514374    1514424      0.003309       1.0           cg00088251
```

This will reveal clusters of CpG probes that were significantly different and annotate these clusters with one or more nearby genes using the UCSC Genome Browser database. Note the CSV file column headings: `genes`, `distances`, `descriptions`. In some cases, a single diff-meth-region can align with multiple genes. In that case you will be a comma separated list for these fields. The descriptions will be separated by a pipe "|" symbol. The distances are the number of base-pairs of separation between the start of the CpG probe and the gene coding start sequence. Only rows with significant region p-values will have annotation.

If you pass in the `tissue=<your tissue type>` argument into the `diff_meth_regions` function, and that tissue type is one of the 54 that are part of the GTEx (genome expression levels in humans by tissue dataset), this file will also include a column showing the expression levels for any genes that match, so that you can further narrow down the search for relevant genomic interactions within each experiment.

There are a lot of additional corrections that researchers make at this stage, and many of them are beyond the scope of `methylsuite`, but this function should get you started. And you can export the output from this function into more sophisticated analysis tools.

Gene annotation with UCSC Genome Browser

University of California Santa Cruz maintains a large database of every version of the human genome and its meta data at https://genome.ucsc.edu/cgi-bin/hgTables. You can browse these database tables.

If you are using the latest genome build (hg38), diff_meth_regions will annotate your database using the `refGene` table. It also (partially) supports the `knownGene` and `ncbiRefSeq` tables, if you want to use those. This is useful for identifying genes that are nearby your regions of interest, and noting the tissue specificity of those genes, in exploring your data.

1.8.2 Methylize Walkthrough

[1]:
```
#Install joblib module for parallelization
import sys
!conda install --yes --prefix {sys.prefix} joblib
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.
```

[2]:
```
import methylize
import methylcheck
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.stats.multitest import multipletests
```

Differentially Methylated Position Analysis with Binary Phenotypes

DMPs are **probes** where methylation levels differ between two groups of samples, while DMRs are **genomic regions** where DNA methylation levels differ between two groups of samples.

Beta or m-values can be used, but the format needs to be samples as rows and probes as columns.

[3]:
```
betas, meta = methylcheck.load_both('../data/asthma') #load in the beta values and␣
↪metadata
betas = betas.T
print(meta.shape)
meta.head()
```

```
INFO:methylcheck.load_processed:Found several meta_data files; attempting to match␣
↪each with its respective beta_values files in same folders.
WARNING:methylcheck.load_processed:Columns in sample sheet meta data files does not␣
↪match for these files and cannot be combined:['../data/asthma/sample_sheet_meta_
↪data.pkl', '../data/asthma/GPL13534/GSE157651_GPL13534_meta_data.pkl']
INFO:methylcheck.load_processed:Multiple meta_data found. Only loading the first file.
INFO:methylcheck.load_processed:Loading 40 samples.
Files: 100%|| 1/1 [00:00<00:00,  8.94it/s]
INFO:methylcheck.load_processed:loaded data (485512, 40) from 1 pickled files (0.118s)
INFO:methylcheck.load_processed:Transposed data and reordered meta_data so sample␣
↪ordering matches.
INFO:methylcheck.load_processed:meta.Sample_IDs match data.index (OK)
```

```
(40, 19)
```

[3]:

	tissue	disease	smoking_status	passage	Sex	age	\
20	Airway Fibroblast	Healthy	Ex	4	M	65	
38	Parenchymal Fibroblast	Asthmatic	Non	4	F	19	
37	Parenchymal Fibroblast	Asthmatic	Non	4	F	14	
32	Parenchymal Fibroblast	Asthmatic	Current	4	F	15	
23	Airway Fibroblast	Asthmatic	Non	4	F	8	

	description	Sample_ID	Sentrix_ID	\
20	fun2norm normalized average beta	9976861129_R03C01	9976861129	
38	fun2norm normalized average beta	9976861129_R04C02	9976861129	
37	fun2norm normalized average beta	9976861129_R05C01	9976861129	
32	fun2norm normalized average beta	9976861129_R06C02	9976861129	

(continues on next page)

(continued from previous page)

```
23  fun2norm normalized average beta  9976861137_R06C01  9976861137

   Sentrix_Position Sample_Group   Sample_Name Sample_Plate Sample_Type  \
20           R03C01         None        HAF091         None     Unknown
38           R04C02         None    7294_P4_LG         None     Unknown
37           R05C01         None    7291_lg_p4         None     Unknown
32           R06C02         None  7188_lg_p4_R         None     Unknown
23           R06C01         None    7016_BR_P4         None     Unknown

   Sub_Type Sample_Well Pool_ID      GSM_ID Control
20     None         None    None  GSM4772065   False
38     None         None    None  GSM4772083   False
37     None         None    None  GSM4772082   False
32     None         None    None  GSM4772077   False
23     None         None    None  GSM4772068   False
```

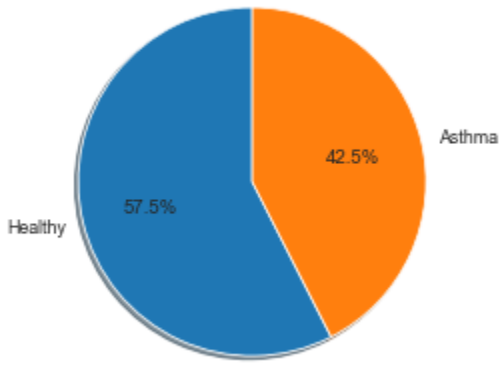
Get binary phenotype

The phenotypes can be anything list-like (i.e. a list, numpy array, or Pandas series).

When using Logistic Regression DMP, use phenotype data that is binary (has only 2 classes). In this example, we will take disease state as our phenotype because there are only 2 classes, Asthmatic and Healthy. The DMP step will automatically convert your string phenotype to `0` and `1` if there are not in that format already.

[4]:
```
pheno_data = meta.disease
print(pheno_data.shape)
dummies = pd.get_dummies(meta, columns=['disease'])[['disease_Asthmatic', 'disease_
↪Healthy']]
healthy = dummies.disease_Healthy.sum()
asthma = dummies.disease_Asthmatic.sum()
fig1, ax1 = plt.subplots()
ax1.pie((healthy, asthma), labels=['Healthy', 'Asthma'], autopct='%1.1f%%',
        shadow=True, startangle=90)
plt.show()
```

```
(40,)
```



DMP Function

This function searches for individual differentially methylated positions/probes (DMPs) by regressing the methylation beta or M-value for each sample at a given genomic location against the phenotype data for those samples. In this first example, we are using the argument `regression_method="logistic"` because we are using binary phenotypes.

`impute` argument:

- Default: 'delete' probes if any samples have missing data for that probe.
- True or 'auto': if <30 samples, deletes rows; if >=30 samples, uses average.
- False: don't impute and throw an error if NaNs present 'average' - use the average of probe values in this batch 'delete' - drop probes if NaNs are present in any sample
- 'fast' - use adjacent sample probe value instead of average (much faster but less precise)

Note: if you want to analyze every probe in your betas dataframe, remove the `.sample(N)`. What this does is randomly samples `N` probes from your dataframe and decreases the time it takes to complete this analysis.

[5]:
```
test_results = methylize.diff_meth_pos(betas.sample(20000, axis=1), pheno_data,␣
↪regression_method="logistic", export=False)
print(test_results.shape)
test_results.head()
```

```
WARNING:methylize.diff_meth_pos:Dropped 2902 probes with missing values from all␣
↪samples
```

```
All samples with the phenotype (Asthmatic) were assigned a value of 0 and all samples␣
↪with the phenotype (Healthy) were assigned a value of 1 for the logistic regression␣
↪analysis.
```

```
  0%|          | 0/17098 [00:00<?, ?it/s]
```

```
(17098, 6)
```

[5]:
```
            Coefficient  StandardError    PValue  95%CI_lower  95%CI_upper  \
cg22048228    -0.320912       0.654584  0.623955    -0.962050     1.603874
cg19366543     0.048680       1.217084  0.968095    -2.434121     2.336761
cg12408293    -0.070025       1.835602  0.969570    -3.527689     3.667739
cg08640790    -0.043054       2.412119  0.985759    -4.684612     4.770720
cg02835494     0.027930       1.860959  0.988025    -3.675343     3.619483

            FDR_QValue
cg22048228    0.999978
cg19366543    0.999978
cg12408293    0.999978
cg08640790    0.999978
cg02835494    0.999978
```

Differentially Methylated Positions Analysis with Continuous Numeric Phenotypes

Get Continuous Numeric Phenotype

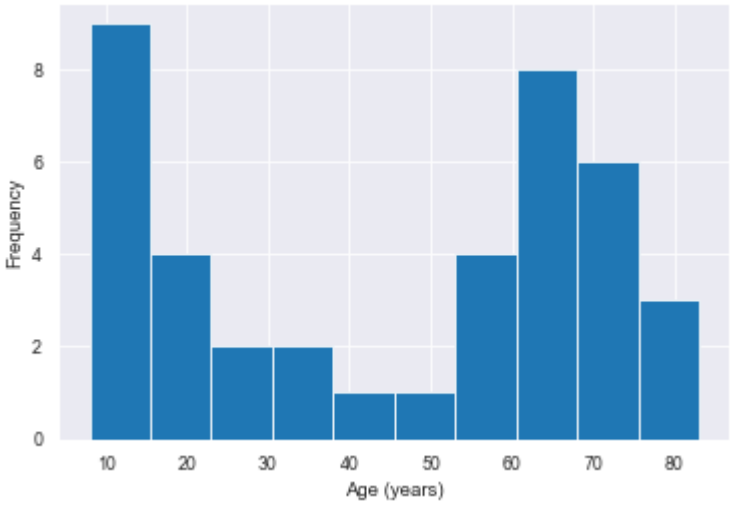
In order to find DMPs against a numeric continuous phenotype, the `linear` regression method must be used.

In this example, we are using age as the continuous numeric phenotype.

[6]:
```
pheno_data = meta.age.astype(int)
print(pheno_data.shape)
(pheno_data.sort_values()).hist()
plt.xlabel('Age (years)'); plt.ylabel('Frequency')
```

```
(40,)
```

[6]:
```
Text(0, 0.5, 'Frequency')
```



DMP Function

In this second example, we are using the argument `regression_method="linear"` because we are using continuous numeric phenotypes.

[7]:
```
test_results2 = methylize.diff_meth_pos(betas, pheno_data, regression_method="linear",
↪ export=False)
print(test_results2.shape)
test_results2.head()
```

```
WARNING:methylize.diff_meth_pos:Dropped 71139 probes with missing values from all␣
↪samples
```

```
  0%|          | 0/414373 [00:00<?, ?it/s]
```

```
(414373, 7)
```

[7]:
```
            Coefficient  StandardError        PValue  95%CI_lower  \
cg03555227     0.004583       0.000380  1.453304e-14     0.890321
cg16867657     0.008122       0.000718  9.888059e-14     0.877810
cg11169102    -0.004757       0.000460  1.323437e-12    -0.859258
cg16909962     0.009223       0.000903  1.871221e-12     0.855763
cg23045594     0.005227       0.000511  1.856802e-12     0.856031

            95%CI_upper  Rsquared    FDR_QValue
cg03555227     0.890629  0.792946  6.022099e-09
cg16867657     0.878454  0.771117  2.048672e-08
cg11169102    -0.858785  0.737918  1.550767e-07
cg16909962     0.856707  0.733140  1.550767e-07
cg23045594     0.856566  0.733247  1.550767e-07
```

To BED

This function converts the stats dataframe from the DMP function into BED format. For more information regarding this function and its arguments, please view the API Reference.

[8]:
```
bed = methylize.to_BED(test_results2, manifest_or_array_type='450k', save=False,␣
↪filename='test_bed.bed')
bed.head()
```

```
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
```

[8]:

	chrom	chromStart	chromEnd	pvalue	name
0	1	69591.0	69641.0	0.060267	cg21870274
1	1	864703.0	864753.0	0.164362	cg08258224
2	1	870161.0	870211.0	0.473560	cg16619049
3	1	877159.0	877209.0	0.267582	cg18147296
4	1	898803.0	898853.0	0.283629	cg13938959

Manhattan Plots

After we run the DMP analysis, we can move on to visualizing the results. One way to do this is a Manhattan plot.

A Manhattan plot has probe position on the x-axis (grouped and colored by chromosome) and `-log(p-value)` on the y-axis. Because of the scale on the y-axis, the greater the probe is on the y-axis, the more significant that probe is to the phenotype that was used. It is common to have a cutoff on Manhattan plots where we set our α, which commonly is `0.05`. Other common α are `0.01` and `0.001`. The lower your α is, the more significant probes are that are above that cufoff on the Manhattan plot. The cutoff is for the p-values, meaning the probe is statistically significant if the p-value < α.

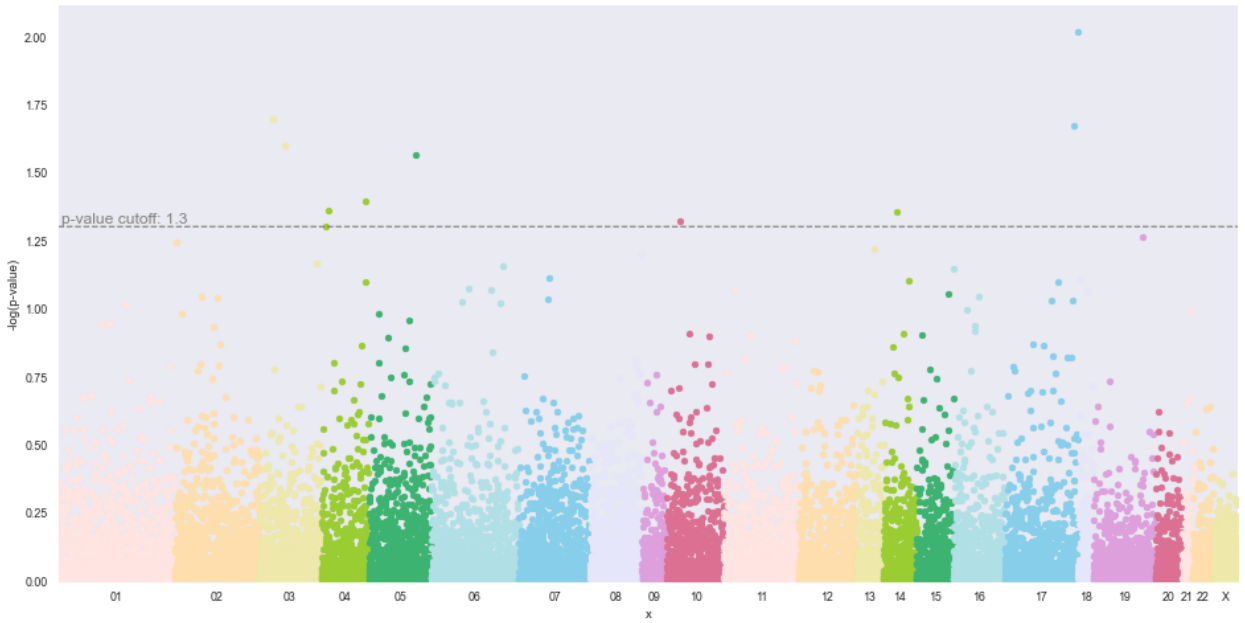
Because there are a high number of probes, a p-value correction is needed to prevent false positives. The Manhattan plot function automatically applies a Bonferroni correction to the cutoff to lower the α (increase the dotted cutoff line on the plot) to a more conservative value. The post-test correction is controlled by the `adjust` argument, where `"bonferroni"` is the default, and other options are `"fdr"` for an FDR correction, and `None` for no post-test correction to keep the cutoff the same as is entered in the `cutoff` argument.

Manhattan Plot for Binary Phenotypes

[18]:
```
methylize.manhattan_plot(test_results, cutoff=0.05, palette='default', save=False,␣
↪array_type='450k', adjust=None)
```

```
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
```

```
Total probes to plot: 17098
01 1692 | 02 1214 | 03 903 | 04 691 | 05 900 | 06 1275 | 07 1029 | 08 755 | 09 357 |␣
↪10 825 | 11 1086 | 12 830 | 13 396 | 14 477 | 15 538 | 16 747 | 17 1058 | 18 221 |␣
↪19 904 | 20 392 | 21 144 | 22 304 | X 360
```



Find Significant Probes

To see which probes are significant (above the cutoff line in the Manhattan plot), filter the results dataframe by the p-value. Because we set `post_test=None` in the arguments, this means the p-value cutoff that was set is correct, and there was no correction to this value. The next DMR example will have a p-value correction.

[10]:
```
interesting_probes = test_results[test_results['PValue'] <= 0.05]
interesting_probes
```

[10]:

	Coefficient	StandardError	PValue	95%CI_lower	95%CI_upper	\
cg11703729	-1.403577	0.708667	0.047638	0.014614	2.792539	
cg01096199	-1.518270	0.754115	0.044082	0.040232	2.996308	
cg14543255	-1.834690	0.796496	0.021254	0.273586	3.395794	
cg26494916	1.621125	0.733396	0.027075	-3.058554	-0.183696	
cg27196131	1.991505	0.985877	0.043380	-3.923787	-0.059222	
cg07351758	1.814658	0.810237	0.025113	-3.402694	-0.226623	
cg23103043	1.830804	0.707215	0.009632	-3.216920	-0.444688	
cg25314266	1.448918	0.706773	0.040360	-2.834168	-0.063668	
cg07538039	1.761092	0.897471	0.049730	-3.520103	-0.002081	
cg21664636	-1.624543	0.698487	0.020029	0.255533	2.993553	

	FDR_QValue	minuslog10pvalue	chromosome	MAPINFO
cg11703729	0.999978	1.322049	10	CHR-13809615.0
cg01096199	0.999978	1.355743	14	CHR-89724701.0
cg14543255	0.999978	1.672569	17	CHR-9002458.0
cg26494916	0.999978	1.567434	05	CHR-149453321.0
cg27196131	0.999978	1.362712	04	CHR-187176529.0
cg07351758	0.999978	1.600105	03	CHR-59766624.0
cg23103043	0.999978	2.016264	17	CHR-71650582.0
cg25314266	0.999978	1.394050	04	CHR-76188042.0
cg07538039	0.999978	1.303385	04	CHR-15374511.0
cg21664636	0.999978	1.698338	03	CHR-70383513.0

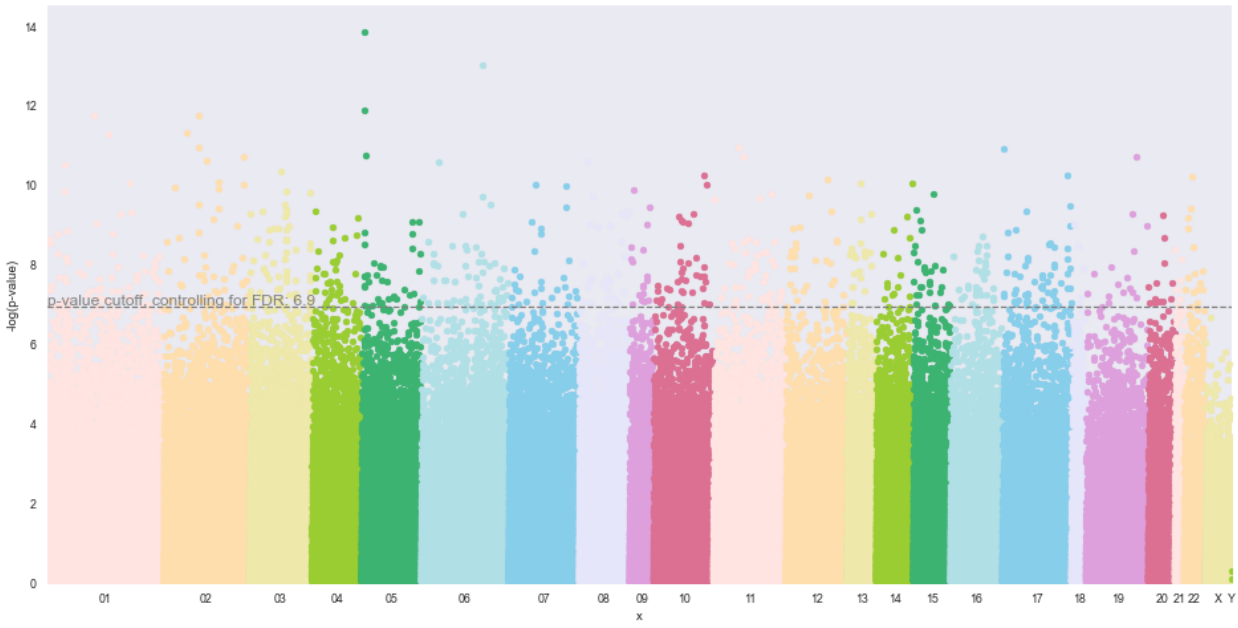
Manhattan Plot for Continuous Numeric Phenotypes

For this example, there will be a Benferoni post-test correction. The adjusted p-value cutoff is displayed on the horizontal cutoff line as well as printed above the plot. Please note that this value is the `-log(p-value)` of the p-value cutoff.

[11]:
```
methylize.manhattan_plot(test_results2, cutoff=0.05, palette='default', save=False,␣
↪array_type='450k', verbose=True)
```

```
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
```

```
Total probes to plot: 414373
01 40528 | 02 30059 | 03 21877 | 04 17328 | 05 20851 | 06 30620 | 07 24564 | 08 17814␣
↪| 09 8501 | 10 20863 | 11 25272 | 12 21229 | 13 10370 | 14 12956 | 15 12921 | 16␣
↪18404 | 17 24069 | 18 5215 | 19 21756 | 20 9259 | 21 3329 | 22 7167 | X 9419 | Y 2
p-value cutoff adjusted: 1.3010299956639813 ==[ Bonferoni ]==> 6.9184214452654755
```



Find Significant Probes

For this Manhattan plot, there has been a p-value cuttoff correction, meaning that the cutoff line is at a more conservative value (lower p-value, higher cutoff line) to account for false positives. Because of this correction, when filtering the results dataframe to find significant probes, you need to filter by the Bonferoni (or FDR) adjust p-value to find the probes above the cutoff line on the Manhattan plot.

[12]:
```
adjusted = multipletests(test_results2.PValue, alpha=0.05)
pvalue_cutoff_y = -np.log10(adjusted[3])
interesting_probes2 = test_results2[test_results2['minuslog10pvalue'] >= pvalue_
↪cutoff_y] #bonferoni correction for cutoff
interesting_probes2
```

[12]:

	Coefficient	StandardError	PValue	95%CI_lower	\
cg03555227	0.004583	0.000380	1.453304e-14	0.890321	
cg16867657	0.008122	0.000718	9.888059e-14	0.877810	

(continues on next page)

(continued from previous page)

```
cg11169102    -0.004757       0.000460  1.323437e-12    -0.859258
cg16909962     0.009223       0.000903  1.871221e-12     0.855763
cg23045594     0.005227       0.000511  1.856802e-12     0.856031
...                 ...            ...           ...          ...
cg04091914    -0.003923       0.000604  1.185223e-07    -0.726027
cg25874108    -0.002309       0.000355  1.190858e-07    -0.725717
cg01235463    -0.002696       0.000415  1.188893e-07    -0.725800
cg11126313    -0.001694       0.000261  1.199705e-07    -0.725505
cg04161137    -0.003023       0.000465  1.199026e-07    -0.725704

            95%CI_upper  Rsquared    FDR_QValue  minuslog10pvalue chromosome  \
cg03555227     0.890629  0.792946  6.022099e-09         13.837644         05
cg16867657     0.878454  0.771117  2.048672e-08         13.004889         06
cg11169102    -0.858785  0.737918  1.550767e-07         11.878297         05
cg16909962     0.856707  0.733140  1.550767e-07         11.727875         01
cg23045594     0.856566  0.733247  1.550767e-07         11.731235         02
...                 ...       ...           ...               ...        ...
cg04091914    -0.724906  0.526302  6.947979e-05          6.926200         02
cg25874108    -0.725057  0.526186  6.947979e-05          6.924140         17
cg01235463    -0.725029  0.526227  6.947979e-05          6.924857         10
cg11126313    -0.725020  0.526006  6.972307e-05          6.920925         03
cg04161137    -0.724839  0.526020  6.972307e-05          6.921172         01

                   MAPINFO
cg03555227  CHR-170862066.0
cg16867657   CHR-11044644.0
cg11169102  CHR-170858836.0
cg16909962  CHR-229270964.0
cg23045594   CHR-71276753.0
...                    ...
cg04091914  CHR-227686822.0
cg25874108   CHR-74896813.0
cg01235463   CHR-91154018.0
cg11126313   CHR-46242771.0
cg04161137   CHR-22647687.0

[713 rows x 10 columns]
```

Volcano Plot

`Logistic Regression Volcano Plots are under construction`

Below shows how to create a Volcano plot using the linear regression DMR analysis.

Positive correlation (**hypermethylated**) in red. Negative correclation (**hypomethylated**) in blue

A higher `|Regression (beta) coefficient|` means that that specific probe is more significant in predicting if the probe is methylated (positive) vs not methylated (negative). The non-gray probes are the probes that are statistically significant, and have a higher absolute value of the regression coefficient as the cutoff. The red probes further to the right on the plot are hypermethylated with increase in age, and the opposite is true for the blue probes.

Useful Arguments:

- `alpha`: Default: 0.05 alpha level The significance level that will be used to highlight the most significant adjusted p-values (FDR Q-values) on the plot. This is the horizontal cutoff line you see on the plot.
- `cutoff`: Default: No cutoff format: a list or tuple with two numbers for (min, max) If specified in kwargs, will exclude values within this range of regression coefficients from being "significant" and put dotted vertical lines

on chart. These are the vertical cutoff lines you see on the plot. These cutoffs are dependent on the study and up to the researcher in choosing which cutoff is the most desirable.

[20]:
```
methylize.volcano_plot(test_results2, fontsize=16, cutoff=(-0.0005,0.0005), alpha=0.
↪05, save=False)
```

```
Excluded 269865 probes outside of the specified beta coefficient range: (-0.0005, 0.
↪0005)
```

Find Significant Probes

[21]:
```
interesting_probes3 = test_results2[(test_results2['FDR_QValue'] <= 0.05) & (np.
↪abs(test_results2['Coefficient']) > 0.0005)]
interesting_probes3
```

[21]:

	Coefficient	StandardError	PValue	95%CI_lower	\
cg03555227	0.004583	0.000380	1.453304e-14	0.890321	
cg16867657	0.008122	0.000718	9.888059e-14	0.877810	
cg11169102	-0.004757	0.000460	1.323437e-12	-0.859258	
cg16909962	0.009223	0.000903	1.871221e-12	0.855763	
cg23045594	0.005227	0.000511	1.856802e-12	0.856031	
...	...	...	...	...	
cg18227216	-0.000972	0.000331	5.680062e-03	-0.430024	
cg16297435	0.001548	0.000528	5.681080e-03	0.428642	
cg05515570	-0.000716	0.000244	5.681038e-03	-0.429877	
cg13346820	-0.001955	0.000667	5.680922e-03	-0.430553	
cg12773197	-0.001329	0.000453	5.681300e-03	-0.430209	

	95%CI_upper	Rsquared	FDR_QValue	minuslog10pvalue	chromosome	\
cg03555227	0.890629	0.792946	6.022099e-09	13.837644	05	
cg16867657	0.878454	0.771117	2.048672e-08	13.004889	06	
cg11169102	-0.858785	0.737918	1.550767e-07	11.878297	05	
cg16909962	0.856707	0.733140	1.550767e-07	11.727875	01	
cg23045594	0.856566	0.733247	1.550767e-07	11.731235	02	

(continues on next page)

(continued from previous page)

```
...                  ...       ...           ...              ...         ...
cg18227216    -0.428965  0.184466  4.997593e-02         2.245647          17
cg16297435     0.430330  0.184459  4.997954e-02         2.245569          06
cg05515570    -0.429096  0.184459  4.997954e-02         2.245572          10
cg13346820    -0.428421  0.184460  4.997954e-02         2.245581          05
cg12773197    -0.428760  0.184457  4.998041e-02         2.245552          13

                    MAPINFO
cg03555227  CHR-170862066.0
cg16867657   CHR-11044644.0
cg11169102  CHR-170858836.0
cg16909962  CHR-229270964.0
cg23045594   CHR-71276753.0
...                    ...
cg18227216   CHR-10225251.0
cg16297435   CHR-43060558.0
cg05515570   CHR-28125056.0
cg13346820     CHR-865203.0
cg12773197  CHR-111586326.0

[43781 rows x 10 columns]
```

Differentiated Methylized Regions (DMR) Analysis

DMRs are **genomic regions** where DNA methylation levels differ between two groups of samples, while DMPs are **probes** where methylation levels differ between two groups of samples. DMR looks at clusters or adjacent probes, and if the whole cluster shows the same direction of effect between groups, then it is more significant (likely to be meaningful).

We have already ran `methylize.diff_meth_pos` with the continuous numeric phenotypes, and the statistics for that fuction is stored in the dataframe `test_restuls2`. This is what we will use in the DMR function below.

This following step should create these files: * dmr.acf.txt

- dmr.args.txt
- dmr.fdr.bed.gz
- dmr.manhattan.png
- dmr.regions-p.bed.gz
- dmr.slk.bed.gz
- dmr_regions.csv
- dmr_regions_genes.csv
- dmr_stats.csv
- stats.bed

This could run for a while. This function compares all of the probes and clusters CpG probes that show a difference together if they are close to each other in the genomic sequence.

There are many adjustible parameters for the following function, and you can refer to the API Reference for more information.

[15]:
```
files_created = methylize.diff_meth_regions(test_results2, '450k', prefix='../data/
↪asthma/dmr/')
```

```
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
/Users/jaredmeyers/opt/anaconda3/lib/python3.8/site-packages/methylize/diff_meth_
↪regions.py:150: DtypeWarning: Columns (0) have mixed types.Specify dtype option on␣
↪import or set low_memory=False.
  results = _pipeline(kw['col_num'], kw['step'], kw['dist'],
Calculating ACF out to: 453
with 17 lags: [1, 31, 61, 91, 121, 151, 181, 211, 241, 271, 301, 331, 361, 391, 421,␣
↪451, 481]
18581032 bases used as coverage for sidak correction
INFO:methylize.diff_meth_regions:wrote: ../data/asthma/dmr/.regions-p.bed.gz,␣
↪(regions with corrected-p < 0.05: 29)
/var/folders/3k/vrkjxj8s7l1gq4x6n7ymqncw0000gn/T/ipykernel_10952/1491901786.py:1:␣
↪DtypeWarning: Columns (0) have mixed types.Specify dtype option on import or set␣
↪low_memory=False.
  files_created = methylize.diff_meth_regions(test_results2, '450k', prefix='../data/
↪asthma/dmr/')
INFO:methylize.genome_browser:Loaded 4802 CpG regions from ../data/asthma/dmr/_
↪regions.csv.
INFO:methylize.genome_browser:Using cached `refGene`: /Users/jaredmeyers/opt/
↪anaconda3/lib/python3.8/site-packages/methylize/data/refGene.pkl with (135634) genes
Mapping genes: 100%|| 135634/135634 [00:30<00:00, 4381.01it/s]
INFO:methylize.genome_browser:Wrote ../data/asthma/dmr/_regions_genes.csv
```

Many of the files created by this function can be useful when using other tools. The file with the main summary of what was just done is found in `dmr_regions_genes.csv`.

[16]:
```
regions_genes = pd.read_csv('../data/asthma/dmr_regions_genes.csv')
regions_genes.head()
```

[16]:

	Unnamed: 0	chrom	chromStart	chromEnd	min_p	n_probes	z_p	z_sidak_p	name	genes	distances	descriptions
0	0	1	22289407	22289457	0.03571	1	0.000094	0.7979	cg02053477	NaN	NaN	NaN
1	1	1	34757646	34757696	0.03110	1	0.000054	0.6018	cg13170235	SNRPC	142	Homo sapiens small nuclear ribonucleoprotein p...
2	2	1	34761106	34761156	0.03401	1	0.000081	0.7500	cg15750705	NaN	NaN	NaN
3	3	1	202858152	202858202	0.03110	1	0.000038	0.4790	cg15569630	NaN	NaN	NaN
4	4	10	22334619	22334669	0.03110	1	0.000038	0.4790	cg13327545	NaN	NaN	NaN

This shows clusters of CpG probes that were significantly different and annotes these clusters with mne or more nearby genes using the UCSV Genome Browswer database.

Distances are the number of base-pairs that separate the start of the CpG probe and the start of the coding sequence of the gene.

Only the rows with signicicant p-values will have an annotation.

Adopted from the `combined-pvalues` package by Brend Pedersen et al, 2013: Comb-p: software for combining, analyzing, grouping and correcting spatially correlated P-values doi: 10.1093/bioinformatics/bts545

More information on DMR on this page

Fetching Genes

This function annotates the DMR region output file using the UCSC Genome Browser database as a reference to what genes are nearby. For more information on this function and the specific argument, please refer to the API Reference

This function can now accept either the filepath or the DMR dataframe output.

[17]:
```
reg = pd.read_csv('../data/asthma/dmr_regions.csv')
genes = methylize.fetch_genes('../data/asthma/dmr_regions.csv')
genes.head(12)
```

```
INFO:methylize.genome_browser:Loaded 60 CpG regions from ../data/asthma/dmr_regions.
↪csv.
INFO:methylize.genome_browser:Using cached `refGene`: /Users/jaredmeyers/opt/
↪anaconda3/lib/python3.8/site-packages/methylize/data/refGene.pkl with (135634) genes
Mapping genes: 100%|| 135634/135634 [00:28<00:00, 4768.33it/s]
INFO:methylize.genome_browser:Wrote ../data/asthma/dmr_regions_genes.csv
```

[17]:

	chrom	chromStart	chromEnd	min_p	n_probes	z_p	z_sidak_p	\
0	1	22289407	22289457	0.03571	1	0.000094	0.7979	
1	1	34757646	34757696	0.03110	1	0.000054	0.6018	
2	1	34761106	34761156	0.03401	1	0.000081	0.7500	
3	1	202858152	202858202	0.03110	1	0.000038	0.4790	
4	10	22334619	22334669	0.03110	1	0.000038	0.4790	
5	11	2170376	2170426	0.03110	1	0.000038	0.4790	
6	11	2170376	2170426	0.03110	1	0.000038	0.4790	
7	11	2346840	2346890	0.04207	1	0.000138	0.9041	
8	11	61755269	61755319	0.04150	1	0.000133	0.8968	
9	11	69403865	69403915	0.03110	1	0.000068	0.6847	
10	11	114063572	114063622	0.03110	1	0.000054	0.6018	
11	11	117861576	117861626	0.03110	1	0.000038	0.4790	

	name	genes	distances	\
0	cg02053477			
1	cg13170235	SNRPC	142	
2	cg15750705			
3	cg15569630			
4	cg13327545			
5	cg11911653	VARS2	-165,-135	
6	cg14714364	VARS2	-165,-135	
7	cg05532869			
8	cg11179625	MYRF	-108	
9	cg05582286	TACR2	-37	
10	cg10096321			
11	cg14848289			

	descriptions
0	
1	Homo sapiens small nuclear ribonucleoprotein p...
2	
3	
4	

(continues on next page)

(continued from previous page)

```
5   Homo sapiens valyl-tRNA synthetase 2, mitochon...
6   Homo sapiens valyl-tRNA synthetase 2, mitochon...
7
8   Homo sapiens myelin regulatory factor (MYRF), ...
9   Homo sapiens tachykinin receptor 2 (TACR2), mR...
10
11
```

1.8.3 Example: Differentially methylated regions – GSE69852

- Using GEO dataset GSE69852 (6 samples, 450k)
- **Title:** Patterns of DNA methylation in human fetal and adult liver
- https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE69852
- We demonstrate a basic (beta_values + meta_data) –> DMP –> DMR calculation.

Claims in the paper: - Nearly half (42%) of the CpGs in human liver show a significant difference in methylation comparing fetal and adult samples. - 69% of the significant sites differed in their mean methylation beta value by 0.2.

Note on phenotype: The original meta data had ages of samples in different units (e.g. 55yr, 21wk) so we manually converted these to a single measurement (converted_age, float, years) so that linear regression could run. Alternatively, you could treat the two groups (55-56 years vs 20-22 weeks) as a phenotype with two groups (1 and 0) and apply logistic regression.

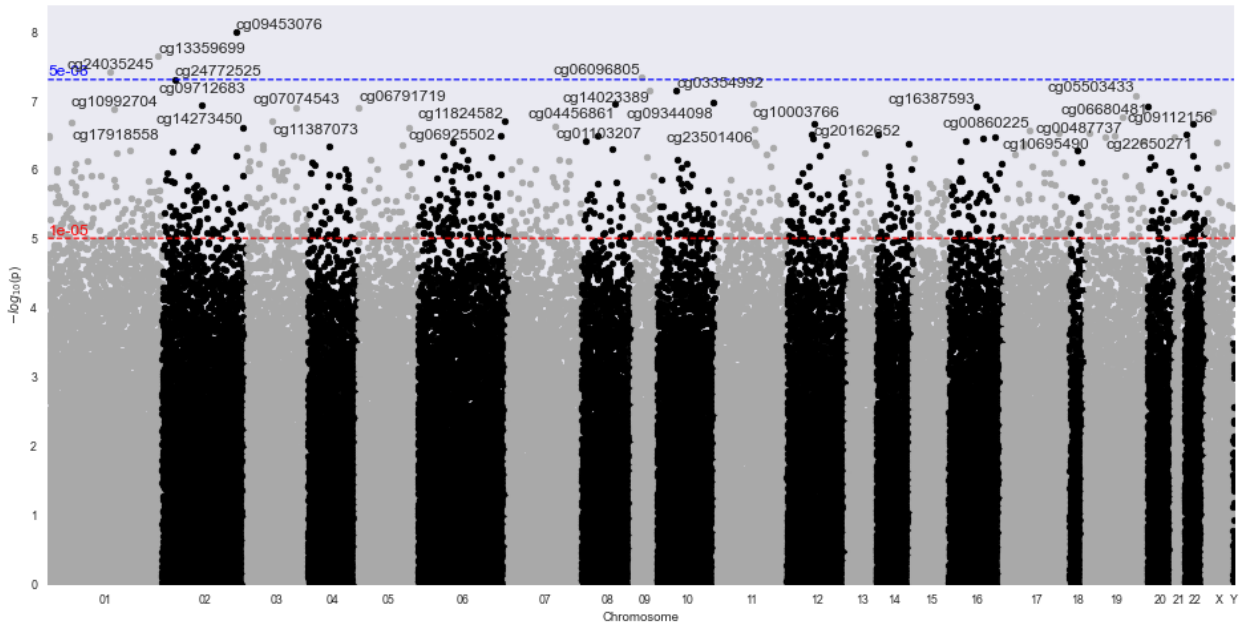
[1]:
```
import methylize as m
import pandas as pd
from pathlib import Path
df = pd.read_pickle(Path('data/GSE69852_beta_values.pkl'))
meta = pd.read_pickle(Path('data/GSE69852_GPL13534_meta_data.pkl'))
```

[2]:
```
res = m.diff_meth_pos(df.sample(100000), meta['converted_age'])
m.manhattan_plot(res, '450k')
bed = m.diff_meth_regions(res, '450k', prefix='data', plot=True)
```

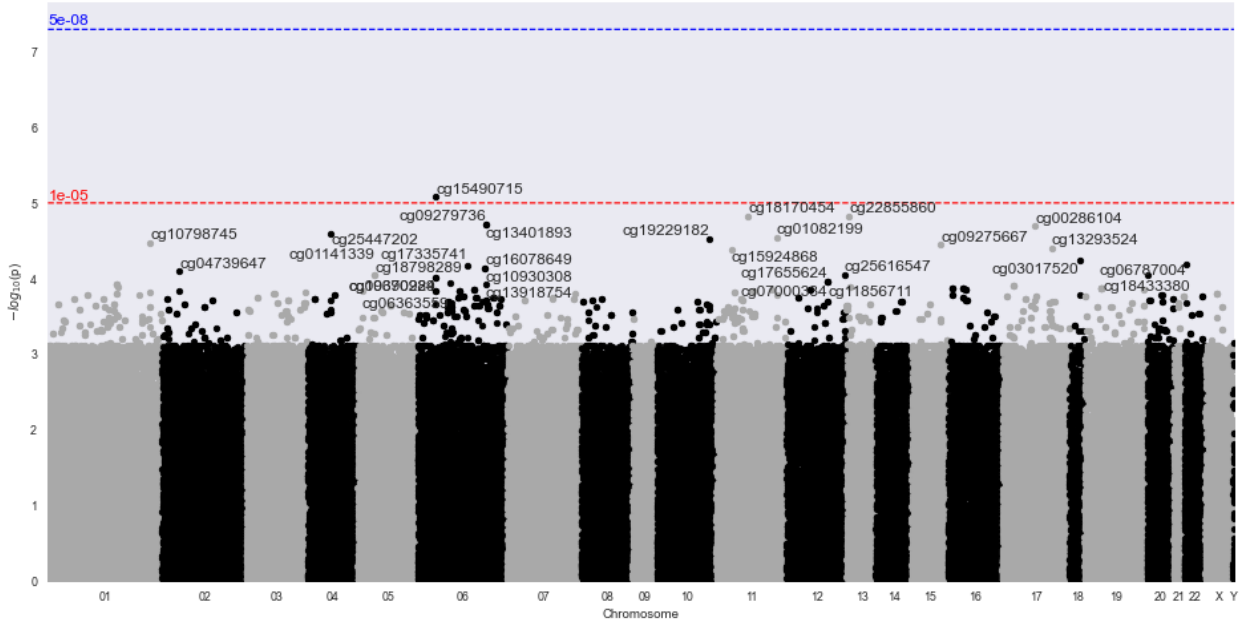
```
INFO:methylize.diff_meth_pos:Converted your beta values into M-values; (6, 100000)
```

```
Probes:   0%|          | 0/100000 [00:00<?, ?it/s]
```

```
8 NaNs dropped
```




```
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
Calculating ACF out to: 367
with 14 lags: [1, 31, 61, 91, 121, 151, 181, 211, 241, 271, 301, 331, 361, 391]
4873754 bases used as coverage for sidak correction
INFO:methylize.diff_meth_regions:wrote: data.regions-p.bed.gz, (regions with␣
↪corrected-p < 0.05: 0)
```




```
INFO:methylize.genome_browser:Loaded 24371 CpG regions from data_regions.csv.
INFO:methylize.genome_browser:Using cached `refGene`: /Users/mmaxmeister/methylize/
↪methylize/data/refGene.pkl with (135634) genes
```

(continues on next page)

(continued from previous page)

```
Mapping genes: 100%|| 135634/135634 [00:59<00:00, 2277.89it/s]
INFO:methylize.genome_browser:Wrote data_regions_genes.csv
```

One would interpret the DMR plot to indicate that there were no significantly different methylated regions. Note, this example only ran 100000 of the 485,512 probes.

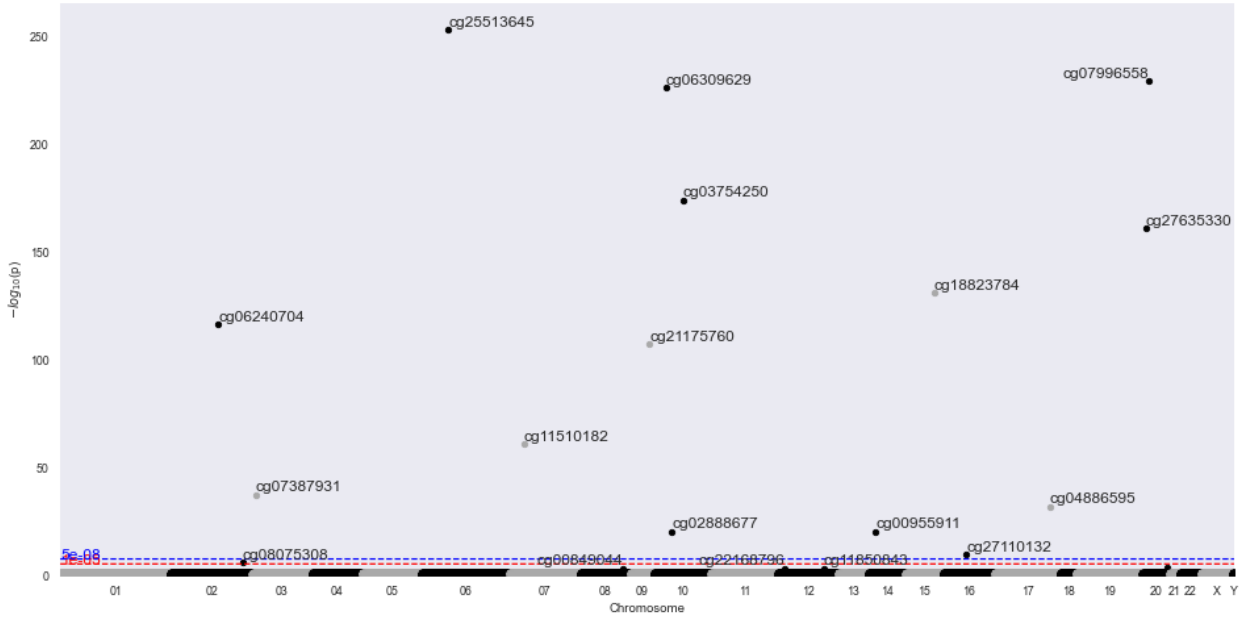
1.8.4 All data, logistic regression

[3]:
```
result = m.diff_meth_pos(df, meta['description'])
m.manhattan_plot(result, '450k')
bed_files = m.diff_meth_regions(result, '450k', prefix='data', plot=True)
```

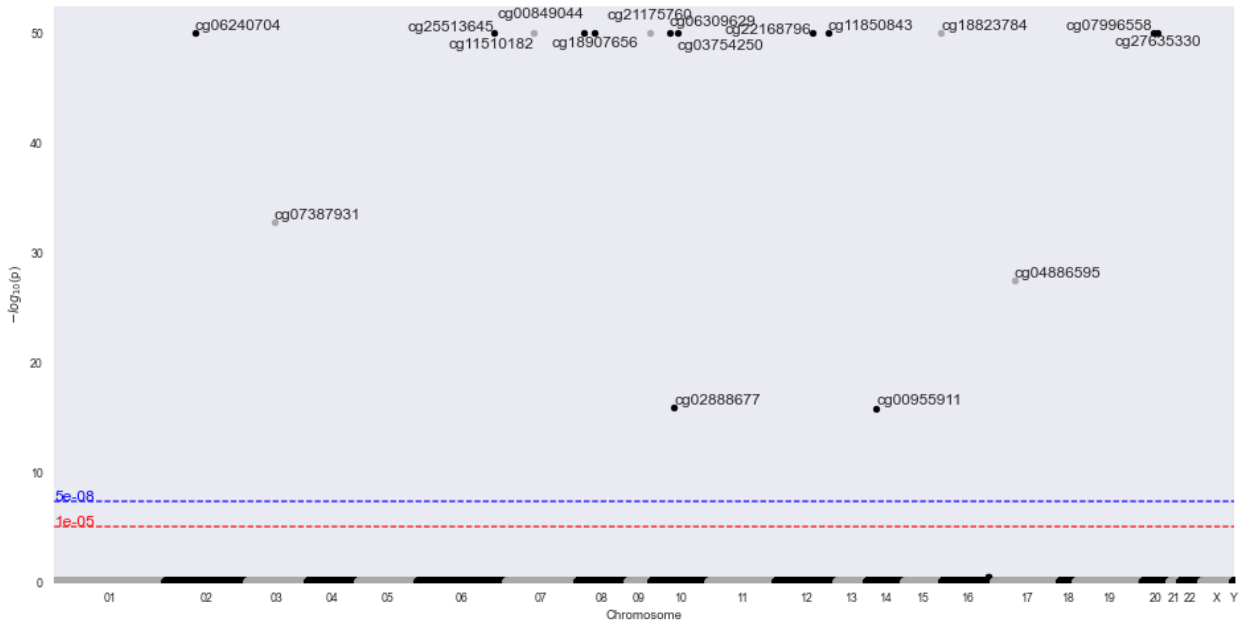
```
INFO:methylize.diff_meth_pos:Converted your beta values into M-values; (6, 485512)
INFO:methylize.diff_meth_pos:Logistic regression: Phenotype (Normal Adult liver) was␣
↪assigned to 0 and (Normal Fetal Liver) was assigned to 1.
```

```
Probes:   0%|          | 0/485512 [00:00<?, ?it/s]
```

```
218806 probes failed the logistic regression analysis due to perfect separation and␣
↪could not be included in the final results.
13209 probes failed the logistic regression analysis due to LinearAlgebra error and␣
↪could not be included in the final results.
148 probes failed the logistic regression analysis due to other unexplained reasons␣
↪and could not be included in the final results.
Linear Algebra and Perfect Separation errors occur when the dataset is (a) too small␣
↪to observe
events with low probabilities, or has (b) too many covariates in the model, leading␣
↪to individual cell
sizes that are too small. Singular Matrix Errors occur when there is no variance in␣
↪the predictors
(probe values and covariates).
27 NaNs dropped
```



```
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
INFO:methylprep.files.manifests:Reading manifest file: HumanMethylation450k_15017482_
↪v3.csv
Calculating ACF out to: 50
with 3  lags: [1, 31, 61]
11192348 bases used as coverage for sidak correction
INFO:methylize.diff_meth_regions:wrote: data.regions-p.bed.gz, (regions with␣
↪corrected-p < 0.05: 16)
```




```
INFO:methylize.genome_browser:Loaded 17 CpG regions from data_regions.csv.
INFO:methylize.genome_browser:Using cached `refGene`: /Users/mmaxmeister/methylize/
↪methylize/data/refGene.pkl with (135634) genes
Mapping genes: 100%|| 135634/135634 [00:56<00:00, 2381.53it/s]
INFO:methylize.genome_browser:Wrote data_regions_genes.csv
```

[4]:
```
bed_files
```

[4]:
```
['data.args.txt',
 'data.acf.txt',
 'data.fdr.bed.gz',
 'data.slk.bed.gz',
 'data.regions-p.bed.gz',
 'data_regions.csv',
 'data_stats.csv',
 'data_regions_genes.csv',
 'data_dmp_stats.bed']
```

[5]:
```
bed = pd.read_csv('data.regions-p.bed.gz', sep='\t')
bed
```

[5]:

	#chrom	start	end	min_p	n_probes	z_p	\
0	10	4034204	4034254	0.000000e+00	1	1.000000e-07	
1	10	31031409	31031459	0.000000e+00	1	1.000000e-07	
2	10	75210411	75210461	2.351000e-12	1	1.392000e-16	
3	12	43840121	43840171	0.000000e+00	1	1.000000e-07	

(continues on next page)

(continued from previous page)

```
4       12  124453780  124453830  0.000000e+00          1  1.000000e-07
5       14   37592783   37592833  2.598000e-12          1  1.641000e-16
6       15   78810879   78810929  0.000000e+00          1  1.000000e-07
7       17   29005861   29005911  6.351000e-24          1  3.510000e-28
8        2  200911124  200911174  0.000000e+00          1  1.000000e-07
9       20    1470868    1470918  0.000000e+00          1  1.000000e-07
10      20   58445879   58445929  0.000000e+00          1  1.000000e-07
11       3  141066318  141066368  3.702000e-29          1  1.900000e-33
12       6   35150826   35150876  0.000000e+00          1  1.000000e-07
13       7   24854884   24854934  0.000000e+00          1  1.000000e-07
14       8  144335305  144335355  0.000000e+00          1  1.000000e-07
15       9  133804066  133804116  0.000000e+00          1  1.000000e-07

       z_sidak_p
0   2.214000e-02
1   2.214000e-02
2   2.485000e-11
3   2.214000e-02
4   2.214000e-02
5   2.485000e-11
6   2.214000e-02
7   7.857000e-23
8   2.214000e-02
9   2.214000e-02
10  2.214000e-02
11  4.253000e-28
12  2.214000e-02
13  2.214000e-02
14  2.214000e-02
15  2.214000e-02
```

1.8.5 API Reference

methylize.diff_meth_pos(meth_data, pheno_data)	This function searches for individual differentially methylated positions/probes (DMPs) by regressing the methylation M-value for each sample at a given genomic location against the phenotype data for those samples.
methylize.diff_meth_regions(stats, ...)	Calculates and annotates diffentially methylated regions (DMR) using the *combined-pvalues pipeline* and returns list of output files.
methylize.fetch_genes([dmr_regions_file, ...])	find genes that are adjacent to significantly different CpG regions provided.
methylize.manhattan_plot(stats_results, ...)	In EWAS Manhattan plots, epigenomic probe locations are displayed along the X-axis, with the negative logarithm of the association P-value for each single nucleotide polymorphism (SNP) displayed on the Y-axis, meaning that each dot on the Manhattan plot signifies a SNP.
methylize.volcano_plot(stats_results, **kwargs)	This function writes the pandas DataFrame output of diff_meth_pos() to a CSV file named by the user.

Continued on next page

Table 1 – continued from previous page

methylize.helpers.to_BED(stats, ...[, save, ...])	Converts & exports manifest and probe p-value dataframe to BED format.

differentially methylated positions

methylize.diff_meth_pos.**diff_meth_pos**(*meth_data*, *pheno_data*, *regression_method=None*, *impute='delete'*, ***kwargs*)

This function searches for individual differentially methylated positions/probes (DMPs) by regressing the methylation M-value for each sample at a given genomic location against the phenotype data for those samples.

Phenotypes can be provided as a list of string-based or integer binary data or as numeric continuous data.

Input Parameters:

meth_data: A pandas dataframe of methylation array data (as M-values) where each column corresponds to a CpG site probe and each row corresponds to a sample. IF a dataframe of beta-values is supplied instead, this function will detect this and convert to M-values before proceeding.

pheno_data: A list or one dimensional numpy array of phenotypes for each sample row in meth_data. Binary phenotypes can be presented as a list/array of zeroes and ones or as a list/array of strings made up of two unique words (i.e. "control" and "cancer").

- linear regression requires a measurement for the phenotype
- note: methylprep creates a *sample_sheet_meta_data.pkl* file containing the phenotype

data for this input. You just need to load it and specify which *column* to be used as the pheno_data. - If pheno_data is a pandas DataFrame, you must specify a *column* and may optionally specify *covariates* from the other columns in the DataFrame. Numpy matrices are NOT supported.

column:

- If pheno_data is a DataFrame, column='label' will select one series to be used as the phenotype data.

covariates: (default None) use to define a more complex model for logistic regression.

- if pheno_data is not a DataFrame, covariates will be ignored.
- if pheno_data is a DataFrame, and *column* is specified, and covariates is None, it will ignore other columns in pheno_data.
- if covariates is a list of strings, only these column names will be used as the covariate data from pheno_data.
- If covariates is set to True, then all other columns will be treated as covariates in logistic regression.

regression_method: (logistic | linear)

- Either the string "logistic" or the string "linear"

depending on the phenotype data available. - Phenotypes with only two options (e.g. "control" and "cancer") can be analyzed with a logistic regression - Continuous numeric phenotypes (e.g. age) are required to run a linear regression analysis. - Default: auto-determine, using the data types found in *pheno_data*. If it is numeric with more than 2 different labels, use "linear."

impute: Because methylprep output contains missing values by default, this function requires user to either delete or impute missing values.

- Default: 'delete' probes if ANY samples have missing data for that probe.
- True or 'auto': if <30 samples, deletes rows; if >=30 samples, uses average.
- False: don't impute and throw an error if NaNs present
- 'average' - use the average of probe values in this batch
- 'delete' - drop probes if NaNs are present in any sample
- 'fast' - use adjacent sample probe value instead of average (much faster but less precise)

alpha: float Default is 0.05 for all tests where it applies.

fwer: float Set the familywise error rate (FWER). Default is 0.05, meaning that we expect 5% of all significant differences to be false positives. 0.1 (10%) is a more conservative convention for FWER.

q_cutoff:

- Select a cutoff value (< 1.0) to return filtered results.

Only those DMPs that meet a particular significance threshold (e.g. 0.1) will be retained. Reported q-values are p-values corrected according to the model's false discovery rate (FDR). - Default: 1 – returns all DMPs regardless of significance.

export:

- default: False
- if True or 'csv', saves a csv file with data
- if 'pkl', saves a pickle file of the results as a dataframe.
- Use q_cutoff to limit what gets saved to only significant results.

By default, *q_cutoff == 1* and this means everything is saved/reported/exported.

filename:

- specify a filename for the exported file.

By default, if not specified, filename will be *DMP_<number of probes in file>_<number of samples processed>_<current_date>.<pkl|csv>*

max_workers: (=INT) By default, this will parallelize probe processing, using all available cores. During testing, or when running in a virtual environment like circleci or docker or lambda, the number of available cores is fewer than the system's reported CPU cores, and it breaks. Use this to limit the available cores to some arbitrary number for testing or containerized-usage.

debug: Default: False – True turns on extra messages and runs the 'solver' in serial mode, disabling parallel processing of probes. This is slower but provides more detail for debugging code.

solver: You can force it to use a different implementation of regression, for debugging. Options include:

- 'statsmodels_OLS'
- 'linregress' # from scipy
- [logit_DMP() is the default, based on statsmodels.api.Logit()]
- 'statsmodels_GLM' # for logistic regression

Returns:

A pandas dataframe of regression statistics with a row for each probe analyzed and columns listing the individual probe's regression statistics of:

- regression coefficient
- lower limit of the coefficient's 95% confidence interval
- upper limit of the coefficient's 95% confidence interval
- standard error
- p-value (phenotype group A vs B - likelihood that the difference is significant for this probe/location)
- FDR_QValue: p-values corrected for multiple comparisons using the Benjamini-Hochberg FDR method

The rows are sorted by q-value in ascending order to list the most significant probes first. If q_cutoff is specified, only probes with significant q-values less than the cutoff will be returned in the dataframe.

Note:

If you get a *RuntimeError: main thread is not in main loop* error running *diff_meth_pos*, add *debug=True* to your function inputs. This error occurs in some command line environments when running this function repeatedly, and the parallel processing steps do not all close. Using debug mode forces it to analyze the probes serially, not parallel. It is a little slower but will always run without error under these conditions.

If Progress Bar Missing: if you don't see a progress bar in your jupyterlab notebook, try this: - conda install -c conda-forge nodejs - jupyter labextension install @jupyter-widgets/jupyterlab-manager

---

Todo: *shrink_var* feature: variance shrinkage / squeeze variance using Bayes posterior means. Variance shrinkage is recommended when analyzing small datasets (n < 10). Paper ref: http://www.uvm.edu/~rsingle/JournalClub/papers/Smyth-SAGMB-2004_eBayes+microarray.pdf [NOT IMPLEMENTED YET]

`methylize.diff_meth_pos.`**`legacy_OLS`**(*probe_data*, *phenotypes*, *alpha=0.05*)

to use this, specify "statsmodels_OLS" in kwargs to diff_meth_pos() – this method gives the same result as the scipy.linregress method when tested in version 1.0.0

`methylize.diff_meth_pos.`**`linear_DMP_regression`**(*probe_data*, *phenotypes*, *alpha=0.05*)

This function performs a linear regression on a single probe's worth of methylation data (in the form of beta or M-values). It is called by the diff_meth_pos().

Inputs and Parameters:

probe_data: A pandas Series for a single probe with a methylation M-value/beta-value for each sample in the analysis. The Series name corresponds to the probe ID, and the Series is extracted from the meth_data DataFrame through a parallellized loop in diff_meth_pos().

phenotypes: A numpy array of numeric phenotypes with one phenotype per sample (so it must be the same length as probe_data). This is the same object as the pheno_data input to diff_meth_pos() after it has been checked for data type and converted to the numpy array pheno_data_array.

Returns:

A pandas Series of regression statistics for the single probe analyzed. The columns of regression statistics are as follows:

- regression coefficient (linregress pearson's 'r')
- lower limit of the coefficient's 95% confidence interval
- upper limit of the coefficient's 95% confidence interval

- standard error
- p-value

methylize.diff_meth_pos.**logistic_DMP_regression**(*probe_data*, *phenotypes*, *covariate_data=None*, *debug=False*)

- Runs parallelized.
- TESTED, and gives almost the same values as logit_DMP() (2022-02-25)
- This function performs a logistic regression on a single probe's worth of methylation

data (in the form of beta/M-values). It is called by the diff_meth_pos().

Inputs and Parameters:

probe_data: A pandas Series for a single probe with a methylation M-value or beta_value for each sample in the analysis. The Series name corresponds to the probe ID, and the Series is extracted from the meth_data DataFrame through a parallellized loop in diff_meth_pos().

phenotypes: A numpy array of binary phenotypes with one phenotype per sample (so it must be the same length as probe_data). This is the same object as the pheno_data input to diff_meth_pos() after it has been checked for data type and converted to the numpy array pheno_data_binary.

Returns:

A pandas Series of regression statistics for the single probe analyzed. The columns of regression statistics are as follows:

- regression coefficient
- lower limit of the coefficient's 95% confidence interval
- upper limit of the coefficient's 95% confidence interval
- standard error
- p-value

If the logistic regression was unsuccessful in fitting to the data due to a Perfect Separation Error (as may be the case with small sample sizes) or a Linear Algebra Error, the exception will be caught and the probe_stats_row output will contain dummy values to flag the error. Perfect Separation Errors are coded with the value -999 and Linear Algebra Errors are coded with value -995. These rows are processed and removed in the next step of diff_meth_pos() to prevent them from interfering with the final analysis and p-value correction while printing a list of the unsuccessful probes to alert the user to the issues.

methylize.diff_meth_pos.**logit_DMP**(*probe_data*, *phenotypes*, *covariate_data=None*, *debug=False*)

DEFAULT method, because tested and works. uses statsmodels.api.Logit pass in a Series for probe data without the constant added

fold_change is log2((pheno1.mean / pheno0.mean))

each covariate will be normalized (to 0...1 range) before running DMP.

methylize.diff_meth_pos.**manhattan_plot**(*stats_results*, *array_type*, ***kwargs*)

In EWAS Manhattan plots, epigenomic probe locations are displayed along the X-axis, with the negative logarithm of the association P-value for each single nucleotide polymorphism (SNP) displayed on the Y-axis, meaning that each dot on the Manhattan plot signifies a SNP. Because the strongest associations have the smallest P-values (e.g., 1015), their negative logarithms will be the greatest (e.g., 15).

- genomic coordinates along chromosomes vs epigenetic probe locations along chromosomes

- p-values are for the probe value associations, using linear or logistic regression,

between phenotype A and B.

Hints of hidden heritability in GWAS. Nature 2010. (https://www.ncbi.nlm.nih.gov/pubmed/20581876)

stats_results: a pandas DataFrame containing the stats_results from the linear/logistic regression run on m_values or beta_values and a pair of sample phenotypes. The DataFrame must contain A "PValue" column. the default output of diff_meth_pos() will work.

array_type: specify the type of array [450k, epic, epic+, mouse, 27k], so that probes can be mapped to chromosomes.

save: specify that it export an image in *png* format. By default, the function only displays a plot.

filename: specify an export filename. The default is *f"manhattan_<stats>_<timestamp>.png"*.

- *verbose* (True/False) - default is True, verbose messages, if omitted.
- *fwer* (default 0.1) familywise error rate (fwer) is used to set p-value threshold and the FDR threshold line.
- *genome_build* – NEW or OLD. Default is NEWest genome_build.
- ***label-prefix* – how to refer to chromosomes. By default, it shows numbers like 1 … 22, and X, Y.** pass in 'CHR-' to add a prefix to plot labels, or rename with 'c' like: c01 … c22.

There are some preset "override" options for threshold lines on plots:

- ***explore*: (default False) include all FOUR significance threshold lines on plot** (suggestive, significant, bonferroni, and false discovery rate). Useful for data exploration.
- *no_thresholds*: (default False) set to True to hide all FOUR threshold lines from plot.
- *plain*: (default False) hide all lines AND don't label significant probes.
- *statsmode*: (default False) show the FDR and Bonferroni thresholds, hide the suggestive and genomic significant lines.

These allow you to toggle lines/labels on or off:

- *fdr*: (default False) draw a threshold line on plot corresponding to the adjusted false discovery rate = 0.05
- *bonferroni*: (default False) draw the Bonferroni threshold, correcting for multiple comparisons.
- ***suggestive*: (default 1e-5) draw the consensus "suggestive significance" theshold at p < 1e-5,** or set to False to hide.
- ***significant*: (default 5e-8) draw the consensus "genomic significance" theshold at p < 5e-8,** or set to False to hide.
- *plot_cutoff_label* (default True) label to each dotted line on the plot
- *label_sig_probes* (default True) labels significant probes showing the greatest difference between groups.

Chart options:

- *ymax* – default: 50. Set to avoid plotting extremely high -10log(p) values.
- *width* – figure width – default is 16
- *height* – figure height – default is 8

- *fontsize* – figure font size – default 16
- *border* – plot border – default is OFF
- *palette* – specify one of a dozen options for colors of chromosome regions on plot: ['default', 'Gray', 'Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20b', 'tab20c', 'Gray2', 'Gray3']

methylize.diff_meth_pos.**probe_corr_plot**(*stats*, *group='sig'*, *colorby='pval'*)

- group='sig' is default (using PValue < 0.05)
- group='chromosome' also kinda works.
- colorby= pval or FDR; what to use to color the significant probes, if group='sig'

methylize.diff_meth_pos.**volcano_plot**(*stats_results*, ***kwargs*)

This function writes the pandas DataFrame output of diff_meth_pos() to a CSV file named by the user. The DataFrame has a row for every successfully tested probe and columns with different regression statistics as follows:

- regression coefficient
- lower limit of the coefficient's 95% confidence interval
- upper limit of the coefficient's 95% confidence interval
- standard error
- p-value
- FDR q-value (p-values corrected for multiple testing using the Benjamini-Hochberg FDR method)

Inputs and Parameters:

stats_results (required): A pandas DataFrame output by the function diff_meth_pos().

'alpha': Default: 0.05, The significance level that will be used to highlight the most significant p-values (or adjusted FDR Q-values) on the plot.

'cutoff': the beta-coefficient cutoff | Default: None format: a list or tuple with two numbers for (min, max) or 'auto'. If specified in kwargs, will exclude values within this range of regression coefficients OR fold-change range from being "significant" and put dotted vertical lines on chart. 'auto' will select a beta coefficient range that excludes 95% of results from appearing significant.

'adjust': (default False) – if this will adjust the p-value cutoff line for false discovery rate (Benjamini-Hochberg). Use 'fwer' to set the target rate.

'fwer': family-wise error rate (default is 0.1) – specify a probability [0 to 1.0] for false discovery rate

'data_type_label': What to put on X-axis. Either 'Fold Change' (default) or 'Regression Coefficient'.

visualization kwargs:

- ***palette* – color pattern for plot – default is [blue, red, grey]** other palettes: ['default', 'Gray', 'Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20b', 'tab20c', 'Gray2', 'Gray3']
- *width* – figure width – default is 16
- *height* – figure height – default is 8
- *fontsize* – figure font size – default 16

- *dotsize* – figure dot size on chart – default 30
- *border* – plot border – default is OFF
- *data_type_label* – (e.g. Beta Values, M Values) – default is 'Beta'
- *plot_cutoff_label* – add label to dotted line on plot – default False

save: specify that it export an image in *png* format. By default, the function only displays a plot.

filename: specify an export filename. default is *volcano_<current_date>.png*.

Returns:

Displays a plot, but does not directly return an object. The data is color coded and displayed as follows:

- the negative log of adjusted p-values is plotted on the y-axis
- the regression coefficient beta value is plotted on the x-axis
- the significance cutoff level appears as a horizontal gray dashed line
- non-significant points appear in light gray
- significant points with positive correlations (hypermethylated probes) appear in red
- significant points with negative correlations (hypomethylated probes) appear in blue

differentially methylated regions

`methylize.diff_meth_regions.`**`diff_meth_regions`**(*stats*, *manifest_or_array_type*, ***kwargs*)

Calculates and annotates diffentially methylated regions (DMR) using the *combined-pvalues pipeline* and returns list of output files.

comb-p is a command-line tool and a python library that manipulates BED files of possibly irregularly spaced P-values and

(1) calculates auto-correlation,

(2) combines adjacent P-values,

(3) performs false discovery adjustment,

(4) finds regions of enrichment (i.e. series of adjacent low P-values) and

(5) assigns significance to those regions.

ref: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3496335/

Input Parameters:

stats: dataframe dataframe output from diff_meth_pos()

manifest_or_array_type: class instance or string pass in the manifest, or the name of the array

filename: filename to prepend to the .BED file created.

creates a <filename>.bed output from diff_meth_pos and to_BED. genome_build: default 'NEW'

by default, it uses the NEWer genome build. Each manifest contains two genome builds, marked "NEW" and "OLD". To use the OLD build, se this to "OLD".

Computational Parameters:

dist: int maximum distance from each probe to scan for adjacent peaks (80)

acf-dist: int window-size for smoothing. Default is 1/3 of peak-dist (dist kwarg),

step: int step size for bins in the ACF calculation (50)

threshold: foat Extend regions after seeding if pvalue exceeds this value (default: same as seed)

seed: float minimum size of a genomic region (0.05)

no_fdr: bool don't use FDR

genome_control: bool correct input pvalues for genome control (inflation factor). This reduces the confounding effects of population stratitication in EWAS data.

Display/output Paramters:

verbose: bool – default False Display additional processing information on screen.

plot: bool – default False False will suppress the manhattan plot step.

prefix: str prefix that gets appended to all output files (e.g. 'dmr' becomes 'dmr_regions.csv')

region_filter_p: max adjusted region-level p-value in final output.

region_filter_n: req at least this many probes in a region

annotate: bool annotate with fetch_genes() function that uses UCSC refGene database to add "nearby" genes to differentially methylated regions in the output CSV. If you want to fine-tune the reference database, the tolerance of what "nearby" means, and other parameters, set this to false and call *methylize.fetch_genes* as a separate step on the '. . . _regions.csv' output file.

tissue: str if specified, adds additional columns to the annotation output with the expression levels for identified genes in any/all tissue(s) that match the keyword. (e.g. if your methylation samples are whole blood, specify *tissue=blood*) For all 54 tissues, use *tissue=all*

Returns:

list A list of files created.

Fetch Genes

find genes that are adjacent to significantly different CpG regions provided.

Summary:

fetch_genes() annotates the DMR region output file, using the UCSC Genome Browser database as a reference as to what genes are nearby. This is an exploratory tool, as there are many versions of the human genome that map genes to slightly different locations.

fetch_genes() is an EXPLORATORY tool and makes a number of simplicifications:

- the DMR regions file saves one CpG probe name and location, even though clusters of probes may map to that nearby area.
- it measures the distance from the start position of the one representative probe per region to any nearby genes, using the 'tol'erance parameter as the cutoff. Tolerance is the max number of base pairs of separation between the probe sequence start and the gene sequence start for it to be considered as a match.
- The default 'tol'erance is 250, but that is arbitrary. Increase it to expand the search area, or decrease it to be more conservative. Remember that Illumina CpG probe sequences are 50 base pairs long, so 100 is nearly overlapping. 300 or 500 would also be reasonable.
- "Adjacent" in the linear sequence may not necessarily mean that the CpG island is FUNCTIONALLY coupled to the regulatory or coding region of the nearby protein. DNA superstructure can position regulatory elements near to a coding region that are far upstream or downstream from the mapped position, and there is no easy way to identify "adjacent" in this sense.

- Changing the ‘tol‘erance, or the reference database will result major differences in the output, and thus one’s interpretation of the same data.
- Before interpreting these “associations” you should also consider filtering candidate genes by specific cell types where they are expressed. You should know the tissue from which your samples originated. And filter candidate genes to exclude those that are only expressed in your tissue during development, if your samples are from adults, and vice versa.

Arguments:

dmr_regions_file: pass in the output file DataFrame or FILEPATH from DMR function. Omit if you specify the *sql* kwarg instead.

ref: default is *refGene* use one of possible_tables for lookup: - ‘refGene’ – 88,819 genes – default table used in comb-b and cruzdb packages. - ‘knownGene’ – 232,184 genes – pseudo genes too (the “WHere TranscriptType == ‘coding_protein’” clause would work, but these fields are missing from the data returned.) - ‘ncbiRefSeq’ – 173,733 genes – this table won’t have gene descriptions, because it cannot be joined with the ‘kgXref’ (no shared key). Additionally, ‘gtexGeneV8’ is used for tissue-expression levels. Pseudogenes are ommited using the “WHERE score > 0” clause in the SQL.

tol: default 250 +/- this many base pairs consistutes a gene “related” to a CpG region provided.

tissue: str if specified, adds additional columns to output with the expression levels for identified genes in any/all tissue(s) that match the keyword. (e.g. if your methylation samples are whole blood, specify *tissue=blood*) For all 54 tissues, use *tissue=all*

genome_build: (None, NEW, OLD) Only the default human genome build, hg38, is currently supported. Even though many other builds are available in the UCSC database, most tables do not join together in the same way.

use_cached: If True, the first time it downloads a dataset from UCSC Genome Browser, it will save to disk and use that local copy thereafter. To force it to use the online copy, set to False.

no_sync: methylize ships with a copy of the relevant UCSC gene browser tables, and will auto-update these every month. If you want to run this function without accessing this database, you can avoid updating using the *no_sync=True* kwarg.

host, user, password, db: Internal database connections for UCSC server. You would only need to mess with these of the server domain changes from the current hardcoded value {HOST}. Necessary for tables to be updated and for *tissue* annotation.

sql: a DEBUG mode that bypasses the function and directly queries the database for any information the user wants. Be sure to specify the complete SQL statement, including the ref-table (e.g. refGene or ncbiRefSeq).

Note: This method flushes cache periodically. After 30 days, it deletes cached reference gene tables and re-downloads.

Manhattan Plot

In EWAS Manhattan plots, epigenomic probe locations are displayed along the X-axis, with the negative logarithm of the association P-value for each single nucleotide polymorphism (SNP) displayed on the Y-axis, meaning that each dot on the Manhattan plot signifies a SNP. Because the strongest associations have the smallest P-values (e.g., 1015), their negative logarithms will be the greatest (e.g., 15).

GWAS vs EWAS

- genomic coordinates along chromosomes vs epigenetic probe locations along chromosomes
- p-values are for the probe value associations, using linear or logistic regression,

between phenotype A and B.

Ref

Hints of hidden heritability in GWAS. Nature 2010. (https://www.ncbi.nlm.nih.gov/pubmed/20581876)

Required Inputs

stats_results: a pandas DataFrame containing the stats_results from the linear/logistic regression run on m_values or beta_values and a pair of sample phenotypes. The DataFrame must contain A "PValue" column. the default output of diff_meth_pos() will work.

array_type: specify the type of array [450k, epic, epic+, mouse, 27k], so that probes can be mapped to chromosomes.

output kwargs

save: specify that it export an image in *png* format. By default, the function only displays a plot.

filename: specify an export filename. The default is *f"manhattan_<stats>_<timestamp>.png"*.

visualization kwargs

- *verbose* (True/False) - default is True, verbose messages, if omitted.
- *fwer* (default 0.1) familywise error rate (fwer) is used to set p-value threshold and the FDR threshold line.
- *genome_build* – NEW or OLD. Default is NEWest genome_build.
- ***label-prefix* – how to refer to chromosomes. By default, it shows numbers like 1 … 22, and X, Y.** pass in 'CHR-' to add a prefix to plot labels, or rename with 'c' like: c01 … c22.

There are some preset "override" options for threshold lines on plots:

- ***explore*: (default False) include all FOUR significance threshold lines on plot** (suggestive, significant, bonferroni, and false discovery rate). Useful for data exploration.
- *no_thresholds*: (default False) set to True to hide all FOUR threshold lines from plot.
- *plain*: (default False) hide all lines AND don't label significant probes.
- *statsmode*: (default False) show the FDR and Bonferroni thresholds, hide the suggestive and genomic significant lines.

These allow you to toggle lines/labels on or off:

- *fdr*: (default False) draw a threshold line on plot corresponding to the adjusted false discovery rate = 0.05
- *bonferroni*: (default False) draw the Bonferroni threshold, correcting for multiple comparisons.

- ***suggestive*: (default 1e-5) draw the consensus "suggestive significance" theshold at p < 1e-5,** or set to False to hide.
- ***significant*: (default 5e-8) draw the consensus "genomic significance" theshold at p < 5e-8,** or set to False to hide.
- *plot_cutoff_label* (default True) label to each dotted line on the plot
- *label_sig_probes* (default True) labels significant probes showing the greatest difference between groups.

Chart options:

- *ymax* – default: 50. Set to avoid plotting extremely high -10log(p) values.
- *width* – figure width – default is 16
- *height* – figure height – default is 8
- *fontsize* – figure font size – default 16
- *border* – plot border – default is OFF
- *palette* – specify one of a dozen options for colors of chromosome regions on plot: ['default', 'Gray', 'Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20b', 'tab20c', 'Gray2', 'Gray3']

Volcano Plot

This function writes the pandas DataFrame output of diff_meth_pos() to a CSV file named by the user. The DataFrame has a row for every successfully tested probe and columns with different regression statistics as follows:

- regression coefficient
- lower limit of the coefficient's 95% confidence interval
- upper limit of the coefficient's 95% confidence interval
- standard error
- p-value
- FDR q-value (p-values corrected for multiple testing using the Benjamini-Hochberg FDR method)

Inputs and Parameters:

stats_results (required): A pandas DataFrame output by the function diff_meth_pos().

'alpha': Default: 0.05, The significance level that will be used to highlight the most significant p-values (or adjusted FDR Q-values) on the plot.

'cutoff': the beta-coefficient cutoff | Default: None format: a list or tuple with two numbers for (min, max) or 'auto'. If specified in kwargs, will exclude values within this range of regression coefficients OR fold-change range from being "significant" and put dotted vertical lines on chart. 'auto' will select a beta coefficient range that excludes 95% of results from appearing significant.

'adjust': (default False) – if this will adjust the p-value cutoff line for false discovery rate (Benjamini-Hochberg). Use 'fwer' to set the target rate.

'fwer': family-wise error rate (default is 0.1) – specify a probability [0 to 1.0] for false discovery rate

'data_type_label': What to put on X-axis. Either 'Fold Change' (default) or 'Regression Coefficient'.

visualization kwargs:

- ***palette* – color pattern for plot – default is [blue, red, grey]** other palettes: ['default', 'Gray', 'Pastel1', 'Pastel2', 'Paired', 'Accent', 'Dark2', 'Set1', 'Set2', 'Set3', 'tab10', 'tab20', 'tab20b', 'tab20c', 'Gray2', 'Gray3']
- *width* – figure width – default is 16
- *height* – figure height – default is 8
- *fontsize* – figure font size – default 16
- *dotsize* – figure dot size on chart – default 30
- *border* – plot border – default is OFF
- *data_type_label* – (e.g. Beta Values, M Values) – default is 'Beta'
- *plot_cutoff_label* – add label to dotted line on plot – default False

save: specify that it export an image in *png* format. By default, the function only displays a plot.

filename: specify an export filename. default is *volcano_<current_date>.png*.

Returns:

Displays a plot, but does not directly return an object. The data is color coded and displayed as follows:

- the negative log of adjusted p-values is plotted on the y-axis
- the regression coefficient beta value is plotted on the x-axis
- the significance cutoff level appears as a horizontal gray dashed line
- non-significant points appear in light gray
- significant points with positive correlations (hypermethylated probes) appear in red
- significant points with negative correlations (hypomethylated probes) appear in blue

To BED

Converts & exports manifest and probe p-value dataframe to BED format. * https://en.wikipedia.org/wiki/BED_(file_format)

- BED format: [chromosome number | start position | end position | p-values]

Where p-values are the output from diff_meth_pos() comparing probes across two or more groups of samples for genomic differences in methylation.

This output is required for combined-pvalues library to read and annotate manhattan plots with the nearest Gene(s) for each significant CpG cluster.

manifest_or_array_type: either pass in a Manifest instance from methylprep, or a string that defines which manifest to load. One of {'27k', '450k', 'epic', 'epic+', 'mouse'}.

genome_build: pass in 'OLD' to use the older genome build for each respective manifest array type.

note: if manifest has probes that aren't mapped to genome, they are omitted in BED file.

TODO: incorporate STRAND and OLD_STRAND in calculations.

returns a BED formatted dataframe if save is False, or the saved filename if save is True.

1.8.6 Release History

v1.1.1

- Minor edits to readme and removing methylcheck import, because it is not used anywhere.
- Note: methylprep is only imported for reading Manifest files and handling ArrayType.

v1.1.0

- We found that `diff_meth_pos` results were not accurate in prior versions and have fixed the regression optimization.
- `diff_meth_pos` function kwargs changed to provide more flexibility in how the model is optimized.
  - Added support for COVARIATES in logistic regression. Provide a dataframe with both the phenotype and covariates, and specify which columns are phenotype or covariates. It will rearrange and normalize to ensure the model works best.
  - Use the new 'solver' kwarg in `diff_meth_pos` to specify which form of linear or logistic regression to run. There are two flavors of each, and both give nearly identical results.
  - Auto-detects logistic or linear based on input: if non-numeric inputs in phenotype of exactly two values, it assumes logistic.
- Upgraded manhattan and volcano plots with many more options. Default settings should mirror most R EWAS packages now, with a "suggestive" and "significant" threshold line on manhattan plots.
- Unit test coverage added.

v1.0.1

- Fixed option to use Differentially methylated regions (DMR) via cached local copy of UCSC database (via fetch_genes) without using the internet. Previously, it would still contact the internet database even if user told it not to.
- Added testing via github actions, and increased speed
- updated documentation

v1.0.0

- fixed bug in fetch_genes() from UCSC browser; function will now accept either the filepath or the DMR dataframe output.

v0.9.9

- Added a differentially methylated regions (DMR) functions that takes the output of the `diff_meth_pos` (DMP) function.
  - DMP maps differences to chromosomes; DMR maps differences to specific genomic locii, and requires more processing.
  - upgraded methylprep manifests to support both `old` and `new` genomic build mappings for all array types. In general, you can supply a keyword argument (`genome_build='OLD'`) to change from the new build back to the old one.

  - Illumina 27k arrays are still not supported, but mouse, epic, epic+, and 450k ARE supported. (Genome annotation won't work with `mouse` array, only human builds.)
  - DMP integrates the `combined-pvalues` package (https://pubmed.ncbi.nlm.nih.gov/22954632/)
  - DMP integrates with UCSC Genome (refGene) and annotates the genes near CpG regions.
  - Annotation includes column(s) showing the tissue specific expression levels of relevant genes (e.g. `filter=blood`) this function is also available with extended options as `methylize.filter_genes()`
  - provides output BED and CSV files for each export into other genomic analysis tools
- `methylize.to_BED` will convert the diff_meth_pos() stats output into a standard BED file (a tab separated CSV format with standardized, ordered column names)

v0.9.8

- fixed methylize diff_meth_pos linear regression. upgraded features too
  - Fixed bug in diff_meth_pos using linear regression - was not calculating p-values correctly. Switched from statsmodels OLS to scipy linregress to fix, but you can use either one with kwargs. They appear to give exactly the same results now after testing.
  - The "CHR-" prefix is omitted from manhattan plots by default now
  - dotted manhattan sig line is Bonferoni corrected (pass in post_test=None to leave uncorrected)
  - added a probe_corr_plot() undocumented function, a scatterplot of probe confidence intervals vs pvalue
  - sorts probes by MAPINFO (chromosome location) instead of FDR_QValue on manhattan plots now
- Support for including/excluding sex chromosomes from DMP (probe2chr map)

v0.9.5

- Added imputation to diff_meth_pos() function, because methylprep output contains missing values by default and cannot be used in this function.
  - This can be disabled, and it will throw a warning if NaNs present.
  - Default is to delete probes that have any missing values before running analysis.
  - if 'auto': If there are less than 30 samples, it will delete the missing rows.
  - if 'auto': If there are >= 30 samples in the batch analyzed, it will replace NaNs with the average value for that probe across all samples.
  - User may override the default using: True ('auto'), 'delete', 'average', and False (disable)
  - diff_meth_pos() now support mouse array, with multiple copies of the same probe names.

v0.9.4

- Fixed bug where methylize could not find a data file in path, causing ImportError
- Improved diff_meth_pos() function and added support for all array types. Now user must specify the array_type when calling the function, as the input data are stats, not probe betas, so it cannot infer the array type from this information.

CHAPTER 2

Indices and tables

- genindex

Python Module Index

m

Index